К.К. Морозов, В.Г. Одиноков, В.М. Курейчик

АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ КОНСТРУКЦИЙ РАДИОЭЛЕКТРОННОЙ АППАРАТУРЫ

К.К. Морозов, В.Г. Одиноков, В.М. Курейчик

К. К. МОРОЗОВ, В. Г. ОДИНОКОВ, В. М. КУРЕЙЧИК

АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ КОНСТРУКЦИЙ РАДИОЭЛЕКТРОННОЙ АППАРАТУРЫ

Допущено
Министерством высшего и среднего
специального образования СССР
в качестве учебного пособия
для студентов вузов,
обучающихся по специальности
«Конструирование и производство радиоаппаратуры»

МОСКВА «РАДИО И СВЯЗЬ» 1983

ББК 32.844
М80
УДК 621.396.6

Морозов К. К., Одиноков В. Г., Курейчик В. М.

М80 Автоматизированное проектирование конструкций радиоэлектронной аппаратуры: Учеб. пособие для вузов. — М.: Радио и связь, 1983. — 280 с., ил.

В пер.: 1 р.

В книге рассмотрены математические методы компоновки, размещения и трассировки радиоэлектронной аппаратуры реализованных на печатных платах и интегральных микросхемах повышенной степени интеграции. Даны особенности подготовки и выдачи конструкторской и технологической документации с использованием ЭВМ. Описаны современные комплексные системы автоматизации конструкторского проектирования печатных плат и интегральных микросхем.

Для студентов специальности «Конструирование и производство радиоаппаратуры».

$\frac{2402020000-057}{046(01)-83}$ 43—83 ББК 32.844

6Ф2.1

РЕЦЕНЗЕНТЫ: КАФЕДРА КОНСТРУИРОВАНИЯ И ТЕХНОЛОГИИ ПРОИЗВОДСТВА РЭА МОСКОВСКОГО АВИАЦИОННОГО ИНСТИТУТА, ДОКТОР ТЕХН. НАУК И. П. НОРЕНКОВ

Редакция литературы по конструированию и технологии производства радиоэлектронной аппаратуры

Константин Константинович Морозов,
Валентин Григорьевич Одиноков,
Виктор Михайлович Курейчик

АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ КОНСТРУКЦИЙ РАДИОЭЛЕКТРОННОЙ АППАРАТУРЫ

Редактор Н. Н. Кузнецова
Художник Н. М. Ковалева
Художественный редактор Г. Н. Кованов
Технический редактор Г. И. Колосова
Корректор Т. В. Покатова

ИБ № 524

Сдано в набор 1.11.82 г. Подписано в печать 3.03.83 г.
Т-05166 Формат 60×90/16 Бумага кн-журн. Гарнитура литературная
Печать высокая Усл. печ. л. 17,5 Усл. кр.-отт. 17,5 Уч.-изд. л. 19,0
Тираж 14 000 экз. Изд. № 19500 Зак. № 144 Цена 1 р.
Издательство «Радио и связь». 101000 Москва, Главпочтамт, а/я 693

Типография издательства «Радио и связь» Госкомиздата СССР
101000 Москва, ул. Кирова, д. 40

ПРЕДИСЛОВИЕ

Научно-технический прогресс в области создания новых средств радиоэлектроники и вычислительной техники стал во многом зависеть от успешного решения проблемы автоматизации проектирования — важной народнохозяйственной задачи. Уровень сложности современной аппаратуры радиоэлектронной и вычислительной техники приблизился к границе, за которой эффективность труда человека-проектировщика резко падает, а число ошибок возрастает. Это особенно наглядно на этапе создания рабочего проекта устройства, когда конструктору приходится выполнять значительный объем нетворческой работы. Автоматизация конструирования — это не только способ повышения производительности труда конструктора, но и надежный способ снижения стоимости и повышения качества конструкторской документации. Проблемами автоматизации конструирования в нашей стране занимаются более 20 лет, однако и сегодня эта проблема актуальна.

Значительный вклад в развитие теории и практики автоматизации конструирования внесли советские ученые В. М. Глушков, Н. Я. Матюхин, Л. Б. Абрайтис, В. И. Анисимов, Р. П. Базилевич, Б. В. Баталов, Б. Ф. Высоцкий, Ю. Х. Вермишев, Б. Н. Деньдобренко, А. В. Каляев, А. М. Карапетян, С. А. Майоров, А. Н. Мелихов, В. Н. Лошаков, И. П. Норенков, Г. В. Орловский, А. И. Петренко, М. И. Песков, Г. Г. Рябов, Л. П. Рябов, К. А. Сапожков, В. А. Селютин, М. А. Штейн, О. Н. Юрин.

В предлагаемом учебном пособии авторы предприняли попытку восполнить пробел в учебной литературе по автоматизации конструирования. Содержание книги в наибольшей степени соответствует программе курса «Автоматизация конструирования» для высших учебных заведений по специальностям «Конструирование и производство радиоаппаратуры», «Конструирование и производство электронно-вычислительной аппаратуры» и «Конструирование и производство технических средств САПР». В книге авторы обобщили свой опыт по чтению лекций в течение ряда последних лет в Московском институте радиотехники, электроники и автоматики, Московском энергетическом институте, Таганрогском радиотехническом институте.

В книге кроме оригинальных и апробированных материалов авторов приводятся научные и учебно-методические материалы современных исследований советских и зарубежных ученых.

Основное внимание в учебном пособии авторы уделили современным методам и перспективам автоматизации конструирования аппаратуры на интегральных микросхемах 4-й и 5-й степеней интеграции и печатных платах со сверхплотной упаковкой с использованием математических методов теории графов и гиперграфов. Учебное пособие дополняет учебник Б. Н. Деньдобренко, А. С. Малика «Автоматизация конструирования РЭА» (М.: Высшая школа, 1980).

Глава 1, § 3.2 и гл. 5 написаны К. К. Морозовым, § 3.1, 3.3 и 3.4, гл. 6 — В. Г. Одиноковым, гл. 2, 4, введение, заключение — В. М. Курейчиком.

Авторы благодарны профессорам А. В. Каляеву, В. Б. Пестрякову, А. Н. Мелихову, кандидатам техн. наук В. А. Калашникову, В. В. Лисяку, Б. К. Лебедеву, а также А. Л. Перельману за помощь и полезные советы при подготовке рукописи к изданию. Улучшению содержания книги способствовали обстоятельные замечания рецензентов профессора И. П. Норенкова и сотрудников кафедры конструирования МАИ. Авторы приносят им глубокую благодарность.

Авторы

ВВЕДЕНИЕ

Современные достижения атомной энергетики, точного приборостроения, промышленных средств связи, медицинской техники и других наук невозможны в настоящее время без широкого использования электронно-вычислительной аппаратуры (ЭВА) и радиоэлектронной аппаратуры (РЭА). Далее будем использовать термин электронная аппаратура (ЭА).

Электронная аппаратура — это сложные комплексы устройств, предназначенные для электронной обработки информации, т. е. для хранения, преобразования и отображения ее в форму, удобную для восприятия человеком в соответствии с заданной программой.

В настоящее время во всем мире наблюдаются резкое увеличение производства ЭА и повышение ее возможностей. Особенно это связано с последними успехами в области микроэлектроники.

Разработка и внедрение ЭА является одними из основных показателей современной нучно-технической революции. Прогресс в области создания ЭА определяется повышением надежности, экономичности, качества и эффективности устройств, совершенствованием схем, конструкций, технологии.

Процесс создания ЭА условно разделяется на три основных этапа: схемотехническое проектирование, конструкторское проектирование, технологическое проектирование. На первом этапе разрабатывается архитектура будущей ЭА. Материализация же основных идей ЭА осуществляется на стадии конструирования и технологии производства. Практика показала, что именно здесь обеспечиваются возможность воплощения электронных схем в микроэлектронные конструкции, рождение жизнеспособных изделий, отвечающих современным требованиям науки, техники и производства.

В процессе создания ЭА тесно переплетаются вопросы разработки математической логики, конструкции и технологии. Даже небольшие изменения в логике ЭА без учета конструкторско-технологических факторов приводят к ухудшению ее основных характеристик.

Расширение функциональных возможностей и усложнение ЭА поставили ученых и инженеров перед необходимостью поиска новых принципов конструирования и технологии, коренного изменения методики конструирования на основе использования современных средств вычислительной техники. Электронная аппаратура четвертого и пятого поколений содается сейчас на базе интегральных микросхем (ИМС) 4-й и 5-й степеней интеграции.

Сегодня конструкторы и технологи ЭА участвуют в разработках ИМС, на кристалле которых размером 3×4 мм и менее размещаются сотни тысяч элементов, реализующих функции транзисторов, резисторов и др. На основе прогнозирования науки ожидается, что следующие поколения ЭА будут реализованы на гигантских ИМС с голографической и оптической памятью, будут иметь программируемую перестраиваемую архитектуру, электронную коммутацию и будут способны принимать и выдавать решения в расплывчатых условиях.

Конструкторы ЭА четвертого и пятого поколений приступили к разработкам методологии построения поисковых алгоритмов, все более приближаясь к созданию искусственного интеллекта. Недалек тот день, когда каждая ЭВМ будет иметь специальные «интеллектуальные процессоры», способные играть в шахматы на уровне гроссмейстера, управлять сложным производством, разыгрывать модели военных сражений, следить за недоступными для человека технологическими операциями и вообще являться помощником, подчеркиваем помощником, человека при принятии решений в неопределенных условиях. Многие конструкторские бюро приступили к разработке промышленных интеллектуальных роботов. Не является фантастической посылка таких роботов в космос для выполнения недоступных человеку работ. Конструкторы недалекого будущего приступят к разработке многопроцессорных систем искусственного интеллекта и вплотную подойдут к построению эффективных моделей человеческого мозга.

Конструирование сейчас и в будущем — это непрерывный творческий процесс, основанный на диалоге человека с вычислительной машиной. Уже сейчас кульманы в конструкторских бюро, на заводах и в научно-исследовательских институтах меняют на автоматизированные рабочие места, оборудованные периферийными устройствами, в частности электронными дисплеями, позволяющими нетворческие, рутинные операции выполнять на ЭВМ, а работу, требующую мышления, взлета фантазии, оставлять конструктору.

Всего лишь 15—20 лет назад появился новый термин «*автоматизация проектирования и конструирования*», обозначающий новое бурно развивающееся направление науки и техники. Методы автоматизированного и автоматического конструирования и проектирования обладают огромными преимуществами по сравнению с методами неавтоматизированного проектирования. Сокращаются сроки проектирования ЭА, снижается стоимость, повышается надежность устройств.

Конструктор-технолог ЭА, вооруженный современной аппаратурой, знаниями в области микроэлектроники, электронной техники и автоматизации конструирования, в настоящее время — это высококвалифицированный специалист, способный сочетать творческую фантазию с возможностями современных вычислительных машин.

1. ОБЩИЕ ВОПРОСЫ АВТОМАТИЗАЦИИ ПРОЕКТИРОВАНИЯ И КОНСТРУИРОВАНИЯ

1.1. ЭТАПЫ АВТОМАТИЗАЦИИ ПРОЕКТИРОВАНИЯ И КОНСТРУИРОВАНИЯ

В общем случае процесс автоматизации проектирования схем ЭА, как и любых дискретных устройств, состоит из трех основных этапов: системотехнического, схемотехнического и конструкторского.

Первый этап включает в себя системное и структурное проектирование. Схемотехническое проектирование состоит из моделирования, логического проектирования, а также контроля и построения диагностических тестов. Конструкторский этап включает техническое и технологическое проектирование.

Одна из возможных укрупненных структурных схем процесса автоматизированного проектирования ЭА показана на рис. 1.1.

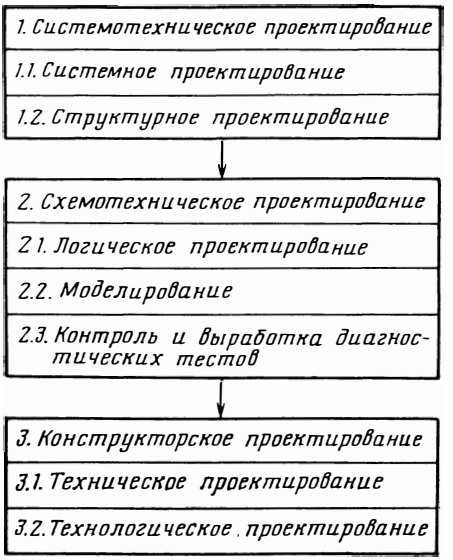


Рис. 1.1. Структурная схема процесса автоматизированного проектирования ЭА

При системном проектировании используются идеи и методы системного анализа. На основе многочисленных факторов проводится всесторонний анализ технического задания на разработку ЭА и принимается решение относительно методики построения и путей реализации вычислительного процесса.

При структурном проектировании разрабатываются общая структурная схема ЭА и алгоритмы выполнения отдельных операций. Для выбора структуры необходимо учитывать требования технологичности, надежности, возможности более широкого использования однородных и квазиоднородных унифицированных узлов.

Системотехнический этап проектирования в основном пока является неформализованным процессом. Здесь используются твор-

ческие возможности инженера. Электронная вычислительная машина просматривает варианты решений, принимаемых разработчиком, и выбирает из них оптимальный. На этом этапе используются специальные языки, формальные методы генерации вариантов вычислительного процесса по исходному заданию методом автоматического получения структурных схем.

На этапе схемотехнического проектирования широко используются логические и вычислительные возможности ЭВМ. Целью логического проектирования ЭА является автоматический или автоматизированный формализованный абстрактный и структурный синтез узлов, выбранных в результате структурного проектирования, при котором проверяется эквивалентность исходного задания конечному результату. В теоретическом плане здесь имеются существенные достижения: автоматически синтезируются управляющие и специального вида операционные устройства. На практике при автоматизации логического проектирования схем требуется решение большого числа задач. К ним относятся: разработка эффективных языков описания исходных заданий и языков структурного проектирования, алгоритмов построения формальных моделей устройств и др. При логическом проектировании важнейшими критериями оптимизации являются: минимизация числа типов логических узлов, достижение максимальной однотипности логических блоков, возможность эффективного моделирования и диагностирования схем, максимальный учет требований конструкторского и технологического проектирования. Задачей моделирования являются построение карты состояний для логических сигналов, проверка временных соотношений при прохождении входных сигналов, анализ функциональных схем на соответствие заданной системе булевых функций. Различают физическое и математическое моделирование. Для схем ЭА более важным является математическое моделирование, так как использование сложных интегральных микросхем в основном исключает возможность физического моделирования.

Отметим, что согласно ГОСТ 17021—75 в книге под *интегральной микросхемой* понимается микроэлектронное изделие, выполняющее определенную функцию преобразования и обработки сигнала и имеющее высокую плотность упаковки электрически соединенных элементов и кристаллов, которое с точки зрения требований к испытаниям, приемке, поставке и эксплуатации рассматривается как единое целое.

Интегральная микросхема 1-й степени интеграции (ИМС1) — это микросхема, содержащая до 10 элементов и компонентов включительно.

Интегральная микросхема 2-й степени интеграции (ИМС2) — это микросхема, содержащая свыше 10 до 100 элементов и компонентов включительно.

Интегральная микросхема 3-й степени интеграции (ИМС3) — это микросхема, содержащая свыше 100 до 1000 элементов и компонентов включительно.

Интегральная микросхема 4-й степени интеграции (ИМС4) — это микросхема, содержащая свыше 1000 до 10 000 элементов и компонентов включительно.

Интегральная микросхема 5-й степени интеграции (ИМС5) — это микросхема, содержащая свыше 10 000 до 100 000 элементов и компонентов включительно.

Интегральная микросхема 6-й степени интеграции (ИМС6) — это микросхема, содержащая свыше 100 000 до 1 000 000 элементов и компонентов включительно.

Большая интегральная микросхема — это интегральная микросхема, содержащая 500 и более элементов, изготовленных по биполярной технологии, 1000 и более элементов, изготовленных по МДП-технологии.

Микропроцессор — это программно-управляемое устройство, осуществляющее процесс обработки цифровой информации и управление им, построенное на основе одной или нескольких больших интегральных микросхем.

Элемент интегральной микросхемы — это часть ИМС, не выделяемая как самостоятельное изделие, выполняющее функцию какого-либо электрорадиоэлемента (транзистора, диода и т. п.).

Компонент интегральной схемы — это часть ИМС, реализующая функции какого-либо электрорадиоэлемента, которая может быть выделена как самостоятельное изделие.

Развитием подэтапа моделирования являются контроль и диагностика. При этом определяется методика построения схем аппаратного контроля, разрабатываются системы тестового обслуживания, определяются необходимые степень и уровень резервирования для выбора минимальной ремонтируемой единицы. Это связано с увеличением надежности используемых элементов и укрупнением типовых элементов замены в устройствах.

Функциональные схемы, полученные в результате логического синтеза и моделирования, служат входной информацией для конструкторского (технического, монтажно-коммутационного, физического) проектирования. При этом необходимо решать следующие основные задачи: покрытие функциональной схемы ячейками из заданного набора, т. е. переход к принципиальной электрической схеме устройства; компоновка элементов схемы в типовые элементы замены (ТЭЗ) — ремонтопригодные конструктивные единицы, панели, блоки, стойки и т. д.; размещение элементов в конструктивных единицах по различным критериям; распределение цепей по слоям — многослойная или двухслойная трассировка и контроль правильности полученной топологии.

Цель технологического проектирования состоит в автоматизированной выдаче технологических документов, разработке алгоритмов управления координатографами и другими периферийными устройствами и методов автоматического получения фотошаблонов, служащих руководящими материалами в системе производства.

Одной из важнейших задач проблемы автоматизации проектирования, конструирования и изготовления схем является автоматизация конструкторского проектирования.

В книге обсуждаются вопросы разработки и исследования методов, алгоритмов и систем автоматизации проектирования с использованием методов современной математики. Основу проектирования составляют математическое описание задач проектирования на заданном формальном языке, разработка основных теорем и алгоритмов, структуры системы, запись программ на алгоритмическом языке и решение их на универсальной или специализированной ЭВМ с дальнейшим выходом на автоматизированные рабочие места (АРМ) и другое оборудование.

Для большинства задач проектирования формальное разбиение процесса поиска часто затруднительно. Если задачи проектирования сформулировать в теоретико-множественном плане, то обычно приходится встречаться с вопросами, которые могут быть решены, только если перебрать большое число вариантов. Поэтому актуальным является вопрос нахождения экономичных способов сокращения перебора. При этом существенную роль играет вопрос о формальном описании тех или иных неформально поставленных задач, методах их расчленения на отдельные шаги, а также об организации оптимальных в том или ином смысле процедур поиска вариантов проектирования.

Использование аппарата теории графов в настоящее время для решения задач автоматизации проектирования и конструирования самых различных объектов находит все более широкое применение. Объясняется это тем, что язык теории графов во многих случаях адекватен в той или иной мере объектам проектирования, описывает их естественным образом и в то же время позволяет абстрагироваться от конкретных объектов и иметь дело с абстрактными моделями. Это в свою очередь дает возможность строить математически обоснованные алгоритмы проектирования, находить простые и высококачественные решения, рационально и эффективно использовать ЭВМ. Следует отметить, что точное решение задач проектирования большой размерности связано с перебором большого числа вариантов, который затруднителен даже для ЭВМ. Поэтому в работе наравне с точными методами проектирования, основанными на методах исследования операций, рассматриваются алгоритмы направленного поиска, которые не дают оптимальных решений, но позволяют получать достаточные для практических целей результаты.

1.2. ЗАДАЧИ ПОСТРОЕНИЯ СИСТЕМ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ

Главными проблемами при создании систем автоматизированного проектирования (САПР) являются улучшение качества конструирования и создание средств, обеспечивающих решение принципиально новых задач, выдвигаемых техническим прогрессом.

В общем случае *системой автоматизированного проектирования* или конструкторского проектирования можно считать некоторый комплекс алгоритмов с диспетчером, реализованный в виде множества программ, объединенных в пакеты, библиотеки или модули, и автоматизированных рабочих мест, включающих необходимое для выпуска конструкторской документации оборудование. *Идеальная система автоматизированного проектирования* предполагает такой порядок работ, когда техническое задание, сформулированное конструктором, полностью обрабатывается с помощью ЭВМ. Система программ определяет порядок их следования и тем самым последовательность выполнения отдельных этапов. На выходе ЭВМ индуцируется модель топологии устройства в виде документации для системы автоматизированного управления технологическими процессами.

Однако даже самые современные ЭВМ не могут заменить конструктора, а лишь способны дополнить его, выполняя нетворческие, рутинные операции. Поэтому в настоящее время наибольшее распространение получили интерактивные системы «человек—машина», работающие в режиме диалога конструктора с ЭВМ. Они особенно эффективны при анализе и решении комбинаторно-логических задач этапа конструкторского проектирования схем. Интерактивные системы должны иметь такую организацию, при которой оптимальным образом сочетаются процессы автоматизированного проектирования с указаниями конструктора, творчески направляющего процесс разработки. Следующим, не менее важным фактором, влияющим на структуру системы, является определение области ее применения и выбор методологии конструирования. Такая постановка задачи связана с неэффективностью универсализации используемых в системе алгоритмов и программ с целью их применения к различным конструкциям. Поэтому целесообразно включение в систему программ-диспетчеров, с помощью которых производится управление остальными программами. Наличие диспетчера позволяет также решить следующие важные вопросы организации системы: возможность свободного «входа» в систему на всех этапах конструирования с целью корректировки промежуточных результатов, возможность использования как пакетов, так и единичных программ, организация наиболее рациональной последовательности этапов разработки.

Выполнение рассмотренных выше требований становится необходимым при поэтапной организации процесса конструирования. При этом работоспособность системы будет во многом зависеть от надежности и удобства стыковки отдельных этапов. Это достигается с помощью унификации входной и выходной информации, единства методов ее записи на носителях, распределения памяти ЭВМ и т. д. Значительное место при организации САПР отводится выбору алгоритмического языка, достаточно простого для описания входной, первичной информации и доступного конструктору. Отметим, что определяющим фактором создания САПР является обязательный количественный или качественный выигрыш от ав-

томатизации, существенно превосходящий те дополнительные затраты труда, которые она вызывает. Система должна обладать высокой жизнеспособностью, т. е. легкой настраиваемостью, возможностью изменения критериев оптимизации, способностью к расширению и дополнению библиотеки программ, стыковки с другими системами проектирования и процессами автоматизированного производства.

В настоящее время САПР развивается в двух направлениях: с одной стороны, широко используются мини- и микро-ЭВМ и микропроцессоры с непосредственным участием конструктора. С другой стороны, создаются системы автоматического проектирования на основе многопроцессорных вычислительных структур без участия человека. Считается, что последнее направление наиболее перспективное. В обоих направлениях определяющими остаются вопросы оптимизации алгоритмов, формализации задач конструирования, представления информации в ЭВМ, организации библиотек программ и др.

Система автоматизированного проектирования должна иметь возможности: автоматического хранения информации о проектируемом устройстве; последовательного расширения и совершенствования системы, активной связи «конструктор — система»; оперировать оптимальными взаимозаменяемыми алгоритмами конструирования, специализации систем на конструирование ЭА на ИМС любой степени интеграции; увеличения мощности системы применением многопроцессорных вычислительных структур и периферийных устройств; стыковки со специальными автоматами (координатографами, графопостроителями и т. д.); изготовления конструкторской и технологической документации.

Качество САПР характеризуется не только возможностью использования системы для проектирования широкого класса ЭА без существенных изменений, но и оптимальностью алгоритмов и способом представления информации.

Основным требованием к размещению информации в памяти ЭВМ является свободный доступ к данным, т. е. такая организация их хранения, при которой разработчик получит возможность на всех этапах конструирования быстро просматривать все имеющиеся параметры с целью выбора требуемых.

Не менее важна правильность построения *языка проектирования* (ЯП), т. е. языка, предназначенного для представления и преобразования описаний объектов при проектировании. Согласно ГОСТ 22487—77 различают: входной язык проектирования, предназначенный для представления задания на проектирование; базовый язык проектирования, предназначенный для представления дополнительных сведений к первичному описанию объекта проектирования, проектных решений, описаний проектных процедур и их последовательности; выходной язык проектирования, предназначенный для представления какого-либо проектного решения, включая результат проектирования в форме, удовлетворяющей требованиям его дальнейшего применения.

Правильность выбора алгоритмов является одним из факторов, определяющим экономическую эффективность использования САПР. Такая постановка вопроса требует проведения работ, направленных на дальнейшее совершенствование математического, информационного, технического, лингвистического, методического, организационного и программного обеспечений САПР.

Математическое обеспечение (МО) автоматизированного проектировани (МОАП) — это совокупность математических методов, моделей и алгоритмов проектирования, необходимых для его выполнения.

Техническое обеспечение (ТО) автоматизированного проектирования — это совокупность взаимосвязанных и взаимодействующих технических средств, предназначенных для его выполнения.

Программное обеспечение (ПО) автоматизированного проектирования (ПОАП) — это совокупность машинных программ, представленных в заданной форме. Соответственно пакет прикладных программ (ППП) — это совокупность представленных в заданной форме машинных программ, необходимых для выполнения проектной процедуры. Часть ПО АП, предназначенная для управления проектированием, называется операционной системой (ОС) автоматизированного проектирования.

Информационное обеспечение (ИО) автоматизированного проектирования — это совокупность представленных в заданной форме сведений, необходимых для выполнения АП.

Составной частью информационного обеспечения САПР являются автоматизированные банки данных (АБД), которые состоят из базы данных (БД) и системы управления базами данных (СУБД). Автоматизированные банки данных создаются как обслуживающие подсистемы САПР и предназначены для автоматизированного обеспечения необходимыми данными подсистемы САПР.

Управление АБД осуществляется специалистами, обеспечивающими целостность, правильность, эффективность использования и функциональные возможности.

К АБД предъявляются требования гибкости, надежности, наглядности и экономичности.

Лингвистическое обеспечение (ЛО) автоматизированного проектирования — это совокупность языков, представленных в заданной форме проектирования с терминами и определениями, правил формализации естественного языка и методы сжатия и развертывания текстов, необходимые для выполнения АП.

Методическое обеспечение (МТО) автоматизированного проектирования — это совокупность документов, устанавливающих состав и правила отбора и эксплуатации средств обеспечения проектирования.

Организационное обеспечение (ОО) автоматизированного проектирования — это совокупность документов, устанавливающих состав проектной организации и ее подразделений, связи между ними, их функции, а также форму представления результата про-

ектирования и порядок рассмотрения проектных документов, необходимых для выполнения автоматизированного проектирования.

Тогда под комплексом средств автоматизации проектирования понимается совокупность различных видов его обеспечения (рис. 1.2).

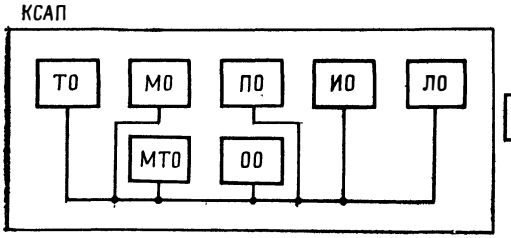


Рис. 1.2. Комплекс средств автоматизации проектирования

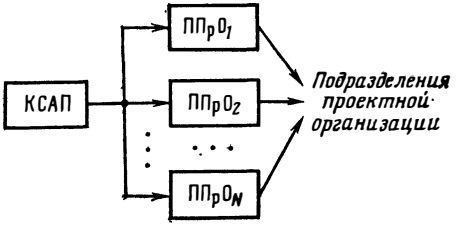


Рис. 1.3. Система автоматизированного (автоматического) проектирования

Теперь можно формально определить, что такое САПР (рис. 1.3).

Система автоматизированного проектирования — это комплекс средств автоматизации проектирования, взаимосвязанных с необходимыми подразделениями проектной организации или коллективом специалистов (пользователем системы).

Для достижения целей создания эффективных САПР ЭА необходимо осуществлять:

автоматизацию процесса поиска, обработки и выдачи информации;

совершенствование проектирования на основе применения математических методов и средств вычислительной техники;

использование методов оптимизационного и многовариантного проектирования;

создание единых банков данных, содержащих систематизированные сведения справочного характера;

повышение качества оформления проектной документации и доли творческого труда конструкторов за счет автоматизации нетворческих работ;

унификацию и стандартизацию методов проектирования;

взаимодействие с САПР различного уровня и функционального назначения.

Составными структурными частями САПР являются подсистемы, обладающие всеми свойствами систем и создаваемые как самостоятельные системы.

По назначению подсистемы САПР разделяются на два вида: проектирующие и обслуживающие. К проектирующим относятся подсистемы, выполняющие проектные процедуры и операции, а к обслуживающим относятся подсистемы, предназначенные для поддержания работоспособности проектирующих подсистем.

В зависимости от отношения к объекту проектирования различают два вида проектирующих подсистем: объектные и инвариантные. К объектным относятся подсистемы, выполняющие одну или несколько проектных процедур или операций, непосредственно зависимых от конкретного объекта проектирования. К инвариантным относятся подсистемы, выполняющие унифицированные проектные процедуры и операции.

Рассмотрим существующие согласно ГОСТ 23501.8—80 классификации САПР.

По типам объектов проектирования различают САПР:

изделий машиностроения и приборостроения;

технологических процессов в машиностроении и приборостроении;

объектов строительства;

организационных систем.

По сложности объектов проектирования различают САПР:

простых объектов, проектирующих объекты до 10^2 составных частей;

объектов средней сложности, проектирующих объекты свыше 10^2 до 10^3 составных частей;

сложных объектов — свыше 10^3 до 10^4 составных частей;

очень сложных объектов — свыше 10^4 до 10^6 составных частей;

объектов очень высокой сложности — свыше 10^6 составных частей.

По уровню автоматизации проектирования различают САПР:

низкоавтоматизированного проектирования, в которых количество автоматизированных проектных процедур (АПП) составляет до 25% общего количества проектных процедур;

среднеавтоматизированного проектирования, в которых количество АПП составляет свыше 25 до 50% общего количества проектных процедур;

высокоавтоматизированного проектирования, в которых количество АПП составляет свыше 50% общего количества проектных процедур.

По комплексности различают САПР:

одноэтапные;

многоэтапные;

комплексные.

По характеру выпускаемых проектных документов различают САПР:

текстовых документов;

текстовых и графических документов;

документов на машинных носителях (перфокартах, перфолентах, магнитных лентах, дисках, барабанах);

документов на фотоносителях;

документов на двух типах носителей данных;

документов на всех типах носителей данных.

По производительности различают САПР:

малой производительности. Выпускает до 10^5 проектных документов (ПД) за год (в пересчете на 11-й формат);

средней производительности. Выпускает свыше 10^5 до 10^6 ПД за год;

высокой производительности. Выпускает свыше 10^6 ПД за год.

По количеству уровней в структуре технического обеспечения различают САПР:

одноуровневые, построенные на основе ЭВМ среднего или высокого класса с набором периферийных устройств;

двухуровневые, построенные на основе ЭВМ среднего или высокого класса и одного или нескольких автоматизированных рабочих мест (АРМ) с мини-ЭВМ;

трехуровневые, построенные на основе ЭВМ высокого класса, одного или нескольких АРМ и периферийного программно-управляемого оборудования.

Интегрированная САПР — это система, имеющая альтернативное программное обеспечение и операционную систему автоматизированного проектирования, позволяющую выбирать совокупность машинных программ применительно к заданному объекту или классу объектов проектирования.

Основными критериями при выборе алгоритмического языка (АЯ) можно считать: затраты машинного времени на реализацию программы, записанной в символах АЯ; удобство стыковки отдельных программ; наличие в языке средств описания информации специального вида; наличие современного математического обеспечения для выбранного языка; популярность языка; наличие трансляторов выбранного языка для широкого класса ЭВМ.

Опыт создания САПР свидетельствует в пользу АЯ ФОРТРАН или подобных ему, как наиболее удовлетворяющих указанным

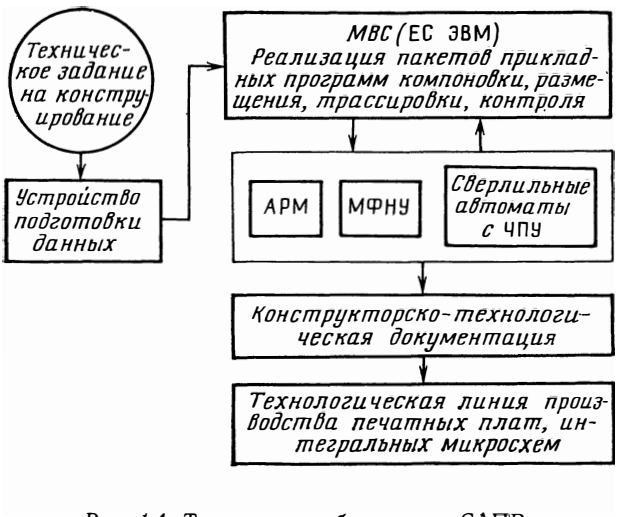


Рис. 1.4. Техническое обеспечение САПР

критериям. Очевидно, требование наличия в языке средств описания информации специального вида влечет к созданию специального входного и выходного языка, который должен связать отдельные этапы конструирования, согласовать расположение информации внутри массива.

Техническое обеспечение САПР представлено на рис. 1.4. Основой ТО является многопроцессорная вычислительная система (МВС) или, в частном случае, ЕС ЭВМ. Периферийное оборудование ТО САПР можно условно разделить на четыре группы (рис. 1.5):

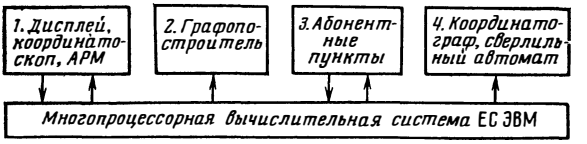


Рис. 1.5. Периферийное оборудование САПР

для получения и обработки эскизной документации и диалога с МВС или ЭВМ *1*;

для получения выходной документации *2*;

для внутренних передач при работе САПР или для работы по линиям связи *3*;

для изготовления фотошаблонов и технологическое *4*.

Программное обеспечение САПР включает в себя специальные вычислительные программы, используемые для получения необходимой инженеру конструкторской документации. Особенностью таких программ является наличие модульной структуры. Каждый программный модуль имеет заданное целевое назначение, прост для разработки, гибок и надежен. Управление и координация использования модулей обеспечиваются наличием программы диспетчера.

Информация, используемая при автоматизированном проектировании, располагается на нескольких уровнях и организуется в некоторую иерархическую систему. Эта система называется *архивом* или *банком данных*. Основная функция архива заключается в выдаче информации по запросу заказчика. Информация выдается в виде чертежей, графиков, таблиц и т. п.

Успешное применение программных модулей возможно только при наличии в ПО специальных программ, предусматривающих «гибкие» средства управления вводом, редактированием, хранением, пересылкой и выводом необходимой информации, а также средствами обработки пакетов прикладных программ, предназначенных для реализации отдельных этапов конструирования.

Упрощенная схема вычислительного процесса конструкторского проектирования показана на рис. 1.6. Одна из возможных схем распределения информации в САПР показана на рис. 1.7.

Взаимодействие конструктора с системой выполняется на *языке директив*. С помощью входного транслятора эти директивы преобразуются в форму внутреннего представления и записываются в банк данных. На первом этапе производится запись директив, опи-

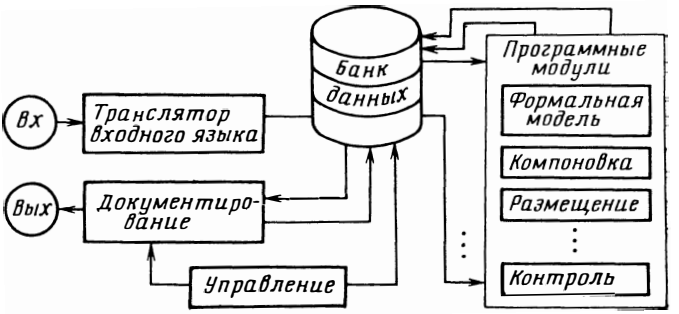


Рис. 1.6. Схема вычислительного процесса конструкторского проектирования

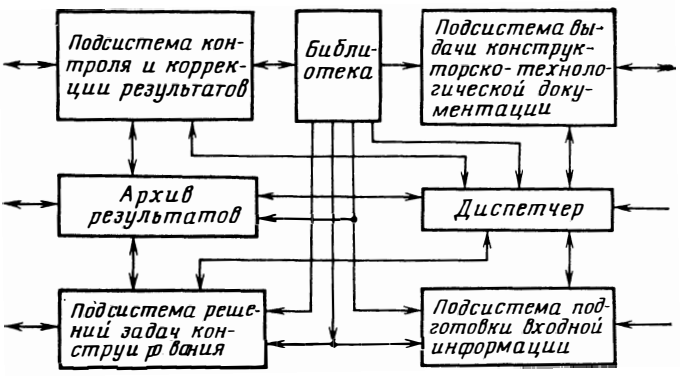


Рис. 1.7. Распределение информации в САПР

сывающих объекты конструирования, при этом в банке данных производится формирование библиотеки. Набор директив включает в себя задание на вызов из библиотеки всех элементов, входящих в объект конструирования, и задания на компиляцию необходимого разработчику набора программных модулей. Другими словами, задаются состав программных модулей и последовательность их реализации. Результаты всех промежуточных решений, а также окончательное решение хранятся в банке данных. С помощью специального программного модуля «Документирование» производится преобразование рабочих массивов и выходной информации из формы внутреннего представления в форму конструкторской и технологической документации. Программные модули каталогизируются в создаваемой библиотеке. Организация вы-

числительного процесса и связь конструктора с банком данных осуществляются с помощью управляющей программы. Директивами служат операторы управления заданиями управляющей программы, по которым модули вызываются из библиотеки на исполнение.

Информация о разрабатываемой ЭА, хранящаяся в банке данных, делится на три области. Первая область предназначается для хранения конструкторской информации, вторая — для хранения промежуточной информации, передаваемой от модуля к модулю, третья — для обеспечения режима прерывания. Здесь хранится вся информация, необходимая для продолжения конструирования с этапа, на котором произошло прерывание. Это позволяет производить параллельное конструирование нескольких устройств одновременно.

Формирование документации производится в соответствии с требованиями рисующих автоматов (координатографов, графопостроителей, дисплеев и т. д.). При этом используются данные из хранящихся в памяти ЭВМ таблиц, форматов и т. п. Кроме того, данные о результатах конструирования заносятся в архив системы.

При реализации задач конструкторского проектирования можно условно выделить следующие крупные блоки (рис. 1.8): ввод и контроль информации, построение формальной модели схемы, т. е. переход от схемы к графу, компоновка, размещение, трассировка, выдача конструкторской документации. Блок компоновки обычно включает в себя модули, обеспечивающие покрытие схемы заданным комплексом элементов, а также различные алгоритмы разбиения графа схемы на конструктивно законченные части. Блок размещения составляется из модулей точного, итерационного и последовательного размещения с минимизацией суммарной длины и внутрисхемных пересечений. Блок трассировки включает модули определения планарности, планарного разбиения схемы, построения кратчайших связывающих деревьев, распределения фрагментов цепей по магистралям, получения координат трасс.

Основные требования, предъявляемые к программным модулям в САПР,— это универсальность, адаптируемость, возможность параллельного решения нескольких задач конструирования, совместимость автоматического, автоматизированного и интерактивного режимов, развитие базы данных.

Каждый блок в САПР в соответствии с иерархическим принципом обработки информации образует подсистему. Исходная информация в САПР делится на три типа: описание схемы, описание конструкции и управляющая информация. С помощью управляющей информации задаются режимы работы программных модулей конструирования. Блок ввода и контроля осуществляет ввод информации, формирование информации, необходимой для работы последующих программных модулей, контроль и размещение сформированной информации в банке данных.

Важнейшим вопросом при автоматизации конструирования ЭА является выбор альтернативных вариантов. При автоматизации конструирования (АК) цель задана и ее записывают обычно в виде $f(\mathbf{x}) \Rightarrow \max$ (min), где f — некоторая скалярная функция, на-

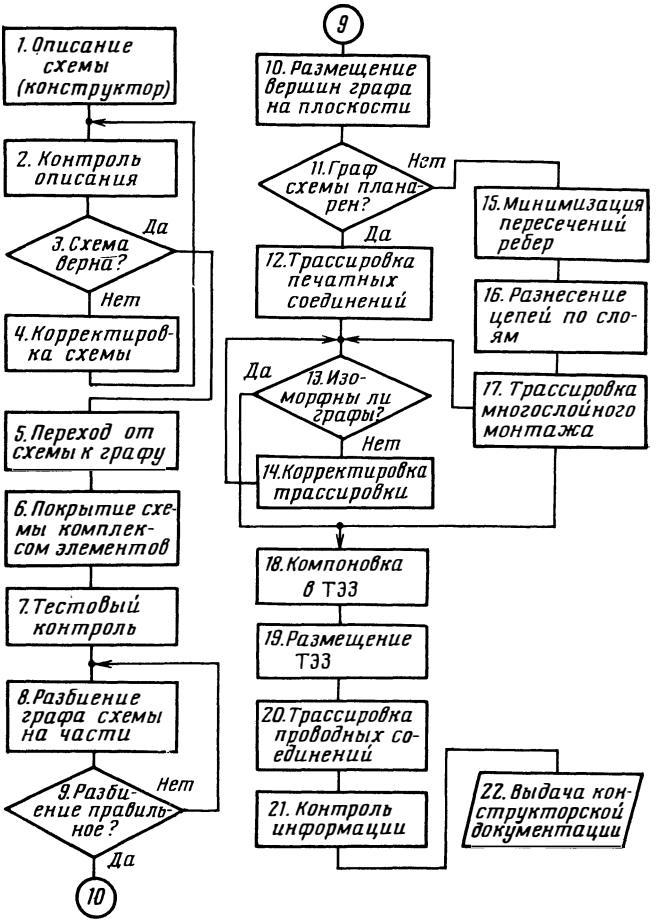


Рис. 1.8. Основные этапы автоматизированного конструкторского проектирования

пример надежность конструкции, диагностируемость ЭА и т. п.; $\mathbf{x}$ — вектор, определяющий управляемые (изменяемые) параметры, например число типовых элементов конструкции, причем $X = \{\mathbf{x}^i\}$, $\mathbf{0} \leqslant \mathbf{x}^i \leqslant \mathbf{x}$.

Задачи такого вида решаются путем нахождения экстремума функции $f(x)$ на некотором множестве X, т. е.

$$f(x) \underset{x \in X}{\Rightarrow} \max(\min).$$

При автоматизации конструирования перед конструктором стоит задача с помощью ЭВМ выбирать способ действия, т. е. век-

тор, дающий максимальное (минимальное) значение нескольким функционалам: $f_1(\mathbf{x})$, $f_2(\mathbf{x})$, ..., $f_v(\mathbf{x})$.

Для решения задач такого типа обычно используются методы линейной свертки, контрольных цифр, компромиссы Парето и др.

При использовании методов линейной свертки вместо v различных критериев учитывают один комплексный

$$F(\mathbf{x}) = \sum_{i=1}^{v} \lambda_i f_i(\mathbf{x}),$$

где λ_i — положительные числа, нормированные заданным образом. Коэффициенты λ_i отражают представление конструктора о содержании компромисса, который он должен принять.

Рассмотрим использование контрольных цифр. При автоматизации конструирования ЭА на ИМС4 и ИМС5 часто задают систему нормативов, имеющих вид f^*_1, f^*_2, ..., f^*_v. Они означают, что параметры ЭА должны быть такими, что $f_i(\mathbf{x}) \Rightarrow \max$ при $f_i(\mathbf{x}) \geqslant f^*_i$.

В этом случае целевую функцию представляют в виде $F(\mathbf{x}) = \min_i f_i(\mathbf{x})/f^*_i$, а затем производят поиск вектора $\mathbf{x}$, который дает возможность определить наибольшее значение $F(\mathbf{x})$. Условие $F(\mathbf{x}) \Rightarrow \max$ означает выбор такой системы значений x, которая максимизирует отношение i-го значения критерия к его контрольному значению. Отметим, что значения f^*_i обычно определяются в результате экспертного опроса или задаются конструктором из опыта работы.

Компромиссы Парето. При решении многокритериальных задач прибегают к сокращению подмножества заведомо «плохих» решений. Пусть сделан выбор $\mathbf{x}'$ и имеется другой выбор $\mathbf{x}''$ — такой, что для всех критериев справедливо $f_i(\mathbf{x}'') \geqslant f_i(\mathbf{x}')$. Тогда видно, что выбор $\mathbf{x}''$ предпочтительнее, чем выбор $\mathbf{x}'$. Следовательно, векторы $\mathbf{x}'$, удовлетворяющие неравенству, следует исключить из рассмотрения. Предлагается исследовать только те векторы $\mathbf{x}$, для которых не существует $\mathbf{x}''$ для всех критериев, удовлетворяющих приведенному неравенству. Множество таких векторов $\mathbf{x}$ называется множеством Парето.

Например, пусть цели автоматизации конструирования определяются двумя функциями: $L(G) \Rightarrow \min$ и $P(G) \Rightarrow \min$, где $L(G)$ — суммарная длина соединений в типовом элементе замены (ТЭЗ); $P(G)$ — количество пересечений соединений в ТЭЗ.

Тогда допустимому значению переменной $x \in G$ будет соответствовать одна точка на плоскости (L, P). Равенства $L = L(G)$ и $P = P(G)$ определяют некоторую кривую $y_1y_2y_3y_4$ (рис. 1.9). К множеству Парето относится участок y_2y_3. Участки y_1y_2 и y_3y_4 к множеству Парето не относятся, так как на них происходит одновременное увеличение P и L.

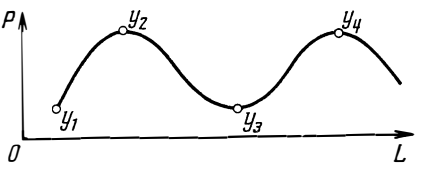


Рис. 1.9. Пример определения множества Парето

Отметим, что принцип Парето не дает единственного решения.

Он уменьшает число исследуемых вариантов, но окончательный выбор всегда за лицом, принимающим решение. В нашем случае это конструктор, который совместно с ЭВМ определяет «цену» увеличения одного из показателей, его влияние на остальные показатели, которые обязательно ухудшаются. Конструктор, построив множество Парето, облегчает процесс выбора решения.

Задачи, решаемые на этапе конструирования, относятся к классу задач многокритериальной оптимизации. Практически осуществить полную оптимизацию при конструировании затруднительно из-за большого числа частных критериев, переменных и ограничений. В настоящее время наиболее приемлемым способом оптимизации при автоматизации конструирования является использование *методов последовательной субоптимизации*. Сначала образуется обобщающий критерий суммированием частных критериев, умноженных на коэффициенты, задаваемые конструктором или определяемые на основе экспертных оценок. Суть метода последовательной субоптимизации состоит в отыскании оптимума по важнейшему критерию, далее по менее важному в допустимой области и т. д. Поскольку этапы конструкторского проектирования в САПР выполняются последовательно, то на каждом этапе можно получить оптимальные или квазиоптимальные результаты в соответствии со своими показателями качества.

При конструкторском проектировании ЭА основными этапами являются компоновка, размещение, трассировка и контроль. Каждому этапу соответствует свой частный критерий оптимальности. Для совместного учета частных критериев можно рассмотреть, например, обобщенный критерий оптимальности, который для задач конструкторского проектирования имеет вид $F(G)=[K(G), R(G), T(G), C(G)]$, где G — объект оптимизации (формальная модель схемы); $K(G)$, $R(G)$, $T(G)$, $C(G)$ — обобщенные критерии этапов компоновки, размещения, трассировки и контроля соответственно.

В свою очередь, каждый из обобщенных критериев любого этапа состоит также из частных критериев оптимальности. Так как каждому критерию в зависимости от класса исследуемых схем, заданных ограничений и т. п. можно поставить в соответствие некоторое число, которое характеризует его важность по сравнению с другими критериями, то возможно выполнить объединение критериев, другими словами, получить аддитивную функцию

$$F(G)=\sum_{i=1}^{s}\lambda_i F_i(G);$$

$$\lambda_i \geqslant 0;\ \sum_{i=1}^{s}\lambda_i = 1.$$

Полученное выражение позволяет объединить частные критерии в одно выражение. Например, для задач конструкторского проектирования, получим $F(G)=\lambda_1 K(G)+\lambda_2 R(G)+\lambda_3 T(G)+\lambda_4 C(G)$. Весовые коэффициенты λ_i определяются в основном из опыта конструктора или на основе экспертных оценок.

Таблица 1.1

Система (производитель)	Этапы автоматизированного конструирования	Оборудование	Характеристика	Конструктивно-техническая база
1	2	3	4	5
Аврора (СССР)	Компоновка, размещение, трассировка, изготовление фотошаблонов	ЭВМ М-222 — аппаратура для изготовления фотопозитивов	Поле 220×170 мм². Получение 8—10 слоев за 8—10 ч, 100 % разведенных соединений	МПП с выступающими выводами или открытыми контактными площадками
IBM (США)	Компоновка, размещение, трассировка, изготовление фотошаблонов, получение КТД	IBM-370, модель 155, координатограф Calcomp	Время конструирования печатной платы с 60 микросхемами 30 мин	МПП
ЕСАП (СССР)	Размещение, трассировка, изготовление фотооригиналов, получение КТД	ЭВМ «Система 4-50», установка изготовления фотооригиналов	Трассировка 500 соединений за 40 мин, 92 % разведенных соединений	МПП со сквозной металлизацией
АСП-1 (СССР)	То же	ЭВМ М-220	Поле 127×170 усл. ед., 3—9 % неразведенных соединений	ДПП, МПП
Автограф (СССР)	»	ЭВМ М-220, координатограф ЭМ-703, фототелеграфный аппарат ФАК-П	Поле 127×180 усл. ед., конструирование платы 5 ч, 2—3 % неразведенных соединений	ДПП, МПП
Каунас (СССР)	»	ЭВМ «Раздан-3», координатограф М-2000	Поле 140×150, 50 % ручной доработки. Конструирование на ЭВМ 20—45 мин	ДПП

Окончание табл. 1.1

1	2	3	4	5
Граф 2Д (СССР)	»	ЕС ЭВМ, координатограф Минск-2004	Поле 700×700 усл. ед. до 100 элементов на плате. Конструирование одной платы 30—50 мин, 90—95 % разведенных соединений	ДПП
Граф 2М (СССР)	То же	То же	Поле 700×700 усл. ед., до 100 элементов на плате, число слоев — до 12. Конструирование одной платы 30—50 мин, 90—95 % разведенных соединений	МПП
SPRINT (США)	Компоновка, размещение, трассировка, получение КТД	IBM-370, Textronix 4013	Интерактивное размещение 15 мин, автоматическая трассировка для плат с 49 элементами 58 с.	ДПП
MALS (Япония)	То же	11/40, Textronix 4014	100% разводки	ДПП
AIDS (США)	Размещение, трассировка, ручная доработка, получение КТД	ЭВМ, координатограф, Calcomp	Цикл конструирования платы 5—10 дней	МПП со сквозной металлизацией
LST (США)	Компоновка, размещение, трассировка, выпуск КТД, изготовление фотошаблонов	IBM-7090, координатограф	Время конструирования печатной платы на 100 выводов 30 мин	ДПП с фиксированными переходами
САПР (СССР)	Размещение, трассировка, выпуск КТД	ЕС ЭВМ, периферийное оборудование	100 % разведенных соединений	МПП с металлизацией сквозных отверстий
САПР (СССР)	То же	То же	95—97 % разведенных соединений	ДПП

Примечание. КТД — конструкторско-технологическая документация; МПП — многослойная печатная плата; ДПП — двусторонняя печатная плата.

В табл. 1.1 приводится характеристика некоторых существующих САПР.

1.3. КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Дайте характеристику основных этапов проектирования ЭА.
2. Опишите состав математического, программного, технического, информационного, лингвистического, организационного, методического обеспечения автоматизированного (автоматического) проектирования.
3. Что входит в комплекс средств автоматизации проектирования?
4. Дайте определение САПР.
5. Нарисуйте структурную схему основных этапов конструкторского проектирования.
6. Опишите методы линейной свертки, использования контрольных цифр, компромиссов Парето и поясните их использование для решения задач многокритериальной оптимизации применительно к конструкторскому проектированию.
7. Каким вы представляете себе конструкторское бюро в 2000 году? Опишите процесс взаимодействия его подразделений.
8. Какой вы представляете себе САПР в 2000 году?

2. МАТЕМАТИЧЕСКИЕ МЕТОДЫ, ИСПОЛЬЗУЕМЫЕ В СИСТЕМАХ АВТОМАТИЗАЦИИ КОНСТРУИРОВАНИЯ

2.1. ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ МНОЖЕСТВ

Понятие множества является первичным неопределяемым понятием. Его можно описать, применяя такие выражения, как «совокупность», «собрание», «класс», «вид» и т. д. Основатель теории множества Г. Кантор описывал множество как собрание определенных и различимых между собой объектов нашей интуации, мыслимое как единое целое.

В этом разделе будут приведены положения и определения теории множеств, которые широко используются для построения математических моделей конструкций ЭА.

Множества состоят из элементов и обычно обозначаются прописными буквами латинского алфавита. Элементы множества обычно обозначаются строчными буквами латинского алфавита.

Например, в выражении $A=\{a, b, c, ..., p\}$ A — множество; $a, b, c, ..., p$ — его элементы; фигурные скобки показывают, что элементы $a, b, c, ..., p$ объединены в одно целое — множество A.

В множестве по определению все элементы различны, а порядок перечисления элементов множества произвольный, поэтому совокупность элементов $A=\{a, a, b, b\}$ не является множеством. Принадлежность элемента x множеству A записывается $x \in A$, соответственно не принадлежность элемента множеству записывается $x \notin A$.

Множества бывают конечные и бесконечные. Множество с конечным числом элементов называется *конечным*. Так, множество элементов ЭА конечно. Множество $A=\{x, y, z, u, w\}$ конечно, так как содержит 5 элементов. Множество называется *бесконечным*,

если оно содержит бесконечное число элементов. Например, множества четных, нечетных, натуральных чисел соответственно: $A_1 = \{2, 4, 6, \ldots\}$, $A_2 = \{1, 3, 5, \ldots\}$, $N = \{1, 2, 3, \ldots\}$ — являются бесконечными множествами.

Множество A, состоящее из одного элемента, например d, обозначается $A = \{d\}$ и называется *одноэлементным*. Множество, которое не содержит элементов, называется *пустым* и обозначается знаком $\varnothing$. Число элементов множества B называется его *мощностью* и обозначается $|B|$. Например, множество $B = \{$ячейка, ТЭЗ, стойка, панель, шкаф$\}$ имеет мощность $|B| = 5$.

Существуют два основных способа задания множеств: *перечисление* и *описание*. Первый способ состоит в том, что задается и перечисляется полный список элементов, входящих в множество. Например, множество элементов схемы ЭА определяется их списком. Очевидно, что данный способ удобно применять только к ограниченному числу конечных множеств. Второй способ применяется, когда множество нельзя или затруднительно задать с помощью списка (это может относиться к конечным и бесконечным множествам). В таком случае множества определяются свойствами их элементов. Множество A задано, если указано свойство α, которым обладают все элементы, принадлежащие множеству A, и которым элементы, не принадлежащие множеству A, не обладают. Если A — произвольное множество, а α — некоторое свойство, то запись $A = \{b \in B / \alpha(b)\}$ означает множество тех и только тех элементов $b \in B$, которые обладают свойством α. Пусть B — множество ТЭЗ ЭА, а α — свойство быть ТЭЗ в блоке управления ЭА, тогда $A = \{b \in B / \alpha(b)\}$ — множество всех ТЭЗ блока управления ЭА.

Заметим, что при задании множеств вторым способом необходимо так задавать свойство, характеризующее элементы множества, чтобы оно было общим непротиворечивым для всех его элементов.

Множества A и B называются *равными*, если они содержат одинаковое число одних и тех же элементов, что обозначается так: $A = B$. Для любых множеств $A = A$; если $A = B$, то $B = A$ и если $A = B$, $B = C$, то $A = C$.

Говорят, что множество B является *подмножеством* множества A $(B \neq A)$, если любой элемент множества B принадлежит A. Обозначается это $B \subset A$, формулируется так: множество B строго включается в A $(B \subset A)$. Если возможно $B = A$, то говорят, что множество B нестрого включается в A и пишут $B \subseteq A$. Например, если множество A определяется свойством α, множество B — свойством β и $A \subset B$, то любой элемент, обладающий свойством α, должен обладать также и свойством β.

Пусть $B = \{$ИМС4, ИМС5, ИМС3$\}$, тогда $A = \{$ИМС5, ИМС3$\}$ включается в B, т. е. $A \subset B$, а $C = \{\{$ИМС4$\}$, ИМС5$\}$ не включается в B, так как множество $\{$ИМС4$\} \in C$, являющееся элементом множества C, не является элементом множества B.

Очевидно, что любое множество $A \neq \varnothing$ содержит, по крайней мере, два подмножества — пустое множество и само множество A, т. е. $\varnothing \subseteq A$ и $A \subseteq A$. Множества A и $\varnothing$ называются *несобственными подмножествами множества.* Остальные подмножества, если они существуют, называются собственными.

Если A — произвольное множество, то семейством всех подмножеств A называется и через (A) обозначается множество, элементами которого является все подмножества множества A. Например, если $A=\{\text{РЭА, ЭВА}\}$, то $(A)=\{\{\text{А}\}, \{\varnothing\}, \{\text{РЭА}\}, \{\text{ЭВА}\}\}$. Отметим, что $|(A)|=2^{A|}$.

Обычно все множества, с которыми имеют дело в построениях, являются подмножествами некоторого фиксированного множества I, называемого *универсальным.* Например, при решении задач компоновки схем на различных уровнях иерархии имеют дело с множеством множеств, т. е. с множеством, каждый элемент которого является множеством. Такие множества называют *системой множеств.*

Введем термин «высказывание». *Высказываниями* называют предложения, которые считаются истинными или ложными. Следовательно, каждое высказывание может быть истиной (ему присваивается значение «единица») или ложью (ему присваивается значение «нуль»). Например, высказывания «$2\times2=4$», «ЭВМ третьего поколения имеют большее быстродействие, чем ЭВМ первого поколения» являются истиной. Высказывания «$3>5$», «ИМС5-й степени интеграции содержит меньше элементов на таком же кристалле, чем ИМС4-й степени интеграции» являются ложью. В алгебре высказываний они рассматриваются с точки зрения их логических значений независимо от содержания.

К основным логическим операциям над высказываниями относятся: отрицание, конъюнкция, дизъюнкция, импликация, эквивалентность.

Отрицанием высказывания A называется новое высказывание, обозначаемое $\bar{A}$, которое считается истинным, если A ложное, и ложным, если A истинно. Операция отрицания соответствует логической частице «не». Например, для истинного высказывания «ЕС ЭВМ состоят из ТЭЗ» отрицанием будет ложное высказывание «неверно», что ЕС ЭВМ состоят из ТЭЗ».

Конъюнкцией высказываний A, B называется новое высказывание, обозначаемое $A \wedge B$ (читается A и B), которое считается истинным, если A и B истинны, и ложным, если хотя бы одно из них ложно. Например, для двух высказываний: «$2>0$», «ИМС4 содержит больше элементов, чем ИМС3» — конъюнкция «$2>0$ и ИМС4 содержит больше элементов, чем ИМС3» будет истинным высказыванием.

Дизъюнкцией высказываний A, B называется новое высказывание, обозначаемое $A \vee B$ (читается A или B), которое считается истинным, если хотя бы одно из высказываний A, B истинно, и ложным, если они оба ложны. Например, для двух высказываний:

«ТЭЗ содержит меньше интегральных микросхем, чем панель из нескольких ТЭЗ» и «$3^2=9$» — дизъюнкция «ТЭЗ содержит меньше интегральных микросхем, чем панель из нескольких ТЭЗ или $3^2=9$» будет истинным высказыванием.

Импликацией высказываний A, B называется высказывание, обозначаемое $A \to B$ (читается «A влечет B» или «если A, то B»), которое ложно, если A истинно и B ложно, и истинно для всех других логических значений A, B. Например, высказывание «если $5 \times 5 \neq 25$, то ЭВА третьего поколения строятся на основе ТЭЗ» истинно.

Эквивалентностью высказываний A, B называется высказывание, обозначаемое $A \leftrightarrow B$ (читается «A равносильно B» или «для того чтобы A, необходимо и достаточно, чтобы B»), которое считается истинным, когда оба высказывания A, B либо истинны, либо ложны, и ложным в остальных случаях. Например, высказывание «для того чтобы автоматически конструировать современную ЭА, необходимо и достаточно иметь действующую систему автоматизированного проектирования» истинно.

На рис. 2.1 показаны таблицы истинности для рассмотренных высказываний. Здесь и — истина, л — ложь.

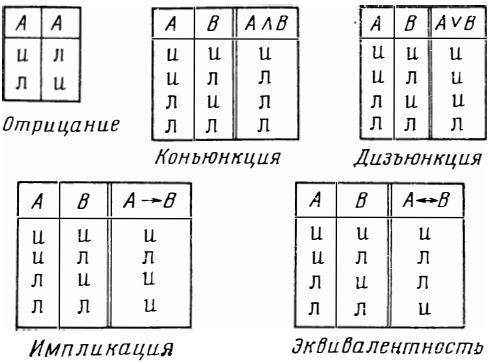


Рис. 2.1. Таблицы истинности высказываний

Дадим понятие квантора существования и квантора общности. Запись $(\forall x \in X) B(x)$ означает, что для любого элемента из множества X истинно высказывание $B(x)$ об элементе x. Знак $\forall$ называют *квантором общности.*

Запись $(\exists x \in X) B(x)$ означает, что существует хотя бы один элемент $x \in X$, для которого истинно высказывание $B(x)$ об этом элементе. Знак $\exists$ называют *квантором существования.*

Объединением множеств A и B называется множество C, состоящее из элементов как множества A, так и множества B. Обозначается $C = A \cup B$. Высказывание $y \in A \cup B$ эквивалентно $y \in A \vee y \in B$. Например, при $A = \{$транзистор, резистор$\}$, $B = \{$конденсатор, индуктивность$\}$, то $C = A \cup B = \{$транзистор, резистор, конденсатор,

индуктивность}. Если A — множество точек левого прямоугольника (рис. 2.2), а B — множество точек правого прямоугольника, то заштрихованная область есть $A \cup B$.

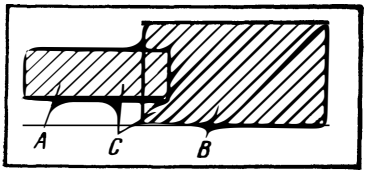


Рис. 2.2. Объединение множеств $C=A \cup B$

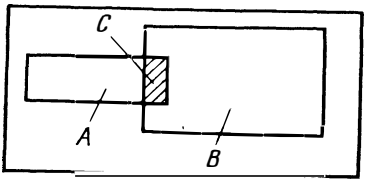


Рис. 2.3. Пересечение множеств $C=A \cap B$

Объединение множеств A_1, A_2, ..., A_n обозначается $\bigcup_{i=1}^{n} A_i$. Оно состоит из всех элементов, которые принадлежат хотя бы одному из множеств A_1, A_2, ..., A_n.

Пересечением множеств A и B называется множество C, состоящее из элементов, принадлежащих одновременно и множеству B, и множеству A. Обозначается $C=A \cap B$. Следовательно, $y \in A \cap B \leftrightarrow y \in A \wedge y \in B$. Например, если $A=$ {ИМС4, ИМС5, электронная лампа, реле}, $B=$ {ИМС4, ИМС3}, то $C=A \cap B=$ {ИМС4}. Если A — множество точек левого прямоугольника (рис. 2.3), а B — множество точек правого прямоугольника, то заштрихованная область есть $A \cap B$. Пересечение множеств A_1, A_2, ..., A_n обозначается $\bigcap_{i=1}^{n} A_i$. Оно состоит из всех элементов, принадлежащих одновременно множествам A_1, A_2, ..., A_n.

Из определения пересечения множеств, следует, что $A \cap A=A$. Если множества A и B не имеют общих элементов, то их пересечение представляет собой пустое множество.

Разностью множеств A и B называется множество C, состоящее из элементов, принадлежащих A и не принадлежащих B. Обозначается $C=A \setminus B$. Тогда $y \in A \setminus B \leftrightarrow y \in A \wedge y \notin B$. Например, если $A=$ {ИМС4, ИМС5, ТЭЗ}, а $B=$ {ИМС4, транзистор}, то $A \setminus B=$ = {ИМС5, ТЭЗ}. Если A — множество точек левого прямоугольника, а B — множество точек правого прямоугольника (рис. 2.4), то заштрихованная область есть множество $C=A \setminus B$.

Очевидно, что $A \setminus B \subseteq A$, $B \setminus A \subseteq B$.

Когда $A \subseteq B$, то множество $\bar{A}=B \setminus A$ называется *дополнением множества* A *до* B. Для произвольного множества $M \in I$ можно определить дополнение до *универсального множества* $\bar{M}=I \setminus M$. Для любого множества можно определить дополнение до любого другого множества, включающего его. Если это не оговорено, то считается, что дополнение берется до универсального множества I.

Пример показан на рис. 2.5. Здесь заштрихованная область представляет собой $\bar{A}=I \setminus A$.

Рассмотрим основные свойства операций $\cup$, $\cap$, $\setminus$.

Законы коммутативности

$$A \cup B = B \cup A; \quad A \cap B = B \cap A.$$

Законы ассоциативности

$$A \cup (B \cup C) = (A \cup B) \cup C; \quad A \cap (B \cap C) = (A \cap B) \cap C.$$

Законы дистрибутивности.

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C);$$
$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C).$$

Законы идемпотентности

$$A \cup A = A; \quad A \cap A = A.$$

Законы де Моргана

$$A \setminus (B \cap C) = (A \setminus B) \cup (A \setminus C); \quad A \setminus (B \cup C) = (A \setminus B) \cap (A \setminus C).$$

Основным методом доказательств тождеств с множествами является метод двух включений (или взаимного включения). Для доказательства $E = F$ необходимо показать, что $E \subseteq F \wedge F \subseteq E$. Чтобы

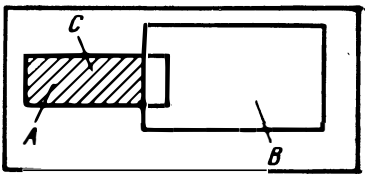


Рис. 2.4. Разность множеств $C = A \setminus B$

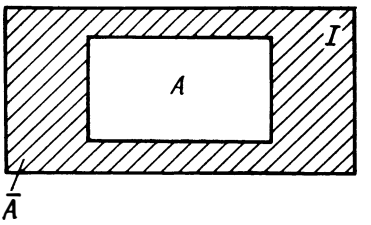


Рис. 2.5. Дополнение $\bar{A} = I \setminus A$

доказать, что $E \subseteq F$, требуется из истинности высказывания $a \in E$ вывести истинность высказывания $a \in F$. И, наоборот, для доказательства $E \subseteq E$ необходимо из истинности высказывания $a \in F$ вывести истинность высказывания $a \in E$.

Например, докажем справедливость

$$\underbrace{A \cap (B \cup C)}_{E} = \underbrace{(A \cap B) \cup (A \cap C)}_{F};$$

Обозначим левую часть через E, а правую — через F. Теперь необходимо доказать, что $E = F$. Это равносильно доказательству

$$E \subseteq F \wedge F \subseteq E.$$

Пусть истинно высказывание $a \in E$, тогда

$$a \in E \to a \in A \cap (B \cup C) \to a \in A \wedge a \in (B \cup C) \to a \in A \wedge (a \in B \vee a \in C) \to a \in$$
$$\in A \wedge a \in B \vee a \in A \wedge a \in C \to a \in (A \cap B) \vee a \in (A \cap C) \to a \in (A \cap B) \cup$$
$$\cup (A \cap C) \to a \in F.$$

Пусть теперь истинно высказывание $a \in F$, тогда

$$a \in F \to a \in (A \cap B) \cup (A \cap C) \to a \in A \wedge a \in B \vee a \in A \wedge a \in C \to a \in A \wedge (a \in B \vee a \in C) \to a \in A \cap (B \cup C) \to a \in E,$$

что и требовалось доказать.

Иногда требуется доказать не равенство двух множеств, а равенство множества пустому множеству, т. е. $A = \varnothing$. Обычно доказательство проводят методом от противного, т. е. предполагают что $A \neq \varnothing$, следовательно, существует хотя бы один элемент $a \in A$, и тогда пытаются доказать, что предположение $A \neq \varnothing$ является ложным и приводит к противоречию, на основании чего заключают, что $A = \varnothing$.

Выше мы рассматривали неупорядоченные множества. Для задания упорядоченных множеств используется термин *кортеж*. Понятие кортежа, как и понятие множества, является неопределяемым понятием. Кортеж состоит из компонент, для которых задается местоположение. Обычно кортежи обозначаются греческими буквами, а их компоненты — латинскими. Например, $\alpha = \langle a, b, c, c, c \rangle$. Такая запись означает, что кортеж α состоит из пяти компонент: a — расположенной на 1-м месте, b — на 2-м месте, c — на 3-м месте, c — на 4-м месте и c — на 5-м месте.

Компонентами кортежей могут быть любые объекты, в том числе множества и кортежи.

Кортежи α и β называются равными, если каждая компонента α совпадает с компонентой β с тем же номером. Например, $\alpha = \langle 9,5 \rangle = \beta = \langle 9,5 \rangle$ и $\alpha = \langle 5,9 \rangle \neq \beta = \langle 9,5 \rangle$. Следовательно, $\langle a, b \rangle = \langle c, d \rangle \leftrightarrow (a = c) \wedge (b = d)$. Число компонент кортежа называют его *длиной*.

Декартовым (прямым) произведением множеств A и B называют множество C, состоящее из всех различных кортежей длины 2, первая компонента которых принадлежит A, а вторая — B. Обозначается $C = A \times B$. Например, если $A = \{a, b\}$, $B = \{x, z\}$, то

$$C = A \times B = \{\langle a, x \rangle, \langle a, z \rangle, \langle b, x \rangle, \langle b, z \rangle\}.$$

Следовательно, для любого кортежа, являющегося элементом множества C, истинно высказывание $\langle a, x \rangle \in C = A \times B \leftrightarrow (a \in A) \wedge (x \in B)$.

Кортежи длины 2 называются *упорядоченными парами*, кортежи длины 3 — *тройками* и т. д.

Декартовым произведением n множеств $A_1, A_2, \ldots, A_n$ называют множество $C = A_1 \times A_2 \times \ldots \times A_n$, состоящее из всех кортежей длины n, первая компонента которых принадлежит A_1, вторая — $A_2, \ldots$, компонента n — A_n.

Очевидно, что $A \times B = \varnothing \leftrightarrow A = \varnothing \vee B = \varnothing$. *Степенью s множества* A называется декартово произведение одинаковых множеств A, т. е.

$$A^s = \underbrace{A \times A \times \ldots \times A}_{s \text{ раз}}.$$

Считают $A^1 = A$ и $A^0 = \{0\}$, где 0 — пустой кортеж.

Множество, каждый элемент которого является кортежем, называется *графиком*. Для графиков определены две основные операции: инверсия и композиция. *Инверсия графика* определяется через инверсию его кортежей. Кортеж $\langle x, y\rangle$ есть инверсия кортежа $\langle p, q\rangle$, если $y=p$ и $x=q$. Обозначается α^{-1}. Например, если $\alpha=\langle x, y\rangle$, $\alpha^{-1}=\langle y, x\rangle$.

График R называется *композицией графиков* P и Q, если $\langle x, y\rangle\in R$, тогда и только тогда, когда существует такое z, что $\langle x, z\rangle\in P$ и $\langle z, y\rangle\in Q$. Обозначается $R=P\bigcirc Q$. Знак $\bigcirc$ обозначает, что выполняется композиция графиков P и Q.

Пусть $P=\{\langle x, y\rangle, \langle x, x\rangle, \langle x, z\rangle, \langle z, z\rangle\}$ и

$Q=\{\langle y, y\rangle, \langle y, z\rangle, \langle y, x\rangle\}$.

Тогда $P\bigcirc Q=\{\langle x, y\rangle, \langle x, z\rangle, \langle x, x\rangle\}$.

Пусть произвольная пара $\langle x, y\rangle\in R=P\bigcirc Q$. В этом случае для нее справедливо высказывание $\langle x, y\rangle\in P\bigcirc Q \leftrightarrow(\exists\, d)\langle x, d\rangle\in P\wedge \wedge\langle d, y\rangle\in Q)$. Элемент d в $P\bigcirc Q$ является *компонирующим элементом*.

Очевидно, что $A\bigcirc\varnothing=\varnothing\bigcirc A=\varnothing$. Операции композиции и пересечения множеств связаны следующим образом:

$$A\bigcirc(B\cap C)\subseteq(A\bigcirc B)\cap(A\bigcirc C);$$
$$(B\cap C)\bigcirc A\subseteq(B\bigcirc A)\cap(C\bigcirc A).$$

Многие задачи конструирования тесно связаны с различными вопросами разбиения множеств. Наибольший интерес представляет разбиение множества на непересекающиеся подмножества. Например, используя классификацию ИМС можно выделить три непересекающихся подмножества: ИМС1, ИМС3 и ИМС5.

Очевидно, что одно и то же множество можно разбить на подмножества различными способами. Система множеств M называется *разбиением множества* A, если

$$(\forall A_i\in M)[A_i\subseteq A \text{ и } A_i\neq\varnothing].$$
$$(\forall A_i, A_j\in M)[A_i\neq A_j\rightarrow A_i\cap A_j=\varnothing].$$
$$A\subseteq\bigcup_{A_i\in M}A_i.$$

Например, множество $A=\{$ИМС3, ИМС1, ИМС5$\}$ имеет пять разбиений:

$M_1=\{A\}$, $M_2=\{\{$ИМС3$\}$, $\{$ИМС1$\}$, $\{$ИМС5$\}\}$, $M_3=\{\{$ИМС3$\}$, $\{$ИМС1, ИМС5$\}\}$, $M_4=\{\{$ИМС1$\}$, $\{$ИМС3, ИМС5$\}\}$, $M_5=$ $=\{\{$ИМС5$\}$, $\{$ИМС3, ИМС1$\}\}$.

Из рассмотренного примера видно, что попарное пересечение подмножеств внутри разбиения пусто.

Более общим понятием является покрытие множеств. Пусть A — произвольное непустое подмножество. *Покрытием множества*

A называется семейство $\mathcal{K}$ множеств, для которых справедливо следующее: $(\forall A_i \in \mathcal{K})(A_i \neq \varnothing)$, $(\forall A_i \in \mathcal{K})(A_i \subseteq A)$, $\bigcup\limits_{A_i \in \mathcal{K}} A_i = A$. Другими словами, покрытие $\mathcal{K}$ — это совокупность непустых подмножеств A_i, объединение которых есть A. Элементы семейства $\mathcal{K}$ называются классами покрытия множества A. Например, классами покрытия множества ЭВА являются множество универсальной ЭВА, множество специализированной ЭВА, множество смешанной ЭВА. Другим примером является покрытие множества $A = \{1, 2, 3, ..., 20\}$ классами, состоящими из чисел. Эти классы, например, будут $A_1 = \{1, 2, 3, 10, 12, 20\}$, $A_2 = \{2, 3, ..., 10, 11, 12, ..., 20\}$.

Многие задачи науки и техники могут иметь формальную интерпретацию на языке теории отношений. Все арифметические операции — это отношения между числовыми множествами. Множество элементов схем ЭВМ, ЭВА или РЭА остается набором элементов, пока между ними не реализуются определенные отношения, превращающие эти элементы в вычислительную машину или прибор. Разнообразные отношения возникают между родителями и детьми, руководителями и подчиненными, в природе и т. д. Вообще говоря, отношение — это связь между любыми объектами в природе.

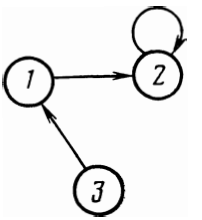


Рис. 2.6. Отношение φ

В теоретико-множественном плане *отношение* — это кортеж длины 2, т. е. пара, первая компонента которой является подмножеством квадрата второй компоненты. Обозначается отношение $\varphi = \langle \Phi, B \rangle$, $\Phi \subseteq B^2$, где Φ — график отношения; B — область задания отношения.

Рассмотрим ряд примеров отношений. Если φ — отношение параллельности, то $a \varphi b$ показывает, что a параллельно b. Здесь a, b — элементы отношения. Если $\Phi = \{\langle 1,2 \rangle, \langle 2,2 \rangle, \langle 3,1 \rangle\}$, $B = \{1, 2, 3\}$, то определим отношение $\varphi = \langle \{\langle 1,2 \rangle, \langle 2,2 \rangle, \langle 3,1 \rangle\}, \{1, 2, 3\} \rangle$. Графическое изображение отношения φ показано на рис. 2.6. В кружках стоят цифры, соответствующие элементам отношения, а линии со стрелками показывают связь между элементами отношения.

Отношение φ называется полным, если $\Phi = B^2$, т. е. если высказывание $(\forall x, y \in B)[x \varphi y]$ всегда истинно.

Отношение φ называется пустым, если $\Phi = \varnothing$, т. е. если высказывание $(\forall x, y \in B)(x \varphi y)$ всегда ложно.

Отношение φ называется отношением равенства, если высказывание $(\forall x, y \in B)(x \varphi y \rightarrow x = y)$ истинно.

Отношение φ называется отношением неравенства, если высказывание $(\forall x, y \in B)(x \varphi y \rightarrow x \neq y)$ истинно.

Заметим, что на отношение переносятся операции над множествами. Пусть $\varphi_1 = \langle \Phi_1, B \rangle$ и $\varphi_2 = \langle \Phi_2, B \rangle$ есть произвольные отношения, заданные на одном и том же множестве B.

Тогда *объединением отношений* φ_1 и φ_2 называется отношение φ_3, график которого Φ_3 равен объединению графиков отношений

φ_1 и φ_2: $\varphi_3=\varphi_1\cup\varphi_2=\langle\Phi_1\cup\Phi_2, B\rangle$. Очевидно, что $x(\varphi_1\cup\varphi_2)y\leftrightarrow$ $\leftrightarrow(x\varphi_1 y)\vee(x\varphi_2 y)$. Аналогично определяются операции пересечения и разности отношений. Например, если

$$\varphi_1=\langle\{\langle a, b\rangle, \langle b, c\rangle, \langle a, d\rangle\}, \{a, b, c, d\}\rangle,$$

$$\varphi_2=\langle\{\langle a, b\rangle, \langle b, c\rangle\}, \{a, b, c, d\}\rangle, \text{ то}$$

$$\varphi_3=\varphi_1\cap\varphi_2=\langle\Phi_1\cap\Phi_2, B\rangle=\langle\{\langle a, b\rangle, \langle b, c\rangle\}\{a, b, c, d\}\rangle.$$

Для отношений, заданных на одном и том же множестве, справедливы тождества, аналогичные тождествам над множествами.

Также справедливы операции инверсии и композиции, которые рассматривались для графиков.

Инверсией отношения φ на множестве B является отношение, график которого есть инверсия графика отношения $\varphi=\langle\Phi, B\rangle$, т. е. $\varphi^{-1}=\langle\Phi^{-1}, B\rangle$. Следовательно, $x\varphi^{-1}y\to y\varphi x$.

Композиция отношений $\varphi_1=\langle\Phi_1, B\rangle$ и $\varphi_2=\langle\Phi_2, B\rangle$ определяется выражением $\varphi_3=\varphi_1\circ\varphi_2=\langle\Phi_1\circ\Phi_2, B\rangle$.

Отношение удобно задать в виде прямоугольной таблицы (матрицы). Строки и столбцы матрицы отношений $\mathbf{R}_\varphi$ соответствуют элементам x, $y\in B$. На пересечении строки, соответствующей элементу x_i, и столбца, соответствующего элементу x_j, ставится единица, если выполняется отношение $x_i\varphi x_j$, и нуль в противном случае. Матрица отношения, показанного на рис. 2.6, имеет вид

$$\mathbf{R}_\varphi=\begin{array}{c|ccc} & 1 & 2 & 3 \\ 1 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 1 & 0 & 0 \end{array}.$$

Размер матрицы $\mathbf{R}_\varphi$ равен $|B|\times|B|$, т. е. общее число клеток (ячеек) R_φ равно мощности B^2.

Для отношения $\varphi=\langle\Phi, B\rangle$, где $B=\{x_1, x_2, x_3, y_1, y_2\}$,

$$\Phi=\{\langle x_1y_1\rangle, \langle x_2, y_2\rangle, \langle x_3, y_1\rangle,$$
$$\langle x_3, y_2\rangle, \langle y_1, x_1\rangle, \langle y_2, x_2\rangle,$$
$$\langle y_2, y_2\rangle, \langle x_2, x_2\rangle, \langle y_2, x_3\rangle,$$
$$\langle x_1, y_2\rangle\},$$

матрица отношения примет вид

$$\mathbf{R}_\varphi=\begin{array}{c|ccccc} & x_1 & x_2 & x_3 & y_1 & y_2 \\ x_1 & 0 & 0 & 0 & 1 & 1 \\ x_2 & 0 & 1 & 0 & 0 & 1 \\ x_3 & 0 & 0 & 0 & 1 & 1 \\ y_1 & 1 & 0 & 0 & 0 & 0 \\ y_2 & 0 & 1 & 1 & 0 & 1 \end{array}.$$

Полному отношению соответствует матрица, все клетки которой заполнены единицами, а пустому — нулевая матрица.

Рассмотрим основные свойства отношений.

Отношение $\varphi=\langle\Phi, B\rangle$ *называется рефлексивным*, если высказывание $(\forall x\in B)[x\varphi x]$ является истинным высказыванием. Отношение «находится в одной и той же схеме ЭА» для одного элемента x является рефлексивным отношением. Матрица рефлексивного отношения имеет все единичные элементы, расположенные по главной диагонали.

Отношение $\varphi=\langle\Phi, B\rangle$ *называется антирефлексивным*, если высказывание $(\forall x\in B)[x\bar{\varphi}x]$ истинно. Здесь $\bar{\varphi}$ обозначает инверсию отношения φ. В матрице антирефлексивного отношения по главной диагонали нет ни одного единичного элемента.

Отношение $\varphi=\langle\Phi, B\rangle$ *называется симметричным*, если

$$(\forall x, y\in B)(x\varphi y\rightarrow y\varphi x).$$

Отношение «находиться в одном ТЭЗ ЭВА» для двух ИМС x и y является симметричным отношением, так как если $ИМС_x$ и $ИМС_y$ находятся в одном ТЭЗ, то $ИМС_y$ и $ИМС_x$ находятся в одном ТЭЗ.

Отношение φ *называется антисимметричным*, если $(\forall x, y\in B)(x\varphi y\wedge x\neq y\rightarrow y\bar{\varphi}x)$. Отношение «больше» является примером антисимметричного отношения.

Отношение φ *называется транзитивным*, если $(\forall x, y, z\in B)(x\varphi y\wedge y\varphi z\rightarrow x\varphi z)$.

Отношение «находиться в одном кристалле ИМС» для транзисторов x, y, z является транзитивным отношением.

Другими словами, если элементы x и y находятся на некотором кристалле ИМС и элементы y и z находятся на том же кристалле, следовательно, на этом же кристалле будут находиться x и z. Отношение равенства является транзитивным, так как если $x=y$ и $y=z$, то и $x=z$.

В практике автоматизации конструирования схем интерес представляют отношения, обладающие совокупностью свойств.

Отношение φ *называется отношением эквивалентности*, если оно рефлексивно, симметрично и транзитивно.

Например, отношение «параллельности» прямых является отношением эквивалентности, так как $a\|a$ (рефлексивность), $a\|b\rightarrow b\|a$ (симметричность), $a\|b\wedge b\|c\rightarrow a\|c$ (транзитивность).

Отношение «перпендикулярности» прямых не является отношением эквивалентности, так как a не перпендикулярно a (рефлексивность не выполняется), $a\perp b\rightarrow b\perp a$ (симметричность), $a\perp b\wedge b\perp c$ не влечет $a\perp c$ (транзитивность не выполняется).

Отношение «находиться в одной и той же схеме ЭА» для ее элементов является отношением эквивалентности.

Основное значение отношения эквивалентности состоит в том, что оно определяет признак, который допускает разбиение множества M на непересекающиеся подмножества, называемые *классами эквивалентности*. Все элементы, принадлежащие некоторому классу M_i разбиения множества M, связаны отношением эквивалентности.

Матрица $\mathbf{R}_\varphi$ отношения эквивалентности φ состоит из клеток (подматриц), расположенных по главной диагонали, причем все элементы каждой подматрицы являются единицами. Матрица отношения эквивалентности может быть приведена к такому виду с помощью перестановок строк и столбцов. Например, если переставить строки и столбцы матрицы, записанной ниже, ее конфигурация изменится, но матрица не перестанет быть матрицей отношения эквивалентности. Пусть, например, на множестве $B=\{$ЭВМ, ЭВА, РЭА, ИМС4, ИМС5, ТЭЗ1, ТЭЗ2, ТЭЗ3, ТЭЗ4$\}$ задано отношение эквивалентности «иметь одинаковую конструктивную реализацию». Тогда множество B можно разбить на три класса, содержащие эквивалентные по заданному отношению элементы: $B_1=\{$ЭВМ, ЭВА, РЭА$\}$, $B_2=\{$ИМС4, ИМС5$\}$, $B_3=\{$ТЭЗ1, ТЭЗ2, ТЭЗ3, ТЭЗ4$\}$.

Матрица этого отношения запишется

$R_\varphi =$

	ЭВМ	ЭВА	РЭА	ИМС4	ИМС5	ТЭЗ1	ТЭЗ2	ТЭЗ3	ТЭЗ4
ЭВМ	1	1	1	0	0	0	0	0	0
ЭВА	1	1	1	0	0	0	0	0	0
РЭА	1	1	1	0	0	0	0	0	0
ИМС4	0	0	0	1	1	0	0	0	0
ИМС5	0	0	0	1	1	0	0	0	0
ТЭЗ1	0	0	0	0	0	1	1	1	1
ТЭЗ2	0	0	0	0	0	1	1	1	1
ТЭЗ2	0	0	0	0	0	1	1	1	1
ТЭЗ4	0	0	0	0	0	1	1	1	1

Отношение эквивалентности позволяет разбивать множество, на котором задано отношение, на непересекающиеся классы, содержащие эквивалентные между собой элементы. Дальнейшее преобразование заданного множества позволяет вместо каждого класса исследовать один его произвольный элемент.

Мы рассматривали отношения, в которых Φ могло состоять из кортежей длины 2. Такие отношения называются бинарными. Если Φ будет состоять из кортежей длины n, то в общем случае можно рассматривать n отношения. Отношение $\varphi=\langle\Phi, B\rangle$, заданное на множестве B, называется n, если $\Phi\subseteq B^n$. Отношение «образовать вычислительную систему из четырех микропроцессоров» на множестве микропроцессоров одного типа является примером 4-го отношения.

Моделью M называется совокупность множества B и конечного числа отношений, определенных на этом множестве: $\varphi_1=\langle\Phi_1, B\rangle$, $\varphi_2=\langle\Phi_2, B\rangle$, ..., $\varphi_l=\langle\Phi_l, B\rangle$.

В общем случае можно записать $M=(B, \Phi_1, \Phi_2, ..., \Phi_l)$. Говорят, что *соответствие* — это кортеж длины 3, первая компонента которой является подмножеством декартового произведения вто-

рой и третьей компонент: $\Gamma=\langle\Phi, X, Y\rangle$, $\Phi\subseteq X\times Y$, где Φ — график соответствия; X — область отправления; Y — область прибытия.

Образом множества A при соответствии Γ называется множество $\Gamma(A)$ тех элементов области прибытия Y, каждый из которых соответствует в Γ какому-нибудь элементу из A.

Например, пусть $\Gamma=\langle\Phi, X, Y\rangle$, где $\Phi=\{\langle a, 2\rangle, \langle b, 3\rangle, \langle c, 2\rangle, \langle b, 1\rangle\}$, $X=\{a, b, c\}$, $Y=\{1, 2, 3\}$.

Соответствие Γ показано на рис. 2.7. Если $A=\{a, b\}$, то $\Gamma(A)=\Gamma(\{a, b\})=\{1, 2, 3\}$. Из примера видно, что $\Gamma(A)\subseteq Y$.

Полным прообразом множества B при соответствии Γ называется множество $\Gamma^{-1}(B)$ тех элементов области отправления X, каждому из которых соответствует какой-нибудь элемент $B\subseteq Y$.

Если в примере (см. рис. 2.7) $B=\{1,3\}$, то $\Gamma^{-1}(B)=\{b\}$, причем $\Gamma^{-1}(B)\subseteq X$.

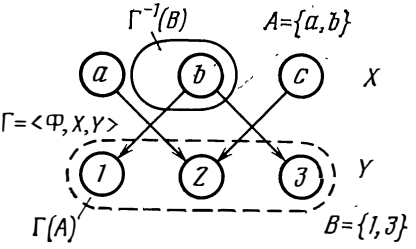


Рис. 2.7. Соответствие Γ

Соответствие Γ можно задать с помощью матрицы $\mathbf{R}_\Gamma$ размером $|X|\times|Y|$. Строки матрицы $\mathbf{R}_\Gamma$ соответствуют элементам $x_i\in X$, а столбцы — элементам $y_j\in Y$. На пересечении i-й строки и j-го столбца ставится единица, если $\langle x_i, x_j\rangle\in\Phi$, и нуль в противном случае. Для соответствия $\Gamma=\langle\Phi, X, Y\rangle$, изображенного на рис. 2.7, матрица $\mathbf{R}_\Gamma$ примет вид

$$\mathbf{R}_\Gamma=\begin{array}{c} \\ a \\ b \\ c \end{array}\begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \left\|\begin{array}{ccc} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{array}\right\| \end{array}.$$

Над соответствиями можно выполнять те же операции, что и над множествами и отношениями.

Соответствие $f=\langle\Phi, X, Y\rangle$ называется функциональным, если график Φ не имеет кортежей с одинаковыми первыми и различными вторыми компонентами, т. е. график не может иметь двух пар вида $\langle x_i, y_1\rangle, \langle x_i, y_2\rangle$, $y_1\neq y_2$. В противном случае соответствие называется нефункциональным. Функциональное соответствие называют однозначным, а нефункциональное — многозначным. Пример функционального графика $\Phi=\{\langle a, 1\rangle\langle b, 1\rangle\langle c, 1\rangle\langle d, 1\rangle\}$ показан на рис. 2.8.

Соответствие $\Gamma=\langle\Phi, X, Y\rangle$ называется инъективным, если в графике Φ не может быть двух пар вида $\langle x_1, y_j\rangle, \langle x_2, y_j\rangle$; $x_1\neq x_2$. В противном случае соответствие называется *неинъективным*.

Соответствие $\Gamma=\langle\Phi, X, Y\rangle$ называется всюду определенным, если для любого $x\in X$ образ $\Gamma(x)\neq\varnothing$.

Соответствие называется сюръективным, если для любого $y\in Y$ прообраз $\Gamma^{-1}(y)\neq\varnothing$. Произвольное соответствие $\Gamma=\langle\Phi, X, Y\rangle$

может обладать или не обладать любым из описанных свойств или их совокупностью.

Соответствие $\Gamma=\langle\Phi, X, Y\rangle$ называется биективным или взаимно-однозначным, если оно функционально, инъективно, всюду определено и сюръективно, т. е. соотвествие Γ взаимно-однозначно, если каждому элементу x соответствует один и только один элемент y и наоборот. На рис. 2.9 показан пример взаимно-однозначного соответствия между множествами X и Y. Здесь $a \leftrightarrow 3$, $b \leftrightarrow 1$, $c \leftrightarrow 2$.

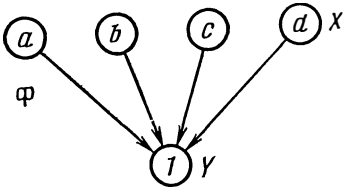


Рис. 2.8. Пример функционального графика Φ

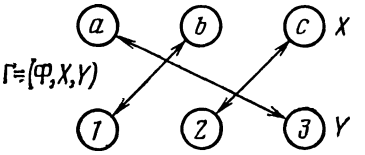


Рис. 2.9. Взаимно-однозначное соответствие между множествами X и Y

Функциональные соответствия являются частным видом соответствий, наиболее часто используемых при разработке алгоритмов автоматизированного конструирования ЭА. Они называются *функциями* и обычно обозначаются $f=\langle\Phi, X, Y\rangle$. Тот факт, что функция f имеет область отправления X и область прибытия Y, выражают словами: функция f определена в X и принимает свои значения в Y. Обозначается $X \xrightarrow{f} Y$ или $f: X \to Y$. Значение функции f на элементе a обозначается $f(a)$. Если кортеж $\langle x, y\rangle \in \Phi$, то образом элемента x при функции f будет элемент y. Это обычно обозначают $f(x)=y$ и говорят, что функция f отображает элемент $x \in X$ в элемент $y \in Y$. Элемент x называется аргументом функции f, а y — значением функции f на агрументе x.

Функция $f=\langle\Phi, X, Y\rangle$ называется нигде не определенной, если $\Phi \neq \varnothing$.

Функция $f=\langle\Phi, X, Y\rangle$ называется всюду определенной, когда $(\forall x \in X)(\exists f(x))$.

Всюду определенную функцию называют отображением, а отображение типа $X \xrightarrow{f} Y$ называют отображением множества X в Y.

Функция $f=\langle\Phi, X, Y\rangle$ называется сюръективной, если $(\forall y \in Y)(\exists x \in X)[f(x)=y]$. Функция $f=\langle\Phi, X, Y\rangle$ называется инъективной, если $(\forall x \in X)(\exists y \in Y)[f(y) \neq x]$. Функция $f=\langle\Phi, X, Y\rangle$ тогда и только тогда биективна, когда она всюду определена, сюръективна и инъективна. Синонимом термину «биекция X на Y» является взаимно-однозначное отображение множества X на Y.

Примеры сюръективной, инъективной и биективной функции показаны на рис. 2.10, 2.11, 2.12 соответственно.

Биективная функция или просто биекция типа t $X \to X$ называется *подстановкой на множестве* X. Подстановку записывают в

виде двух строк, заключенных в круглые скобки, причем под элементом $x \in X$ записывается элемент $t(x)$. Например, пусть $X = \{x_1, x_2, x_3, x_4\}$. Тогда некоторая произвольная подстановка t может иметь вид

$$t = \begin{pmatrix} x_1, & x_2, & x_3, & x_4 \\ x_2, & x_4, & x_1, & x_3 \end{pmatrix}.$$

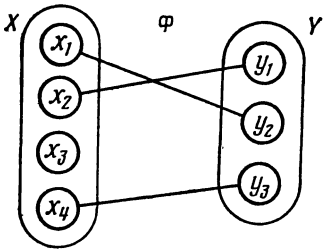


Рис. 2.10. Сюръективная функция $f = (\Phi, X, Y)$

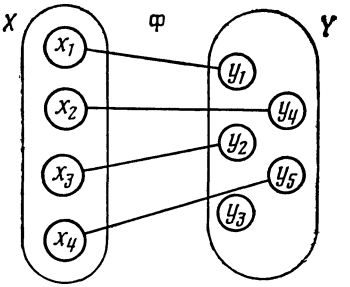


Рис. 2.11. Иньективная функция $f = (\Phi, X, Y)$

Такая запись означает, что элемент x_1 отображается в x_2; x_2 — в x_4; x_3 — в x_1 и x_4 — в x_3. Существуют и обратные подстановки, которые можно применять для возвращения в исходное состояние. Для рассматриваемого примера

$$t^{-1} = \begin{pmatrix} x_1, & x_2, & x_3, & x_4 \\ x_3, & x_1, & x_4, & x_2 \end{pmatrix}.$$

Отметим, что для соответствий и функций также справедливы все операции над множествами и отношениями.

В практике конструирования часто приходится иметь дело с множествами, в которых нельзя указать резкую границу, отделяющую элементы, принадлежащие к данному множеству и не принадлежащие к нему. Такие множества называют расплывчатыми или нечеткими. Следует отличать понятия случайности и расплывчатости. Случайность связана с неопределенностью, относящейся к принадлежности или непринадлежности элемента к множеству. Расплывчатость связана с множествами, в которых могут иметься различные степени принадлежности элементов к данному множеству.

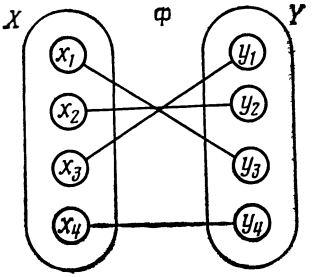


Рис. 2.12. Биективная функция $f = (\Phi, X, Y)$

Пусть задано множество $X = \{x_1, x_2, \ldots, x_n\}$. Тогда говорят, что расплывчатое множество A на множестве X есть совокупность кортежей $\tilde{A} = \{\langle \mu_A(x_i); x_i \rangle\}$, $x_i \in X$, где $\mu_A(x_i)$ — степень принадлежности элемента x_i к A; μ_A — функция, отображающая x_i

в пространство M, называемое пространством принадлежности. Предполагается, что M — это интервал [0,1], причем 1 и 0 представляют высшую и низшую степени принадлежности. Заметим, что степень принадлежности может являться расплывчатым множеством.

Основное положение теории расплывчатых множеств состоит в том, что A, несмотря на «размытость» границ, задается точно путем сопоставления каждому элементу $x \in X$, числа, лежащего между 0 и 1.

Например, пусть $X = \{1, 3, 5, ...\}$ — множество положительных нечетных чисел. Тогда расплывчатое множество A можно, например, определить как набор кортежей вида $A = \{\langle 0{,}6; 1\rangle, \langle 0{,}2; 3\rangle, \langle 0{,}7; 5\rangle ...\}$.

Носитель A есть множество элементов $x \in X$, для которого $\mu_A(x)$ положительно. Точкой перехода A называется элемент $x \in X$, для которого $\mu_A(x) = 0{,}5$. Одноточечным *расплывчатым множеством* (иногда используют термин нечеткие или размытые множества) называется множество, носитель которого состоит из единственной точки. Если $\tilde{A}$ — одноточечное расплывчатое множество, то его обозначают $\tilde{A} = \mu/x$, где μ — степень принадлежности x к A. Тогда одноточечное множество можно обозначить $A = 1/x$.

Расплывчатое множество часто рассматривают в виде объединения входящих в него одноточечных множеств и записывают в виде

$$\tilde{A} = \int_x \mu_A(x)/x, \quad x \in X.$$

Здесь знак интегрирования означает объединение одноточечных расплывчатых множеств $\mu_A(x)/x$. Если $\tilde{A}$ состоит из конечного числа элементов, то

$$\tilde{A} = \mu_1/x_1 \cup \mu_2/x_2 \cup \ldots \cup \mu_n/x_n = \bigcup_{i=1}^{n} \mu_i/x_i.$$

Очевидно, что тогда конечное множество $X = \{x_1, x_2, ..., x_n\}$ можно записать в виде

$$X = 1/x_1 \cup 1/x_2 \cup \ldots \cup 1/x_n = \bigcup_{i=1}^{n} 1/x_i.$$

Пусть $X = \{$кристалл, плата, панель$\}$; $\tilde{A}$ — расплывчатое множество, определяемое признаком «степень интеграции (СИ) элементов». Тогда, например, можно записать: $\tilde{A} =$ СИ большая/кристалл $\cup$ СИ средняя/плата $\cup$ СИ малая/ панель. Расплывчатые степени принадлежности большая, средняя, малая в данном примере можно определить как расплывчатые подмножества множества $W = \{0; 0{,}1; 0{,}2; ...; 1\}$. Эти подмножества можно определить, например, так: малая $= 0{,}5/0{,}2 \cup 0{,}7/0{,}3 \cup 1/0{,}4$; средняя $= 0{,}6/0{,}4 \cup 0{,}8/0{,}5 \cup 1/0{,}6$; большая $= 0{,}7/0{,}7 \cup 0{,}8/0{,}8 \cup 1/1$. Приведем еще ряд примеров расплывчатых множеств.

Пусть $X=\{1, 2, ..., 20\}$, тогда расплывчатое множество $\tilde{A}$ на X, описываемое понятием «много», можно, например, записать в виде «много» = 0,5/10 ∪ 0,6/11 ∪ 0,7/12 ∪ 0,8/13 ∪ 0,9/14 ∪ 0,9/15 ∪ 0,9/16 ∪ ∪ 0,9/17 ∪ 0,9/18 ∪ 1/19 ∪ 1/20.

Пусть X — множество ЭВМ первого, второго, третьего и четвертого поколений, запишем это так: $X=\{$ЭВМ1, ЭВМ2, ЭВМ3, ЭВМ4$\}$. Пусть $\tilde{A}$ — расплывчатое множество, определяемое термином «эффективный» в смысле времени решения задач на ЭВМ. Тогда условно можно, например, записать $\tilde{A}$ = эффективные очень слабо/ЭВМ1 ∪ эффективные слабо/ЭВМ2 ∪ эффективные средне/ЭВМ3 ∪ ∪ эффективные достаточно/ЭВМ4. Здесь, как и в ранее рассмотренных примерах, термины «очень слабо», «слабо», «средне», «достаточно» можно определить с помощью расплывчатых множеств:

$$W=\{1, 2, 3, 4\};\ \text{«очень слабо»}=1/1\cup 0{,}7/2\cup 0{,}1/3\cup 0/4;$$

$$\text{«слабо»}=0{,}7/1\cup 1/2\cup 0{,}2/3\cup 0/4;$$

$$\text{«достаточно»}=0/1\cup 0{,}3/2\cup 0{,}7/3\cup 1/4.$$

В практических задачах конструирования функция принадлежности μ_A определяется эвристически из конкретных условий. Расплывчатые множества $\tilde{A}$ и $\tilde{B}$ называются равными, если $(\forall x\in X)$ $(\mu_A(x)=\mu_B(x))$.

Расплывчатое множество $\tilde{A}$ является подмножеством $\tilde{B}(\tilde{A}\subset\tilde{B})$, если $(\forall x\in X)\,(\mu_A(x)\leqslant\mu_B(x))$. Расплывчатое множество $\tilde{A}^1$ называется дополнением к $\tilde{A}$, если $(\forall x\in X)\,(\mu_A(x)=1-\mu_A(x))$. Например, расплывчатые множества $\tilde{A}=\{$«большие» микросхемы$\}$ и $\tilde{A}^1=$ $=\{$«небольшие» микросхемы$\}$ являются дополнениями друг к другу, если отрицание «не» определяется как операция, заменяющая $\mu_A(x)$ на $1-\mu_A(x)$, $\forall x\in X$.

Говорят, что расплывчатое множество $\tilde{C}$ называется пересечением множеств $\tilde{A}$ и $\tilde{B}$, если оно определяется как наибольшее расплывчатое множество, содержащееся как в $\tilde{A}$, так и в $\tilde{B}$: $\tilde{C}=\tilde{A}\cap\tilde{B}$; $\mu_{A\cap B}(x)=\min(\mu_A(x),\ \mu_B(x))$, $x\in X$.

Например, если $\tilde{A}=\{\langle 0{,}7/\text{ИМС3}\rangle, \langle 0{,}3/\text{ИМС5}\rangle, \langle 0{,}9/\text{ИМС1}\rangle\}$, $\tilde{B}=$ $=\{\langle 0{,}3/\text{ИМС3}\rangle, \langle 0{,}2/\text{ИМС5}\rangle, \langle 1/\text{ИМС1}\rangle\}$, тогда $\tilde{C}=\tilde{A}\cap\tilde{B}=$ $=\{\langle 0{,}3/\text{ИМС3}\rangle, \langle 0{,}2/\text{ИМС5}\rangle, \langle 0{,}9/\text{ИМС1}\rangle\}$.

Если $\tilde{A}=\{\langle 0{,}8/1\rangle, \langle 0{,}5/3\rangle\}$, а $\tilde{B}=\{\langle 0{,}2/1\rangle, \langle 0{,}7/3\rangle\}$, тогда $\tilde{C}=$ $=\tilde{A}\cap\tilde{B}=\{\langle 0{,}2/1\rangle, \langle 0{,}5/3\rangle\}$.

Объединением расплывчатых множеств $\tilde{A}$ и $\tilde{B}$ называется наименьшее расплывчатое множество $\tilde{C}$, содержащееся как в $\tilde{A}$, так и в $\tilde{B}$: $\tilde{C}=\tilde{A}\cup\tilde{B}$; $\mu_{A\cup B}(x)=\max(\mu_A(x))$, $x\in X$.

Пусть $\tilde{A}$ — множество ИМС3, а $\tilde{B}$ — множество ИМС1, тогда $\tilde{C}=\tilde{A}\cup\tilde{B}=\{$ИМС3 или ИМС1$\}$.

Расплывчатым отношением φ на декартовом произведении множеств $X\times Y=\{\langle x, y\rangle, x\in X, y\in Y\}$ называется выражение, описываемое функцией принадлежности μ_φ, которая сопоставляет каждому кортежу $\langle x, y\rangle$ его степень принадлежности $\mu_\varphi(\langle x, y\rangle)$ к φ.

Пусть $X=\{$РЭА, ЭВА$\}$, $Y=\{$ИМС3, ИМС1$\}$. Тогда бинарное расплывчатое отношение «компоновка» между элементами мно-

жеств X и Y можно, например, записать следующим образом: *Компоновка* = ⟨0,7/(РЭА, ИМС3)⟩ ∪ ⟨0,9/(РЭА, ИМС1)⟩ ∪ ∪⟨0,8/(ЭВА, ИМС1)⟩∪⟨0,1/(ЭВА, ИМС3)⟩. Расплывчатые отношения удобно задавать с помощью матрицы отношений $\mathbf{R}_\varphi = \|r_{i,j}\|_{|X|\times|Y|}$. В нашем случае матрица отношений запишется в виде

$$\mathbf{R}_\varphi = \begin{array}{c} \\ \text{РЭА} \\ \text{ЭВА} \end{array} \begin{array}{c} \begin{array}{cc} \text{ИМС3} & \text{ИМС1} \end{array} \\ \left\| \begin{array}{cc} 0{,}7 & 0{,}9 \\ 0{,}1 & 0{,}8 \end{array} \right\| \end{array},$$

где элемент $r_{i,j}$ равен значению функции μ_φ для i-го элемента x и j-го элемента y. Заметим, что для расплывчатых отношений справедливы операции, определенные на расплывчатых множествах. Для расплывчатых отношений можно определить такие свойства, как рефлексивность, симметричность, транзитивность и т. д.

Приведенные определения и понятия можно применять для нахождения решения в условиях, когда неточность является следствием расплывчатости, что характерно для некоторых эвристических задач автоматизации конструирования ЭА.

2.2. ЭЛЕМЕНТЫ ТЕОРИИ АЛГОРИТМОВ

Понятие алгоритма является одним из основополагающих понятий современной науки и техники. Неформально можно определить алгоритм как конечную совокупность точно заданных правил решения произвольного класса задач.

Отметим основные требования к алгоритмам. Алгоритмы применяются к исходным данным и выдают конечные результаты, т. е. каждый алгоритм имеет входы и выходы. В результате работы алгоритма появляются промежуточные результаты. Следовательно, алгоритм имеет дело с данными, которые можно отнести к трем группам: входные, выходные, промежуточные. Для размещения данных, с которыми имеют дело алгоритмы, требуется память. Обычно считают, что память состоит из одинаковых ячеек, причем каждая из них может содержать один или несколько элементов данных, называемых символами.

Алгоритм состоит из отдельных конечных действий, которые называются «шагами». Примерами шагов могут служить операции сложения, умножения, сдвига информации в ЭВМ и т. п. В используемых в настоящее время алгоритмах последовательность шагов обычно детерминирована, т. е. после каждого шага алгоритма определяется номер следующего шага. Детерминированный алгоритм при одних и тех же входных данных выдает одинаковые результаты, а при разных входных данных может менять порядок выполнения шагов. От каждого алгоритма требуется остановка после конечного числа шагов. Для практики представляют интерес алгоритмы, которые завершаются после конечного числа шагов.

Обычно различают описание алгоритмов, механизм реализации алгоритмов (в основном для этих целей используется ЭВМ) и про-

цесс реализации алгоритмов. Алгоритмы реализуются на ЭВМ в виде программ.

Существуют три основных типа универсальных алгоритмических моделей. В первом понятие алгоритма связывается с вычислимыми и числовыми функциями. Во втором алгоритм представляется как некоторое детерминированное устройство, способное выполнять в отдельные моменты времени простейшие операции. В третьем типе — преобразование слов в произвольных алфавитах. Эти типы близки друг к другу и отличаются эвристическими описаниями определений алгоритма.

Одним из эвристических определений алгоритма считают конструктивно задаваемые соответствия между словами в абстрактных алфавитах.

Под абстрактным алфавитом обычно понимается конечная совокупность объектов, называемых буквами или символами этого алфавита. В качестве символов абстрактных алфавитов могут использоваться буквы, цифры, знаки и даже слова, в частности слова русского языка. Следовательно, абстрактный алфавит — это конечное множество различных символов, и его можно задать путем перечисления элементов.

Примерами алфавитов можно считать:

$$A = \{\text{схема, ИМС3, 0, 1, 2, } a, \beta, \text{ кристалл}\},$$

$$B = \{—, |, \times\},\ C = \{+, —, \varnothing,$$

$$\text{плата, РЭА}\}$$

и т. д.

Под словом в алфавите понимают любую конечную упорядоченную последовательность символов. Например, выражение «+, плата, РЭА» является словом в абстрактном алфавите C. Видно, что термин «слово» является синонимом термину «кортеж» в теории множеств. Число символов в слове называется длиной этого слова. Пустое слово обозначается так же, как и пустое множество $\varnothing$.

Алфавиты бывают входные, выходные и промежуточные.

Алфавитным оператором называется функция, которая задает соответствие между словами входного и выходного алфавитов. Например, пусть заданы алфавиты X и Y: $X = \{a, b, \text{ИМС3}, +, \text{ИМС1}\}$, $Y = \{c, d, \text{кристалл, ТЭЗ}, e, f\}$.

Пусть также заданы слова в алфавитах X и Y: «a, b», «b, ИМС3, +», «ИМС3, ИМС1, a», «c, d, ТЭЗ», «кристалл, e, f», «d, e, f» и «d, ТЭЗ». Соответствия между этими словами показаны на рис. 2.13.

Если x — слово в алфавите X, а y — слово в алфавите Y, то алфавитный оператор $\Delta x = y$ перерабатывает входное слово x в выходное слово y.

Алфавитные операторы бывают однозначными и многозначными. В однозначном алфавитном операторе каждому входному слову ставится в соответствие не более одного выходного слова. В многозначном алфавитном операторе каждому входному слову

может быть поставлено в соответствие некоторое множество выходных слов. Алфавитный оператор, который не ставит в соответствие входному слову никакого выходного слова (включая пустое), не определен на этом слове. Алфавитный оператор, изображенный на рис. 2.13, является многозначным, так как входному слову «*a*, *b*» поставлены в соответствие слова «*c*, *d*, ТЭЗ», «*d*, *e*, *f*».

Множество слов, на которых алфавитный оператор определен, называется его *областью определения*. Если область определения

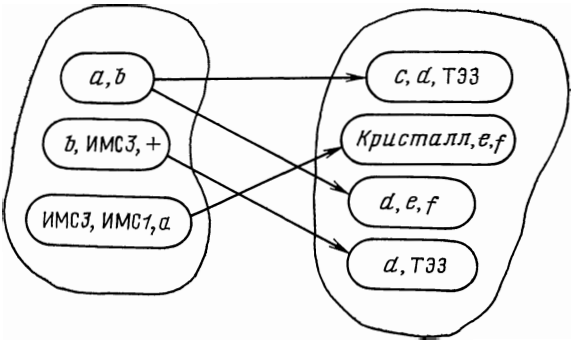


Рис. 2.13. Пример алфавитного оператора

конечна, то алфавитный оператор может быть представлен таблицей соответствия, в левой части которой выписываются слова, входящие в область определения, в правой части — слова, получающиеся в результате применения алфавитного оператора к входным словам. Если область определения алфавитного оператора бесконечна, то он задается набором правил, которые за конечное число шагов определяют выходное слово, соответствующее словам, на которых он определен. Говорят, что алфавитные операторы (АО), задаваемые с помощью конечной системы правил, называются *алгоритмами*.

Следует различать понятия алгоритма и алфавитного оператора. Основное в алфавитном операторе — это соответствие между входными и выходными словами. Основное в алгоритме — это способ установления соответствия.

Алгоритмы равны, если равны соответствующие им АО и совпадают системы правил, задающих действия этих алгоритмов на входные слова. *Алгоритмы эквивалентны*, если совпадают их алфавитные операторы, но не обязательно совпадают способы задания операторов. *Алгоритмы называются детерминированными*, если им соответствуют однозначные АО и любому входному слову соответствует одно выходное слово.

Основными свойствами алгоритмов являются массовость и результативность. *Массовость алгоритма* — это его свойство быть применимым для множества входных данных. *Результативность алгоритма* — это свойство, обеспечивающее получение результата через конечное число шагов.

Алгоритм называется случайным, если в системе правил предусматривается возможность случайного выбора слов или правил. *Алгоритм называют самоизменяющимся*, если он не только перерабатывает входные слова, но и сам изменяется в процессе такой переработки.

Из определений и описаний алгоритма следует, что он ставит в соответствие входным данным выходные. Следовательно, с каждым детерминированным алгоритмом можно однозначно сопоставить функцию, которую он вычисляет. Исследования теории алгоритмов привели к построению и анализу функций, для которых алгоритмы существуют.

Процедуру, которая прямо или косвенно обращается к себе, называют *рекурсивной*. Рекурсией называют способ задания функции, когда значение определяемой функции для произвольных значений аргумента выражается известным образом через значения определяемой функции для других значений аргументов.

Численные функции, значения которых можно установить на основе некоторого алгоритма, называются *вычислимыми функциями*. Совокупность численных функций, совпадающих с совокупностью всех вычислимых функций, называется *совокупностью рекурсивных функций*.

Существует гипотеза о том, что класс рекурсивных функций тождествен классу всюду определенных вычислимых функций.

Использование рекурсивных функций в теории алгоритмов основано на идеях нумерации слов в произвольном алфавите последовательными натуральными числами. После нумерации слов в произвольном АО он превращается в функцию $y=f(x)$, где аргумент x и функция y принимают неотрицательные целочисленные значения. Такие функции называют арифметическими функциями.

Выделим элементарные арифметические функции: тождественно равные нулю; тождественные функции, повторяющие значения своих аргументов; функции непосредственного следования типа $f(x)=x+1$.

На основе перечисленных функций с помощью конструктивных приемов можно строить все более и более сложные арифметические функции.

Для решения той или иной задачи ее часто разбивают на части, определяют их решения, а затем из них получают решение всей задачи. Этот прием, если применять его рекурсивно, в основном приводит к эффективному решению задачи, подзадачи которой представляют собой менее сложные варианты.

Концепции теории алгоритмов были впервые описаны в математических терминах в 30-е годы, когда велись поиски условий, с помощью которых доказательство математических теорем может генерироваться автоматически. Британский математик А. М. Тьюринг был одним из пионеров в развитии точной формулировки такого процесса. Он описал модель машины, названной его именем, которая позже была построена реально. Машина Тьюринга состоит из бесконечной в обе стороны ленты с отмеченными

квадратами и рабочей (считывающей) головки. Лента способна к автоматическому перемещению. Машина может выполнять только четыре действия:

движение ленты на один шаг (квадрат);

размещение метки в квадрате;

стирание метки, представленной в квадрате;

конец вычисления.

Условная схематическая модель машины показана на рис. 2.14,*а*. Она состоит из информационной ленты, представляющей

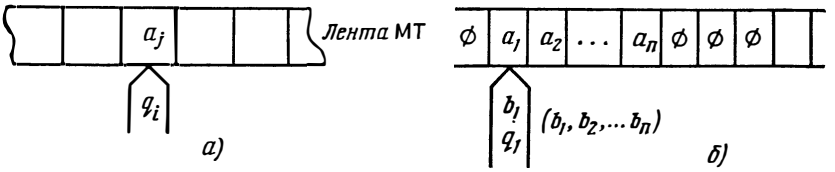


Рис. 2.14. Условная схема машины Тьюринга (*а*), пример ее работы (*б*)

собой бесконечную память машины. В качестве информационной ленты можно использовать магнитную или бумажную ленту, разделенную на квадраты (ячейки). В каждой ячейке можно поместить один любой символ конечного алфавита $A=\{a_0, a_1, a_2, ..., a_n\}$. Считывающая головка способна просматривать содержимое ячеек. Вдоль нее лента перемещается в обе стороны так, чтобы в любой рассматриваемый момент времени головка находилась в одной ячейке ленты. Считывающая головка машины Тьюринга рассматривается как устройство управления, которое в данный момент времени находится в одном из состояний $Q=\{q_1, q_2, ..., q_n\}$. Состояние устройства управления называют внутренним состоянием машины Тьюринга. Одно из состояний является заключительным и управляет окончанием работы. Машина Тьюринга работает в дискретные моменты времени $t=0, 1, 2, ...$ В каждый момент времени в одной ячейке ленты может быть записана одна буква из алфавита A. В этот же момент времени считывающая головка в зависимости от содержимого ячейки и своего внутреннего состояния $q_i \in Q$ может: заменить символ в ячейке на новый или оставить его прежним, перейти в новое состояние или остаться в прежнем, сдвинуть ленту на один шаг вправо, влево или оставить ее неподвижной. Следовательно, машина Тьюринга состоит из внутренней памяти (конечное множество состояний) и внешней памяти (лента). Данные машины Тьюринга — это слова в алфавите ленты. На ленте записывают исходные данные и конечные результаты. Последовательность шагов в машине Тьюринга (детерминированность) определяют путем задания трех условий. Для любого q_i, a_j должны быть однозначно заданы:

новое состояние q'_i;

новый символ a'_j, который необходимо записать вместо a_j в ту же клетку машины;

направление сдвига считывающей головки (влево — L, вправо — R, E — на месте).

Последовательности шагов можно описать таблицей, в строках которой записаны состояния q_i, в столбцах — символы a_j, а на пересечении строки q_i и столбца a_j ставится $q'_i a'_j s_k$, где s_k — сдвиг головки машины Тьюринга. Задание можно также описать диаграммой либо выражением $q_i a_j \rightarrow q'_i a'_j s_k$.

Работа машины Тьюринга состоит в изменении конфигураций. *Конфигурацией* машины Тьюринга называется ее полное состояние, по которому можно однозначно определить дальнейшее поведение машины. Оно обозначается тройкой $\alpha_1 q_i \alpha_2$, где α_1 — слово слева от головки, α_2 — слово, состоящее из символа, просматриваемого головкой и расположенного справа. Стандартной начальной конфигурацией называется конфигурация вида $q_1\alpha$, где просматривается крайний слева символ, записанный на ленте. Стандартной заключительной конфигурацией называется выражение вида $q_2\alpha$. Ко всякой произвольной конфигурации k в машине Тьюринга применима только одна команда типа $q_i a_j \rightarrow q'_i a'_j s_k$, которая переводит k в k'.

Совокупность всех команд, которые может выполнять машина Тьюринга, называется *программой*. При решении задач со входными данными сопоставляется начальная конфигурация k_0, а выходные данные определяются заключительной конфигурацией, в которую машина Тьюринга переводит k_0.

Например, рассмотрим машину Тьюринга, переводящую последовательность $a_1, a_2, \ldots, a_n$ в последовательность $b_1, b_2, \ldots, b_n$, работающую в конфигурации, показанной на рис. 2.14,*б* в соответствии с программой $q_i a_j \rightarrow q'_i b_j s_k$. Считывающая головка, перемещая ленту направо в соответствии с программой, будет стирать символы $a_1, a_2, \ldots, a_n$ и вместо них соответственно записывать выходные символы $b_1, b_2, \ldots, b_n$. Если потребовать, чтобы при просмотре клетки ленты с пустым символом машина Тьюринга переходила в заключительное состояние, то после остановки машины на ленте будет выходная последовательность $b_1, b_2, \ldots, b_n$.

Было показано, что на машине Тьюринга можно имитировать все алгоритмические процессы, которые когда-либо описывались математически.

Тьюринг показал, что если проблемы не могут быть решены на его машине, то они не могут быть решены ни на какой другой автоматической ЭВМ, т. е. это проблемы, для которых алгоритмы не могут быть составлены даже в принципе. Следовательно, невозможность построения машины Тьюринга означает отсутствие алгоритма решения данной проблемы. Следует отметить, что речь идет об отсутствии алгоритма, решающего всю данную проблему, что не исключает возможности решения этой проблемы в частных случаях различными для каждого случая методами.

Один из основных результатов Тьюринга — разделение всех представляемых в математике проблем на два класса:

проблемы, для которых алгоритмы никогда не могут быть написаны, т. е. в формальном смысле постоянно нерешаемые;

проблемы, которые могут быть решены с помощью алгоритмов.

Класс решаемых проблем может быть разделен на два подкласса:

подкласс, содержащий только полиномиальные алгоритмы;

подкласс, содержащий только экспоненциальные алгоритмы.

Функции типа 2^n, n^n, $n!$.. могут рассматриваться, как имеющие одинаковое свойство экспоненциального роста. Функции типа kn, kn^2, kn^3, ..., где k — коэффициент, могут рассматриваться как полиномиальные. Основное отличие полиномиальной функции от экспоненциальной состоит в том, что n никогда не появляется в экспоненте. Для достаточно больших n значения экспоненциальной функции догоняют и превышают значения любой полиномиальной функции. Иногда для достаточно малых n значения полиномиальной функции могут превышать значения экспоненциальной функции, но всегда существует величина n, начиная с которой значения экспоненциальной функции будут превышать значения полиномиальной функции.

Оценку максимального числа элементарных операций, выполняемых при работе алгоритма, называют его эффективностью. Ее записывают в виде $O(f(n))$. По определению $O(f(n))$ — это множество всех функций $y(n)$, для которых существует положительная постоянная c и такой номер n_0, что $|q(n)| \leqslant c|f(n)|$ для всех $n > n_0$.

Тогда полиномиальные алгоритмы могут иметь оценки эффективности $O(n)$, $O(n^2)$, $O(n \log n)$ и т. п. Экспоненциальные алгоритмы имеют оценки эффективности типа $O(2^n)$ или $O(n!)$. Оценки эффективности (иногда их называют оценками сложности) зависят от параметра n, задающего длину исходных данных. Например, при конструировании электронной аппаратуры в качестве исходных данных может рассматриваться число элементов схемы, цепей схемы и т. п.

Сложность алгоритма характеризуется в основном числом операций N и общим объемом памяти V, необходимой для его реализации. С числом операций алгоритма связано время его выполнения на конкретной ЭВМ. Следовательно, время реализации алгоритма есть функция $T = f(N)$. Отметим, что выполнение различных операций алгоритма требует различного времени. В этой связи для определения сложности алгоритма необходимо знать не только общее число операций, но и число используемых в алгоритме типов операций и число операций каждого типа в отдельности. На основании анализа алгоритма можно построить множество встречаемости операций в алгоритме $N = \{Q_1, Q_2, ..., Q_n\}$, где Q_i — число операций i-го типа; n — число типов операций в исследуемом классе задач.

Умножая время выполнения каждой операции на число этих операций и суммируя результаты по всем операциям, получаем

время реализации алгоритма $T=\sum_{1=i}^{n} Q_i t_i$, где t_i — время реализации i-й операции.

Если принять за эталон произвольную простейшую операцию, то можно определить общее число приведенных к ней операций. Для этого надо знать относительные веса различных операций по отношению к эталонной: $B_i=t_i/t_э$, где $t_э$ — время реализации эталонной операции.

Тогда общее число условных элементарных операций составит

$$N_э=\sum_{i=1}^{n} B_i Q_i,$$

а общее время их реализации $T_э=N_э t_э$.

Объем памяти, которую необходимо зарезервировать в ЭВМ для реализации алгоритма, определяется как суммарный объем, необходимый для размещения программы, а также входной, промежуточной и выходной информации: $V=V_п+V_и+V_{пр}+V_в$, где $V_п$ — объем памяти для размещения программы; $V_и$ — объем памяти для размещения исходной информации $V_{пр}$ — объем памяти для размещения промежуточной информации; $V_в$ — объем памяти для размещения выходной информации; V — общий объем памяти.

Тогда обобщенный коэффициент сложности алгоритма $K_{сл}=N_э/V$. Он характеризуется числом эталонных операций, выполняемых при использовании памяти V. По данной методике можно определять сложность алгоритма, если известно множество N.

Отметим, что, так как время выполнения операций различно, требуется различная память для размещения программы входных, промежуточных и выходных данных. Иногда требуется много оперативной памяти (ОП) ЭВМ. Иногда ее без ущерба можно заменить другой памятью, например памятью прямого доступа (диски). В некоторых случаях распределением памяти занимается операционная система (ОС) ЭВМ и пользователь (конструктор) не знает, какую память он использует. В этой связи приведенная оценка памяти является приближенной.

В настоящее время наиболее перспективными методами оценки сложности алгоритмов при автоматизации конструирования ЭА являются методы, позволяющие оценить алгоритм до его представления на алгоритмическом языке. Показано, что в практических задачах (особенно это относится к проектированию и конструированию ЭА) алгоритмы, в которых время решения растет экспоненциально как функция размера входов, являются неэффективными. Эффективными считаются полиномиальные алгоритмы. Конечно, даже среди эффективных алгоритмов некоторые реализуются быстрее, чем другие. Но эта классификация делает скорость решения алгоритма свойствами самого алгоритма независимо от ЭВМ, на которой он реализуется.

Алгоритмы, которые используются на практике, можно условно разбить на два класса: детерминированные и недетерминированные. Как рассматривалось выше, детерминированные алгоритмы

состоят из операций, результат каждой из которых определен однозначно. Если алгоритм содержит операции, результат выполнения которых неоднозначен, он относится к классу недетерминированных алгоритмов.

Задачей на допустимость считается задача определения хотя бы одного возможного решения. Задачей на оптимальность считается задача определения допустимого решения, дающего экстремум целевой функции рассматриваемой системы. Множество всех задач на допустимость, которые могут быть решены на основе детерминированного алгоритма за полиномиальное время, относится к подклассу полиномиальных алгоритмов и обозначается P.

Подклассом NP называется множество всех задач на допустимость, которые могут быть решены с помощью недетерминированного алгоритма также за полиномиальное время. В подкласс NP входят проблемы, всегда решаемые в принципе, но для которых сейчас известны только детерминированные алгоритмы с экспоненциальным временем решения. Так как детерминированные алгоритмы являются частным случаем недетерминированных алгоритмов, то $P \subseteq NP$.

При рассмотрении практических алгоритмов конструирования часто встречается следующее соотношение: если некоторая проблема может быть решена эффективно, то и другие проблемы, сводимые к ней, могут быть решены эффективно. Пусть имеются две задачи конструирования ЭА, например компоновка K и трассировка T. Говорят, что задача K сводится к задаче T, если задачу K можно решить за полиномиальное время на основе полиномиального детерминированного алгоритма, используемого для решения задачи T.

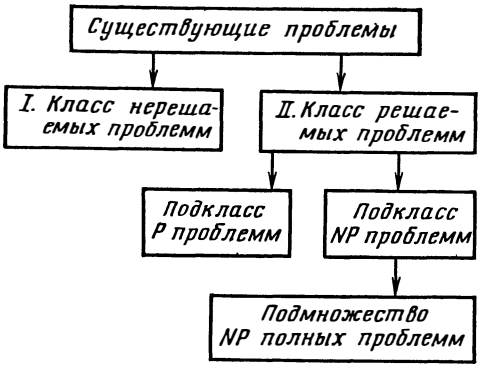


Рис. 2.15. Классификация существующих проблем

В последние годы было показано, что некоторое подмножество проблем в классе NP имеет замечательное свойство: все проблемы в NP эффективно сводятся к этому подмножеству. Это подмножество проблем в NP называется NP-полным и обозначается NPC, причем $NPC \subseteq NP$. Тогда если некоторая из проблем в NPC будет иметь эффективный алгоритм, то и каждая проблема в NP может быть решена эффективно. Заметим, что определение такого алгоритма будет доказательством того, что $P = NP$.

На рис. 2.15 показана одна из возможных классификаций существующих проблем. Примеры построения, разработки и описа-

ния алгоритмов P, NP и NPC будут рассмотрены в дальнейших разделах книги.

Рассмотрение практических алгоритмов показывает, что все операции, выполняемые в алгоритмах, распадаются на группу арифметических и группу логических операторов. Арифметические операторы выполняют непосредственное преобразование информации, а логические операторы определяют последовательность выполнения арифметических. В этой связи необходимо использование более универсальных логических операторов. Этим требованиям удовлетворяют рассматриваемые ниже способы записи алгоритмов по Ван Хао и Ляпунову.

Операторный алгоритм Ван Хао задается последовательностью приказов специального вида. Каждый приказ имеет определенный номер и указания, какую операцию необходимо выполнять над заданным объектом, и приказ, с каким номером следует далее выполнить действие над результатом данной операции.

В общем виде приказ запишется так:

$$i: \boxed{\omega \mid \alpha \mid \beta},$$

где i — номер приказа; ω — элементарная операция над объектом; α, β — номера некоторых приказов.

Выполнить приказ i над числом x в операторном алгоритме значит найти число $\omega(x)$ и затем перейти к выполнению последнего приказа с номером α. Если $\omega(x)$ не определено, то перейти к выполнению над числом x приказа с номером β. Заключительному состоянию в алгоритме Ван Хао соответствует приказ

$$i: \boxed{\text{стоп}}.$$

Это означает, что вычисления необходимо остановить. Число, над которым необходимо выполнить этот приказ, есть результат вычисления.

Для описания алгоритмов в форме Ляпунова используют *логические схемы алгоритмов* (ЛСА). В ЛСА операторами называются действия алгоритма, перерабатывающие информацию. Их обозначают заглавными буквами, например A, B, C, ... Малыми буквами обозначаются проверяемые логические условия. Последовательное выполнение нескольких операторов обозначается в виде произведения, причем сомножитель, стоящий справа, действует после сомножителя, стоящего слева.

Логическими схемами алгоритмов называются выражения, составленные из операторов и логических условий, следующих один за другим. От каждого логического условия начинается стрелка, которая оканчивается у какого-либо из операторов. Например, $A\downarrow^1 q\uparrow^1 B\downarrow^2 p\uparrow^2 C$. Здесь знак $\uparrow^i$ обозначает начало стрелки, а знак $\downarrow^i$ — ее конец. Одинаковыми номерами отмечаются начало и конец одной и той же стрелки. Работа алгоритма начинается с выполнения самого левого члена. После того как некоторый член

ЛСА выполнится, определяется следующий. Если это был оператор, то за ним выполняется тот член ЛСА, который стоит непосредственно справа от него. Если последний выполнившийся член ЛСА был логическим условием, то возможны два случая. Если проверявшееся условие выполнено, то должен выполняться член, находящийся справа. Если оно нарушено, должен выполняться тот член, к которому ведет стрелка, начинающаяся после данного условия. Работа алгоритма оканчивается, когда последний из выполняющихся операторов содержит указание о прекращении работы либо на некотором этапе не оказывается члена, который должен был бы выполняться. В ЛСА обычно выделяют операторы A_0 и $A_к$, символизирующие начало и конец работы алгоритма соответственно.

Например, в ЛСА $A_0\downarrow^2 A p_1\uparrow^1 B\downarrow^1 p_2\uparrow^2 C A_к$ получим следующий порядок выполнения операторов:

если $p_1=0$ и $p_2=0$, то $AA...A...$, т. е. бесконечно будет выполняться оператор A;

если $p_1=0$, а $p_2=1$, то AC;

если $p_1=1$, а $p_2=0$, то $ABAB...AB...$;

если $p_1=1$ и $p_2=1$, то последовательно выполняются все три оператора, т. е. ABC и алгоритм заканчивает работу.

В общем случае логические схемы алгоритмов Ляпунова удовлетворяют следующим условиям:

1) содержат один начальный и один конечный оператор, соответственно A_0, $A_к$ (перед A_0 и после $A_к$ стрелки не ставятся);

2) за каждым логическим условием всегда ставится стрелка, направленная вверх;

3) не содержат одинаково помеченных стрелок, направленных вниз;

4) для каждой верхней (нижней) стрелки должна быть точно одна нижняя (верхняя) стрелка.

Пусть алгоритм задан в виде ЛСА следующего вида: $A_0 p_1\uparrow^1 A_1\downarrow^1 p_2\uparrow^2 A_2 A_3\downarrow^2 A_4 A_к$. В данной ЛСА четыре оператора A_1—A_4, два логических условия p_1, p_2 и операторы начала A_0 и конца $A_к$. Если в начальном состоянии на вход p_1 устройства придет сигнал, равный $p_1=1$, то произойдет переход устройства в новое состояние, в котором выполняется оператор A_1. Если $p_1=0$, то, пропуская $\uparrow^1 A_1\downarrow_1$, в ЛСА выходим на логическое условие p_2 со стрелкой $\uparrow^2$, т. е. если $p_1=0$ и $p_2=1$, то устройство A_0 переходит в новое состояние с выдачей оператора A_2, а если $p_1=0$ и $p_2=0$, то выдается оператор A_4. После выполнения A_2 независимо от значения логических условий выполняется стоящий справа от него оператор A_3, затем A_4.

Для графической записи ЛСА можно использовать язык граф-схем алгоритмов.

Граф-схемы алгоритмов (ГСА) удовлетворяют следующим условиям:

имеют конечное число вершин;

имеют одну начальную и одну конечную вершины;

входы и выходы вершин соединяются друг с другом с помощью стрелок, направленных от входа к выходу;

выход соединяется только с одним входом, вход соединяется, по крайней мере, с одним выходом.

На рис. 2.16,*а* показаны вершины граф-схемы алгоритмов: в условных вершинах записываются элементы из множества логических условий, в операторной вершине — оператор. Для рассмотренного выше примера записи устройства с помощью логической схемы алгоритмов на рис. 2.16,*б* показана граф-схема.

Заметим, что один и тот же алгоритм может быть представлен как двумя описанными способами, так и множеством других, а также на любом алгоритмическом языке.

Рассмотрим *структурные схемы алгоритмов*. При такой записи алгоритмов процесс решения задачи расчленяется на отдельные этапы — блоки. Блоки изображаются графически в виде геометрических фигур. Блоку присваивается номер, и внутри него указывается информация, характеризующая выполняемые ими функции, которые записываются в виде формул или словесно. Блоки соединяются стрелками, показывающими связи между ними. Если один блок передает управление другим блокам в зависимости от выполнения условий, то на стрелках указывается условие, при котором процесс разветвляется. Блоки бывают двух видов: операторы, т. е. блоки, из которых выходит одна стрелка, и логические условия, т. е. блоки, из которых выходит несколько стрелок. Кроме этого, имеются блок вывода, из которого не выходит ни одной стрелки, и блок ввода, в который не входит ни одна стрелка. При такой записи допускаются комментарии, соединители, распараллеливание вычислительных процессов и другие операции. Блоки могут описывать элементарные шаги алгоритма и представлять собой части алгоритмов или отдельные алгоритмы.

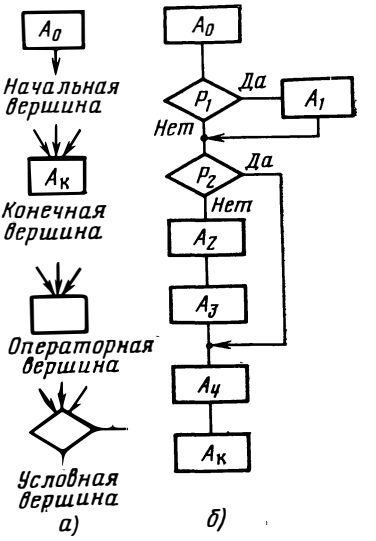


Рис. 2.16. Вершины граф-схемы алгоритма (*а*), пример граф-схемы алгоритма (*б*)

В практических задачах конструирования на интегральных микросхемах сложный алгоритм разбивается на блоки, которые разрабатывают разные конструкторы, причем разные блоки могут разрабатываться независимо друг от друга. Аналогично с помощью структурных схем можно несколько алгоритмов, рассматриваемых как блоки, объединять в один большой алгоритм. Операция разбиения алгоритма на блоки называется *декомпозицией*, а операция соединения алгоритмов — *композицией*.

Структурные схемы позволяют различать описание алгоритмов и процесс их реализации. Описание алгоритма — это сама структурная схема, а процесс реализации алгоритма — это процесс выполнения последовательности операций, задаваемых структурной схемой.

Рассмотрим пример составления структурной схемы алгоритма.

Пусть надо вычислить 50 значений функции

$$y_i = \sin(\alpha x_i)/x_i \quad (x_i = 1, 2, \ldots, 50).$$

Структурная схема алгоритма показана на рис. 2.17,*а*.

Поясним блоки *2*, *5*, *6*. Блок *2* задает начальное значение аргумента. Блок *5* изменяет значение аргумента x_i на единицу после каждого выполнения цикла. Блок *6* управляет циклом, для чего проверяется условие повторения цикла

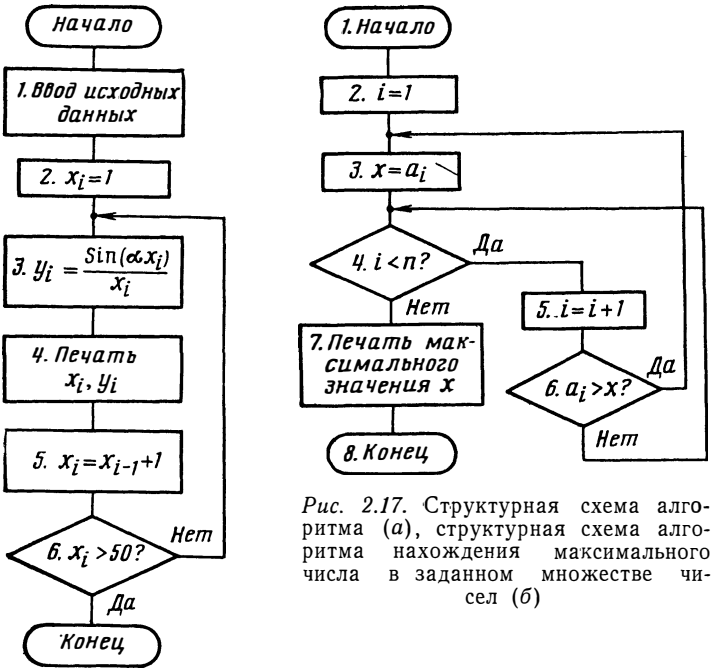


Рис. 2.17. Структурная схема алгоритма (*а*), структурная схема алгоритма нахождения максимального числа в заданном множестве чисел (*б*)

$x_i > 50$. При невыполнении этого условия цикл должен повторяться, а при выполнении осуществляется переход к следующему по порядку блоку.

Построим структурную схему алгоритма нахождения максимального числа в заданном множестве числе. Пусть $A = \{a_1, a_2, a_3, a_4\}$, $a_1=1$, $a_2=2$, $a_3=8$, $a_4=4$ $|A|=n=4$. Структурная схема алгоритма показана на рис. 2.17,*б*. Блок *2* задает начальное значение переменной *i*. Блок *3* присваивает переменной *x* значение a_i (на первом шаге $a_i=a_1$). Блок *4* сравнивает значение переменной *i* с *n*. Это необходимо для определения момента окончания алгоритма. Блок *5* осуществляет увеличение переменной *i* на единицу. Блок *6* сравнивает предыдущее число a_i с последующим a_{i+1}. Блок *7* печатает значение найденного максимального элемента $a_3=8$ и его номер, блок «Конец» оканчивает процесс.

Основные достоинства описанного выше алгоритма следующие: обеспечивается обмен структурными схемами алгоритмов между

специалистами; облегчаются чтение и понимание алгоритмов; уменьшается число ошибок при программировании.

Следует отметить, что наряду с преимуществами структурные схемы имеют ряд недостатков. Они не содержат информации о данных, о памяти, об используемом наборе элементарных шагов. Вообще говоря, структурная схема — это средство для описания алгоритмов.

Следует упонянуть и о часто применяемом *«словесном» способе записи алгоритмов*. При этом перечисляются блоки алгоритма и в них записываются логические условия. Запишем, например, словесный алгоритм, структурная схема которого показана на рис. 2.17,*а*.

1°. Ввод исходных данных.

2°. Присвоить x значение 1.

3°. Вычислить $y=\sin a(x)/x$.

4°. Напечатать x, y.

5°. Присвоить x значение $x+1$.

6°. Проверить $x>50$. Если условие выполняется, то перейти к 7°, если нет, то к 3°.

7°. Конец работы алгоритма.

Отметим, что, хотя словесная запись алгоритма не формализована, она используется в интерактивном режиме работы конструктора с ЭВМ.

Для ускорения процесса перевода алгоритмов задач с языка алгоритмической системы в язык вычислительной машины разработаны специальные алгоритмические языки. Строгая формализация и однозначность описания алгоритмов в алгоритмическом языке позволила автоматизировать процесс его перевода в язык ЭВМ. Для этого машина снабжается транслятором (программирующей программой), который, анализируя описание, сделанное на алгоритмическом языке, перерабатывает его в программу, состоящую из команд машины. В настоящее время наибольшее распространение получили алгоритмические языки АЛГОЛ, ФОРТРАН, ПЛ/1, КОБОЛ и др.

Приведем теперь понятия о *расплывчатых алгоритмах*. Говорят, что расплывчатый алгоритм — это упорядоченное множество расплывчатых инструкций, которые при их реализации позволяют получать приближенное решение проблем. Инструкции в расплывчатых алгоритмах делят на три группы:

1) назначающие, например, «x — большой» или «x — небольшой» или «x — малая степень интеграции» и т. д.;

2) расплывчатые высказывания, например, «если x — не трассируется, тогда увеличить размер y», «если x — планарен, тогда стоп»;

3) безусловные активные предложения, например, «немного увеличить x», «перейти к другому блоку».

Данные инструкции могут быть расплывчатыми или обычными. Рассмотрим одну из возможных классификаций расплывчатых алгоритмов по областям их применения. Это алгоритмы опреде-

ления и идентификации, алгоритмы порождения, бихевиористические алгоритмы и алгоритмы принятия решений.

Расплывчатые алгоритмы определения — это конечное множество инструкций, определяющих расплывчатое множество в терминах других расплывчатых множеств или дающих метод для определения степени принадлежности каждого элемента для определяемого множества. *Алгоритмы идентификации* устанавливают степень принадлежности элементов множеству.

Алгоритмы порождения используются для порождения новых, а не для определения данных расплывчатых множеств. Примерами таких алгоритмов могут служить алгоритмы сочинения стихов, конструирования ЭВА, разработки технических заданий на РЭА.

Расплывчатые алгоритмы, используемые для описания поведения произвольной степени, называются *бихевиористическими.*

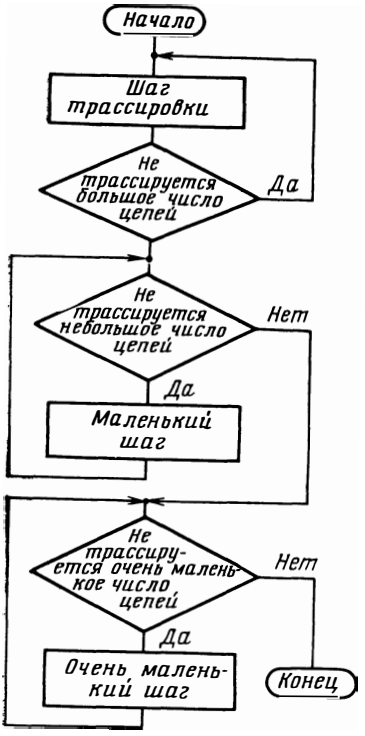


Рис. 2.18. Структурная схема расплывчатого алгоритма «Трассировка»

Например, пусть y — быстродействие системы, состоящей из двух ЭВМ, а x — быстродействие одной ЭВМ в системе.

Тогда можно записать такой алгоритм:

1°. Если x мало́ и x незначительно увеличить, то y незначительно увеличится.

2°. Если x мало́ и x значительно увеличить, то y увеличится значительно.

3°. Если x велико и x увеличить значительно, то y увеличится очень значительно.

Заметим, что значения расплывчатых условных предложений в этом алгоритме можно определить, если даны значения первичных терминов «большой», «маленький» и расплывчатых терминов «незначительно», «значительно», «очень значительно».

Говорят, что расплывчатый алгоритм, который служит для получения приближенного описания стратегии и решающего правила, называется расплывчатым *алгоритмом принятия решений.* К нему можно отнести алгоритм пересечения перекрестка, компоновку кристалла интегральной микросхемы и др.

Рассмотрим пример трассировки кристалла интегральной микросхемы. Структурная схема одного из возможных алгоритмов такого типа показана на рис. 2.18. Ее можно перевести в следующие инструкции:

1°. Сделать шаг трассировки. Если не трассируется большое число цепей, то перейти к 1°.

2°. Если не трассируется небольшое число цепей, то сделать малый шаг и перейти к 2°.

3°. Если не трассируется очень маленькое число цепей, то сделать очень маленький шаг и перейти к 3°.

4°. Конец работы алгоритма.

Следует отметить, что конструктор хорошо оперирует расплывчатыми понятиями «большое», «маленькое», «очень маленькое» и с успехом реализует данный алгоритм в интерактивном режиме связи с ЭВМ. При определении степени принадлежности можно формализовать алгоритм и реализовать его на ЭВМ. Очевидно, что операция «сделать шаг» может быть сложным алгоритмом перетрассировки всего кристалла, или небольшой его области, или простого удаления нескольких трасс и т. п.

Реальные варианты расплывчатых алгоритмов часто значительно сложнее. Важным в разработке и применении таких алгоритмов является то, что они могут служить эффективным средством распространения и изучения опыта конструктора. Особенно это актуально при конструировании ЭА на интегральных микросхемах в интерактивном режиме.

2.3. ЭЛЕМЕНТЫ ТЕОРИИ ГРАФОВ И ГИПЕРГРАФОВ

При изучении связи различных элементов (взаимоотношений людей в коллективе, связей в ЭА, в молекулах и т. п.) для наглядности упрощения и формализации представления элементы обозначаются точками и соединяются линиями. Такое изображение позволяет определять наиболее существенные связи, находить оптимальное решение задачи, определять ошибки в рассуждениях и т. д. В частности, такие объекты, состоящие из точек и соединяющих их линий, оказываются удобными моделями схем РЭА и ЭВА и в этой связи представляют интерес.

Считается, что Кёниг первый назвал такие объекты графами и начал их систематическое изучение, хотя отдельные вопросы, относящиеся к теории графов, рассматривались математиками значительно раньше.

Теория графов, развитая в трудах Р. Басакера, К. Бержа, А. А. Зыкова, Д. Кёнига, Н. Кристофидеса, А. Кофмана, О. Оре, Ф. Харари и многих других как абстрактная математическая наука оперирует формальными моделями объектов, имеет дело со свойствами графов независимо от того, какова природа объектов, описываемых графами. Использование аппарата теории графов оказало существенное влияние на разработку алгоритмов конструкторского проектирования схем. В книге будут рассмотрены лишь некоторые основные определения, правила, теоремы и положения из общей теории графов, которые будут представлять интерес в дальнейшем изложении.

Понятие графа опирается на понятие множества. *Графом* обычно называется объект, состоящий из двух множеств: множества точек и множества линий, которые находятся между собой в некотором отношении. Следует отметить, что в многочисленной литературе по теории графов встречается большое число описаний и определений графов, которые в общем являются эквивалентными.

Будем придерживаться терминологии, которая наиболее часто встречается в литературе по автоматизации конструирования.

Множество точек графа обозначается $X=\{x_1, x_2, ..., x_n\}$, $|X|=n$, и называется множеством вершин. Множество линий, соединяющих некоторые заданные пары вершин $(x_i, x_j)\in X$, называется множеством ребер или дуг и обозначается $U=\{u_1, u_2, ..., u_m\}$, $|U|=m$. Тогда графом можно считать математический объект, который обозначается $G=(X, U)$ и состоит из множества вершин и множества ребер или дуг, находящихся между собой в некотором отношении.

В общем случае множество U можно представить в виде $U=\tilde{U}\cup\vec{U}\cup\mathring{U}$, где $\tilde{U}$ — подмножество неориентированных ребер или просто подмножество ребер, в котором каждое ребро $u_i\in\tilde{U}$ определяется неупорядоченной парой соединяемых им вершин x_k, y_l. Записывается $u_i=(x_k, y_l)$ или $u_i=(y_l, x_k)$; $\vec{U}$ — подмножество дуг или ориентированных ребер (орребер), причем каждая дуга $u_i\in\vec{U}$ определяется упорядоченной парой (кортежем длины 2) соединяемых ею вершин x_k, y_l. Записывается $u_i=\langle x_k, y_l\rangle$ (заметим, что $u_i=\langle x_k, y_l\rangle$ и $u_j=\langle y_l, x_k\rangle$ — это различные дуги в графе G); $\mathring{U}$ — подмножество петель. Каждая петля $u_i\in\mathring{U}$ может определяться упорядоченной или неупорядоченной парой вершин, например парой вида $u_i=(x_k, x_k)$ или $u_i=\langle x_k, x_k\rangle$. В дальнейшем будем определять петли как неупорядоченные пары и оговаривать их применение в графах.

Граф называется смешанным, если множество U включает ребра, дуги и, возможно, петли, т. е. $\tilde{U}\neq\varnothing\wedge\vec{U}\neq\varnothing\wedge(\mathring{U}\neq\varnothing\vee\mathring{U}=\varnothing)$. На рис. 2.19 показан смешанный граф. Здесь $|X|=5$, $|U|=9$, $U=\tilde{U}\cup\vec{U}\cup\mathring{U}$, $\tilde{U}=\{u_3, u_4, u_7, u_9\}$, $\vec{U}=\{u_1, u_2, u_8\}$, $\mathring{U}=\{u_5, u_6\}$.

Граф, у которого $U=\vec{U}\cup\mathring{U}$, а $\tilde{U}=\varnothing$, *называется ориентированным* или орграфом. Заметим, что в орграфе $\mathring{U}$ может быть пустым подмножеством. На рис. 2.20 показан пример орграфа с петлями. Очевидно, что подмножество $\vec{U}$ можно представить как множество кортежей длины 2.

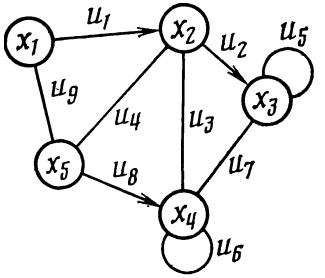


Рис. 2.19. Смешанный граф

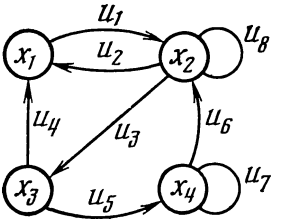


Рис. 2.20. Орграф с петлями

Например, для графа на рис. 2.20 $\vec{U}=\{\langle x_1, x_2\rangle, \langle x_2, x_1\rangle, \langle x_2, x_3\rangle, \langle x_3, x_1\rangle, \langle x_4, x_2\rangle, \langle x_3, x_4\rangle, \langle x_4, x_4\rangle, \langle x_2, x_2\rangle\}$. Здесь каждое ребро $u_i \in \vec{U}$ представляется парой соединяемых вершин, причем первой в кортеже стоит вершина, из которой выходит дуга, а второй — вершина, куда дуга входит.

Граф, у которого $U=\tilde{U} \cup \mathring{U}$, называется *неориентированным*, или неорграфом. Пример неорграфа с петлями показан на рис. 2.21. Заметим, что в неорграфе $\mathring{U}$ может быть пустым подмножеством.

В дальнейшем неорграфы с петлями и без петель будем называть просто графами.

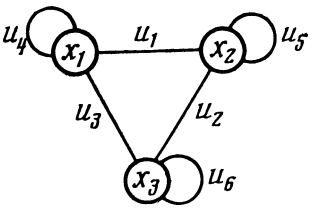


Рис. 2.21. Неорграф с петлями

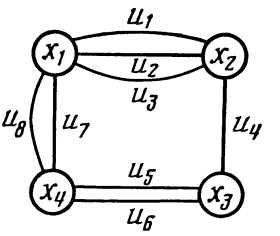


Рис. 2.22. Мультиграф

Граф, у которого существует хотя бы одна пара вершин, соединяемых m ребрами $u_i \in U$, называется *мультиграфом* ($m>1$), а максимальное m — мультичислом графа ($m \in M=\{2, 3, \ldots\}$). Ребра, соединяющие одну и ту же пару вершин, называются *кратными*. На рис. 2.22 показан пример мультиграфа.

Если ребро $u_k \in U$ графа соединяет вершины x_i, $x_j \in X$, т. е. $u_k=(x_i, x_j)$, то говорят, что *ребро* u_k *инцидентно вершинам* x_i, x_j. Вершины x_i, x_j в этом случае называют *инцидентными ребру* u_k.

Любые две вершины x_i, $x_j \in X$ графа называются *смежными*, если существует соединяющее эти вершины ребро $u_k \in U$, т. е. $u_k=(x_i, x_j)$. Если два ребра u_k, $u_l \in U$ инцидентны одной и той же вершине, то они называются *смежными*. Следовательно, отношения инцидентности и смежности могут иметь место как на множестве U, так и на множестве X. Смежность является отношением между элементами одного и того же множества вершин или ребер, а инцидентность — отношением между элементами разных множеств, т. е. множеств ребер и вершин.

Основными способами задания графа являются геометрический, аналитический и матричный.

Основой геометрического способа задания графа является рисунок, дающий изображение графа. Изображение графа в виде рисунка наглядно раскрывает содержательный смысл представляемого объекта.

Рассмотрим аналитический способ задания графа. Говорят, что задан граф, если даны множество вершин X, множество ребер U и инцидентор F, определяющий, какую пару вершин $(x_i, x_j) \in X$ соединяет ребро $u_k = (x_i, x_j)$.

Большинство задач автоматизации конструирования схем удобно решать при использовании матричного представления графов.

Квадратная таблица $\mathbf{R} = \|r_{i,j}\|_{n \times n}$ называется *матрицей смежности*, если ее элементы образуются по правилу

$$r_{i,j} = \begin{cases} 1, \text{ если вершина } x_i \text{ соединена с } x_j \\ \quad \text{ребром, т. е. } x_i \text{ смежна } x_j; \\ 0 \text{ в противном случае.} \end{cases}$$

Заметим, что для мультиграфа

$$r_{i,j} = \begin{cases} q, \text{ если вершина } x_i \text{ соединена с } x_j \\ q \text{ ребрами;} \\ 0 \text{ в противном случае.} \end{cases}$$

Очевидно, что для неорграфов $r_{i,j} = r_{j,i}$, и для задания графа можно использовать треугольную матрицу $\mathbf{R}$. Для графа на рис. 2.23 матрица смежности

$$\mathbf{R} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \left\| \begin{array}{ccccc} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{array} \right\| \end{array},$$

а треугольная матрица $\mathbf{R}$ для этого же графа запишется

$$\mathbf{R} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \left\| \begin{array}{ccccc} 0 & 1 & 0 & 0 & 1 \\ & 0 & 1 & 0 & 1 \\ & & 0 & 1 & 0 \\ & & & 0 & 1 \\ & & & & 0 \end{array} \right\| \end{array}.$$

Строки и столбцы $\mathbf{R}$ соответствуют вершинам графа. На пересечении i-й строки и j-го столбца ставится элемент $r_{i,j}$, соответствующий числу ребер, соединяющих вершины x_i и x_j. Заметим, что строки и столбцы матрицы $\mathbf{R}$ также можно нумеровать числами натурального ряда, соответствующими индексам помеченных вершин графа. Петли в графе соответствуют элементам $r_{i,j}$, расположенным по главной диагонали матрицы $\mathbf{R}$. Преимущество использования матриц смежности — это простота выполнения преобразований и операций над графами как для конструктора, так и для ЭВМ.

Основной недостаток применения матрицы смежности заключается в том, что для ее хранения в ЭВМ требуется память для $|X|^2$ чисел, даже если она содержит только нулевые элементы. Устранить этот недостаток для разряженных матриц можно представлением графа в виде списков. Так, списком смежности для вершины x_i является совокупность всех вершин, смежных с x_i.

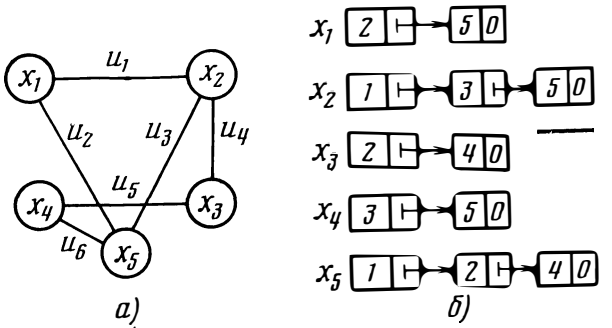


Рис. 2.23. Граф (*а*) и списки смежности для него (*б*)

Тогда граф можно представить с помощью списков смежности его вершин. На рис. 2.23,*б* показаны пять списков смежности — по одному для каждой вершины графа на рис. 2.23,*а*. Представление графа в виде списков смежности требует памяти ЭВМ порядка $0(|X|+|U|)$. В основном таким представлением графа пользуются, когда $|U| \ll |X|$.

Прямоугольная таблица вида $\mathbf{I}=\|i_{k,l}\|_{n\times m}$ называется матрицей инцидентности, если ее элементы образуются по правилу

$$i_{k,l}=\begin{cases}1, \text{ если вершина } x_k \text{ инцидентна ребру } u_l;\\ 0 \text{ в противном случае.}\end{cases}$$

Матрица инцидентности для графа, изображенного на рис. 2.23, примет вид

$$\mathbf{I}=\begin{array}{c} \\ x_1\\ x_2\\ x_3\\ x_4\\ x_5\end{array}\begin{array}{c}\begin{array}{cccccc} u_1 & u_2 & u_3 & u_4 & u_5 & u_6\end{array}\\ \left\|\begin{array}{cccccc} 1 & 1 & 0 & 0 & 0 & 0\\ 1 & 0 & 1 & 1 & 0 & 0\\ 0 & 0 & 0 & 1 & 1 & 0\\ 0 & 0 & 0 & 0 & 1 & 1\\ 0 & 1 & 1 & 0 & 0 & 1\end{array}\right\|\end{array}.$$

Строки матрицы $\mathbf{I}$ соответствуют вершинам графа, столбцы — ребрам, а элемент $i_{k,l}$ указывает на инцидентность вершины x_k и ребра u_l. В каждом столбце матрицы $\mathbf{I}$ не более двух единиц, так как каждое ребро соединяет ровно две вершины. При наличии в графе петель соответствующие им столбцы в матрице $\mathbf{I}$ будут иметь по одной единице, так как петля инцидентна только одной вершине графа.

Матрицы **R** и **I** однозначно дают информацию о графе. Существуют простые способы перехода от одной матрицы к другой. При переходе от матрицы **I** к матрице **R** теряется нумерация ребер. Рассмотрим алгоритм перехода от **I** к **R** на примере графа, изображенного на рис. 2.23. Просматривается первый столбец **I**. Он содержит единицы в строках x_1, x_2. Следовательно, в матрице **R** ставится элемент $r_{1,2}=1$. Просматривая второй, третий, четвертый, пятый и шестой столбцы матрицы **I**, получаем единичные элементы матрицы **R**: $r_{1,5}=1$; $r_{2,5}=1$; $r_{2,3}=1$; $r_{3,4}=1$; $r_{4,5}=1$.

При переходе от матрицы **R** к **I** необходимо предварительно пронумеровать $r_{i,j}$ соответствующими значениями ребер $u_k \in U$.

Определим смежностный (двойственный) граф $G_s=(U, V)$ для графа $G=(X, U)$ и построим его матрицу смежности. Вершинами G_s являются ребра G, а ребрами — пары (u_i, u_j), причем ребро $v_k=(u_i, u_j)$ соединяет вершины u_i, u_j графа G_s, если ребра u_i, u_j в графе G смежны. Для построения G_s по графу G на каждом ребре $u_i \in U$ графа G выбирают точку и считают ее вершиной $u_i \in U$ графа G_s. Затем пару u_i, u_j соединяют ребром $v=(u_i, u_j)$, если ребра u_i, u_j имеют общую вершину в G. Для графа G на рис. 2.24,*а* двойственный ему граф G_s изображен на рис. 2.24,*б*.

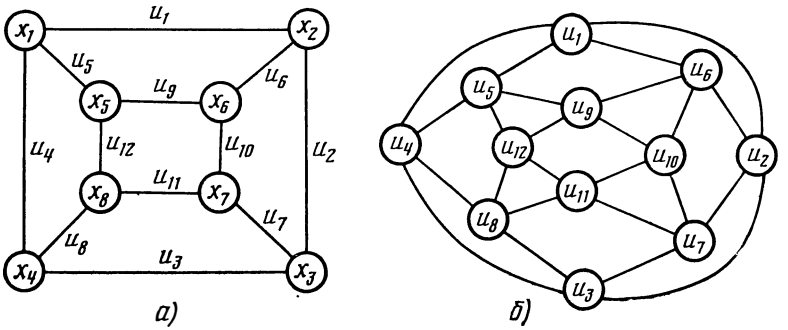


Рис. 2.24. Граф G (*а*) и двойственный ему граф G_s (*б*)

Следует отметить, что существуют различного рода матрицы и списки, производные от **R** и **I**, которые используются при записи алгоритмов конструирования схем.

Граф называется конечным, если конечны множества его вершин и ребер. Далее будем рассматривать только конечные графы, так как схемы ЭА, представляемые графами, состоят из конечного числа элементов.

Граф, у которого $X \neq \varnothing$ и $U=\varnothing$, называется *нуль-графом*, а вершины его называются изолированными. Нуль-граф обозначается через G_0.

Граф $G=(X, U)$, $|X|=n$, называется *полным*, если между любой парой вершин x_i, $x_j \in X$ имеется ребро $u_k \in U$. Полный граф обозначается через K_n.

Число ребер, инцидентных вершине $x_i \in X$ графа, называется *локальной степенью вершины* и обозначается $\rho(x_i)$. Тогда число ребер графа $G=(X, U)$ без петель с $|X|=n$, $|U|=m$ равно

$$m=\sum_{i=1}^{n}\frac{\rho(x_i)}{2}.$$

Заметим, что если в графе имеются петли, то для определения локальных степеней вершин каждая из них должна считаться дважды.

Так как для полного графа с n вершинами $\rho(x_1)=\rho(x_2)=\ldots=\rho(x_n)=n-1$, то в K_n $m=n(n-1)/2$.

Заметим, что в графе каждое ребро соединяет две вершины, и, следовательно, число вершин нечетной степени четно.

Графы, у которых все вершины имеют одинаковую локальную степень, называются *регулярными*. Регулярные графы степени 3 называются кубическими. Число ребер регулярного графа с локальной степенью r равно $m=nr/2$.

Подграфом графа G называется граф, у которого все вершины и инцидентные им ребра принадлежат G, т. е. $G'=(X', U')$ есть подграф графа $G=(X, U)$, если $X' \subseteq X$, $U' \subseteq U$ и ребра G' инцидентны только вершинам из X'. Удаление из графа G вершины x_i со всеми инцидентными ей ребрами приводит к подграфу $G'=G-x_i$. Существует гипотеза, что набор подграфов $G-x_i$ несет полную информацию о графе.

На рис. 2.25 показаны граф $G=(X, U)$ и два его подграфа

$$G_1=(X_1, U_1) \text{ и } G_2=(X_2, U_2);$$

$$|X|=6,\ |U|=8,\ X=\{x_1, x, \ldots, x_6\},$$

$$U=\{u_1, u_2, \ldots, u_8\};$$

$$|X_1|=3,\ |U_1|=3,\ X_1=\{x_1, x_2, x_4\},$$

$$U_1=\{u_1, u_7, u_6\};$$

$$|X_2|=4,\ |U_2|=4,\ X_2=\{x_2, x_3,$$

$$x_5, x_6\},\ U_2=\{u_2, u_3, u_4, u_8\}.$$

Заметим, что подграф G_1 можно получить, удалив из G вершины x_3, x_5, x_6.

Суграфом $G'=(X', U')$ графа $G=(X, U)$ называется граф, для которого $X'=X$, $U' \subseteq U$ (рис. 2.25,*г*).

Граф $\bar{G}$ называется *дополнением графа* G до полного, если множество его ребер состоит из ребер полного графа K_n, не принадлежащих G: $\bar{G}=(X, \bar{U})$, $\bar{U}=U_k \setminus U$, $K_n=(X, U_k)$ (рис. 2.25,*д*).

Маршрутом в графе называется некоторая конечная последовательность ребер вида $s=(x_0, x_1), (x_1, x_2), \ldots, (x_{l-1}, x_l)$, где x_0, x_l — соответственно его начальная и конечная вершины. Число ребер в марштуре s называется его длиной. Существует простой способ определения маршрутов длины q по матрице $\mathbf{R}$ графа G путем

возведения ее в q-ю степень. Например, пусть задана матрица $\mathbf{R}$ графа, изображенного на рис. 2.26:

$$\mathbf{R}=\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \left\|\begin{array}{cccc} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right\| \end{array}.$$

Возведем ее в квадрат:

$$\mathbf{R}^2=\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \left\|\begin{array}{cccc} 2 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \end{array}\right\| \end{array}.$$

Каждый $r_{i,j}$-элемент $\mathbf{R}^2$ равен числу маршрутов длины 2, ведущих из вершины x_i в x_j. Например, $r_{3,2}=2$, означает, что в графе два маршрута длины 2: $s_1=(x_3, x_1), (x_1, x_2)$ и $s_2=(x_3, x_4), (x_4, x_2)$.

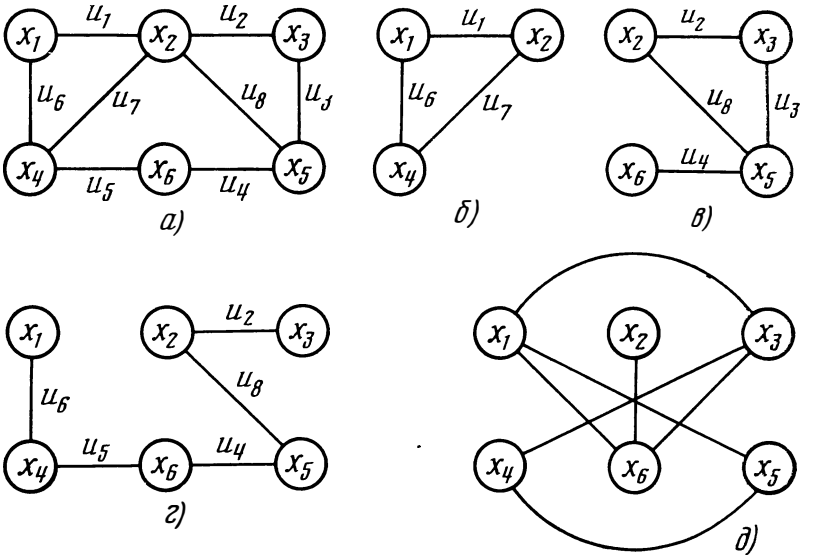


Рис. 2.25. Граф $G=(X, U)$ $(а)$, его подграфы $G_1=G-\{x_3, x_5, x_6\}$ $(б)$ и $G_2=G-\{x_1, x_4\}$ $(в)$, суграф G' $(г)$ и дополнение $\bar{G}$ до полного K_6 $(д)$

Возведение матрицы в степень производится умножением матрицы на себя по правилу умножения матриц. Для определения элемента $r_{i,j}$ матрицы $\mathbf{R}^2$ производится поэлементное перемножение i-й строки на столбец матрицы j и полученные результаты скла-

дываются. Например, элемент $r_{3,3}$ матрицы $\mathbf{R}^2$ определится $r_{3,3}=r_{1,3}r_{3,1}+r_{2,3}r_{3,2}+r_{3,3}r_{3,3}+r_{4,3}r_{3,4}=2$.

Для определения маршрутов длины 3 определим матрицу $\mathbf{R}^3$ умножением матрицы $\mathbf{R}^2$ на $\mathbf{R}$:

$$\mathbf{R}^3=\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \left\|\begin{array}{cccc} 0 & 4 & 4 & 0 \\ 4 & 0 & 0 & 4 \\ 4 & 0 & 0 & 4 \\ 0 & 4 & 4 & 0 \end{array}\right\| \end{array}.$$

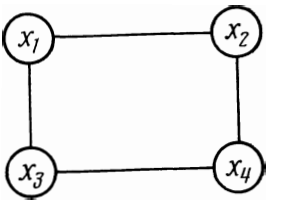


Рис. 2.26. Граф

Например, элемент матрицы $\mathbf{R}^3$ $r_{1,2}=4$ показывает, что существуют четыре маршрута длины 3, ведущие из вершины x_1 в x_2 (см. рис. 2.26): $s_1=(x_1, x_2)\,(x_2, x_1)\,(x_1, x_2)$; $s_2=(x_1, x_3)\,(x_3, x_4)\,(x_4, x_2)$; $s_3=(x_1, x_2)\,(x_2, x_4)\,(x_4, x_2)$; $s_4=(x_1, x_3)\,(x_3, x_1)\,(x_1, x_2)$.

Маршрут, в котором нет повторяющихся ребер, называется *цепью*. Замкнутая цепь, в которой $x_0=x_l$, называется *циклом*. Соответственно цепи и циклы называются простыми, если они не содержат повторяющихся вершин, кроме, разумеется, первой и последней в случае цикла. На рис. 2.25,*а* в графе G

$s_1=(x_1, x_2)$, (x_2, x_3), (x_3, x_5), (x_5, x_2), (x_2, x_1), (x_1, x_4) — маршрут;

$s_2=(x_1, x_2)$, (x_2, x_3), (x_3, x_5), (x_5, x_6) — цепь;

$s_3=(x_5, x_2)$, (x_2, x_4), (x_4, x_1), (x_1, x_2), (x_2, x_3), (x_3, x_5) — цикл;

$s_4=(x_1, x_2)$, (x_2, x_3), (x_3, x_5), (x_5, x_6) — простая цепь;

$s_5=(x_1, x_2)$, (x_2, x_3), (x_3, x_5), (x_5, x_6), (x_6, x_4), (x_4, x_1) — простой цикл.

Две произвольные вершины x_i, $x_j\in X$ графа называются связными, если существует маршрут s, в котором вершины x_i, x_j будут концевыми. Граф называется *связным*, если любые две его вершины связаны. В противном случае граф не связан, а каждый из составляющих его связных подграфов G_1, G_2, .., G_l называется компонентой связности. Из определения связности следует: в связном графе вершина x_i связана сама с собой; если x_i связана с x_j, то x_j связана с x_i, если x_i связана с x_j, а x_j связана с x_k, то и x_i связана с x_k ($x_i, x_j, x_k\in X$). Тогда отношение связности является отношением эквивалентности. В этом случае множество вершин X графа $G=(X, U)$, который моделирует схему, можно разбить на непересекающиеся классы X_i, причем ребра графа будут соединять только вершины внутри этих классов. Таким образом получим разбиение графа G на связные подграфы (компоненты связности) $G=\bigcup_{i=1}^{l} G_i$. Например, графы, изображенные на рис. 2.24—2.26, связные, а граф на рис. 2.27 состоит из трех компонент связности G_1, G_2, G_3, т. е. не связан. Заметим, что связный граф состоит из единственной компоненты связности. Если граф имеет несколько

компонент связности, то он не связан, так как вершины из разных компонент связности нельзя соединить маршрутом.

При конструировании часто надо знать, какое наименьшее число связей необходимо удалить из схемы, чтобы она перестала быть связной.

Одной из характеристик связных графов является число ребер в графе с n вершинами и заданным числом k компонент связности. Оно удовлетворяет неравенству

$$n-k \leqslant m \leqslant (n-k)(n-k+1)/2.$$

Из неравенства следует, что граф с n вершинами и более чем $(n-1)(n-2)/2$ ребрами связан.

Пусть задан связный граф $G=(X, U)$. Подмножество $U' \subseteq U$ называется *разделяющим*, если после его удаления граф становится несвязным. Заметим, что разделяющее подмножество в графе всегда существует. Тривиальным разделяющим подмножеством G является $U'=U$. Если разделяющее множество состоит из одного ребра $u_i \in U$, то u_i называется *перешейком* или *мостом*.

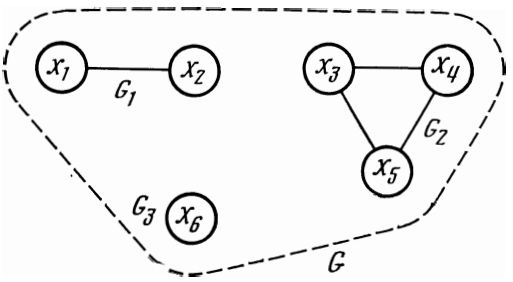


Рис. 2.27. Граф G, состоящий из трех компонент связности G_1, G_2, G_3

При решении задач автоматизации конструирования ЭА важное значение имеют графы специального вида — эйлеровы и гамильтоновы.

Связный граф $G=(X, U)$ называется эйлеровым, если существует замкнутая цепь (т. е. цикл), проходящая через каждое его ребро только 1 раз. Граф называется полуэйлеровым, если существует незамкнутая цепь, проходящая через каждое ребро графа только 1 раз. На рис. 2.28—2.30 показаны неэйлеров, полуэйлеров и эйлеров графы. Порядок обхода графов (см. рис. 2.29, 2.30) показан стрелками. Известно условие существования эйлерового цикла: конечный G является эйлеровым, если он связен и все его локальные степени четны. Например, граф G на рис. 2.24,*а* не является эйлеровым, так как степени всех его вершин нечетны. Граф G на рис. 2.27 также не является эйлеровым, так как он не связен. Обозначается эйлеров цикл $S_э$. Так, для графа на рис. 2.30 $S_э = (x_7, x_2)(x_2, x_4)(x_4, x_6)(x_6, x_3)(x_3, x_5)(x_5, x_4)(x_4, x_3)(x_3, x_1)(x_1, x_5)(x_5, x_2)(x_2, x_1)(x_1, x_7)$.

Заметим, что связный граф является полуэйлеровым, когда в нем не более двух вершин имеют нечетные локальные степени, причем одна из этих вершин будет начальной, а другая — конечной вершиной цепи.

Запишем идею алгоритма Флери построения эйлеровой цепи в эйлеровом графе. Пусть G — эй-

леров граф. Тогда необходимо выйти из произвольной вершины и проходить по ребрам G соблюдая правила:

стирать ребра по мере их прохождения и стирать образующиеся изолированные вершины;

по мосту можно проходить только тогда, когда нет других возможностей.

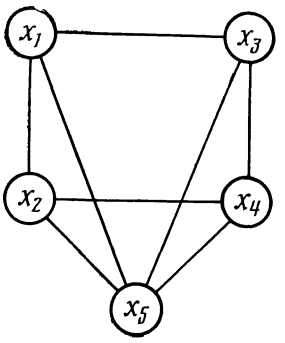


Рис. 2.28. Неэйлеров граф

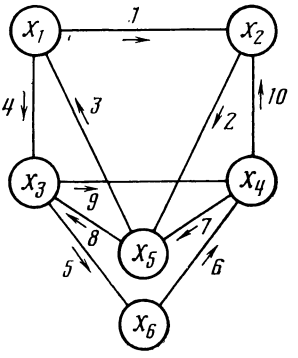


Рис. 2.29. Полуэйлеров граф

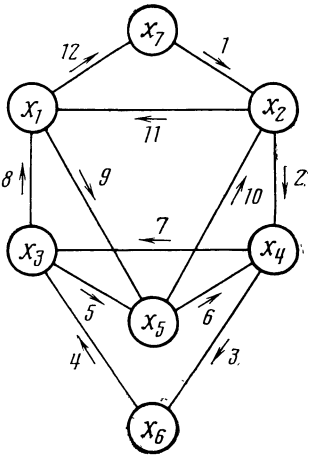


Рис. 2.30. Эйлеров граф

Цикл, проходящий по всем вершинам графа G 1 раз, называется гамильтоновым, а G называется гамильтоновым графом. Например, граф на рис. 2.25,*д* не имеет гамильтонова цикла, а граф на рис. 2.30 имеет.

В отличие от эйлерового цикла, для гамильтонова цикла неизвестен общий критерий существования. Известны только теоремы, дающие достаточные условия существования или отсутствия гамильтонова цикла.

Теорема 2.1. Если в графе с n ($n \geqslant 3$) вершинами для любой пары несмежных вершин x_i, x_j $\rho(x_i)+\rho(x_j) \geqslant n$, то граф имеет гамильтонов цикл.

Теорема 2.2. В графе без гамильтонова цикла длина его наибольших простых цепей удовлетворяет неравенству $l \geqslant \rho(x_{н1}) + \rho(x_{н2})$, где $\rho(x_{н1})$ и $\rho(x_{н2})$ — две наименьшие локальные степени графа.

Введем понятие жордановой кривой и приведем один из вариантов теоремы Жордана. *Жордановой кривой* на плоскости называется непрерывная кривая, не имеющая самопересечений. Соответственно замкнутой жордановой кривой называется жорданова кривая, начало и конец которой совпадают.

Теорема 2.3 (Жордана). Если $\mathscr{L}$ — замкнутая жорданова кривая, а x_i, x_j две различные точки, расположенные на ней, то любая жорданова кривая, соединяющая x_i и x_j, должна лежать це-

ликом внутри $\mathscr{L}$, или вне $\mathscr{L}$ (за исключением точек x_i, x_j), или пересекать $\mathscr{L}$ в некоторой точке, отличной от x_i, x_j.

Если вершины графа $G=(X, U)$ расположить на плоскости так, чтобы гамильтонов цикл представлял собой самонепересекающуюся замкнутую кривую, то гамильтонов цикл разделит плоскость на две области — внутреннюю и внешнюю. Естественно, что внутренняя и внешняя области взаимно обратимы. Вообще говоря, согласно теореме Жордана любой простой цикл, расположенный на плоскости, разбивает ее на внешнюю и внутреннюю области, причем если имеются две вершины: x_j, расположенная во внешней области, и x_i, расположенная во внутренней области, — то соединить их ребром без пересечения ребер цикла невозможно (рис. 2.31). При этом ребру (x_i, x_j) не разрешается проходить через любую вершину цикла. Здесь и далее под пересечением понимается соединение двух ребер графа в точке, не соответствующей никакой вершине.

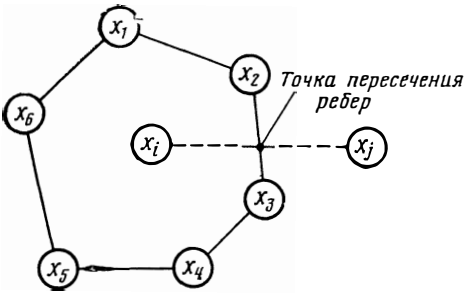


Рис. 2.31. Пример соединения вершин x_i и x_j

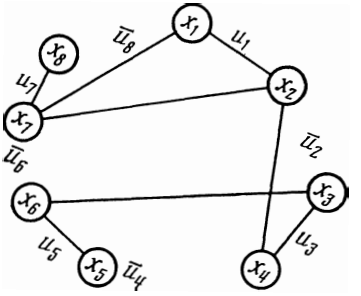


Рис. 2.32. Гамильтонов псевдоцикл

Если множество вершин графа $G=(X, U)$ расположить на самонепересекающейся кривой, ребра, соединяющие близлежащие друг к другу вершины графа, образуют гамильтонов псевдоцикл, содержащий как ребра $u_i \in U$, так и фиктивные ребра $\bar{u}_i \in U$, которые на самом деле отсутствуют. Например, гамильтонов псевдоцикл показан на 2.32. Здесь $s_{\text{гпц}}=x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_1$, его ребра $u_1=(x_1, x_2)$, $u_3=(x_3, x_4)$, $u_5=(x_5, x_6)$, $u_7=(x_7, x_8)$, а фиктивными ребрами являются $\bar{u}_2$, $\bar{u}_4$, $\bar{u}_6$, $\bar{u}_8$. Очевидно, что полный граф всегда содержит гамильтонов цикл.

Связный граф без циклов называется *деревом* и обозначается $T=(X, U)$, $|X|=n$. Любое дерево T имеет $n-1$ ребро. Рассматривают корневые и некорневые деревья. Начальная вершина, т. е. вершина с наименьшим номером, называется *корнем*, из которого выходят ребра, называемые ветвями дерева. Очевидно, что в дереве любые две вершины x_i, x_j связаны единственной цепью и в любом связном графе G можно выделить некоторое дерево T. Для задач конструирования наибольший интерес представляют деревья, у которых число вершин равно числу вершин графа, из ко-

торого выделено это дерево. Такие деревья называются *покрывающими*. Для одного и того же связного графа можно выделить некоторое множество покрывающих деревьев.

Теорема 2.4. Число t покрывающих деревьев в полном графе K_n составляет $t=n^{n-2}$.

Множество деревьев называется *лесом*. На рис. 2.33 показан лес, состоящий из трех деревьев: T_1, T_2 и T_3.

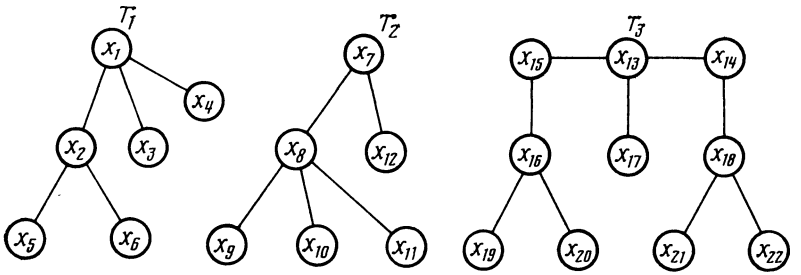


Рис. 2.33. Лес, состоящий из трех деревьев: T_1, T_2, T_3

Задачи выделения эйлеровых и гамильтоновых циклов и покрывающих деревьев связаны с задачами о лабиринте, коммивояжере и с построением деревьев минимальной стоимости.

Задача о лабиринте в терминах теории графов формулируется как задача отыскания в связном графе $G=(X, U)$ маршрута с наименьшим числом ребер, который начинается в заданной вершине $x_i \in X$ и приводит в искомую вершину $x_j \in X$.

Рассмотрим метод Тремо решения задачи о лабиринте. От вершины x_i необходимо перейти ко всем вершинам, находящимся на расстоянии один. Каждое ребро $u_i=(x_i, x_l)$ помечается 1 раз, когда оно смежно вершинам x_i и x_l. В вершине x_i это ребро помечается как «открытое». Если окажется, что нет больше ребер, инцидентных x_l, кроме u_i, то, вернувшись в x_i, ребро u_i помечается как «закрытое». Если некоторое другое ребро $u_i=(x_i, x_l)$ также ведет из x_i в x_l, то оно также помечается как закрытое. Очевидно, если вершина x_j расположена на расстоянии одного ребра, то она будет найдена таким просмотром. Для попадания в вершины, лежащие от x_i на расстоянии два, берется открытое ребро $u=(x_i, x_l)$ и снова помечается. В x_l открытые ребра проходятся и помечаются как закрытые, если они ведут к уже пройденным вершинам. После просмотра всех ребер происходит возврат в x_i из x_l по ранее отмеченному входящему ребру. Если не осталось ребер, открытых в x_l, то это открытое ребро закрывается в x_i. После возвращения в x_i аналогичная операция повторяется для других открытых ребер и процесс продолжается, пока все ребра в x_i не будут помечены дважды. Если искомая вершина x_j расположена на расстоянии n, то при ее достижении все открытые ребра в x_i будут помечены n раз, открытые ребра в любой вершине x_l будут

помечены $n-1$ раз и т. д. Такой процесс поиска последовательно покрывает все вершины графа и всегда находит выход из лабиринта.

Пусть нам необходимо проложить сеть проводов, связывающих n блоков вычислительной аппаратуры, причем так, чтобы из одного блока можно было связаться с любым другим. Если из экономических соображений требуется, чтобы количество затраченного провода было минимально, то граф, вершины которого соответствуют блокам, а ребра — соединяющим их проводам, должен быть деревом. Задача состоит в определении одного из n^{n-2} возможных деревьев с минимизацией суммарной длины ребер (проводов). На языке теории графов эту задачу можно сформулировать следующим образом. Пусть $G=(X, U)$ — связный граф, каждому ребру которого $u_i \in U$ ставится в соответствие некоторое неотрицательное число $\nu(u)$, называемое его мерой (весом). Необходимо найти алгоритм построения покрывающего дерева T, у которого сумма мер, взятая по всем ребрам,

$$\sum_{i=1}^{m} \nu(u_i) = \min.$$

Приведем алгоритм Краскала решения данной задачи.

1°. В связном графе $G=(X, U)$, $|X|=n$, определяется ребро с наименьшей мерой $\nu(u_1)$.

2°. Строится по индукции последовательность ребер $u_2, u_3, \ldots, u_{n-1}$, причем на каждом шаге выбирается ребро, не совпадающее с предыдущим, обладающее наименьшей мерой и не образующее циклов с предыдущими ребрами u_i.

3°. Получается подграф T графа G с ребрами $u_1, \ldots, u_{n-1}$, который и является искомым деревом с наименьшей суммой мер.

Рассмотрим основные понятия метрики графов. Метрика графа основана на понятии расстояния. Назовем *расстоянием* $d(x_i, x_j)$ (для простоты будем также использовать обозначение $d_{i,j}$) между вершинами x_i, $x_j \in X$ графа $G=(X, U)$ длину кратчайшей цепи, соединяющей эти вершины. *Под длиной цепи* понимается число входящих в нее ребер. Функция $d(x_i, x_j)$, определенная на множестве ребер графа G, называется *метрикой графа*. Определенная метрика удовлетворяет следующим аксиомам Фреше:

$$\forall x_i, x_j \in X [d(x_i, x_j) \geqslant 0];$$

$$\forall x_i, x_j \in X [d(x_i, x_j) = 0 \leftrightarrow x_i = x_j];$$

$$\forall x_i, x_j \in X [d(x_i, x_j) = d(x_j, x_i)];$$

$$\forall x_i, x_j, x_k \in X [d(x_i, x_j) + d(x_j, x_k) \geqslant d(x_i, x_k)].$$

Очевидно, что для нахождения метрики графа G достаточно знать его матрицу смежности.

Функцию расстояний для графа G удобно задавать матрицей расстояний $\mathbf{D}=\|d_{i,j}\|_n$ или ее списком при большой разреженности

матрицы **D**. Элемент матрицы **D** определяется следующим образом:

$$d_{i,j}=\begin{cases}0, & \text{если } x_i=x_j;\\ d_{i,j}, & \text{если } x_i\neq x_j.\end{cases}$$

Если принять, что для графа на рис. 2.26 $d_{1,2}=d_{2,1}=d_{1,3}=d_{3,1}=d_{2,4}=d_{4,2}=d_{3,4}=d_{4,3}=1$, то матрица расстояний имеет вид

$$\mathbf{D}=\begin{array}{c|cccc} & 1 & 2 & 3 & 4\\ 1 & 0 & 1 & 1 & 2\\ 2 & 1 & 0 & 2 & 1\\ 3 & 1 & 2 & 0 & 1\\ 4 & 2 & 1 & 1 & 0\end{array}.$$

Список расстояний можно записать в виде множества, состоящего из кортежей длины 3: $D=\{\langle 1, 2, 1\rangle, \langle 1, 3, 1\rangle, \langle 1, 4, 2\rangle, \langle 2, 3, 2\rangle, \langle 2, 4, 1\rangle, \langle 3, 4, 1\rangle\}$. Первый элемент в кортеже соответствует x_i — вершине графа, второй элемент — x_j также вершине, а третий элемент кортежа соответствует расстоянию $d_{i,j}$.

Диаметр графа $d(G)$ определяется как максимальное расстояние между его вершинами: $d(G)=\max d_{i,j}\ x_i,\ x_j\in X$.

Для задач конструирования представляет интерес нахождение расстояний для графов G_r частного вида, называемых координатной сеткой. Введем декартову систему координат с осями s и t, в которой расположим граф G_r. В графе $G_r=(X_r, U_r)$ множество вершин X_r соответствует узлам сетки, а множество ребер U_r — горизонтальным и вертикальным отрезкам, соединяющим узлы сетки. Пример координатной сетки, т. е. графа G_r, показан на рис. 2.34. Очевидно, что граф G_r удовлетворяет аксиомам Фреше. Расстояние между двумя смежными вершинами в G_r, называе-

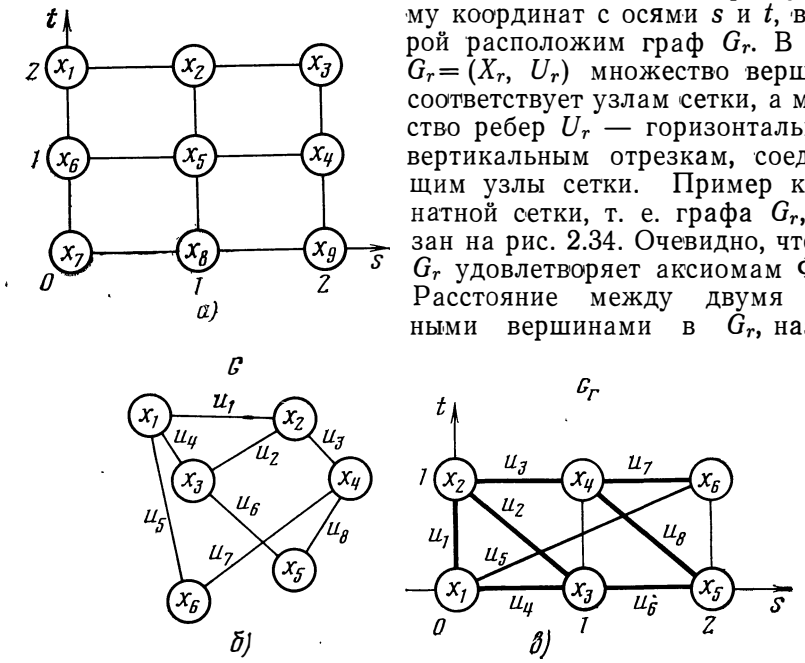


Рис. 2.34. Граф G_r — координатная сетка (*а*), граф (*б*) и его отображение в сетке G_r (*в*)

мое шагом сетки, обычно принимается равным единице. Расстояние $d_{i,j}$ между двумя произвольными вершинами в G_r можно определить по формуле $d_{i,j}=|s_i-s_j|+|t_i-t_j|$, где s_i, s_j и t_i, t_j — координаты вершин x_i, $x_j \in G_r$ в сетке G_r с координатными осями s и t.

Обычно задаются размеры сетки $p \times q$, где p число узлов сетки по оси s, а q — по оси t.

Например, для графа G_r на рис. 2.34, а расстояние $d_{6,9}=|s_6-s_9|+|t_6-t_9|=|0-2|+|1-0|=3$.

Расстояние $d_{i,j}$, определенное таким образом, называют *манхеттоновым расстоянием.*

Иногда расстояние $d_{i,j}$ определяют как длину кратчайшей прямой, соединяющей вершины x_i, x_j, тогда

$$d_{i,j}=\sqrt{|s_i-s_j|^2+|t_i-t_j|^2}.$$

Для рассмотренного примера $d_{6,9}=\sqrt{5}\approx 2{,}24$.

Заметим, что в первом случае использовать $d_{i,j}$ проще, так как при этом оно принимает только целочисленные значения.

Любой граф G может быть отображен в сетку G_r. Для $G=(X, U)$ при этом $|X_{G_r}| \geqslant |X_G|$ и каждым двум вершинам G соответствуют разные вершины G_r. Для подсчета суммарной длины $L(G)$ ребер графа $G=(X, U)$, отображенного в сетку G_r, введем понятие матрицы геометрии $\mathbf{D}_\gamma$. Матрица геометрии $\mathbf{D}_\gamma$ графа G представляет собой часть матрицы расстояний $\mathbf{D}$ графа G_r, в которой исключены элементы $d_{i,j}$, если вершины x_i, $x_j \in X$ не смежны в графе G. Элемент

$$d^{\gamma}_{i,j}=\begin{cases}0, \text{ если } r_{i,j}=\varnothing;\\ d_{i,j}, \text{ если } r_{i,j}\neq\varnothing.\end{cases}$$

Для построения матрицы геометрии $\mathbf{D}_\gamma$ графа G необходимо каждый элемент матрицы $\mathbf{D}$ умножить на соответствующий элемент матрицы смежности $\mathbf{R}$, т. е. $\mathbf{D}_\gamma=\|r_{i,j}d_{i,j}\|_n$. Например, на рис. 2.34, *б* показан граф, а на рис. 2.34, *в* его отображение в сетку G_r размером 3×2.

Матрицы $\mathbf{R}$ графа G и $\mathbf{D}$ графа G_r (рис. 2.34) примут вид

$$\mathbf{R}=\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6\\ 1 & 0 & 1 & 1 & 0 & 0 & 1\\ 2 & 1 & 0 & 1 & 1 & 0 & 0\\ 3 & 1 & 1 & 0 & 0 & 1 & 0\\ 4 & 0 & 1 & 0 & 0 & 1 & 1\\ 5 & 0 & 0 & 1 & 1 & 0 & 0\\ 6 & 1 & 0 & 0 & 1 & 0 & 0 \end{array};\qquad \mathbf{D}=\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6\\ 1 & 0 & 1 & 1 & 2 & 2 & 3\\ 2 & 1 & 0 & 2 & 1 & 3 & 2\\ 3 & 1 & 2 & 0 & 1 & 1 & 2\\ 4 & 2 & 1 & 1 & 0 & 2 & 1\\ 5 & 2 & 3 & 1 & 2 & 0 & 1\\ 6 & 3 & 2 & 2 & 1 & 1 & 0 \end{array}.$$

Тогда можно определить матрицу геометрии $\mathbf{D}_\gamma=\mathbf{R}\otimes\mathbf{D}$. Здесь $\otimes$ — знак поэлементного умножения матриц;

$$\mathbf{D}_\gamma = \begin{array}{c|cccccc|c} & 1 & 2 & 3 & 4 & 5 & 6 & \Sigma \\ \hline 1 & 0 & 1 & 1 & 0 & 0 & 3 & 5 \\ 2 & 1 & 0 & 2 & 1 & 0 & 0 & 4 \\ 3 & 1 & 2 & 0 & 0 & 1 & 0 & 4 \\ 4 & 0 & 1 & 0 & 0 & 2 & 1 & 4 \\ 5 & 0 & 0 & 1 & 2 & 0 & 0 & 3 \\ 6 & 3 & 0 & 0 & 1 & 0 & 0 & 4 \end{array}$$

Сумма элементов матрицы $\mathbf{D}_\gamma$ определяет удвоенную суммарную длину ребер графа в G при данном отображении его в сетку G_r. Для рассмотренного примера суммарная длина ребер графа $L(G)$ составляет 12 усл. ед.

При другом отображении графа в сетку суммарная длина ребер $L(G)$ изменяется, хотя в частном случае может совпадать с предыдущей.

Заметим, что изучение свойств графов и оптимизация в алгоритмах конструирования упрощаются при использовании таких числовых инвариантных характеристик графа, как числа цикломатическое, хроматическое, внутренней и внешней устойчивости и др.

Наименьшее число ребер, которое необходимо удалить из графа G, чтобы он стал ациклическим (деревом или лесом, если G состоит из нескольких компонент связности), называется *цикломатическим числом графа.* Для графа $G=(X, U)$, $|X|=n$, $|U|=m$, цикломатическое число $\gamma(G)=m-n+k$, где k — число компонент связности графа. Очевидно, для графа, состоящего из одной компоненты связности, $\gamma(G)=m-n+1$. Например, для графа G на рис. 2.34, *б*, $\gamma(G)=8-6+1=3$, т. е. после удаления трех определенных ребер он становится деревом. В примере это могут быть ребра (x_1, x_2), (x_2, x_4), (x_4, x_6). Тогда получим дерево T (рис. 2.35).

Заметим, что формула определения цикломатического числа графа верна для орграфов и мультиграфов. Из определения $\gamma(G)$ следует, что оно может быть только положительным или равным нулю. Граф (мультиграф) не имеет циклов тогда и только тогда, когда $\gamma(G)=0$. Очевидно, что граф (мультиграф) имеет единственный цикл тогда и только тогда, когда $\gamma(G)=1$.

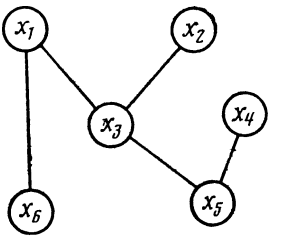


Рис. 2.35. Дерево T

Чтобы узнать, какие конкретно ребра нужно удалять для устранения циклов, необходимо каждый раз удалять ребро, которое разрушает хотя бы один цикл.

Раскраской вершин графа называется разбиение множества вершин графа на l непересекающихся классов (подмножеств)

$$X_1, X_2, \ldots, X_l;\ X=\bigcup_{i=1}^{l} X_i;\ X_i \cap X_j=\varnothing i,\ j \in I=\{1, 2, \ldots, l\},\ i \neq j,$$

таких, что внутри каждого подмножества X_i не должно содержаться смежных вершин. Если каждому подмножеству X_i поставить в соответствие определенный цвет, то вершины внутри этого подмножества можно окрасить в один цвет, вершины другого подмножества X_j — в другой цвет и так далее до раскраски всех подмножеств, причем раскраска выполняется таким образом, что смежные вершины окрашиваются в разные цвета. В этом случае граф называется l-раскрашиваемым. Наименьшее число подмножеств, на которое разбивается граф при раскраске, называется *хроматическим числом* $\mathrm{K}(G)$.

Очевидно, что полный граф K_n можно раскрасить только n цветами, следовательно, $\mathrm{K}(K_n)=n$. Для связного графа $G=(X, U)$ с $(n-1)\leqslant m\leqslant n(n-1)/2$ верхняя оценка хроматического числа

$$\mathrm{K}(G)=\left[\frac{3+\sqrt{9+8(m-n)}}{2}\right].$$

Нижней оценкой $\mathrm{K}(G)$ является число вершин в наибольшем полном подграфе графа G.

Хроматическое число обычно определяют аналитически с помощью методов линейного программирования (подробнее см. § 2.4). Пусть задан граф $G=(X, U)$, $|X|=n$, $|U|=m$, в виде матрицы инцидентности $I=\|i_{k,l}\|_{n\times m}$. Рассмотрим один из возможных методов определения $q=\mathrm{K}(G)$.

Будем считать, что

$$\xi_{j,p}=\begin{cases}1, & \text{если вершина } x_j \text{ окрашена цветом } p\ (p=1, 2, \ldots, q);\\ 0 & \text{в противном случае.}\end{cases}$$

Тогда можно составить следующую систему линейных соотношений:

$$\sum_{p=1}^{q}\xi_{j,p}=1,\ j=1, 2, \ldots, n; \tag{2.1}$$

$$\sum_{k=1}^{n} i_{k,l}\xi_{k,p}\leqslant 1,\ l=1, 2, \ldots, m; \tag{2.2}$$

$$\xi_{j,p}\geqslant 0,\ p=1, 2, \ldots, q. \tag{2.3}$$

Соотношение (2.1) говорит о том, что каждая вершина окрашена только одним цветом из множества всех цветов, система неравенств (2.2), (2.3) — что любые две смежные вершины графа не должны быть окрашены одним цветом. Проверим это. Например, если бы вершины x_a и x_b, образующие ребро u_c, были окрашены одним цветом, тогда $\xi_{a,p}=1$, $i_{a,c}=1$; $\xi_{b,p}=1$, $i_{b,c}=1$ и сумма $\sum_{k=1}^{n} i_{k,c}\xi_{kd}=i_{a,c}\xi_{a,p}+i_{b,c}\xi_{b,p}=2$, что противоречит неравенству (2.2). Следовательно, первые и вторые уравнения показывают, что каждому способу раскраски вершин графа q цветами при обязательном условии разноцветности смежных вершин соответствует система целых чисел $\xi_{j,p}$, удовлетворяющих условиям (2.1)—(2.3), и, наоборот, каждому целочисленному решению си-

стемы (2.1)—(2.3) соответствует один из способов раскраски. Наименьшее q, для которого система (2.1)—(2.3) имеет целочисленное решение, будет искомым хроматическим числом графа.

Важное практическое применение имеют частный вид l-раскрашиваемых графов, 2-раскрашиваемые или *двудольные (бихроматические)* графы. Обозначается двудольный граф $G_{n1,\,n2}=(X_1,\ X_2,\ U)$, причем $X_1 \cup X_2 = X$, $X_1 \cap X_2 = \varnothing$, а ребра U соединяют только подмножества X_1 и X_2 между собой. На рис. 2.36 показан пример двудольного графа

$$G_{n_1,\,n_2}=(X_1,\ X_2,\ U),\quad |X_1|=n_1=3,\quad |X_2|=n_2=2,\quad X_1=\{x_1,\ x_2,\ x_3\},$$
$$X_2=\{x_4,\ x_5\}.$$

Граф $K_{n1,\,n2}$ называется *полным двудольным графом,* если любая вершина $x_i \in X_1$ смежна каждой вершине $x_j \in X_2$ $(i \neq j)$. Например, при добавлении к графу $G_{n1,\,n2}$ на рис. 2.36 ребра $(x_3,\ x_4)$ он становится полным двудольным графом.

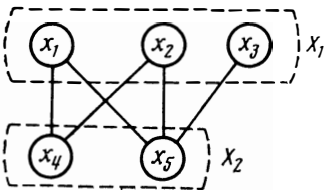


Рис. 2.36. Двудольный граф $G_{n_1,n_2}=(X_1,\ X_2,\ U)$

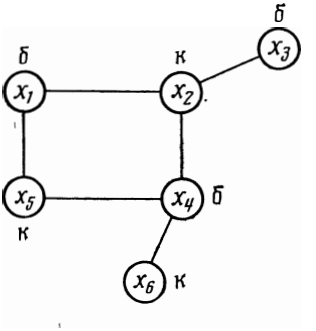


Рис. 2.37. Пример графа

Очевидно, что в графе $G_{n1,\,n2}=(X_1,\ X_2,\ U)$ подмножество X_1 можно раскрасить одним цветом, а подмножество X_2 — другим. Число ребер m полного двудольного графа определяется выражением $m=n_1 n_2$.

При разработке алгоритмов компоновки, размещения и трассировки схем часто возникает необходимость определения двудольности некоторого графа или выделения в этом графе, если он недвудолен, максимальных непересекающихся двудольных частей.

Теорема 2.5. Граф $G_{n1,\,n2}$ является двудольным тогда и только тогда, когда он не имеет простых циклов нечетной длины.

Рассмотрим теперь число внутренней устойчивости. Если в графе $G=(X,\ U)$ две любые вершины подмножества $X' \subseteq X$ не смежны, то оно называется *внутренне устойчивым.* Подмножество $\Psi_i \subseteq X$ графа $G=(X,\ U)$ называется максимальным внутренне устойчивым или независимым, если добавление к нему любой вершины $x_i \in X$ делает его не внутренне устойчивым. Подмножество Ψ_i будет независимым, если $\forall\, x_i \in \Psi_i\,(\Gamma x_i \cap \Psi_i = \varnothing)$. Независимые под-

множества различаются по числу входящих в них элементов. В произвольном графе G можно выделить семейство всех независимых подмножеств вида $\Psi=\{\psi_1, \psi_2, ..., \psi_s\}$. Независимые подмножества, содержащие наибольшее число элементов, называются предельными. Тогда число внутренней устойчивости $\eta(G)$ определяется мощностью предельного независимого подмножества

$$\eta(G)=|\max_{\psi_i\in\Psi}\psi_i|.$$

Число внутренней устойчивости $\eta(G)$ можно связать с хроматическим числом $K(G)$ следующим образом: $\eta(G)K(G)\geqslant n$.

Рассмотрим вопросы построения семейства независимых подмножеств на основе матрицы смежности графа или ее списка.

Идея алгоритма построения семейства $\Psi=\{\psi_1, \psi_2, ..., \psi_l\}$ заключается в следующем. В матрице смежности **R** графа определяется строка, имеющая наибольшее число ненулевых элементов. Если таких строк несколько, то выбирается любая. Для определенной таким образом строки i записывается выражение $c_i=(x_i\vee x_k\wedge x_b\wedge ... \wedge x_q)$. Здесь x_i — вершина графа, соответствующая выделенной строке i, а x_k, x_b, ..., x_q — вершины, смежные x_i. Далее в **R** исключается строка i, а также элементы с индексом i из всех оставшихся строк списка. Указанная операция соответствует выделению из графа G звездного подграфа с вершиной x_i. В **R** снова определяется строка **j** с наибольшим числом элементов и записывается выражение $c_j=(x_j\vee x_l\wedge x_r\wedge ... \wedge x_v)$. Процесс повторяется до тех пор, пока оставшиеся строки **R** будут пустыми. Затем составляется произведение $\Pi=c_ic_j ... c_k$, в котором раскрываются скобки, и в полученной сумме выполняется минимизация на основе операций булевой алгебры с учетом выражений

$$x_i^n=x_i;\ x_i\vee x_i\vee ... \vee x_i=x_i;\ x_i\vee 1=1;\ x_i\wedge x_i\wedge ... \wedge x_i=x_i;\ x_i\wedge 1=x_i.$$

Для выделения семейства Ψ необходимо в выражении, полученном после минимизации, для каждого элемента суммы найти не включенные в него вершины графа, которые образуют независимое подмножество.

Работу алгоритма покажем на графе, изображенном на рис. 2.37. Запишем матрицу смежности **R** графа в несколько измененном виде:

$$\mathbf{R}=\left\|\begin{array}{ll} x_1 & (x_2,\ x_5)\\ x_2 & (x_1,\ x_3,\ x_4)\\ x_3 & (x_2)\\ x_4 & (x_2,\ x_5,\ x_6)\\ x_5 & (x_1,\ x_4)\\ x_6 & (x_4) \end{array}\right\|.$$

В **R** выделим строки 2 и 4, которые содержат наибольшее число элементов. Выберем строку 2, удалим ее и элементы x_2 из остальных строк списка. Запишем элемент $c_2=(x_2\vee x_1\wedge x_3\wedge x_4)$.

Получим

$$\mathbf{R}'=\begin{array}{c|c}x_1 & (x_5)\\ x_3 & (\varnothing)\\ x_4 & (x_5,\ x_6)\\ x_5 & (x_1,\ x_4)\\ x_6 & (x_4)\end{array}.$$

В $\mathbf{R}$ определим строки 4 и 5. Выберем строку 4 и запишем $c_4=(x_4\vee x_5\wedge x_6)$.

Получим

$$\mathbf{R}''=\begin{array}{c|c}x_1 & (x_5)\\ x_3 & (\varnothing)\\ x_5 & (x_1)\\ x_6 & (\varnothing)\end{array}.$$

Определим строку 1 и запишем $c_1=(x_1\vee x_5)$. Получим

$$\mathbf{R}'''=\begin{array}{c|c}x_3 & (\varnothing)\\ x_5 & (\varnothing)\\ x_6 & (\varnothing)\end{array}.$$

На этом процесс преобразования матрицы смежности закончим. Сформируем произведение $П=c_2c_4c_1=(x_2\vee x_1\wedge x_3\wedge x_4)\wedge(x_4\vee x_5\wedge x_6)\wedge(x_1\vee x_5)$.

Раскрыв скобки и выполнив минимизацию на основе выше приведенных выражений, получим $П=(x_1\wedge x_2\wedge x_4)\vee(x_2\wedge x_4\wedge x_5)\vee(x_1\wedge x_2\wedge x_5\wedge x_6)\vee(x_2\wedge x^2_5\wedge x_6)\vee(x^2_1\wedge x_3\wedge x^2_4)\vee(x_1\wedge x_3\wedge x^2_4\wedge x_5)\vee(x^2_1\wedge x_3\wedge x_4\wedge x_5\wedge x_6)\vee(x_1\wedge x_3\wedge x_4\wedge x^2_5\wedge x_6)=(x_1\wedge x_2\wedge x_4)\vee(x_2\wedge x_4\wedge x_5)\vee(x_2\wedge x_5\wedge x_6)\vee(x_1\wedge x_3\wedge x_4)$.

Запишем $П$ в виде $П=K_1\vee K_2\vee K_3\vee K_4$. Записав вместо каждого слагаемого в полученной дизъюнкции недостающие вершины графа G, получим $\overline{П}=(x_3\wedge x_5\wedge x_6)\vee(x_1\wedge x_3\wedge x_6)\vee(x_1\wedge x_3\wedge x_4)\vee(x_2\wedge x_5\wedge x_6)$. Тогда семейство независимых подмножеств будет иметь вид $\Psi=\{\psi_1,\ \psi_2,\ \psi_3,\ \psi_4\}$, $\psi_1=\{x_3,\ x_5,\ x_6\}$, $\psi_2=\{x_1,\ x_3,\ x_6\}$, $\psi_3=\{x_1,\ x_3,\ x_4\}$, $\psi_4=\{x_2,\ x_5,\ x_6\}$. Число внутренней устойчивости $\eta(G)=3$.

При раскраске множества вершин графа G для каждой вершины $x_i\in X$ определяются множества K_i из $П$, в которые не входит x_i.

В нашем примере $x_1\notin K_2\vee K_3$; $x_2\notin K_4$; $x_3\notin K_1\vee K_2\vee K_3$; $x_4\notin K_3$; $x_5\notin K_1\vee K_4$; $x_6\notin K_1\vee K_2\vee K_4$. Заметим, что здесь K_i считаются булевыми переменными.

После этого составляется произведение $П'=(K_2\vee K_3)\wedge K_4\wedge(K_1\vee K_2\vee K_3)\wedge K_3\wedge(K_1\vee K_4)\wedge(K_1\vee K_2\vee K_4)$, в котором раскрываются скобки и выполняется минимизация. В результате $П'=K_3\wedge K_4$.

Число элементов K_i в конъюнктивном члене $П'$ определяет число различных цветов, а каждый K_i — подмножество вершин, которые можно окрасить одним цветом. Тогда граф можно раскрасить двумя цветами, например красным и белым. В нашем

примере $K_3=\{x_2, x_5, x_6\}$ окрашивается красным цветом; $K_4=\{x_1, x_3, x_4\}$ — белым.

Заметим, что если Π' будет состоять из нескольких дизъюнктивных членов, то каждый из них будет определять один из возможных способов раскраски графа.

По аналогии с числом внутренней устойчивости введем число внутренней полноты, характеризующее максимальное число вершин в подмножествах взаимно смежных вершин. Подмножество из максимального числа взаимно смежных вершин называется *кликой.* Мощность клики, содержащей наибольшее число вершин графа, определит число *внутренней полноты* $Q(G)=|\max\limits_{Q_i\in Q} Q_i|$, где $Q(G)$ — семейство клик графа G.

В графе на рис. 2.38 имеем семейство из трех клик: $Q_1=\{x_1, x_4, x_5\}$; $Q_2=\{x_1, x_4, x_2\}$; $Q_3=\{x_2, x_3, x_4\}$, а число внутренней полноты $Q(G)=3$.

Подмножество вершин $\varphi_i\subseteq X$ графа G называется *внешне устойчивым,* если $\forall\, x_i \notin \varphi_i(\Gamma x_i\cap\varphi_i\neq\varnothing)$. *Внешне устойчивое подмножество называется минимальным,* если удаление из него произвольной вершины делает его не внешне устойчивым. В каждом графе можно выделить семейство внешне устойчивых подмножеств $\varphi=\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$. Неформально минимальное внешне устойчивое подмножество (его еще называют доминирующим) — это подмножество, состоящее из наименьшего числа вершин графа, смежных всем другим вершинам. Мощность доминирующего подмножества, имеющего наименьшее число вершин, называется числом внешней устойчивости

$$\varphi(G)=|\min_{\varphi_i\in\varphi}\varphi_i|.$$

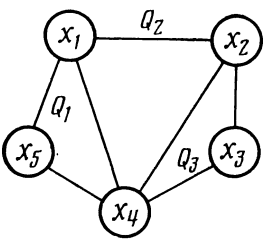


Рис. 2.38. Граф, содержащий три клики: Q_1, Q_2, Q_3

В графе на рис. 2.38 имеем семейство φ, состоящее из пяти доминирующих подмножеств $\varphi=\{\varphi_1, \ldots, \varphi_5\}$: $\varphi_1=\{x_4\}$; $\varphi_2=\{x_1, x_3\}$; $\varphi_3=\{x_2, x_5\}$; $\varphi_4=\{x_3, x_5\}$; $\varphi_5=\{x_1, x_2\}$; $\varphi(G)=1$.

В некоторых случаях подмножество $A\subseteq X$ может быть одновременно внутренне и внешне устойчивым. Такие подмножества называются ядрами графа. Например, подмножество $\{x_1, x_3\}$ графа (см. рис. 2.38) является ядром. Существуют графы, содержащие ядра и не содержащие их.

При конструировании ЭА многие задачи контроля схем могут быть сведены к различным тождественным преобразованиям графов этих устройств. Тождественные преобразования графов путем переобозначения вершин и ребер, приводят к получению *изоморфных графов.*

Два графа $G=(X, U)$ и $G'=(X', U')$ называются изоморфными, если можно установить взаимно-однозначное соответствие

$X \leftrightarrow X'$, $U \leftrightarrow U'$ такое, что если $(x_i, x_j) \in X \leftrightarrow (x'_i, x'_j) \in X'$, то ребро $u = (x_i, x_j) \in U \leftrightarrow u' = (x'_i, x'_j) \in U'$.

Очевидно, что изоморфизм, есть отношение эквивалентности на графах. Графы G и G', изображенные на рис. 2.39, *а, б,* изоморфны. Подстановка

$$t = \begin{pmatrix} x_1, & x_2, & x_3, & x_4, & x_5, & x_6 \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow \\ x'_1, & x'_3, & x'_5, & x'_2, & x'_4, & x'_6 \end{pmatrix}$$

устанавливает соответствие графов G и G'. Для перехода от графа G' к графу G необходимо применить обратную подстановку t^{-1} к графу G':

$$t^{-1} = \begin{pmatrix} x'_1, & x'_2, & x'_3, & x'_4, & x'_5, & x'_6 \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow \\ x_1, & x_4, & x_2, & x_5, & x_3, & x_6 \end{pmatrix}.$$

Если изоморфные преобразования проводятся с графом, заданным матрицей смежности, то они сводятся к перестановке местами соответствующих строк и столбцов. В общем случае для определения изоморфизма графов необходимо сделать $n!$ сравнений или перестановок строк и столбцов матрицы.

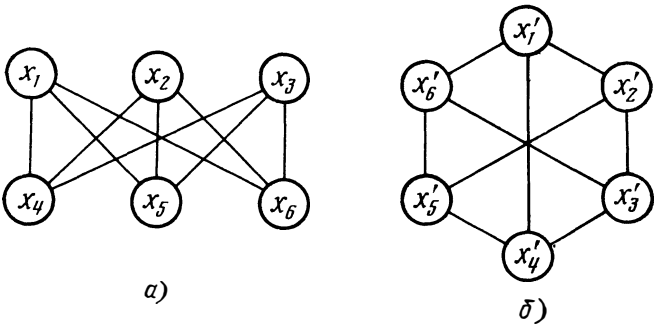


Рис. 2.39. Граф G (полный двудольный граф $K_{3,3}$) (*а*), граф G^1 (*б*)

Например, матрицы смежности графов G и G' на рис. 2.39, *а, б* соответственно имеют вид:

$$\mathbf{R} = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 2 & 0 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 0 & 1 & 1 & 1 \\ 4 & 1 & 1 & 1 & 0 & 0 & 0 \\ 5 & 1 & 1 & 1 & 0 & 0 & 0 \\ 6 & 1 & 1 & 1 & 0 & 0 & 0 \end{array};$$

$$\mathbf{R}'=\begin{array}{c|cccccc} & 1' & 2' & 3' & 4' & 5' & 6' \\ 1' & 0 & 1 & 0 & 1 & 0 & 1 \\ 2' & 1 & 0 & 1 & 0 & 1 & 0 \\ 3' & 0 & 1 & 0 & 1 & 0 & 1 \\ 4' & 1 & 0 & 1 & 0 & 1 & 0 \\ 5' & 0 & 1 & 0 & 1 & 0 & 1 \\ 6' & 1 & 0 & 1 & 0 & 1 & 0 \end{array}.$$

Если к матрице $\mathbf{R}$ применить подстановку t^{-1}, то получим матрицу $\mathbf{R}_t$, тождественную $\mathbf{R}'$:

$$\mathbf{R}_t=\begin{array}{c|cccccc} & 1 & 4 & 2 & 5 & 3 & 6 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 4 & 1 & 0 & 1 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 1 & 0 & 1 \\ 5 & 1 & 0 & 1 & 0 & 1 & 0 \\ 3 & 0 & 1 & 0 & 1 & 0 & 1 \\ 6 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}.$$

При покрытии функциональной схемы набором стандартных схем (модулей) или при решении любой задачи типизации необходимо устанавливать изоморфизм между графом G и какой-либо частью другого графа G'. Задачи такого рода называются *задачами изоморфного вложения*. Известно, что задача изоморфного вложения графов является NP-полной.

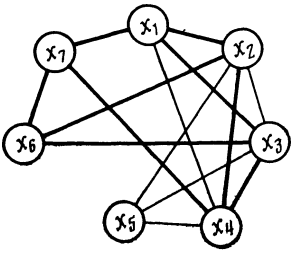


Рис. 2.40. Граф G

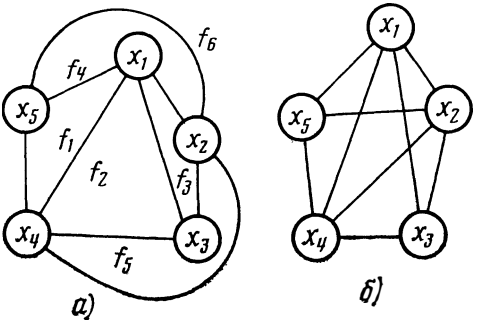


Рис. 2.41. Плоский граф (*а*) и изоморфный ему планарный граф (*б*)

Пусть дан граф, показанный на рис. 2.40. Можно показать, что подграф $H=(X', U')$, $|X'|=6$, $|U'|=9$, $X'=\{x_7, x_2, x_3, x_4, x_1, x_6\}$, выделенный жирными линиями, изоморфен графу на рис. 2.39, *а*. Следовательно, последний изоморфно вложим в граф G на рис. 2.40.

В общем случае для установления изоморфного вложения графа G_1 с n_1 вершинами в граф G_2 с n_2 вершинами требуется $C_{n_2}^{n_1} n_1!$

сравнений. При конструировании ЭА к топологическому чертежу часто предъявляется требование расположения проводников схемы на плоскости без пересечений. В этой связи возникает задача определения планарности графа.

Граф $G=(X, U)$ называется *плоским,* если он расположен на плоскости таким образом, что ребра имеют общие точки лишь в вершинах. Граф, изоморфный плоскому, расположенный на плоскости и имеющий пересечения ребер, называется *планарным.* На рис. 2.41, *а* показан плоский граф, а на рис. 2.41, *б* — изоморфный ему планарный граф.

Область плоскости, ограниченная ребрами плоского графа, внутри которой нет ни вершин, ни ребер, называется гранью. Ребра грани образуют простой цикл. Считают, что плоский граф имеет всегда одну бесконечную грань, не ограниченную ребрами. По сути дела это часть внешней плоскости, окружающая граф. Существует формула Эйлера, позволяющая установить связь между числами граней, вершин и ребер плоского графа $n-m+f=2$, где f — число граней плоского графа. Для графа G на рис. 2.41, *а* получим $5-9+6=2$. Здесь грань f_6 является бесконечной гранью. Используя формулу для определения цикломатического числа графа, получаем $f=\gamma(G)+1$.

Минимальное число ребер, которое необходимо удалить из графа, чтобы он стал планарным, называется *числом планарности* и обозначается $\Theta(G)$. Для полного графа K_n с $n\geqslant 4$ $\Theta(K_n)=(n-3)(n-4)/2$. Из формулы следует, что $\Theta(K_n)=0$, когда $n=4$; $\Theta(K_5)=1$, следовательно, чтобы полный граф K_5 стал планарным, из него надо удалить одно ребро.

Очевидно, что верхней оценкой числа планарности является цикломатическое число $\gamma(G)$, так как граф, не имеющий ни одного цикла, всегда планарен. Для полных графов $\Theta(K_n)=\gamma(K_n)-2n+5$. Методы определения планарности связаны с нахождением минимальных раскрасок графа. Известна теорема о том, что каждый планарный граф может быть раскрашен пятью красками, т. е. $K(G)\leqslant 5$. Одной из основных гипотез теории графов является гипотеза о четырех красках. Любой планарный граф может быть раскрашен четырьмя красками.

Доказательство этой гипотезы, основанное на переборе на ЭВМ огромного числа случаев, предложили Аппель и Хайкен.

Определение планарности графа можно производить на основе различных критериев. Рассмотрим два из них.

Пусть задан граф $G=(X, U)$. Подразбиением ребра $U_k=(x_i, x_j)$ называется замена его двумя ребрами $u_p=(x_i, x_p)$ и $u_q=(x_p, x_j)$ с введением новой вершины x_p. Два графа называются гомеоморфными, если они обладают изморфными подразбиениями.

Теорема 2.6. (Понтрягина—Куратовского). Граф планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных полному графу K_5 (см. рис. 2.42, *а*) и полному двудольному графу $K_{3,3}$ (рис. 2.39, *а*).

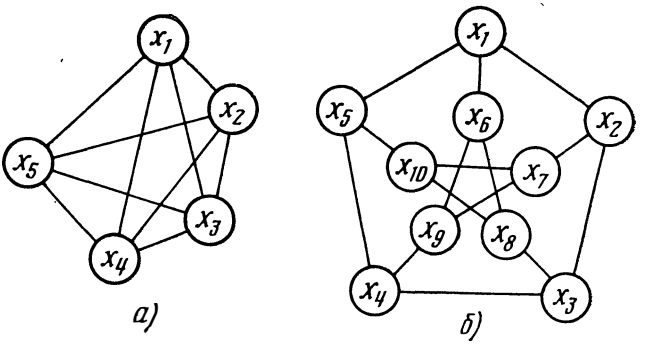


Рис. 2.42. Полный граф $K_{3,5}$ (*а*), граф, стягиваемый в граф K_5 (*б*)

Операция, обратная подразбиению ребра, называется *стягиванием*. В теории графов под элементарным стягиванием понимается замена двух смежных вершин x_i и x_j новой вершиной x_k, смежной вершинам графа, которые были смежны x_i и x_j. Граф $G_1=(X_1, U_1)$ называют стягиваемым к графу $G_2=(X_2, U_2)$, если G_2 можно получить из G_1 на основе некоторой последовательности элементарных стягиваний. Например, граф на рис. 2.42, *б* стягивается в граф K_5 путем элементарного стягивания в новую вершину x_i пяти ребер (x_1, x_6), (x_2, x_7), (x_3, x_8), (x_4, x_9), (x_5, x_{10}).

Теорема 2.7 (Харари — Татта). Граф планарен, если он не содержит подграфов, стягиваемых к графу K_5 или к графу $K_{3,3}$.

Известно, что любой граф может быть изображен без пересечений в трехмерном евклидовом пространстве. Пусть задан неориентированный граф с петлями. Тогда для каждой петли можно поставить в соответствие окружность, проходящую через ее вершину, а для каждого ребра, соединяющего две вершины, сопоставить в соответствующей плоскости полуокружность, приходящую через эти вершины. Никакие из этих кривых не могут пересечься, так как они находятся в разных плоскостях.

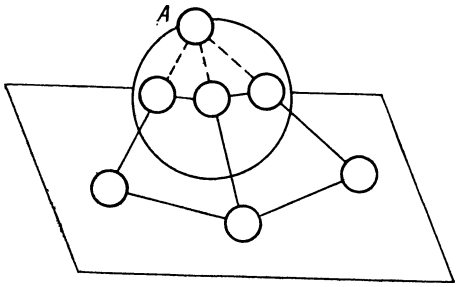


Рис. 2.43. Пример стереографической проекции из точки A

Говорят, что граф $G=(X, U)$ укладывается в пространстве или может быть уложен в нем, если он изоморфен графу $G'=(X', U')$, изображенному в этом пространстве с помощью вершин $X \leftrightarrow X'$ и жордановых кривых, соответствующих ребрам $U \leftrightarrow U'$, причем ребра не пересекаются друг с другом.

Теорема 2.8. Граф планарен тогда и только тогда, когда он укладывается на поверхность сферы.

Предположим, что граф G уложен на поверхности сферы. Поместим сферу на плоскость таким образом, чтобы ее «верхняя точка» (точка, противоположная точке касания) не соответствовала ни одной вершине графа и не лежала ни на каком его ребре. Тогда искомое плоское представление планарного графа можно получить стереографической проекцией из верхней точки, как, например, показано на рис. 2.43.

При конструировании многослойного монтажа необходимо решать задачи планарного размещения графа в нескольких плоскостях с переходами по соответствующим вершинам графа.

Граф G можно расположить в z плоскостях, если существует z планарных суграфов $G_1, G_2, \ldots, G_z$ таких, что $\bigcup_{i=1}^{z} G_i = G$. Наименьшее число плоских суграфов, на которые можно разбить граф, называется *толщиной графа* и обозначается $\Omega(G)$.

Очевидно, что толщина планарного графа $\Omega(G)=1$, а толщина графов K_5 и $K_{3,3}$ равна двум. Толщина произвольного графа удовлетворяет неравенствам

$$\Omega(G) \geqslant \left\{\frac{m}{3n-6}\right\}; \quad \Omega(G) \geqslant \left[\frac{m+3n-7}{3n-6}\right],$$

где $\{d\}$ — наименьшее целое число не меньше чем d, а $[d]$ — наибольшее целое число, не превосходящее d.

Толщина полных графов удовлетворяет неравенству $\Omega(K_n) \geqslant \geqslant [\frac{n+7}{6}]$.

Предполагается, что толщина K_n не превышает величины $n/4$.

Иногда в графе требуется выделять не пересекающиеся по ребрам полные непланарные подграфы. Максимальное число таких подграфов в G называется *шероховатостью* или крупностью и обозначается $v(G)$.

Для K_n при $n=3p$

$$v(K_n) = \begin{cases} C^2_p, \text{ если } p \leqslant 5; \\ C^2_p + C^5_p, \text{ если } p \geqslant 10; \end{cases}$$

C^j_i — это число сочетаний из i по j; при $n=3\ p+1\ v(K_n)=C^2_p+$ $+[p/3]$ и при $n=3p+2\ v(K_n)=C^2_p+[(14p+1)/5]$.

Число попарных пересечений ребер (иногда называется числом скрещиваний) на плоскости обозначается $p(G)$ и для полного графа определяется выражением

$$p(K_n) \leqslant \begin{cases} (n-1)^2(n-3)^2/64, \text{ если } n \text{ нечетно}; \\ n(n-2)^2(n-4)/64, \text{ если } n \text{ четно}, \end{cases}$$

или

$$p(K_n) \leqslant \frac{p(K_{n-1})}{1-8/2n-1+(-1)^n}.$$

Зная, что $p(K_5)=1$, можно определить минимальное число пересечений для полного графа с любым числом вершин.

Заметим, что если граф G связный и плоский, то и двойственный ему граф G_s также будет плоским и связным, причем число вершин, ребер и граней этих графов будет связано следующими соотношениями: $n_s=f$; $r_s=r$; $f_s=n$, где n_s, r_s, f_s — число вершин, ребер и граней двойственного графа G_s.

Часто при описании соединений в схемах необходимо учитывать направление прохождения сигнала.

Ориентированный граф будем обозначать $D=(X, U)$ и называть графом. *Маршрутом* графа D считается чередующаяся последовательность вершин и дуг $(x_0, u_1, x_1, \ldots, u_n, x_n)$, в которой каждая дуга u_i есть кортеж $u_i=\langle x_{i-1}, x_i\rangle$. Маршрут, в котором все вершины различны, называется *путем*. Замкнутый маршрут, у которого все вершины различны, за исключением первой и последней, называется контуром. Если существует путь из x_i в x_j, то говорят, что x_j достижима из x_i.

Граф D называется сильно связным, если любые две его вершины взаимно достижимы.

Граф G, полученный из графа D заменой каждой дуги $u_i=\langle x_k, x_l\rangle$ на соответствующее ребро $u_i=(x_k, x_l)$, т. е. устранением стрелок, называется основанием D.

Два орграфа называются изоморфными, если можно установить изоморфизм между их основаниями при сохранении порядка стрелок на каждой дуге.

Матрицей смежности графа D называется матрица $\mathbf{R}(D)=\|r_{i,j}\|_{n\times n}$, причем

$$r_{i,j}=\begin{cases}1, \text{ если } \langle x_i, x_j\rangle \text{ — дуга } D;\\ 0 \text{ в противном случае.}\end{cases}$$

Для графа D, показанного на рис. 2.44, матрица $\mathbf{R}(D)$ имеет вид

$$\mathbf{R}(D)=\begin{array}{c|ccccc|c} & 1 & 2 & 3 & 4 & 5 & \rho(x_j)\\ 1 & 0 & 1 & 0 & 0 & 1 & 2\\ 2 & 0 & 0 & 1 & 1 & 0 & 2\\ 3 & 0 & 0 & 0 & 0 & 0 & 0\\ 4 & 1 & 1 & 1 & 0 & 0 & 3\\ 5 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}.$$

$$\rho^-(x_j)\ \boxed{1\,|\,2\,|\,2\,|\,2\,|\,1\,|}$$

Так как $r_{i,j}\neq r_{j,i}$, то матрица не симметрична относительно главной диагонали.

Дуга $u_i=\langle x_i, x_j\rangle$, считается положительно инцидентной ее конечной вершине x_j. Число дуг, положительно инцидентных вершине x_j, называется *полустепенью захода* и обозначается $\rho^+(x_j)$. Число дуг, отрицательно инцидентных x_j, т. е. выходящих из x_j, называется полустепенью исхода и обозначается через $\rho^-(x_j)$.

Очевидно, что $\rho(x_j)=\rho^+(x_j)+\rho^-(x_j)$. Так как любая дуга положительно инцидентна некоторой вершине x_j и также отрицательно инцидентна той же самой вершине x_j, то

$$\sum_{x_j\in X}\rho^+(x_j)=\sum_{x_j\in X}\rho^-(x_j)=|U|.$$

Из матрицы $\mathbf{R}(D)$ видно, что суммы элементов по строкам равны полустепеням захода вершин D, а сумма элементов по столбцам — полустепеням исхода.

Неорграф G называется ориентируемым, если каждое его ребро можно ориентировать так, что полученный граф D будет сильно связным. Такой процесс обычно называют заданием ориентации графа G. Очевидно, что произвольный эйлеров граф может быть ориентируем, так как достаточно пройти по любой эйлеровой цепи, ориентируя ребра графа в направлении движения.

Граф D называется эйлеровым, если в нем существует замкнутая цепь, содержащая каждую его дугу.

Необходимым условием существования эйлерова орграфа является его сильная связность.

Связный граф $D=(X, U)$ является эйлеровым, когда $\forall\ x_i\in X(\rho^+(x_i)=\rho^-(x_i))$.

Орграф называется гамильтоновым, если в нем существует контур, содержащий каждую вершину орграфа.

Теорема 2.9. Пусть D — сильно связный граф. Если $\forall\ x_i\in X(\rho^+(x_i)\geqslant n/2$ и $\rho^-(x_i)\geqslant n/2)\Rightarrow D$, то D — гамильтонов граф. Элементы матрицы инцидентности $\mathbf{I}(D)$ графа D принимают значения 0, +1, —1. Элемент равен 0, если вершина не инцидентна дуге, +1, если дуга ориентирована от вершины, и —1, если дуга ориентирована к вершине.

Для графа D на рис. 2.44 матрица инцидентности имеет вид

$$\mathbf{I}(D)=\begin{array}{c}\\ x_1\\ x_2\\ x_3\\ x_4\\ x_5\end{array}\begin{array}{c}\begin{array}{cccccccc}u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 & u_8\end{array}\\ \left\|\begin{array}{cccccccc}+1 & 0 & 0 & 0 & +1 & -1 & 0 & 0\\ -1 & +1 & 0 & 0 & 0 & 0 & +1 & -1\\ 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & +1 & -1 & 0 & +1 & -1 & +1\\ 0 & 0 & 0 & +1 & -1 & 0 & 0 & 0\end{array}\right\|\end{array}.$$

Рассмотрим метод Мальгранжа разбиения графа D на максимально связные подграфы.

Предварительно определим прямое и обратное транзитивные замыкания. Прямым транзитивным замыканием Γ^+x_i называют подмножество вершин $X'\subseteq X$, в которые можно попасть из вершины x_i по некоторому пути. Обозначают его так: $\Gamma^+x_i=\{x_i\}\cup\Gamma^+x_i\cup\Gamma^{+2}x_i\cup\Gamma^{+3}x_i\cup\ldots$ Здесь $\Gamma^{+2}x_i=\Gamma^+\{\Gamma x_i\}$, а $\Gamma^{+3}x_i=\Gamma^+\{\Gamma^{+2}x_i\}=\Gamma^+\{\Gamma^+\{\Gamma^+x_i\}\},\ldots$ Обратным транзитивным замыканием называют подмножество вершин, из которых можно попасть в x_i по некоторому пути, обозначается Γ^-x_i.

Например, для графа $D=(X, U)$ на рис. 2.45 определим прямое и обратное транзитивные замыкания для вершины x_7: $\Gamma^+x_7=\{x_7\}\cup\Gamma^+x_7\cup\Gamma^{+2}x_7\cup\Gamma^{+3}x_7$, $\Gamma^+x_7=\{x_7, x_4, x_6\}$, $\Gamma^{+2}x_7=\Gamma^+\{\Gamma^+x_7\}=\Gamma^+\{x_7, x_4, x_6\}=\{x_7, x_4, x_6, x_2, x_5\}$, $\Gamma^{+3}x_7=\Gamma^+\{\Gamma^{+2}x_7\}=\Gamma^+\{x_7, x_4, x_6, x_2, x_5\}=\{x_7, x_6, x_4, x_5, x_1, x_2, x_3\}$; $\Gamma^-x_7=\{x_7\}\cup\Gamma^{-1}x_7\cup\Gamma^{-2}x_7$; $\Gamma^-x_7=\{x_7, x_2\}$, $\Gamma^{-2}x_7=\Gamma^-\{\Gamma^-x_7\}=\{x_7, x_2, x_4\}$, тогда $\Gamma^-x_7=\{x_2, x_4, x_7\}$, а $\Gamma x_7=\{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}=X$.

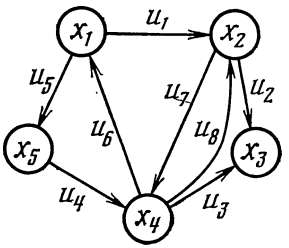


Рис. 2.44. Граф D

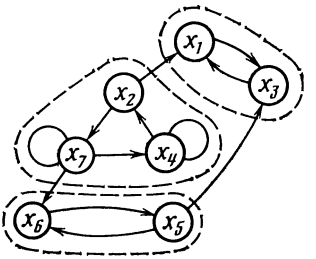


Рис. 2.45. Разложение графа на максимально связные подграфы

Основная идея алгоритма заключается в следующем. Выбирается произвольная вершина $x_i\in X$ в графе D и для нее определяется Γ^+x_i, Γ^-x_i и $C(x_i)=\Gamma^+x_i\cap\Gamma^-x_i$. Далее выбирается вершина $x_j\notin c(x_i)$, и процесс продолжается аналогично, пока возможно.

Рассмотрим работу алгоритма на примере графа $D=(X, U)$, заданного матрицей смежности

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	Γ^+x_1
x_1	0	0	1	0	0	0	0	0
x_2	1	0	0	0	0	0	1	—
x_3	1	0	0	0	0	0	0	1
x_4	0	1	0	1	0	0	0	—
x_5	0	0	1	0	0	1	0	—
x_6	0	0	0	0	1	0	0	—
x_7	0	0	0	1	0	1	1	—

$\mathbf{R}=$

Γ^-x_1	0	1	1	2	2	3	3

Столбец Γ^+x_1 строится следующим образом. В клетке столбца против стрелки x_1 ставится 0. В клетку столбца против строки x_3 ставится 1, так как строка x_1 содержит I на месте x_3. Строка x_3 содержит I только на уже отмеченном месте x_1. Тогда в оставшиеся клетки столбца Γ^+x_1 проставляются знаки «—». Числа в клетках Γ^+x_1 — это количество путей из вершины x_1 в другие вершины графа D. Тогда $\Gamma^+x_1=\{x_1, x_3\}$.

Аналогично заполняются клетки строки $\Gamma^{-}x_1$. Против столбца x_1 в клетке $\Gamma^{-}x_1$ ставится 0. В клетках строки $\Gamma^{-}x_1$ против x_2 и x_3 ставится 1, так как столбец имеет I против x_2 и x_3. В столбце x_2 имеется 1 против строки x_4, следовательно, в клетку $\Gamma^{-}x_1$ против x_4 записывается 2. В столбце x_3 имеем единицы против строк x_1 и x_5. Строка x_1 уже рассмотрена, а в клетке $\Gamma^{-}x_1$, расположенной против x_5, ставится 2. Далее аналогично заполняются оставшиеся клетки $\Gamma^{-}x_1$. Тогда $\Gamma^{-}x_1=\{x_1, x_2, x_3, x_5, x_6, x_7\}$, $C(x_1)=\Gamma^{+}x_1\cap\Gamma^{-}x_1=\{x_1, x_3\}$. Теперь удалим из графа D вершины x_1 и x_3. Получим матрицу $\mathbf{R}'$ оставшегося графа D':

$$\mathbf{R}'=\begin{array}{c|ccccc||c}
 & x_2 & x_4 & x_5 & x_6 & x_7 & \Gamma^{+}x_2 \\
\hline
x_2 & 0 & 0 & 0 & 0 & 1 & 0 \\
x_4 & 1 & 1 & 0 & 0 & 0 & 2 \\
x_5 & 0 & 0 & 0 & 1 & 0 & 3 \\
x_6 & 0 & 0 & 1 & 0 & 0 & 2 \\
x_7 & 0 & 1 & 0 & 1 & 1 & 1 \\
\hline
\Gamma^{-}x_2 & 0 & 1 & — & — & 2 &
\end{array}.$$

Аналогично, заполняя клетки $\Gamma^{+}x_2$ и $\Gamma^{-}x_2$, определяем $\Gamma^{+}x_2=\{x_2, x_4, x_5, x_6, x_7\}$, $\Gamma^{-}x_2=\{x_2, x_4, x_7\}$. Тогда $C(x_2)=\Gamma^{+}x_2\cap\Gamma^{-}x_2=\{x_2, x_4, x_7\}$. Снова удалим из D' вершины x_2, x_4, x_7, получим граф D''. Запишем его матрицу смежности

$$\mathbf{R}''=\begin{array}{c|cc||c}
 & x_5 & x_6 & \Gamma^{+}x_5 \\
\hline
x_5 & 0 & 1 & 0 \\
x_6 & 1 & 0 & 1 \\
\hline
\Gamma^{-}x_5 & 0 & 1 &
\end{array}.$$

Получим $\Gamma^{+}x_5=\{x_5, x_6\}$, $\Gamma^{-}x_5=\{x_5, x_6\}$, $C(x_5)=\{x_5, x_6\}$. Следовательно, в результате работы алгоритма получили разложение графа на три части, показанные на рис. 2.45 штриховыми линиями.

Заметим, что любой неорграф G можно перевести в орграф путем замены каждого ребра двумя противоположно направленными дугами. В этой связи многие результаты для неорграфов могут быть интерпретированы на орграфы.

В описанных выше графах использовались бинарные отношения на множестве вершин. Заметим, что в общем случае на множестве вершин X графа $G=(X, U)$ можно задать K-ные отношения. Такое обобщение понятия графа позволяет строить объекты, в которых каждое ребро может соединять не только две вершины, но и любое подмножество множества вершин.

Пусть $X=\{x_1, x_2, \ldots, x_n\}$ — конечное множество и $E=\{l_1, l_2, \ldots, l_m\}$ — семейство подмножеств X. Говорят, что семейство E — гиперграф на множестве X, если $E\neq\varnothing$ и $\bigcup\limits_{i\in I} l_i=X$, $I=\{1, 2, \ldots, m\}$. *Объект* $H=(X, E)$ будем считать *гиперграфом*, если он состоит из множества вершин X и множества ребер E, причем каждое ребро $l_i\in E$ представляет собой некоторое подмножество

множества вершин, т. е. $l_i \subseteq X$. Если $\forall\, l_i \in E(|l_i|=2)$, то гиперграф H является графом G без изолированных вершин. На рис. 2.46,*а* показан пример гиперграфа $H=(X, E)$, $|X|=7$, $|E|=5$. Заметим, что ребро l_5 с $|l_5|=1$ есть петля. Ребрами гиперграфа яв-

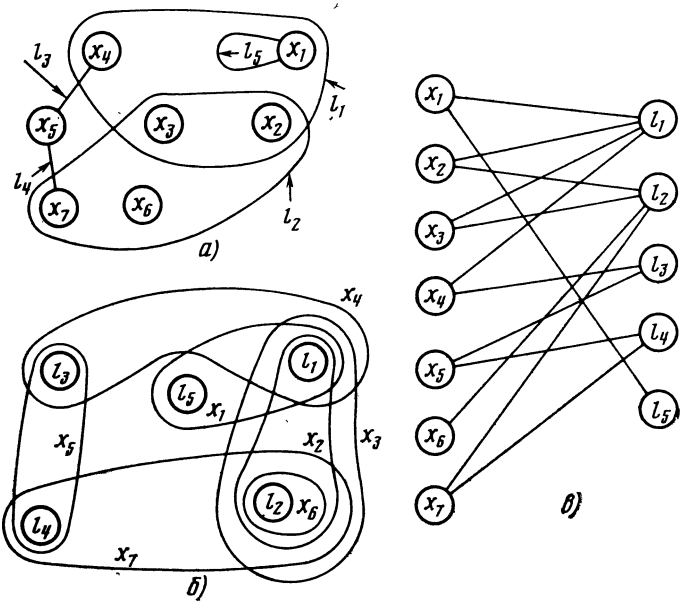


Рис. 2.46. Гиперграф $H=(X, E)$ (*а*), двойственный ему гиперграф H^* (*б*) и граф Кёнига $K(H)$ (*в*)

ляются: $l_1=\{x_1, x_2, x_3, x_4\}$; $l_2=\{x_2, x_3, x_6, x_7\}$; $l_3\{x_5, x_4\}$; $l_4=\{x_5, x_7\}$; $l_5=\{x_1\}$. В гиперграфе $H=(X, E)$ две вершины считаются смежными, если существует ребро l_i, содержащее эти вершины, два ребра считаются смежными, если их пересечение непустое подмножество.

Матрицей инцидентности гиперграфа H называется матрица $\mathbf{I}(H)=\|h_{i,j}\|_{m\times n}$, причем

$$h_{i,\,j}=\begin{cases}1, & \text{если } x_j\in l_i;\\ 0, & \text{если } x_j\notin l_i.\end{cases}$$

Для гиперграфа H на рис. 2.46,*а* матрица инцидентности примет вид

$$\mathbf{I}(H)=\begin{array}{c} \\ l_1\\ l_2\\ l_3\\ l_4\\ l_5\end{array}\begin{array}{c}\begin{array}{ccccccc}x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7\end{array}\\ \left\|\begin{array}{ccccccc}1 & 1 & 1 & 1 & 0 & 0 & 0\\ 0 & 1 & 1 & 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 1\\ 1 & 0 & 0 & 0 & 0 & 0 & 0\end{array}\right\|\end{array}.$$

Любому гиперграфу $H=(X, E)$ соответствует гиперграф $H^*=(E, X)$, вершинами которого являются ребра $E=\{l_1, l_2, \ldots, l_m\}$, а ребрами — вершины $x_1, x_2, \ldots, x_n$. Гиперграф H^* называется двойственным H. Матрица инцидентности гиперграфа H^* есть транспонированная матрица гиперграфа H. Если два ребра l_i, $l_j \in E$ в H смежны, то соответствующие вершины l_i, $l_j \in E$ в H^* также смежны, то же выполняется и для вершин x_i, $x_j \in X$ в H, когда они становятся ребрами H^*. Гиперграф H^*, двойственный H, показан на рис. 2.46,*б*,*а*, его матрица инцидентности запишется в виде

$$\mathbf{I}(H)^* = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccc} l_1 & l_2 & l_3 & l_4 & l_5 \end{array} \\ \left\| \begin{array}{ccccc} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{array} \right\| \end{array}.$$

В гиперграфе $H=(X, E)$ цепь длины q определяется как последовательность вида $s(H)=x_1, l_1, x_2, l_2, \ldots, l_q, x_{q+1}$. Если $q>1$ и $x_{q+1}=x_1$, то цепь с такими параметрами называется циклом длины q.

Если H-гиперграф с n вершинами, m ребрами и k связными компонентами, то он является деревом, когда

$$\sum_{i=1}^{m} (|l_i|-1)=n-k.$$

При решении задач автоматизации конструирования возникает необходимость сопоставлять гиперграфу граф.

Гиперграф $H=(X, E)$ можно представить в виде двудольного графа Кёнига $K(H)=(X \cup E, V)$, где X — первое подмножество вершин двудольного графа, соответствующее множеству вершин гиперграфа H; E — второе подмножество вершин двудольного графа, соответствующее множеству ребер гиперграфа H; V — множество ребер двудольного графа, устанавливающее смежность между двумя множествами вершин X и E в графе $K(H)$ в соответствии с инцидентностью вершин и ребер гиперграфа H.

Заметим, что в двудольном графе $K(H)$ вершины внутри подмножеств X и E не смежны, причем вершины $x_i \in X$ и $l_j \in E$ в $K(H)$ смежны тогда и только тогда, когда в гиперграфе H вершина x_i принадлежит ребру l_j. На рис. 2.46,*в* приведен граф Кёнига для гиперграфа H.

Отметим, что не только любой конечный гиперграф представляется в виде $K(H)$, но и каждый двудольный граф является представлением некоторого гиперграфа.

Под раскраской вершин гиперграфа $H=(X, E)$ понимается некоторое разбиение множества его вершин на классы, не содер-

жащие пустого подмножества, когда инцидентные вершины должны быть окрашены в разные цвета, причем каждому классу $X_i (X_1 \cup X_2 \cup \ldots X_q = X;\ X_1 \cap X_2 \cap \ldots \cap X_q = \varnothing)$ отвечает определенный цвет, в который окращены вершины, а q соответствует количеству различных цветов, которые можно использовать для раскраски.

В некоторых случаях для построения моделей схем используются расплывчатые графы и гиперграфы.

Граф $G = (X, U)$ *называется расплывчатым*, если для каждой вершины $x_i \in X$ множество U является расплывчатым множеством. Множество U характеризуется *функцией принадлежности* μ_u, принимающей значения из отрезка $[0,1]$. Значение $\mu(u)$ показывает степень принадлежности ребра u к множеству U. Очевидно, что если $\mu_u(x)$ для любых x, $y \in X$ принимает значение 0 или 1, то расплывчатый граф G становится обыкновенным. Для расплывчатого гиперграфа функция принадлежности определяется так же, как и для расплывчатого графа.

Любая логическая или принципиальная схема ЭВА и РЭА состоит из набора элементов, заданным образом соединенных между собой. Поэтому схему можно рассматривать как некоторое множество элементов $X = \{x_1, x_2, \ldots, x_n\}$, соединенных между собой цепями из множества $E = \{l_1, l_2, \ldots, l_m\}$. Такое представление схемы обычно называют *схемой соединений*. На рис. 2.47 показан условный фрагмент схемы. Каждый элемент схемы имеет некоторое множество соединительных выводов, которые будем называть множеством контактов и обозначать $C = \{c_1, c_2, \ldots, c_s\}$. Кроме контактов элементов в схеме имеются внешние контакты C_0, которые осуществляют связь рассматриваемой схемы с другими схемами.

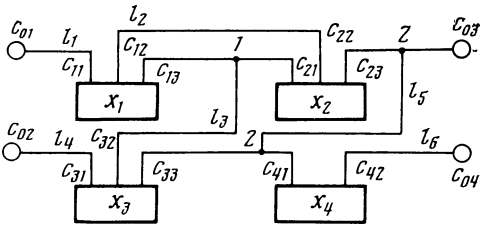


Рис. 2.47. Условный фрагмент схемы

Два контакта считаются связанными, если они объединяются одной электрической цепью. Под электрической цепью будем понимать некоторое множество контактов, принадлежащих одному эквипотенциалу.

Рассмотрим несколько способов задания схем графами, гиперграфами и их матричными и списковыми представлениями.

Зададим схему в виде графа $G = (X \cup E \cup C, U)$, где X — вершины графа, соответствующие элементам схемы; E — вершины, соответствующие цепям схемы; C — вершины, соответствующие контактам элементов. Множество ребер $U = F \cup W$ состоит из ребер эле-

ментных F и сигнальных W. Ребра подмножества F определяют принадлежность контактов из C элементам X и задаются парами (x_i, C_s). Ребра подмножества W задают вхождение контактов из C в цепи E и описываются парами (C_s, l_j). На рис. 2.48 показан граф схемы рис. 2.47.

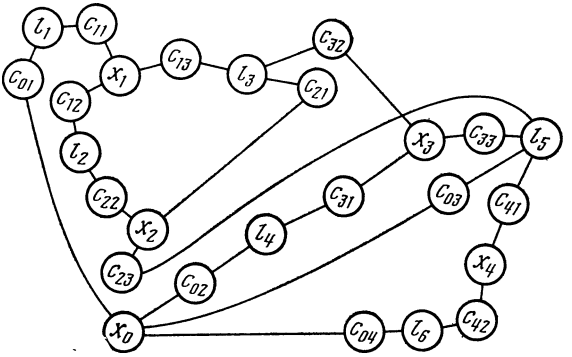


Рис. 2.48. Граф $G=(X \cup E \cup C, U)$

Обычно граф $G=(X \cup E \cup C \cup U)$ задается в виде двух матриц $\mathbf{A}_1$ и $\mathbf{A}_2$:

$$\mathbf{A}_1=\|a^1{}_{i,j}\|_{|E|\times|C|} \quad \text{и} \quad a^1{}_{i,j}=\begin{cases}1, \text{ если контакт } c_i \in l_j;\\ 0 \text{ в противном случае;}\end{cases}$$

$$\mathbf{A}_2=\|a^2{}_{i,j}\|_{|X|\times|C|} \quad \text{и} \quad a^2{}_{i,j}=\begin{cases}1, \text{ если } c_j \in X_i;\\ 0 \text{ в противном случае.}\end{cases}$$

Для графа G на рис. 2.48 матрицы $\mathbf{A}_1$ и $\mathbf{A}_2$ запишутся

$$\mathbf{A}_1=\begin{array}{c} \\ l_1\\ l_2\\ l_3\\ l_4\\ l_5\\ l_6\end{array}\!\!\begin{array}{c}\begin{array}{cccccccccccccc} c_{0,1} & c_{0,2} & c_{0,3} & c_{0,4} & c_{1,1} & c_{1,2} & c_{1,3} & c_{2,1} & c_{2,2} & c_{2,3} & c_{3,1} & c_{3,2} & c_{3,3} & c_{4,1} & c_{4,2}\end{array}\\ \left\|\begin{array}{cccccccccccccc} 1&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&1&0&0&0&1&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&1&0&0&1&1&0\\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&1\end{array}\right\|\end{array}.$$

Из матрицы $\mathbf{A}_1$ видно, что каждый столбец имеет только одну единицу, так как каждый контакт входит только в одну цепь.

$$\mathbf{A}_2=\begin{array}{c} \\ x_0\\ x_1\\ x_2\\ x_3\\ x_4\end{array}\!\!\begin{array}{c}\begin{array}{cccccccccccccc} c_{0,1} & c_{0,2} & c_{0,3} & c_{0,4} & c_{1,1} & c_{1,2} & c_{1,3} & c_{2,1} & c_{2,2} & c_{2,3} & c_{3,1} & c_{3,2} & c_{3,3} & c_{4,1} & c_{4,2}\end{array}\\ \left\|\begin{array}{cccccccccccccc} 1&1&1&1&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&1&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&1&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&1&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&1\end{array}\right\|\end{array}.$$

На рис. 2.48 x_0 соответствует условной вершине, объединяющей все контакты. Каждый столбец $\mathbf{A}_2$ также имеет только одну единицу, так как любой контакт относится лишь к одному элементу. Так как матрицы $\mathbf{A}_1$ и $\mathbf{A}_2$ содержат большое число нулевых элементов, то хранить их в памяти ЭВМ при реализации алгоритмов конструирования более эффективно в виде списков.

Граф G также задают в виде матрицы цепей $\mathbf{T}=\|t_{i,j}\|_{n\times c}$, строки которой соответствуют элементам схемы, а столбцы — контактам. Если элементы имеют различное число контактов, то в качестве c принимается $\max c_i$, $i=0, 1, \ldots, n$. Элемент $t_{i,j}$ представляет номер цепи, связанный с контактом c_j элемента x_i, заметим, что если в матрице $\mathbf{T}$ элементы $t_{i,j}=t_{p,q}=\ldots=t_{\alpha,\beta}$, то это означает, что контакты $j, q, \ldots, \beta$, расположенные на элементах с индексом $i, p, \ldots, \alpha$, принадлежат одной электрической цепи (одному узлу) и, следовательно, могут в дальнейшем быть соединены между собой электрически произвольным образом. Для фрагмента схемы на рис. 2.47 матрица $\mathbf{T}$ запишется в виде

$$\mathbf{T}=\begin{Vmatrix} x_1 & l_1 & l_2 & l_3 & 0 \\ x_2 & l_3 & l_2 & l_5 & 0 \\ x_3 & l_4 & l_3 & l_5 & 0 \\ x_4 & l_5 & l_6 & 0 & 0 \\ x_0 & l_1 & l_4 & l_5 & l_6 \end{Vmatrix}.$$

Например, элемент $t_{3,3}$ матрицы $\mathbf{T}$ означает, что цепь l_5 соединена с контактом *3* элемента схемы x_3. Данный способ удобен с точки зрения экономии памяти ЭВМ, но требует предварительной нумерации цепей.

При решении задач компоновки и размещения конструктор должен устанавливать различия между входами и выходами логических элементов схем. Различие между входами и выходами можно отразить, приписав ребрам направление прохождения сигнала, т. е. превратив граф схемы в орграф. Входной сигнал логического элемента исходит из соответствующей вершины, а выходной сигнал имеет направление к вершине. Каждому ребру припишем вес, равный номеру контакта, что позволяет полностью идентифицировать схему коммутации. Тогда фрагмент схемы на рис. 2.47 можно представить в виде двудольного графа D (рис. 2.49,*а*), которому можно поставить в соответствие гиперграф (рис. 2.49,*б*). Основное преимущество такого задания — изменяющаяся информация о цепях, которые могут быть конструктивно представлены любым из n^{n-2} покрывающих деревьев.

Иногда удобно задать схему в виде графа $G=(X\cup V, U)$, где V — узловые точки схемы. Тогда для фрагмента, изображенного на рис. 2.47, такой граф примет вид рис. 2.49,*в*.

Основное преимущество всех описанных моделей — адекватное задание в смысле планарности. Доказано, что если граф схемы, представленный одним из описанных методов, планарен, то планарна и представляемая им схема. Схема считается планарной,

если ее проводники могут быть расположены на плоскости без пересечений.

Существует много модификаций описанных методов. Одним из них является построение мультиграфа схемы. Для этого подсчитывается для каждой пары элементов число соединяющих их цепей. Затем строится мультиграф $G=(X, U)$, в котором вершины

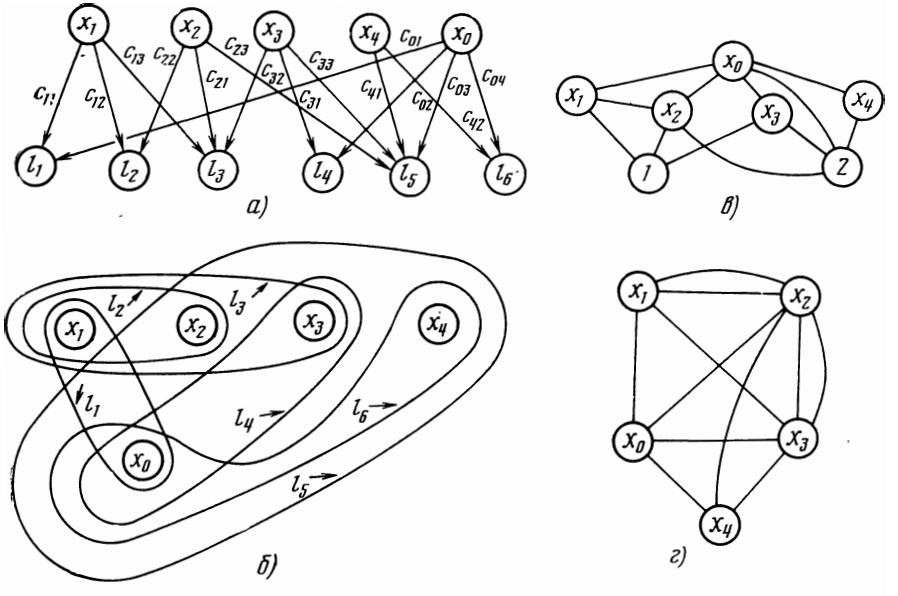


Рис. 2.49. Двудольный орграф $D=(X \cup E, U)$ (*а*), гиперграф (*б*), граф $G=(X \cup V, U)$ (*в*) и мультиграф (*г*) фрагмента схемы на рис. 2.47

соответствуют элементам, а ребра $u_{i,j}$ — числу цепей между ними. Для фрагмента схемы на рис. 2.47 мультиграф показан на рис. 2.49,*г*.

Иногда используется способ перехода от схемы к графу, когда контакты элементов представляются вершинами графа $G=(X, U)$, а соединения между ними — ребрами. При этом полученный граф представляет собой объединение полных подграфов, соответствующих узлам схемы. Если в каждом полном подграфе выделить деревья, покрывающие все вершины данного подграфа, т. е. выполнить «развязку узлов», то граф G представится множеством несвязанных деревьев, т. е. лесом.

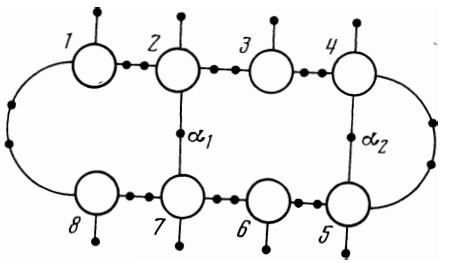


Рис. 2.50. Учет эквивалентных контактов

В настоящее время для возможного проведения цепей под

электрорадиоэлементами и между контактами рассматриваются графовые модели для самих элементов. При этом каждый элемент представляется в виде цикла, как, например, показано на рис. 2.50. Для задания пропускной способности каждого ребра, т. е. количества трасс, которые можно провести между контактами, между вершинами вводятся дополнительные (зачерненные) вершины. Такое представление элементов резко увеличивает число вершин и ребер графа схемы, но позволяет учитывать эквивалентные контакты.

При разработке алгоритмов конструирования в настоящее время все описанные методы используются для формализации описания схемы. Основными критериями выбора той или иной модели являются необходимый объем памяти ЭВМ, сложность реализации вычислительных процедур, время решения задачи, оптимальность получаемых результатов конструирования.

2.4. МЕТОДЫ МАТЕМАТИЧЕСКОГО ПРОГРАММИРОВАНИЯ ПРИ АВТОМАТИЗАЦИИ КОНСТРУИРОВАНИЯ

Многие задачи конструкторского проектирования получают более эффективное решение при использовании иерархических методов, когда исходная задача Q_0 разбивается на некоторое число подзадач $Q_1, Q_2, \ldots, Q_k$ и делаются попытки решить каждую из них. Данный вопрос особенно актуален при конструировании ИМС, содержащих сотни тысяч элементов, так как в настоящее время без разбиения их на подсхемы выполнить конструирование невозможно.

Под решением задачи понимаются:
нахождение оптимального решения;
доказательство, что подзадача неразрешима;
доказательство, что значение полученного решения хуже, чем предыдущее.

Такое разбиение обычно описывают деревом, изображенным на рис. 2.51. Смысл разбиения состоит в том, что после разбиения подзадачи легче решать, или они имеют значительно меньший размер, или имеют новую структуру. Заметим, что каждую подзадачу можно снова разбивать на подзадачи. Очевидно, что если разбиение окончено, то множество подзадач, на которые разбита задача, должно представлять всю задачу. Из рис. 2.51 видно, что после окончания каждая подзадача представляется вершиной дерева, которая называется висячей. Важной проблемой является нахождение оптимального решения на основе перебора путей на дереве решений. Задачи такого типа называют поисковыми. А сам поиск связан с исследованием графовых структур.

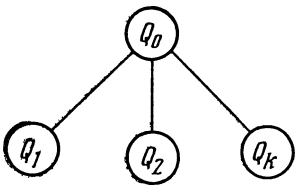


Рис. 2.51. Разбиение задачи Q_0 на подзадачи

Существуют три основных типа поиска в зависимости от того, как выбираются вершины дерева для прохождения ветвления. Это поиск с возвращением, поиск в глубину и поиск в ширину.

Поиск с возвращением. Он заключается в попытках расширить частное решение. На каждом шаге поиска, если расширение текущего частичного решения невозможно, происходит возврат к предыдущему частичному решению и предпринимается попытка снова его продолжить. Другими словами, поиск с возвращением — это продолжение расширения исследуемого решения до тех пор, пока это возможно. Когда решение нельзя расширить, производятся возврат назад и попытка сделать другой выбор на ближайшем шаге, где имеется такая возможность. Поиск с возвращением относится к классу экспоненциальных алгоритмов.

Поиск в глубину. Неформально поиск в глубину на дереве решений как бы соответствует «нырянию» в глубь дерева. Идея метода состоит в том, чтобы в каждой исследуемой вершине дерева выбирать один из возможных путей и исследовать его до конца, причем другие, существующие пути решений при этом не рассматриваются, пока сохраняется возможность получить конечный результат, исследуя выбранное направление. Дерево поиска в глубину показано на рис. 2.52. Как следует из рисунка, при поиске в глубину выбирается определенный путь по дереву и производится его продолжение до получения решения или до тех пор, пока дальнейшее продвижение окажется невозможным. Тогда процесс поиска возобновляется от ближайшей вершины дерева, имеющей смежные неисследованные вершины.

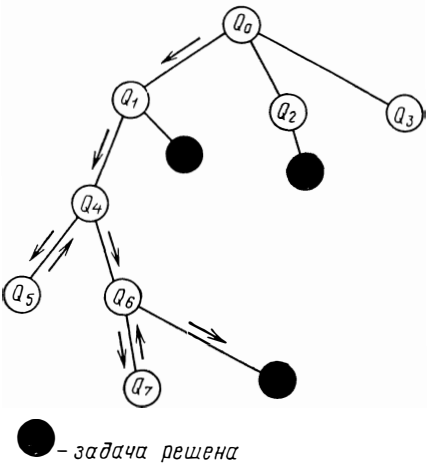


Рис. 2.52. Дерево поиска в глубину

Рассмотрим алгоритм просмотра вершин графа $G=(X, U)$ на основе поиска в глубину. Выбираем произвольную начальную вершину x_i. Затем выбираем ребро (x_j, x_i), инцидентное x_i, и просматриваем вершину x_j. Пусть x_p — последняя просмотренная вершина. Для продолжения процесса определяем произвольное нерассмотренное ребро (x_p, x_s), инцидентное x_p. Если вершина x_s ранее просматривалась, ищем новое ребро, инцидентное x_p. Если x_s ранее не просматривалась, то идем в x_s и снова начинаем поиск от x_s. Пройдя все пути, начинающиеся в x_s, возвращаемся в x_p. Затем продолжаем выбор нерассмотренных ребер, инцидентных вершине x_p, пока не будет исчерпан их список. Видно, что в

рассмотренной процедуре поиск идет вперед (вглубь), пока это возможно.

Поиск в глубину можно осуществлять и на орграфе D. В этом случае, пройдя в вершину x_p, выбираем ребра $\langle x_p, x_s \rangle$, выходящие из x_p. После исследования всех ребер, выходящих из x_s, возвращаемся в x_p, даже если имеются нерассмотренные ребра, входящие в x_s.

Для поиска в глубину на несвязном графе G, состоящем из нескольких компонент, необходимо исследовать каждую компоненту в отдельности.

Рассмотрим построение покрывающего дерева в графе G (рис. 2.53) на основе поиска в глубину. В качестве начальной возьмем вершину x_1. Для просмотра вершин на каждом уровне будем выбирать вершины с меньшим номером. Тогда в соответствии с описанным методом получим покрывающее дерево T, состоящее из вершин x_1, x_2, x_4, x_7, x_5, x_3, x_6, x_8 и ребер u_1, u_5, u_7, u_8 u_9, u_6, u_3. Порядок поиска в глубину показан на рис. 2.54.

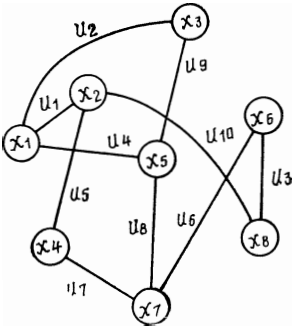


Рис. 2.53. Граф для построения покрывающего дерева

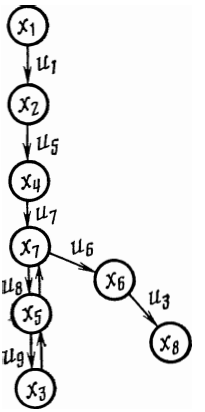


Рис. 2.54. Пример построения покрывающего дерева на основе поиска в глубину

Из описанного следует, что при поиске в глубину перебор сначала выполняется вдоль одного пути, пока не будет достигнута максимально возможная глубина. Далее рассматриваются другие пути такой же или меньшей глубины, которые отличаются от первого последним шагом, затем исследуются пути, отличающиеся последними двумя шагами и т. д. Процесс повторяется до получения требуемого результата.

Основное достоинство методов поиска в глубину то, что большинство из них относится к классу полиномиальных алгоритмов и может быть в основном реализовано за время $O(n)$.

Основной недостаток поиска в глубину заключается в том, что при исследовании дерева решений с большой вероятностью мож-

но пройти мимо той ветви, на которой раньше всего появляется окончательное решение.

Поиск в ширину при использовании этого метода ветвления происходит от уровня к уровню, причем если на уровне 1 начальная задача Q_0 разбивается на подзадачи $Q_1, Q_2, \ldots, Q_k$, то каждая из них исследуется раньше, чем задачи уровня 2. Задачи в уровне, которые трудны для разрешения, разбиваются на подзадачи уровня 2 и процесс продолжается аналогично. Примерный вид дерева решений при поиске в ширину показан на рис. 2.55. Для графа (см. рис. 2.53) порядок построения покрывающего дерева методом поиска в ширину показан на рис. 2.56.

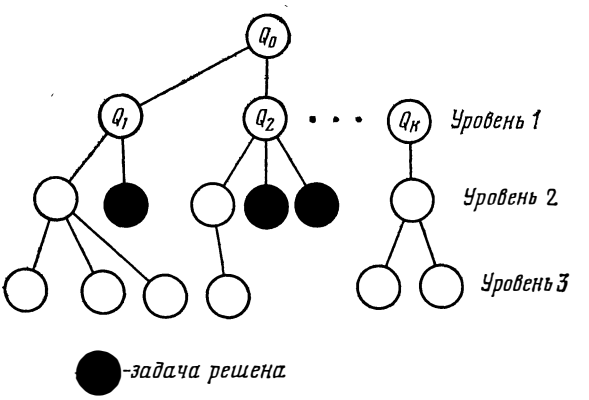


Рис. 2.55. Дерево поиска в ширину

Поиск в ширину эффективно используется при установлении изоморфизма и изоморфного вложения графов при разбиении и размещении схем. Алгоритмы поиска в ширину в основном реализуются за время $O(n^2)—O(n^4)$. Существует много процедур повы-

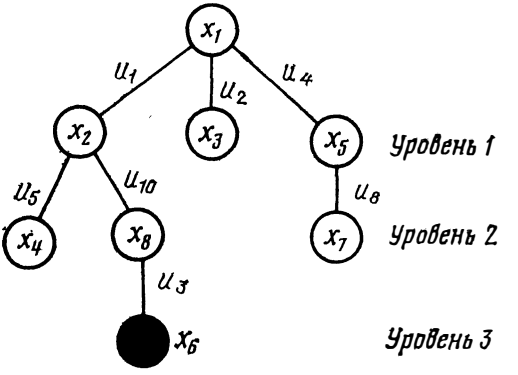


Рис. 2.56. Пример построения покрывающего дерева на основе поиска в ширину

шения эффективности поисковых методов. Основные из них — это упорядочивание ветвей, отходящих от каждой вершины, так что в первую очередь рассматриваются наиболее перспективные ветви, и движение вперед из вершины, которая является наилучшей из всех найденных независимо от того, где она расположена в дереве.

Заметим, что нами описаны только общие схемы организации поиска. Непосредственное их применение может привести к алгоритмам, время реализации которых велико. Поэтому необходимо приспосабливать эти методы к конкретным задачам конструирования.

Многие задачи конструирования схем сводятся к значительному перебору вариантов, приближающемуся к полному. В некоторых случаях полный перебор удается заменить перебором с отсечениями. Наиболее употребительным при конструировании вычислительным методом сокращения перебора является *метод ветвей и границ*.

Метод основан на переборе определенных вариантов решений задачи при условии, что число этих вариантов конечно. При этом в общем случае полный перебор заменяется частичным, так как используются априорные оценки вариантов, позволяющие отбрасывать «неперспективные» решения.

Метод ветвей и границ состоит из операций ветвления и отсечения.

Операция ветвления. Выбираем произвольный способ разбиения множества Q на подмножества Q_i такие, что $\cup Q_i = Q$, $Q_i \cap Q_j = \varnothing$, $\forall_{i \neq j}$. Затем разбиваем по такому же правилу подмножества и т. д. На каждом шаге вариант, оптимальный для множества Q, будет принадлежать одному из Q_i. Следовательно, решение задачи для множества Q сводится к решению задач для составляющих его множеств Q_i и определению оптимального решения среди найденных. Операцию ветвления можно осуществлять на основе методов поиска как в глубину, так и в ширину.

Операция отсечения. Отсечение вариантов решения задачи выполняется с помощью оценочной функции, которая задается на вершинах дерева перебора и дает верхнюю или нижнюю границу значений функции для вариантов решений, входящих в множество Q. Оценочная функция, дающая нижнюю границу в процессе разбиения множества, не убывает, а оценочная функция с верхней границей не возрастает.

Заметим, что точность оценки зависит от способа разбиения множества Q, поэтому необходимо выбирать такой способ разбиения, при котором оценки получаемых подмножеств были бы ближе к оптимальной.

Существуют три основных способа отсечения ветвей:

по результатам сравнения текущего значения с уже найденным;

по результатам сравнения двух оценок;

прекращение ветвления в вершине, не содержащей оптимального решения.

Порядок продолжения ветвления выбирается на основе последовательного построения ветвей, или по минимальной нижней (верхней) границе, или на основе их комбинаций.

Рассмотрим теперь общую схему метода ветвей и границ для нахождения решения с наименьшим значением целевой функции. Пусть необходимо решить одну из задач конструирования схемы, заданной в виде графа или гиперграфа. Пусть также известно, что Q — это конечное множество решений данной задачи. Каждое решение $q \in Q$ может быть оценено с помощью оценочной (целевой) функции $f(q)$. Требуется найти такое решение $t \in Q$, для которого $f(t)$ — наименьшее среди всех $f(q)$, $q \in Q$. Использование специфики конкретной задачи конструирования позволяет в ряде случаев найти нижнюю границу значения целевой функции f, или ее оценку на множестве решений Q, или на каком-либо его подмножестве $Q_i \subseteq Q$. Нижней границей $\Theta(q)$ целевой функции f будем называть значение, которое меньше или равно значению целевой функции $f(q)$ для любого решения $q \in Q$ или $q' \in Q'$. Метод ветвей и границ наиболее эффективен, если при разбиении множества решений на подмножества Q_i удается улучшать оценки, т. е. получать строго неравенство $\Theta(q_i) > \Theta(q)$.

Разбиение множества решений на подмножества представляется, как было показано выше, с помощью дерева решений, которое можно задавать в графическом или матричном виде. В качестве исходной вершины дерева решений принимается множество Q. Затем вычисляется нижняя граница $\Theta(q)$ множества Q. Заметим, что вычисление нижней границы при решении задач проектирования в огромной степени зависит от искусства конструктора, причем чем ближе оценка к точной, тем быстрее будет найдено оптимальное решение. Если будет найдено такое решние t, что $f(t) = \Theta(q)$, то t — искомое оптимальное решение. Если найти сразу такое решение не удалось, то множество Q разбиваем на ряд подмножеств, образующих на первом шаге вершины дерева, которые обозначаются $Q^1_1, Q^1_2, \ldots, Q^1_s$. Верхний индекс при Q соответствует номеру шага разбиения, нижний — определяет номер подмножества множества решений на данном шаге. Определяются нижние границы $\Theta(Q^1_j)$, $j = 1, 2, \ldots, s$. Если найдено такое решение $t \in Q^1_p$, что $f(t) = \Theta(Q^1_p)$, причем $\Theta(Q^1_p) \leqslant \Theta(Q^1_j)$, $j \neq p$, $j = 1, 2, \ldots, s$, то t — искомое оптимальное решение. Если его найти не удалось, то выбираем из подмножеств $Q^1_1, Q^1_2, \ldots, Q^1_s$ подмножество Q^1_i по правилу $\Theta(Q^1_i) = \min \Theta(Q^1_j)$, $j = 1, 2, \ldots, s$ и разбиваем его на определенное число подмножеств $Q^1_i = Q^1_{i_1} \cup Q^1_{i_2} \cup \ldots \cup Q^1_{i_l}$. Все подмножества $Q^1_1, Q^1_2, \ldots, Q^1_{i-1}, Q^1_{i+1}, \ldots, Q^1_s, Q^1_{i_1}, Q^1_{i_2}, \ldots, Q^1_{i_l}$, ранее не подвергавшиеся разбиению, переобозначаются $Q^2_1, Q^2_2, \ldots, Q^2_v$.

Производится переход ко второму шагу и определяются нижние границы $\Theta(Q^2_j)$, $j = 1, 2, \ldots, v$. Если найдено такое решение

$t \in Q^2_p$, что $f(t) = \Theta(Q^2_p)$, причем $\Theta(Q^2_p) \leqslant \Theta(Q^2_j)$, $j \neq p$, $j = 1, 2, \ldots, v$, то t — искомое решение. Если найти его не удалось, то снова выбирается из подмножеств $Q^2_1, Q^2_2, \ldots, Q^2_v$ подмножество Q^2_i по правилу $\Theta(Q^2_i) = \min \Theta(Q^2_j)$, $j = 1, 2, \ldots, v$, и производится разбиение Q^2_i на определенное число подмножеств $Q^2_i = Q^2_{i1} \cup Q^2_{i2} \cup \ldots \cup Q^2_{iw}$. Все подмножества $Q^2_1, Q^2_2, \ldots, Q^2_{i-1}, Q^2_{i+1}, \ldots, Q^2_v, Q^2_{i1}, \ldots, Q^2_{iw}$, ранее не подвергавшиеся разбиению, переобозначаются $Q^3_1, Q^3_2, \ldots, Q^3_2$ и производится переход к третьему шагу и т. д. Процедура продолжается до v-го шага. Если среди решений v-го шага удается найти решение $t \in Q^v_p$ такое, что $f(t) = \Theta(Q^v_p) \leqslant \Theta(Q^v_j)$, $j \neq p$, $j = 1, 2, \ldots, v$, то t — искомое оптимальное решение. В противном случае процесс продолжается. Так как множество решений Q конечно, то в результате последовательных ветвлений получаются подмножества, состоящие из одного решения. Среди них будет найдено оптимальное решение.

Метод ветвей и границ позволяет также находить квазиоптимальные решения с заданной точностью Δ. Если на произвольном шаге μ ветвления дерева найдено решение t' такое, что $f(t') - \Theta(Q^\mu_r) \leqslant \Delta$, где Q^μ_r — подмножество с наименьшей нижней границей среди всех подмножеств данного шага, то t' — искомое квазиоптимальное решение.

Рассмотрим пример ветвления дерева решений задачи определения соответствия принципиальной схемы и ее реализации на кристалле интегральной микросхемы. Пусть известно, что множество решений Q. Разобьем Q на три подмножества: Q^1_1, Q^1_2, Q^1_3. Среди них оптимальное решение t не найдено. Из оценок границ определено, что должна ветвиться вершина Q^1_2. Подмножество решений Q^1_2 разбивается на три подмножества: $Q^1_{2,1}$, $Q^1_{2,2}$, $Q^1_{2,3}$. Нулевой и первый шаги показаны на рис. 2.57. Вершины Q^1_1, Q^1_3, которые не ветвились, а также $Q^1_{2,1}$, $Q^1_{2,2}$, $Q^1_{2,3}$ переобозначаем соответственно Q^2_1, Q^2_2, Q^2_3, Q^2_4, Q^2_5. Согласно специфике задачи определения соответствия схемы и ее реализации находятся нижние границы для этих подмножеств. Если не найдено оптимальное решение t, то ветвится, например, вершина Q^2_1 следующим образом: $Q^2_1 = Q^2_{1,1} \cup Q^2_{1,2}$.

Дерево решений для второго шага алгоритма показано на рис. 2.57,*б*. Далее переобозначаются вершины и находятся границы для полученных подмножеств решений третьего шага. Если оптимальное решение t, т. е. подстановка между схемой и ее реализацией на кристалле, не найдено, ветвится, например, вершина Q^3_6 (рис. 2.57,*в*). Процедура повторяется до получения оптимального или квазиоптимального решения.

Следовательно, для эффективного применения метода ветвей и границ необходимо, учитывая специфику рассматриваемой задачи, вычислять границы целевой функции как можно ближе к оптимальному значению. Это определит число ветвящихся вершин и соответственно степень сокращения перебора при решении задачи.

При описании конкретных алгоритмов конструирования схем будут показаны примеры использования метода ветвей и границ.

Часто при решении задач автоматизации конструирования схем требуется найти такие значения действительных переменных $q_1, q_2, \ldots, q_n$, при которых целевая функция F нескольких переменных обращается в максимум или минимум. Причем $Q_i = \{q^i_1, q^i_2, \ldots, q^i_n\}$ — это решения из множества $Q = \{Q_1, Q_2, \ldots, Q_n\}$.

Задачи определения значений параметров, обеспечивающих экстремум (максимум или минимум) функции при заданных ограничениях на аргументы, называются задачами математического программирования.

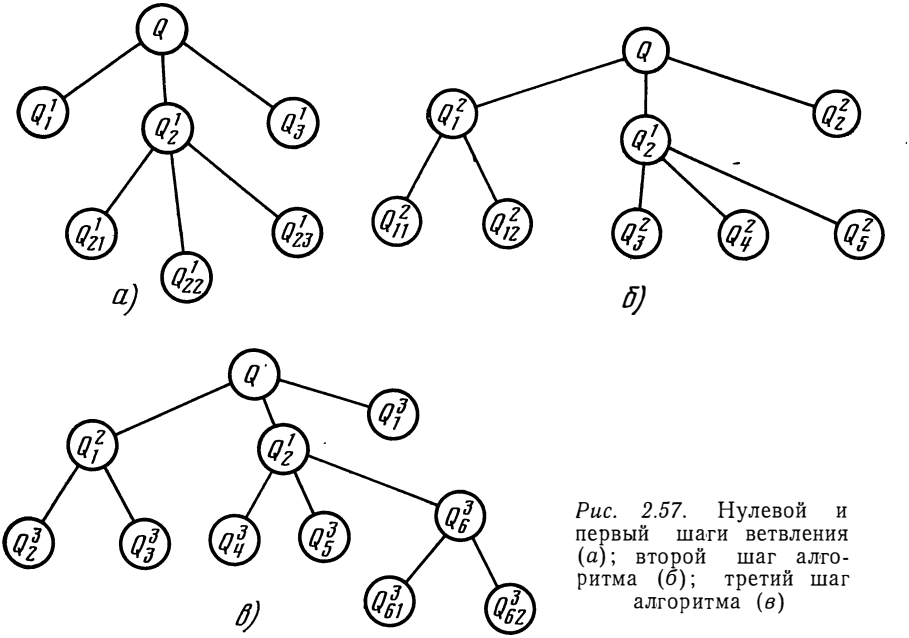


Рис. 2.57. Нулевой и первый шаги ветвления (*а*); второй шаг алгоритма (*б*); третий шаг алгоритма (*в*)

Основными представителями класса задач математического программирования являются задачи линейного, динамического и нелинейного программирования.

Задачи линейного программирования являются наиболее простыми. Они характеризуются тем, что показатель эффективности (целевая функция) F линейно зависит от параметров решения $q_1, q_2, \ldots, q_n$, и ограничения, налагаемые на параметры решения, имеют вид линейных равенств или неравенств относительно элементов множества Q.

Рассмотрим пример задачи линейного программирования. Цех завода выпускает многослойные печатные платы (МПП) и двухслойные печатные платы (ДПП). В соответствии с планом он должен выпустить МПП в количестве не меньше α_1 и не больше β_1, ДПП — не меньше α_2 и не больше β_2. Для изготовления плат необходимы три вида комбинированного сырья γ_1, γ_2, γ_3, запасы которого ограничены числами Δ_1, Δ_2 и Δ_3.

Допустим, что на изготовление одной платы требуется $a_{i,j}$ сырья вида γ_i ($i=1,2,3$; $j=1,2$). Индекс j означает вид изделия, индекс i — вид сырья. Величины $a_{i,j}$ можно представить в виде матрицы следующего вида:

Сырье	МПП	ДПП
γ_1	$a_{1,1}$	$a_{1,2}$
γ_2	$a_{2,1}$	$a_{2,2}$
γ_3	$a_{3,1}$	$a_{3,2}$

При использовании в аппаратуре МПП приносят прибыль b_1, а ДПП — b_2. Необходимо так спланировать производство, чтобы план был выполнен и перевыполнен без затоваривания, а суммарная прибыль от реализации была максимальна. (Заметим, что данный пример описывает упрощенную и идеализированную ситуацию в производстве печатных плат, но тем не менее позволяет оценивать реальные ситуации.)

Будем считать, что элементами решения будут количества произведенных МПП и ДПП, соответственно q_1 и q_2. Требование выполнения плана можно записать в виде ограничений:

$$q_1 \geqslant \alpha_1; \quad q_2 \geqslant \alpha_2. \tag{2.4}$$

Недопустимость лишней продукции можно выразить следующими неравенствами:

$$q_1 \leqslant \beta_1; \quad q_2 \leqslant \beta_2. \tag{2.5}$$

Тогда трем комбинированным видам сырья будут соответствовать три ограничительных неравенства:

$$\begin{cases} a_{1,1}q_1 + a_{1,2}q_2 \leqslant \gamma_1; \\ a_{2,1}q_1 + a_{2,2}q_2 \leqslant \gamma_2; \\ a_{3,1}q_1 + a_{3,2}q_2 \leqslant \gamma_3, \end{cases} \tag{2.6}$$

а прибыль, получаемая от реализации продукции,

$$\Pi = b_1q_1 + b_2q_2. \tag{2.7}$$

Таким образом, составлена задача линейного программирования, которую можно сформулировать следующим образом: определить такие неотрицательные значения q_1 и q_2, чтобы они удовлетворяли всем неравенствам-ограничениям, а также обращали в максимум линейную функцию этих переменных Π:

$$\Pi = b_1q_1 + b_2q_2 \Rightarrow \max. \tag{2.8}$$

Неравенства системы (2.6) представляют собой полуплоскости с граничной прямой $a_{i1}q_1 + a_{i1}q_2 = \gamma_1$. Условия неотрицательности определяют полуплоскости с граничными прямыми $q_1 = \alpha_1$, β_1; $q_2 = \alpha_2$, β_2. Система совместна, поэтому полуплоскости, пересекаясь, образуют общую часть, которая представляет собой пары точек, являющихся решением данной системы. Объединение этих точек называют многоугольником решений. Геометрическая задача линейного программирования сводится к определению такой точки многоугольника решений, координаты которой дают линейной функции экстремальное значение. Отметим, что все точки многоугольника решений являются допустимыми.

Предположим, что система (2.6) при условиях (2.4) и (2.5) совместна и ее многоугольник решений ограничен. Тогда существует такая угловая точка, в которой линейная функция задачи программирования достигает оптимума. Линейная функция (2.7) при фиксированных значениях является уравнением прямой линии $b_1q_1 + b_2q_2 = \text{const}$.

Построим теперь многоугольник решений системы (2.4)—(2.6) и график линейной функции при $\Pi = 0$ и найдем точку многоугольника решений, в которой прямая $b_1q_1 + b_2q_2 = \text{const}$ является опорной и функция Π достигает максимума.

Построим многоугольник решений (рис. 2.58). Для этого в системе координат $q_1\,0\,q_2$ на плоскости построим граничные прямые:

$$a_{1,1}q_1+a_{1,2}q_2=\gamma_1,\ (A_1)$$
$$a_{2,1}q_1+a_{2,2}q_2=\gamma_2,\ (A_2)$$
$$a_{3,1}q_1+a_{3,2}q_2=\gamma_3,\ (A_3)$$
$$q_1=(\alpha_1,\ \beta_1),\ (A_4)$$
$$q_2=(\alpha_2,\ \beta_2),\ (A_5).$$

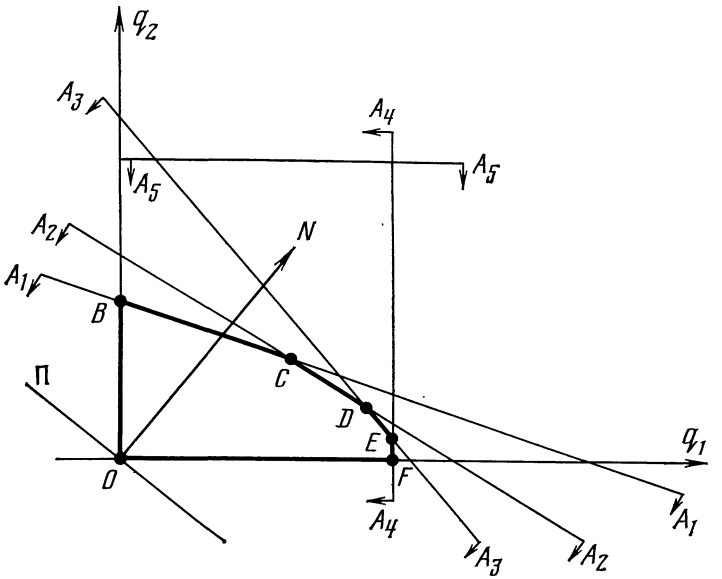


Рис. 2.58. Многоугольник решений

Полуплоскости, определяемые неравенствами, показаны стрелками. Многоугольником решений данной задачи является шестиугольник *OBCDEF*.

Для построения прямой $b_1q_1+b_2q_2=0$ строится радиус-вектор $N=(b_1,\ b_2)$ и через точку O проводится прямая, перпендикулярная ему. Прямая перемещается параллельно самой себе в направлении вектора N. Из рис. 2.58 следует, что в точке D функция $П$ принимает максимальное значение. Точка D лежит на пересечении прямых A_3 и A_2. Тогда для определения координат точки D необходимо решить систему уравнений

$$\left.\begin{aligned} a_{21}q_1+a_{22}q_2&=\gamma_2;\\ a_{31}q_1+a_{32}q_2&=\gamma_3 \end{aligned}\right\} \qquad (2.9)$$

и определить переменные q_1 и q_2. Подставляя значения q_1 и q_2 в линейную функцию $П=b_1q_1+b_2q_2$, получаем $П_{\max}$. Следовательно, для получения максимальной прибыли $П_{\max}$ необходимо запланировать выпуск найденных значений q_1 МПП и q_2 ДПП.

Заметим, что любая задача линейного программирования сводится к стандартной форме, называемой основной задачей линейного программирования. Она формулируется так.

Определить неотрицательные значения переменных $q_1,\ q_2,\ \ldots,\ \ldots,\ q_n$, которые удовлетворяют уравнениям

$$\left.\begin{aligned} &a_{1,1}q_1+a_{1,2}q_2+\ \ldots\ +a_{1,n}q_n=\gamma_1;\\ &a_{2,1}q_1+a_{2,2}q_2+\ \ldots\ +a_{2,n}q_n=\gamma_2;\\ &\ldots\ldots\ldots\ldots\ldots\ldots\\ &a_{m,1}q_1+a_{m,2}q_2+\ \ldots\ +a_{m,n}q_n=\gamma_m \end{aligned}\right\} \qquad (2.10)$$

и обращают в максимум линейную функцию этих переменных

$$\Pi=b_1q_1+b_2q_2+\ \ldots\ +b_nq_n\Rightarrow\max. \qquad (2.11)$$

Оптимальные решения основной задачи линейного программирования достигаются при таком наборе $q_1, q_2, \ldots, q_n$, когда, по крайней мере, l из них обращаются в нуль, а $n-l$ неотрицательны.

Задачи линейного программирования, в которых значения переменных должны быть обязательно целыми числами, называются задачами целочисленного программирования. Заметим, что решать задачи целочисленного программирования в общем случае труднее, чем обычные задачи линейного программирования.

Многие задачи размещения при автоматизации конструирования схем сводятся к так называемым задачам нелинейного программирования. В них необходимо определить неотрицательные значения переменных $q_1, q_2, \ldots, q_n$, удовлетворяющих заданным ограничениям произвольного вида и обращающих в максимум нелинейную функцию этих переменных $F=F(q_1, q_2, \ldots, q_n)\Rightarrow\max$.

Задачи, в которых оптимальное решение ищется в условиях неполной определенности, а параметры, входящие в целевую функцию, и ограничения представляют собой случайные величины, называются задачами стохастического программирования.

Многие задачи автоматизации конструирования, такие, как трассировка, компоновка, размещение, часто разбивают на этапы или шаги. Метод многоэтапной оптимизации называется методом динамического программирования.

Рассмотрим задачу трассировки (подробнее см. разд. 3), распадающуюся на m этапов. Пусть эффективность каждого этапа характеризуется «выигрышем» f_i. Предположим, что выигрыш F по всей задаче состоит из выигрыша на отдельных этапах

$$F=\sum_{i=1}^{m} f_i.$$

Задачу трассировки можно представить как управляемый процесс выбора параметров, влияющих на ход решения, причем на каждом этапе выбирается такое решение, от которого зависит выигрыш на этапе. Этот процесс называется *этапным управлением*. Множество всех этапных управлений называется *управлением задачи в целом*.

Требуется найти такое управление q из множества всех управлений, при котором выигрыш F обращается в максимум:

$$F=\sum_{i=1}^{m} f_i\Rightarrow\max.$$

Управление $q^* \in Q$, при котором достигается этот максимум, называется оптимальным. Максимальный выигрыш, который достигается при этом, обозначается $F^* = \max\{F(q)\}$.

В основе методов динамического программирования лежит построение оптимального управления на каждом этапе.

Рассмотрим пример прокладки трассы между двумя контактами A, B интегральной микросхемы на координатной плоскости (рис. 2.59), причем трасса может проходить только в горизонтальном или вертикальном направлении. Необходимо проложить трассу, для которой сумма весов вошедших в нее ребер из графа сетки G_r будет минимальной.

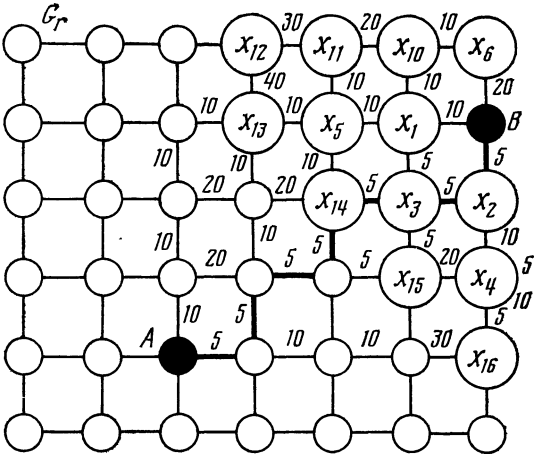


Рис. 2.59. Пример прокладки трассы между контактами A и B

Для использования метода динамического программирования разделим процесс перехода из A в B на отдельные шаги (один шаг — подключение к трассе одного ребра графа сетки G_r) и оптимизируем управление по шагам. Процедуру оптимизации выполним от конца к началу, т. е. от B к A. Сначала произведем оптимизацию последнего N-го шага. Рассмотрим, куда должна быть доведена трасса на $(N-1)$-м шаге, чтобы на N-м шаге мы могли закончить ее построение. Очевидно, что только в вершины, из которых можно за один шаг попасть в B, т. е. в x_1, x_2 или x_6. Если мы находимся в x_1, то при попадании в B сумма весов возрастет на десять единиц, если в x_2, то на пять единиц, а если в x_6, то на 20 единиц. Следовательно, условная оптимизация сделана и выигрыш для вершин x_1 и x_2 соответственно равен 10 и 5 усл. ед. Теперь перейдем к оптимизации предпоследнего $(N-1)$-го шага. После $(N-2)$-го шага трасса могла оказаться в одной из четырех вершин: x_3, x_4, x_5, x_{10}. После анализа получим следующие фрагменты трасс: x_{10}—x_1—B — сумма весов 20; x_5—x_1—B — 20; x_3—x_2—B — 10; x_4—x_2—B — 15. Как видно, наибольший выигрыш будет в вершине x_3. Продолжая аналогично, определяем выигрыш на всех вершинах графа сетки и находим путь, ведущий из контакта A в контакт B (на рис. 2.59 показан жирной линией).

При выполнении условной оптимизации возможен неоднозначный выбор оптимального управления. Это говорит о том, что во многих задачах математического программирования решение в общем случае не единственное.

Приведем теперь *принцип оптимальности*, на основе которого решаются все задачи динамического программирования: при любом состоянии системы перед очередным этапом необходимо выбирать управление на этапе таким образом, чтобы выигрыш на нем в сумме с оптимальным выигрышем на всех следующих этапах был максимальным.

2.5. КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Опишите способы задания множеств.
2. В чем отличие между понятиями принадлежности множеству и включения в множество?
3. Справедливо ли, что $\{1, 2, 3\} \in \{\{1, 2, 3\}, \{1\}, \{2\}, \{3\}, 1, 2\}$?
4. Верно ли, что $\{1, 2, 3\} \subset \{\{1, 2, 3\}; 1, 2, 3, \{1\}\}$?
5. Привести пример таких множеств A, B, C, D, что $A \subset B \wedge B \subset C \wedge C \not\subset D \wedge B \subset D$.
6. Доказать, что $(A \setminus B) \cup B = A$, $(A \cap B) \cap C = A \cap (B \cap C)$.
7. Что понимается под термином «высказывание»?
8. Доказать или опровергнуть методом от противного следующие выражения: $((A \cap B) \cup (A \cap \bar{B})) \setminus A = \varnothing$, $A \cap (B \setminus A) = \varnothing$; $(A \setminus B) \setminus (A \cap B) = \varnothing$.
9. Приведите пример разбиения трехэлементного множества $A = \{$ЭВМ, ЭВА, РЭА$\}$.
10. Приведите множества A, B, для которых справедливо $A \times B = B \times A$.
11. Определить, чему равно $A \setminus (A \setminus B)$.
12. Доказать или опровергнуть следующие выражения: $(A \cap B) \times C = (A \times C) \cap (B \cap C) \cdot (A \setminus B) \times C = (A \times C) \cap (B \times C)$.
13. Доказать, что для графиков P и Q справедливо $(P \circ Q)^{-1} = Q^{-1} \circ P^{-1}$.
14. Какими общими свойствами обладают бинарные отношения, заданные в некотором коллективе людей X и выражающие соотношениями $(x_i, x_j \in X)$:

x_i уважает x_j;
x_i ненавидит x_j;
x_i подчинен x_j;
x_i родственник x_j?

15. Приведите примеры рефлексивных, симметричных, транзитивных отношений эквивалентности и постройте их матрицы.
16. Докажите, что если отношения A и B есть отношения эквивалентности, то отношением эквивалентности будет и $A \cap B$.
17. На множестве $B = \{1, 2, 3, 4\}$ заданы отношения:

$$\varphi_1 = \langle \{\langle 1,1\rangle, \langle 1,2\rangle, \langle 1,3\rangle, \langle 1,4\rangle\}, \{1, 2, 3, 4\}\rangle;$$

$$\varphi_2 = \langle \{\langle 1,4\rangle, \langle 2,4\rangle, \langle 3,4\rangle\}, \{1, 2, 3, 4\}\rangle;$$

$$\varphi_3 = \langle \{\langle 1,2\rangle, \langle 2,1\rangle, \langle 1,3\rangle, \langle 3,1\rangle, \langle 4,4\rangle\}, \{1, 2, 3, 4\}\rangle;$$

$$\varphi_4 = \langle \{\langle 1,4\rangle, \langle 1,3\rangle, \langle 2,2\rangle, \langle 2,4\rangle, \langle 2,3\rangle\}, \{1, 2, 3, 4\}\rangle;$$

$$\varphi_5 = \langle \{\langle 1,3\rangle, \langle 4,1\rangle, \langle 4,2\rangle, \langle 4,3\rangle\}, \{1, 2, 3, 4\}\rangle.$$

Определите, какие из них являются соответствием и функцией?
18. Определите, какие из приведенных отношений являются отношениями эквивалентности и почему:

взаимозаменяемость ИМС в схеме РЭА;
равенство размеров ИМС в схеме РЭА;
равенство весов элементов в схеме РЭА;
принадлежность к одной группе плат в множестве плат РЭА;
кратность размеров ИМС в схеме РЭА.

19. Чем отличается понятие отношения от понятия соответствия?
20. Приведите примеры функционального, инъективного, сюръективного, всюду определенного и биективного соответствий.
21. Найдите достаточные условия, которым должны удовлетворять соответствия Γ и Δ, чтобы $\Gamma \circ \Delta$ было всюду определенным, сюръективным, инъективным, биективным (взаимно-однозначным).

22. Определите $\Gamma(A)$ и $\Gamma^{-1}(B)$, если

$$\Gamma=\langle\Phi,\ X,\ Y\rangle;\ \Phi=\{\langle 1, a\rangle, \langle 2, b\rangle, \langle 3, c\rangle, \langle 4, d\rangle\};$$
$$X=\{1,\ 2,\ 3,\ 4\},\ Y=\{a,\ b,\ c,\ d\};\ A=\{1,\ 3\};\ B=\{a,\ c,\ d\}.$$

23. Дайте основные определения расплывчатого множества.
24. Пусть $X=(1, 2, 3, 4)$, постройте пример расплывчатого множества A на X.
25. Определите основные операции над расплывчатыми множествами и приведите примеры их использования.
26. Определите расплывчатое отношение на множествах $A=\{$ТЭЗ, ИМС$\}$; $B=\{$плата, стойка$\}$.
27. Определите расплывчатое соответствие и постройте примеры.
28. Пусть $X=\{x_1,\ x_2,\ x_3\}$, $Y=\{y_1,\ y_2,\ y_3,\ y_4\}$. Задайте расплывчатое отношение φ матрицей, элементами которой будут степени принадлежности $\mu(x_i,\ y_j)$.
29. Опишите три основных типа универсальных алгоритмических моделей.
30. Дайте определение алгоритма как алфавитного оператора.
31. Постройте схему машины Тьюринга и опишите принцип ее работы.
32. В чем состоит один из основных результатов Тьюринга?
33. Приведите классификацию решаемых и нерешаемых проблем.
34. Определите алгоритмы подклассов P, NP, NPC.
35. Составьте операторный алгоритм Ван Хао, реализующий операцию объединения двух произвольных множеств.
36. Составьте алгоритм поиска минимального элемента из множества M, заданного в виде $M=\{4, 8, 9, 12, 7, 3\}$ на основе ЛСА.
37. Составьте алгоритм поиска максимального элемента из множества $M=\{1, 8, 5, 7, 12\}$ на основе ГСА.
38. Составьте логическую схему алгоритма упорядочивания массива из n чисел в порядке убывания элементов.
39. Составьте логическую схему алгоритма вычисления по формуле

$$C_k=\sum_{i=1}^{n} A_i-b_k\ \ (k=1,\ 2,\ \ldots,\ m).$$

40. Составьте структурную схему алгоритма вычисления по формуле

$$\prod_{i=1}^{n} c_i.$$

41. Составьте структурную схему алгоритма вычисления значений функции $y_{i,j}=x_i z_j$, $x_i=x_1,\ \ldots,\ x_5$; $z_j=z_1,\ z_2,\ \ldots,\ z_{10}$.
42. Составьте ГСА алгоритма вычисления по формуле

$$\prod_{j=1}^{n}\prod_{i=1}^{n} c_{ij}.$$

43. Составьте словесный алгоритм определения

$$y=\sum_{i=1}^{5}\sum_{j=1}^{4} a_i,\ b_j.$$

44. Определите расплывчатые алгоритмы и приведите их классификацию.
45. Постройте структурную схему расплывчатого алгоритма «размещение разногабаритных элементов» на кристалле интегральной микросхемы по критерию минимизации площади кристалла.
46. Постройте смешанный граф, орграф и неорграф $G=(X, U)$, $|X|=10$, $|U|=14$.
47. Постройте структурные схемы алгоритмов выделения из графа сугрaфов и подграфов с заданным числом ребер.
48. В полном графе K_n, $n=7$, выделите одно из n^{n-2} покрывающих деревьев.

49. Для полного графа с четырьмя вершинами постройте все покрывающие неизоморфные деревья.

50. Постройте произвольный мультиграф $G=(X, U)$, $|X|=n$, $|U|=m$ с $n=8$, $m=14$, определите его мультичисло.

51. Постройте граф $G=(X, U)$ с $n=7$, $m=13$. Задайте его с помощью матриц смежности и инцидентности. Постройте граф-схему алгоритма перехода от одной матрицы к другой.

52. Подсчитайте число суграфов, включая изоморфные, в графе $G=(X, U)$ $|X|=n$.

53. Разработайте структурную схему алгоритма определения связности графа фрагмента схемы (см. рис. 2.47).

54. Предложите методы построения графов с эйлеровыми и гамильтоновыми циклами на заданных наборах вершин и ребер.

55. Постройте алгоритм построения двудольных графов с заданным числом вершин.

56. Постройте произвольный граф $G=(X, U)$, $|X|=n=10$, $|U|=m=18$, определите его цикломатическое число. Укажите, какие ребра должны быть удалены из графа, чтобы он стал ациклическим.

57. Определите хроматическое число неполного связного графа $G=(X, U)$ с $n=8$, $m=18$.

58. Постройте алгоритм раскраски графа схемы (см. рис. 2.47).

59. Определите множество полных графов, содержащих одновременно эйлеровы и гамильтоновы циклы.

60. Постройте ЛСА выделения компонент связности на примере фрагмента схемы (см. рис. 2.47).

61. Постройте граф G_r размером 3×4 и определите $d_{3,12}$.

62. Для графа на рис. 2.41,*б* определите семейство внутренне устойчивых подмножеств.

63. Для графа на рис. 2.24 определите семейство клик.

64. Для графа на рис. 2.24 определите семейство внешне устойчивых подмножеств и ядра графа, если они имеются.

65. Определите число планарности полного графа с 26 вершинами.

66. Определите толщину, шероховатость и число скрещиваний полного графа с 37 вершинами.

67. Задайте граф $D=(X, U)$, $|X|=7$, $|U|=15$, постройте его матрицы смежности и инцидентности. Определите для каждой вершины $\rho^+(x_j)$ и $\rho^-(x_j)$.

68. Постройте алгоритм раскраски гиперграфа $H=(X, E)$, изображенного на рис. 2.46,*б*.

69. Задайте фрагмент схемы и для него постройте граф $G=(X \cup E \cup C, U)$, матрицы A_1, A_2, T, граф $G=(X \cup E, U)$, гиперграф H, двудольный орграф, мультиграф, граф $G=(X, U)$.

70. Задайте гиперграф $H=(X, E)$, $|X|=7$, $|E|=6$, постройте его матрицу инцидентности $I(H)$ и соответствующий ему граф Кёнига.

71. Задайте фрагмент схемы и для него постройте граф, в котором бы учитывалась возможность прохождения трасс под элементами и между контактами.

72. Опишите методы поиска в глубину и ширину.

73. На основе поиска в глубину и ширину определите в произвольном графе $G=(X, U)$ кратчайшую цепь.

74. Нарисуйте схему метода ветвей и границ.

75. В произвольном графе G на основе метода ветвей и границ определите семейство внутренне устойчивых подмножеств.

76. Постройте алгоритм раскраски графа с использованием поиска в ширину. Рассмотрите его на примере фрагмента схемы (см. рис. 2.47).

77. Постройте алгоритм выделения в графе покрывающего дерева с использованием метода ветвей и границ.

78. В графе $G=(X, U)$, заданном следующей матрицей смежности:

$$R = \begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 2 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 3 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 4 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 5 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 6 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 7 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{array}$$

определите эйлеров цикл на основе поиска в глубину и ширину.

79. Постройте схему метода ветвей и границ для определения существования гамильтонова цикла в графе, заданном в предыдущем примере.

80. Сформулируйте задачу раскраски графа как задачу линейного программирования.

81. Опишите метод линейного программирования.

82. Приведите примеры решения задач линейного программирования геометрическим методом.

83. Сформулируйте основную задачу линейного программирования.

84. Дайте определения нелинейного и стохастического программирования.

85. Дайте определение оптимального управления.

86. Опишите метод динамического программирования. Постройте пример.

87. Сформулируйте задачу определения кратчайшего пути в графе как задачу линейного программирования, задачу динамического программирования. Постройте примеры для фрагмента схемы на рис. 2.47.

3. АЛГОРИТМИЧЕСКОЕ КОНСТРУИРОВАНИЕ СХЕМ

3.1. КОМПОНОВКА

На этапе конструирования решаются вопросы, связанные с компоновкой элементов логической схемы в модули, модулей в ячейки, ячеек в панели и т. д. Эти задачи в общем случае тесно связаны между собой, и их решение позволяет значительно сократить временные затраты и трудоемкость указанного этапа в САПР. Обычно задачи компоновки рассматриваются как процесс принятия решений в определенных или неопределенных условиях, в результате выполнения которого части логической схемы располагаются в конструктивных элементах i-го уровня, а эти элементы размещаются в конструктивных элементах $(i+1)$-го уровня и т. д., причем расположение выполняется с оптимизацией по выбранному критерию.

Компоновкой электрической схемы ЭА на конструктивно законченные части называется процесс распределения элементов низшего конструктивного уровня в высший в соответствии с выбранным критерием. Основным для компоновки является критерий электромагнитотепловой совместимости элементов низшего уровня. Данный критерий определяет область допустимых разбиений схемы, на которой формулируются другие критерии. Такими критериями могут быть: минимум типов конструктивно законченных частей, плотность компоновки, минимум соединений между устройствами, простота диагностирования схем и др. Очевидно, что внешние соединения между частями схем являются одним из

важнейших факторов, определяющих надежность ЭА. Поэтому наиболее распространенным критерием является критерий минимума числа внешних связей. Выполнение этого критерия обеспечивает минимизацию взаимных наводок, упрощение конструкции, повышение надежности и т. д. В связи с этим рассмотрение методов компоновки электрических схем будет проводиться в основном на примере критерия минимума числа внешних связей.

Исходной для решения задачи компоновки ЭА является электрическая схема соединений. Для алгоритмизации и формального решения задачи производится переход от электрической схемы соединений ЭА к графу, к мультиграфу, к гиперграфу одним из методов, описанных выше.

Сформулируем теперь задачу компоновки схемы ЭА как задачу разбиения графа $G=(X, U)$ на части $G_i=(X_i, U_i)$, $X_i \subseteq X$; $U_i \subseteq U$, $i \in I=\{1, 2, \ldots, l\}$, где l — число частей, на которое разбивается граф.

Совокупность частей $B(G)$ называется разбиением (разрезанием) графа $G=(X, U)$ если

$$\forall G_i \in B(G)\,(G_i \neq \varnothing), \quad i \in I; \quad \forall\, G_i, G_j \in B(G)[G_i \neq G_j \Rightarrow X_i \cap X_j = \varnothing \wedge (U_i \cap U_j = U_{i,j} \vee U_i \cap U_j = \varnothing)], \quad \bigcup_{i \in I} G_i = G. \tag{3.1}$$

Другими словами, совокупность $B(G)=\{G_1, G_2, \ldots, G_i, \ldots, G_l\}$ является разбиением графа G, если любая часть из этой совокупности не пустая, если для любых двух частей из $B(G)$ пересечение множества вершин пусто, а пересечение множества ребер может быть не пустым, а также если объединение всех частей в точности равно графу G.

В выражении (3.1) $U_{i,j}$ определяет подмножество ребер $U_{i,j} \subseteq U$, попадающих в разрез (сечение) между частями G_i и G_j графа G. Например, пусть задан граф G, содержащий 8 вершин и 17 ребер. Одно из возможных разбиений графа G на три части показано на рис. 3.1.

Обозначим $|U_{i,j}|=k_{i,j}$ и назовем его *числом реберного соединения частей* G_i и G_j графа G. Число реберного соединения всех частей графа

$$K=1/2 \sum_{i=1}^{l} \sum_{j=1}^{l} k_{i,j}, \; i \neq j. \tag{3.2}$$

В рассмотренном примере число реберного соединения всех трех частей графа $K=12$.

Задачей разбиения графа $G=(X, U)$ является нахождение такой совокупности частей, при которой число реберного соединения графа G удовлетворяло заданному критерию оптимальности.

Пусть граф G разбит на части $G_1, G_2, \ldots, G_l$. В соответствии с этим разбиением множество ребер U графа G можно представить в виде

$$U=U_1 \cup U_2 \cup \ldots \cup U_l = \bigcup_{i=1}^{l} U_i. \tag{3.3}$$

Тогда каждое подмножество U_i запишем следующим образом:

$$\left.\begin{aligned} U_1 &= U_{1,1} \cup U_{1,2} \cup \ldots \cup U_{1,i} \ldots \cup U_{1,l}; \\ U_2 &= U_{2,1} \cup U_{2,2} \cup \ldots \cup U_{2,i} \ldots \cup U_{2,l}; \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ U_i &= U_{i,1} \cup U_{i,2} \cup \ldots \cup U_{i,i} \ldots \cup U_{i,l}; \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ U_l &= U_{l,1} \cup U_{l,2} \cup \ldots \cup U_{l,i} \ldots \cup U_{l,l}, \end{aligned}\right\} \qquad (3.4)$$

где U_i — подмножество всех ребер, инцидентных вершинам X_i части G_i; $U_{i,i}$ — подмножество ребер, соединяющих подмножество вершин X_i внутри G_i между собой; $U_{i,j}(U_{j,i})$ — подмножество ребер, соединяющих части G_i и G_j между собой.

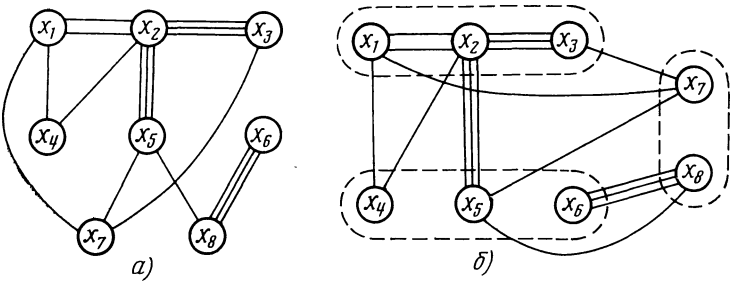


Рис. 3.1. Разбиение графа G (*а*) на три части: G_1, G_2, G_3 (*б*)

Назовем отношение суммарного числа внутренних ребер (ребер подмножеств $U_{i,i}$) к суммарному числу соединяющих ребер (ребер подмножеств $U_{i,j}$) *коэффициентом разбиения* $\Delta(G)$ *графа* G:

$$\Delta(G) = \sum_{i=1}^{l} |U_{i,i}| \Big/ \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} |U_{i,j}| = \sum_{i=1}^{l} |U_{i,i}| \Big/ K. \qquad (3.5)$$

Коэффициент разбиения $\Delta(G)$ служит для оценки разбиения графа G на части, его можно также использовать для сравнения разбиения графов. Так, для графа на рис. 3.1 $\Delta(G) = 5/12 \approx 0{,}4$.

Под оптимальным разбиением графа $G = (X, U)$ будем понимать такое разбиение $B(G)$, при котором выполняется условие $K = \min$, т. е. критерием оптимальности является минимальное число ребер, соединяющих части графа между собой.

Задача разбиения графа G на части относится к задачам комбинаторно-логического типа, получение оптимального решения которых связано с полным перебором различных вариантов разбиения.

Разбиение может быть выполнено двумя путями: упорядоченным и неупорядоченным. При упорядоченном разбиении графа $G = (X, U)$ для полученных частей устанавливается определенный порядок. Число упорядоченных разбиений B_y графа $G = (X, U)$,

$|X|=n$ на l частей $G_1=(X_1, U_1)$, $G_2=(X_2, U_2)$, ..., $G_l=(X_l, U_l)$ с числом вершин в каждой части соответственно $|X_1|=n_1$, $|X_2|=n_2, \ldots, |X_l|=n_l$ и

$$n_1+n_2+\ \ldots\ +n_l=\sum_{i=1}^{l} n_i=n$$

определится по формуле

$$B_{\text{у}}=n!/\prod_{i=1}^{l}(l_i!). \tag{3.6}$$

Например, число упорядоченных разбиений графа на рис. 3.1 на три части по 3,3 и 2 вершины в каждом равно

$$B_{\text{у}}=\frac{8!}{3!\,3!\,2!}=560.$$

Число неупорядоченных разбиений графа $G=(X, U)$, $|X|=n$ на l частей $G_1, G_2, \ldots, G_l (n_1+n_2+\ldots+n_l=n)$

$$B_{\text{н}}=B_{\text{у}}/\prod_{j=1}^{n} P_j!, \tag{3.7}$$

где P_j определяет число частей, имеющих по j элементов. Величина $1/\prod_{j=1}^{n}(P_j!)$ введена в выражение (3.7) для учета эквивалентных разбиений, связанных с упорядочением частей графа. Например, комбинации подмножеств вершин X_1, X_2 и X_2, X_1 при разбиении графа $G=(X, U)$ на $G_1=(X_1, U_1)$ и $G_2=(X_2, U_2)$ являются эквивалентными неупорядоченными разбиениями, и считать их различными при разбиении моделей схем нецелесообразно. Например, для графа на рис. 3.1,*а* число неупорядоченных разбиений составит

$$B_{\text{н}}=B_{\text{у}}/\prod_{i=1}^{l}(P_j!)=560/2!1!=280.$$

При разбиении графа схемы на конструктивно законченные части необходимость учета эквивалентных разбиений очевидна, так как упорядочение в разбиении не приводит к изменению числа внешних ребер. Поэтому в дальнейшем будут рассматриваться только неупорядоченные разбиения.

Поскольку число возможных разбиений, как видно из (3.6), (3.7), достаточно велико, возникает задача отыскания такого разбиения, при котором либо $\Delta(G)$ принимает наибольшее значение, либо K — наименьшее.

Для получения оптимального результата при разбиении графа G на части можно предложить тривиальный метод, заключающийся в получении всех возможных разбиений и выборе из него наилучшего. Такой путь для конструктора неприемлем, так как он чрезвычайно затруднителен даже для современных быстродейст-

вующих ЭВМ. Наибольшее применение нашли эвристические алгоритмы разбиения. Они позволяют за конечное число шагов получать локальный минимум, который в общем случае достаточен для практических целей.

Введем некоторые классы разбиений. *Поэлементным* назовем такое разбиение графа $G=(X, U)$, при котором в каждую часть $G_i=(X_i, U_i)$ попадает по одной вершине. В этом случае число реберного соединения всех частей равно общему числу ребер графа, т. е. $K=m$.

Разбиение назовем *целым,* если $B(G_i)=\{G\}$. Число реберного соединения всех частей при целом разбиении равно нулю. Для графа G, изображенного на рис. 3.1,*а*, поэлементное и целое разбиения показаны на рис. 3.2. Очевидно, поэлементное и целое разбиения тривиальны, но играют важную роль при разработке эвристических алгоритмов. В частности, на практике, если это возможно, нужно стремиться к целому разбиению схемы. Другими словами размещать всю схему в одном корпусе ИМС.

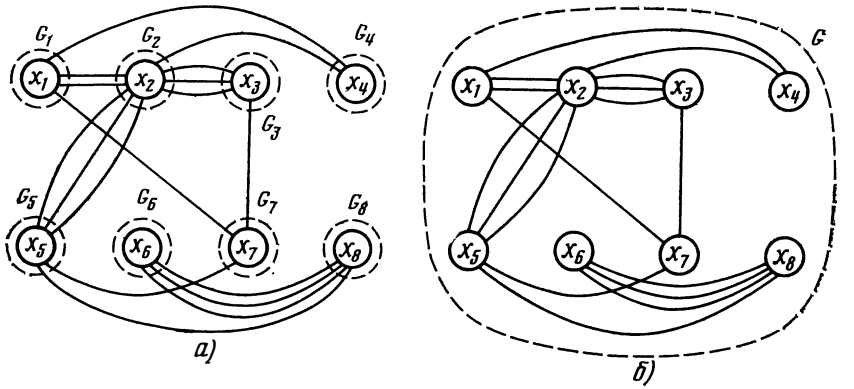


Рис. 3.2. Поэлементное (*а*) и целое (*б*) разбиения графа на рис. 3.1,*а*

Существует большое число алгоритмов разбиения графа, которые можно условно подразделить следующим образом: последовательные; итерационные, основанные на идеях математического программирования; смешанные.

Суть последовательных алгоритмов разбиения графа заключается в выборе по определенному правилу вершины или группы вершин, к которым затем присоединяются другие вершины графа с целью образования первой части. Далее процесс повторяется для второй части и т. д.

При использовании итерационных алгоритмов сначала граф разбивается на определенное число частей произвольным образом. Затем по некоторым правилам производится перестановка вершин из одной части в другую с целью минимизации числа внешних ребер.

В алгоритмах разбиения, опирающихся на идеи математического программирования, в основном используются методы ветвей и границ и решение задачи о назначении.

Алгоритмы разбиения, использующие методы ветвей и границ, состоят из следующих этапов. Сначала определяется нижняя оценка разбиения графа на заданное число частей. Затем производится построение дерева решений и осуществляется поиск оптимального результата. Задачу разбиения графа схемы на части можно свести к задаче о назначении так. Ищется назначение кандидатов (вершин графа) на все части, дающее минимальные суммарные затраты, причем каждая вершина графа может быть назначена только в одну часть и в каждой части должны содержаться различные вершины графа.

К классу смешанных алгоритмов можно отнести алгоритмы разбиения, не вошедшие в рассмотренные.

Важной задачей в общей проблеме конструирования является покрытие, т. е. преобразование функциональных схем в принципиальные. Под покрытием схемы понимается представление функциональной схемы конструктивными элементами, на которых она будет реализована, и связями между ними.

Как правило, форма и результат конструктивной реализации схемы в значительной степени предопределяет надежность работы устройства. Решение задачи покрытия дает возможность представить функциональную схему проектируемого устройства в виде принципиальной схемы соединения электрических элементов. Однако элементы могут быть различными (резисторы, конденсаторы, транзисторы, интегральные схемы и т. д.). Поэтому при решении задачи покрытия можно рассматривать вопросы выбора класса элементов и минимизации числа их типов. Конечной целью покрытия является выбор оптимальной элементно-технической базы проектируемого устройства.

Обычно функциональная схема представляется графом $D=(X, U)$, множество вершин X которого интерпретирует логические элементы, а множество ребер U — связи. Пусть функциональные схемы ЭА необходимо покрыть заданным комплексом интегральных микросхем. Тогда каждую компоненту интегральной микросхемы можно представить в виде подграфа D'_i. В результате получим некоторое множество подграфов, соответствующих интегральным микросхемам заданного комплекса.

Задача покрытия сформулируется теперь как покрытие графа D подграфами из множества $M=\{D'_1, D'_2, \ldots\}$. Можно предложить следующий путь решения этой задачи. Каждой вершине графа D присваивается вес S_{hq}, где h определяет номер вершины графа и q — тип элемента логической схемы, соответствующего выбранной вершине графа. Каждому ребру графа ставится в соответствие значение функции веса

$$q_{i,j}=\begin{cases}1, \text{ если ребро } u_{i,j} \text{ инцидентно однотипным вершинам;}\\ 0 \text{ в противном случае.}\end{cases}$$

Внутри интегральных микросхем обычно объединяются однотипные компоненты. Поэтому из нескольких вариантов покрытия схемы выбирается тот, для которого

$$q = 1/2 \sum_{i=1}^{n} \sum_{j=1}^{n} q_{i,j}$$

минимальна или близка к минимуму.

С целью улучшения результата покрытия, полученного на предыдущем этапе, можно производить парные перестановки элементов. Для этого необходимо менять местами однотипные элементы разных интегральных микросхем так, чтобы это приводило к увеличению их внутренних связей. После перебора перестановок получается окончательный вариант покрытия, который представляется в форме, пригодной для использования на дальнейших этапах конструкторского проектирования.

Во многих случаях современная технология выдвигает требования, связанные с унификацией и стандартизацией методики покрытия. Для интегральных микросхем важнейшим становится этап представления функциональных, а чаще принципиальных схем электронной аппаратуры в виде набора типовых ячеек. Это позволяет сократить время проектирования.

Как правило, типовой элемент стремятся спроектировать в виде одной специальной интегральной микросхемы. Однако сложность и объем принципиальной схемы, а также недостаточно развитая база приводят к необходимости изготовления типового элемента в виде нескольких микросхем, расположенных на одной подложке. В связи с тем, что число таких микросхем невелико, задача разбиения может быть решена методами линейного программирования.

В тех случаях, когда граф схемы содержит небольшое число вершин $n < 100$ и ограничение во времени решения не является критическим, можно ставить задачу получения точного результата при разбиении. Для этой цели эффективно используются методы математического программирования. Указанное разбиение наиболее целесообразно осуществлять на высшем иерархическом уровне для разбиения схем ИМС4, ИМС5.

Пусть дан граф микросхемы $G = (X, U)$, где $x_i \in X$ — элемент множества вершин, $i \in I = \{1, 2, \ldots, n\}$; $n = |X|$, $u_p \in U$ — элемент множества ребер, $p \in P = \{1, 2, \ldots, m\}$, $m = |U|$. Граф G обычно задается матрицей смежности $\mathbf{R} = \|r_{i,j}\|$ или списком ребер.

Необходимо разбить его на l частей:

$$G_1 = (X_1, U_{1,1}); \; G_2 = (X_2, U_{2,2}); \; \ldots; \; G_l = (X_l, U_{l,l});$$

$$X_1 \cup X_2 \cup \ldots \cup X_l = X; \; U_1 \cup U_2 \cup \ldots \cup U_l = U; \; U_i = U_{i,i} \cup U_{i,m},$$

где $U_{i,i}$ — множество ребер, принадлежащих G_i; $U_{i,m}$ — множество ребер, соединяющих G_i с остальными частями графа G, $i, m = 1, 2, \ldots, l$. Граф G надо разбить на части так, чтобы мощность множества

$$\left| \bigcup_{\substack{m=1 \\ i=1}}^{l} U_{i,m_i} \right| = K, \quad i \neq m, \tag{3.8}$$

была минимальна.

Разобъем множество ребер U графа G на подмножества U^i, причем в подмножество U^i включим ребра, инцидентные вершине x_i. $U^1 \cup U^2 \cup \ldots \cup U^i \cup \ldots \cup U^n = U$, $U^i = \{u^i_p, u^i_s, \ldots\}$. Эти подмножества могут быть определены по i-й строке матрицы смежности R графа G.

Построим специального вида граф V, который определяется множеством вершин $F = \{U^1, U^2, \ldots, U^n\}$ и условием: U^i и U^j смежны тогда и только тогда, когда $i \neq j$ и $U^i \cap U^j \neq \varnothing$. Далее будем оперировать графом V.

Задачу разбиения графа G можно свести к задаче о назначениях множества вершин F графа V в l групп при выполнении следующих условий: каждая вершина может быть назначена только в одну группу, в каждую группу должно быть назначено заданное число вершин n_l.

Введем булеву переменную

$$y_{i,q} = \begin{cases} 1, \text{ если вершина } u^i \text{ назначена в } q\text{-ю группу;} \\ 0 \text{ в противном случае.} \end{cases}$$

Согласно описанным условиям должны выполняться следующие ограничения:

$$\sum_{q=1}^{l} y_{i,q} = 1; \quad \sum_{i=1}^{n} y_{i,q} = n_q; \quad y_{i,q} \in \{0,1\}, \tag{3.9}$$

$$i = 1, 2, \ldots, n; \quad q = 1, 2, \ldots, l.$$

Так как каждое ребро графа может быть инцидентно только двум вершинам x_i, x_j, то оно входит в состав двух и только двух подмножеств U^i, U^j, $i \neq j$. Если подмножества U^i и U^j, которые являются вершинами графа V, будут назначены в одну группу, то ребро u_p будет внутренним. Если подмножества U^i и U^j поместить в разные группы, то ребро u_p будет соединительным, связывающим эти группы. Заметим, чтобы ребро u_p было внутренним, необходимо, чтобы в составе подмножеств, помещенных в одну группу, оно встречалось 2 раза, а для того, чтобы оно было соединительным, — 1 раз.

Если просуммировать элементы подмножеств U^i, помещенных в одну группу, по модулю 2, то получим число k_q соединительных ребер рассматриваемой группы, которое запишется так:

$$k_q = \sum_{i=1}^{n} \left(\sum_p u^i_p\right) y_{i,q} \pmod 2,$$

$$u^i_p \in U^i. \tag{3.10}$$

Если x_i и x_j — смежные вершины, то $u^i_p \equiv u^j_p \equiv u_p$. Так как необходимо минимизировать общее число K, то целевая функция примет вид: минимизировать

$$K=\sum_{i=1}^{n}(\sum_{p}u^{i}{}_{p})\,y_{i,1}\,(\mathrm{mod}\ 2)+\ \ldots\ +\sum_{i=1}^{n}(\sum_{p}u^{i}{}_{p})\,y_{i,q}\,(\mathrm{mod}\ 2)\qquad(3.11)$$

при выполнении ограничений (3.9). Если общие элементы, входящие в подмножества U^i, вынести за скобки, то выражение (3.11) примет вид: минимизировать

$$K=\sum_{p=1}^{m}u_p\,(y_{i,1}+y_{j,1})\,(\mathrm{mod}\ 2)+\ \ldots\ +\sum_{p=1}^{m}u_p\,(y_{i,q}+y_{j,q})\,(\mathrm{mod}\,2).$$

Так как $y_{i,q}$ — булева переменная, то можно утверждать, что $(y_{i,q}+y_{j,q})\,(\mathrm{mod}\,2)\equiv|y_{i,q}-y_{j,q}|$. В этом случае целевая функция запишется следующим образом: минимизировать

$$K=\sum_{p=1}^{m}u_p|y_{i,1}-y_{j,1}|+\ \ldots\ +\sum_{p=1}^{m}u_p|y_{i,q}-y_{j,q}|.\qquad(3.12)$$

Для однозначного решения уравнения (3.12) введем дополнительные переменные $z_{p,q}$, приняв

$$|y_{i,q}-y_{j,q}|\leqslant z_{p,q};\ \ z_{p,q}-(y_{i,q}-y_{j,q})\geqslant 0\quad\text{и}\quad z_{p,q}+(y_{i,q}-y_{j,q})\geqslant 0.\qquad(3.13)$$

Тогда задача минимизации числа соединительных ребер графа при его разбиении окажется эквивалентной следующей задаче целочисленного программирования: минимизировать

$$K=\sum_{p=1}^{m}u_pz_{p,1}+\ \ldots\ +\sum_{p=1}^{m}u_pz_{p,l}$$

при ограничениях (3.9) и (3.13).

Если расположение некоторых вершин в определенных группах задается заранее, т. е. $y_{i,q}=1$, то число ограничений уменьшается и решение задачи упрощается. Современные ЭВМ могут решать задачи математического программирования, содержащие несколько сотен ограничений, а в некоторых случаях — и 1000.

Пример. Пусть задан граф (рис. 3.3,*а*), который необходимо разбить на три части с $n_1=3$, $n_2=3$, $n_3=2$ числом вершин в каждом. Тогда целевая функция запишется следующим образом: $K=u_1|y_{2,1}-y_{8,1}|+u_2|y_{2,1}-y_{4,1}|+u_3|y_{8,1}-y_{4,1}|+u_4|y_{7,1}-y_{3,1}|+u_5|y_{5,1}-y_{3,1}|+u_6|y_{7,1}-y_{5,1}|+u_7|y_{8,1}-y_{7,1}|+u_8|y_{6,1}-y_{4,1}|+u_9|y_{6,1}-y_{1,1}|+u_1|y_{2,2}-y_{8,2}|+u_2|y_{2,2}-y_{4,2}|+u_3|y_{8,2}-y_{4,2}|+u_4|y_{7,2}-y_{3,2}|+u_5|y_{5,2}-y_{3,2}|+u_6|y_{7,2}-y_{5,2}|+u_7|y_{8,2}-y_{7,2}|+u_8|y_{6,2}-y_{4,2}|+u_9|y_{6,2}-y_{1,2}|+u_1|y_{2,3}-y_{8,3}|+u_2|y_{2,3}-y_{4,3}|+u_3|y_{8,3}-y_{4,3}|+u_4|y_{7,3}-y_{3,3}|+u_5|y_{5,3}-y_{3,3}|+u_6|y_{7,3}-y_{5,3}|+u_7|y_{8,3}-y_{7,3}|+u_8|y_{6,3}-y_{4,3}|+u_9|y_{6,3}-y_{1,3}|;$

$y_{1,1}+y_{1,2}+y_{1,3}=1;\ y_{2,1}+y_{2,2}+y_{2,3}=1;\ y_{3,1}+y_{3,2}+y_{3,3}=1;\ y_{4,1}+y_{4,2}+y_{4,3}=1;$

$y_{5,1}+y_{5,2}+y_{5,3}=1;\ y_{6,1}+y_{6,2}+y_{6,3}=1;\ y_{7,1}+y_{7,2}+y_{7,3}=1;\ y_{8,1}+y_{8,2}+y_{8,3}=1;$

$y_{1,1}+y_{2,1}+y_{3,1}+y_{4,1}+y_{5,1}+y_{6,1}+y_{7,1}+y_{8,1}=3;\ y_{1,2}+y_{2,2}+y_{3,2}+y_{4,2}+y_{5,2}+y_{6,2}+y_{7,2}+y_{8,2}=3.$

Полагаем $u_p=1$. Решая данную задачу, находим значения переменных, минимизирующих целевую функцию $y_{8,1}=1;\ y_{2,1}=1;\ y_{4,1}=1;\ y_{7,2}=1;\ y_{3,2}=1;$

$y_{5,2}=1$; $y_{1,3}=1$; $y_{6,3}=1$. Остальные значения $y_{i,q}$ — нулевые. В соответствии со значениями переменных $y_{i,q}$ разбиваем граф. Вершины x_2, x_4, x_8 помещаются в первую группу; x_3, x_5, x_7 — во вторую; x_1, x_6 — в третью. На рис. 3.3,*б* представлен граф разбиения.

Отметим, что приведенная модель решения задачи разбиения графа пригодна и для разбиения мультиграфа. Для этого в уравнения 3.12 вместо u_p нужно подставить соответствующие значения кратности ребер.

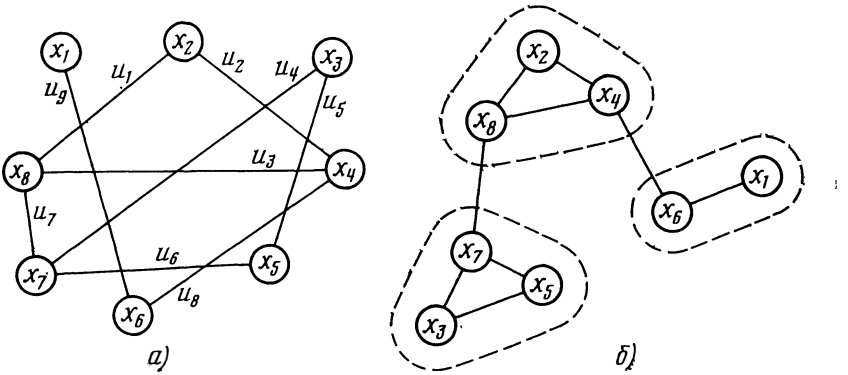


Рис. 3.3. Граф до (*а*) и после (*б*) разбиения на три части

Выше был сформулирован критерий разбиения графа на части с минимизацией числа соединяющих ребер. Если так формировать части $G_i=(X_i, U_i)$ графа G, чтобы каждая содержала возможно большее число ребер в множестве $U_{i,i}$, то нетрудно видеть, что при этом будем получать локальный минимум K.

Формирование частей графа схемы в соответствии с записанным положением составляет суть последовательных алгоритмов разбиения. Опишем несколько алгоритмов последовательного типа.

Пусть задан граф схемы $G=(X, U)$, который необходимо разбить на l частей $G_1, G_2, \ldots, G_l$ с числом вершин в каждой соответственно $n_1, n_2, \ldots, n_l (n_1+n_2+\ldots+n_l=n)$.

Работа начинается с формирования первой части G_1. В графе G определяется вершина $x_i \in X$ с наименьшей локальной степенью $\rho(x_i)=\min$. Если таких вершин несколько, то предпочтение отдается той вершине, которая имеет большее число кратных ребер. С этой вершины начинается построение. С этой целью в G_1 первоначально включаются x_i и все вершины, смежные ей. Обозначим это множество Γx_i. Если полученное число вершин равно n_1, то G_1 образована. Если это число больше n_1, то удаляем «лишние» вершины, связанные с остающимися вершинами G меньшим числом ребер. В случае, когда мощность множества Γx_i меньше n_1, то из Γx_i выбирается вершина, удовлетворяющая условию

$$\sigma(x_j)=\rho(x_j)-a_j=\max_{x_j \in \Gamma x_i} \sigma(x_j),$$

где a_j — число ребер, соединяющих вершину x_j со всеми невыбранными вершинами. Строим множество вершин Γx_j, смежных x_j, и процесс выборки вершин G_1 повторяется. Образованный подграф G_1 исключаем из исходного. Получаем граф $G^*=(X^*, U^*)$, где $X^*=X \setminus X_1$, $U^*=U \setminus U_1$. Далее в графе G^* выбирается вершина с наименьшей локальной степенью. Производится ее помещение в G_2, и процесс повторяется до тех пор, пока граф G не будет разрезан на l частей.

Нетрудно видеть, что описанный алгоритм, прост, позволяет быстро получать результаты разбиения, однако в общем случае может привести к неоптимальным результатам. Наибольшая эффективность данного метода последовательного разбиения графа имеет место, когда число вершин графа G значительно больше числа вершин в любой части разбиения $n \gg n_1, n_2, \ldots, n_l$.

Рассмотрим пример работы алгоритма. Пусть граф G (рис. 3.4) задан матрицей смежности

$$\mathbf{R}= \begin{array}{c|ccccccc|l} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & \\ \hline x_1 & 0 & 2 & 3 & 0 & 0 & 0 & 0 & \rho(x_1)=5, \\ x_2 & 2 & 0 & 1 & 1 & 0 & 0 & 0 & \rho(x_2)=4, \\ x_3 & 3 & 1 & 0 & 0 & 0 & 1 & 0 & \rho(x_3)=5, \\ x_4 & 0 & 1 & 0 & 0 & 1 & 4 & 0 & \rho(x_4)=6, \\ x_5 & 0 & 0 & 0 & 1 & 0 & 1 & 5 & \rho(x_5)=7, \\ x_6 & 0 & 0 & 1 & 4 & 1 & 0 & 1 & \rho(x_6)=7, \\ x_7 & 0 & 0 & 0 & 0 & 5 & 1 & 0 & \rho(x_7)=6. \end{array}$$

Необходимо разрезать граф G на три части: G_1, G_2, G_3, содержащих соответственно 3, 2 и 2 вершины. Определим локальную степень вершин путем суммирования элементов строк матрицы $\mathbf{R}$. Выбираем вершину x_2, обладающую наименьшей локальной степенью, и строим множество $\Gamma x_2=\{x_2, x_1, x_3, x_4\}$. Так как мощность множества $|\Gamma x_2|>n_1$, то необходимо удалить лишнюю вершину. Для этого произведем условное удаление каждой вершины из Γx_2 и подсчитаем число ребер, связывающих эту вершину с оставшимися вершинами Γx_2. При удалении вершины x_1 число ребер, соединяющих ее с $\Gamma x_2 \setminus x_1$, равно $z_{x_1}=5$, аналогично для x_3 $z_{x_3}=4$, для x_4 $z_{x_4}=1$. Следовательно, из множества Γx_2 удаляется вершина x_4. Получаем $G_1=(X_1, U_1)$, $X_1=\{x_1, x_2, x_3\}$.

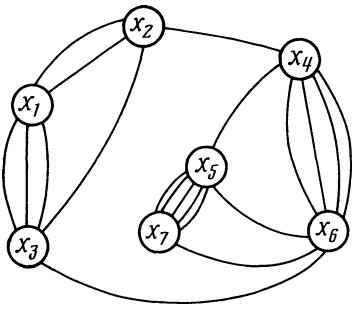


Рис. 3.4. Пример графа G для последовательного разбиения

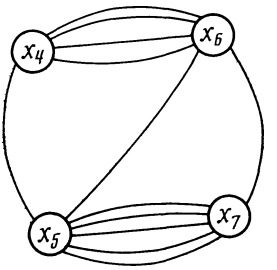


Рис. 3.5. Граф $G^*=G \setminus G_1$

Из графа G удаляем G_1, получаем граф G^*, изображенный на рис. 3.5, матрица смежности $\mathbf{R}^*$ которого примет вид

$$\mathbf{R}^* = \begin{matrix} & x_4 & x_5 & x_6 & x_7 \\ \| & 0 & 1 & 4 & 0 \\ \| & 1 & 0 & 1 & 5 \\ \| & 4 & 1 & 0 & 1 \\ \| & 0 & 5 & 1 & 0 \end{matrix} \begin{matrix} \| \\ \| \\ \| \\ \| \end{matrix} \begin{matrix} \rho(x_4)=5, \\ \rho(x_5)=7, \\ \rho(x_6)=6, \\ \rho(x_7)=6. \end{matrix}$$

Определяем локальные степени вершин. Выбираем вершину x_5 и образуем $\Gamma x_5 = \{x_4, x_5, x_6, x_7\}$. Получаем, что $|\Gamma x_5| > n_2$, поэтому производим условное удаление вершин из Γx_5 и определяем $z_{x_4}=5$, $z_{x_6}=6$, $z_{x_7}=6$. Удалим вершину x_4. Но и для оставшегося подмножества $\Gamma x_5 \setminus x_4$ мощность больше n_2. Поэтому снова производим условное удаление вершин из $\{x_5, x_6, x_7\}$ и определяем $z_{x_6}=2$, $z_{x_7}=6$. Следовательно, G_2 состоит из вершин x_5 и x_7, а G_3 соответственно будет содержать вершины x_4 и x_6.

Искомое разбиение графа G показано на рис. 3.6. Число реберного соединения графа $K=5$, а коэффициент разбиения $\Delta(G)=3$.

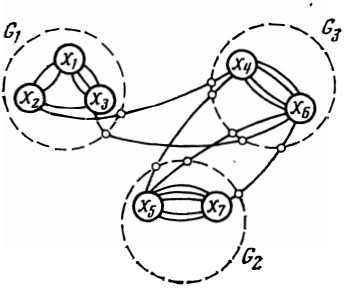


Рис. 3.6. Разбиение графа на рис. 3.4

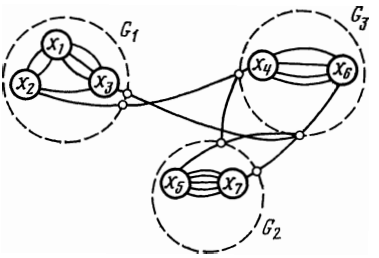


Рис. 3.7. Разбиение графа на рис. 3.4 с минимизацией C

Можно предложить и другой критерий для задачи разбиения графа на части. Это минимум числа внешних контактов C, через которые производится соединение частей между собой.

Сходство и отличие K и C покажем на примере. На рис. 3.6 и 3.7 приведены два одинаковых варианта разбиения графа G (см. рис. 3.4) на три части по критерию K и различные по критерию C. При разбиении графа рис. 3.6 $C=10$, а для варианта разбиения рис. 3.7 $C=6$. Видно, что минимизация K ведет к уменьшению C.

Заметим, что объединение ребер возможно лишь в том случае, если они принадлежат к одной и той же цепи.

Часто при разбиении ставится требование получения равных по числу вершин частей графа. Кроме того, как правило, некоторые вершины графа жестко закрепляются за определенными частями. Такие вершины называются запрещенными.

Опишем последовательный алгоритм разбиения графа, приводящий к получению локального минимума числа реберного соединения за конечное число шагов при наличии ограничений.

Пусть заданы граф схемы $G=(X, U)$ и подмножество запрещенных элементов $Q \subseteq X$, $|Q|=q$. Требуется найти такое разбиение $B(G)$ графа G на l одинаковых частей $G_1, G_2, \ldots, G_l$, чтобы суммарное число соединяющих ребер было минимальным.

Основная идея метода заключается в следующем. Перед началом работы в каждой части нет ни одной вершины графа. Сначала распределяются запрещенные вершины. Формирование первой части начинается с вершины $x_\varepsilon \in Q$, которую априорно считаем входящей в множество вершин X_1 из $G_1=(X_1, U_1)$. Составление частей разбиения будем вести по уровням. Вершина x_ε в G_1 образует первый уровень. На первом уровне множество $X_1=\{x_\varepsilon\}$. Для определения вершины следующего уровня, т. е. второй вершины, которую необходимо поместить в G_1, строится множество вершин, смежных x_ε, $\varepsilon \in E=\{1, 2, \ldots, q\}$. Обозначим это множество Γx_ε. Введем понятие относительно веса для любой вершины графа

$$\delta(x_i)=\rho(x_i)-\sum_{k=1}^{n_1} a_{i,k}, \tag{3.14}$$

где $\sum_{k=1}^{n_1} a_{i,k}$ — число ребер, соединяющих вершину x_k с вершинами подмножества X_1, $|X_1|=n_1$.

Из определения критерия компоновки следует, что для получения требуемого разбиения из множества Γx_ε необходимо выбрать вершину с минимальной величиной $\delta(x_i)$, т. е. такую вершину, для которой $\delta(x_i)=\min\limits_{x_i \in \Gamma x_\varepsilon} \delta(x_i)$, где $i \in I=\{1, 2, \ldots, t\}$, $t=|\Gamma x_\varepsilon|$. Вершина x_i является вершиной второго уровня. На втором уровне $X_1=\{x_\varepsilon, x_i\}$.

Далее рассматривается множество $\Gamma x_\varepsilon \cup \Gamma x_i$, и для каждой вершины $x_k \in (\Gamma x_\varepsilon \cup \Gamma x_i)$ определяется относительный вес по формуле (3.14). Выбирая вершину x_k с минимальным весом, получаем $X_1=\{x_\varepsilon, x_i, x_k\}$. Указанный процесс продолжается до тех пор, пока множество X_1 не будет содержать n_1 элементов. Полученная часть G_1 удаляется из графа G. Формирование последующих частей ведется аналогично.

Если при формировании части G_i графа G несколько просматриваемых вершин имеет наименьший относительный вес, то в эту часть необходимо помещать вершину, имеющую большую локальную степень. Покажем это. Пусть для вершин $x_i, x_j \in X$ графа $G=(X, U)$ локальные степени $\rho(x_i)>\rho(x_j)$, а по определению

$$\rho(x_i)-\sum_{k=1}^{t} a_{i,k}=\rho(x_j)-\sum_{k=1}^{t} a_{j,k}, \text{ т. е. } \delta(x_i)=\delta(x_j). \tag{3.15}$$

Из равенства (3.15) следует, что

$$\sum_{k=1}^{t} a_{i,k} > \sum_{k=1}^{t} a_{j,k}.$$

Общее число соединяющих ребер G_i до внесения в него вершины

x_i обозначим через f_i. Тогда при помещении в G_i вершины x_i новый относительный вес

$$\delta_H(x_i) = \delta(x_i) + f_i - \sum_{k=1}^{r} a_{i,k}, \tag{3.16}$$

а при помещении вершины x_j в G_i ее новый относительный вес

$$\delta_H(x_j) = \delta(x_j) + f_j - \sum_{k=1}^{r} a_{j,k}. \tag{3.17}$$

Сравнивая (3.16) и (3.17) с учетом того, что $\delta(x_i) = \delta(x_j)$, получаем $\delta_H(x_j) > \delta_H(x_i)$. Следовательно, в G_1 необходимо поместить вершину x_i, имеющую бо́льшую локальную степень, чем x_j.

Итак, задача разбиения графа G свелась к построению на каждом шаге множества Γx_i, его упорядочиванию и помещению в G_i вершины из Γx_i с минимальным значением $\delta(x_i)$. Проанализируем работу алгоритма на примере.

Пусть дан граф $G=(X, U)$, изображенный на рис. 3.8, матрица смежности которого имеет вид

$$\mathbf{R} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} \end{array} \\ \left\| \begin{array}{cccccccccccc} 0 & 1 & 4 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 & 3 & 1 \\ 0 & 2 & 0 & 2 & 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 6 & 1 & 1 \\ 0 & 0 & 0 & 0 & 3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 3 & 1 & 0 & 0 & 1 & 1 & 0 \end{array} \right\| \end{array} \begin{array}{l} \\ \rho(x_1)=7, \\ \rho(x_2)=8, \\ \rho(x_3)=8, \\ \rho(x_4)=8, \\ \rho(x_5)=8, \\ \rho(x_6)=5, \\ \rho(x_7)=11 \\ \rho(x_8)=4, \\ \rho(x_9)=3, \\ \rho(x_{10})=11 \\ \rho(x_{11})=5, \\ \rho(x_{12})=8. \end{array}$$

Необходимо разбить этот граф на три части по четыре вершины в каждой. Задано множество запрещенных вершин $Q=\{x_1, x_5, x_{10}\}$. Построим G_1. Выбираем вершину x_1 и записываем $X_1=\{x_1\}$. Далее рассматриваем множество $\Gamma x_1=\{x_1, x_2, x_3, x_9\}$. Вершина x_{10} в Γx_1 не включается, так как она является запрещенной. По формуле (3.14) определяем относительные веса вершин из Γx_1, кроме вершины x_1, уже вошедшей в X_1:

$$\delta(x_2)=\rho(x_2)-\sum_{k=1}^{|X_1|} a_{2,k}=8-1=7;\ \delta(x_3)=8-4=4;\ \delta(x_9)=\rho(x_9)-\sum_{k=1}^{|X_1|} a_{9,k}=3-1=2.$$

Вершину x_9 включаем в $G_1=(X_1, U_1)$. Получаем $X_1=\{x_1, x_9\}$.

Построим теперь множество $\Gamma x_1 \cup \Gamma x_9=\{x_1, x_9, x_3, x_2\}$ и определим относительные веса для вершин x_3, x_2:

$$\delta(x_3)=\rho(x_3)-\sum_{k=1}^{|X_1|} a_{3,k}=8-5=3;\ \delta(x_2)=8-2=6.$$

Вершину x_3 с наименьшим весом помещаем в G_1, тогда $X_1=\{x_1, x_9, x_3\}$. Составляем множество $\Gamma x_1 \cup \Gamma x_9 \cup \Gamma x_3=\{x_1, x_9, x_3, x_2, x_6\}$ и записываем $\delta(x_6)=$

$=5-1=4$, $\delta(x_2)=8-2=6$. Вершину x_6 включаем в G_1, получаем $X_1=\{x_1, x_9, x_3, x_6\}$, на этом образование в G_1, заканчиваем. После удаления его из графа G получаем граф $G'=G\setminus G_1$. Аналогичным образом проведем разбиение графа G' и выделим из него части G_2 и G_3. Из множества Q берем запрещенную вершину x_5, следовательно, $\Gamma x_5=\{x_5, x_2, x_7, x_8, x_4\}$, $X_2=\{x_5\}$. Относительные веса вершин из Γx_5 определяются $\delta(x_2)=5-2=3$; $\delta(x_7)=11-1=10$; $\delta(x_8)=4-3=1$; $\delta(x_4)=6$. В X_2 помещается вершина x_8, так как у нее относительный вес меньший. Тогда запишем $X_2=\{x_5, x_8\}$. Строим множество $\Gamma x_5 \cup \Gamma x_8=\{x_5, x_2, x_7, x_8, x_4\}$ и определяем новые относительные веса $\delta(x_7)=11-2=9$; $\delta(x_2)=5-2=3$; $\delta(x_4)=8-2=6$. В подмножество X_2 помещаем вершину x_2, $X_2=\{x_5, x_2, x_8\}$. Записываем $\Gamma x_5 \cup \Gamma x_2 \cup \Gamma x_8=\{x_5, x_2, x_7, x_8, x_{12}, x_4\}$ и определяем $\delta(x_7)=11-2=9$, $\delta(x_{12})=5-1=4$, $\delta(x_4)=8-3=5$. Следовательно, $X_2=\{x_5, x_2, x_8, x_{12}\}$. Тогда подмножество вершин третьей части $X_3=\{x_{10}, x_7, x_{11}, x_4\}$. Граф после разбиения показан на рис. 3.7,*б*. Число реберного соединения $K=19$, а коэффициент разбиения $\Delta(G)=24/19$. Алгоритм наиболее эффективен, если матрица, представляющая граф, содержит мало нулевых элементов.

Рассмотрим алгоритм разбиения схем на подсхемы с использованием матрицы цепей. Критерием разбиения является минимум числа внешних соединений между подсхемами. Исходной информаций для выполнения алгоритма являются матрица **T** и список запрещенных элементов. Оптимальному разбиению графа схемы соответствует разбиение матрицы **T** на подматрицы $\mathbf{T}_1, \ldots, \mathbf{T}_l$ такие, что $\mathbf{T}=\cup\mathbf{T}_i$; $\sum\limits_{i,j} |\mathbf{T}_i \cap \mathbf{T}_j| = K \to \min$.

Для подсчета числа связей $k_{i,j}$ между частями графа схемы и числа контактов электрического соединителя каждой части введем вспомогательную матрицу $\mathbf{S}=\|s_{i,j}\|_{p\times l}$, где l — число частей разбиения; p — число электрических цепей в схеме;

$$s_{i,j}=\begin{cases}1, \text{ если в } G_i \subseteq G \text{ имеется вершина, инцидентная } j \text{ цепи;}\\ 0 \text{ в противном случае.}\end{cases}$$

Матрицу **S** удобно строить непосредственно по матрице **T**, так как построение по ее графу схемы требует значительных затрат ручного труда.

Например, пусть задана матрица

$\mathbf{T}=$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
x_1	1	0	0	0	0	0	2	4	0	0	0	0	0	3
x_2	5	0	0	0	0	0	6	8	0	0	0	0	0	7
x_3	0	0	0	0	0	0	9	10	0	0	0	0	0	15
x_4	14	4	13	2	11	3	15	12	1	15	15	0	15	0
x_5	19	6	18	8	17	7	15	16	5	15	15	0	15	0
x_6	20	10	21	9	0	0	0	0	0	0	15	0	15	0
x_7	0	0	12	0	22	0	11	0	0	24	0	0	23	0
x_8	0	0	13	24	22	23	14	0	0	26	0	0	25	0
x_9	0	0	16	26	22	25	17	0	0	28	0	0	27	0
x_{10}	0	0	18	28	22	27	19	0	0	30	0	0	29	0
x_{11}	0	31	21	30	22	29	20	0	0	0	0	0	0	0
x_{12}	0	0	0	31	0	0	0	0	0	0	0	0	0	0
x_{13}	0	0	0	0	0	0	0	0	0	0	0	0	0	22

некоторого фрагмента схемы (рис. 3.9), которую необходимо разбить на три части. Произвольно разобъем схему G на три части:

$$G_1=(X_1,\ U_1);\ G_2=(X_2,\ U_2);\ G_3=(X_3,\ U_3),\ X_1=\{x_1,\ x_2,\ x_3,\ x_4\};$$

$$X_2=\{x_5,\ x_6,\ x_7,\ x_8\};\ X_3=\{x_9,\ x_{10},\ x_{11},\ x_{12},\ x_{13}\}.$$

Используя алгоритм, по матрице **T** определяем матрицу **S**:

$$\mathbf{S}=\left\|\begin{array}{ccccccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{array}\right\|$$

$$\left\|\begin{array}{ccccccccccccccc}
18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 &
\end{array}\right\|.$$

Подсчитаем теперь число связей между частями, полученными при таком разбиении. Определение число связей K основано на том, что цепь, инцидентную ξ частям, всегда можно провести так, чтобы она образовала не более чем $(\xi-1)$ внешнюю связь (соединяющее ребро).

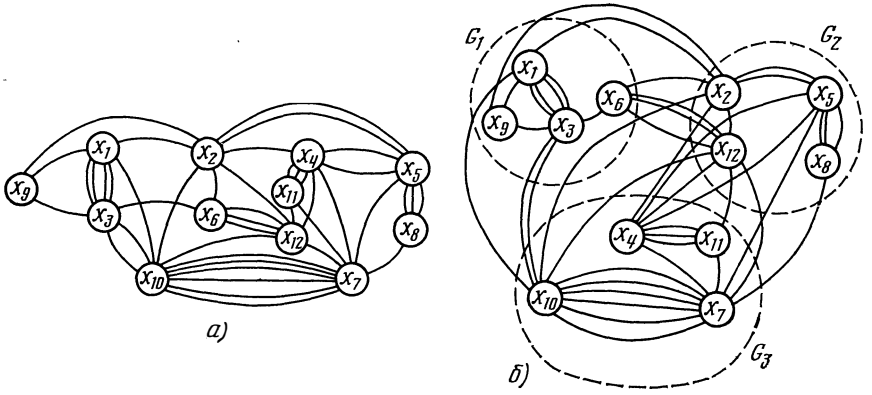


Рис. 3.8. Граф G до (*а*) и после (*б*) разбиения с использованием запрещенных вершин

Рассматривая матрицу **S**, можно заметить, что число единиц в любом из столбцов матрицы равно числу частей G_i, инцидентных цепи, номер которой совпадает с номером столбца. Поэтому для нахождения K необходимо найти сумму элементов каждого столбца, уменьшенную на единицу, и полученные результаты просуммировать. По матрице **S** легко определить число контактов соединителя каждой части разбиения. Для этого достаточно подсчитать число единиц в строке, соответствующей рассматриваемой части, и сумму уменьшить на величину, равную числу столбцов, в которых имеется единственная единица, находящаяся в этой строке. После разбиения схемы (см. рис. 3.9) произвольным образом на три части по матрице **S** найдем, что $K=20$, а число контактов соединителя каждой части соответственно равно $k_1=11$; $k_2=20$; $k_3=9$.

Рассмотрим идею алгоритма разбиения по матрицам **T** и **S** путем последовательного формирования частей разбиения с минимизацией K.

Сначала (число элементов в каждой подсхеме пусто) распределяются запрещенные элементы. Если таких нет, то разбиение начинается с предварительного распределения остальных элементов. После этого осуществляются последовательная выборка строк $l_q \in \mathbf{T}$ и определение той G_i, при помещении в которую элемент $x_q \in X$ дает наименьшее приращение внешних связей. Процесс повторяется, пока все элементы матрицы **T** не будут распределены.

Рассмотрим более подробно процесс определения приращения числа связей $\Delta K^i{}_j$. Пусть в каждой части имеется некоторое число элементов. Любая цепь q схемы образует связь между G_j и остальными частями, если она инцидентна элементам из G_j и хотя бы одному элементу из любой другой части. В матрице **S** этому случаю соответствуют наличие единицы в строке s_j на пересечении ее со столбцом q и присутствие единицы среди остальных элементов этого столбца.

Каждой строке x_i матрицы **T** можно сопоставить строку $s^i{}_0 = \|s_{i,\delta}\|_{1 \times p}$, где — $s_{i,\delta}$ — принимает значение 1, если элемент x_i подключен к цепи δ, и 0 в противном случае.

Покажем на примере схемы, изображенной на рис. 3.9, типичный шаг работы алгоритма компоновки. Исходной информацией являются граф схемы $G = (X, U)$, $|X| = 13$, $X = \{x_1, x_2, \ldots, x_{13}\}$ и его матрица цепей **T**, которая для рас-

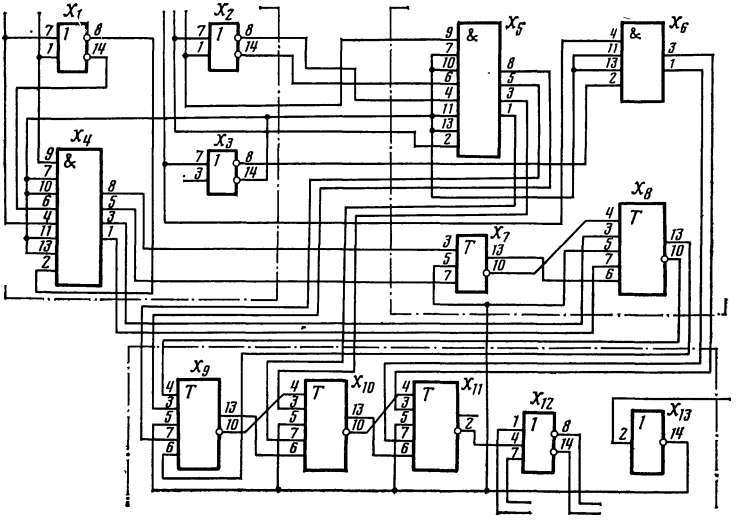


Рис. 3.9. Фрагмент схемы

сматриваемой схемы приведена выше. Граф схемы необходимо разбить на три части: $G_1 = (X_1, U_1)$; $G_2 = (X_2, U_2)$; $G_3 = (X_3, U_3)$, $\bigcup_{i=1}^{3} G_i = G$ с минимизацией числа соединительных ребер. Число вершин в частях должно быть соответственно равно $|X_1| = 4$; $|X_2| = 4$; $|X_3| = 5$.

Пусть после нескольких шагов работы алгоритма найдено, что элементы $x_4, x_1, x_7 \in X_1$; $x_2, x_5 \in X_2$; $x_3, x_6 \in X_3$. Вспомогательная матрица в этом случае примет следующий вид:

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	0	0
S=	2	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	1 →
	3	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0

	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
	0	0	0	0	1	1	1	0	0	0	0	0	0	0	‖
→	1	1	0	0	0	0	0	0	0	0	0	0	0	0	‖
	0	0	1	1	0	0	0	0	0	0	0	0	0	0	‖

Согласно алгоритму разбиения из матрицы **T** выбираем строку t_8, соответствующую нерассмотренному элементу x_8, и строим для нее вектор-строку

$$s^8_0 = 00000000000011000000011111100000.$$

Далее находим приращение внешних связей при расположении элемента x_8 в G_1, G_2 и G_3, т. е. определяем ΔK^8_1, ΔK^8_2 и ΔK^8_3. Для этого сначала определяем поразрядную дизъюнкцию вектор-строк s_2 и s_3, s_1 и s_3, s_1 и s_2:

$$s_2 = 00001111000000111110000000000$$
$$s_3 = 00000000110000100001100000000000$$

$$s_2 \vee s_3 = 00001111111000011111110000000000;$$
$$s_1 = 11110000001111100000011100000000$$
$$s_3 = 00000000110000100001100000000000$$

$$s_1 \vee s_3 = 11110000111111100001111100000000;$$
$$s_1 = 11110000001111100000011100000000$$
$$s_2 = 00001111000000111110000000000000$$

$$s_1 \vee s_2 = 11111111001111111110011100000000.$$

Затем определяем инверсии строк s_1, s_2, s_3 матрицы **S**:

$$\bar{s}_1 = 00001111111000000111111000111111;$$
$$\bar{s}_2 = 11110000111111100000111111111111;$$
$$\bar{s}_3 = 11111111001111011001111111111.$$

Для вычисления ΔK^i_j находим поразрядную конъюнкцию строк $(s_2 \vee s_3)$, $s^8_0 \bar{s}_1$; $(s_1 \vee s_3)$, $s^8_0 \bar{s}_2$; $(s_1 \vee s_2)$, s^8_0, $\bar{s}_3$;

$$(s_2 \vee s_3) s^8_0 \bar{s}_1 = 00000000000000000000000000000000;$$
$$(s_1 \vee s_3) s^8_0 \bar{s}_2 = 00000000000011000000011110000000;$$
$$(s_1 \vee s_2) s^8_0 \bar{s}_3 = 00000000000011000000011110000000.$$

Суммируя число единиц в каждом из полученных выражений, имеем $\Delta K^8_1 = 0$; $\Delta K^8_2 = 5$; $\Delta K^8_3 = 5$. Так как $\Delta K^8_1 = 0$, то элемент x_8 помещается в G_1. При этом число внешних связей не изменится. Далее строка s_1 матрицы **S** модифицируется поразрядным объединением со строкой s^8_0. Заметим, что при внесении x_8 в G_1 число элементов оказалось равным заданному. На этом формирование G_1 закончилось. Осталось распределить элементы x_9, x_{10}, x_{11}, x_{12}, x_{13} между G_2 и G_3.

Из матрицы **T** выбирается строка x_9, соответствующая элементу x_9, и выполняются те же операции, что и для строки x_8.

Аналогично поступаем с оставшимися строками x_{10}—x_{13}, пока не будут сформированы G_2 и G_3.

В рассматриваемом примере в результате работы алгоритма были сформированы следующие части: $G_1=(X_1, U_1)$; $X_1=\{x_1, x_4, x_7, x_8\}$; $G_2=(X_2, U_2)$; $X_2=\{x_2, x_5, x_9, x_{10}\}$; $G_3=(X_3, U_3)$; $X_3=\{x_3, x_6, x_{11}, x_{12}, x_{13}\}$. Число внешних связей уменьшилось с $K=20$ (для произвольного разбиения) до $K=8$. Окончательный вариант компоновки графа схемы рис. 3.9 показан на рис. 3.10. Алгоритм наиболее эффективен, когда схема содержит значительное число разветвленных цепей и элементов.

Если свести задачу разбиения к последовательному формированию выделяемых конструктивных единиц, то на языке гиперграфов эта задача формулируется следующим образом. Необходимо разбить множество вершин X гиперграфа $H=(X, E)$ на два подмножества X_1 и $X \setminus X_1$ так, чтобы минимизировать значение целевой функции

$$f=\sum_{j=1}^{m} p_j, q_j; \tag{3.18}$$

$$p_j=\begin{cases}1, \text{ если } l_j \cap X_1 \neq \varnothing \cdot \\ 0, \text{ если } l_j \cap X_1 = \varnothing;\end{cases} \qquad q_j=\begin{cases}1, \text{ если } l_j \cap (X \setminus X_1) \neq \varnothing; \\ 0, \text{ если } l_j \cap (X \setminus X_1) = \varnothing.\end{cases}$$

Таким образом, произведение p_j, q_j равно единице, если вершины ребра l_j принадлежат как X_1, так и $X \setminus X_1$, т. е. электрическая цепь, представляемая l_j, разрывается. Иначе говоря, f — это число ребер гиперграфа H, которые содержат хотя бы одну вершину из X_1 и хотя бы одну вершину из $X \setminus X_1$ одновременно.

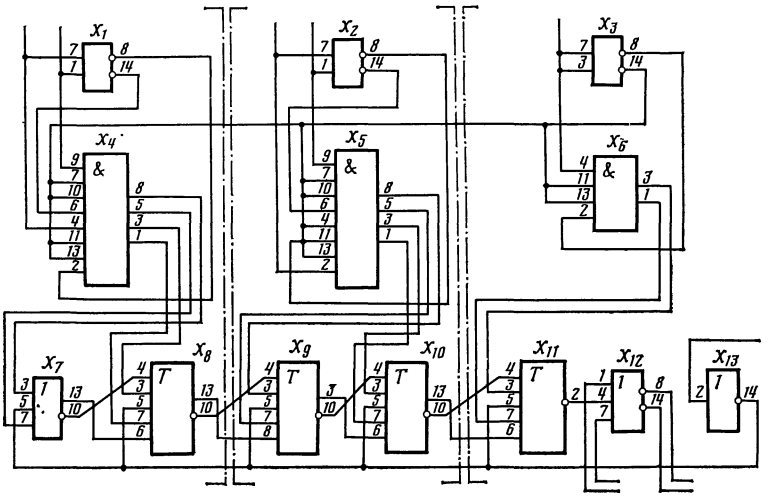


Рис. 3.10. Окончательный вариант компоновки ($K=8$)

Для построения последовательного алгоритма разбиения множества вершин X гиперграфа на X_1 и $X \setminus X_1$ необходимо ввести оценку каждой вершины гиперграфа относительно уже сформиро-

ванной части множества X_1. Для получения оценок удобно использовать матрицу инцидентности гиперграфа $H=(X, E)$

$$\mathbf{I}=\|r_{i,j}\|_{n\times m}, \qquad r_{i,j}=\begin{cases}1, \text{ если } x_i\in l_j;\\ 0, \text{ если } x_i\notin l_j.\end{cases}$$

Пусть гиперграф задан следующей матрицей

$$\mathbf{I}=\begin{array}{c} \\ x_1\\ x_2\\ x_3\\ x_4\\ x_5\\ x_6\\ x_7\\ x_8\\ x_9\\ x_{10}\end{array}\begin{array}{c}\begin{array}{ccccccccccc} l_1 & l_2 & l_3 & l_4 & l_5 & l_6 & l_7 & l_8 & l_9 & l_{10} & l_{11}\end{array}\\ \left\|\begin{array}{ccccccccccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0\\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1\\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0\\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1\end{array}\right\|\end{array}.$$

Пусть некоторые вершины $x_\alpha\in X$ уже вошли в множество X_1. Тогда для произвольной вершины $x_t\in X\setminus X_1$ предлагается оценка $\alpha(x_t)$ следующего вида:

$$\alpha(x_t)=\sum_{j=1}^{m} r_{t,j}(q'_j-p_j), \tag{3.19}$$

где $r_{t,j}$ — элемент матрицы $\mathbf{I}$;

$$q'_j=\begin{cases}1, \text{ если } l_j\cap[X\setminus(X_1\cap\{x_t\})]=\varnothing;\\ 0 \text{ в противном случае.}\end{cases}$$

Из этого видно, что величина q'_j равна единице только в том случае, если ребро l_j кроме вершины x_t соединяет хотя бы одну вершину из $X\setminus X_1$, т. е. электрическая цепь, представляемая ребром l_j, при переносе элемента x_t в выделяемую область разрывается. Величина p_j равна единице, если хотя бы один элемент из l_j уже включен в множество X_1. Таким образом, оценка $\alpha(x_t)$ отражает изменение числа разрезаемых цепей при переносе элемента t схемы из одной области в другую и может принимать значения положительные, отрицательные и равные нулю. Исходя из содержательного смысла оценки $\alpha(x_i)$ каждой вершины $x_i\in X\setminus X_1$, предлагается включать в множество X_1 на данном шаге ту вершину x_t, для которой

$$\alpha(x_t)=\min_{x_i\in X\setminus X_1}(\alpha(x_i)). \tag{3.20}$$

Запишем алгоритм в виде логической схемы алгоритмов

$$A_0A_1\downarrow^1A_2A_3p_1\uparrow^1A_к,$$

где A_0, $A_к$ — операторы начала и конца алгоритма соответственно; A_1 — включение в X_1 вершины $x_\alpha \in X$; A_2 — определение $\alpha(x_i)$ для всех $x_i \in X \setminus X_1$; A_3 — определение x_t на основе (3.20) и включение ее в X_1; p_1 — логическое условие; если $p_1=1$ («да»), то после оператора A_3 выполняется $A_к$, если $p_1=0$ («нет»), то после A_3 выполняется A_2. Логическое условие проверяет, равно ли $|X_1|=n_1$.

Рассмотрим работу алгоритма на примере разбиения гиперграфа, матрица инцидентности I которого записана выше. Пусть необходимо выделить множество X_1, $|X_1|=5$. Включим произвольно вершину x_1 в множество X_1. Определим по (3.20): $\alpha(x_2)=3$; $\alpha(x_3)=1$; $\alpha(x_4)=3$; $\alpha(x_5)=1$; $\alpha(x_6)=1$; $\alpha(x_7)=3$; $\alpha(x_8)=2$; $\alpha(x_9)=2$; $\alpha(x_{10})=4$. Следовательно, можно выбрать вершины x_3, x_5, x_6. Для определенности выберем вершину с меньшим индексом и включим ее в X_1. На втором шаге $X_1=\{x_1, x_3\}$. Определим $\alpha(x_2)=3$; $\alpha(x_4)=3$; $\alpha(x_5)=1$; $\alpha(x_6)=1$; $\alpha(x_7)=3$; $\alpha(x_8)=1$; $\alpha(x_9)=1$; $\alpha(x_{10})=4$. По (3.20) определяем, что в множество X_1 необходимо включить вершину x_6. На третьем шаге получим $X_1=\{x_1, x_3, x_6\}$. Находим $\alpha(x_2)=3$; $\alpha(x_4)=3$; $\alpha(x_5)=1$; $\alpha(x_7)=3$; $\alpha(x_8)=1$; $\alpha(x_9)=0$; $\alpha(x_{10})=4$. В множество X_1 включим x_9, тогда $X_1=\{x_1, x_3, x_6, x_9\}$. Получим $\alpha(x_2)=2$; $\alpha(x_4)=3$; $\alpha(x_5)=1$; $\alpha(x_7)=2$; $\alpha(x_8)=1$; $\alpha(x_{10})=4$. В множество X_1 включим x_5. Получим $X_1=\{x_1, x_3, x_5, x_6, x_9\}$; $X \setminus X_1=\{x_2, x_4, x_7, x_8, x_{10}\}$. Разбиение закончено.

Нетрудно видеть, что результат использования последовательного алгоритма существенно зависит от выбора первой вершины и последующих вершин с равными оценками.

Выше были рассмотрены методы разбиения, при которых части разбиения формируются путем последовательного присоединения вершин по определенным критериям. Представляет интерес рассмотрение такого последовательного метода, при котором образуются всевозможные разбиения данного графа G с минимизацией числа реберного соединения.

Предварительно введем понятие массива вершин в графе $G=(X, U)$. Любой массив вершин H представляет собой подмножество множества вершин X, т. е. $H \subseteq X$. Естественно, что вершины $x \in H$ соединены внутренними ребрами между собой и внешними ребрами с остальными вершинами $X \setminus H$ графа G. Число ребер, соединяющих вершины массива H с вершинами $X \setminus H$, будем обозначать малой буквой h.

Массив H называется минимальным, если для любого другого массива B, $B \subset H$, выполняется условие $h<b$, т. е. удаление каких-либо вершин из H приводит к увеличению числа внешних ребер. По определению минимальный массив не может быть пустым. Кроме того, принимается, что каждая вершина графа G образует минимальный массив. Например, на рис. 3.11 показан минимальный массив H, для которого $h=3$. Нетрудно видеть, что выделение всех минимальных массивов возможно с помощью рассмотрения всех подмножеств множества X. Если мощность множества $|X|=n$, то известно, что число подмножеств множества X составляет 2^n. Следовательно, возникает необходимость исследовать свойства минимальных массивов с целью существенного сокращения числа рассматриваемых подмножеств.

Введем некоторые обозначения. Пусть $H \subset X$ — некоторый массив, состоящий из непересекающихся массивов $H_1, H_2, \ldots, H_l$,

причем $\bigcup_{i=1}^{l} H_i = H$. В общем случае H_i может представлять собой отдельную вершину. Обозначим через K_{H_i} число ребер, соединяющих H_i с вершинами $X \setminus H$, т. е. число внешних ребер относительно всего массива H. Обозначим через K_{H_i, H_j} число ребер, соединяющих массивы H_i и H_j между собой. Тогда число $h = \sum_{i=1}^{l} K_{Hi}$.

На рис. 3.12 показан массив H, состоящий из H_1, H_2, H_3, H_4. Сформулируем основные свойства минимальных массивов.

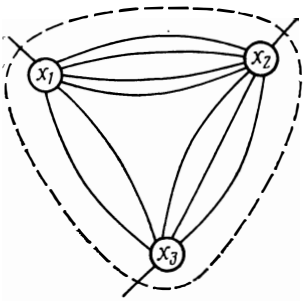


Рис. 3.11. Минимальный массив H ($h=3$)

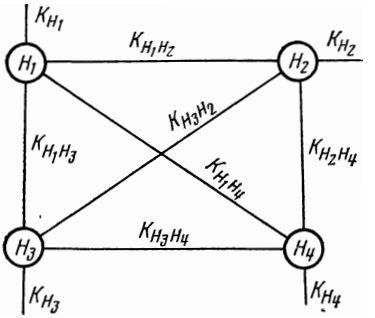


Рис. 3.12. Массив H_i

Теорема 3.1. Если массивы H_i и H_j минимальны и не включают друг друга, т. е. $H_i \not\subset H_j$ и $H_j \not\subset H_i$, то $H_i \cap H_j = \varnothing$, т. е. они не пересекаются. Покажем это.

Предположим, что $H_i \cap H_j = M \neq \varnothing$. Тогда нетрудно видеть, что $H_i = E \cup M$ и $H_j = F \cup M$, где E и F — два непустых массива, дополняющих массив M соответственно до H_i и H_j. Описанная ситуация изображена на рис. 3.13. Используя введенные обозначения, подсчитаем число внешних ребер массивов H_i, H_j, E и F: $h_i = K_E + K_M + K_{EF} + K_{MF}$; $h_j = K_F + K_M + K_{FE} + K_{ME}$; $l = K_E + K_{EF} + K_{EM}$; $f = K_F + K_{FE} + K_{FM}$.

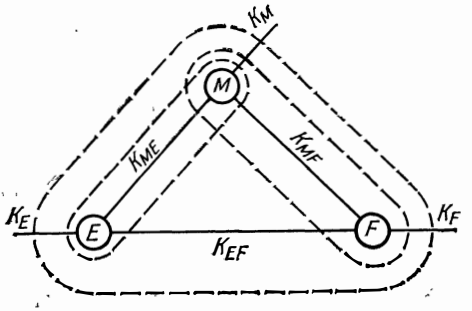


Рис. 3.13. Пример массивов H_i, H_j

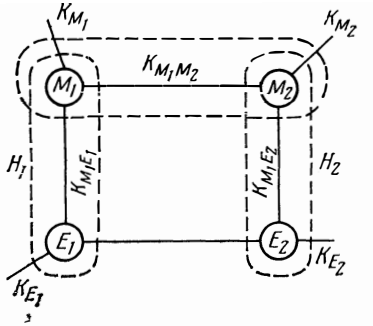


Рис. 3.14. Массивы M_1, M_2, F_1, F_2

Так как массивы H_j и H_i минимальны (по определению), то $h_i<l$, $h_j<f$. Тогда запишем $K_E+K_M+K_{EF}+K_{MF}<K_E+K_{EF}+K_{EM}$; $K_F+K_M+K_{EF}+K_{ME}<K_F+K_{FE}+K_{FM}$. Проведя сокращения, получим

$$\begin{cases} K_M+K_{MF}<K_{EM}; \\ K_M+K_{ME}<K_{FM}, \end{cases}$$

так как $K_{MF}=K_{FH}$; $K_{EM}=K_{ME}$ по определению, то

$$\begin{cases} K_M+K_{MF}<K_{ME}; \\ K_M+K_{ME}<K_{MF}. \end{cases}$$

Полученные неравенства являются противоречивыми, т. е. предположение, что $M\neq\varnothing$, ложно, тогда $H_i\cap H_j=\varnothing$.

Рассмотрим соотношения, возникающие между массивами и минимальными массивами в графе.

Следствие I. Пусть E, F и M — непустые непересекающиеся массивы, причем $H_i=E\cup M$ — минимальный массив, а $H_j=F\cup M$ — неминимальный массив. В этом случае $h_j\geqslant f$.

Следствие II. Теперь рассмотрим случай (рис. 3.14), когда H_1 и H_2 — минимальные массивы, а $M_1\subset H_1$ и $M_2\subset H_2$ — непустые массивы и $M=M_1\cup M_2$ — массив. В этом случае $m>h_1$ и $m>h_2$.

Рассмотрим теорему, определяющую отношения между минимальными массивами.

Теорема 3.2. Пусть H_1 и H_2 — два минимальных массива. Если $P=H_1\cup H_2$ и $p<h_1$, $p<h_2$, то массив P также является минимальным.

Докажем эту теорему от противного. Предположим, что массив $P=H_1\cup H_2$ не является минимальным, т. е. содержит в себе такой массив $Q\subset P$, что $p\geqslant q$. При этом массив Q может быть образован следующим образом:

а) $Q\subseteq H_1$ ($Q\subseteq H_2$). Так как H_1 (H_2) — минимальный массив, то $q\geqslant h_1$ ($q\geqslant h_2$). Но $p\geqslant q$, т. е. $p\geqslant h_1$ ($p\geqslant h_2$). Пришли к противоречию с условием теоремы.

б) $Q=M_1\cup M_2$, где $M_1\subset H_1$, а $M_2\subset H_2$. Из следствия II получаем $q>h_1$, $q>h_2$, т. е. $p>h_1$, $p>h_2$. Пришли к противоречию с условием теоремы.

в) $Q\supset H_1$ ($Q\supset H_2$). Из следствия I получаем, что $q>h_1$, $q>h_2$, т. е. $p>h_1$, $p>h_2$. Пришли к противоречию.

Таким образом, во всех рассмотренных случаях получаем противоречие, т. е. предположение, что P — неминимальный массив неверно.

Используя метод математической индукции, нетрудно распространить утверждение теоремы 3.2 на случай нескольких минимальных массивов.

Теорема 3.3. Пусть H_1, H_2, ..., H_l — такие минимальные массивы ($l>2$), что объединение S любых из них не является минимальным массивом, где $2\leqslant s\leqslant(l-1)$. Если $P=H_1\cup H_2\cup\ldots\cup H_l$ и $p<h_1$, $p<h_2$, ..., $p<h_s$, то массив P является минимальным.

Теоремы 3.1—3.3 позволяют построить алгоритм выделения минимальных массивов. Из теоремы 3.1 следует, что процедура выделения массивов сходится и решение единственно, так как вершины графа, вошедшие в какой-либо массив, можно в дальнейшем рассматривать как одну вершину.

Теоремы 3.2 и 3.3 определяют способ образования массивов как объединение уже существующих массивов и сравнение общего числа внешних ребер.

Для построения алгоритма выделения всех минимальных массивов графа G в соответствии с теоремами удобно использовать его матрицу смежности **R**. Заметим, что первоначально каждая вершина $x \in X$ графа G считается как минимальный массив с числом внешних ребер, соответствующим локальной степени $\rho(x)$ вершины x. Полученные минимальные массивы будем заносить в список. Суть алгоритма заключается в следующем.

1°. Попарно анализируются все вершины графа G. Если находится такая пара $H=(x_i, x_j)$, что $K=\rho(x_i)+\rho(x_j)-2r_{i,j}$, причем $K<\rho(x_i)$, $K<\rho(x_j)$, где $r_{i,j}$ — элемент матрицы **R**, определяющий число ребер, соединяющих x_i с x_j, то H в соответствии с теоремой 3.2 образует минимальный массив. Массив H заносится в список A, а элементы x_i, x_j из рассмотрения исключаются. Переход к 2°. Если новых минимальных массивов не образуется, то переход к 3°.

2°. Граф G факторизуется относительно полученных массивов, т. е. вершины x_i, x_j, входящие в один массив, заменяются одной вершиной $x_{i,j}$, причем ребра, соединяющие x_i, x_j между собой, исключаются, а ребра, идущие из x_i, x_j к другим вершинам, приписываются $x_{i,j}$. На матрице **R** факторизация заключается в поэлементном сложении соответствующих строк и столбцов между собой и записи нулей в диагональные элементы. Переход к 1°.

3°. Анализируются все вершины графа G', полученного в результате последней факторизации, по $\alpha=3$ (по тройкам). Если находится такая тройка $L=(x_i, x_j, x_k)$, что $l=\rho(x_i)+\rho(x_j)+\rho(x_k)-2r_{i,j}-2r_{i,k}-2r_{j,k}$, причем $l<\rho(x_i)$, $l<\rho(x_j)$, $l<\rho(x_k)$, то L в соответствии с теоремами 3.2 и 3.3 образует минимальный массив. Массив L заносится в список A. Переход к 3°. Если массивов по α нет, то переход к 4°.

4°. Параметр α увеличивается на единицу. Переход к 3°. Если число α больше или равно числу вершин графа, полученного в результате последней факторизации, то переход к 5°.

5°. Конец работы алгоритма.

Проиллюстрируем работу алгоритма на примере.

Пусть дан граф схемы, показанной на рис. 3.15. Выделим из него все минимальные массивы.

Построим матрицу смежности **R** этого графа:

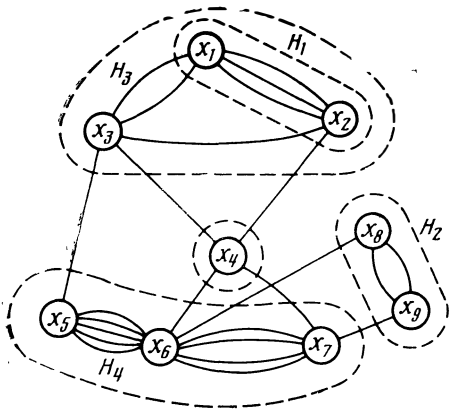


Рис. 3.15. Пример разбиения графа

$$\mathbf{R}=\begin{array}{c|ccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 0 & 3 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 5 & 0 & 0 & 1 & 0 & 0 & 5 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 1 & 5 & 0 & 4 & 1 & 0 \\ 7 & 0 & 0 & 0 & 1 & 0 & 4 & 0 & 0 & 1 \\ 8 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \end{array}.$$

Определим локальную степень каждой вершины как сумму элементов соответствующей строки или столбца: $\rho(x_1)=5$; $\rho(x_2)=5$; $\rho(x_3)=5$; $\rho(x_4)=4$; $\rho(x_5)=6$; $\rho(x_6)=11$; $\rho(x_7)=6$; $\rho(x_8)=3$; $\rho(x_9)=3$.

В результате выполнения 1° алгоритма определяем минимальные массивы $H_1=\{x_1, x_2\}$; $H_2=\{x_8, x_9\}$. Заносим их в список $A=\{H_1, H_2\}$. Относительно полученных минимальных массивов факторизуем граф G. Запишем матрицу $\mathbf{R}_1$ факторизованного графа G_1:

$$\mathbf{R}_1=\begin{array}{c|ccccccc} & (1,2) & 3 & 4 & 5 & 6 & 7 & (8,9) \\ (1,2) & 0 & 3 & 1 & 0 & 0 & 0 & 0 \\ 3 & 3 & 0 & 1 & 1 & 0 & 0 & 0 \\ 4 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 5 & 0 & 1 & 0 & 0 & 5 & 0 & 0 \\ 6 & 0 & 0 & 1 & 5 & 0 & 4 & 1 \\ 7 & 0 & 0 & 1 & 0 & 4 & 0 & 1 \\ (8,9) & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}.$$

Выполняя 1° алгоритма, по матрице $\mathbf{R}_1$ определяем минимальный массив $H_3=\{x_{1,2}, x_3\}=\{x_1, x_2, x_3\}=\{H_1, x_3\}$. Заносим его в список $A=\{H_1, H_2, H_3\}$. Факторизуем граф G_1 относительно полученного массива H_3. Получаем граф G_2 с матрицей смежности

$$\mathbf{R}_2=\begin{array}{c|ccccccc} & (1,2,3) & 4 & 5 & 6 & 7 & (8,9) \\ (1,2,3) & 0 & 2 & 1 & 0 & 0 & 0 \\ 4 & 2 & 0 & 0 & 1 & 1 & 0 \\ 5 & 1 & 0 & 0 & 5 & 0 & 0 \\ 6 & 0 & 1 & 5 & 0 & 4 & 1 \\ 7 & 0 & 1 & 0 & 4 & 0 & 1 \\ (8,9) & 0 & 0 & 0 & 1 & 1 & 0 \end{array}.$$

Поскольку применение 1° не приводит к образованию новых массивов, то выполняем 3° при $\alpha=3$. Определяем массив $H_4=\{x_5, x_6, x_7\}$. Заносим его в список $A=\{H_1, H_2, H_3, H_4\}$, факторизуем граф G_2 относительно массива H_4. Получаем граф G_3 с матрицей смежности

$$\mathbf{R}_3=\begin{array}{c|cccc} & (1,2,3) & 4 & (5,6,7) & (8,9) \\ (1,2,3) & 0 & 2 & 1 & 0 \\ 4 & 2 & 0 & 2 & 0 \\ (5,6,7) & 1 & 2 & 0 & 2 \\ (8,9) & 0 & 0 & 2 & 0 \end{array}.$$

Поскольку для графа G_3 не удается найти минимальные массивы по 2 и 3 вершины, то увеличиваем параметр $\alpha=3$ на 1. Получаем $\alpha=4$, что равняется порядку матрицы $\mathbf{R}_3$. Таким образом определяем конец работы алгоритма. В результате работы получаем список $A=\{H_1, H_2, H_3, H_4\}$, в котором минимальные массивы расположены по возрастанию числа входящих в них вершин исходного графа. Вершина x_4 исходного графа G образует самостоятельный минимальный массив.

На рис. 3.15 полученные минимальные массивы обведены штриховой линией. Следует заметить, что выделение минимальных массивов связано с большим перебором. Для его сокращения при разбиении графа на части необходимо выделять квазиминимальные массивы с последующим объединением их в массивы с максимальным числом внутренних ребер. Число вершин в каждой части разбиения

$$M \leqslant n_i \leqslant N, \tag{3.21}$$

где N — верхняя оценка разбиения, соответствующая полусумме общего числа вершин; M — нижняя оценка соответствующая наименьшему допустимому числу вершин графа, которые возможно поместить в рассматриваемую часть разбиения.

Методика выделения квазиминимальных массивов заключается в следующем. Выделяются массивы по 2, 3, 4 вершины до тех пор, пока выполняется условие (3.21). Квазиминимальные массивы определяются по матрице смежности $\mathbf{R}$ графа G. Сначала в

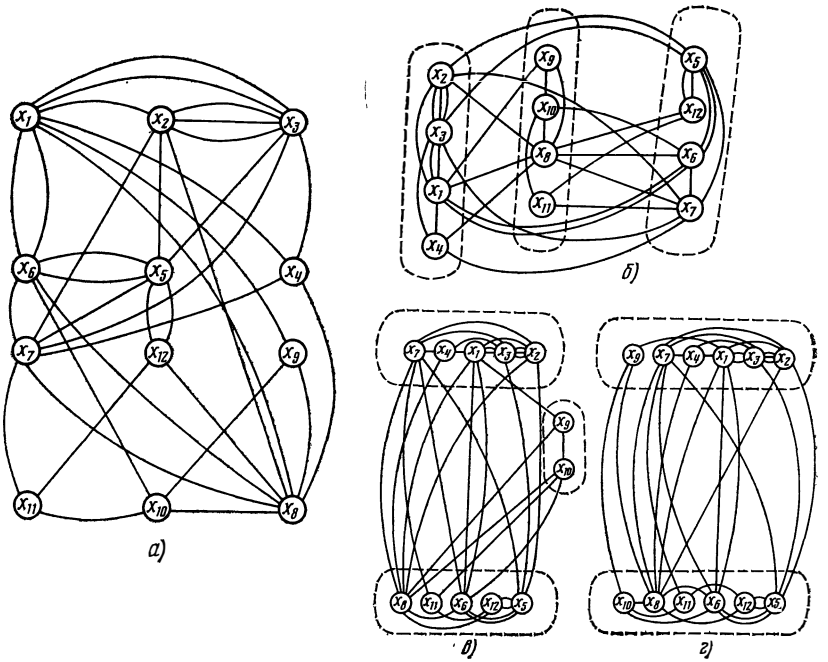


Рис. 3.16. Пример графа для выделения квазиминимальных массивов (*а*) и три варианта разбиения графа (*б*)

каждой строке матрицы отыскивается элемент $r_{i,j}$ с максимальным весом, если их несколько, то фиксируется их число. Кроме того, производится определение суммы элементов в каждой строке. Среди максимальных элементов каждой строки находится наибольший. В том случае, когда в строке имеется несколько одинаковых максимальных чисел, то предпочтение отдается строке, в которой таких чисел меньше. Если число максимальных элементов в строках матрицы смежности совпадает, то выбирается строка с наибольшей суммой элементов. После этого выбранные вершины группируются в квазиминимальный массив. Далее процесс повторяется, пока граф не разобьется на квазиминимальные массивы, содержащие по две вершины. При этом могут получаться единичные вершины. Аналогичным образом строятся массивы по 3, 4, 5, ... вершин.

Работу алгоритма покажем на примере графа G, изображенного на рис. 3.16,*а*, матрица смежности которого имеет вид

$\mathbf{R}=$

	1	2	3	4	5	6	7	8	9	10	11	12	A_1	A_2	A_3
1	0	1	2	1	0	2	0	1	1	0	0	0	2	2	8
2	1	0	3	0	1	0	1	1	0	0	0	0	1	3	7
3	2	3	0	1	1	0	1	0	0	0	0	0	1	3	8
4	1	0	1	0	0	0	1	1	0	0	0	0	4	1	4
5	0	1	1	0	0	2	1	0	0	0	0	2	2	2	7
6	2	0	0	0	2	0	1	1	0	1	0	0	2	2	7
7	0	1	1	1	1	1	0	1	0	0	1	0	7	1	7
8	1	1	0	1	0	1	1	0	1	1	0	1	8	1	8
9	1	0	0	0	0	0	0	1	0	1	0	0	3	1	3
10	0	0	0	0	0	1	0	1	1	0	1	0	4	1	4
11	0	0	0	0	0	0	1	0	0	1	0	1	3	1	3
12	0	0	0	0	2	0	0	1	0	0	1	0	1	2	4

Пусть число вершин в частях разбиения лежит в пределах $2 \leqslant n_i \leqslant 6$. Матрицу **R** дополним столбцами A_1, A_2 и A_3. В столбце A_2 записаны максимальные числа строк, в столбце A_3 — сумма элементов строк матрицы, а в столбце A_1 — количество максимальных чисел в строке. По столбцу A_2 определяем максимальные числа. Они находятся во второй и третьей строках матрицы **R**. Находим первый массив, состоящий из вершин x_2, x_3. Аналогично получаем следующие квазиминимальные массивы, состоящие из двух вершин: (x_5, x_{12}); (x_1, x_6); (x_9, x_{10}); (x_{11}, x_7); (x_8, x_4).

Строим матрицу

$\mathbf{R}'=$

	1	2	3	4	5	6	7	8	9	10	11	12	A'_1	A'_2	A'_3
(2, 3)	3	0	0	1	2	0	2	1	0	0	0	0	1	3	9
(5, 12)	0	1	1	0	0	2	1	1	0	0	1	0	1	2	7
(1, 6)	0	1	2	1	2	0	1	2	1	1	0	0	3	2	11
(9, 10)	1	0	0	0	0	1	0	2	0	0	1	0	1	2	5
(11, 7)	0	1	1	1	1	1	0	1	0	1	0	1	8	1	8
(8, 4)	2	1	1	0	0	1	2	0	1	1	0	1	2	2	10

Исследуя $\mathbf{R}'$, строим столбцы A'_1, A'_2, A'_3. По A'_2 выделяем строку (x_2, x_3) и в ней максимальный элемент с весом 3, лежащий в первом столбце. Складываем первый столбец со списком внутренних связей каждого массива. Для массива (x_2, x_3) полученная сумма максимальна. Следовательно, образуем новый массив (x_2, x_3, x_1).

Продолжая аналогично, выделяем следующие массивы по три вершины: (x_5, x_{12}, x_6); (x_9, x_{10}, x_8); (x_{11}, x_7, x_4). Далее процесс повторяется, и в результате получаем три варианта разбиения множества вершин графа (рис. 3.16,*б*, *в*, *г*): $X_1=\{x_2, x_3, x_1, x_4\}$, $X_2=\{x_9, x_{10}, x_8, x_{11}\}$, $X_3=\{x_5, x_{12}, x_6, x_7\}$; $X_1=\{x_2, x_3, x_1, x_4, x_7\}$, $X_2=\{x_9, x_{10}\}$, $X_3=\{x_5, x_{12}, x_6, x_{11}, x_8\}$; $X_1=\{x_2, x_3, x_1, x_4, x_7, x_9\}$, $X_2=\{x_5, x_{12}, x_6, x_{11}, x_8, x_{10}\}$.

Рассмотрим вопрос разбиения графа схемы на части методом выделения некоторой системы $\Phi=\{\varphi_i\}$, $i\in I=\{1, 2, \ldots, s\}$, попарно непересекающихся максимально полных подмножеств множества вершин X графа $G=(X, U)$. Для этого применим алгоритм построения клик по матрице смежности $\mathbf{R}$ графа, а для нахождения разбиения с минимизацией числа внешних ребер графа используем операцию раскраски его вершин.

Пусть задан граф $G=(X, U)$. Подграф $G_i=(X_i, U_i)$, $X_i\subseteq X$, $U_i\subseteq U$, в котором $\forall\ x_i, x_j\in X$ и существует ребро $u_k=(x_i, x_j)$, называется полным.

Подграф $G_i\subset G$, не являющийся частью никакого другого большего полного подграфа, называется максимально полным или *кликой*.

Список вершин графа и семейство $\Phi=\{\varphi_i\}$ определяют однозначно исходный граф. Для определения семейства Φ будем использовать матрицу смежности $\mathbf{R}=\|r_{i,j}\|_n$, где $n=|X|$. Сформулируем алгоритм построения семейства клик по матрице смежности. Заметим, что алгоритм выделения семейства клик аналогичен алгоритму выделения семейства внутренне устойчивых подмножеств множества вершин графа.

1°. В матрице смежности $\mathbf{R}$ графа G из i-й строки ($i\in I=\{1, 2, \ldots, n\}$) выбирается первый встречающийся по порядку элемент $r_{i,j}=1$ ($j\in I$, $i<j$), который определяет j-ю строку. Переход к 2°. Если в строке все элементы $r_{i,j}=0$, то вершина x_i определяет максимально полные подмножества. Обращаемся к строке $i+1$, считая $i=i+1$, и переход к 1°.

2°. Осуществляем поэлементную конъюнкцию i-й и j-й строк. Определяем $M_{i,j}=x_i\wedge x_j$. Переход к 3°.

3°. Из строки $M_{i,j}$ выбираем элемент $r_k=1$, $k\in I$. Переходим к 4°. Если все $r_k=0$, то переход к 5°.

4°. Осуществляем поэлементную конъюнкцию строк $M_{i,j}$ и x_k. Определяем $M_{i,j,k}=M_{i,j}\wedge x_k$. Заменяем индексы (i, j) на i, а k на j. Переход к 6°.

5°. Получено $\varphi_j=\{x_i, x_j, x_k\}$. Переход к 6°.

6°. Из строки x_i выбираем элемент $r_{i,l}=1$, $l\in I$, $x_l\in\varphi_\nu$. Переход к 7°. Если все $r_{i,l}=0$, то переход к 8°.

7°. Заменяем в выражении $M_{i,j}=x_i\wedge x_j$ индекс j на l. Переход к 2°.

8°. Заменяем индекс i на $i+1$. Переход к 2°. Если $i=n$, то переход к 9°.

9°. Выделено семейство $\Phi=\{\varphi_i\}$, $i\in$I.

Можно показать, что семейство $\Phi=\{\varphi_i\}$, $i\in I$, полученное по алгоритму, является семейством максимально полных подграфов.

Наибольший практический интерес представляет задача нахождения наименьшего числа попарно непересекающихся полных подграфов, объединение которых составляет данный граф. Эту задачу можно решить методом нахождения минимальной раскраски множества вершин графа. Известно, что раскрасить множество вершин графа $G=(X, U)$ значит поставить в соответствие каждой вершине графа G целое неотрицательное число $g(x)$ с соблюдением условия: вершины, отмеченные одинаковыми числами, должны быть несмежными. В нашем случае необходимо определить модифицированную раскраску, для которой необходимо соблюдение противоположного условия: вершины, отмеченные одинаковыми цветами, должны быть смежными. Модифицированная раскраска вершин равносильна выделению в графе G системы попарно непересекающихся полных подграфов. Нахождение минимальной раскраски является сложной комбинаторной задачей, поэтому в практике обычно не ставят целью получение минимальной раскраски, а, как правило, ведут поиск простой формальной процедуры, легко реализуемой на ЭВМ.

Рассмотрим алгоритм локальной модифицированной раскраски вершин графа, отвечающий поставленным требованиям, который основан на анализе системы максимально полных подграфов графа.

Введем критерий выбора подмножеств X для полной раскраски вершин

$$\alpha_{\gamma,\delta}=|\Phi_\gamma|+|\Phi_\delta|-|\Phi_\gamma\cap\Phi_\delta|,\ \gamma,\ \delta\in I,\ I=\{1, 2, ..., s\}. \qquad (3.22)$$

В соответствии с этим критерием множества вершин $(\Phi_\gamma\setminus\Phi_\delta)$ и и $(\Phi_\delta\setminus\Phi_\gamma)$, для которых $\alpha_{\gamma,\delta}=\max\alpha_{\gamma,\delta}$, принимают значение функции раскраски $g(\Phi_\gamma\setminus\Phi_\delta)$ и $g(\Phi_\delta\setminus\Phi_\gamma)$ соответственно.

Для удобства реализации алгоритма составим матрицу

$$A=\|\alpha_{\gamma,\delta}\|_m, \qquad (3.23)$$

где $m=|\Phi|$, а каждый элемент $\alpha_{\gamma,\delta}$, матрицы A говорит о максимальном числе вершин в $(\Phi_\gamma\cap\Phi_\delta)$. После выбора первых двух подмножеств строится новое семейство Φ' путем удаления из каждого $\varphi_i\in\Phi$, $i\in I$, вершин, вошедших в подмножества φ_γ и φ_δ, выбранные на предыдущих шагах. Сформулируем теперь алгоритм раскраски.

1°. По графу G определяется семейство максимально полных подграфов $\Phi=\{\varphi_i\}$, $i\in I$ Переход к 2°.

2°. Используя выражение (3.22), строим матрицу А. Переход к 3°.

3°. Из матрицы А выбирается максимальное значение $\alpha_{\gamma,\delta}$. Затем получаются подмножества φ_γ и φ_δ. Переход к 4°.

4°. В каждом $\varphi_i \in \Phi$ удаляются вершины, входящие в $\varphi_j \cup \varphi_\delta$, находится Φ' и переход к 5°, если $\Phi' = \varnothing$, то переход к 6°.

5°. Строится новая матрица **A** и переход к 3°.

6°. Получена раскраска вершин графа G.

Пример. Пусть задан граф G, изображенный на рис. 3.17,*а*. Матрица смежности **R** этого графа имеет вид

$$\mathbf{R} = \begin{array}{c|c} & \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \hline 1 & 0\,0\,0\,1\,0\,0\,1\,0\,1 \\ 2 & 0\,0\,1\,0\,0\,1\,0\,0\,0 \\ 3 & 0\,1\,0\,1\,0\,1\,0\,0\,0 \\ 4 & 1\,0\,1\,0\,0\,0\,1\,0\,0 \\ 5 & 0\,0\,0\,0\,0\,0\,0\,1\,1 \\ 6 & 0\,1\,1\,0\,0\,0\,0\,0\,0 \\ 7 & 1\,0\,0\,1\,0\,0\,0\,0\,0 \\ 8 & 0\,0\,0\,0\,1\,0\,0\,0\,1 \\ 9 & 1\,0\,0\,0\,1\,0\,0\,1\,0 \end{array}.$$

Выделим семейство максимально полных подграфов Φ. Для этого обратимся к x_i строке матрицы **R**. Выберем в ней первый ненулевой элемент $r_{1,4}$ и выполним поэлементную конъюнкцию первой и четвертой строк: $M_{1,4} = 0\,0\,0\,1\,0\,0\,1\,0\,1 \wedge 1\,0\,1\,0\,0\,0\,1\,0\,0 = 0\,0\,0\,0\,0\,0\,1\,0\,0$. Так как в объединенной строке $M_{1,4}$ есть ненулевой элемент, расположенный в 7-й позиции, то выполняем конъюнкцию $M_{1,4}$ и 7-й строки. В результате получим $M_{1,4,7} = 0\,0\,0\,0\,0\,0\,1\,0\,0 \wedge 1\,0\,0\,1\,0\,0\,0\,0\,0 = 0\,0\,0\,0\,0\,0\,0\,0\,0$. Образован первый элемент $\varphi_1 = \{x_1, x_4, x_7\}$, соответствующий полному подграфу графа G из семейства Φ.

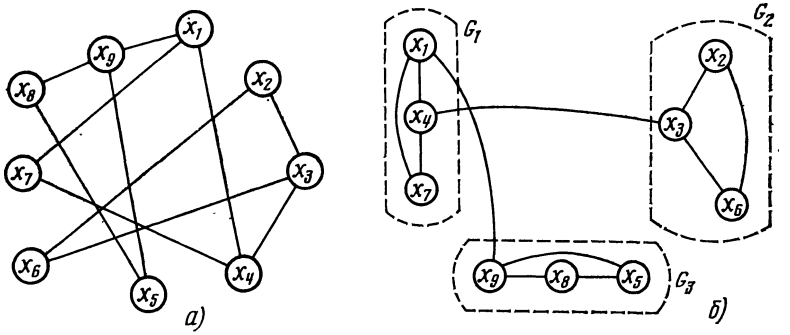


Рис. 3.17. Пример графа для разбиения на основе модифицированной раскраски (*а*), разбиение графа (*б*)

Далее снова обращаемся к строке 1 и проверяем, есть ли в ней ненулевые элементы, не вошедшие в φ_1. Такой элемент есть — это $r_{1,9}$. Обращаемся к 9-й строке матрицы **R** и определяем $M_{1,9} = 0\,0\,0\,1\,0\,0\,1\,0\,1 \wedge 1\,0\,0\,0\,1\,0\,0\,1\,0 = 0\,0\,0\,0\,0\,0\,0\,0\,0$. Получаем $\varphi_2 = \{x_1, x_9\}$.

Переходим ко 2-й строке матрицы смежности и выделяем элемент $r_{2,3}$. Осуществляем конъюнкцию строк и получаем $M_{2,3} = 0\,0\,1\,0\,0\,1\,0\,0\,0 \wedge 0\,1\,0\,1\,0\,1\,0\,0\,0 = 0\,0\,0\,0\,0\,1\,0\,0\,0$; $M_{2,3,6} = 0\,0\,0\,0\,0\,1\,0\,0\,0 \wedge 0\,1\,1\,0\,0\,0\,0\,0\,0 = 0\,0\,0\,0\,0\,0\,0\,0\,0$. Тогда $\varphi_3 = \{x_2, x_3, x_6\}$.

Продолжая аналогично, выделяем подмножества $\varphi_4=\{x_3, x_4\}$ и $\varphi_5=\{x_5, x_8, x_9\}$. На этом выделение семейства максимально полных подмножеств заканчивается $\Phi=\{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5\}$.

Для раскраски графа G построим матрицу

$$A=\begin{array}{c} \\ \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \varphi_4 \\ \varphi_5 \end{array}\begin{array}{c} \begin{array}{ccccc} \varphi_1 & \varphi_2 & \varphi_3 & \varphi_4 & \varphi_5 \end{array} \\ \left\|\begin{array}{ccccc} 3 & 4 & 6 & 4 & 6 \\ 4 & 2 & 5 & 4 & 4 \\ 6 & 5 & 3 & 4 & 6 \\ 4 & 4 & 4 & 2 & 5 \\ 6 & 4 & 6 & 5 & 3 \end{array}\right\| \end{array}.$$

Элементы матрицы **A** определяются из выражения (3.22). В матрице выбираем $\max \alpha_{\gamma,\delta}=\alpha_{1.3}$. Получаем два полных непересекающихся подграфа φ_1 и φ_3. Теперь, удаляя в каждом φ_i элементы, вошедшие в $\varphi_1 \cup \varphi_3$, построим новое семейство $\Phi'=\{\varphi'\}=\{x_5, x_8, x_9\}$. Так как все вершины графа распределены по полным подграфам, то получено разбиение графа на три части: $G_1=(X_1, U_1)$; $G_2=(X_2, U_2)$; $G_3=(X_3, U_3)$, где $X_1=\varphi_1=\{x_1, x_4, x_7\}$; $X_2=\varphi_3=\{x_2, x_3, x_6\}$; $X_3=\varphi'=\{x_5, x_8, x_9\}$.

Полученное разбиение графа на три части с минимизацией внешних связей показано на рис. 3.17,*б*; $K=2$; $\Delta(G)=9/2=4{,}5$.

Рассмотрим итерационные методы разбиения графов схем. Идея итерационных методов разбиения заключается в выборе некоторого разбиения графа с дальнейшими перестановками вершин или групп вершин из одной части в другую с целью минимизации числа соединительных ребер.

Сформулируем постановку задачи разбиения графа на части применительно к итерационным алгоритмам. Пусть задан граф $G=(X, U)$, $|X|=n$, $|U|=m$. Требуется найти разбиение его на части с максимизацией функции

$$m_0=\sum_{j=1}^{l}\sum_{i=1}^{|U_{j,j}|}\alpha^j{}_i, \qquad (3.24)$$

где m_0 — общее число ребер внутри всех частей графа G; l — число частей, на которые разбивается граф;

$$\alpha^j{}_i=\begin{cases}1, \text{ если } U_j \in U_{j,j};\\ 0 \text{ в противном случае;}\end{cases}$$

$U_{j,j}$ — множество внутренних ребер G_j.

Зададим стандартную матрицу $\mathbf{F}=\|f_{i,j}\|_n$, $i, j \in I=\{1, 2, \ldots, n\}$, в которой по главной диагонали расположено столько единичных подматриц, на сколько частей разбивается граф G. Порядок q-й единичной подматрицы определяется числом вершин, которое должно быть помещено в G_q.

В соответствии с разбиением матрицы **F** матрицу смежности **R** графа G разобъем на подматрицы, что отвечает некоторому определенному разбиению графа G на части.

Например, пусть задан граф $G=(X, U)$, изображенный на рис. 3.18,*а*, который необходимо разбить на три части по 3, 3 и 2 вершины в каждой. Матрица для этого графа

$$\mathbf{F}=\begin{array}{c|cc|ccc|ccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
3 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
4 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
5 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
\hline
6 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
7 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
8 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{array}.$$

Для простоты будем разбивать матрицу **F** на клетки, начиная с тех, которые имеют наименьшее число элементов. Запишем теперь матрицу смежности **R** графа G и разобьем ее на подматрицы в соответствии с разбиением матрицы **F**:

$$\mathbf{R}=\begin{array}{c|cc|ccc|ccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
\hline
3 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
4 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
6 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
8 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0
\end{array}.$$

Указанное разбиение графа показано на рис. 3.18,*б*. При этом $G=\bigcup\limits_{i=1}^{3} G_i$; $G_1=(X_1, U_1)$; $G_2=(X_2, U_2)$; $G_3=(X_3, U_3)$; $X_1=\{x_1, x_2\}$; $X_2=\{x_3, x_4, x_5\}$; $X_3=\{x_6, x_7, x_8\}$; $U_1=U_{1,1}\cup U_{1,2}\cup U_{1,3}=\{(x_1, x_5), (x_2, x_4), (x_2, x_6)\}$; $U_2=U_{2,2}\cup U_{2,1}\cup U_{2,3}=\{(x_3, x_4), (x_1, x_5), (x_2, x_4), (x_3, x_8), (x_3, x_7), (x_5, x_8), (x_4, x_6)\}$; $U_3=U_{3,3}\cup U_{3,1}\cup U_{3,2}=\{(x_6, x_2), (x_3, x_7), (x_8, x_3), (x_8, x_5), (x_7, x_8), (x_6, x_4)\}$. Общее число внутренних ребер $m_0=2$, а количество внешних связей $K=7$.

Рассмотрим теперь методику перестановки строк и столбцов матрицы **R** для увеличения m_0. Для реализации итерационного метода разбиения графа G введем некоторые вспомогательные матрицы и определим действия над ними. Определим матрицу $\mathbf{M}=\|m_{i,j}\|_n$, $i, j\in I=\{1, 2, \ldots, n\}$ как результат умножения матриц **F** и **R**:

$$\mathbf{M}=\mathbf{F}\otimes\mathbf{R}, \tag{3.25}$$

тогда элементы матрицы **M** можно вычислить по следующим формулам:

$$m_{i,j}=\sum_{q=1}^{n} f_{i,q}r_{j,q};\ m_{j,i}=\sum_{q=1}^{n} f_{j,q}r_{i,q}. \tag{3.26}$$

Заметим, что матрица **M** не обязательно симметричная, т. е. в общем случае $m_{i,j}\neq m_{j,i}$, если $i\neq j$.

Для графа G рис. 3.18,*а*, используя матрицы **R** и **F**, построим матрицу

$$\mathbf{M}=\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 3 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 2 \\ 4 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 2 \\ 5 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 2 \\ 6 & 0 & 1 & 2 & 1 & 1 & 0 & 1 & 1 \\ 7 & 0 & 1 & 2 & 1 & 1 & 0 & 1 & 1 \\ 8 & 0 & 1 & 2 & 1 & 1 & 0 & 1 & 1 \end{array}.$$

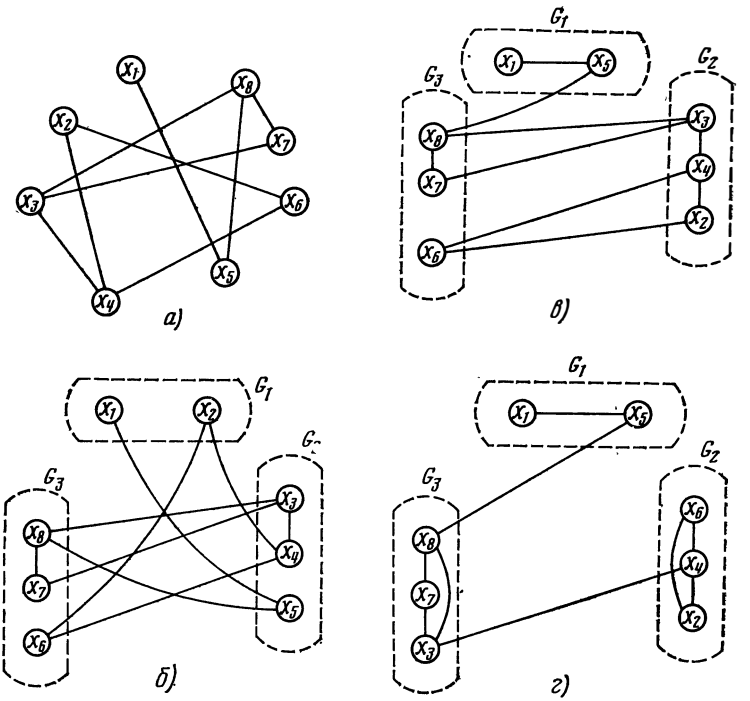


Рис. 3.18. Пример графа для итерационного метода (*а*) и его разбиения: первоначальное (*б*), после перестановки вершин x_2, x_5 (*в*), окончательное (*г*)

Используя выражения (3.26), определяем значения $m_{i,j}$ и $m_{j,i}$:

$$m_{1,1}=\sum_{q=1}^{n} f_{1,q}r_{1,q}=f_{1,1}r_{1,1}+f_{1,2}r_{1,2}+f_{1,3}r_{1,3}+f_{1,4}r_{1,4}+f_{1,5}r_{1,5}+f_{1,6}r_{1,6}+f_{1,7}r_{1,7}+f_{1,8}r_{1,8}=$$
$$=1\cdot 0+1\cdot 0+0\cdot 0+0\cdot 0+0\cdot 1+0\cdot 0+0\cdot 0+0\cdot 0=0;$$

$$m_{1,2}=\sum_{q=1}^{n} f_{1,q}r_{2,q}=f_{1,1}r_{2,1}+f_{1,2}r_{2,2}+f_{1,3}r_{2,3}+f_{1,4}r_{2,4}+f_{1,5}r_{2,5}+f_{1,6}r_{2,6}+f_{1,7}r_{2,7}+f_{1,8}r_{2,8}=$$
$$=1\cdot 0+1\cdot 0+0\cdot 0+0\cdot 1+0\cdot 0+0\cdot 1+0\cdot 0+0\cdot 0=0.$$

Продолжая аналогично, определяем, например, элемент

$$m_{7,3}=\sum_{q=1}^{n} f_{7,q}r_{3,q}=f_{7,1}r_{3,1}+f_{7,2}r_{3,2}+f_{7,3}r_{3,3}+f_{7,4}r_{3,4}+f_{7,5}r_{3,5}+f_{7,6}r_{3,6}+f_{7,7}r_{3,7}+f_{7,8}r_{3,8}=$$

$$=0\cdot 0+0\cdot 0+0\cdot 0+0\cdot 1+0\cdot 0+1\cdot 0+1\cdot 1+1\cdot 1=2.$$

Процесс выполняется до тех пор, пока не будут найдены все элементы матрицы **M**.

Введем матрицу $\mathbf{V}=\|v_{i,j}\|_n$, $i,\ j\in I=\{1, 2, ..., n\}$. Матрица **V** строится путем поэлементного перемножения матриц $\overline{\mathbf{F}}$ и **R**:

$$\mathbf{V}=\overline{\mathbf{F}}\times\mathbf{R}, \tag{3.27}$$

где $\overline{\mathbf{F}}$ — инверсия матрицы **F**.

Для рассматриваемого примера

$\overline{\mathbf{F}}=$

	1	2	3	4	5	6	7	8
1	0	0	1	1	1	1	1	1
2	0	0	1	1	1	1	1	1
3	1	1	0	0	0	1	1	1
4	1	1	0	0	0	1	1	1
5	1	1	0	0	0	1	1	1
6	1	1	1	1	1	0	0	0
7	1	1	1	1	1	0	0	0
8	1	1	1	1	1	0	0	0

Элемент $v_{i,j}$ матрицы **V** определяется по формуле $v_{i,j}=\bar{f}_{i,j}r_{i,j}$, где $\bar{f}_{i,j}$ — элемент матрицы $\overline{\mathbf{F}}$.

Отметим, что матрица **V** является симметричной относительно главной диагонали, т. е. $v_{i,j}=v_{j,i}$.

Используя матрицы $\overline{\mathbf{F}}$ и **R** графа G на рис. 3.18,*а*, по формуле (3.27) построим матрицу

$\mathbf{V}=$

	1	2	3	4	5	6	7	8
1	0	0	0	0	1	0	0	0
2	0	0	0	1	0	1	0	0
3	0	0	0	0	0	0	1	1
4	0	1	0	0	0	1	0	0
5	1	0	0	0	0	0	0	1
6	0	1	0	1	0	0	0	0
7	0	0	1	0	0	0	0	0
8	0	0	1	0	1	0	0	0

Элементы первой строки матрицы **V** равны: $v_{1,1}=r_{1,1}\bar{f}_{1,1}=0$; $v_{1,2}=r_{1,2}\bar{f}_{1,2}=0$; $v_{1,3}=r_{1,3}\bar{f}_{1,3}=0$; $v_{1,4}=r_{1,4}\bar{f}_{1,4}=0$; $v_{1,5}=r_{1,5}\bar{f}_{1,5}=1$; $v_{1,6}=r_{1,6}\bar{f}_{1,6}=0$; $v_{1,7}=r_{1,7}\bar{f}_{1,7}=0$; $v_{1,8}=r_{1,8}\bar{f}_{1,8}=0$. Все остальные матрицы **V** определяются аналогично.

С помощью введенных матриц **M** и **V** построим матрицу $\mathbf{P}=\|p_{i,j}\|_n$, $i,\ j\in I=\{1, 2, ..., n\}$.

Матрица **P** определяется с помощью матриц **M** и **V** следующим образом:

$$\mathbf{P}=\mathbf{M}-\mathbf{V}. \tag{3.28}$$

В выражении (3.28) используется поэлементное вычитание. Если вместо **M** и **V** подставить выражения (3.25) и (3.27), то получим

$$\mathbf{P}=(\mathbf{F}\otimes\mathbf{R})-(\bar{\mathbf{F}}\times\mathbf{R}). \tag{3.29}$$

Из способа построения следует, что матрица **P** несимметричная, $p_{i,j}\neq p_{j,i}$, поэтому определим ее элементы, расположенные по разные стороны главной диагонали:

$$p_{i,\,j}=m_{i,\,j}-v_{i,\,j}=\Big(\sum_{q=1}^{n} f_{i,\,q}r_{j,\,q}\Big)-\bar{f}_{i,\,j}r_{i,\,j};$$

$$p_{j,\,i}=m_{j,\,i}-v_{i,\,j}=\Big(\sum_{q=1}^{n} f_{j,\,q}r_{i,\,q}\Big)-\bar{f}_{i,\,j}r_{i,\,j}. \tag{3.30}$$

По формулам (3.30) построим матрицу **P** для графа G (рис. 3.18, *б*):

$$\mathbf{P}=\begin{array}{c|cccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
3 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\
4 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 2 \\
5 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
6 & 0 & 0 & 2 & 0 & 1 & 0 & 1 & 1 \\
7 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
8 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1
\end{array}.$$

Покажем, например, как вычисляются элементы четвертой строки: $p_{4,\,1}=m_{4,\,1}-v_{4,\,1}=1-0=1$; $p_{4,\,2}=m_{4,\,2}-v_{4,\,2}=1-1=0$, $p_{4,\,3}=m_{4,\,3}-v_{4,\,3}=1$; $p_{4,\,4}=m_{4,\,4}-v_{4,\,4}=1$; $p_{4,\,5}=m_{4,\,5}-v_{4,\,5}=0$; $p_{4,\,6}=m_{4,\,6}-v_{4,\,6}=0$; $p_{4,\,7}=m_{4,\,7}-v_{4,\,7}=1$; $p_{4,\,8}=m_{4,\,8}-v_{4,\,8}=2$.

Остальные элементы матрицы **P** определяются аналогично.

Для максимизации функции (3.24) после разбиения необходимо выбрать в графе пару вершин, принадлежащих разным частям в разбиении (пару строк и столбцов в матрице **R**, принадлежащих разным подматрицам), и переставить их, если значение m_0 увеличится. Для переразмещения вершин x_i и x_j между частями разбиения (перестановки строк и столбцов матрицы **R**) введем понятие *перестановочного коэффициента.*

$$\eta_{i,\,j}=p_{i,\,j}+p_{j,\,i}-p_{i,\,i}-p_{j,\,j}. \tag{3.31}$$

Заметим, что вершина x_i лежит в $G_i=(X_i, U_i)$, а вершина x_j лежит в $G_j=(X_jU_j)$. С помощью выражений (3.29) и (3.30) определим коэффициент η_{ij}:

$$\eta_{i,j}=[(\sum_{q=1}^{n} f_{i,q}r_{j,q})-\bar{f}_{i,j}r_{i,j}] + [(\sum_{q=1}^{n} f_{j,q}r_{i,q})-\bar{f}_{i,j}r_{i,j}]-[(\sum_{q=1}^{n} f_{i,q}r_{i,q})-$$

$$-r_{i,i}\bar{f}_{i,i}]-[(\sum_{q=1}^{n} f_{j,q}r_{j,q})-f_{j,j}r_{j,j}].$$

Заметим, что рассматриваются графы без петель, поэтому элементы матрицы **R**, расположенные по главной диагонали, равны нулю, т. е. $r_{i,i}=r_{j,j}=0$. Так как $\eta_{i,j}$ имеет смысл находить только для вершин, лежащих в разных частях, то в этом случае $\bar{f}_{i,j}=1$, $r_{i,j}\bar{f}_{i,j}=r_{i,j}$. Тогда перестановочный коэффициент примет вид

$$\eta_{i,j}=\sum_{q=1}^{n} f_{i,q}r_{j,q}+\sum_{q=1}^{n} f_{j,q}r_{i,q}-\sum_{q=1}^{n} f_{i,q}r_{i,q}-\sum_{q=1}^{n} f_{j,q}r_{j,q}-2r_{i,j}. \quad (3.32)$$

Максимальное значение $\eta_{i,j}>0$ является условием для перестановки вершин из одной части в другую с целью максимизации функции (3.24). Коэффициент $\eta_{i,j}$ определяет, как изменится число внешних ребер между G_i и G_j после перестановки вершины x_i в G_j, а вершины x_j в G_j.

Разность

$$\Delta m_{i,j}=(m_{i,j}+m_{j,i})-(m_{i,i}+m_{j,j}) \quad (3.33)$$

указывает на изменение числа связей между соответствующими подграфами при перестановке вершин x_i и x_j из одного в другой. Заметим, что если вершины x_i, x_j смежны, то в выражение (3.33) необходимо ввести поправку на величину $\xi_{i,j}$, равную числу ребер между x_i и x_j. Так как после перестановки вершин указанные ребра останутся в общем числе внешних связей, то их число вычитается из Δm_{ij}. Тогда получим

$$\eta_{i,j}=\Delta m_{i,j}-2\xi_{i,j}. \quad (3.34)$$

Нетрудно заметить, что числа $\xi_{i,j}$ представляют собой элементы $v_{i,j}$, которые заключены в матрице **V**. Получив матрицу **P**=**M**—**V** и вычислив по формуле (3.34) коэффициент $\eta_{i,j}$, определим действительное изменение числа связей в разбиении после перестановки вершин $x_i\in X_i$ и $x_j\in X_j$. Следовательно, если $\eta_{i,j}<0$, то число ребер между $G_i=(X_i, U_i)$ и $G_j=(X_j, U_j)$ после перестановки вершин $x_i\in X_i$ и $x_j\in X_j$ увеличивается, если $\eta_{i,j}=0$ — не изменяется, $\eta_{i,j}>0$ уменьшается. Так как критерием данного итерационного алгоритма разбиения является максимальное число ребер внутри частей, то перестановку вершин между частями необходимо выполнять при $\eta_{i,j}>0$.

Заметим, что для нахождения перестановочного коэффициента $\eta_{i,j}$ можно использовать как выражение (3.31), так и (3.32).

Для рассмотренного графа G, изображенного на рис. 3.18, *а*, используя выражение (3.31), определяем перестановочные коэффициенты $\eta_{1,3}=p_{1,3}+p_{3,1}-p_{1,1}-p_{3,3}=0+1-0-1=0$; $\eta_{1,4}=1$; $\eta_{1,5}=0$; $\eta_{1,6}=1$; $\eta_{1,7}=-1$; $\eta_{1,8}=-1$; $\eta_{2,3}=0$; $\eta_{2,4}=-1$; $\eta_{2,5}=2$; $\eta_{2,6}=0$; $\eta_{2,7}=0$; $\eta_{2,8}=0$.

Выберем максимальный положительный коэффициент $\eta_{2,5}=2$. После перестановки вершин x_2 и x_5 из одной части в другую, что показано на рис. 3.18,*в*, получим число внешних ребер $K=5$. Общее число внутренних ребер $r=4$.

После перестановки второй и пятой строк и столбцов матрица **R** графа на рис. 3.18,*а* примет вид (отметим, что матрица **F** остается неизменной на всех шагах алгоритма)

$$\mathbf{R}'=\begin{array}{c|cccccccc} & 1 & 5 & 3 & 4 & 2 & 6 & 7 & 8 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 4 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 6 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 7 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 8 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}.$$

Снова определим перестановочные коэффициенты для вершин x_1 и x_5 из первой части разбиения графа G по формуле (3.32):

$$\eta_{1,3}=\left(\sum_{q=1}^{8} f_{1,q}\,r_{3,q}\right)+\left(\sum_{q=1}^{8} f_{3,q}\,r_{1,q}\right)-\left(\sum_{q=1}^{8} f_{1,q}\,r_{1,q}\right)-$$

$$-\left(\sum_{q=1}^{3} f_{3,q}\,r_{3,q}\right)-2r_{1,3}=(f_{1,1}\,r_{3,1}+f_{1,2}\,r_{3,2}+f_{1,3}\,r_{3,3}+f_{1,4}\,r_{3,4}+$$

$$+f_{1,5}\,r_{3,5}+f_{1,6}\,r_{3,6}+f_{1,7}\,r_{3,7}+f_{1,8}\,r_{3,8})+(f_{3,1}\,r_{1,1}+f_{3,2}\,r_{1,2}+$$

$$+f_{3,3}\,r_{1,3}+f_{3,4}\,r_{1,4}+f_{3,5}\,r_{1,5}+f_{3,6}\,r_{1,6}+f_{3,7}\,r_{1,7}+f_{3,8}\,r_{1,8})-$$

$$-(f_{1,1}\,r_{1,1}+f_{1,2}\,r_{1,2}+f_{1,3}\,r_{1,3}+f_{1,4}\,r_{1,4}+f_{1,5}\,r_{1,5}+f_{1,6}\,r_{1,6}+$$

$$+f_{1,7}\,r_{1,7}+f_{1,8}\,r_{1,8})-(f_{3,1}\,r_{3,1}+f_{3,2}\,r_{3,2}+f_{3,3}\,r_{3,3}+f_{3,4}\,r_{3,4}+$$

$$+f_{3,5}\,r_{3,5}+f_{3,6}\,r_{3,6}+f_{3,7}\,r_{3,7}+f_{3,8}\,r_{3,8})-2\,r_{1,3}=-2;$$

$$\eta_{1,4}=-3;\quad \eta_{1,2}=-2;\quad \eta_{1,6}=-1;\quad \eta_{1,7}=-2;\quad \eta_{1,8}=-1;$$

$$\eta_{5,3}=-2;\quad \eta_{5,4}=-3;\quad \eta_{5,6}=0;\quad \eta_{5,7}=-1;\quad \eta_{5,8}=-1;\quad \eta_{5,2}=-2.$$

Так как все коэффициенты отрицательны, то переставлять вершины из G_1 не имеет смысла. Определим перестановочные коэффициенты для возможного переразмещения вершин между G_2 и G_3 исследуемого графа:

$$\eta_{3,6}=\sum_{q=1}^{8} f_{3,q}r_{6,q}+\sum_{q=1}^{8} f_{6,q}r_{3,q}-\sum_{q=1}^{8} f_{3,q}r_{3,q}-\sum_{q=1}^{8} f_{6,q}r_{6,q}-2r_{3,6}=3;\quad \eta_{3,7}=$$

$$=-1;\ \eta_{3,8}=-1;\ \eta_{4,6}=-1;\ \eta_{4,7}=-1;\ \eta_{4,8}=-1;\quad \eta_{2,6}=0;\quad \eta_{2,7}=0;$$

$$\eta_{2,8}=0.$$

Выбираем максимальный положительный коэффициент $\eta_{3,6}=3$ и меняем местами вершины x_3 и x_6. Получим, что число ребер в сече-

нии графа $K=2$. Общее число внутренних ребер $m_0=7$. Разбиение графа после применения указанной операции изображено на рис. 3.18,*г*. Матрица смежности $\mathbf{R}'$ после перестановки третьей и шестой строк и столбцов примет вид

$$\mathbf{R}''=\begin{array}{c|cc|ccc|ccc} & 1 & 5 & 6 & 4 & 2 & 3 & 7 & 8 \\ \hline 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 6 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 3 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 7 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 8 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}.$$

Снова определим перестановочные коэффициенты по формуле (3.34) для вершин второй и третьей частей: $\eta_{6,3}=-3$; $\eta_{6,7}=-4$; $\eta_{6,8}=-4$; $\eta_{4,3}=-4$; $\eta_{4,7}=-3$; $\eta_{4,8}=-3$; $\eta_{2,3}=-3$; $\eta_{2,7}=-4$; $\eta_{2,8}=-4$.

Так как все коэффициенты $\eta_{i,j}$ отрицательны, то дальнейшая перестановка вершин нецелесообразна. На этом итерационный алгоритм разбиения графа с минимизацией числа соединительных ребер заканчивается. Таким образом, изображение на рис. 3.18,*г* представляет собой окончательное разбиение графа G (см. рис. 3.18,*а*).

Заметим, что методика разбиения рассмотрена для графов с однократными ребрами. Предложенный алгоритм в равной степени применим и для мультиграфов. В этом случае в формуле (3.32) за значение $r_{i,j}$ принимается сумма ребер, соединяющих вершины x_i и x_j мультиграфа.

Рассмотрим теперь методику дихотомического разбиения графа схемы, заданного матрицей смежности, на части с использованием чисел связности. Для каждой вершины графа введем числовую характеристику, использующую локальную степень этой вершины. Эта характеристика должна количественно оценивать связь рассматриваемой вершины с другими, лежащими внутри части разбиения, по отношению к вершинам, находящимся вне этой части. Назовем $\alpha(x_k)$ *числом связности* вершины x_k, причем

$$\alpha(x_k)=\begin{cases} r_k(G_i)-r_k(G_j), & \text{если } x_k\in X_j; \\ r_k(G_j)-r_k(G_j), & \text{если } x_k\in X_i, \end{cases}$$

где $r_k(G_i)$ — число ребер, соединяющих вершину x_k с вершинами $G_i=(X_i, U_i)$; $r_k(G_j)$ — число ребер, соединяющих вершину x_k с вершинами $G_j=(X_j, U_i)$. Поясним понятие числа связности на примере графа G, разбитого на две части G_1 и G_2 (рис. 3.19). Тогда числа связности для вершин G_i определятся следующим об-

разом: $\alpha(x_1)=r_1(G_2)-r_1(G_1)=1-2=-1$; $\alpha(x_2)=0$; $\alpha(x_3)=+1$. Для вершин G_2 числа связности определятся: $\alpha(x_4)=r_4(G_1)-r_4(G_2)=1-3=-2$; $\alpha(x_5)=0$; $\alpha(x_6)=-2$; $\alpha(x_7)=2$.

Очевидно, что число связности может быть отрицательным, положительным и равным нулю ($\alpha(x_k) \lesseqgtr 0$). Физический смысл числа связности следующий. Например, $\alpha(x_1)=-1$ означает, что при перестановке вершины x_1, лежащей в G_1, в G_2 число ребер в сечении увеличится на единицу. Значение $\alpha(x_2)=0$ говорит о том, что перестановка вершины x_2 из G_1 в G_2 оставит без изменения число соединительных ребер [1].

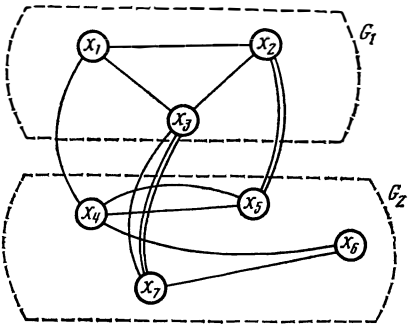


Рис. 3.19. Граф G после разбиения на две части: G_1 и G_2

Теорема 3.4. Перестановка двух произвольных вершин $x_k \in X_i$ и $x_l \in X_j$ ($X_i \cap X_j = \varnothing$) приводит к уменьшению числа соединительных ребер в сечении если:

а) вершина x_k не смежна вершине x_l и выполняется неравенство

$$\alpha_k+\alpha_l>0; \tag{3.35}$$

б) вершина x_k смежна вершине x_l и справедливо неравенство

$$\alpha_l+\alpha_k>2. \tag{3.36}$$

Доказательство. Очевидно, что перестановка вершин из одной части в другую целесообразна лишь в том случае, если

$$\Delta k_{i,j}=k_{i,j}-k'_{i,j}>0, \tag{3.37}$$

где $k'_{i,j}$ — число реберного соединения G_i и G_j после перестановки вершин x_k и x_l; $\Delta k_{i,j}$ — приращение в результате перестановки вершин x_k и x_l.

Рассмотрим теперь возможные случаи смежности вершин, указанные в теореме:

а) вершины x_k и x_l не смежны. Тогда до перестановки $k_{i,j}=r_k(G_i)+r_l(G_j)$, а после перестановки $k'_{i,j}=r_k(G_j)+r_l(G_i)$.

Подставляя значения $k_{i,j}$ и $k'_{i,j}$ в выражение (3.38), получаем $\Delta k_{i,j}=r_k(G_i)+r_l(G_j)-r_k(G_j)-r_l(G_i)=\alpha_k+\alpha_l>0$.

б) вершины x_k и x_l смежны. В этом случае необходимо учитывать появление нового ребра, соединяющего G_i и G_j, инцидентного вершинам x_k, x_l. Тогда $k_{i,j}=r_k(G_i)+r_l(G_j)-1$. Единица вычитается, так как при подсчете числа соединяющих ребер G_i, G_j ребро $u_{k,l}$ учитывалось дважды. После перестановки вершин x_k, x_l получим $k'_{i,j}=r_k(G_j)+r_l(G_i)+1$. Единица прибавляется, поскольку ребро $u_{k,l}$ остается соединяющим после перестановки x_k и x_l. Теперь $\Delta k_{i,j}=r_k(G_i)+r_l(G_j)-1-r_k(G_i)-r_l(G_i)-1>0$ или $\alpha_k+\alpha_l-2>0$*), что и требовалось доказать.

---

[1] Будем использовать также обозначение α_k вместо $\alpha(x_k)$.

*) В общем случае для мультиграфов неравенство принимает вид $\alpha_l+\alpha_k>2r_{l,k}$.

Приведем матрицу смежности графа, показанного на рис. 3.19:

$$\mathbf{R}=\begin{array}{c|ccc|cccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
2 & 1 & 0 & 1 & 0 & 2 & 0 & 0 \\
3 & 1 & 1 & 0 & 0 & 0 & 0 & 3 \\
\hline
4 & 1 & 0 & 0 & 0 & 2 & 1 & 0 \\
5 & 0 & 2 & 0 & 2 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
7 & 0 & 0 & 3 & 0 & 0 & 1 & 0
\end{array}.$$

Рассмотрим первую строку этой матрицы $\|0111000\|$. Первые три элемента матрицы $r_{1,1}=0$, $r_{1,2}=1$ и $r_{1,3}=1$ соответствуют внутренним ребрам (x_1, x_1), (x_1, x_2), (x_1, x_3) части G_1. Элементы $r_{1,4}$, $r_{1,5}$, $r_{1,6}$ и $r_{1,7}$ соответствуют соединяющим ребрам вершины x_1. Тогда $\alpha(x_1)$ по матрице смежности можно определить следующим образом: $\alpha(x_1)=(r_{1,4}+r_{1,5}+r_{1,6}+r_{1,7})-(r_{1,1}+r_{1,2}+r_{1,3})=1+0+0+0-0-1-1=-1$.

В общем виде для разбиения матрицы $\mathbf{R}=\|r_{i,j}\|_n$ на две подматрицы $\mathbf{R}_1$ и $\mathbf{R}_2$, где i, $j\in\mathcal{J}=\{1, 2, \ldots, n\}$, число связности

$$\alpha(x_k)=\begin{cases}\sum\limits_{j\in F} r_{k,j}-\sum\limits_{l\in I} r_{k,l}; \\ \sum\limits_{l\in I} r_{k,l}-\sum\limits_{j\in F} r_{i,j},\end{cases} \tag{3.38}$$

где $l\in I=\{1, 2, \ldots, p\}$, $j\in F=\{p+1, p+2, \ldots, n\}$, причем строка x_j принадлежит подматрице $\mathbf{R}_2$, а строка x_l — подматрице $\mathbf{R}_1$, $k\in\mathcal{J}=\{1, 2, \ldots, n\}$, p — старший индекс подматрицы $\mathbf{R}_1$.

Используя формулу (3.38), подсчитаем значения чисел связности для каждой строки матрицы и, введя дополнительный столбец к матрице $\mathbf{R}$, запишем их:

$$\mathbf{R}=\begin{array}{c|ccc|cccc|c}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \alpha_{k,q} \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & -1 \\
2 & 1 & 0 & 1 & 0 & 2 & 0 & 0 & 0 \\
3 & 1 & 1 & 0 & 0 & 0 & 0 & 3 & 1 \\
\hline
4 & 1 & 0 & 0 & 0 & 2 & 1 & 0 & -2 \\
5 & 0 & 2 & 0 & 2 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -2 \\
7 & 0 & 0 & 3 & 0 & 0 & 1 & 0 & 2
\end{array}.$$

Рассмотрим основную идею итерационного алгоритма разбиения графа G, заданного матрицей смежности, на части с минимизацией числа соединительных ребер.

Разбиение графа $G=(X, U)$ на l частей $G_1=(X_1, U_1)$, $G_2=(X_2, U_2), \ldots, G_l=(X_l, U_l)$ сведем к последовательному разбиению на две части. С этой целью в матрице $\mathbf{R}$ будем выделять по главной диагонали две подматрицы: $\mathbf{R}_1$ и $\mathbf{R}_2$. При этом порядок подмат-

рицы $\mathbf{R}_1$ приравняем к числу вершин, которые должны находиться в первой части, а порядок подматрицы $\mathbf{R}_2$ — к числу всех остальных вершин графа. Необходимо так переставить строки и столбцы матрицы $\mathbf{R}$, чтобы число ребер между G_1 и G_2 было минимальным. После этого подматрицу $\mathbf{R}_1$ из матрицы $\mathbf{R}$ исключим путем вычеркивания из $\mathbf{R}$ строк и столбцов соответствующих элементов. Далее подматрицу $\mathbf{R}_2$ разобьем снова на две подматрицы $\mathbf{R}^1{}_2$ и $\mathbf{R}^2{}_2$ так, чтобы порядок $\mathbf{R}^1{}_2$ соответствовал числу вершин второй выделяемой части, а порядок $\mathbf{R}^2{}_2$ — числу оставшихся вершин графа. Произведем перестановку строк и столбцов $\mathbf{R}_2$ для минимизации числа соединительных ребер, после чего подматрицу $\mathbf{R}^1{}_2$ исключим из $\mathbf{R}_2$ и процесс повторим аналогично до тех пор, пока не будет выполнено разбиение графа G на l частей.

Итерационная часть алгоритма заключается в выборе по заданному правилу определенных строк и столбцов матрицы $\mathbf{R}$ и перестановке их с целью минимизации числа соединительных ребер, что соответствует сосредоточению в диагональных подматрицах матрицы $\mathbf{R}$ максимального числа элементов. С этой целью построим прямоугольную матрицу $\mathbf{B}=\|b_{i,j}\|_{n_l\times(n-n_l)}$, в которой строки определяются вершинами из множества I, а столбцы — из множества F, $i\in I=\{1, 2, \ldots, l-1\}$. На пересечении k-й строки ($k\in I$) и q-го столбца ($q\in F$) находим

$$b_{k,\,q}=\alpha_k+\alpha_q-2r_{k,\,q}, \tag{3.39}$$

где $r_{k,q}$ — элемент матрицы смежности $\mathbf{R}$.

Элемент $b_{k,q}$ матрицы $\mathbf{B}$ характеризует изменение числа соединительных ребер между частями G_i и G_j при перестановке вершин $x_q\in X_i$ и $x_k\in X_j$. Используя матрицу $\mathbf{B}$, можно найти подстановку, которая увеличит число элементов в подматрицах $\mathbf{R}_1$ и $\mathbf{R}_2$. Дальнейшие процедуры выполняются аналогично до тех пор, пока в подматрице $\mathbf{R}_1$ не будет сосредоточено максимальное число единиц, т. е. не будет получен локальный минимум.

Заметим, что при работе алгоритма число итератий, время решения и оптимальность результата в значительной степени зависят от того, насколько удачно выбрано начальное разбиение матрицы смежности графа схемы. Для устранения этого недостатка можно применять алгоритм несколько раз для различных начальных условий.

Сформулируем итерационный алгоритм разбиения графа G на части, выполняемый по матрице смежности, в котором используется принцип попарных перестановок вершин из одной части в другую.

1°. По графу G строим матрицу смежности $\mathbf{R}_0$ порядка n. Переход к 2°.

2°. В матрице $\mathbf{R}_0$ графа G выделяем подматрицу порядка, равного числу элементов n_1 в G_1. Матрица $\mathbf{R}_0$ при этом разбивается на две подматрицы: $\mathbf{R}_{0,1}$ и $\mathbf{R}_{0,2}$. Переход к 3°.

3°. По матрице $\mathbf{R}_0$ находим числа связности α_k и α_q и строим матрицу $\mathbf{B}_0$. Переход к 4°.

4°. Используя вспомогательную матрицу $\mathbf{B}_0$, определяем максимальный положительный элемент $b_{k,q}$. Если таких элементов несколько, то выбираем такой, соответствующие которому вершины x_k и x_q имеют меньшую локальную степень. Осуществляем перестановку строк и столбцов k и q в матрице $\mathbf{R}_0$. Получаем матрицу $\mathbf{R}^1{}_0$, изоморфную матрице $\mathbf{R}_0$. Переход к 5°. Если в матрице $\mathbf{B}_0$ нет положительных элементов, то переход к 6°.

5°. Выполняем последовательно пункты 3°, 4° алгоритма для матрицы $\mathbf{R}_{0,1}$. Аналогичные преобразования выполняем с матрицами $\mathbf{R}^2{}_0, \ldots, \mathbf{R}_0{}^{j-1}$ (j — номер цикла повторения пунктов 3°, 4°, где $\mathbf{R}_0{}^{j-1}$ — матрица смежности, для которой в B_j не содержится ни один положительный элемент.

6°. Исключаем из графа G часть G_1, соответствующую подматрице $\mathbf{R}_1$, полученной в результате выполнения пунктов 1°—5° алгоритма. Переход к 7°.

7°. Повторяем работу пунктов 1°—6° алгоритма для графов с матрицами смежности $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_{l-1}$. Переход к 8°.

8°. Получаем разбиение графа G на l частей $G_1, G_2, \ldots, G_l$ с числом вершин в каждой части соответственно $n_1, n_2, \ldots, n_l$. Конец работы алгоритма.

Рассмотрим работу алгоритма на примере разбиения графа G, изображенного на рис. 3.20, на три части, содержащие соответственно 4, 3 и 3 вершины. Запишем матрицу смежности $\mathbf{R}_0$ графа G:

$\mathbf{R}_0 =$

	1	2	3	4	5	6	7	8	9	10	α_k, α_q
1	0	5	0	0	0	0	4	0	1	0	0
2	5	0	1	0	0	0	1	1	0	0	−4
3	0	1	0	2	1	1	0	0	3	0	+2
4	0	0	2	0	1	0	0	0	0	0	−1
5	0	0	1	1	0	1	0	0	0	1	0
6	0	0	1	0	1	0	0	0	0	0	0
7	4	1	0	0	0	0	0	1	1	1	+2
8	0	1	0	0	0	0	1	0	1	0	−1
9	1	0	3	0	0	0	1	1	0	0	+2
10	0	0	0	0	1	0	1	0	0	0	−2

Разобьем ее на две подматрицы: $\mathbf{R}_{0,1}$ и $\mathbf{R}_{0,2}$ — и определим числа связности по формуле (3.38).

Используя выражение (3.39), построим матрицу

$\mathbf{B}_0 =$

	5	6	7	8	9	10
1	0	0	−6	−1	0	−2
2	−4	−4	−4	−7	−2	−6
3	0	0	+4	+1	−2	0
4	−3	−1	+1	−2	+1	−3

Например, элементы $b_{1,5}$, $b_{2,8}$, $b_{3,7}$ матрицы $\mathbf{B}_0$ определяются следующим образом: $b_{1,5}=\alpha_1+\alpha_5-2r_{1,5}=0+0-2\cdot 0=0$; $b_{2,8}=\alpha_2+\alpha_8-2r_{2,8}=-4+(-1)-2\cdot 1=-7$; $b_{3,7}=\alpha_3+\alpha_7-2r_{3,7}=+2+2-2\cdot 0=+4$.

Из матрицы **B** получаем, что для минимизации числа соединительных ребер между частями $G_{0,1}$ и $G_{0,2}$ графа G в матрице $\mathbf{R}_0$ необходимо переставить третьи и седьмые строки и столбцы, что соответствует переразмещению вершин x_3 и x_7. После выполнения указанной процедуры матрица $\mathbf{R}'_0$ принимает вид

$$\mathbf{R}'_0 = \begin{array}{c|cccc|cccccc||c} & 1 & 2 & 7 & 4 & 5 & 6 & 3 & 8 & 9 & 10 & \alpha_{k,q} \\ \hline 1 & 0 & 5 & 4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -8 \\ 2 & 5 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -4 \\ 7 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & -2 \\ 4 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 & +3 \\ \hline 5 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & -2 \\ 6 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & -2 \\ 3 & 0 & 1 & 0 & 2 & 1 & 1 & 0 & 0 & 3 & 0 & -2 \\ 8 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & +1 \\ 9 & 1 & 0 & 1 & 0 & 0 & 0 & 3 & 1 & 0 & 0 & -2 \\ 10 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}.$$

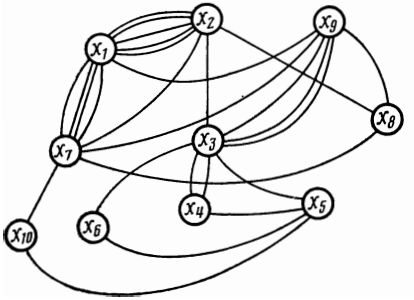


Рис. 3.20. Пример графа для разбиения на основе чисел связности

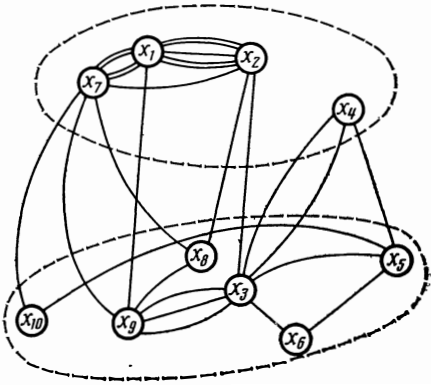


Рис. 3.21. Граф (см. рис. 3.20) после перестановки вершин x_3 и x_7

Граф G после перестановки вершин x_3 и x_7 изображен на рис. 3.21. Построим матрицу

$$\mathbf{B}'_0 = \begin{array}{c|cccccc||} & 5 & 6 & 3 & 8 & 9 & 10 \\ \hline 1 & -10 & -10 & -10 & -7 & -12 & -8 \\ 2 & -6 & -6 & -8 & -5 & -6 & -4 \\ 7 & -4 & -4 & -4 & -3 & -6 & -4 \\ 4 & -1 & +1 & -3 & +4 & +1 & +3 \end{array}.$$

Для максимальной минимизации числа соединительных ребер между частями графа на данном шаге необходимо выбрать элемент $b_{4,8}=+4$ и переставить четвертые и восьмые строки и столбцы матрицы $\mathbf{R}'_0$, что соответствует перестановке вершин x_4, x_8 в графе G (см. рис. 3.21). Граф G, полученный в результате перестановки вершин x_4 и x_8, показан на рис. 3.22. Если определить коэф-

фициенты связности по $\mathbf{R}_0''$, то получим α_k, $\alpha_q \leqslant 0$, следовательно, строить матрицу $\mathbf{B}_0''$ не будем, так как в ней не найдется ни один элемент $b_{k,q}>0$ и в графе G на данном шаге нет пары вершин, перестановка которых уменьшит K. Удаляя из матрицы R''_0 подматрицу $R_{0,1}$, получаем матрицу R_1. Разобьем ее на две подматрицы: $R^1{}_1$ и $R^2{}_1$. Порядок $R^1{}_1$ определяется числом вершин в G_2. Граф G' после разбиения имеет вид (рис. 3.23)

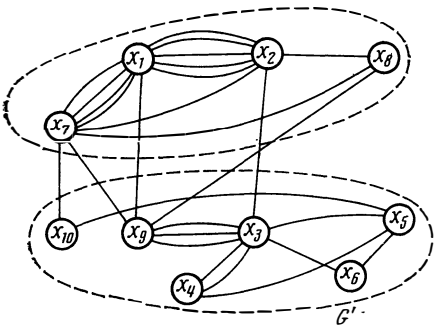


Рис. 3.22. Перестановка вершин x_4 и x_8

$$\mathbf{R}_1=\begin{array}{c|cccccc||c||}
 & 5 & 6 & 3 & 4 & 9 & 10 & \alpha_{k,q}\alpha_q \\
\hline
5 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
6 & 1 & 0 & 1 & 0 & 0 & 0 & -2 \\
3 & 1 & 1 & 0 & 2 & 3 & 0 & +3 \\
\hline
4 & 1 & 0 & 2 & 0 & 0 & 0 & +3 \\
9 & 0 & 0 & 3 & 0 & 0 & 0 & +3 \\
10 & 1 & 0 & 0 & 0 & 0 & 0 & +1
\end{array}.$$

Построим матрицу

$$\mathbf{B}_1=\begin{array}{c|ccc||}
 & 4 & 9 & 10 \\
\hline
5 & +1 & +3 & -1 \\
6 & +1 & +1 & -1 \\
3 & +2 & 0 & +4
\end{array}.$$

Выберем элемент $b_{3,10}=\alpha_3+\alpha_{10}-2r_{3,10}=+3+1-20=+4$. Переставляя третьи и десятые строки и столбцы матрицы $\mathbf{R}_1$, получаем

$$\mathbf{R}'_1=\begin{array}{c|cccccc||c||}
 & 5 & 6 & 10 & 4 & 9 & 3 & \alpha_{k,q},\alpha_q \\
\hline
5 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
6 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
10 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\
\hline
4 & 1 & 0 & 0 & 0 & 0 & 2 & -1 \\
9 & 0 & 0 & 0 & 0 & 0 & 3 & -3 \\
3 & 1 & 1 & 0 & 2 & 3 & 0 & -3
\end{array}.$$

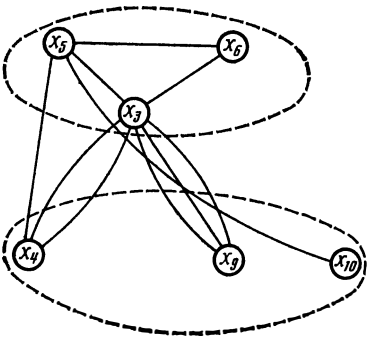


Рис. 3.23. Граф $G'=G/G_0$

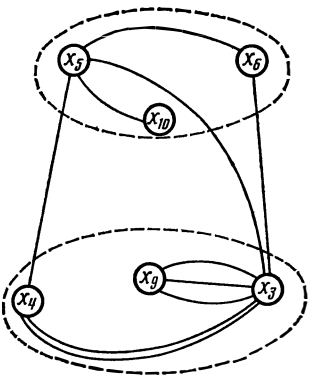


Рис. 3.24. Граф G' после перестановки вершин x_3 и x_{10}

Граф после перестановки вершин x_3 и x_{10} приведен на рис. 3.24. Матрица $\mathbf{B}'_1$ тогда запишется в виде

$$\mathbf{B}'_1 = \begin{array}{c|ccc} & 4 & 9 & 3 \\ 5 & -3 & -3 & -5 \\ 6 & -1 & -3 & -5 \\ 10 & -2 & -4 & -4 \end{array}.$$

В матрице $\mathbf{B}'_1$ нет положительных элементов, следовательно, на этом заканчивается работа алгоритма.

Граф G после разбиения на три части $G_1=(X_1, U_1)$, $X_1=\{x_1, x_2, x_7, x_8\}$; $G_2=(X_2, U_2)$; $X_2=\{x_5, x_6, x_{10}\}$; $G_3=(X_3, U_3)$; $X_3=\{x_3, x_4, x_9\}$ по 4, 3 и 3 вершины в каждой изображен на рис. 3.25. Общее число соединительных ребер $K=8$, а $\Delta(G)\approx 2{,}4$.

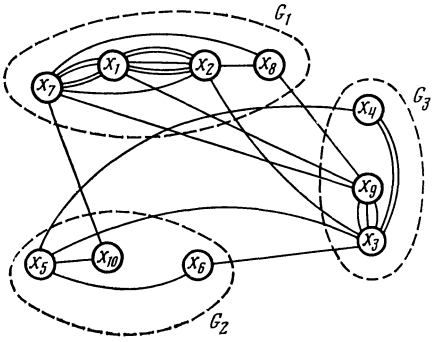


Рис. 3.25. Окончательное разбиение графа (см. рис. 3.20)

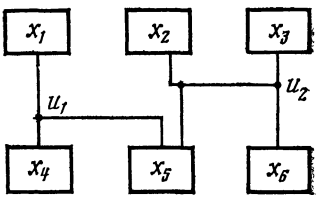


Рис. 3.26. Фрагмент схемы

Можно за счет усложнения процедуры, заключающейся в перестановке групп вершин, получать лучшие решения получаемого разбиения.

Рассмотрим алгоритм итерационного разбиения схемы, когда в качестве модели используется гиперграф, заданный матрицей инцидентности. Для фрагмента схемы (рис. 3.26) матрица, заданная списком, примет вид (здесь первый столбец соответствует номерам элементов, а второй — номерам цепей фрагмента схемы)

$$\mathbf{I} = \begin{bmatrix} [x_1]\ [1] \\ [x_2]\ [2] \\ [x_3]\ [2] \\ [x_4]\ [1] \\ [x_5]\ [1,2] \\ [x_6]\ [2] \end{bmatrix}.$$

Разбиение схемы S на l частей $S_1, S_2, \ldots, S_l$ сводится к многократному разбиению матрицы $\mathbf{I}$ на две части $\mathbf{I}_1$ и $\mathbf{I}_2$ так, чтобы число строк в $\mathbf{I}_1$ соответствовало числу элементов, которое необходимо поместить в S_1, а число строк в $\mathbf{I}_2$ — остальным элементам S. Затем производится перестановка строк из $\mathbf{I}_1$ в $\mathbf{I}_2$ по заданному критерию с целью минимизации внешних связей. Далее S_1

исключается из S, $\mathbf{I}_1$ исключается из $\mathbf{I}$, матрица снова разбивается на две части и процесс повторяется, пока схема не будет разбита на l частей.

При работе итерационных алгоритмов важным является подсчет числа внешних связей. Для этой цели можно перейти от «плавающей» информации о цепях к построению покрывающих деревьев. Однако в этом случае могут быть минимизированы фрагменты цепей, которые исключаются построением другого покрывающего дерева. Поэтому перестройку цепей ведут после каждого шага алгоритма таким образом, чтобы минимизировать число внешних связей. Подсчет K основан на том утверждении, что цепь, инцидентная только элементам одной из частей схемы, внешних связей не образует. Если цепь инцидентна элементам обеих частей разбиения, то ее можно перестроить так, что в ней будет не более одной внешней связи.

Перестановку строк $\mathbf{I}$ между $\mathbf{I}_1$, $\mathbf{I}_2$ необходимо вести таким образом, чтобы возможно большее число цепей находилось полностью или в $\mathbf{I}_1$, или в $\mathbf{I}_2$.

Общее число внешних связей K по $\mathbf{I}$ можно определить следующим образом. Выбирается первая строка из $\mathbf{I}$ и выделяется номер первой цепи, инцидентной x_1. Просматриваются строки из $\mathbf{I}_2$. Если хотя бы в одной строке $\mathbf{I}_2$ встретится такой же номер, следовательно, эта цепь образует одну внешнюю связь. В остальных строках $\mathbf{I}_1$ исключается номер рассматриваемой цепи. Производится переход к следующей цепи, и операции продолжаются аналогично. Затем суммируется число цепей, образующих внешние связи, и в результате получаем число внешних связей, инцидентных x_1. Такая же процедура проводится для всех строк $\mathbf{I}$. Общее число внешних связей $K=\sum_{i=1}^{n_1} K_i$, где K_i — число внешних связей, инцидентных x_i элементу; $n_1=|x_1|$.

Перестановочный коэффициент определяется следующим образом. По матрице $\mathbf{I}$ для каждой строки подсчитываются числа связности α_i. Числа связности для $x_i \in \mathbf{I}_1$ вычисляются как разность между числом фрагментов цепей, ведущих в $\mathbf{I}_2$ и лежащих в $\mathbf{I}_1$. Для элементов $x_i \in \mathbf{I}_2$ числа связности определяются как разность между числом фрагментов цепей, ведущих в $\mathbf{I}_1$ и находящихся в $\mathbf{I}_2$.

На основе чисел связности определяется коэффициент $b_{i,j}$, показывающий, какие строки $i \in \mathbf{I}_1$ и $j \in \mathbf{I}_2$ необходимо поменять местами в $\mathbf{I}$ для минимизации K:

$$b_{i,j}=\alpha_i+\alpha_j-2a_{i,j}-c, \tag{3.40}$$

где α_i, α_j — числа связности; $a_{i,j}$ — число фрагментов цепей, соединяющих элементы x_i, x_j; c — коэффициент, позволяющий учитывать изменение числа внешних связей K при «плавающей» информации о цепях при перестановке элементов x_i, x_j.

Рассмотрим работу итерационного алгоритма на примере фрагмента схемы, изображенного на рис. 3.26. Схему необходимо

разбить на две части по три элемента в каждой с минимизацией числа внешних связей.

Матрицу **I** разделим на две части:

$$\mathbf{I}=\left[\begin{array}{ll}[x_1] & [1]\\ [x_2] & [2]\\ [x_3] & [2]\\ \hline [x_4] & [1]\\ [x_5] & [1,2]\\ [x_6] & [2]\end{array}\right]\begin{array}{l}\}\mathbf{I}_1\\ \\ \\ \\ \}\mathbf{I}_2\end{array}.$$

Для элементов строк $\mathbf{I}_1$ и $\mathbf{I}_2$ определим числа связности: $\alpha_1=2$; $\alpha_2=1$; $\alpha_3=1$; $\alpha_4=0$; $\alpha_5=1$; $\alpha_6=1$. Вычислим значения перестановочных коэффициентов: $b_{1,4}=\alpha_1+\alpha_4-2a_{1,4}-c=2+0-2-0=0$; $b_{1,5}=0$; $b_{1,6}=1$; $b_{2,4}=0$; $b_{2,5}=0$; $b_{2,6}=0$; $b_{3,4}=0$; $b_{3,5}=0$; $b_{3,6}=0$. Так как $b_{1,6}=1$, то переставляем первую и шестую строки, что соответствует переразмещению элементов x_1 и x_6. Матрица **I** примет вид

$$\mathbf{I}=\left[\begin{array}{ll}[x_6] & [2]\\ [x_2] & [2]\\ [x_3] & [2]\\ \hline [x_4] & [1]\\ [x_5] & [1,2]\\ [x_1] & [1]\end{array}\right]\begin{array}{l}\}\mathbf{I}_1\\ \\ \\ \\ \}\mathbf{I}_2\end{array}.$$

Определяя новые числа связности и перестановочные коэффициенты, получаем, что дальнейшая перестановка нецелесообразна. Окончательное разбиение показано на рис. 3.27. Число внешних связей после минимизации $K=1$.

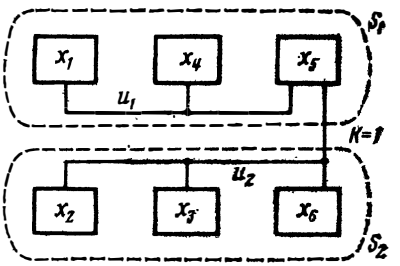


Рис. 3.27. Разбиение схемы на две части

При решении практических задач разбиения электрических схем на части вполне приемлемыми считаются приближенные результаты, при этом конструктору интересно знать, с какой погрешностью получено решение. В этом плане заслуживает внимания *метод случайных назначений,* отличительной особенностью которого является то, что он связывает затраты машинного времени с точностью и надежностью полученных решений. Суть метода состоит в следующем.

Предположим, что граф $G=(X, U)$ необходимо разбить на l частей G_1, G_2, ..., G_l с числом вершин n_1, n_2, ..., n_l $(n_1+n_2+\ ...\ +n_l)=n$.

Рассмотрим случай, когда $n_1=n_2=\ ...\ =n_l=n/l$. Предположим, что $X=\{x_1, x_2, ..., x_n\}$. Пусть y_j — вакантное место, причем $Y_1=$

$= \{y_1, y_2, ..., y_{n/l}\}$ — первая часть разбиения; $Y_2 = \{y_{n/l+1}, y_{n/l+2}, ..., y_{2n/l}\}$ — вторая часть разбиения, ..., $Y_l = \{y_{(l-1)n/l+1}, y_{(l-1)n/l+2}, ..., y_n\}$ — l-я часть разбиения. Тогда определим случайное назначение

$$t = \begin{pmatrix} x_1, & x_2, & ..., & x_n \\ Y_{i_1}, & Y_{i_2}, & ..., & Y_{i_l} \end{pmatrix}$$

как результат случайной подстановки

$$\begin{pmatrix} 1, & 2, & 3, & ..., & n \\ i_1, & i_2, & i_3, & ..., & i_l \end{pmatrix}.$$

Выберем произвольно какую-нибудь вершину графа $G = (X, U)$ и отнесем ее к G_1. Затем из числа оставшихся $(n-1)$ вершин выберем таким же образом вершину для G_2 и т. д. Первое случайное назначение заканчивается после того, как будут размещены все n вершин по l частям. Такая процедура обеспечивает равновероятное распределение вершин и практически реализуется с помощью датчика случайных чисел. После каждого назначения вычисляется число реберных соединений K. Если $K_i \leqslant K_j$, то K_i запоминается. В противном случае запоминается K_j. Величина K является случайной. Для реализации условия $\forall\, G \in B(G_i)[K = K_{\min}]$ необходимо осуществить такое число случайных назначений Q, чтобы с вероятностью не менее β в проделанной серии встретилось хотя бы одно значение K, которое не превосходило бы минимально возможное $K_{\min}$ больше, чем на допустимую ошибку ε, т. е.

$$P\{K - K_{\min} < \varepsilon\} \geqslant \beta. \tag{3.41}$$

Показано, что

$$Q \geqslant \frac{|\ln(1-\beta)|}{\left|\ln \frac{\Phi(K_1 - \varepsilon')}{\Phi(K_1)}\right|}, \tag{3.42}$$

где $\Phi(K_1 - \varepsilon'_1) = \frac{1}{\sqrt{2\pi}} \int_{\infty}^{K_1 - \varepsilon'} e^{-t/2}\, dt$ и $\Phi(K_1) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{K_1} e^{-t/2}\, dt$

— нормальные функции распределения.

В свою очередь

$$K_1 - \varepsilon' = \frac{M(K) - K_{\min} - \varepsilon}{\sigma(K)}; \quad K_1 = \frac{M(K) - K_{\min}}{\sigma(K)}; \quad \varepsilon' = \frac{\varepsilon}{\sigma(K)},$$

где $M(K)$, $\sigma(K) = \sqrt{D(K)}$ — математическое ожидание и среднее квадратическое отклонение случайной величины K соответственно; $K_{\min}$ — нижняя граница усечения распределения случайной величины K.

Математическое ожидание $M(K)$ и среднее квадратическое отклонение $\sigma(K)$ можно определить следующим образом. Для заданной ошибки ε и вероятности β выполняется некоторая серия случайных назначений Q. По полученной совокупности случайной величины K и известным формулам из теории вероятностей опре-

деляется оценка $M'(K)$ и $D'(K)$. Полученные данные позволяют по формуле (3.42) уточнить необходимый объем случайных назначений Q''. Проделав еще $Q''-Q'$ назначений и определив $M''(K)$, $D''(K)$, снова уточняем объем назначений Q'''. Если $Q'''<Q''$, то процесс заканчивается. Среди Q''' назначений встретится, по крайней мере, одно, для которого выполняется условие (3.41). В противном случае вычисления продолжаются.

Для сравнения различных методов перебора при разбиении графов схем можно использовать две оценки: целенаправленность и эффективность ветвления. *Целенаправленность* Z перебора позволяет узнать, в какой мере перебор идет в направлении цели: $Z=D/C$. Здесь D — длина найденного пути до цели; C — общее число вершин, построенных в течение перебора, включая целевую вершину, но исключая начальную. Целенаправленность показывает, насколько дерево, построенное на переборе, вытянуто (не кустисто).

Фактор эффективного ветвления $У$ меньше зависит от длины пути до цели. Его определение основано на представлении о дереве, имеющем глубину, равную длине этого пути и числу вершин, построенных в процессе перебора. Величина $У$ связана с C следующим образом: $C=У(У^2-1)/(У-1)$. Величина $У\to 1$ соответствует перебору, направленному к цели и очень мало ответвляемому на другие направления.

Оценка перебора для выполнения одной итерации алгоритмов разбиения определяется выражением $0(n^\gamma)$, где γ — величина, принимающая для реализуемых на ЭВМ алгоритмов значения из диапазона 1—4. Обычно для последовательных алгоритмов $\gamma=1$, 2, для итерационных $\gamma=2$, 3, 4.

В целях универсализации вида выходной информации на всех этапах конструирования и возможности контроля результаты работы выдаются в виде списков, таблиц на перфолентах, перфокартах и магнитной ленте. Выходная информация этапа компоновки характеризуется следующими параметрами: число частей разбиения; число элементов в каждой части и их номера, число внешних и внутренних соединений; число использованных элементов; число использованных выводов; использованная площадь. В настоящее время перспективными исследованиями при компоновке ЭА на ИМС4 и ИМС5 является учет требований контроля и диагностики. Задача компоновки в этом случае состоит в нахождении некоторого варианта распределения элементов по блокам, оптимизирующего выбранный функционал при соблюдении схемотехнических ограничений. Идея разбиения ЭА на блоки — сбалансировать общую сумму цен элементарных проверок для каждого блока, т. е. найти такое разбиение, при котором затраты на диагностику любого блока ЭА были бы примерно одинаковыми. Тогда задачу разбиения ЭА на блоки можно свести к нахождению минимального разрезающего множества в графе, моделирующем схему.

Обоснованный выбор на этапе разбиения дополнительных контрольных точек улучшает диагностические свойства компонуемых блоков. Для этих целей используются оставшиеся неподключенными внешние контакты кристалла ИМС. В схеме производится поиск цепи с наибольшей нагрузкой. Найденная цепь выводится на внешний контакт, который в этом случае становится дополнительной контрольной точкой. Разбиение ЭА на блоки с учетом диагностических характеристик может быть связано с алгоритмами, минимизирующими суммарную площадь, число внешних соединений и контактов.

Отметим, что при разбиении имеется возможность применять дублирование элементов, если некоторые элементы схем могут принадлежать нескольким модулям. Использование дублирования позволяет уменьшить число внешних соединений за счет увеличения числа логических элементов.

При конструировании ЭА на печатных платах общая длительность задержки, связанная с внутрисхемными соединениями, меньше, чем длительность задержки, определяемая внешними соединениями. Поэтому для оптимизации разбиения схемы также используются критерии минимизации максимальной длительности задержки.

Из рассмотрения методов компоновки ЭА следует, что одним из основных является метод разбиения с минимизацией внешних соединений. Системы автоматизированного проектирования следующих поколений ЭА будут иметь в своем составе многокритериальные алгоритмы компоновки.

3.2. РАЗМЕЩЕНИЕ ЭЛЕМЕНТОВ ГРАФА СХЕМЫ НА ПЛОСКОСТИ

Высокая плотность размещения элементов интегральных микросхем и печатных плат создает большие трудности при реализации соединений между элементами. В этой связи размещение элементов на печатной плате становится определяющим быстроту и качество трассировки. Главное внимание уделяется минимизации внутрисхемных пересечений, суммарной длины соединений и реализации алгоритмов за приемлемое время. Исследования показывают, что критерии минимума пересечений и суммарной длины соединений являются наиболее общими для схем ЭА. Выполнение этих условий обеспечивает повышение надежности, уменьшение размеров конструктивных единиц, минимизацию взаимных наводок, задержек сигналов, уменьшение общей длины соединений и т. п.

Формально задача размещения заключается в определении оптимального варианта расположения элементов на плоскости в соответствии с введенным критерием, например с минимальной взвешенной длиной соединений. Сформулируем задачу размещения. Пусть задано множество элементов с множеством цепей, определенных на подмножествах этих элементов, и пусть задано множество позиций. Необходимо назначать элементы

в позиции таким образом, чтобы суммарная длина связывающих деревьев для всех цепей схемы была минимальна.

Исходными данными при решении задачи размещения являются монтажная плоскость (плата, кристалл, панель и т. д.), как правило, прямоугольной формы, число элементов, которое получено в результате компоновки, т. е. разбиения графа схемы на части, и граф схемы соединений элементов $G=(X, U)$. На монтажную плоскость наносят координатную сетку и оси координат s и t и, таким образом, представляют ее в виде графа G_r.

Задача размещения сводится теперь к отображению заданного графа схемы $G=(X, U)$ в координатную сетку G_r таким образом, чтобы вершины множества X размещались в ее узлах и суммарная длина связей

$$L(G)=1/2\sum_{i,j} d_{i,j} r_{i,j} \tag{3.43}$$

или суммарное число пересечений

$$P(G)=1/2 \sum_{u_j \in U} p(u_{i,j}) \tag{3.44}$$

были наименьшими для возможных способов отождествления вершин графа и узлов сетки. В выражениях (3.43) и (3.44) $r_{i,j}$ — число кратных ребер между вершинами x_i, x_j; $p(u_{i,j})$ — число пересечений ребра $u_{i,j}$; $d_{i,j}$ — расстояние между узлами i и j графа G_r.

При размещении схем ЭА критерии минимума суммарной длины внутрисхемных пересечений и ограничение по тепловому и электрическому режимам некоторым образом противоречивы и оптимизация по одному из них может привести к ухудшению другого критерия. Для совместного учета рассматриваемых критериев рассмотрим обобщающий критерий размещения схем на интегральных микросхемах

$$Q(G)=[L(G), P(G), Э(G)]. \tag{3.45}$$

Здесь $Э(G)$ — электромагнитная и тепловая совместимость элементов; $L(G)$, $P(G)$, $Э(G)$ — управляемые переменные, зависящие от числа ячеек, связей, расположения ячеек относительно друг друга на плоскости.

Это выражение приводит к задаче оптимизации, решение которой, не являясь оптимальным для $L(G)$, $P(G)$ или $Э(G)$, оказывается приемлемым для $Q(G)$ в целом. Оптимальность $Q(G)$ говорит о том, что дальнейшее уменьшение $L(G)$ невозможно без увеличения $P(G)$ или $Э(G)$ и наоборот. Так как построение алгоритмов по обобщенной оценке затруднительно, объединим заданные критерии в один, т. е. построим аддитивную функцию

$$Q(G)=\lambda_1 L(G)+\lambda_2 P(G)+\lambda_3 Э(G). \tag{3.46}$$

Здесь величина λ_i, $i \in I=\{1, 2, ..., l\}$, — весовой коэффициент, характеризующий важность одного критерия по сравнению с другими.

Существует большое число алгоритмов размещения графа схемы с минимизацией суммарной длины соединений и внутрисхемных пересечений. Все алгоритмы можно условно разделить на две группы: непрерывно-дискретные и дискретные. К первой группе относятся алгоритмы, основанные на построении механических аналогов, градиентные методы и др. Ко второй группе относятся итерационные, последовательные, смешанные, а также основанные на идеях метода ветвей и границ.

Последовательные алгоритмы (часто встречается название «конструктивные с начальным размещением») заключаются в выборе первоначально размещаемого элемента или группы элементов с последующим подсоединением неразмещенных элементов. После размещения элементов они уже не перемещаются. Последовательные алгоритмы размещения требуют небольших затрат времени ЭВМ, как правило, они относятся к классу полиномиальных алгоритмов и их сложность $0(n)$. Они широко используются в практике проектирования ИМС4 и ИМС5. Правила выбора и расстановки элементов зависят от конкретных методов.

Итерационные алгоритмы размещения с улучшением качества работают в итеративном режиме. Для изменения позиций элементов выбираются одиночные элементы или группы элементов. Затем по выбранным правилам производится переразмещение элементов для уменьшения общей длины соединений. Каждый этап алгоритма должен приводить к новому размещению с меньшей общей суммарной длиной. Итерационные алгоритмы позволяют получать более качественные результаты, чем последовательные, но за счет бо́льших затрат машинного времени. Они также относятся к классу полиномиальных алгоритмов, но их сложность порядка $0(n^2)$—$0(n^4)$. К итерационным алгоритмам относятся: метод назначений Штейнберга, метод парного обмена, метод соседнего обмена, метод силонаправленной релаксации. К группе итерационных относятся стохастические методы размещения. Основная идея этих методов следующая. Случайным образом распределяются элементы по посадочным местам плоскости с учетом плотности распределения вероятности, которую обычно считают равномерной. Далее определяется, например, суммарная длина соединений в полученном размещении и сравнивается с предыдущим. Лучшее размещение оставляется. Процесс продолжается до тех пор, пока не окончится отведенное время или не будет просмотрено заданное число размещений.

Алгоритмы, основанные на идеях метода ветвей и границ, относятся к точным. При этом множество всех допустимых решений разбивается на меньшие по мощности подмножества, в которых производится поиск оптимального размещения. Метод сопровождается вычислением низших границ, и поиск оптимального размещения прекращается, когда граничное значение начинает превышать значение при найденном допустимом размещении. Процесс продолжается до тех пор, пока не будет закончен поиск

в каждом подмножестве разбиения или не будет найдено оптимальное размещение.

При реализации алгоритмов в общем случае могут получаться локальные минимумы целевой функции. Поэтому представляют интерес получение оценок суммарной длины соединений и числа пересечений для графа схемы с данными числами вершин и ребер. Приведем оценку минимальной суммарной длины ребер графа.

Пусть дан произвольный граф $G=(X, U)$, причем $|X|=n$, $|U|=m$, и задана сетка G_r с шагом, равным 1, число узлов которой больше или равно m. Тогда ребра G будут иметь вес 1, 2, ..., N (N — вес наиболее длинного ребра графа).

Вес любого ребра однозначно определяется матрицей геометрии D_Γ. Если G_r имеет размеры $m\times l$, то $N=m+l-2$. Введем понятие стандартного графа $G_\Delta=(X_\Delta, U_\Delta)$ для графа $G=(X, U)$, отображенного в сетку. Граф G_Δ имеет $|X|=|X_\Delta|$ и $|U|=|U_\Delta|$. Основная идея нахождения нижней оценки суммарной длины ребер произвольного графа заключается в следующем. Сначала подсчитывается число вершин и ребер графа G. Далее в координатной сетке строится стандартный граф $G_\Delta=(X_\Delta, U_\Delta)$, имеющий такое же число вершин и ребер, как и граф G. Построение ведется путем последовательного помещения в сетку сначала всех ребер G_Δ, вес которых равен 1. Если число единичных ребер графа G_Δ равно или больше числа ребер графа G, то процесс построения заканчивается. В противном случае последовательно добавляются ребра с весом 2,3 и далее до тех пор, пока общее число ребер графа G_Δ не станет равным числу ребер графа G. Подсчитав суммарную длину ребер G, равную

$$L(G_\Delta)=f_1+2f_2+\ \dots\ +nf_n, \tag{3.47}$$

где f_i — число ребер с весом i, получим нижнюю оценку минимальной суммарной длины для графа G и для любых графов с таким же числом вершин и ребер, таким же образом отображенных в заданную сетку. Граф G_Δ в общем случае не изоморфен графу G. Из приведенных рассуждений следует следующая теорема.

Теорема 3.5. Минимальная суммарная длина всех произвольных графов с n вершинами и m ребрами не может быть меньше суммарной длины ребер соответствующего стандартного графа G_Δ.

Доказательство теоремы следует из построения стандартного графа.

Рассмотрим последовательно-итерационный алгоритм размещения вершин графа на плоскости с минимизацией суммарной длины соединений. Последовательная часть, которая применяется для упорядочивания множества вершин и получения начального размещения, использует понятие коэффициента связности для каждой вершины. Пусть задан граф схемы $G=(X, U)$ и задана плоскость, на которой размещен граф сетки G_r. Необходимо разместить вершины $x_i\in X$ в узлы сетки с минимизацией суммарной

длины ребер $u_j \in U$. Число узлов G_r больше или равно числу вершин графа G. Для общности рассуждений можно предположить, что часть узлов сетки уже отождествлена с некоторыми вершинами графа схемы. Пусть вершина $x_i \in X$ графа помещена в $(j+1)$-ю позицию, где j — число занятых позиций. Тогда коэффициентом связности $\Delta(x_i)$ называется выражение $\Delta(x_i) = a_{i,h} - a_{i,z}$, где $a_{i,h}$ — число ребер, соединяющих вершину x_i с подмножеством ранее размещенных вершин графа; $a_{i,z} = \rho(x_i) - a_{i,h}$ — число ребер, соединяющих вершину x_i с неразмещенными вершинами графа, тогда

$$\Delta(x_i) = 2a_{i,h} - \rho(x_i). \tag{3.48}$$

Идея алгоритма заключается в определении коэффициента связности для всех неразмещенных вершин и помещении в первую свободную позицию (узел сетки) вершины с максимальным значением $\Delta(x_i)$. Заметим, что, как правило, размещение вершин в узлы G_r начинают с крайнего левого узла G_r. Процесс последовательно продолжается до тех пор, пока не будут размещены все вершины графа. Перейдем теперь к описанию итерационной части алгоритма размещения, основанной на понятии центра тяжести вершины.

Для оценки степени «удобства» v-й позиции для вершины графа x_j введем понятие средней длины ребра для каждой его вершины

$$L_j = \sum_{i=1}^{n} d_{i,j} r_{i,j} / \sum_{i=1}^{n} r_{i,j}, \quad i \neq j, \tag{3.49}$$

где L_j — средняя длина ребер, инцидентных вершине графа x_j, помещенной в позицию сетки v_j. Из (3.49) следует, что

$$(\forall u_{i,j} \in U_j)[(d_{i,j} = 1) \Rightarrow L_j = 1], \tag{3.50}$$

где U_j — множество ребер, инцидентных вершине графа x_j; L_j — средняя длина ребер, инцидентных x_j. Выражение (3.50) показывает, что необходимо стремиться к такому размещению вершин графа, когда средняя длина ребер стремится к минимальному значению. Основная идея алгоритма заключается в следующем.

Из множества вершин графа, расположенного в сетке, выбирается вершина x_j с максимальным L_j и производится ее перестановка с другой вершиной с целью минимизации (3.43). Процесс продолжается до тех пор, пока возможно уменьшение суммарной длины ребер графа. Рассмотрим процедуру выбора второй вершины для переразмещения вершин графа. Представим, что данный граф моделирует некоторую систему материальных точек.

Пусть все точки $x_j \in X$ закреплены, а $y \in X$ движется по некоторому закону. Тогда известно, что минимум суммарной длины расстояний от y до точек x_j (рис. 3.28) имеет место, когда точка y помещена в точку с координатами центра тяжести (s_c, t_c) рассматриваемой системы.

Итак, можно сделать выводы: ищется такое положение для вершины $y \in X$, которое обеспечит равновесие в сетке при условиях, когда все остальные вершины закреплены и между y и точками $x_j \in X$ ($j=1, 2, \ldots, n-1$) действует сила притяжения, пропорциональная $d_{i,j} r_{i,j}$. Координаты центра тяжести системы материальных точек, расположенных в сетке G_r, определяются:

$$s_c = \sum_{j=1}^{n} m_j s_j / \sum_{j=1}^{n} m_j; \; t_c = \sum_{j=1}^{n} m_j t_j / \sum_{j=1}^{n} m_j, \tag{3.51}$$

где m_j — масса материальной точки (вершины графа) x_j; s_j, t_j — координаты материальной точки x_j. Принимая во внимание, что между двумя точками (вершинами) может находиться несколько связей $r_{i,j} \neq 1$ и считая $m_j = 1$, $i \neq j$, получаем

$$s_c = \sum_{j=1}^{n} s_j r_{i,j} / \rho(x_i); \; t_c = \sum_{j=1}^{n} t_j r_{i,j} / \rho(x_i). \tag{3.52}$$

В общем случае полученная точка (s_c, t_c) не принимает целочисленного значения координат прямоугольной сетки. Поэтому необходимо выделить некоторые подмножества вершин, являющихся ближайшими соседями для вершины с координатами (s_c, t_c). Получим некоторое подмножество вершин сетки, которое образует область возможных перестановок с первой выбранной вершиной. Рассмотрим теперь способ выделения из указанного подмножества такой вершины графа, перестановка которой с уже выбранной дает максимальное уменьшение суммарной длины ребер графа. Пусть для $x_j \in X$ определены (s_c, t_c) и область соседних вершин $X' \subset X$. Определим вершину $x_i \in X'$, которую необходимо переставить с x_j. Пусть вершина x_j находится в q-й позиции, и для нее определена L_j. Определим для вершины x_j значение $L^{\varphi}{}_j$, где $\varphi \in \Psi$ — множество ближайших позиций. Для выбора необходимой позиции найдем значения отклонений

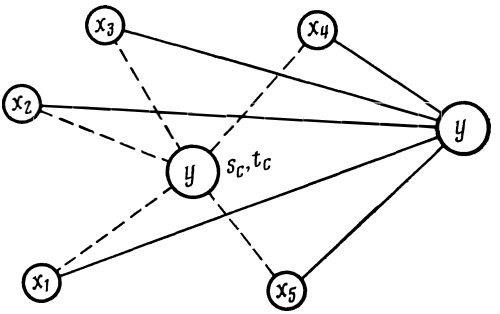


Рис. 3.28. Пример размещения точки y в точке с координатами (s_c, t_c)

$$\sigma^{\varphi}{}_j = L_j - L^{\varphi}{}_j. \tag{3.53}$$

Здесь $L^{\varphi}{}_j = \sum_{i=1}^{n} d_{i,j} r_{i,j} / \rho(x_j)$. Позиция сетки, для которой выражение (3.53) максимально, будет наиболее выгодной для помещения в нее вершины x_j, так как обеспечивает наибольшее уменьшение

суммарной длины ребер, инцидентных вершине x_j. Аналогично для каждой вершины $x_i \in X'$ определяется отклонение

$$\sigma^q{}_i = L^{\varphi}{}_q - L_i, \tag{3.54}$$

где q — номера узлов графа сетки G_r. Для перестановки вершины определим алгебраическую сумму

$$\delta_{i,\,j} = \sigma^{\varphi}{}_j + \sigma^q{}_i. \tag{3.55}$$

Для переразмещения выбирается та пара вершин, для которой выражение (3.55) имеет максимальное положительное значение. Если среди X' нет вершины, для которой (3.55) является положительной величиной, то необходимо выбрать подмножество $X'' \subset X\,(X' \subset X'')$. Если и в этом случае не окажется вершины, для которой $\delta_{i,\,j} > 0$, то производится переход к выбору вершины с другим значением L_j. Процесс минимизации $L(G)$ следует считать законченным, если все коэффициенты L_j будут меньше двух. Так как на каждом шаге для перестановки выбирается пара x_i, x_j, для которой $\delta_{i,\,j} > 0$, то итерационный процесс размещения всегда сходится.

Рассмотрим работу итерационной части алгоритма для графа схемы G, отображенного в сетку и показанного на рис. 3.29. Матрица смежности имеет вид

$$R = \begin{array}{c|ccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 0 & 0 & 0 & 3 & 0 & 0 & 2 & 3 & 0 \\ 2 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 4 & 3 & 1 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 5 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 4 \\ 6 & 0 & 0 & 0 & 5 & 2 & 0 & 5 & 6 & 0 \\ 7 & 2 & 0 & 0 & 0 & 0 & 5 & 0 & 3 & 2 \\ 8 & 3 & 0 & 0 & 0 & 0 & 6 & 3 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 4 & 0 & 2 & 0 & 0 \end{array}.$$

Определим L_j для всех вершин графа. Для этого, используя матрицу геометрии, определяем значение $d_{i,j}$. Далее по формуле (3.49) получим L_j. Рассмотрим вершину x_1. Для нее $d_{1,4} = |x_1 - x_4| + |y_1 - y_4| = |0-0| + |0-1| = 1$; $d_{1,7} = 2$, $d_{1,8} = 3$; $L_1 = (r_{1,4}d_{1,4} + r_{1,7}d_{1,7} + r_{1,8}d_{1,8}) / (r_{1,4} + r_{1,7} + r_{1,8}) = 2$. Остальные коэффициенты определяются аналогично: $L_2 \approx 1{,}3$; $L_3 = 1{,}5$; $L_4 \approx 1{,}7$; $L_5 \approx 1{,}7$; $L_6 \approx 2{,}2$; $L_7 \approx 2{,}2$; $L_8 = 2$; $L_9 = 2$.

С помощью выражения (3.52) найдем координаты центра тяжести для вершины графа x_6:

$$s^6{}_c = (5s_4 + 2s_5 + 5s_7 + 6s_8) / (r_{6,4} + r_{6,5} + r_{6,7} + r_{6,8}) \approx 1{,}4;$$
$$t^6{}_c = (5t_4 + 2t_5 + 5t_7 + 6t_8) / (r_{6,4} + r_{6,5} + r_{6,7} + r_{6,8}) \approx 1{,}4.$$

Видим, что в подмножество X' (круг радиуса 1 с центром 1,4, 1,4) попали вершины x_4, x_5, x_7, x_8.

Теперь произведем условную перестановку вершины x_6 с вершинами подмножества X', т. е. x_4, x_5, x_7, x_8. Для этого, используя выражения (3.49), определяем значения L_j и L^{φ}_j:

$$L_6^{\Phi_4} = \frac{d_{4,6}\,r_{6,4} + d_{4,5}\,r_{6,5} + d_{4,7}\,r_{6,7} + d_{4,8}\,r_{6,8}}{r_{6,4} + r_{6,5} + r_{6,7} + r_{6,8}} =$$

$$= \frac{2\cdot 5 + 1\cdot 2 + 1\cdot 5 + 2\cdot 6}{5+2+5+6} \approx \frac{29}{18} \approx 1{,}6\ ;$$

$$L_6^{\Phi_5} = \frac{d_{5,6}\,r_{6,5} + d_{5,4}\,r_{6,4} + d_{5,7}\,r_{6,7} + d_{5,8}\,r_{6,8}}{r_{6,5} + r_{6,4} + r_{6,7} + r_{6,8}} = \frac{23}{18} \approx 1{,}3\ ;$$

$$L_6^{\Phi_7} = \frac{d_{7,6}\,r_{6,7} + d_{7,4}\,r_{6,4} + d_{7,5}\,r_{6,5} + d_{7,8}\,r_{6,8}}{r_{6,7} + r_{6,4} + r_{6,5} + r_{6,8}} = \frac{30}{18} \approx 1{,}7\ ;$$

$$L_6^{\Phi_8} = \frac{29}{18} \approx 1{,}6\ ;$$

$$L^6_4 = \frac{d_{6,4}\,r_{4,6} + d_{6,1}\,r_{4,1} + d_{6,2}\,r_{4,2}}{r_{4,6} + r_{4,1} + r_{4,2}} = \frac{2\cdot 5 + 3\cdot 3 + 2\cdot 1}{5+3+1} \approx 2{,}3\ ;$$

$$L^6_5 \approx 1;\ L^6_7 \approx 2{,}4;\ L^6_8 \approx 2{,}5.$$

Здесь, например, $L_6^{\Phi_4}$ — это значение средней длины ребер вершины x_6 при условном расположении в сетке G_r на месте вершины x_4. Величины L^6_4, L^6_5, L^6_7, L^6_8 — это значения средней длины ребер вершин x_4, x_5, x_7, x_8 при условном их расположении на месте вершины x_6.

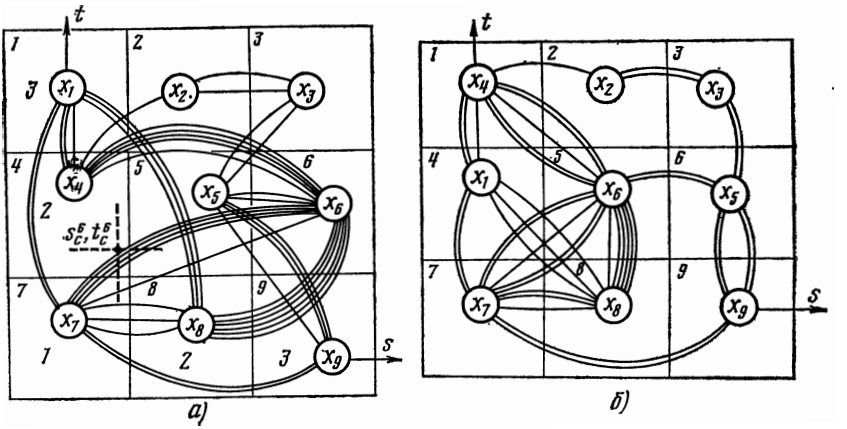


Рис. 3.29. Пример графа для иллюстрации итерационной части алгоритма (*а*) и его окончательное размещение (*б*)

Используя выражения (3.54) и (3.55), получаем отклонения σ_{i,q_6} и $\delta_{i,j}$. Максимальное $\delta_{5,6}$ соответствует паре вершин x_6, x_5, поэтому, произведя первую итерацию, переставим вершины x_6 и x_5. Выполнив аналогичные действия, получим расположение графа на плоскости (в сетке) и показанное на рис. 3.29,*б*. Как видим, суммарная длина уменьшилась с $L(G)=76$ до $L(G)=55$.

Рассмотрим еще один итерационный алгоритм размещения. Предположим, что задано первоначальное произвольное расположение графа на плоскости. Из множества его вершин выбирается подмножество, для которых разность между числом «длинных» и «коротких» ребер положительна. Из них выбирается такая вершина среди смежных, для которой имеется множество вершин, связанных с данной вершиной «длинными» ребрами, причем расстояние между ними не превышает длины «коротких» ребер. После этого из указанного множества необходимо найти такую вершину в графе, перестановка которой с ранее выбранной на данном шаге приводит к наибольшему уменьшению суммарной длины ребер. Для этого множество вершин графа разбивается на два подмножества, в первое из которых включаются вершины, отстоящие от выбранной на величину, не превышающую длину «коротких» ребер, а во второе подмножество — вершины, отстоящие от нее на расстоянии, равном величине «длинных» ребер. Из второго подмножества выбирается вершина, у которой среди множества инцидентных ей ребер наибольшее число инцидентно вершинам, принадлежащим первому подмножеству. Аналогично находится и переставляется другая пара вершин графа. Этот процесс продолжается до тех пор, пока возможно уменьшение суммарной длины ребер графа. В том случае, когда выполнять одиночные перестановки нецелесообразно, предлагается вариант групповой перестановки.

Пусть задан граф соединения элементов схемы, обозначаемый через $G=(X, U)$. Пусть, кроме того, задана сетка, число узлов которой не меньше общего числа вершин графа. На сетке выделяется область требуемой конфигурации. Произвольно занумеруем узлы внутри выбранной области числами натурального ряда 1, 2, 3, ... Затем произведем размещение вершин графа в узлы сетки так, чтобы совпали их соответствующие номера.

Очевидно, что в общем случае такое размещение графа G не является оптимальным. Для минимизации суммарной длины ребер графа предлагается выбрать и переставить некоторые вершины $x_i \in X$ и $x_j \in X$.

Оптимизацию размещения вершин графа G будем проводить следующим образом. Для каждой вершины $x_i \in X$ графа G зададим разбиение множества U_i инцидентных ей ребер на два непересекающихся подмножества U'_i и U''_i. Подмножеству U'_i отнесем ребра, весовая функция которых не превышает некоторого предельного значения α, а подмножеству U_i'' — все остальные ребра, принадлежащие U_i. Назовем первое подмножество α-ребер, а второе подмножеством β-ребер. Подсчитаем число ребер каждого подмножества U'_i и U''_i. Запишем таблицу, в которой для каждой вершины $x_i \in X$ проставим соответствующие значения U'_i, U''_i и определим разность $|U'_i| - |U''_i| = \Delta_i$. Очевидно, что $\Delta_i \lesssim 0$. Ясно, что вершины, приводящие к уменьшению суммарной длины ребер, необходимо выбирать среди вершин, у которых соответ-

ствующие элементы столбца Δ_i положительны. Для каждой вершины графа из подмножества с положительными Δ_i рассечем плоскость сетки на две части горизонтальной секущей линией относительно этой вершины. Подсчитаем число «длинных» ребер, инцидентных этой вершине, входящих в обе части. Зафиксируем ту область, которая содержит наибольшее число вершин, связанных с выбранной «длинными» ребрами. После этого рассечем граф вертикальной секущей линией, проходящей через ту же вершину, и снова подсчитаем число таких вершин в обеих частях. Сравним число вершин после первого и второго сечений графа. Из вершин графа с положительными Δ_i выберем вершину, имеющую наибольшее число инцидентных ей β-ребер в горизонтальной или вертикальной областях. Эту вершину обозначим $x_i \subset X$. При выборе второй вершины предпочтение отдадим той области, где имеется наибольшее вхождение β-ребер.

Суммарную длину ребер графа можно уменьшить за счет уменьшения длины β-ребер или, если это возможно, полного их устранения, а также за счет увеличения числа α-ребер. Учитывая сказанное, будем искать вторую вершину, которую целесообразно переставить с уже выбранной x_i. Для поиска такой вершины предлагается следующий формальный метод. Выделим из множества вершин графа непересекающиеся подмножества X'_1 и X''_1, $X'_1 \cup X''_1 = X$; в подмножество X'_1 включаются кроме вершины x_i вершины, которые отстоят от x_i на расстоянии не больше α. В подмножество X''_1 включаются все остальные вершины. Вторую вершину выберем из области, в которой лежат вершины, инцидентные β-ребрам вершины x_i. Множество вершин этой области обозначим через X_j. Для каждой вершины $x_j \in X_j$ подсчитаем число ребер, инцидентных вершинам из подмножества X'_1 и X''_1. Построим вторую таблицу, в первом столбце которой для каждой вершины $x_j \in X_j$ укажем число ребер $|U''_j|$, инцидентных вершинам подмножества X'_j, а во втором — число ребер $|U'_j|$, инцидентных вершинам подмножества X''_1. После этого найдем разность $|U''_j| - |U'_j| = \sigma_j$. В качестве вершины $x_j \in X_j$, которую необходимо переставить с вершиной x_i, выберем вершину, имеющую наибольшее значение σ_j. Легко показать, что перестановка вершин x_i и x_j приводит к уменьшению суммарной длины ребер графа, если

$$\Delta_i + \sigma_j > 0. \qquad (3.56)$$

Действительно, пусть переставлены вершины, для которых выполняется условие (3.56), и произошло увеличение суммарной длины ребер графа. Однако увеличение может произойти лишь в том случае, если у переставляемых вершин увеличится число β-ребер и уменьшится число α-ребер, что невозможно, так как это противоречит выбору Δ_i и G_j для соответствующих вершин графа.

Так как эффективность перестановки вершин x_i и x_j графа в значительной мере зависит от того, насколько удачно произведено разбиение ребер графа на α- и β-ребра, то приведем приближенную оценку этого разбиения. Иначе говоря, определим, какие

ребра графа следует включать в подмножество β-ребер U'', а какие — в подмножество α-ребер U'.

До перестановки вершин графа G суммарная длина ребер $L(G)$ определяется следующим образом:

$$L(G)=L_\alpha+L_\beta.$$

Представим L_α и L_β в виде

$$L_\alpha=s_1+2s_2+\ \dots\ +(k-1)s_{k-1}+ks_k=\sum_{\xi=1}^{k}\xi s_\xi;$$

$$L_\beta=(k+1)s_{k+1}+(k+2)s_{k+2}+\ \dots\ +(N-1)s_{N-1}+Ns_N=\sum_{\eta=k+1}^{N}\eta s_\eta,$$

где s_ξ (s_η) — число ребер длины $\xi(\eta)$; ξs_ξ (ηs_η) — суммарная длина ребер длины $\xi(\eta)$; N — наибольшая из длин ребер графа [если граф расположен в сетке с размерами $p\times q$, то $N\leqslant(p+q-2)$]. После перестановки двух выбранных вершин суммарная длина ребер изменится и станет равной $L'(G)=L'_\alpha+L'_\beta$. Предположим, что все α-ребра увеличились, а все β-ребра уменьшились на некоторую величину. Тогда можно записать

$$L'_\alpha=(s_1+a_1)+2(s_2+a_2)+\ \dots\ +k(s_k+a_k)=\sum_{\xi=1}^{k}\xi(s_\xi+a_\xi);$$

$$L'_\beta=(k+1)(s_{k+1}-b_{k+1})+(k+2)(s_{k+2}-b_{k+2})+\ \dots\ +N(s_N-b_N)=$$
$$=\sum_{\eta=k+1}^{N}\eta(s_\eta-b_\eta),$$

где ξa_ξ и ηb_η — соответственно прирост и уменьшение суммарной длины α- и β-ребер. Очевидно, что перестановка целесообразна, если выполняется условие $L'(G)<L(G)$. Отсюда следует, что

$$a_1+2a_2+\ \dots\ +ka_k<(k+1)b_{k+1}+(k+2)b_{k+2}+\ \dots\ +Nb_N \qquad (3.57)$$

или

$$\sum_{i=1}^{k}a_ii<\sum_{j=k+1}^{N}b_jj.$$

Рассмотрим неравенство

$$a(1+2+\ \dots\ +k)<b(k+1+k+2+\ \dots\ +N), \qquad (3.58)$$

где $a=\max(a_1, a_2, \dots, a_k)$; $b=\min(b_{k+1}, b_{k+2}, \dots, b_N)$.

Очевидно, что если k определить из неравенства (3.57), то (3.58) будет заведомо выполняться. Если $a>b$, то $(1+2+\ \dots\ +k)<(k+1+k+2+\ \dots\ +N)$, откуда

$$k(k+1)<0{,}5N(N+1). \qquad (3.59)$$

Если выбрать $b=1$, а $a=N$, то

$$k'(k'+1)<0{,}5(N+1). \qquad (3.60)$$

Таким образом, k и k' определяют границу изменения α.

Заметим, что если невозможно выбрать перестановку вершин, уменьшающую суммарную длину ребер, то можно произвести перестановку групп вершин. Для нахождения групповых перестановок вершин графа предлагается эвристический прием, использующий операцию факторизации графа G. Разобьем множество вершин X графа G на n классов $X_1, X_2, \ldots, X_n$ и отнесем к ним вершины, лежащие в строках сетки внутри области размещения. В каждом классе выделим внутренние и соединительные ребра. Теперь построим новый граф, в котором вершинами являются классы X_i. Получим линейно-расположенный мультиграф G_n, содержащий только соединительные ребра. Аналогично проведем операцию разбиения на m классов, вершины каждого из которых принадлежат столбцам области размещения. Получим мультиграф G_m. Для мультиграфов G_n и G_m нетрудно определить перестановку вершин на основе последовательного алгоритма. После получения оптимального размещения мультиграфа производится переход к графу G. Заметим, что указанный прием факторизации легко распространить на случай размещения графа в области произвольной конфигурации.

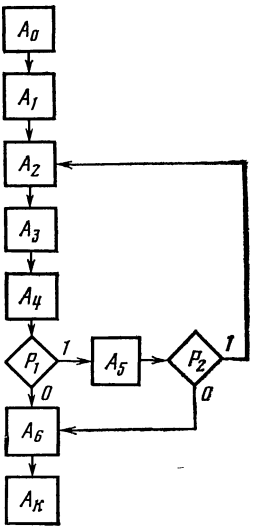


Рис. 3.30. Граф-схема алгоритма размещения графа на плоскости

Сформулируем теперь алгоритм размещения графа на плоскости и запишем его в виде граф-схемы алгоритма (рис. 3.30).

A_0, A_k — операторы начала и конца соответственно;

A_1 — оператор, выполняющий произвольное размещение графа G в узлах заданной решетки;

A_2 — построение таблицы, определение x_i с наибольшим положительным Δ_i;

A_3 — разбиение множества X на два подмножества X'_1 и X''_1 относительно x_i;

A_4 — построение таблицы, выбор x_j, для которой σ_j — наибольшее положительное число;

A_5 — перестановка вершин x_i и x_j;

A_6 — факторизация графа G по строкам и столбцам, минимизация суммарной длины связей мультиграфов, переход к графу G;

p_1, p_2 — логические условия; если $p_1=1$, то за оператором A_4 выполняются A_5 и логическое условие p_2; если $p_1=0$, то за A_4 выполняется A_6; если $p_2=1$, то за оператором A_5 выполняется A_2, A_3; ...; если $p_2=0$, то за A_5 выполняется A_6. Логическое условие p_1 проверяет выполнение условия (3.56), p_2 проверяет наличие β-ребер.

Работу алгоритма покажем на примере размещения графа в сетке, показанного на рис. 3.31.

По графу G построим матрицу расстояний, элементы которой образуются путем поэлементного умножения матрицы графа G_r на матрицу смежности графа G:

$$\mathbf{D}=\begin{array}{c|cccccccccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 5 & 0 & 4 & 5 & 0 \\
2 & 0 & 0 & 1 & 2 & 2 & 1 & 0 & 3 & 3 & 2 & 0 & 4 & 0 & 0 & 0 & 5 \\
3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 5 & 4 & 0 & 4 \\
4 & 0 & 2 & 0 & 0 & 0 & 3 & 2 & 1 & 5 & 4 & 3 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 2 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 5 \\
6 & 0 & 1 & 0 & 3 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 4 \\
7 & 0 & 0 & 0 & 2 & 2 & 0 & 0 & 0 & 3 & 2 & 0 & 0 & 0 & 0 & 2 & 0 \\
8 & 0 & 3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\
9 & 0 & 3 & 0 & 5 & 0 & 0 & 3 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
10 & 0 & 2 & 0 & 4 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
11 & 4 & 0 & 2 & 3 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
12 & 5 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 3 & 0 & 0 \\
13 & 0 & 0 & 5 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \\
14 & 4 & 0 & 4 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 3 & 1 & 0 & 0 & 0 \\
15 & 5 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 \\
16 & 0 & 5 & 4 & 0 & 5 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}.$$

Суммарная длина ребер графа G равна $L(G)=114$. Из условия (3.59) найдем $k=3$. Построим табл. 3.1, из которой выберем вершину x_1. Для вершины x_1 определим область перестановок и построим табл. 3.2. Из табл. 3.2 выберем вершину x_{16}. Произведем перестановку вершин x_1 и x_{16}. Одиночные перестановки повторим до тех пор, пока выполняется условие (3.56). В результате получим размещение, показанное на рис. 3.32.

Таблица 3.1

№	$\lvert U''_i\rvert$	$\lvert U'_i\rvert$	Δ_1	№	$\lvert U''_i\rvert$	$\lvert U'_i\rvert$	Δ_1
1	0	4	4	9	3	1	—2
2	7	2	—5	10	4	1	—3
3	3	3	0	11	5	1	—4
4	5	2	—3	12	3	2	—1
5	4	1	—3	13	3	1	—2
6	4	1	—3	14	2	3	1
7	5	0	—5	15	3	1	—2
8	3	1	—2	16	0	4	4

Произведем факторизацию графа G по строкам и столбцам, которая показана соответственно на рис. 3.33,*а* и *б*. В итоге получим размещение графа G, представленное на рис. 3.34, суммарная длина ребер которого $L(G)=65$.

Рассмотрим кратко перспективные исследования при разработке итерационных алгоритмов, к которым относят:

квадратичное размещение;

размещение при делении плоскости пополам;

размещение на основе частичного деления плоскости пополам.

При квадратичном размещении исходным является размещение, полученное последовательным алгоритмом. Далее плоскость размещения делится пополам вертикальной разрезающей линией на две полуплоскости: p_1 и p_2. Затем p_1 и p_2 делятся пополам го-

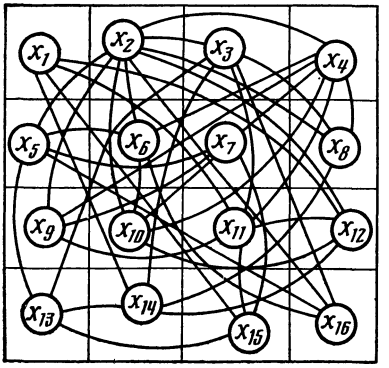


Рис. 3.31. Пример графа для размещения

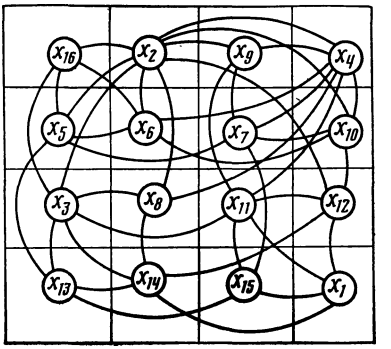


Рис. 3.32. Размещение после выполнения единичных перестановок

ризонтальной разрезающей линией. Этот процесс по возможности продолжается, а затем начинается процесс восходящего переразмещения с перестановкой элементов между разрезающими линиями для минимизации суммарной длины.

Таблица 3.2

№	$\lvert U''_j \rvert$	$\lvert U'_j \rvert$	σ_j
11	4	2	2
12	3	2	1
14	3	2	1
15	3	1	2
16	4	0	4

В алгоритмах второго типа выбирается стратегия деления плоскости в столбце (строке), а затем в строке (столбце). Сначала выполняется вертикальное (горизонтальное) деление плоскости пополам. Затем каждая полуплоскость снова делится до тех пор, пока каждый элемент схемы не будет назначен в столбец (строку). Этот метод аналогичен методам, в которых сначала назначаются элементы в столбцы, а затем производится перестановка элементов в столбцах, чтобы оптимизировать размещение.

Алгоритмы третьего типа аналогичны разбиению схемы на две части. Здесь плоскость разбивается на две полуплоскости с l элементами и $n—l$ элементами, затем снова выполняется разбиение и т. д., пока все элементы не будут назначены в строку. Назна-

чение элементов в столбец производится через вертикальное деление плоскости. Заметим, что эти подходы наиболее целесообразно применять для совместного разбиения, размещения и трассировки печатных плат.

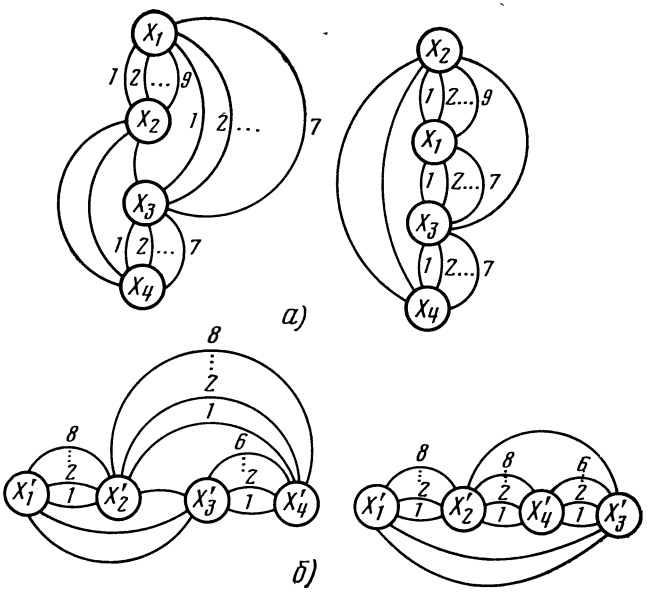


Рис. 3.33. Факторизация:
а — по столбцам; *б* — по строкам

Рассмотрим алгоритм линейного размещения графа схемы методом ветвей и границ с минимизацией суммарной длины соединений. При проектировании внутренней топологии элемента интегральной микросхемы наиболее целесообразно применять точные методы, легко реализуемые на ЭВМ. Пусть задан граф коорди-

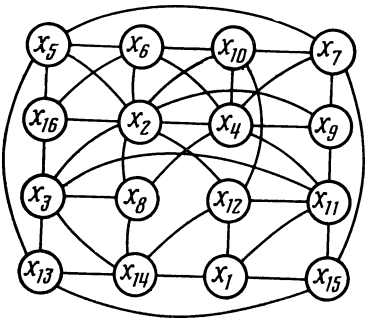


Рис. 3.34. Окончательное размещение графа (см. рис. 3.31)

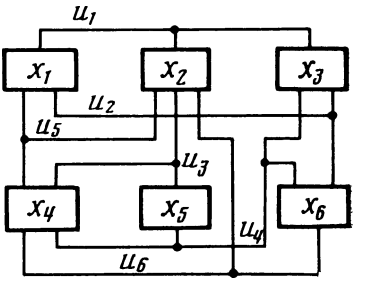


Рис. 3.35. Фрагмент схемы

натной сетки G_r, в котором число столбцов равно числу вершин размещаемого графа схемы $G=(X, U)$, а число строк равно единице. Задача состоит в отображении вершин графа в ячейки графа сетки G_r по критерию минимума суммарной длины ребер, т. е. в нахождении подстановки

$$t=\begin{pmatrix} 1, & 2, & \ldots, & n \\ x_1, & x_2, & \ldots, & x_n \end{pmatrix},$$

где 1, 2, ..., n — номера позиций сетки;

$i=1, 2, \ldots, n$ — вершины размещаемого графа. Известно, что число таких подстановок равно $n!$

Для эффективного решения задачи линейного размещения методом ветвей и границ необходимо: а) выбрать метод нахождения нижней границы суммарной длины ребер графа и б) определить метод ветвления дерева решений, позволяющий отсекать ветви, содержащие заведомо непригодные решения.

Рассмотрим три способа вычисления нижней оценки. В первом из каждой строки треугольной матрицы геометрии D_r выбираются элементы с наименьшим весом и определяется их сумма. Затем из D_r выбираются нерассмотренные элементы $d_{i,j}$, и для них определяются наименьшие возможные значения расстояний при их размещении в линейке. Общая сумма $L(G)=\sum_{i,j} \min d_{i,j}+\delta$, где δ — суммарная длина элементов $d_{i,j}$, выбранных на втором шаге. Во втором способе в D_r выбираются последовательно все элементы, соответствующие ребрам G, и размещаются в линейке (каждое оптимальным образом). Общая сумма расстояний всех ребер дает нижнюю оценку минимальной суммарной длины (стандартный граф).

В третьем способе последовательно выбираются все вершины графа с инцидентными им ребрами и располагаются по очереди в линейке таким образом, чтобы суммарная длина связей была минимальна. Очевидно, что для каждой вершины можно найти такое размещение. Сумма длин связей звездных подграфов определит нижнюю оценку минимальной суммарной длины.

Рассмотрим пример нахождения нижней оценки суммарной длины фрагмента схемы, изображенного на рис. 3.35:

$$\mathbf{R}=\begin{array}{c|l} x_1 & u_1,\ u_5,\ u_2 \\ x_2 & u_1,\ u_3,\ u_5,\ u_6 \\ x_3 & u_1,\ u_4,\ u_2 \\ x_4 & u_5,\ u_3,\ u_4,\ u_6 \\ x_5 & u_3,\ u_4 \\ x_6 & u_2,\ u_4,\ u_6 \end{array}.$$

При замене цепей u_1—u_6 покрывающими деревьями схема может принять вид графа на рис. 3.36, а список связности вершин и ребер запишется в виде

$$(x_1u_1x_2),\ (x_1u_5x_2),\ (x_1u_2x_3),$$
$$(x_2u_1x_3),\ (x_2u_6x_4),\ (x_2u_3x_4),$$

$$(x_2u_5x_4),\ (x_3u_4x_4),\ (x_4u_3x_5),$$
$$(x_4u_4x_6),\ (x_4u_6x_6),\ (x_5u_4x_6),$$
$$(x_3u_2x_6).$$

Последовательно разместим вершины графа схемы в узлах линейки (рис. 3.37), построим матрицу геометрии и каждым из описанных методов определим нижнюю оценку минимальной суммарной длины ребер. Матрица геометрии

$$D_{\text{г}} = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 2 & 2 & 0 & 0 & 0 \\ 2 & 2 & 0 & 1 & 6 & 0 & 0 \\ 3 & 2 & 1 & 0 & 1 & 0 & 3 \\ 4 & 0 & 6 & 1 & 0 & 1 & 4 \\ 5 & 0 & 0 & 0 & 1 & 0 & 1 \\ 6 & 0 & 0 & 3 & 4 & 1 & 0 \end{array}.$$

В первом случае выберем в каждой строке $D_{\text{г}}$ элементы $d_{i,j}$ с минимальным весом и определим их сумму $\sum_{i,j} \min d_{i,j} = 2+1+1+1+1=6$. Неучтенными остались элементы $d_{1,3}$, $d_{2,4}$, $d_{3,6}$ и $d_{4,6}$. Расположив соответствующие ребра с минимальной суммарной длиной, получим $d_{1,3}=1$, $d_{2,4}=3$, $d_{4,6}=2$, $d_{3,6}=1$, $\delta=7$ и $L(G)=13$.

Во втором случае используем построение стандартного графа, тогда получим $L(G_{\Delta}) = 3+2+2+1+1+2+2+2+2=17$.

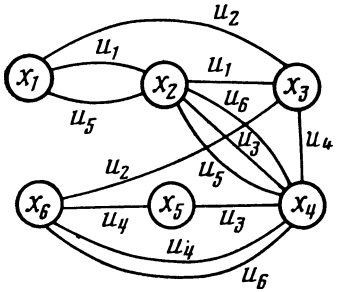


Рис. 3.36. Граф фрагмента схемы

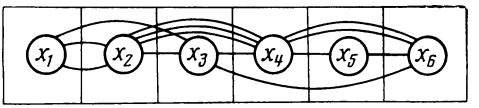


Рис. 3.37. Размещение графа в линейке

В третьем случае, выбрав наилучшее размещение для каждой вершины с инцидентными ребрами, определим суммарную длину всех звездных подграфов:

$$\sum_{\substack{i=2 \\ i\neq 1}}^{n} d_{1,i}=3; \quad \sum_{\substack{i=1 \\ i\neq 2}}^{n} d_{2,i}=2+3+2=7; \quad \sum_{\substack{i=1 \\ i\neq 3}}^{n} d_{3,i}=1+1+2+2=6;$$

$$\sum_{\substack{i=1 \\ i\neq 4}}^{n} d_{4,i}=3+2+2+2=9; \quad \sum_{\substack{i=1 \\ i\neq 5}}^{n} d_{5,i}=2; \quad \sum_{\substack{i=1 \\ i\neq 6}}^{n} d_{6,i}=2+1+2=5.$$

Тогда $L(G)=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{i,j}=16$.

Заметим, что проще всего определять первую оценку, хотя она и является завышенной. Вторая и третья оценки точнее, но процесс их вычисления сложнее.

Для определения нижней границы суммарной длины $L(G)$ ребер графа G воспользуемся понятием стандартного графа G_{Δ} (вторая оценка).

Для разбиения множества решений Q на подмножества, т. е. ветвления дерева решений, будем фиксировать некоторые вершины в определенных позициях сетки. Если, например, зафиксировать вершину x_i на любом месте сетки, то получим подмножество $Q_i \subset Q$, содержащее $(n-1)!$ решений. При фиксации двух, трех и т. д. вершин получим соответственно $(n-2)!$, $(n-3)!$ и т. д. решений. Нетрудно видеть что, используя такой метод ветвления, получаем подмножество, содержащее одно решение.

Пусть вершины $x_1, x_2, \ldots, x_q$ фиксированы соответственно в позициях 1, 2, ..., q графа сетки. Нефиксированные вершины графа G могут располагаться в свободных узлах графа сетки произвольным образом. При этом суммарная длина $L(G)$ состоит из трех частей: $L_1(G)$, $L_2(G)$, $L_3(G)$, где $L_1(G)$ — суммарная длина ребер, смежных только фиксированным вершинам — определяется из способа построения матрицы геометрии $\mathbf{D}'_\Gamma$ — точное значение; $L_2(G)$ — суммарная длина ребер, связывающих фиксированные и нефиксированные вершины является нижней оценкой, поскольку учитывает длину связей при расположении нефиксированных вершин по стандартному графу; $L_3(G)$ — суммарная длина ребер, смежных только нефиксированным вершинам определяется как суммарная длина ребер стандартного графа, построенного на $n-q$ вершинах, и оставшихся неучтенных ребер графа.

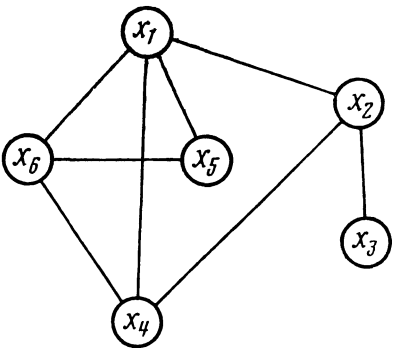


Рис. 3.38. Пример графа G для размещения в линейке

Заметим, что фиксация вершин при размещении графа может лишь увеличить или оставить без изменения суммарную длину по сравнению с нижней оценкой, полученной по стандартному графу. Кроме того, увеличение числа фиксированных вершин также не уменьшает суммарной длины, поскольку число вершин, входящих в стандартный граф уменьшается.

Рассмотрим пример размещения графа G, показанного на рис. 3.38, в линейку G_Γ (рис. 3.39,*а*). Определим нижнюю границу суммарной длины по стандартному графу G_Δ, показанному на рис. 3.39,*б*. Получим $\Theta(Q) = L(G_\Delta) = 11$. Разобьем множество решений Q на n подмножеств, фиксируя поочередно вершины графа G_1 на первой позиции графа G_Γ. Для каждого подмножества решений Q^1_1; Q^1_2; Q^1_3; Q^1_4; Q^1_5; Q^1_6 определим суммарную длину, предположив, что Q^1_1 получено фиксированием x_1 в первой позиции, Q^1_2 — фиксированием x_2 также в первой позиции и т. д. Получим $\Theta(Q^1_1) = L_1(G) + L_2(G) + L_3(G) = 0 + 10 + 4 = 14$; $\Theta(Q^1_2) = 12$; $\Theta(Q^1_3) = 11$; $\Theta(Q^1_4) = 12$; $\Theta(Q^1_5) = 11$; $\Theta(Q^1_6) = 12$. Для дальнейшего ветвления выберем подмножество Q^1_3, как имеющее наименьшую оценку $\Theta(Q^1_3)$ среди всех $\Theta(Q^1_i)$, $i = 1, 2, \ldots, n$, и произведем его разбиение на пять подмножеств $Q^1_{3,1}$, $Q^1_{3,2}$, $Q^1_{3,3}$, $Q^1_{3,4}$, $Q^1_{3,5}$ (рис. 3.40). Вершины Q^1_1, Q^1_2, Q^1_4, Q^1_5, Q^1_6, $Q^1_{3,1}$, $Q^1_{3,2}$, $Q^1_{3,3}$, $Q^1_{3,4}$ $Q^1_{3,5}$, не подвергавшиеся ветвлению, переобозначим соответственно Q^2_1, Q^2_2, Q^2_3, Q^2_4, Q^2_5, Q^2_6, Q^2_7, Q^2_8, Q^2_9, Q^2_{10}. Вершины Q^2_6, Q^2_7, Q^2_8, Q^2_9, Q^2_{10} получены фиксацией оставшихся вершин графа (кроме x_3) на второй позиции сетки. Определим ранее не находившиеся грани-

цы второго шага $\Theta(Q^2_6)=15$; $\Theta(Q^2_7)=11$; $\Theta(Q^2_8)=13$; $\Theta(Q^2_9)=12$; $\Theta(Q^2_{10})=13$. Определим, что ветвлению подлежит вершина Q^2_7, и произведем ее разбиение на четыре подмножества (рис. 3.41). Продолжив аналогично, получим дерево решений, показанное на рис. 3.42. Получили размещение, соответствующее подстановке

$$t=\begin{pmatrix}1 & 2 & 3 & 4 & 5 & 6\\ x_3 & x_2 & x_4 & x_1 & x_6 & x_5\end{pmatrix}.$$

Граф, размещенный в G_r с минимальной суммарной длиной $L(G)=11$, показан на рис. 3.43.

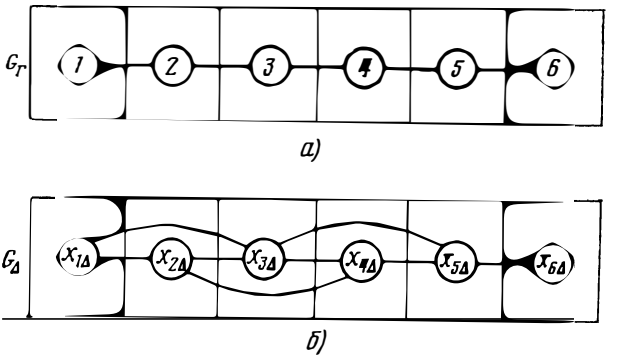


Рис. 3.39. Линейка G_r (*а*), стандартный граф G_Δ (*б*)

Заметим, что, используя понятие стандартного графа и операцию факторизации множества вершин графа G по критерию принадлежности к строкам или столбцам графа G_r, можно находить методом ветвей и границ минимум суммарной длины ребер графа G и при двумерном размещении на плоскости.

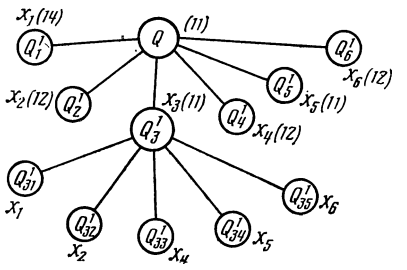


Рис. 3.40. Пример ветвления вершины Q^1_3

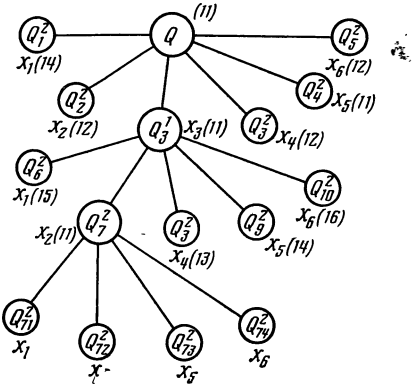


Рис. 3.41. Пример ветвления вершины Q^2_7

Рассмотрим теперь размещение в линейке не элементов, а фрагментов цепей на основе метода ветвей и границ для фрагмента схемы и ее графа (см. рис. 3.36, 3.37). Будем считать, что $L(G_\Delta)=17$. Разобьем множество решений

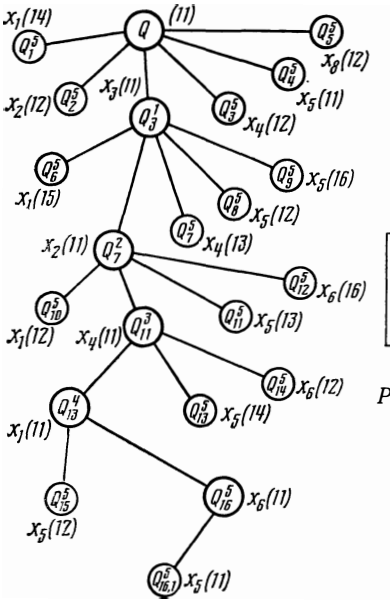


Рис. 3.42. Дерево решений

Q на s подмножеств, причем $s=2\cdot c^2_n$. Далее будем поочередно фиксировать ребра графа схемы в первых двух позициях решетки. В нашем случае в первые две позиции сетки будем последовательно помещать вершины x_1, x_2; x_1, x_3; ..., пока не рассмотрим все 30 вариантов. Для каждого подмножества

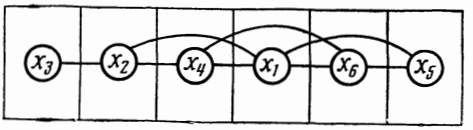


Рис. 3.43. Оптимальное размещение графа (см. рис. 3.38)

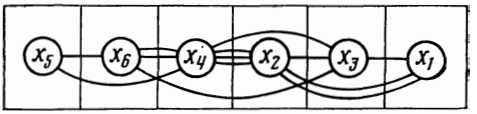


Рис. 3.44. Окончательное размещение графа на рис. 3.37 в линейке

решений Q^1_1, Q^1_2, ..., Q^1_{30} определим суммарную длину как нижнюю оценку, предполагая, что Q^1_1 получено фиксированием вершин (x_1, x_2) в первой и второй позициях; Q^1_2 — фиксированием (x_1, x_3); ...; Q^1_{30} — фиксированием (x_5, x_6). Для большей ясности вместо Q^1_1, ..., Q^{30}_1 будем писать $Q^1_{x_1 x_2}, \ldots, Q^{30}_{x_5 x_6}$.

После фиксации вершин (x_1, x_2) в первой и второй позициях линейки получим

$$\Theta(Q^1_{x_1 x_2}) = L_1(G) + L_2(G) + L_3(G) = 2 + 8 + 8 = 18;$$

$$\Theta(Q^1_{x_1 x_3}) = 1 + 10 + 8 = 19; \quad \Theta(Q^1_{x_1 x_4}) = 0 + 22 + 3 = 25;$$

$$\Theta(Q^1_{x_1 x_5}) = 0 + 14 + 9 = 23; \quad \Theta(Q^1_{x_1 x_6}) = 0 + 19 + 6 = 25;$$

$$\Theta(Q^1_{x_2 x_1}) = 2 + 11 + 8 = 21; \quad \Theta(Q^1_{x_2 x_3}) = 11 + 8 + 4 = 23;$$

$$\Theta(Q^1_{x_2 x_4}) = 23; \quad \Theta(Q^1_{x_2 x_5}) = 27; \quad \Theta(Q^1_{x_2 x_6}) =$$

$$= 28; \quad \Theta(Q^1_{x_3 x_1}) = 18, \quad \Theta(Q^1_{x_3 x_2}) = 21; \quad \Theta(Q^1_{x_3 x_4}) = 24;$$

$$\Theta(Q^1_{x_3 x_6}) = 22, \quad \Theta(Q^1_{x_4 x_1}) = 28; \quad \Theta(Q^1_{x_4 x_2}) = 23; \quad \Theta(Q^1_{x_4 x_3}) = 27;$$

$$\Theta(Q^1_{x_4 x_5}) = 25; \quad \Theta(Q^1_{x_4 x_6}) = 22; \quad \Theta(Q^1_{x_5 x_1}) = 24;$$

$$\Theta(Q^1_{x_5 x_2}) = 24; \quad \Theta(Q^1_{x_5 x_3}) = 22; \quad \Theta(Q^1_{x_5 x_4}) = 20;$$

$$\boxed{\Theta(Q^1_{x_5 x_6}) = 17;} \quad \Theta(Q^1_{x_6 x_1}) = 28; \quad \Theta(Q^1_{x_6 x_2}) = 27;$$

$$\Theta(Q^1_{x_6 x_3}) = 22; \quad \Theta(Q^1_{x_6 x_4}) = 25; \quad \Theta(Q^1_{x_6 x_5}) = 19.$$

Выберем оценку $\Theta(Q^1_{x_5 x_6}) = 17$ и вершины $x_5 x_6$ поместим в первой и второй позициях линейки. Далее фиксируем оставшиеся вершины в третью позицию линейки и получим $\Theta(Q^2_{6,1}) = 25$; $\Theta(Q^2_{6,2}) = 25$; $\Theta(Q^2_{6,3}) = 20$; $\boxed{\Theta(Q^2_{6,4}) = 19}$.

Поэтому вершину x_4 поместим в третью позицию линейки. На третьем шаге $\Theta(Q^3_{4,1})=25$; $\overline{\Theta(Q^3_{4,2})=19}$; значит, вершину x_2 поместим в четвертую позицию линейки. На последнем шаге $\Theta(Q^4_{2,1})=20$; $\Theta(Q^4_{2,3})=19$ и вершину x_3 поместим в пятую позицию линейки. Последняя вершина x_1, естественно, попадает в шестую позицию линейки (рис. 3.44). Получим размещение графа в линейке $L(G)=19$. После перехода от графа к схеме получим окончательное размещение, показанное на рис. 3.45.

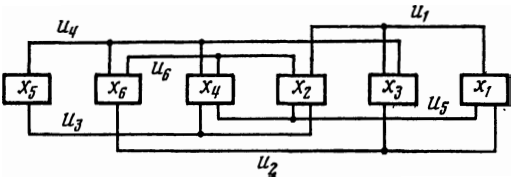


Рис. 3.45. Окончательное размещение фрагмента схемы на рис. 3.35

Отметим, что при размещении фрагментов цепей, т. е. ребер графа схемы, в отличие от вершин, появляется возможность прогнозировать размещение на шаг вперед, что повышает точность размещения. Результат окончательного размещения в значительной степени зависит от выбора метода ветвления и способа оценки минимальной суммарной длины соединений.

Рассмотрим минимизацию пересечений ребер графа схемы. При конструировании ЭА на интегральных микросхемах и печатных платах важно отыскать такое расположение схемы, в котором сведено до минимума число пересечений проводников. Для простоты изложения предварительно рассмотрим графы с гамильтоновым циклом, а затем результат распространим на случай произвольных графов.

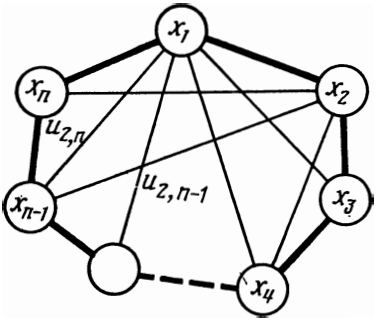


Рис. 3.46. Пересечение ребер внутри гамильтонова цикла

Пусть дан произвольный граф $G=(X, U)$ без петель и кратных ребер. Будем считать, что граф G имеет гамильтонов цикл, а его вершины соединяются кратчайшими ребрами. Два любых ребра $u_{i,j}\in U$ и $u_{k,l}\in U$ могут пересекаться друг с другом не более 1 раза; ребра графа, инцидентные одной вершине, не пересекаются между собой, а также ни одно ребро графа не пересекает ребер гамильтонова цикла.

Перейдем теперь к подсчету числа пересечений ребер графа G. Для этого будем располагать ребра графа G, не принадлежащие гамильтонову циклу внутри области, ограниченной этим циклом. Обозначим множество ребер, инцидентных вершине x_i, через U_i, а число пересечений, образованное ребрами этого множества, — через p_i. Очевидно, что при проведении ребер, инцидентных вершине $x_1\in X$, пересечения не образуются. В этом случае $p_1=0$. Проведем теперь ребра, инцидентные вершине $x_2\in X$, и

определим число пересечений этих ребер с ребрами множества U_1. Ребро $u_{2,n}$ пересекается со всеми ребрами, инцидентными вершине x_1 (рис. 3.46). В дальнейшем ребру $u_{s,t}$ сопоставляется элемент матрицы смежности $r_{s,t}$, где $s,\ t \in I = \{1, 2, ..., n\}$; n — число вершин графа.

Нетрудно видеть, что число пересечений, образованное ребрами $U_{2,n}$, определяется по формуле

$$p_{2,n} = r_{2,n} \sum_{i=3}^{n-1} r_{1,i}.$$

Ребро $u_{2,(n-1)}$ образует

$$p_{2,(n-1)} = r_{2,(n-1)} \sum_{i=3}^{n-2} r_{1,i}$$

пересечений. Аналогично легко определить число пересечений других ребер множества U_2 с ребрами множества U_1, которое будет иметь вид

$$p_{2,j} = r_{2,j} \sum_{i=3}^{j-1} r_{1,i}.$$

Общее число пересечений ребер, инцидентных вершине x_2,

$$P_2 = \sum_{j=4}^{n} p_{2,i}.$$

По индукции определим число пересечений ребер, инцидентных вершине x_k графа G,

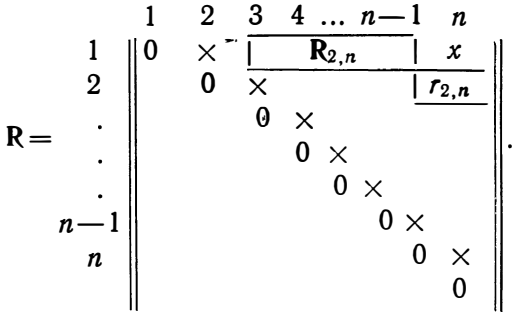
$$P_k = \sum_{j=k+2}^{n} p_{k,j}. \qquad (3.61)$$

Общее число пересечений всех ребер графа G определяется выражением

$$P = \sum_{k=2}^{n-2} P_k, \quad \text{т. е.} \quad P = \sum_{k=2}^{n-2} \sum_{j=k+2}^{n} \sum_{i=k+1}^{j-1} \sum_{m=1}^{k-1} r_{k,j} r_{m,i}. \qquad (3.62)$$

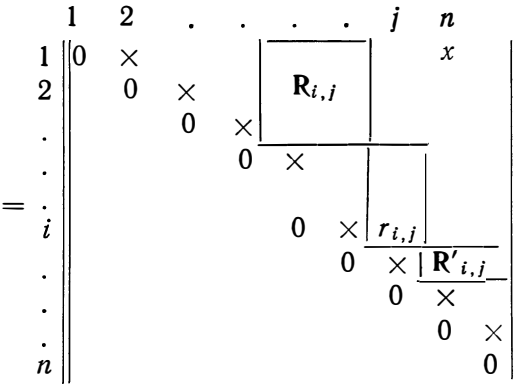
Непосредственный подсчет числа пересечений ребер графа по формуле (3.62) затруднителен, поэтому для облегчения этого процесса предлагается использовать матрицу смежности $\mathbf{R}$ графа G. Поскольку G — неориентированный граф, будем использовать треугольную матрицу, учитывая, что ребра гамильтонова цикла не образуют пересечений, единицы, соответствующие этим ребрам, заменены крестиками.

Для нахождения числа пересечений ребер графа G в соответствии с формулой (3.62) при задании графа матрицей смежности $\mathbf{R}$ поступим следующим образом. Сначала определим число $p_{2,n}$, для чего в матрице смежности $\mathbf{R}$ выделим подматрицу $\mathbf{R}_{2,n}$, которая имеет вид



Сумма единичных элементов подматрицы $\mathbf{R}_{2,n}$ соответствует числу $p_{2,n}$. Далее определим число $p_{2,(n-1)}$ путем выделения подматрицы $\mathbf{R}_{2,(n-1)}$ и подсчета суммарного числа единиц в ней и т. д. Число пересечений ребер, инцидентных вершине x_2, с ребрами, инцидентными вершине x_1, определяется суммой единиц во всех $\mathbf{R}_{2,i}$. Процесс выделения и подсчета единиц в подматрицах продолжается аналогично до тех пор, пока не будет определено общее число пересечений графа G как сумма единиц всех подматриц от $\mathbf{R}_{2,n}$ до $\mathbf{R}_{(n-2),n}$.

В том случае, когда необходимо не только подсчитывать число пересечений ребер графа G, но и знать число пересечений какого-нибудь ребра $u_{i,j}$, со всеми остальными ребрами графа будем выделять в матрице $\mathbf{R}$ две подматрицы $\mathbf{R}_{i,j}$ и $\mathbf{R}'_{i,j}$



Покажем, что ребро $u_{i,j}$ графа G пересекается со всеми ребрами, которым соответствуют единицы в выделенных подматрицах $\mathbf{R}_{i,j}$ и $\mathbf{R}'_{i,j}$. Рассмотрим ребро $u_{i,j}$ графа G. Относительно него можно выделить два множества вершин графа X_1 и X_2: $X_1=\{x_{j+1}, x_{j+2}, \ldots, x_{i-1}\}$; $x_2=\{x_{i+1}, x_{i+2}, \ldots, x_{j-1}\}$.

Согласно теореме Жордана с ребром $u_{i,j}$ могут пересекаться только ребра, одновременно инцидентные вершинам, находящимся в X_1 и X_2, т. е. $x_k \in X_1$, $x_l \in X_2$ или $x_l \in X_1$, $x_k \in X_2$. Единицы в матрице смежности, соответствующие ребрам, как легко видеть, располагаются в подматрицах $\mathbf{R}_{i,j}$ и $\mathbf{R}'_{i,j}$. Таким образом, суммарное

число единиц подматриц $\mathbf{R}_{i,j}$ и $\mathbf{R}'_{i,j}$ определяет число пересечений ребра $u_{i,j}$ со всеми остальными ребрами графа G.

На примере графа, показанного на рис. 3.47, определим число пересечений ребра $u_{3,6}$. Для этого запишем матрицу смежности

$$\mathbf{R}=\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 0 & \times & 1 & 1 & 0 & 1 & \times \\ 2 & & 0 & \times & 0 & 1 & 0 & 1 \\ 3 & & & 0 & \times & 0 & 1 & 0 \\ 4 & & & & 0 & \times & 0 & 1 \\ 5 & & & & & 0 & \times & 0 \\ 6 & & & & & & 0 & \times \\ 7 & & & & & & & 0 \end{array},$$

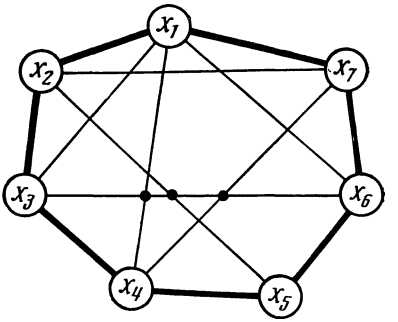


Рис. 3.47. Пример графа для определения пересечений

по которой определим, что число единиц в выделенных подматрицах равно трем. Это значит, что ребро $u_{3,6}$ имеет три пересечения. Если для каждого ребра подсчитать число пересечений и записать это число в соответствующие клетки матрицы смежности, то получим так называемую матрицу пересечений $\mathbf{R}_p$ графа G. Заметим, что каждое пересечение образуется наложением двух ребер, поэтому для подсчета общего числа пересечений ребер графа G необходимо просуммировать все элементы матрицы $\mathbf{R}_p$ и сумму разделить на 2.

Данный метод подсчета числа пересечений можно распространить и на графы без гамильтонова цикла. Это следует из того, что ребра последнего не образуют пересечений и матрица пересечений $\mathbf{R}_p$ при отсутствии части или всех ребер выделенного гамильтонова цикла не меняет своего вида. Вершины располагаются так же, как и в случае графа, имеющего гамильтонов цикл.

Выше рассматривались графы, у которых ребра находились внутри области, ограниченной гамильтоновым циклом. Естественно предложить расположение ребер графа не только во внутренней, но и во внешней области. Для определения числа пересечений ребер во внутренней и внешней областях графа G целесообразно составить соответствующие матрицы смежности $\mathbf{R}_1$ и $\mathbf{R}_2$ такие, что $\mathbf{R}=\mathbf{R}_1 \oplus \mathbf{R}_2$, т. е. для всех элементов матрицы смежности $\mathbf{R}$ выполняется условие $r_{i,j}=r_{1,i,j} \oplus r_{2,i,j}$, где $\oplus$ — сумма по модулю 2.

При расположении ребер графа G в двух областях формула (3.62) имеет вид

$$p=\sum_{k=2}^{n-2}\sum_{j=k+2}^{n}\sum_{i=k+1}^{j-1}\sum_{m=1}^{k-1} r_{1,k,j}r_{1,m,i} + \\ +\sum_{k=2}^{n-2}\sum_{j=k+2}^{n}\sum_{i=k+1}^{j-1}\sum_{m=1}^{k-1} r_{2,k,j}r_{2mi}.$$

Отметим, что данный метод подсчета числа пересечений можно применять и к графам, имеющим петли и кратные ребра. Тогда в матрице смежности вместо единиц будут стоять кратности ре-

бер. Введение петель в граф не меняет числа пересечений, так как любую петлю можно провести без пересечений.

Рассмотрим теперь вопрос подсчета числа пересечений ребер при размещении вершин графа схемы в узлах прямоугольной сетки G_r. Пусть задан некоторый мультиграф $G=(X, U)$, отображенный в сетку G_r, и необходимо найти значение функции $p(G)$.

Для однозначности решения поставленной задачи условимся в единообразии проведения ребер графа G. Будем считать, что вершины x_i и x_j соединяются ребром, которое проводится от вершины x_i по соответствующей строке до столбца, в котором находится вершина x_j. Такое проведение ребра справедливо при $i<j$. Если $i>j$, то ребро $u_{i,j}$ ведется по столбцу, соответствующему вершине x_j до строки, в которой находится вершина x_i, далее по строке к x_j. Все ребра, оба конца которых принадлежат одному столбцу, проводятся слева от соответствующего столбца, а ребра, оба конца которых принадлежат одной строке, проводятся над соответствующей строкой.

Идея нахождения числа пересечений $p(G)$ ребер графа G, отображенного в G_r, заключается в выделении относительно каждого ребра некоторых подмножеств вершин X_k, $X_l \subset X$.

Рассмотрим произвольное ребро $u_{p,q}$ и выделим относительно него подмножества вершин X_k и X_l, которые назовем «подмножествами пересечений». Функция пересечений двух ребер $u_{i,j}$ и $u_{p,q}$ может принимать два значения:

$$p(u_{i,j}, u_{p,q}) = \begin{cases} 1, \text{ если ребра } u_{i,j} \text{ и } u_{p,q} \text{ пересекаются;} \\ 0 \text{ в противном случае.} \end{cases}$$

Тогда если $(\forall u_{i,j} \in U_{k,l})\,(P(u_{i,j}, u_{p,q})=1$, то подмножества вершин X_k и X_l будут подмножествами пересечений. Здесь $U_{k,l}$ — множество ребер, инцидентных как X_k, так и X_l. Очевидно, что $X_k \cap X_l = \varnothing$, а $X_k \cup X_l = H$, где $H \subset X$.

Разобьем все множество ребер графа G на три подмножества U_1, U_2, U_3 $(U_1 \cup U_2 \cup U_3 = U)$ и выделим относительно каждого из них подмножества пересечений. Рассмотрим произвольное ребро $u_{i,j}$. Пусть вершина x_i находится в узле сетки с координатами $s=\pi$, $t=q$, а x_j — вершины в узле с координатами $s=k$, $t=l$. Будем считать, что ребро $u_{i,j} \in U_1$, если $q=l$, $u_{i,j} \in U_2$, если $\pi \leqslant k$, $u_{i,j} \in U_3$, если $\pi > k$. Подсчет функции пересечений начнем с ребер, инцидентных вершине x_1.

После определения числа пересечений ребер, инцидентных вершине x_1, вершина x_1 отбрасывается и рассматривается вершина x_2 и т. д. Заметим, что на каждом шаге определения функции пересечения вершины x_i в подмножества пересечений будут входить вершины x_j, для которых $j \in \mathcal{J} = \{i+1, i+2, \ldots, n\}$. Тогда относительно ребер подмножества U_1 получим подмножества пересечений X_1 и X_2, для которых выполняются условия

$$(\forall x_r \in X_1)\ (\pi < s < k,\ t = q = l);$$
$$(\forall x_m \in X_2)\ (s > k,\ t = q = l), \qquad (3.63)$$

где s, t — текущие координаты вершин.

Число ребер K_{x_1, x_2} между подмножествами X_1 и X_2 будет равно числу пересечения рассматриваемого ребра $u_{i,j}$, т. е. $p(u_{i,j}) = k_{x_1, x_2}$. Подмножества пересечений X_1 и X_2 для ребра (x_i, x_j) изображены на рис. 3.48.

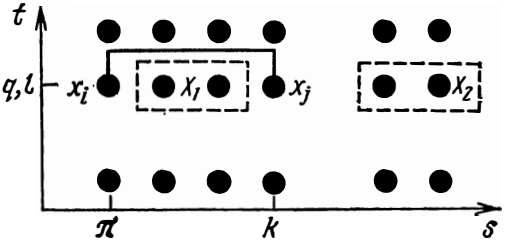


Рис. 3.48. Подмножество пересечений X_1, X_2

Относительно ребер, принадлежащих к U_2 и U_3, подмножества пересечений можно выделить по-разному, однако число пересечений от этого не изменится. Поэтому предлагается для ребер $u_{i,j} \in U_2$ выделять следующие подмножества пересечений:

$$(\forall x_r \in X_1)\ (s \geqslant k;\ l < t \leqslant q)\ (\forall x_m \in X_2)\ (s < k;\ t < q);$$
$$(\forall x_r \in X_3)\ (s < k;\ q > t > l)\ (\forall x_m \in X_4)\ (s > k;\ t \leqslant l);$$
$$(\forall x_r \in X_5)\ (\pi \leqslant s < k;\ t = l)\ (\forall x_m \in X_6)\ (s > k;\ t = l);$$
$$(\forall x_r \in X_7)\ (k \geqslant s > \pi;\ t = q)\ (\forall x_m \in X_8)\ (s = k;\ t < l);$$
$$(\forall x_r \in X_9)\ (s = k;\ l < t < q)\ (\forall x_m \in X_{10})\ (s = k;\ t < l). \qquad (3.64)$$

На рис. 3.49 изображены подмножества пересечений для ребер подмножества U_2. Аналогичная процедура выполняется для $u_{i,j} \in U_3$:

$$(\forall x_r \in X_1)\ (s > k;\ q \geqslant t > l)\ (\forall x_m \in X_2)\ (s \leqslant k;\ t < q);$$
$$(\forall x_r \in X_3)\ (s \leqslant k;\ l < t < q)\ (\forall x_m \in X_4)\ (s > k;\ t \leqslant l); \qquad (3.65)$$
$$(\forall x_r \in X_5)\ (s < k;\ t = l)\ (\forall x_m \in X_6)\ (s > k;\ t = l).$$

Тогда функция пересечения для $u_{i,j} \in U_2$ $p(u_{i,j}) = k_{x_1, x_2} + k_{x_3, x_4} + k_{x_5, x_6} + k_{x_7, x_8} + k_{x_9, x_{10}}$, а для $u_{i,j} \in U_3$ $p(u_{i,j}) = k_{x_1, x_2} + k_{x_3, x_4} + k_{x_5, x_6}$. На рис. 3.50 показаны подмножества пересечений для ребер $u_{i,j} \in U_3$.

Информация о графе задается матрицей смежности **R** или ее списком.

Для графа, изображенного на рис. 3.51, покажем определение функции пересечений $p(G)$:

$$\mathbf{R} = \left\| \begin{array}{lll} x_1(x_3, x_5, x_6, x_9, x_{10}) & x_7(x_{16}) & x_{13}(x_{14}) \\ x_2(x_4, x_6) & x_8(x_{11}, x_{12}, x_{16}) & x_{14}(x_{15}) \\ x_3(x_{14}, x_4, x_{15}) & x_9(x_{12}, x_{13}) & x_{15}(x_{16}) \\ x_4(x_7, x_8, x_{12}) & x_{10}(x_{13}, x_{14}) & x_{16}(\varnothing) \\ x_5(x_7, x_6, x_9) & x_{11}(x_{15}, x_{16}) & \\ x_6(x_{13}, x_9) & x_{12}(\varnothing) & \end{array} \right\|.$$

Выделим первое ребро $u_{1,3}$. Найдем координаты вершин x_1 ($\pi=1$, $q=1$), x_3 ($k=3$, $l=1$). Ребро $u_{1,3}\in U_1$, поэтому подмножества пересечений необходимо выделить по условию (3.63): $X_1=\{x_2\}$, $X_2=\{x_4\}$, $p(u_{1,3})=k_{x_1x_2}=1$. Выделим следующее ребро $u_{1,10}$ и определим координаты вершин x_1 ($\pi=1$, $q=1$), x_{10} ($k=2$, $l=3$). Ребро $u_{1,10}\in U_2$, поэтому для выделения подмножества пересе-

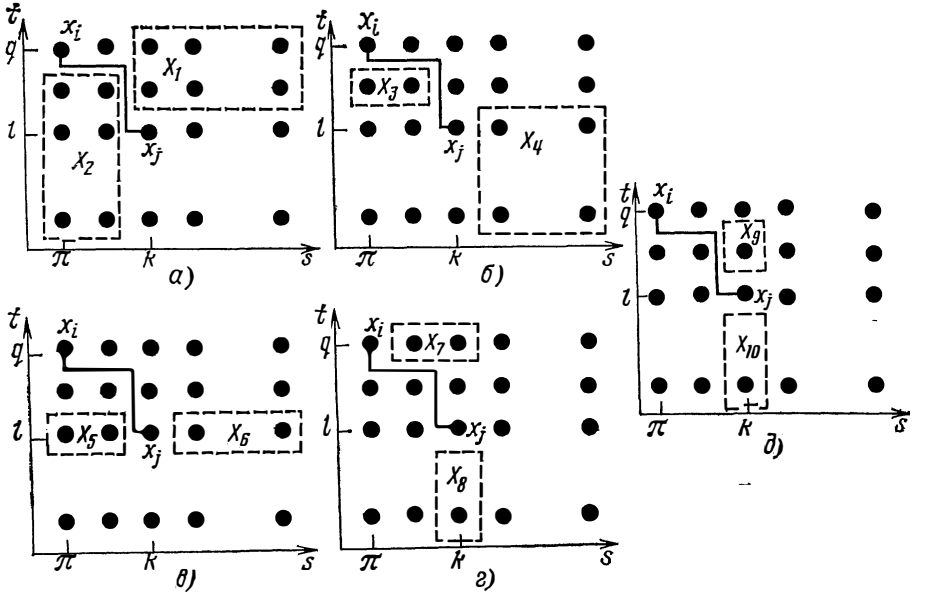


Рис. 3.49. Подмножества пересечений для ребер U_2

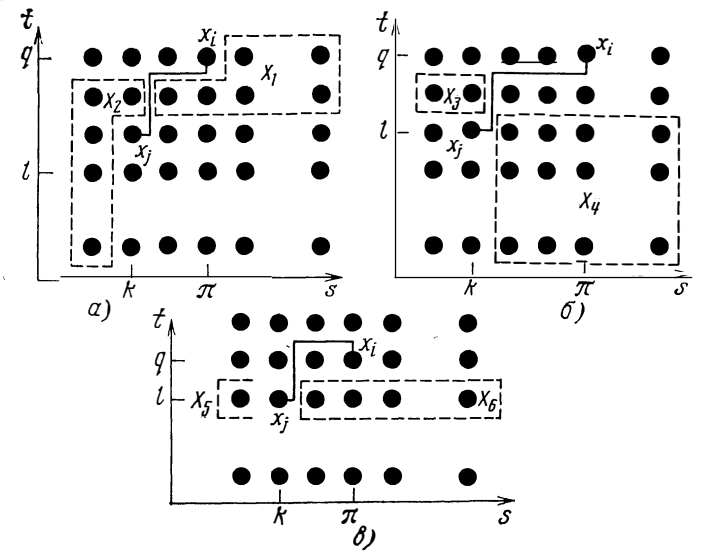


Рис. 3.50. Подмножества пересечений для ребер U_3

чений используем условие (3.64): $X_1=\{x_2, x_3, x_4, x_6, x_7, x_8\}$; $X_5=\{x_9\}$; $X_9=\{x_6\}$; $X_2=\{x_5, x_9, x_{13}\}$; $X_6=\{x_{11}, x_{12}\}$; $X_{10}=\{x_{14}\}$; $X_3=\{x_5\}$; $X_7=\{x_2\}$; $X_4=\{x_{11}, x_{12}, x_{15}, x_{16}\}$; $X_8=\{x_{14}\}$;

$$p=(u_{1,10})=k_{x_1,x_2}+k_{x_3,\ x_4}+k_{x_5,\ x_6}+k_{x_7,\ x_8}+k_{x_9,x_{10}}=4.$$

Аналогичным образом, выделяя подмножества пересечений для остальных ребер. Получаем, что значение функции пересечений ребер графа G равно $p(G)=19$.

Рассмотрим два метода минимизации числа пересечений ребер произвольных графов. В первом методе в графе сначала определяется такое ребро $u_{i,j}$, которое имеет наибольшее число пересечений. Через это ребро проводится линия (секущая ось), которая делит плоскость на две части (полуплоскости N_1 и N_2). Относительно секущей оси исходный граф разбивается на две части: G_1 и G_2. Часть ребер графа G будет принадлежать как G_1, так и G_2. Тогда если $G_1=(X_1, U_1)$; $G_2=(X_2, U_2)$, то вершины из X_1 лежат в полуплоскости N_1, а вершины из X_2 — в N_2. Заметим, что $X_1 \cap X_2=\varnothing$, а $U_1 \cup U_2=U_{i,j}$. Множество ребер $U_{i,j}$ является соединяющим для полуплоскостей N_1 и N_2. Часть ребер множества $U_{i,j}$ пересекает выбранное ребро $u_{i,j}$. Для устранения пересечений этого ребра часть G_1 переносим из полуплоскости N_1 в полуплоскость N_2, а именно в ту ее область, которая «прилегает» к секущей оси.

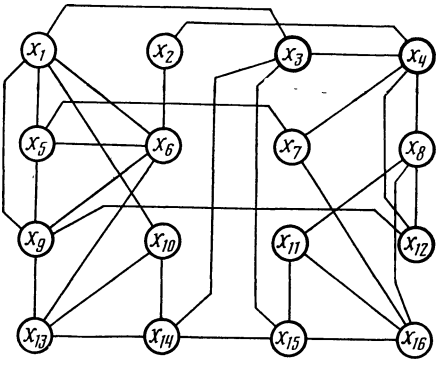


Рис. 3.51. Пример графа для определения функций пересечений

При этом необходимо выбирать такие ребра $u_{k,l}$, для которых разница между числом пересечений $u_{i,j}$ и числом пересечений ребер множества $U_{i,j}$, образующихся после переноса G_1 в полуплоскость N_2, имеет положительную величину. Далее в $G=(X, U)$ выбирается следующее ребро, имеющее наибольшее число пересечений, проводится новая секущая ось и вышеописанный процесс повторяется. При достаточно большом числе проведения таких операций можно добиться существенной минимизации числа пересечений ребер графа. Однако практически можно ограничиться сравнительно небольшим числом шагов после значительного уменьшения числа пересечений ребер по сравнению с исходным.

Пусть дан граф $G=(X, U)$, имеющий 10 вершин и 16 пересечений, показанный на рис. 3.52,*а*. Выберем ребро $u_{3,6}$, так как оно имеет максимальное число пересечений. Тогда после рассечения графа определим G_1 и G_2: $G_1=(X_1U_1)$, где $X_1=\{x_1, x_2\}$, и $G_2=(X_2, U_2)$, где $X_2=\{x_4, x_5, x_7, x_8, x_9, x_{10}\}$; $U_1 \cap U_2=U_{1,2}=\{u_{1,8}, u_{2,8}, u_{1,5}, u_{2,7}, u_{2,4}\}$.

После переноса G_1 в полуплоскость N_2 получаем граф, показанный на рис. 3.52,*б*. Затем секущая ось проводится через ребро $u_{3,9}$, и вновь полученная часть $G^1{}_1$ переносится в полуплоскость $N^1{}_2$. Окончательный результат работы

алгоритма показан на рис. 3.52,*в*. Число пересечений ребер графа уменьшилось с 16 до 1. На рис. 3.52,*б*, *в* ребро $u_{2,7}$ не показано.

Опишем теперь алгоритм минимизации числа пересечений ребер графа схемы, основанный на расположении ребер графа во внутренней и внешней областях гамильтонова цикла. Очевидно, что исследование графа с гамильтоновым циклом не уменьшает общности предлагаемого метода, так как отсутствие ребер цикла после применения алгоритма минимизации числа пересечений ребер позволит только уменьшить число пересечений за счет проведения некоторых ребер графа в том месте, где отсутствуют ребра гамильтонова цикла.

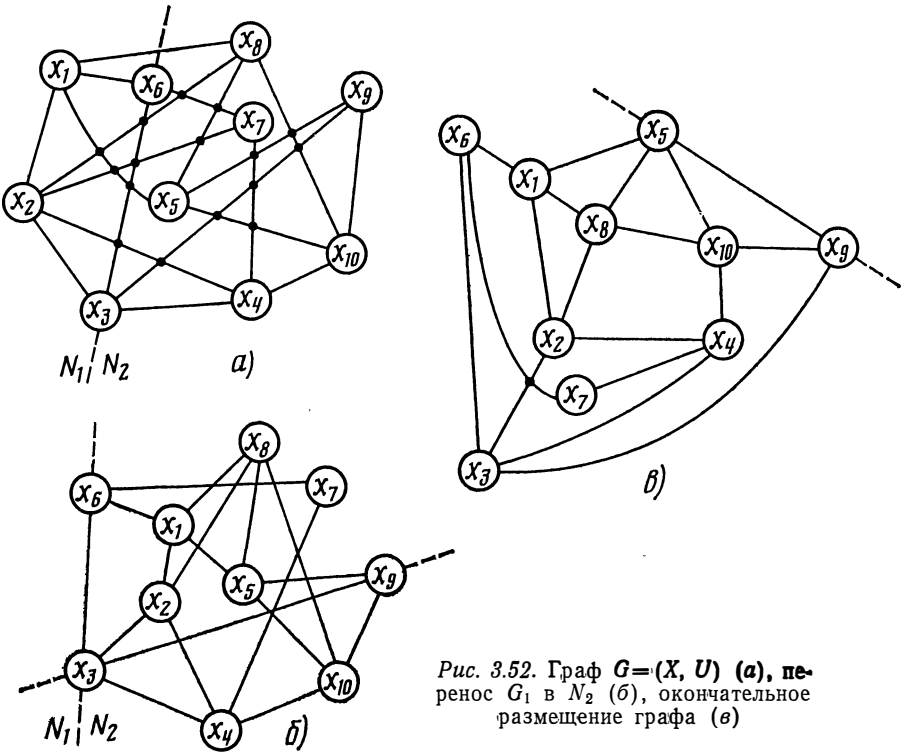


Рис. 3.52. Граф $G=(X, U)$ (*а*), перенос G_1 в N_2 (*б*), окончательное размещение графа (*в*)

При представлении каждого элемента схемы вершиной графа не учитываются пересечения проводников, соединенных с одним и тем же элементом. Однако число таких пересечений трудно поддается минимизации с помощью ЭВМ, кроме того, оно обычно незначительно по сравнению с общим числом пересечений ребер графа G. Основная идея алгоритма минимизации числа пересечений внутрисхемных ребер заключается в следующем. По исходной схеме строится мультиграф $G=(X, U)$, а затем он располагается на плоскости с выделенным

гамильтоновым циклом и по нему записывается видоизмененная матрица смежности $\mathbf{R}_1$. По матрице находятся «отклонения» для каждого ребра графа G, а затем строится матрица отклонений ребер $\mathbf{R}_\Delta$. После этого из матрицы $\mathbf{R}_\Delta$ выбирается ребро, для которого отклонение на данном шаге имеет максимальное положительное значение. Далее это ребро выносится во внешнюю область цикла. Указанный процесс продолжается аналогично до тех пор, пока все элементы матрицы $\mathbf{R}_\Delta$ не станут отрицательными или равными нулю. В конце работы алгоритма производится обратный переход от графа к схеме.

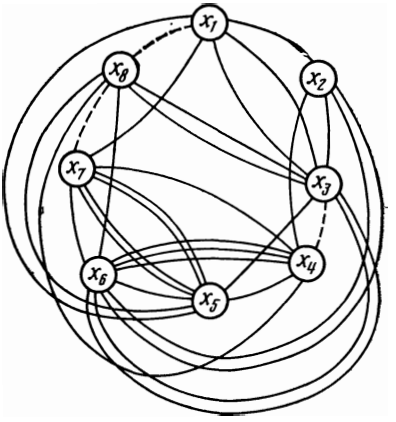


Рис. 3.53. Граф G_1

Рассмотрим более подробно процесс построения матрицы $\mathbf{R}_\Delta$ и этапы алгоритма минимизации общего числа пересечений ребер графа. Под отклонением $\Delta_{i,j}$ ребра $u_{i,j}$ графа G будем понимать разность между числом его пересечений со всеми остальными ребрами графа, когда оно расположено во внутренней области, и числом пересечений этого же ребра, проведенного во внешней области цикла.

Пусть, например, дан граф G_1, изображенный на рис. 3.53. Для ребра $u_{3,8}$, мультичисло которого равно двум, найдем значение отклонения $\Delta_{3,8}$. Построим сначала видоизмененные матрицы смежности $\overset{\circ}{\mathbf{R}}_1$ и $\overline{\mathbf{R}_1}$ соответственно для внутренней и внешней областей графа G_1 (в матрицах $\overset{\circ}{\mathbf{R}}_1$ и $\overline{\mathbf{R}_1}$ крестиками отмечены ребра гамильтонова псевдоцикла, а черточками — отсутствие ребер):

$$\overset{\circ}{\mathbf{R}}_1 = \begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 0 & \times & 2 & 0 & 0 & 0 & 1 & — \\ 2 & & 0 & \times & 1 & 0 & 0 & 0 & 0 \\ 3 & & & 0 & — & 1 & 0 & 0 & 2 \\ 4 & & & & 0 & \times & 3 & 1 & 0 \\ 5 & & & & & 0 & \times & 4 & 0 \\ 6 & & & & & & 0 & \times & 1 \\ 7 & & & & & & & 0 & — \\ 8 & & & & & & & & 0 \end{array}; \quad \overline{\mathbf{R}_1} = \begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 0 & \times & 0 & 0 & 1 & 0 & 0 & — \\ 2 & & 0 & \times & 0 & 0 & 2 & 0 & 0 \\ 3 & & & 0 & — & 0 & 2 & 0 & 0 \\ 4 & & & & 0 & \times & 0 & 0 & 1 \\ 5 & & & & & 0 & \times & 0 & 1 \\ 6 & & & & & & 0 & \times & 0 \\ 7 & & & & & & & 0 & — \\ 8 & & & & & & & & 0 \end{array}.$$

Далее по алгоритму, описанному выше, легко определить число пересечений ребра $u_{3,8}$, когда оно находится внутри гамильтонова псевдоцикла, подсчитав сумму элементов в выделенной подматрице пересечений в $\overset{\circ}{\mathbf{R}}_1$ и умножив ее на мультичисло ребра $u_{3,8}$. Аналогично число пересечений ребра $u_{3,8}$ при помещении его во внешнюю область определяется из подматрицы пересечений, вы-

деленной в $\overline{\mathbf{R}}_1$. Итак, запишем число пересечений для ребра $u_{3,8}$ во внутренней области

$$\overset{\circ}{p}_{3,8}=r_{3,8}\sum_{r_{k,l}\in\overset{\circ}{R}_{i,j}} r_{k,l}=2(1+1)=4; \tag{3.66}$$

во внешней области

$$\overline{\overline{p}}_{3,8}=r_{3,8}\sum_{r_{k,l}\in\overline{R}_{i,j}} r_{k,l}=2(1+2)=6. \tag{3.67}$$

Здесь $\sum_{r_{k,l}\in\mathbf{R}_{i,j}} r_{k,l}$ — сумма элементов в подматрицах пересечений, выделенных в матрице смежности.

Отклонение ребра $u_{3,8}$ определится теперь как $\Delta_{3,8}=\overset{\circ}{p}_{3,8}-\bar{p}_{3,8}=-2$. Это означает, что при переносе ребра $u_{3,8}$ из внутренней области во внешнюю общее число пересечений графа увеличится на 2.

Матрица отклонений ребер $\mathbf{R}_\Delta$ получается путем замены элементов матрицы смежности $\mathbf{R}$ на соответствующие значения отклонений ребер. Заметим, что на первом шаге выполнения алгоритма все элементы матрицы $\overline{\mathbf{R}}_i$ равны нулю, а матрица $\overset{\circ}{\mathbf{R}}_i$ совпадает с матрицей смежности $\mathbf{R}_i$ графа G_i, так как все ребра по условию проводятся во внутренней области. Следовательно, на первом шаге $\Delta_{i,j}=p_{i,j}$ и матрица R_Δ строится путем замены элементов матрицы $\mathbf{R}$ на соответствующие им значения пересечений ребер.

Рассмотрим на примере минимизацию числа пересечений в графе, изображенном на рис. 3.54,*а*. При таком расположении графа число пересечений $p(G)=69$. Определив для каждого ребра графа по матрице смежности $\mathbf{R}_1$ значение отклонения, как показано выше, строим матрицу отклонений ребер

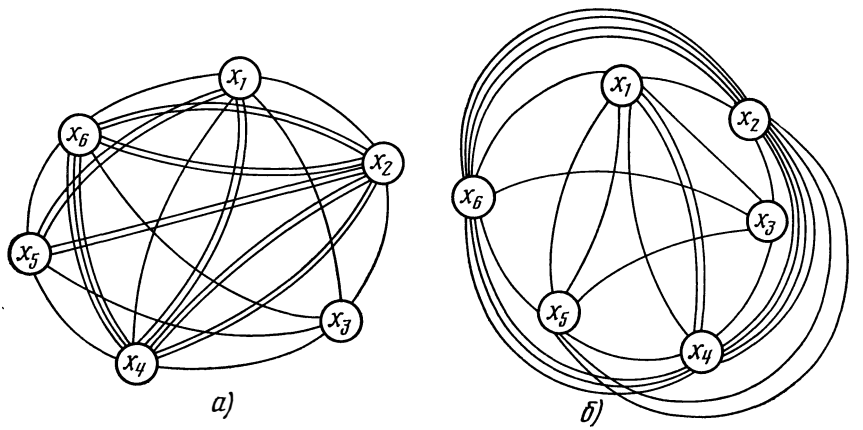


Рис. 3.54. Пример графа для минимизации числа пересечений (*а*), окончательное размещение графа (*б*)

$$\mathbf{R}_\Delta = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & \times & 10 & 24 & 16 & \times \\ 2 & & 0 & \times & 12 & 16 & \boxed{24} \\ 3 & & & 0 & \times & 10 & 11 \\ 4 & & & & 0 & \times & 15 \\ 5 & & & & & 0 & \times \\ 6 & & & & & & 0 \end{array}.$$

По матрице $\mathbf{R}_\Delta$ определим элемент с максимальным положительным отклонением. Если таких элементов несколько, то выбираем любой из них и ребро (несколько ребер) графа G, соответствующее определенному отклонению, вынесем во внешнюю область гамильтонова цикла. В нашем случае таким ребром является $u_{2,6}$. После этого матрицы смежности для внутренней и внешней областей графа примут вид

$$\overset{\circ}{\mathbf{R}}_1 = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & \times & 1 & 3 & 2 & \times \\ 2 & & 0 & \times & 4 & 2 & 0 \\ 3 & & & 0 & \times & 1 & 1 \\ 4 & & & & 0 & \times & 3 \\ 5 & & & & & 0 & \times \\ 6 & & & & & & 0 \end{array}; \quad \bar{\mathbf{R}}_1 = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & \times & 0 & 0 & 0 & \times \\ 2 & & 0 & \times & 0 & 0 & 4 \\ 3 & & & 0 & \times & 0 & 0 \\ 4 & & & & 0 & \times & 0 \\ 5 & & & & & 0 & \times \\ 6 & & & & & & 0 \end{array},$$

а матрица отклонений ребер после первого шага запишется в виде

$$\mathbf{R}^1_\Delta = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & \times & 2 & 0 & 0 & \times \\ 2 & & 0 & \times & 12 & \boxed{16} & - \\ 3 & & & 0 & \times & 10 & 11 \\ 4 & & & & 0 & \times & 15 \\ 5 & & & & & 0 & \times \\ 6 & & & & & & 0 \end{array}.$$

Прочерк в матрице $\mathbf{R}^1_\Delta$ соответствует вынесенному ребру $u_{2,6}$. Заметим, что в матрице $\mathbf{R}^1_\Delta$ по сравнению с $\mathbf{R}_\Delta$ изменились значения тех элементов, соответствующие которым ребра пересекаются с ребром $u_{2,6}$. На втором шаге аналогичная процедура выполняется для ребра $u_{2,5}$, имеющего максимальное отклонение. После этого алгоритм выполняется для ребер $u_{2,4}$ и $u_{4,6}$, и на этом процесс минимизации числа пересечений ребер графа заканчивается. Для окончательного варианта расположения графа ребра $u_{1,3}$, $u_{1,4}$, $u_{1,5}$, $u_{3,5}$ и $u_{3,6}$ находятся во внутренней, а ребра $u_{2,4}$, $u_{2,5}$, $u_{2,6}$ и $u_{4,6}$ — во внешней областях гамильтонова цикла. Матрица отклонений примет вид

$$\mathbf{R}^4_\Delta = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & \times & -10 & -12 & -12 & \times \\ 2 & & 0 & \times & - & -6 & - \\ 3 & & & 0 & \times & -4 & -1 \\ 4 & & & & \mathbf{0} & \times & \mathbf{-6} \\ 5 & & & & & 0 & \times \\ 6 & & & & & & 0 \end{array}.$$

Число пересечений ребер графа уменьшилось с 69 до 14. Изображение графа G на плоскости после выполнения алгоритма показано на рис. 3.54,*б*.

Отметим, что при минимизации числа пересечений ребер графа нет необходимости строить на каждом шаге матрицы $\overset{\circ}{\mathbf{R}}_i$ и $\overline{\mathbf{R}}_i$ и подсчитывать отклонение ребер, как показано выше для наглядности метода. Для этого достаточно использовать только матрицу смежности $\mathbf{R}_i$ и матрицу отклонений ребер, полученную на первом шаге алгоритма. Начиная со второго шага, новые значения матрицы $\mathbf{R}^j{}_\Delta$ определяются выражением

$$\Delta'_{p,\,q}=\Delta_{p,\,q}-2r_{p,\,q}r_{i,\,j}, \tag{3.68}$$

где $\Delta_{p,\,q}$ — значение отклонения ребра $u_{p,\,q}$ графа G на предыдущем шаге, $r_{p,\,q}$ — элемент матрицы $\mathbf{R}_i$, соответствующий ребру $u_{p,\,q}$; $r_{i,\,j}$ — элемент матрицы $\mathbf{R}_i$, соответствующий вынесенному ребру $u_{i,\,j}$ на данном шаге выполнения алгоритма. Алгоритм имеет временную сложность $0(n^3)$.

Заметим, что перед переходом от графа к схеме можно выделить ребра, вносящие пересечения, оставшуюся плоскую часть графа расположить с помощью, например, метода стереографической проекции в заданных точках плоскости с последующим внесением выделенных ребер.

Рассмотрим минимизацию числа внутрисхемных пересечений методом ветвей и границ. Пусть предварительно по критерию минимальной длины соединений произведено размещение фрагментов цепей схемы на плоскости. Пусть плоскость разбита на горизонтальные и вертикальные линейки, называемые каналами, в которых размещаются фрагменты цепей.

Исходной информацией служит расширенная матрица цепей $\mathbf{T}=\|t_{i,\,j}\|$, где $i\in I$, $I=\{1, 2, ..., f\}$ — номер элемента; $j\in\mathcal{J}_1$, $\mathcal{J}_1=\{1, 2, ..., d\}$ — номер контактов верхней части элемента; $j\in\mathcal{J}_2$, $\mathcal{J}_2=\{d+r+1, ..., 2d+r\}$ — номер контактов нижней части элемента; r — число магистралей вертикального канала. Под магистралью понимается часть канала для размещения фрагментов цепей схемы. Элемент $t_{i,\,j}$ для $j\in\mathcal{J}_1$ или $j\in\mathcal{J}_2$ является номером цепи, инцидентной j-му контакту i-го элемента схемы. Значение $t_{i,\,j}$ для $j\in\mathcal{J}_3$ или $j\in\mathcal{J}_4$, где $\mathcal{J}_3=\{d+1, ..., d+r\}$; $\mathcal{J}_4=\{2d+r+1, ..., 2d+2r\}$, означает номер цепи, фрагменты которой, размещенные в горизонтальном канале, примыкающем к i-му элементу, соединены непосредственно друг с другом, причем для одного из фрагментов это соединение является началом или концом распространения. Если $j\in\mathcal{J}_3$, то рассматривается горизонтальный канал, расположенный над линейкой, если $j\in\mathcal{J}_4$, — под линейкой. Границы распространения фрагментов цепей в каналах помечаются индексами. Если в матрице $\mathbf{T}$ рядом с номером цепи стоит индекс a, то цепь, начиная с соответствующего контакта или фрагмента цепи, расположенного в вертикальном канале, распространяется в горизонтальном канале вправо, если же стоит индекс b, — то влево, если стоит индекс c, то фрагмент цепи распростра-

няется с этого места в вертикальном канале вниз, если стоит индекс d, — вверх. Полученная матрица содержит все сведения о размещении фрагментов цепей по каналам. Необходимо теперь разместить эти фрагменты по магистралям в пределах канала с минимизацией числа пересечений между ними.

Для работы алгоритма строится вспомогательная матрица пересечений $\mathbf{P}=\|p_{m,n}\|$, где $p_{m,n}$ равно числу пересечений между фрагментами цепей t_m и t_n, расположенными в одном канале, причем фрагмент цепи t_m расположен под фрагментом цепи t_n (m, $n=1, 2, ..., \gamma, m+n$), где γ — число фрагментов в канале.

Строка m матрицы $\mathbf{P}$ ставится в соответствие с номером цепи t_m, строка n — с номером t_n. Если фрагмент цепи t_m не может быть расположен ниже t_n, то элементу $p_{m,n}$ присваивается значение ∞. Значения $p_{m,n}$ определяются по матрице цепей $\mathbf{T}$.

Для решения задачи о размещении фрагментов цепей по магистралям в пределах одного канала по критерию минимального числа пересечений воспользуемся методом ветвей и границ. Метод в применении к данной задаче заключается в том, что множество всех допустимых решений, т. е. возможных размещений фрагментов цепей по магистралям, разбивается на подмножества с последовательно уменьшающимся числом элементов. Этот процесс сопровождается вычислением нижних границ числа пересечений между фрагментами цепей. На некотором этапе получается подмножество, состоящее из одного решения, для которого число пересечений не больше, чем для любого иного решения.

Обозначим через Q множество всех решений. Множество Q разбивается на подмножества $Q^1{}_t$, $t\in T^l$, где T^l — множество фрагментов цепей, расположенных в l-м канале. Для всех решений подмножества $Q^1{}_t$ фрагмент цепи t размещен ниже всех остальных фрагментов цепей. Множество $Q^1{}_t$ разбиваем на подмножества $Q^2{}_t$. Для всех решений подмножества $Q^2{}_t$ фрагмент цепи t_2 расположен над фрагментом цепи t_1, но ниже всех остальных фрагментов. Процесс ветвления сопровождается вычислением нижних границ числа пересечений для каждого подмножества решений. Получение нижней границы подмножества решений основывается на следующем утверждении. Относительно любого фрагмента цепи остальные в пределах канала можно расположить так, чтобы он давал с ними наименьшее число пересечений $s_{t,m}$, $t_m\in T^l$.

Сумма всех чисел s_t дает нижнюю границу на множестве решений для целевой функции

$$\xi(Q)=1/2\sum_{t_m} s_{t_m}, \tag{3.69}$$

т. е. для любого решения $q\in Q$ имеет место соотношение $f(q)\geqslant \geqslant\xi(Q)$.

Определим отклонение δ_{t_m} от наименьшего числа пересечений s_{t_m} для каждого фрагмента цепи, если его поместить ниже остальных фрагментов:

$$\delta_{t_m} = \sum p^m{}_n,\ t_m,\ t_n \in T^l; \tag{3.70}$$

$$p^m{}_n = \begin{cases} p_{m,n} - p_{n,m}, \text{ если } p_{m,n} - p_{n,m} > 0; \\ 0, \text{ если } p_{m,n} - p_{n,m} \leqslant 0. \end{cases} \tag{3.71}$$

Положительное значение $p_{m,n} - p_{n,m}$ указывает на то, что цепь t_m, помещенная ниже цепи t_n, дает увеличение пересечений на значение этой разницы. Отрицательное значение $p_{m,n} - p_{n,m}$ указывает на то, что цепь t_m, помещенная ниже цепи t_n, дает с ней меньшее число пересечений, и это число вошло в состав величины s_{t_m}. Для каждого подмножества решений $Q^1{}_{t_m}$ значение нижней границы определяется следующим образом:

$$\xi(Q^1{}_{t_m}) = \xi(Q) + \delta_{t_m}. \tag{3.72}$$

Дальнейшему разбиению подвергается то подмножество $Q^1{}_{t_m}$, для которого $\xi(Q^1{}_{t_m})$ имеет наименьшее значение. Из матрицы вычеркивается m-й столбец и m-я строка, соответствующие цепи t_m. Над полученной матрицей $\mathbf{P}_1$ производятся аналогичные действия. Подсчитываются значения $\delta^1{}_{t_m}$ и дальнейшему разбиению подвергается то подмножество $Q^2{}_{t_m}$, для которого $\xi(Q^2{}_{t_m}) = \xi(Q^1{}_{t_m}) + \delta^1{}_{t_m}$ имеет наименьшее значение. В матрице вновь вычеркиваются соответствующие строка и столбец и получается матрица $\mathbf{P}_2$.

Аналогичные действия производятся до тех пор, пока не будет получено подмножество, состоящее из одного решения. Это решение представляет собой упорядоченное множество номеров фрагментов цепей. Фрагменты цепей должны размещаться по магистралям в соответствии с этим порядком. На некотором шаге n может оказаться, что для нескольких фрагментов цепей $t_m \in T_j$ значения $\delta^n{}_{t_m} = 0$. Тогда из матрицы $\mathbf{P}_n$ вычеркиваются строки и столбцы, соответствующие указанным цепям. В упорядоченное множество номеров фрагментов цепей, являющееся решением, эти номера можно записать в любом порядке относительно друг друга. Вершины $Q_{t_m}{}^{n+1}$, $t_m \in T_j$, объединяются в одну вершину $Q_{T_j}{}^{n+1}$, и ветвление продолжается от вершины $Q_{T_j}{}^{n+1}$.

Если для нескольких цепей $t_m \in T_j$ величина δ_{t_m} имеет одинаковое, но отличное от нуля значение, то строится матрица $\mathbf{P}^1{}_n$, содержащая столбцы и строки, соответствующие цепям множества T_j. По матрице $\mathbf{P}^1{}_n$ подсчитывается значение δ_{t_m}. Если наименьшее и отличное от нуля значение δ_{t_m} имеет несколько цепей, то по матрице $\mathbf{P}^1{}_n$ строится $\mathbf{P}^2{}_n$ и т. д. до тех пор, пока не будет найден один фрагмент цепи, для которого δ_{t_m} имеет наименьшее значение, либо группа, для которой $\delta_{t_m} = 0$. Ветвление будем продолжать соответственно либо от вершины $Q_{t_m}{}^{n+1}$, либо от $Q_{T_j}{}^{n+1}$. После того как будет достигнута конечная вершина $Q^v{}_{t_m}$, необходимо отыскать не подвергавшиеся ветвлению вершины, у которых

нижняя граница меньше, чем у вершины $Q^{v}{}_{t_m}$, и продолжить их ветвление. Процесс продолжается до тех пор, пока не будет найдена вершина, для которой число пересечений будет не больше, чем у остальных. На практике алгоритм с целью экономии машинного времени можно ограничить одной ветвью.

Например, для части схемы, изображенной на рис. 3.55, разместим фрагменты цепей с минимизацией числа пересечений между ними. Матрица цепей **T** для данного фрагмента имеет вид

$$\mathbf{T}=\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ 1 & 1a & 2a & 3a & 4a & — \\ 2 & — & 2 & 3b & 5a & — \\ 3 & 5b & 6a & 7a & 1 & 1c \\ 4 & 1b & 4 & 2 & 8a & 2c \\ 5 & 7 & 8 & 4b & 9a & 9d \\ 6 & 6b & 8 & 7 & 10a & — \\ 7 & 8b & 9b & 7b & 10b & — \end{array}.$$

По матрице **T** строим матрицу **P**:

$$\mathbf{P}=\begin{array}{c|cccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 0 & 2 & 2 & 1 & 2 & 1 & 1 & 0 & 0 & 0 \\ 2 & 3 & 0 & 2 & 2 & 2 & 1 & 1 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 3 & 3 & 1 & 0 & 2 & 1 & 2 & 2 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 3 & 2 & 0 & 2 & 0 & 0 & 2 & 2 & 1 & 0 \\ 7 & 3 & 2 & 0 & 2 & 0 & 1 & 0 & 4 & 2 & 1 \\ 8 & 0 & 1 & 0 & 1 & 0 & 1 & 2 & 0 & 1 & 1 \\ 9 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 3 & 0 & 1 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}.$$

4	4	0	6	0	6	5	0	3	0
δ_1	δ_2	δ_3	δ_4	δ_5	δ_6	δ_7	δ_8	δ_9	δ_{10}

Разбиение множества решений Q имеет вид рис. 3.56. По формулам (3.71) подсчитаем значения: $p_{1,2}—p_{2,1}=—1; p^1{}_2=0;$ $p_{1,3}—p_{3,1}=2;$ $p^1{}_3=2;$ $p_{1,4}—p_{4,1}=—2;$

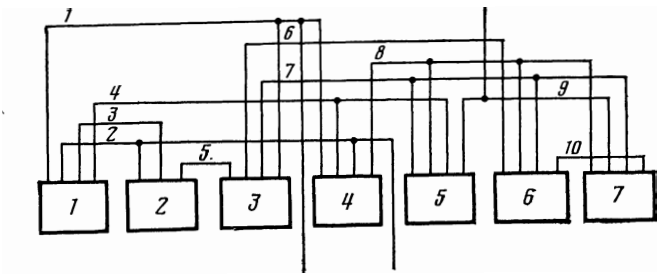


Рис. 3.55. Случайное размещение фрагментов цепей

$p^1_4=0$; $p_{1,5}-p_{5,1}=2$; $p^1_5=2$; $p_{1,6}-p_{6,1}=-2$; $p^1_6=0$; $p_{1,7}-p_{7,1}=-2$, $p^1_7=0$; $p_{1,8}-p_{8,1}=0$; $p^1_8=0$; $p_{1,9}-p_{9,1}=0$, $p^1_9=0$; $p_{1,10}-p_{10,1}=0$, $p^1_{10}=0$.

Отсюда $\delta_1=p^1_2+p^1_3+p^1_4+p^1_5+p^1_6+p^1_7+p^1_8+p^1_9+p^1_{10}=4$. Аналогично подсчитаем и другие значения δ_{t_m}. В результате получим, что δ_3, δ_5, δ_8 и δ_{10} равны нулю. Объединим вершины Q^1_3, Q^1_5, Q^1_8 и Q^1_{10} в одну Q^1_{T1}, где $T=\{3, 5, 8, 10\}$ (см. рис. 3.56), и процесс ветвления продолжим от нее. В матрице **P** вычеркнем 3, 5, 8, 10-й столбцы и строки. Получим матрицу

$$\mathbf{P}_1=\begin{array}{c|cccccc} & 1 & 2 & 4 & 6 & 7 & 9 \\ 1 & 0 & 2 & 1 & 1 & 1 & 0 \\ 2 & 3 & 0 & 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 0 & 1 & 2 & 0 \\ 6 & 3 & 2 & 2 & 0 & 2 & 1 \\ 7 & 3 & 2 & 2 & 1 & 0 & 2 \\ 9 & 0 & 0 & 0 & 2 & 2 & 0 \\ \hline & 0 & 1 & 3 & 5 & 3 & 1 \\ & \delta^1_1 & \delta^1_2 & \delta^1_4 & \delta^1_6 & \delta^1_7 & \delta^1_9 \end{array}.$$

Подсчитаем величины $\delta^1 t_m$ для матрицы $\mathbf{P}_1$. Наименьшее значение имеет $\delta^1_1=0$. Процесс ветвления продолжим от вершины Q^2_1. В матрице $\mathbf{P}_1$ вычеркнем первые строку и столбец и перейдем к матрице $\mathbf{P}_2$.

Для матрицы $\mathbf{P}_2$ наименьшее значение имеет $\delta^2_2=0$. Процесс ветвления продолжим от вершины Q^3_2. В матрице $\mathbf{P}_2$ вычеркнем строку и столбец, соответствующие второй цепи, и перейдем к матрице $\mathbf{P}_3$.

Наименьшее значение имеют δ^3_4 и δ^3_7, равные нулю. Объединим вершины Q^4_3 и Q^4_7 в одну Q^4_{T2}, где $T_2=\{4, 7\}$.

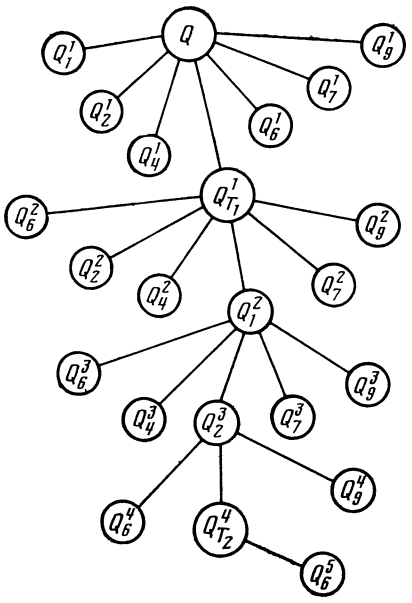


Рис. 3.56. Разбиение множества Q

Вычеркнем в матрице $\mathbf{P}_3$ строку и столбцы, соответствующие цепям 4 и 7. Получаем матрицу

$$\mathbf{P}_4=\begin{array}{c|cc} & 6 & 9 \\ 6 & 0 & 1 \\ 9 & 2 & 0 \\ \hline & 0 & 1 \\ & \delta^4_6 & \delta^4_9 \end{array}.$$

Наименьшее значение имеет $\delta^4_6=0$. Вершина Q^5_6 будет концевой. Упорядоченное множество цепей T^l запишем следующим образом: $T^l=\langle T_1, 1, 2, T_2, 6, 9\rangle$, где $T_1=\{3, 5, 8, 10\}$; $T_2=\{4, 7\}$.

Размещение фрагментов цепей множества T^l произведем в соответствии со следующим правилом. Пусть уже заполнено n магистралей и необходимо разместить очередной фрагмент цепи t_m упорядоченного множества T^l. Определяется, пересекается ли фрагмент цепи t_m с фрагментами цепей, размещенных в n магистрали. Если нет, то в n—1 и т. д., пока не найдется такая n—k магистраль, в которой расположены фраг-

менты цепей, с которыми пересекается фрагмент цепи t_m. Фрагмент цепи t_m помещается в $(n—k+1)$-ю магистраль. В соответствии с правилом для примера в первую магистраль будут помещены фрагменты цепей 3, 5, 8, во вторую — 1, 10, в третью — 2, в четвертую — 4, в пятую — 7, в шестую — 6, в седьмую — 9. При случайном размещении фрагментов цепей по магистралям они давали 41 пересечений (см. рис. 3.55), а при размещении в результате работы алгоритма — 26 (рис. 3.57).

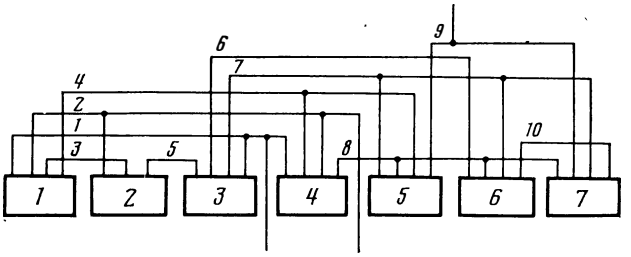


Рис. 3.57. Размещение фрагментов цепей в результате работы алгоритма

Рассмотрим методику решения задачи размещения разногабаритных элементов на плоскости с предварительной оценкой необходимой площади. Это могут быть размещение фрагментов топологии на кристалле, размещение интегральных микросхем разного размера на печатных платах и т. п. Пусть задано $X=\{x_i\}$, $i=1, 2, ..., n$ — множество типовых элементов схем ЭА. Каждый элемент $x_\alpha \in X$ характеризуется шириной l_α, высотой h_α.

Задача размещения заключается в нахождении такого варианта расположения элементов множества X на плоскости, при котором требуемая площадь минимальна.

Решение задачи размещения разногабаритных элементов обычно проводят в два этапа. На первом этапе используется дискретная графотеоретическая модель схемы с представлением элементов материальными точками. На втором этапе используются графовые модели компонентов, причем задаются размеры элементов $x_\alpha \in X$. На первом этапе размещения производится предварительное размещение материальных точек, соответствующих вершинам $x_\alpha \in X$ графа схемы. Здесь можно использовать последовательный, итерационный или точные алгоритмы, описанные выше. Затем производится уточнение размещения элементов $x_\alpha \in X$ в линейках плоскости с учетом размеров плоскости и элементов схемы.

Перед вторым этапом производятся оценка необходимой площади плоскости (кристалла или платы) и вычисление размеров рабочего поля. На втором этапе обычно выполняют размещение прямоугольников. Для этой цели используются последовательные и итерационные алгоритмы. Последовательные алгоритмы обычно состоят из двух процедур: а) выбора наилучшего (по заданному критерию) элемента для размещения; б) выбора наилучшей (по заданному критерию) для данного элемента позиции.

После выполнения первой процедуры выбирается элемент схемы x_α, имеющий наиболее неблагоприятные условия для размещения (большие габариты, большое число связей, наиболее длинные связи и т. д.). Вторая процедура обеспечивает непосредственное размещение разногабаритных элементов. Позиция на плоскости и ориентация элемента $x_\alpha \in X$ выбираются по критерию, учитывающему суммарную длину соединений и плотность упаковки элементов. Длина соединений оценивается длиной полупериметра зоны реализации цепи. При необходимости указывают области на плоскости, запрещенные для размещения.

Итерационные алгоритмы размещения разногабаритных элементов имеют ряд особенностей:

размещаемые элементы стандартизованы по высоте и имеют в общем случае различную ширину, что создает ограничения подвижности элементов при перестановках;

ориентация и распределение выводов элементов не конкретизируются;

позиции размещения отдельных элементов не фиксируются.

В качестве целевой функции размещения, подлежащей минимизации, выбрана $\mathbf{F}=\sum_{i=1}^{m}|l_i|$, где $|l_i|$ — оценка длины i-й цепи; m — число цепей в схеме.

Вводятся следующие условия. Пробная перестановка i-го и j-го элементов между собой на плоскости фиксируется, если $\Delta\mathbf{F}(i, j)\leqslant 0$, где $\Delta\mathbf{F}(i, j)$ — изменение функции при (i, j)-перестановке; $\mathbf{F}(i, j)$ — значение $\mathbf{F}$ после (i, j)-перестановки.

При $\Delta\mathbf{F}(i, j)=0$ и выполнении ограничений (i, j)-перестановка фиксируется, если она уменьшает значение разности ширины незаполненных частей плоскости размещения элементов x_i и x_j, т. е. $\Delta s(i, j)=|\Delta s_i(x_i)-\Delta s_j(x_j)|-|\Delta s_i(x_j)-\Delta s_j(x_i)|>0$, где $\Delta s(i, j)$ — величина изменения разности ширины незаполненной части плоскости при (i, j)-перестановке элементов x_i и x_j; $\Delta s_i(x_j)$ — ширина незаполненной части плоскости после установки элемента x_j на место x_i.

Пусть $\mathbf{E}$ — множество цепей схемы; $E_{i, j}$ — подмножество цепей, связанных с элементами x_i или x_j.

Образуем кортеж элементов начального размещения $\beta=\langle x_1, x_2, \ldots, x_q, \ldots, x_N\rangle$, где q — номер позиции элемента. Назовем *циклом* q последовательность перестановок элемента x_q с остальными элементами, *основным* — элемент x_q, а дополняющим элементом парной перестановки — произвольный элемент x_j. В цикле q в роли дополняющих испытывают элементы x_j, для которых $j>q$. Для исключения повторяемости перестановок вводится матрица $\mathbf{R}_n=\|r_{i, j}\|$, элементы $r_{i, j}$ и $r_{j, i}$ которой отмечаются признаком запрета z при фиксации (i, j)-перестановки. При фиксации перестановки элементы меняются своими позиционными номерами.

Алгоритм размещения элементов итерационным методом:

1°. Образование кортежа размещенных элементов β. Вычисление начальных значений $|l_1|$, $|l_2|$, ..., $|l_m|$.

2°. Выбор основного элемента, организация цикла q, $i:=i+1$, $j:=i$.

3°. Выбор дополняющего элемента $j:=j+1$. Если $x_{i,j}=x_{j,i}=z$, переход к 9°.

4°. Пробная (i, j)-перестановка. Вычисление новых значений $|l|$ для цепей из $E_{i,j}$.

5°. Вычисление $\mathbf{F}(i, j)$. Если $\Delta\mathbf{F}(i, j)<0$, переход к 9°.

6°. Если $\Delta\mathbf{F}(i, j)>0$, переход к 8°.

7°. Вычисление $\Delta s(i, j)$. Если $\Delta s(i, j)\leqslant 0$, переход к 9°.

8°. Фиксация (i, j)-перестановки. Присвоение $x_{i,j}$, $x_{j,i}=z$. Запись новых значений $|l|$ для цепей из $E_{i,j}$.

9°. Проверка конца цикла q. Если $j<N$, возврат к 3°.

10°. Проверка последнего цикла. Если $i<N-1$, возврат к 2°.

11°. Конец работы алгоритма.

Описанный алгоритм имеет временную сложность $0(n^2)$—$0(n^4)$. Из-за ограничения подвижности элементов при перестановках требуемый результат не всегда удается получать однократным применением алгоритма. В этом случае требуется повторить процесс решения.

3.3. ТРАССИРОВКА СОЕДИНЕНИЙ

Основные трудности, возникающие при трассировке соединений, определяются особенностями монтажа (они заключаются в возможности получения проводников весьма сложных конфигураций, зависящих от вида ранее проведенных соединений), а также сложностью составления программ алгоритмов трассировки и большими затратами машинного времени.

Одновременная оптимизация всех соединений при трассировке за счет полного перебора вариантов в настоящее время невозможна. Поэтому разрабатываются в основном локально оптимальные методы трассировки, когда трасса оптимальна лишь на данном шаге при наличии ранее проведенных соединений.

Основная задача трассировки формулируется следующим образом: по заданной схеме соединений (или полученной из нее информации в виде размещения) проложить необходимые пути прохождения проводников на плоскости (плате, ТЭЗ, кристалле и т. п.), чтобы реализовать заданные электрические соединения с учетом заранее заданных ограничений. Основными являются ограничения на ширину проводников и минимальные расстояния между ними. Эту задачу можно условно представить в виде четырех последовательно выполняемых этапов:

1) определение перечня соединений;
2) размещение по слоям;
3) определение порядка соединений;
4) трассировка проводников.

На первом этапе формируется точный список, указывающий, какие соединения (цепи или фрагменты цепей схемы) проведены.

На втором этапе предпринимается попытка точно решить, где каждое соединение может располагаться.

На третьем этапе определяется порядок соединений для каждого слоя, т. е. указывается, когда каждый проводник, помещенный в этот слой, будет проведен.

На четвертом этапе дается ответ, каким образом должно быть проведено каждое соединение.

Рассмотрим указанные этапы более подробно. Исходная информация для первого этапа — это перечень соединений контактов каждого элемента с контактами других элементов. Одиночные поконтактные соединения записываются непосредственно в перечень соединений (матрицу цепей или список). Трудность при реализации возникает, когда один контакт необходимо соединить с несколькими контактами. Для решения этого вопроса используют алгоритмы построения минимальных связывающих деревьев Прима или Штейнера. Опишем алгоритм Прима, основанный на соединении пар точек.

Правило 1. Две вершины (два контакта) соединяются кратчайшим путем, если они ближе всего расположены друг к другу.

Правило 2. Вершина присоединяется к ранее связанным вершинам кратчайшим путем.

Правило 3. Если следующая пара наиболее близко расположенных вершин представляет собой два контакта, для которых трасса соединений уже существует, то эта пара пропускается. На этом этапе ЭВМ не производит соединений, она только решает, какие проводники должны быть растрассированы. Один из возможных подходов к решению задачи соединения множества вершин заключается во введении промежуточных вершин с целью формирования еще более короткого дерева соединений. Данная задача называется задачей Штейнера. В общем случае для $n>4$ не известно точного ее решения.

При определении перечня соединений необходимо также решать вопросы распределения выводов элементов по контактам электрического соединителя разъема. Наиболее применимым на практике критерием является отсутствие пересечений между элементами и соответствующими контактами электрического соединителя.

После определения перечня трассируемых соединений решается задача размещения проводников по слоям. Она сводится к отысканию такого расположения проводников в слоях, чтобы в каждом из них число взаимных пересечений было минимальным. Окончательный вариант размещения по слоям можно улучшить, если проанализировать каждый проводник.

На третьем этапе производится исследование слоев с целью точного определения возможности трассировки в них проводников. Для двух проводников существует правило выбора: если контакт B появляется в прямоугольнике A, т. е. в прямоугольнике, имею-

щем в противоположных углах контакты проводника A, то проводник B надо трассировать первым (рис. 3.58). К сожалению, не всегда это правило применимо. Сформулируем общее эвристическое правило Айкерса о порядке трассировки проводников: проводники трассируются в порядке приоритетных номеров. Приоритетный номер проводника v равен числу контактов в прямоугольнике. Пусть необходимо определить порядок трассировки проводников v_{AA}, v_{BB}, v_{CC}, v_{DD} (рис. 3.59). Согласно правилу Айкерса определим приоритетные номера $N_{AA}=1$, $N_{BB}=3$, $N_{CC}=2$, $N_{DD}=0$. Тогда получим следующий порядок трассировки: I — v_{DD}, II — v_{AA}, III — v_{CC}, IV — v_{BB}.

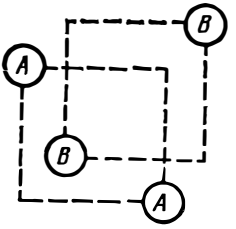


Рис. 3.58. Пример правила выбора для двух проводников

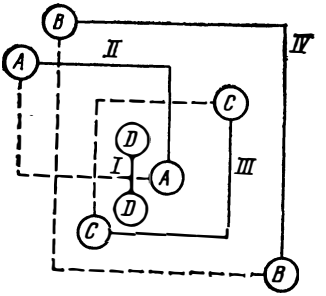


Рис. 3.59. Пример для четырех проводников

На практике обычно короткие горизонтальные и вертикальные отрезки проводников трассируются первыми, далее трассируются окружающие их почти вертикальные и горизонтальные проводники. В заключение трассируются длинные соединения, которые чаще располагаются по отношению к другим с внешней стороны.

Обычно проблему размещения соединений по слоям рассматривают как задачу раскраски специального графа, после чего исследуют каждый слой с целью определения внутри него порядка обработки соединений. После того как все соединения распределены по слоям и определен порядок их обработки, решается задача трассировки фрагментов всех цепей, что и выполняется на четвертом этапе.

Четвертый этап трассировки является основным, наиболее трудоемким, определяющим эффективность и качество задачи трассировки в целом. На этом этапе используются четыре фундаментальных алгоритма:

волновой трассировки,
лучевой трассировки,
канальной трассировки,
гибкой (топологической) трассировки.

Опишем основные принципы волнового алгоритма Ли.

1°. Плоскость трассировки (плата, кристалл) разбивается на прямоугольные площадки — дискреты заданного размера. Размер

дискретной площадки определяется допустимыми размерами проводников и расстояниями между ними. Задача проведения трасс сводится к получению последовательности дискретов, соединяющих элементы a, b, соответствующие выводам (контактам) элементов схемы.

2°. Вводится весовая функция $\mathbf{F}=\mathbf{F}(f_1, f_2, \ldots, f_r)$ как критерий качества пути. Здесь f_i характеризует путь с конкретной точки зрения — длины, числа пересечений, числа переходных отверстий, числа изгибов и т. п.

3°. Начиная с элемента a, дискретам, соседним с ранее просмотренными, присваивается определенное значение весовой функции $\mathbf{F}=m(i, j)$. Этот этап проводится итерационно и продолжается до тех пор, пока элементу b не будет присвоено некоторое значение веса $m(i_b, j_b)$.

4°. Начиная от элемента b, производится перемещение к элементу a по пройденным дискретам таким образом, чтобы значение весовой функции дискретов убывало монотонно. В результате получается трасса, соединяющая элементы a и b.

Рассмотрим подробнее процесс проведения трассы из элемента a в элемент b. На плоскости, где необходимо провести трассу, моделируется распространение волны из «источника» a, до тех пор, пока фронт волны не достигнет «приемника» b или пока на шаге k фронт волны не сможет включить ни одного незанятого дискрета. Отметим, что волна распространяется только по свободным дискретам. Эта часть алгоритма называется *распространением волны*. Если после ее окончания достигнут элемент b, то выполняется вторая часть алгоритма — *проведение трассы*. При распространении волны алгоритм последовательно шаг за шагом строит 1-й фронт, 2-й фронт, ..., k-й фронт источника a.

Процесс построения нового фронта начинается с присваивания дискретам, соседним с дискретами предыдущего фронта, весов $m(i, j)$. Вес $m(i_k, j_k)$, присвоенный дискретам k-го фронта, является суммой весов дискретов фронта k—1. Кроме веса $m(i_k, j_k)$. каждому дискрету фронта k приписывается путевая координата, определяющая дискрет, из которого в данный поступила волна. Таким образом трассы следуют по путевым координатам, выполняя 4°.

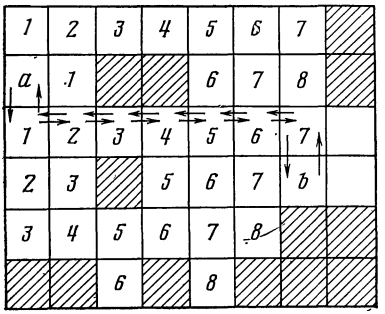


Рис. 3.60. Пример соединения элементов a и b в волновом алгоритме

На рис. 3.60 показан пример соединения элементов a и b волновым алгоритмом. Здесь заштрихованы запрещенные площадки. Заметим, что существует несколько вариантов проведения пути, из которых конструктор (ЭВМ) выбирает один, наиболее удовлетворяющий заданным критериям. Для повышения быстродействия волно-

вых алгоритмов предлагают: а) одновременное распространение волны от двух соединяемых точек, б) начало трассировки с вершины (контакта), максимально удаленной от центра области трассировки; в) сокращение области трассировки за счет использования пары соединяемых контактов, охватывающих искусственную границу в виде прямоугольника.

Для уменьшения объема требуемой оперативной памяти ЭВМ предлагается соблюдать условие: метки-предшественники должны отличаться от меток-преемников. Это дает возможность использовать последовательность кодирования 1—1—2—2—1—1—2—2—1. Следовательно, для описания состояния любой ячейки достаточно иметь два бита памяти ЭВМ.

Известная модификация волнового алгоритма учитывает степень близости к препятствиям. Она использует представление плоскости в виде трехмерной топографической модели. При работе волнового алгоритма используется информация о «высотах», присвоенных ячейкам плоскости, для того чтобы вызвать различную скорость распространения числовой волны на различных участках поля, т. е. вызвать замедление волны на участках, примыкающих к преградам. Алгоритм состоит из трех этапов: распространение волны, проведение пути и коррекция топографической модели. Идея всех алгоритмов коррекции по топографической модели конфигурации полученной трассы заключается в следующем. Для каждого из отрезков цепи подсчитывается сумма «высот» площадок, такие же суммы подсчитываются при смещении этих отрезков параллельно самим себе в одну или другую сторону. Далее берется разность сумм, и трасса пойдет по тем площадкам, для которых сумма «высот» будет минимальной.

В некоторых алгоритмах используются идеи построения максимальных потоков на сети для трассировки проводников на слоях печатной платы. При этом множество площадок платы заменяется сетью, т. е. клеткам (площадкам) графа G_r ставятся в соответствие узлы сети, границам площадок ненулевой длины — ребра, соединяющие эти узлы. На предварительном этапе в узлы и ребра сети ставятся определенные признаки запрета и свободы. Введение запретов и задержек прохождения по ребрам сети соответствует введению запретов и ограничений на проведение трасс. Это позволяет бороться с возникновением паразитных емкостей проводников в процессе трассировки.

Рассмотренные модификации волнового алгоритма обеспечивают построение минимальной длины между точками на плоскости, если между ними существуют препятствия для проведения соединений.

Следует заметить, что волновой алгоритм связан со значительными временны́ми затратами на решение задач трассировки, причем для распространения волны производится около 90% всех вычислений, а для этапа проведения пути — 10%. В связи с тем, что в некоторых практических схемах ЭА большинство соединений имеет несложную форму и для проведения пути нет необходимо-

сти просматривать все клетки сетки, в которой располагается схема соединений, целесообразно применять *лучевой алгоритм*. Основная идея алгоритма заключается в исследовании сетки для определения пути между вершинами a, b по некоторым заранее заданным направлениям, подобным лучам. Перед выполнением алгоритма задается число лучей, распространяемых из площадки a и площадки b, а также порядок присвоения путевых координат. Обычно число лучей для каждой из площадок (источников) принимается равным двум. Распространение лучей происходит одновременно из обоих источников до встречи двух разноименных лучей в некоторой площадке c.

Рассмотрим пример проложения трассы для соединения площадок a и b с помощью двухлучевого алгоритма (рис. 3.61). Для источников a и b взято по два луча с противоположными направ-

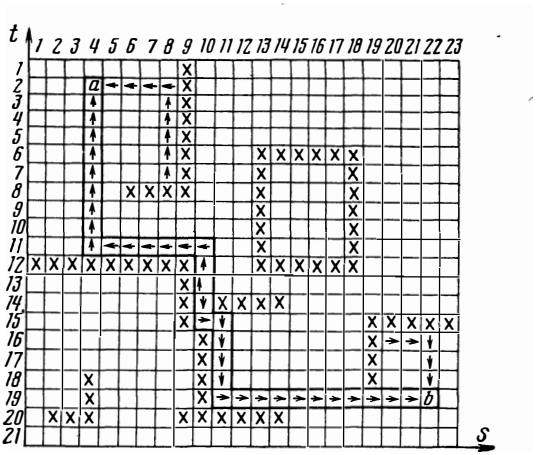


Рис. 3.61. Пример двухлучевого алгоритма

лениями: $a^{(1)}$ — вниз, вправо; $b^{(1)}$ — вверх, влево; $a^{(2)}$ — вправо, вниз; $b^{(2)}$ — влево, вверх. Если площадка b будет расположена не слева от a, а справа, то путевые координаты влево и вправо надо поменять местами. После первого шага алгоритма занимаются площадки с координатами $(t_2; s_5)$, (t_{19}, s_{21}), (t_{18}, s_{22}), (t_3, s_4). На пятом шаге луч $b^{(2)}$ заблокирован. На 18-м шаге встретились лучи $b^{(2)}$ и $a^{(1)}$. Так как с помощью лучевых алгоритмов не всегда возможно получать решения, то его целесообразно применять совместно с волновым алгоритмом. Обычно с помощью лучевого алгоритма удается проведение 60 % трасс, остальные трассы проводятся с помощью волнового алгоритма или конструктором. Лучевые алгоритмы наиболее целесообразно применять для трассировки плат с небольшой степенью заполнения ячеек. В этом случае он позволит значительно экономить время по сравнению с волновым алгоритмом.

В настоящее время распространение получают *канальные алгоритмы* трассировки, основанные на проложении трасс по укрупненным дискретам рабочего поля, в качестве которого служит система горизонтальных и вертикальных каналов. Ширина каждого канала зависит от числа прокладываемых в них соединений. Канал разбивается на линейки, называемые магистралями. Любое соединение при канальной трассировке будет представлять совокупность объединенных в одну цепь участков магистралей. Реализация канального алгоритма предполагает выполнение двух процедур: распределение соединений по каналам с учетом их оптимальной загрузки и оптимизация расположения соединений на магистралях каналов. Для заданной конструктивно-технологической базы изготовления печатных плат каждому каналу монтажной плоскости можно поставить в соответствие число, называемое пропускной способностью канала и обозначающее максимально допустимое число проводников, проходящее через сечение канала с выполнением технологических ограничений. При этом процедура оптимального распределения соединений по каналам в простом случае сводится к равномерной их загрузке, а в более сложных случаях дополнительно осуществляется учет фактора электромагнитной и тепловой совместимости соседних проводников. Целью выполнения второй процедуры — оптимизация расположения соединений на магистралях каналов — является минимизация числа переходных отверстий с одного слоя монтажной плоскости на другой.

Класс канальных алгоритмов трассировки выгодно отличается от волновых повышенным быстродействием, меньшим расходом памяти ЭВМ, однако он менее универсален, что налагает определенные ограничения на класс конструктивно-технологических решений. В волновые, лучевые и в меньшей мере канальные алгоритмы заложен последовательный принцип трассировки, что предполагает оптимальное построение каждой отдельной трассы без прогноза на прокладку последующих соединений на монтажной плоскости. Недостаток этого принципа наиболее ярко проявляется по мере приближения к концу решения задачи, когда, как правило, становится очевидным тот факт, что отдельные ранее проведенные трассы можно было бы построить неоптимально, но при этом возникли бы дополнительные возможности к прокладке ряда последующих трасс.

Метод гибкой трассировки состоит из двух этапов: а) топологического (макротрассировка); б) геометрического (микротрассировка).

На первом этапе строятся модели топологии для трасс цепей, для их плоского вложения в дискретное топологическое рабочее поле. На втором этапе строятся модели геометрии, которые определяют геометрические характеристики трасс цепей. Другими словами, на первом этапе гибкой трассировки трассы назначаются в зоны (области, дискреты), внутри которых их расположение не

фиксируется. Трассы как бы «плавают» внутри выбраннои зоны. На втором этапе трассы фиксируются жестко.

Для описания топологии трасс строится *дискретная топологическая модель рабочего поля.* Ее можно представить в виде графа $G_r=(X_r, U_r)$, где X_r соответствует контактам элементов, внешним контактам, характерным точкам края платы и запрещенным зонам; U_r — линиям, соединяющим элементы $x \in X_r$. Одна из возможных моделей дискретного рабочего поля показана на рис. 3.62.

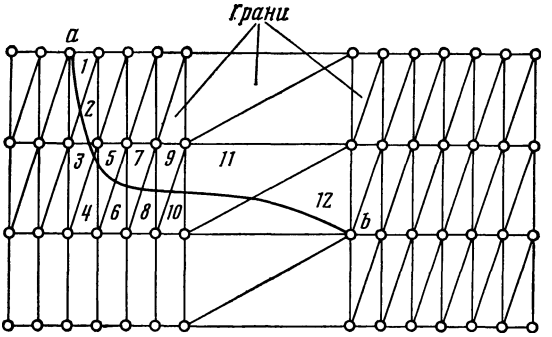


Рис. 3.62. Пример соединения контактов *a*, *b* на основе метода гибкой трассировки

Здесь дискретами являются треугольные грани. Заметим, что дискретами могут быть прямоугольные грани и вообще грани любой конфигурации. Ненасыщенные трассами грани могут сжиматься, а те, через которые проходит большое количество трасс, растягиваются. В алгоритмах гибкой трассировки ранее проложенные трассы закрепляются не жестко на монтажной плоскости, а имеют свойство деформироваться при построении последующих трасс. Построение соединений осуществляется путем организации волнового процесса на множестве граней, причем грань считается запрещенной для прохождения волны, если исчерпаны ресурсы пропускных способностей ее ребер. Соседними к некоторому ребру являются те ребра граней, которые достижимы из него без пересечений. Прокладка проводников в алгоритмах гибкой трассировки осуществляется последовательно, однако ранее проложенные трассы не закрепляются «жестко», что создает благоприятные условия для прокладки последующих соединений. Кроме того, представление монтажной плоскости в виде системы граней оказывается удобным для конструкций, использующих разногабаритные элементы.

Рассмотрим теперь вопросы, связанные с разбиением графа схемы на плоские суграфы. В гл. 2 было введено понятие числа планарности $\Theta(K_n)$ для полных графов как наименьшего числа ребер, которое необходимо удалить из графа K_n, чтобы он стал планарным. Приведем метод, позволяющий для любого графа най-

ти оценку числа планарности и указать, какие ребра следует удалить, чтобы граф G стал плоским.

Пусть задан произвольный граф $G=(X, U)$, имеющий гамильтонов цикл, хотя в дальнейшем полученные результаты распространяются и на графы без циклов. Пусть ребра, инцидентные одной вершине, не пересекаются между собой, два любых ребра $u_{i,j}$ и $u_{p,q}$ пересекаются между собой не более чем в одной точке, ребра гамильтонова цикла не образуют пересечений. Каждому ребру графа $u_{i,j}\in U$ поставим в соответствие двоичный символ $a_{i,j}$, причем

$$a_{i,j}=\begin{cases}1, \text{ если ребро } u_{i,j} \text{ лежит во внутренней области;}\\ 0, \text{ если ребро } u_{i,j} \text{ лежит во внешней области.}\end{cases}$$

Будем обозначать инверсию $a_{i,j}$ как $\bar{a}_{i,j}$.

Граф G можно рассматривать как прямую сумму двух непересекающихся по ребрам суграфов G_0 (все ребра которого расположены во внутренней области) и $\bar{G}_0$ (ребра находятся во внешней области), причем ребра гамильтонова цикла не принадлежат ни G_0, ни $\bar{G}_0$, хотя их можно отнести к любому из суграфов. По графу G, как показано выше, можно построить видоизмененную матрицу смежности $\mathbf{R}$, а для суграфов G_0 и $\bar{G}_0$ — соответственно матрицы $\mathbf{R}_0$ и $\overline{\mathbf{R}}_0$.

Общее число пересечений ребер графа G равно $P(G)=P(G_0)+P(\bar{G}_0)$, где $P(G_0)$, $(P(\bar{G}_0))$ — число пересечений ребер в суграфе $G_0(\bar{G}_0)$.

Используя формулу (3.62), получаем

$$P(G)=\sum_{k=2}^{n-2}\sum_{j=k+2}^{n}\sum_{i=k+1}^{j-1}\sum_{m=1}^{k-1}(a_{k,j}a_{m,i}+\bar{a}_{k,j}\bar{a}_{m,i}). \qquad (3.73)$$

Граф G будет плоским, если произведено его разбиение на G_0 и $\bar{G}_0$, так что $P=P_0+\bar{P}_0=0$, т. е.

$$\sum_{k=2}^{n-2}\sum_{j=n+2}^{n}\sum_{i=k+1}^{j-1}\sum_{m=1}^{k-1}(a_{k,j}a_{m,i}+\bar{a}_{k,j}\bar{a}_{m,i})=0. \qquad (3.74)$$

Таким образом, для определения планарности графа необходимо решить уравнение (3.74), т. е. найти значения переменных $a_{k,j}$ и $a_{m,i}$. Заметим, что левая часть (3.74) состоит из суммы однородных по виду элементов $a_{k,j}a_{m,i}+\bar{a}_{k,j}\bar{a}_{m,i}$, число которых равно числу пересечений ребер графа G. Так как любой член $a_{s,t}$ неотрицателен $(a_{s,t}\geqslant 0)$, то левая часть (3.74) будет равна нулю в том случае, если будут равны нулю все составляющие ее суммы однородных элементов. Поэтому необходимо, чтобы

$$a_{k,j}a_{m,i}+\bar{a}_{k,j}\bar{a}_{m,i}=0. \qquad (3.75)$$

Левая часть (3.75) является двоичной логической функцией эквивалентности, которая равна нулю, когда переменные $a_{k,j}$ и $\bar{a}_{m,i}$ не равны друг другу. Такая запись означает, что ребра $u_{k,j}$ и $u_{m,i}$ не образуют пересечений, если их провести в разных областях.

Решая (3.75), будем получать выражения вида $a_{k,\,j}=\bar{a}_{m,\,i}$, число которых равно числу однородных элементов из (3.74). В результате получим систему равенств следующего вида:

$$\begin{cases} a_{p,\,q}=\bar{a}_{m,\,n}; \\ a_{k,\,j}=\bar{a}_{m,\,i}; \\ \quad\ldots\ldots \\ a_{\alpha,\beta}=\bar{a}_{s,\,t}. \end{cases} \tag{3.76}$$

В системе (3.76) можно выделить некоторое множество подсистем, каждая из которых содержит равные элементы, причем индексы правой или левой части двух или более равенств могут совпадать, что дает возможность свести выражение (3.76) к виду

$$a_{p,\,q}=a_{k,\,j}=a_{\alpha,\beta}=\ \ldots\ =\bar{a}_{m,\,n}=\bar{a}_{m,\,i}=\ \ldots\ =\bar{a}_{s,\,t}. \tag{3.77}$$

Если в (3.77) найдется хотя бы один такой элемент $a_{s,\,t}$, для которого в этом же выражении существует $\bar{a}_{s,\,t}$, то $\Theta(G)\neq 0$. Число пар вида $a_{s,\,t}=\bar{a}_{s,\,t}$ является верхней оценкой числа планарности графа.

Рассмотрим на примере нахождение оценки числа планарности графа $\Theta(G)$. Пусть задан граф, показанный на рис. 3.63.

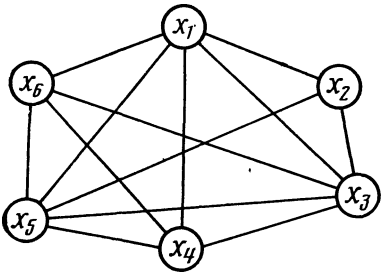


Рис. 3.63. Пример графа

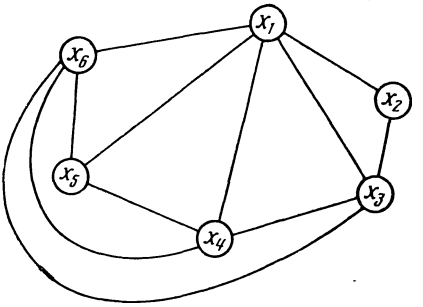


Рис. 3.64. Плоский граф

По методу, описанному выше, составим уравнения и определим $P_0(G)$ и $\bar{P}_0(G)$:

$$P_0(G)=a_{2,5}(a_{1,3}+a_{1,4})+a_{3,6}(a_{1,4}+a_{1,5}+a_{2,5})+a_{3,5}a_{1,4}+a_{4,6}(a_{1,5}+a_{2,5}+a_{3,5});$$

$$\bar{P}_0(G)=\bar{a}_{2,5}(\bar{a}_{1,3}+\bar{a}_{1,4})+\bar{a}_{3,6}(\bar{a}_{1,4}+\bar{a}_{1,5}+\bar{a}_{2,5})+\bar{a}_{3,5}\bar{a}_{1,4}+\bar{a}_{4,6}(\bar{a}_{1,5}+\bar{a}_{2,5}+\bar{a}_{3,5}).$$

Определим P как сумму: $P=P_0(G)+\bar{P}_0(G)=(a_{2,5}a_{1,3}+\bar{a}_{2,5}\bar{a}_{1,3})+(a_{2,5}a_{1,4}+$
$+\bar{a}_{2,5}\bar{a}_{1,4})+(a_{3,6}a_{1,4}+\bar{a}_{3,6}\bar{a}_{1,4})+(a_{3,6}a_{1,5}+\bar{a}_{3,6}\bar{a}_{1,5})+(a_{3,6}a_{2,5}+\bar{a}_{3,6}\bar{a}_{2,5})+(a_{3,5}a_{1,4}+$
$+\bar{a}_{3,5}\bar{a}_{1,4})+(a_{4,6}a_{1,5}+\bar{a}_{4,6}\bar{a}_{1,5})+(a_{4,6}a_{3,5}+\bar{a}_{4,6}\bar{a}_{3,5})+(a_{4,6}a_{2,5}+\bar{a}_{4,6}\bar{a}_{2,5})$.

Приравнивая элементы, заключенные в скобки, получаем систему $\{a_{2,5}=\bar{a}_{1,3},\ a_{2,5}=\bar{a}_{1,4};\ a_{3,6}=\bar{a}_{1,4},\ a_{3,6}=\bar{a}_{1,5},\ a_{3,6}=\bar{a}_{2,5},\ a_{3,5}=\bar{a}_{1,4},\ a_{4,6}=\bar{a}_{1,5},\ a_{4,6}=\bar{a}_{2,5},\ a_{4,6}=\bar{a}_{3,5}\}$.

Сведем полученную систему к выражению типа (3.77):

$$a_{2,5}=\bar{a}_{1,3}=\bar{a}_{1,4}=a_{3,6}=\bar{a}_{1,5}=\bar{a}_{2,5}=a_{3,5}=a_{4,6}=\bar{a}_{3,5}. \tag{3.78}$$

Так как в полученной подсистеме две пары вида

$$a_{i,j}=\bar{a}_{i,j}\,(a_{2,5}=\bar{a}_{2,5},\ a_{3,5}=\bar{a}_{3,5}),\ \text{то}\ \Theta'(G)=2. \tag{3.79}$$

Заметим, что в данном случае $\Theta'(G)\neq\Theta(G)$, т. е. $\Theta'(G)$ не является минимальным числом ребер, удаление которых делает граф планарным.

После получения выражения (3.78) любому элементу приписываем значение 0 или 1 и автоматически получаем значения остальных элементов. Так, в (3.78), принимая, что $a_{3,6}=0$, получаем $\bar{a}_{1,5}=\bar{a}_{1,4}=\bar{a}_{1,3}=1$, а $a_{3,6}=a_{4,6}=0$. Следовательно, чтобы граф G стал плоским, проведем ребра $u_{1,5}$, $u_{1,4}$ и $u_{1,3}$ во внутренней области, ребра $u_{3,6}$ и $u_{4,6}$ во внешней области гамильтонова цикла, а ребра $u_{2,5}$ и $u_{3,5}$ удалим. Получим граф, показанный на рис. 3.64.

Разобьем все множество связных неорграфов $M(G)$ на три непересекающихся подмножества: M_1, M_2 и M_3. В подмножество M_1 включаются все связные графы, для числа вершин и ребер которых выполняется неравенство

$$m\leqslant(n+2). \tag{3.80}$$

Подмножество графов M_1 объединяет все заведомо планарные графы. Для связных графов, входящих в подмножество M_2, число вершин и ребер должно быть связано следующим соотношением:

$$m>3(n-2). \tag{3.81}$$

Все графы, включенные в подмножество M_2, заведомо непланарны. Это следует из того, что число ребер максимально планарного графа равно $3(n-2)$.

Подмножество M_3 включает те связные графы, у которых число ребер лежит в пределах

$$(n+2)<m\leqslant 3(n-2). \tag{3.82}$$

В подмножество M_3 входят как планарные, так и непланарные графы. Рассмотрим методику определения планарности графов, входящих в подмножество M_3. Сущность алгоритма заключается в следующем. По графу строится гамильтонов цикл и записывается матрица смежности, в результате исследования которой определяются пересечения ребер графа. Затем строится граф пересечений и записывается матрица смежности этого графа. Далее по матрице смежности графа пересечений определяется его двудольность. Пусть, например, дан граф $G=(X, U)$, имеющий известный гамильтонов цикл.

После расположения графа схемы производится перенумерация вершин так, чтобы для смежных вершин x_i и x_j гамильтонова цикла выполнялось условие $i-j=1$. По графу G записывается видоизмененная треугольная матрица смежности $\mathbf{R}$. Согласно методике, описанной выше, можно определить как общее число пересечений ребер графа, так и число пересечений любого ребра. После определения для каждого ребра $u_{i,j}$ ребер, с которыми оно имеет пересечения, строится граф пересечений. Граф $G'=(U', V')$ называется графом пересечений графа G, если множество вершин графа U' определяется как $U'\subset U$ — подмножество пересекающихся ребер графа G, множество ребер $v_{i,j}=(u_i, u_j)$, если ребро u_i пересекается с u_j $(u_i, u_j\in U')$. Для графа G на рис. 3.65,*а* граф

пересечений показан на рис. 3.65,*б*. Для определения планарности графа G необходимо определить двудольность графа G', откуда следует по теореме Кёнига планарность G. Если граф G' двудольный, то G планарен. Это следует из того, что в двудольном графе можно выделить два подмножества несмежных вершин Y' и Y'' таких, что $Y' \cup Y'' = U'$ и $Y' \cap Y'' = \varnothing$, а поэтому пересекающиеся

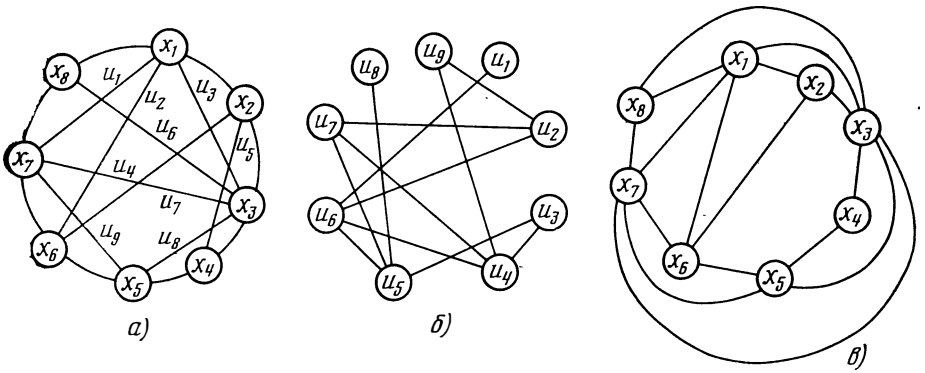


Рис. 3.65. Пример графа $G(а)$; граф пересечений G^1 $(б)$, плоское изображение графа $(в)$

ребра графа G, соответствующие вершинам подмножеств Y' и Y'', можно расположить по разные стороны гамильтонова или псевдогамильтонова цикла без пересечений. Если граф пересечений G' является деревом, то исходный граф G планарен. Если граф G' не является деревом, то можно определять максимальную двудольную часть. Она определяет разбиение множества вершин G' на два таких максимальных непересекающихся подмножества Y' и Y'', что вершины внутри каждого из них не смежны между собой. В оставшейся части графа снова определяется максимальная часть, и процесс продолжается до тех пор, пока не получим m-планарное разбиение графа.

Для графа G' (рис. 3.65,*б*) подмножества Y' и Y'' имеют вид: $Y' = \{u_1, u_2, u_4, u_5\}$; $Y'' = \{u_6, u_7, u_8, u_3, u_9\}$. Граф, изображенный на рис. 3.65,*а*, планарен. Плоское его изображение имеется на рис. 3.65,*в*.

В настоящее время находит широкое применение использование поисковых методов для определения планарности. Опишем идею одного из таких алгоритмов. По графу схемы на основе поиска в глубину строится покрывающее дерево и производится упорядочение входящих в него ребер. Далее последовательно производится образование циклов путем добавления оставшихся ребер, причем, как только добавляемое ребро образует пересечения, производится поиск области, где размещение этого ребра пересечений не образует. Если такой области нет, то граф непланарен. Весь алгоритм состоит из ряда поисков в глубину, поэтому требуется $0(n)$ операций и объем необходимой памяти также $0(n)$. Отметим,

что алгоритмы такого типа не размещают графа на плоскости, а только исследуют его на планарность. Для размещения планарного графа на плоскости требуется $0(n^2)$ операций.

После минимизации суммарной длины ребер графа получим определенное размещение графа в сетке. Переразмещение вершин может привести к увеличению суммарной длины, что является нежелательным. Поэтому необходимо решать задачу определения планарности и m-планарного разбиения графа при фиксированном расположении вершин в узлах сетки. Так как граф $G=(X, U)$ является плоским, когда

$$P(G)=\sum_{u_{ij}\in U} P(u_{i,j})=0, \tag{3.83}$$

то число ребер, которые будут удалены из графа при выполнении условия (3.83), является оценкой числа планарности.

Идея алгоритма нахождения оценки числа планарности заключается в выделении на каждом шаге ребра с максимальным числом пересечений и удаления его на второй слой. Ребра удаляются из первого слоя на второй до тех пор, пока не выполняется условие (3.83). Если поставить задачу m-планарного разбиения, то аналогичные операции удаления ребер необходимо провести с ребрами второго и т. д. слоев, пока последний не окажется планарным.

В результате работы описанного алгоритма подсчета числа пересечений ребер графа, отображенного в сетку, получим матрицу пересечений ребер $\mathbf{P}_k=\|p_{i,j}\|$. Заметим, что при построении матрицы $\mathbf{P}_k$ кратность ребер можно не учитывать, так как это не влияет на планарность графа. Для нахождения ребра, которое необходимо удалить на первом шаге работы алгоритма, построим вектор-столбец $\mathbf{B}=\|b_i\|$, элементами которого являются числа пересечений ребер. Из $\mathbf{B}$ выберем $\max b_i$. Ребро, соответствующее выбранному элементу, переносится на второй слой. Элементы b_k, b_p, ..., b_r столбца уменьшим на единицу, если вынесенное ребро пересекалось с ребрами k, p, ..., r. Получим $B^1=\|b^1_i\|$. Аналогичным образом поступим с B^1, B^2, ..., B^m до тех пор, пока все элементы столбца b^m не будут равны 0. Таким образом, получим планарный слой.

Рассмотрим работу алгоритма на примере графа схемы, изображенного на рис. 3.66. Список пересечений имеет вид

$$\mathbf{P}_k=\left\|\begin{array}{ll}
u_{1,3}(u_{2,4}) & u_{5,7}(u_{1,10},\ u_{2,6},\ u_{3,14},\ u_{3,15},\ u_{1,6}) \\
u_{1,6}(u_{5,7}) & u_{6,9}(u_{1,10}) \\
u_{1,10}(u_{5,7},\ u_{6,9},\ u_{6,13},\ u_{9,12}) & u_{6,13}(u_{1,10},\ u_{9,12}) \\
u_{2,4}(u_{1,3}) & u_{7,16}(u_{8,11},\ u_{9,12}) \\
u_{2,6}(u_{5,7}) & u_{8,11}(u_{4,12},\ u_{7,16},\ u_{9,12}) \\
u_{3,14}(u_{5,7},\ u_{9,12}) & u_{8,16}(u_{4,12},\ u_{9,12}) \\
u_{3,15}(u_{5,7},\ u_{9,12}) & u_{9,12}(u_{1,10},\ u_{3,14},\ u_{3,15},\ u_{8,16},\ u_{8,11},\ u_{7,16},\ u_{6,13}) \\
u_{4,12}(u_{8,11},\ u_{8,16}) &
\end{array}\right\|$$

По матрице $\mathbf{P}_k$ построим вектор-столбец $\mathbf{B}_1$, из него выберем максимальный элемент, если их несколько, то берется любой:

	$\mathbf{B}_1$	$\mathbf{B}^1{}_1$	$\mathbf{B}^2{}_1$	$\mathbf{B}^3{}_1$	$\mathbf{B}^4{}_1$	$\mathbf{B}^5{}_1$	$\mathbf{B}^6{}_1$
$u_{1,6}$	1	1	0	0	0	0	0
$u_{1,3}$	1	1	1	1	1	1	[0]
$u_{1,10}$	4	3	2	2	[0]	0	0
$u_{2,4}$	1	1	1	1	1	1	0
$u_{2,6}$	1	1	0	0	0	0	0
$u_{3,14}$	2	1	0	0	0	0	0
$u_{3,15}$	2	1	0	0	0	0	0
$u_{4,12}$	2	2	2	[0]	0	0	0
$u_{5,7}$	5	5	[0]	0	0	0	0
$u_{6,13}$	2	1	1	1	0	0	0
$u_{7,16}$	2	1	1	1	1	[0]	0
$u_{8,11}$	3	2	2	1	1	0	0
$u_{8,16}$	2	1	1	0	0	0	0
$u_{9,12}$	7	[0]	0	0	0	0	0
$u_{6,9}$	1	1	1	1	0	0	0

Итак, выберем элемент $u_{9,12}$ и на его место запишем ноль, что соответствует переносу ребра $u_{9,12}$ на второй слой. По матрице $\mathbf{P}_k$ определим ребра, которые пересекались с ребром в столбце $\mathbf{B}_1$. Значения элементов, соответствующих этим ребрам, уменьшаются на единицу. Получим столбец $\mathbf{B}^1{}_1$ после первого шага алгоритма. Из столбца $\mathbf{B}^1{}_1$ выберем элемент $u_{5,7}$ и процедуру повторим пока не получим столбец $\mathbf{B}^6{}_1$, все элементы которого равны нулю.

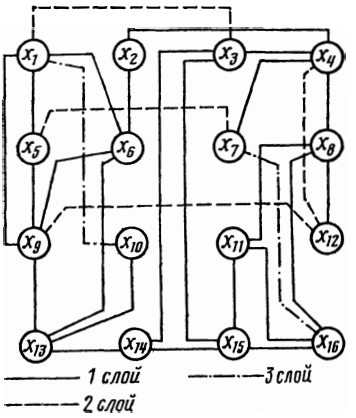


Рис. 3.66. Пример графа схемы

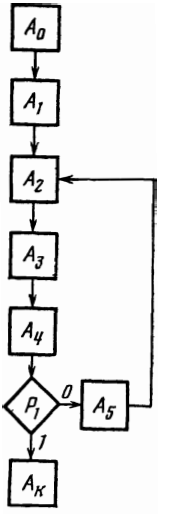


Рис. 3.67. Граф-схема алгоритма разбиения графа на плоские супрафы

На шестом шаге вынесем ребро $u_{1,3}$ и получим разбиение ребер графа на два слоя — планарный и слой, который надо проверить на планарность. На второй слой попало шесть ребер, следовательно, число планарности $\Theta(G)=6$.

Из матрицы R_k выберем ребра второго слоя и построим для них матрицу пересечений и вектор-столбец $\mathbf{B}_2$. Перенесем ребро $u_{1,10}$ на третий слой и запишем $\mathbf{B}^1{}_2$. После выноса ребра $u_{7,16}$ второй слой будет планарный. Получим разбиение графа на три плоских суграфа, изображенное на рис. 3.66.

Рассмотрим алгоритм разбиения графа на плоские суграфы, основанный на использовании условия Бадера и нахождения семейства максимальных внутренне устойчивых подмножеств множества вершин графа пересечений. Известно, что граф G планарный, если его граф пересечений G' является бихроматическим (двудольным). Указанный критерий справедлив, когда G имеет гамильтонов цикл. Будем теперь предполагать, что в графе он выделен. Такое предположение приводит к тому, что значительно сокращается процедура разбиения графа на плоские суграфы, однако в общем случае получается большее число слоев. По матрице смежности $\mathbf{R}$ графа G можно построить матрицу смежности графа пересечений G'. Заметим, что ребра графа G, которые не имеют пересечений, не учитываются при построении матрицы графа G', так как их можно расположить в любом из полученных плоских суграфов. В дальнейшем эти ребра не рассматриваются.

Для определения двудольности графа G' используется семейство независимых подмножеств $\Psi=\{\psi_\gamma\}$. Рассмотрим выделение в графе пересечений G' максимальных двудольных подграфов. Сначала с помощью семейства Ψ выделим из графа G' максимальный двудольный подграф H'. Если двудольным оказывается только подграф H' графа G', то соответственно плоским является суграф H графа G. Далее аналогично исследуется оставшаяся часть графа G'. В результате получаем разложение графа G на плоские суграфы.

Для выделения максимального двудольного подграфа H' графа G' введем критерий $\alpha_{\gamma,\delta}=|\psi_\gamma|+|\psi_\delta|-|\psi_\gamma\cap\psi_\delta|$, $\gamma, \delta\in\Gamma$. В соответствии с этим критерием граф G плоский, если

$$\alpha_{\gamma\delta}=|\psi_\gamma|+|\psi_\delta|=|U'|,\ |\psi_\gamma\cap\psi_\delta|=0.$$

Естественно, чем ближе $\alpha_{\gamma,\delta}$ к $|U'|$, тем большее число вершин содержит выделяемый двудольный подграф H'. Чтобы выбрать максимальный подграф, необходимо определить все $\alpha_{\alpha,\delta}$ при $\gamma, \delta\in\Gamma=\{1, 2, \ldots, s\}$, $s=|\psi|$. Тогда ψ_γ и ψ_δ, соответствующие максимальному $\alpha_{\gamma,\delta}$, определяют максимальный двудольный подграф H'. Выделение максимального $\alpha_{\gamma,\delta}$ удобно выполнять по матрице $\alpha=\|\alpha_{\gamma,\delta}\|$. Для получения последующих плоских суграфов исследуется подграф $\bar{H}=G-H$. По $\bar{H}$ можно построить граф пересечений $\bar{H}'$, для $\bar{H}'$ определить семейство $\Psi_{\bar{H}'}$, по семейству $\Psi_{\bar{H}'}$ образовать матрицу $\mathbf{R}_{\bar{H}'}$ и выделить из $\bar{H}'$ максимальный двудольный подграф.

Сформулируем граф-схему алгоритма разбиения графа на плоские суграфы (рис. 3.67). Здесь A_0, A_k — операторы начала и конца алгоритма соответственно; A_1 — построение графа пересечений G' по графу G; A_2 — определение семейства независимых подмножеств $\psi_{G'}$, графа G'; A_3 — построение матрицы α для семейства $\psi_{G'}$, определение в ней максимального $\alpha_{\gamma,\delta}$, ψ_γ, ψ_δ, образование двудольного подграфа H'; A_4 — получение $\bar{H}'$, построение $\Psi_{\bar{H}'}$; A_5 — замена G' на $\bar{H}'$; p_1 — логическое условие, проверяющее $\Psi_{\bar{H}'} = \varnothing$, если $p_1 = 0$, то переход к A_5, если $p_1 = 1$, то переход к A_k.

Пример. Пусть задан граф G, показанный на рис. 3.68,*а*, граф его пересечений показан на рис. 3.68,*б*, и матрица $\mathbf{R}_{G'}$ имеет вид

$$\mathbf{R}_{G'} = \begin{array}{c|ccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 2 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 5 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 6 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 7 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 9 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}.$$

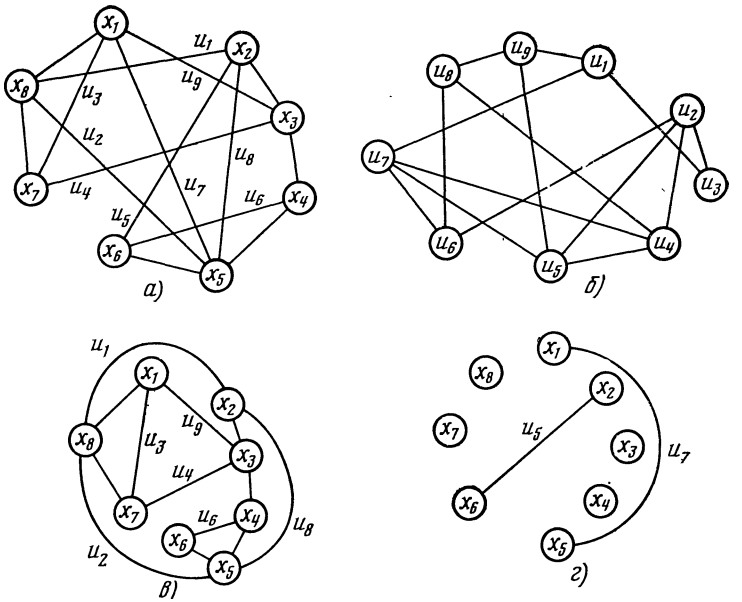


Рис. 3.68. Пример графа для определения пересечений (*а*); граф пересечений G^1 (*б*); первый плоский сграф (*в*); второй плоский сграф (*г*)

Определим семейство $\Psi_{G'} = \{\psi_1, \psi_2, ..., \psi_9\}$:

$$\psi_1 = \{u_1, u_2, u_8\}; \ \psi_2 = \{u_1, u_4, u_6\}; \ \psi_3 = \{u_1, u_5, u_6\};$$
$$\psi_4 = \{u_2, u_7, u_8\}; \ \psi_5 = \{u_2, u_9, u_7\}; \ \psi_6 = \{u_3, u_5, u_6\};$$
$$\psi_7 = \{u_3, u_7, u_8\}; \ \psi_8 = \{u_3, u_9, u_4, u_6\}; \ \psi_9 = \{u_5, u_8, u_1\}.$$

По семейству $\psi_{G'}$ построим матрицу

$$\alpha = \begin{array}{c|ccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \psi_1 & 3 & 5 & 5 & 4 & 5 & 6 & 5 & 7 & 4 \\ \psi_2 & & 3 & 4 & 6 & 6 & 5 & 6 & 5 & 5 \\ \psi_3 & & & 3 & 6 & 6 & 4 & 6 & 6 & 4 \\ \psi_4 & & & & 3 & 4 & 6 & 4 & 7 & 5 \\ \psi_5 & & & & & 3 & 6 & 5 & 6 & 6 \\ \psi_6 & & & & & & 3 & 5 & 5 & 5 \\ \psi_7 & & & & & & & 3 & 6 & 5 \\ \psi_8 & & & & & & & & 4 & 7 \\ \psi_9 & & & & & & & & & 3 \end{array}.$$

Определяем, что максимальное $\alpha_{\psi,\delta}=7$. Выбираем, например, $\alpha_{1,8}=7$. Ему соответствуют подмножества ψ_1 и ψ_8, определяющие первый суграф H, показанный на рис. 3.68,*в*.

Находим семейство $\Psi_{\overline{H}'} = \{\psi_1, \psi_2\}$, $\psi_1 = \{u_5\}$, $\psi_2 = \{u_7\}$. Матрица $\alpha_{\overline{H}'}$ имеет вид

$$\alpha_{\overline{H}'} = \begin{array}{c|cc} & 1 & 2 \\ 2 & 1 \end{array}.$$

Выбираем $\alpha_{1,2}=2$, соответствующее подмножествам ψ_1 и ψ_2, которые определяют второй плоский суграф графа G (рис. 3.68,*г*). В итоге получено разбиение исходного графа на два плоских суграфа.

Рассмотрим методы двухслойной трассировки. Появление достаточно экономичной технологической базы для производства интегральных микросхем и печатных плат со сверхплотной упаковкой и их широкое распространение повлекли пересмотр оценок конкретных алгоритмов трассировки. Так, количество непроведенных соединений не может служить характеристикой алгоритма без тщательного анализа конкретной конструкции, технологии и технических параметров схемы. Известные в настоящее время алгоритмы ориентированы на применение ручной дотрассировки непроведенных соединений. Однако возрастающая плотность монтажа, более эффективное использование ресурсов платы делают ручную дотрассировку трасс весьма трудоемкой. Поэтому задача совершенствования алгоритмов трассировки весьма актуальна.

Рассмотрим алгоритм, основанный на выборе связанных контактов из условия минимума суммы длин соединений в электрической цепи и построении трасс с помощью модифицированного волнового алгоритма. Интегральная микросхема состоит из совокупности ячеек. Ячейки располагаются на кристалле в линейках регулярно. К исходным данным также относятся геометрические параметры кристалла микросхемы и список цепей схемы. Соеди-

нения проводятся в двух ортогональных направлениях. Проведение соединений разрешено лишь в межъячеечном пространстве. Контактные переходы располагаются в произвольных местах площади кристалла. Пусть задана система горизонтальных и вертикальных равноудаленных линий, образующих двумерную евклидову решетку. Имеется множество $K=\{k_1, k_2, ..., k_n\}$ контактов цепи s, расположенных в узлах сетки. Если $d(k_i, k_j)$ — расстояние между элементами $k_i, k_j \in K$ в метрике сетки, то необходимо найти такое соединение элементов множества K, при котором

$$\sum_{i,\, j=1}^{n} d(k_i, k_j) = \min.$$

Точного решения задачи построения кратчайшего связывающего дерева для $n>3$ нет. Ниже приводится алгоритм, позволяющий получать приближенное решение указанной задачи. Будем называть отрезки (d_i, d_{i+1}) в G_r фрагментами. Путь длины d в сетке есть последовательность фрагментов Π^1: $(d_1, d_2)(d_2, d_3)...(d_{l-1}, d_l)$. Положим $d_1 \sim k_1$, $d_l \sim k_l$ $(l=1, ..., n)$. Следовательно, путь будет объединять два элемента $k_i, k_l \in K$, расположенных в сетке G_r. В общем случае путь может включать как конечные k_1, k_l, так и промежуточные контакты $k_j \in K$. Методика построения связывающего (покрывающего) дерева основана на определении квазиминимального пути Π между элементами $k_1, k_l \in K$, для которых $d(k_1, k_l) = \min$, и присоединений к этому пути остальных элементов множества K. Процесс построения представим в виде трех процедур.

Процедура 1. Помечается начальный элемент $k_1 \in K$, который выбирается произвольно или задается априорно, и для него определяется элемент k_2 такой, чтобы $d(k_1, k_2) = \min$ из всех $d(k_1, k_j)$, $j=2, ..., n$. На следующем шаге присоединенный элемент k_2 будет начальным и он также помечается. Элементы k_1 и k_2 составляют «фронт» Π_1 пути на данном шаге. Далее находится элемент k_3 из условия $k_3 \in \Pi$, $\underset{i=1,2}{d}(k_i, k_3) = \min$.

Условия включения элемента $k_j \in K$ в путь Π будет иметь вид

$$(\forall k_j \in \Pi_{j-1})[d(k_i, k_j) = \min,\ k_i \in \Pi_{j-2}]. \qquad (3.84)$$

Для избежания возникновения замкнутых циклов кроме (3.84) вводится условие: хотя бы один из контактов присоединяемого отрезка пути имеет отметки начального или конечного. Описанный процесс повторяется до тех пор, пока элемент k_l не будет помечен как конечный.

Процедура 2. Начиная с элемента k_l, по меткам возвращаемся в исходную точку, определив таким образом весь путь.

Процедура 3. Элементы, вошедшие в Π, образуют подмножество $K' \subset K$. Для элементов подмножества $K'' = K \setminus K^1$ определяется $\min d(k', k_i)$, $k_i \in K''$. Соответствующий элемент (контакт) присоединяется к пути. Эта процедура повторяется до получения $K'' = \varnothing$.

Пример. Пусть необходимо построить покрывающее дерево для элементов цепи, показанной на рис. 3.69. На нем изображены все возможные связи и условные расстояния между контактами. На первом шаге выберем связь k_1k_2. Контакт пометим как конечный на шаге 1 и как начальный на шаге 2. Значение расстояния между k_1 и k_2 уменьшается на величину $d(k_1k_2)=2$. На шаге 2 имеем три варианта продолжения пути: к контактам k_3, k_5 и к контакту k_4. Эти контакты пометим и перейдем к шагу 3. Далее продолжим построение пути до достижения k_7. Затем выполним процедуру 2, с помощью которой легко определить путь $П$: (k_1, k_2, k_3, k_4). Остальные элементы пути подсоединим по процедуре 3. Окончательный вид связывающего дерева приведен на рис. 3.70. Аналогичным образом строятся связывающие деревья для всех цепей схемы.

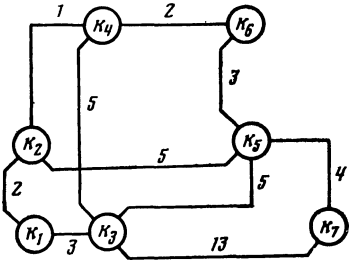


Рис. 3.69. Цепь схемы

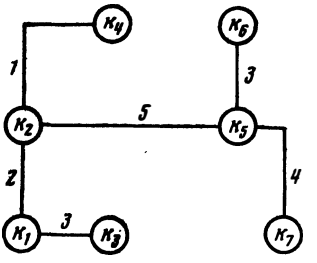


Рис. 3.70. Покрывающее дерево

Основными критериями для трассировки являются максимальная разводимость соединений в пределах возможностей алгоритма и соблюдение заданных ограничений. В рассматриваемом алгоритме распространение волны ограничено областью минимального прямоугольника, в которой расположены все контакты данного связывающего дерева. При этом оба слоя моделируются одновременно и трасса может переходить из слоя в слой, принимая вид «стежка». Это позволяет в максимальной степени сохранить конфигурацию связывающего дерева, построенного на предыдущем этапе конструирования.

Каждому элементу сетки G_r по мере его включения во фронт распространяющейся волны присваивается путевая координата. Определим множество $\Gamma^t{}_{x_j}$, $t=1, 2$, элементов $x_i \in G_r$, смежных узлу сетки x_j, в одном из двух слоев. Через $D^t(x_j)$ обозначим множество элементов $x_i \in G_r$, которым присвоены путевые координаты волны, распространяющейся от узла источника. Очевидно, что если на данном шаге узел $x_i \in \Gamma^t(x_j)$, то на следующем шаге продвижения волны он будет принадлежать множеству $D^t(t_j)$.

Введем правила:

1. Если для элемента x_i выполняются условия $x_i \in D^1(x_j)$ и $x_i \in D^2(x_j)$, то для включения x_i в множество $D^2(x_i)$ необходимо присвоить ему путевую координату $m^2{}_r = m^1{}_r + 1$, где $m^1{}_r$ — значение путевой координаты x_i в $D^1(x_j)$.

2. Если для элемента x_i выполняются условия $x_i \in D^2(x_j)$ и $x_i \in D'(x_j)$, то x_i включается в множество со значением путевой координаты $m^1{}_r = m^2{}_r$.

Правила дают возможность волнам, распространяющимся в обоих слоях, одновременно достигать цели, что приводит к сокращению цикла проведения трассы. После определения пути прохождения трассы в каждом слое проводится разнесение отрезков трассы по слоям. При выходе из конечной точки определяется направление трассы и выбирается тот слой (основной), в котором это направление разрешено. Переход в другой слой производится лишь в случае невозможности дальнейшего построения трассы или при изменении ее направления. В общем случае может быть произведено несколько межслойных переходов, прежде чем трасса будет полностью построена.

Рассмотрим пример построения трасс цепи, связывающее дерево которой показано на рис. 3.70. В обоих случаях имеются ранее проведенные соединения. Первыми строятся трассы, выходящие из контакта. Как видно на рис. 3.71, 3.72, трасса k_1, k_3 располагается в слое горизонталей, а k_1k_2 — в слое вертикалей. Далее

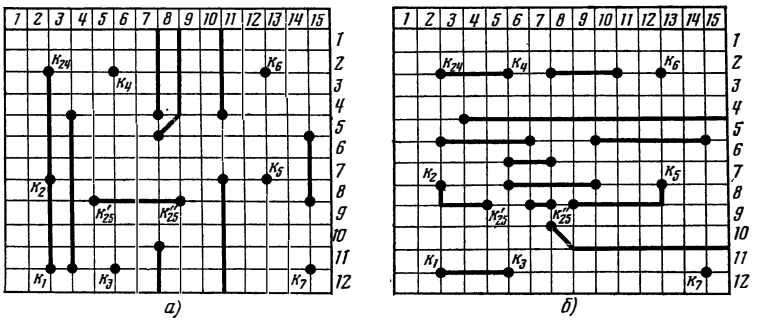


Рис. 3.71. Слой вертикалей (*а*); слой горизонталей (*б*)

волна распространяется из k_2 до контактов k_4 и k_5. Трасса k_2k_4 будет состоять из двух отрезков, поскольку при определении последовательности элементов, занятых этой трассой, в узле $k_{2,4}$ происходит изменение горизонтального направления на вертикальное. Следовательно, $k_{2,4}$ помечается как переход в другой слой. На рис. 3.71,*а*, *б* показаны также распространенные волны из k_2 в k_5 и вид трассы без коррекции.

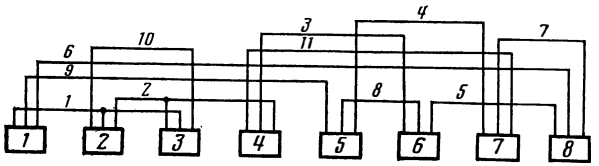


Рис. 3.72. Фрагмент схемы

Рассмотрим описание параллельно-последовательной трассировки двусторонних печатных плат с планарными выводами. Контактные площадки располагаются с четырех сторон платы. В алгоритмах используются критерии минимальной суммарной длины соединений, максимальной реализуемости и минимального числа пересечений. Поле платы разбивается на вертикальные и горизонтальные каналы. Пропускная способность каналов определяется числом магистралей, в которых размещаются трассы. Предварительно все цепи независимо друг от друга распределяются по каналам в одном слое по критерию минимума суммарной длины и максимальной реализуемости, т. е. по аналогии с «гибкой» трассировкой определяется конфигурация цепей без закрепления их в магистралях. Распределение по каналам осуществляется в два этапа. На первом этапе для каждой цепи строится кратчайшее связывающее дерево и определяются пары связных вершин. На втором этапе осуществляется равномерное распределение фрагментов цепей по каналам с тем, чтобы свести к минимуму число непроведенных соединений. На следующем этапе с учетом свойства инвариантности некоторых групп контактов осуществляется закрепление цепей за контактами микросхем по критерию максимально плотной упаковки. Это позволяет уменьшить число пересечений между цепями, подходящими к контактам элемента. Затем выполняется минимизация числа внутрисхемных пересечений. После этого осуществляются упаковка и размещение фрагментов цепей по магистралям. Одновременно производятся определение и печать списка непроведенных соединений. На последнем этапе осуществляется внесение изменений в трассировку по результатам ручной доработки, а также размещение и минимизация числа переходных отверстий.

Исходной информацией являются матрица цепей **Т** и множество цепей, выводимых на разъемы $Ш_i = \|w_{i,j}\|$. Размещение элементов отражено в матрице **Т**. В библиотеке конструктивных данных хранятся геометрические параметры платы.

Сначала по матрице **Т** осуществляется расчет координат вершин, к которым подходит заданная цепь. Далее строятся полный граф расстояний и его матрица, по которой одним из известных методов осуществляется построение кратчайшего связывающего дерева. Следующий этап может выполняться в двух вариантах. В первом строятся кратчайшие связывающие цепи. Во втором производится распределение фрагментов цепей по каналам с минимизацией числа непроведенных соединений. Каждый канал разбивается на дискреты. В горизонтальном канале дискретом будет промежуток между двумя соседними контактами, а в вертикальном — промежуток между двумя горизонтальными каналами. Для каждого фрагмента вводится множество позиций, в которые он может назначаться. В зависимости от того, в какую позицию назначается фрагмент, определяются дискреты, через которые он проходит. Для каждого фрагмента определяется множество вариантов размещения, из которых выбирается один с наименьшим чис-

лом изгибов. На следующем этапе осуществляется закрепление цепей за конкретными контактами. У каждого элемента есть инвариантные контакты. Этим можно воспользоваться для повышения плотности размещения. Далее выполняется минимизация внутрисхемных пересечений. На следующем этапе производятся упаковка и размещение фрагментов цепей внутри каждого канала по магистралям. После упаковки и печати списка непроведенных соединений предусмотрен этап ручной доработки. После доработки все изменения вносятся в модель платы. На следующем этапе производятся разнесение фрагментов цепей по слоям, определение и минимизация переходных отверстий. На последнем этапе производятся расчет координат фрагментов цепей и вывод информации на координатограф или дисплей с дальнейшим автоматизированным получением фотошаблонов.

Рассмотрим подробнее метод определения минимально необходимого числа каналов для расположения в них горизонтальных участков цепей при заданном размещении.

Исходная информация о схеме записывается в виде матрицы цепей $\mathbf{T}=\|t_{i,j}\|_{k\times n}$, где $t_{i,j}$ — номер цепи, проходящей через i-й контакт j-го элемента; $n=|X|$ — число элементов; k — число контактов элементов. После разбиения схемы на l подсхем, каждая из которых представляет некоторое множество ячеек, расположенных в одной линейке, по матрице $\mathbf{T}$ получаем $2l$ матриц цепей $\mathbf{T}^1{}_l$ и $\mathbf{T}^2{}_l$; $\mathbf{T}^1{}_l(\mathbf{T}^2{}_l)$ составлена по левой (правой) половине матрицы $\mathbf{T}$ для верхней (нижней) половины линейки. По матрице $\mathbf{T}$ построим матрицу цепей $\mathbf{T}'=\|t'_{i,j}\|_{2\mu\times r}$, где r — число цепей в схеме; μ — число линеек;

$$t'_{i,j}=\begin{cases}1, \text{ если цепь } j \text{ при } i \text{ четном инцидентна контактам}\\ \text{нижней части линейки, номер которой } i/2;\\ \text{при } i \text{ нечетном — контактам верхней части линей-}\\ \text{ки, номер которой } (i+1)/2,\\ 0 \text{ в противном случае.}\end{cases}$$

Контакты ячеек, расположенные в одной линейке, пометим индексами a, если контакт инцидентен цепи, проведенной по горизонтали, только вправо от него; b — если контакт инцидентен цепи, проведенной только влево. Разметку можно выполнить по матрице $\mathbf{T}_l$, воспользовавшись вектором-строкой $\mathbf{H}=\|H_t\|$, где

$$H_t=\begin{cases}1, \text{ если произвольный контакт, инцидентный цепи,}\\ \text{помечен индексом;}\\ 0 \text{ в противном случае.}\end{cases}$$

Индексы контактов занесем в матрицы $\mathbf{T}_l$, проставив их рядом с номерами цепей, инцидентных помеченным контактам.

Например, для фрагмента схемы, изображенного на рис. 3.72, матрица $T^1{}_l$ имеет вид

$\mathbf{T}^1{}_l =$

	1	2	3
1	1	9	6
2	10	1	2
3	2	1	10
4	11	3	2
5	9	8	4
6	3	8	5
7	4	7	11
8	5	6	7

После индексации матрица

$\mathbf{T}^{1*}{}_l =$

1	1*a*	9*a*	6*a*
2	10*a*	1	2*a*
3	2	1*b*	10*b*
4	11*b*	3*a*	2*b*
5	9*b*	8*a*	4*a*
6	3*b*	8*b*	5*a*
7	4*b*	7*a*	11*b*
8	5*b*	6*b*	7*b*

Рассмотрим верхнюю часть некоторого элемента с помеченными контактами (рис. 3.73). Заметим, что если контакт, помеченный индексом a, расположен справа от контакта, помеченного индексом b, то горизонтальные участки этих цепей, будучи расположенными в одном канале, не пересекаются. Чем большее число таких пар контактов можно составить, тем меньше потребуется каналов.

Кроме того, если образовать пару между контактом z_j ячейки, помеченным индексом b, и ближайшим, расположенным справа от него и не вошедшим в состав другой пары контактом, помеченным индексом a, то число пар таких контактов будет максимальным.

Запишем алгоритм разбиения множества горизонтальных участков цепей f^{l_1} на минимальное число подмножеств таких, что $\bigcup_i f^{l_1}{}_i = f^{l_1}$ и $f^{l_1}{}_i \cap f^{l_1}{}_j = \varnothing$.

1°. Присваиваем индексу i, определяющему номер формируемого подмножества, значение, равное единице.

2°. Просматриваем слева направо, начиная с первого контакта первой ячейки, контакты верхней части линейки до тех пор, пока не встретится контакт, помеченный индексом a и не вошедший в состав никакой пары. Определяем по индексированной матрице $\mathbf{T}^{1*}_l$ номер цепи, инцидентной этому контакту. Если такого контакта нет, то переходим к 7°. Переход к 3°.

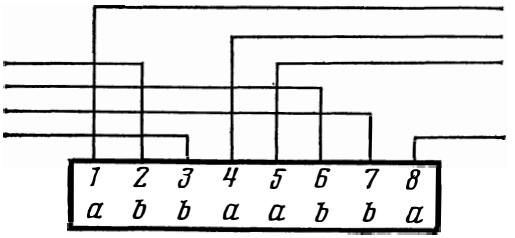


Рис. 3.73. Верхняя часть элемента

3°. По матрице $\mathbf{T}^{1*}_l$ отыскиваем контакт, инцидентный той же цепи, но помеченный идексом b.

4°. Включаем участок цепи в состав подмножества $f^{l_1}_i$.

5°. По матрице $\mathbf{T}^{1*}_l$ отыскиваем ближайший, расположенный справа от контакта, найденного в 3°, контакт, помеченный индексом a, определяем номер цепи, инцидентной этому контакту. Если такого контакта нет, то переходим к 6°. Иначе переход к 3°.

6°. Увеличиваем порядковый номер множества $f^{l_1}_i$ на единицу, т. е. $i := 1+i$. Переход к 2°.

7°. Конец работы алгоритма.

Например, с помощью данного алгоритма сформируем подмножества горизонтальных участков цепей $f^{l_1}_i$ для фрагмента схемы, приведенного на рис. 3.72. Воспользуемся приведенной выше для этой схемы матрицей $\mathbf{T}^{1*}_l$. В соответствии с алгоритмом образуется пять подмножеств: $f^{l_1}_1=\{1, 11\}$; $f^{l_1}_2=\{9, 8, 5\}$; $f^{l_1}_3=\{6\}$; $f^{l_1}_4=\{10, 3, 7\}$; $f^{l_1}_5=\{2, 4\}$. Следовательно, для размещения горизонтальных участках цепей потребуется пять каналов. На рис. 3.74 представлена схема с минимизированным числом каналов. Здесь $f^{l_1}_5$ размещено в первом канале, $f^{l_1}_1$ — во втором, $f^{l_1}_4$ — в третьем, $f^{l_1}_2$ — в четвертом, $f^{l_1}_3$ — в пятом.

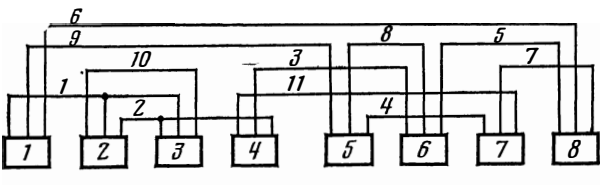


Рис. 3.74. Фрагмент рис. 3.72 после минимизации

Если цепь t инцидентна контактам верхней и нижней частей линейки, необходимо проведение вертикальной трассы, связывающей участки $f^{l_1}_t$ и $f^{l_2}_t$, инцидентные контактам верхней и нижней частей линейки. Проведение вертикальных участков производится до размещения по каналам горизонтальных участков.

В процессе распределения цепей и их фрагментов по магистралям может оказаться, что пропускная способность канала не позволяет поместить все фрагменты. Таким образом, появляются цепи, для которых необходима ручная доработка. Фрагменты цепей, назначенные в канал, можно разбить на два класса: к первому относятся фрагменты цепей, которые можно перенести в соседний канал без изменения «качества» трассы (например, увеличения длины трассы, числа изгибов и т. п.); ко второму — фрагменты цепей, которые в пределах заданных ограничений на конфигурацию цепи могут быть проведены только в одном канале. Поэтому целесообразно проводить в первую очередь фрагменты, относящиеся ко второму классу, а для остальных предусмотреть хранение запасных вариантов конфигурации цепи.

Пусть задано множество F цепей схемы. Цепь e представляет собой подмножество фрагментов $e=\{f_1, f_2, \ldots, f_p\}$. Задача распределения фрагментов цепей по каналам сводится к разбиению F на подмножества по критерию минимума числа непроведенных соединений. Разбиение производится на этапе построения кратчайших связывающих деревьев и представляет собой на каждом шаге приведение фрагмента цепи $f_i \in e$ в соответствие рассматриваемому каналу.

Представим e как расплывчатое множество $\tilde{e}=\bigcup_{j=1}^{v} \tilde{e}_j$, где j — номер варианта конфигурации цепи e; v — число рассматриваемых вариантов; $\tilde{e}_j$ — расплывчатое подмножество фрагментов j-го варианта: $\tilde{e}_j=\{\langle\mu_e(f^j_1), f^j_1\rangle, \langle\mu_e(f^j_2), f^j_2\rangle, \ldots, \langle\mu_e(f^j_p), f^j_p\rangle\}$. Если при построении дерева выбирается только один вариант с номером q, а остальные отбрасываются, то

$$\mu_e(f^j_p)=\begin{cases}1, \text{ если } j=q;\\ 0 \text{ — в противном случае}\end{cases}$$

и $\tilde{e}$ превращается в обычное конечное множество. Чем больше существует вариантов, тем меньше должна быть степень принадлежности отдельного фрагмента. Для определения степени принадлежности фрагмента цепи используется выражение $\mu_e(f^j_p)=|e|/v$. Идея алгоритма заключается в параллельном рассмотрении нескольких вариантов конфигурации для каждой цепи и установлении очередности распределения фрагментов цепей в канале. Для каждого фрагмента цепи определяется $\mu_e f^j_p$. В дальнейшем при распределении фрагментов в канале очередность проведения определяется степенью принадлежности. В первую очередь производится реализация фрагментов, общих для всех вариантов цепи. Затем реализуются фрагменты с меньшей степенью принадлежности. В результате для некоторых цепей оказывается возможной одновременная реализация двух или нескольких вариантов конфигурации. Трудоемкость алгоритма $0(|e|^2)$.

Рассмотрим трассировку многослойных печатных плат (МПП). При этом необходимо сведение к минимуму

длины проводников, идущих параллельно в разных слоях. Наличие таких проводников приводит к паразитным наводкам и препятствует повышению быстродействия схемы.

Исходными данными на предварительном этапе трассировки являются результаты размещения элементов на плате, список связей схемы, геометрические размеры интегральных микросхем и платы, на которую наложена координатная сетка.

На основе экспериментальных данных считается, что целесообразно начинать проведение трасс с кратчайших соединений, переходя затем к трассировке следующих по длине связей. Это даст повышение плотности монтажа. Поэтому после определения координат соединительных выводов элементов производится упорядочение связей по длине. Далее все связи из матрицы цепей или списка связей схемы распределяются в четыре самостоятельных списка. Условно каждое соединение разбивается на две составляющие — ортогональную и диагональную (идущую под углом 45° к первой). Такое разложение связи продиктовано условием проведения соединений в слоях лишь в ортогональных и диагональных направлениях.

После упорядочения и разнесения связей по спискам начинается проведение трасс. Трассировка печатных соединений производится методом распространения числовой волны на дискретном поле модели платы. Алгоритм представляет собой систему из вспомогательного алгоритма разнесения и упорядочения связей и алгоритма проведения трасс. Упорядочение связей по длине сводится к определению их длин с дальнейшим расположением связей в списках в порядке возрастания величины $d_{i,j}$. Затем производится разнесение связей по слоям. Распределение цепей без пересечений с минимизацией числа слоев эквивалентно минимизированной раскраске графа пересечений схемы G', который строится по цепям схемы, расположенным в одном слое: $G'=(U, V)$, где U — множество вершин графа, соответствующее множеству цепей схемы. Ребро $v \in V$ существует и соединяет две вершины u_1, $u_2 \in U$, если соответствующие цепи пересекаются. После раскраски цепи, соответствующие одноцветным вершинам, помещаются в один слой. Алгоритм основан на последовательном сокращении подмножества вершин для размещения на данном шаге. Из итогового подмножества выбирается вершина и помещается в один из одноцветных классов. Затем строится новое подмножество вершин и так далее до размещения всех вершин.

Пусть имеется множество B распределенных по одноцветным классам вершин и множество A не распределенных к данному шагу вершин, причем $A \cup B = U$. Пусть на B задано разбиение $\overset{\circ}{B}$ вершин на одноцветные классы.

1°. Строится подмножество A_1, состоящее из тех вершин $u \in A$, которые могут быть помещены в наименьшее число одноцветных классов из $\overset{\circ}{B}$.

2°. Строится подмножество A_2, состоящее из тех вершин $u \in A$, которые максимально связаны с вершинами из A.

3°. Образуется подмножество A_3 из тех вершин $u \in A_2$, которые минимально связаны с A_2. Если вершины A_1 были смежны всем классам из $\overset{\circ}{B}$, то любая вершина из A_3 выбирается как новая компонента в $\overset{\circ}{B}$ с очередным цветом и исключается из A. Далее идет переход к формированию A_1. В противном случае продолжаются следующие построения.

4°. Определяется разбиение $\overset{\circ}{B}_1$ на множестве B_1, состоящее из всех тех компонент разбиения $\overset{\circ}{B}$, которым не смежна хотя бы одна вершина из A_3.

5°. Определяется разбиение $\overset{\circ}{B}_2$, состоящее из тех компонент, число вершин в которых минимально.

6°. Определяется подмножество A_4 таких вершин из A_3, которые могут быть помещены в наименьшее ненулевое число классов из $\overset{\circ}{B}_2$.

7°. Определяется подмножество вершин A_5 из A_4 с минимальной смежностью с вершинами из A_4. Из подмножества A_5 выбирается произвольная вершина и относится к любому ей не смежному классу из B_2. Вершина исключается из A. Далее идет переход к формированию A_1 (1°), если A не пусто, иначе конец работы.

Реализация этой схемы алгоритма обеспечивает степенную оценку трудоемкости. Для графов с ограниченной средней локальной степенью трудоемкость близка к $0(n)$.

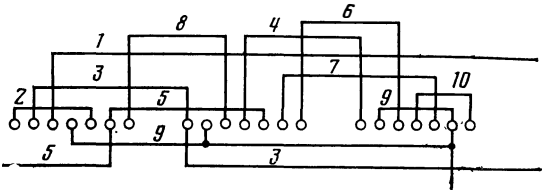


Рис. 3.75. Фрагмент схемы

Пример. На рис. 3.75 приведен фрагмент схемы. На рис. 3.76 показан граф пересечений цепей этого фрагмента. Классов B в начале работы алгоритма нет, поэтому выбираем подмножество вершин с максимальной локальной степенью. Согласно 2° это будет $A2 = \{u_1, u_3, u_9\}$. При выполнении 3° выясняем, что для вершины u_3 число ребер, соединяющих ее с вершинами из подмножества $A2$, равно 2, а для u_1 и u_9 равно 1. Поэтому вершину u_3 вы-

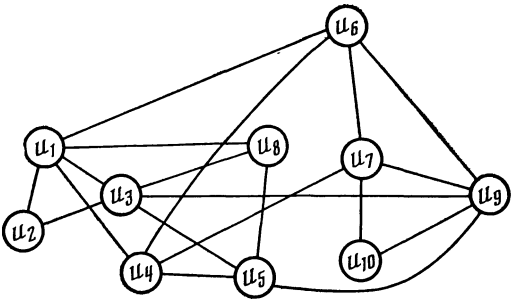


Рис. 3.76. Граф пересечений

черкиваем. Классов, не смежных вершинам из A_2, нет, поэтому организуется класс B_1 с вершиной u_1, т. е. $B_1=\{u_1\}$, $B=\{B_1\}$.

Далее среди вершин $A=U\setminus\{u_1\}$ определяем смежные максимальному числу классов из B, т. е. смежные вершине u_1: $A_1=\{u_2, u_3, u_4, u_6, u_8\}$. Среди этих вершин согласно 3° определяем те, которые имеют максимальное число ребер с вершинами из A. Тогда $A_2=\{u_3\}$ и вершина u_3 образует новый класс: $B_2=\{u_3\}$; $B=\{B_1, B_2\}$. Аналогично для вершин $A=U\setminus\{u_1\cup u_3\}$ получаем $A_1=\{u_2, u_4\}$; $A_2=\{u_4\}$ и вершина u_4 образует класс $B_3=\{u_4\}$ $B=\{B_1, B_2, B_3\}$. Для вершин $A=U\setminus\{u_1\cup u_3\cup u_4\}$ имеем $A_1=\{u_2, u_5\}$, затем $A_2=\{u_5\}$, после чего $B_1=\{B_1\}$. В итоге имеем $B_1=\{u_1, u_5\}$. Аналогично получаем изменение классов по шагам: $B_3=\{u_4, u_8\}$; $B_2=\{u_3, u_6\}$; $B_1=\{u_1, u_5, u_7\}$; $B_3=\{u_2, u_4, u_8\}$; $B_2=\{u_3, u_6, u_{10}\}$. Затем распределяем последнюю вершину u_9. Она попадает в класс B_3: $B_1=\{u_1, u_5, u_7\}$; $B_2=\{u_3, u_6, u_{10}\}$; $B_3=\{u_2, u_4, u_8, u_9\}$.

Конечная раскраска имеет вид, изображенный на рис. 3.77 в виде трехдольного графа. Цепи трех слоев изображены на рис. 3.78.

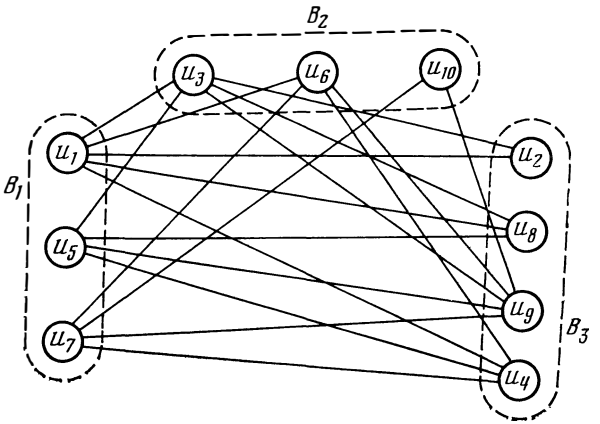


Рис. 3.77. Трехдольная раскраска

В результате работы алгоритма получим три таблицы соединений, элементы которых соответствуют связям одного слоя. Далее выполняется алгоритм трассировки, который повторяется до тех пор, пока не будут проведены все трассы данного слоя. После этого производится переход к трассировке следующего слоя.

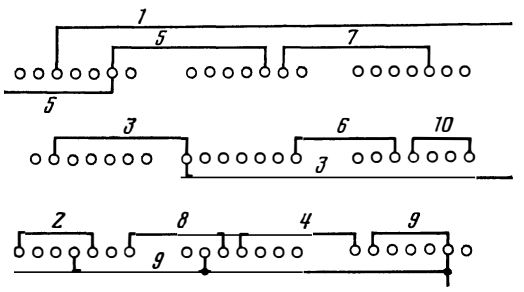


Рис. 3.78. Цепи трех слоев

Традиционный подход к трассировке соединений в минимальном числе слоев заключается в последовательном заполнении слоев цепями и применении к каждой цепи алгоритма построения кратчайшего связывающего дерева. Число слоев определяется очередностью размещаемых и трассируемых цепей.

Рассмотрим канальный алгоритм трассировки МПП. Предварительно все цепи размещаются по каналам в одном слое по критерию минимума суммарной длины соединений. Затем фрагменты цепей в пределах канала упорядочиваются относительно друг друга по критерию минимума числа пересечений. Далее производится раскраска вершин графа пересечений в минимальное число цветов, причем одним цветом помечаются не связанные между собой вершины, что соответствует разнесению цепей схемы в наименьшее число слоев. На следующих этапах производятся упаковка и размещение фрагментов цепей по магистралям, расчет координат фрагментов цепей и вывод информации на периферийные устройства.

Поле платы разбивается на вертикальные и горизонтальные каналы, в которых размещаются элементы и проводятся трассы. Межслойные переходы осуществляются только по контактам элементов, поэтому каждая цепь полностью трассируется в одном из слоев.

На основании электрической схемы составляется следующая исходная информация: матрица $\mathbf{T}=\|t_{i,j}\|$, вектор $\mathbf{Ш}=\|w_i\|$ — множество цепей, выводимых на контрольные гнезда. Множество контактов $C_1=\{C_j \mid j\in У_1,\ У_1=\{1, 2, ..., n\}$, ячейки, расположенной в

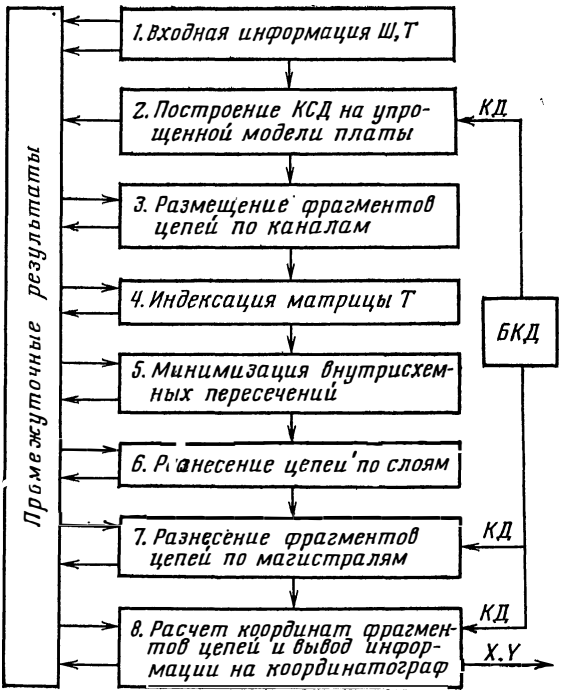


Рис. 3.79. Структурная схема комплекса алгоритмов трассировки

горизонтальном канале, образует ряд, ограничивающий горизонтальный канал снизу, а множество контактов $C_2=\{C_j \mid j \in У_2$, $У_2=\{h, h+1, ..., 2h\}$, — сверху. В матрице **Т** строки, соответствующие номерам ячеек, располагаются в том же порядке, в каком элементы размещены на плате.

Для некоторого фрагмента схемы матрица **Т** имеет вид

$\mathbf{T} =$

	1	2	3	4	5	6
1	5	2	5	2	2	1
2	6	5	2	5	2	8
3	3	9	4	8	5	3
4	4	9	5	5	3	6
5	9	6	3	6	7	9
6	10	9	3	6	7	1
7	7	9	10	7	1	10
8	14	11	10	7	14	11
9	4	3	1	3	13	8
10	12	11	4	13	14	4
11	11	4	2	2	14	13
12	13	12	12	2	13	12

Ячейки x_1—x_4 образуют 1-ю линейку; x_5—x_8 — 2-ю и x_9—x_{12} — 3-ю; k_1—k_3 — контакты нижнего ряда; k_4—k_6 — верхнего. Слева от x_i—x_j расположен вертикальный канал a_i.

В библиотеке конструктивных данных (БКД) хранятся геометрические параметры платы: размеры; координаты контактных мест, к которым подводятся цепи и по которым осуществляются межслойные переходы; координаты контрольных гнезд; координаты гнезд электрического соединителя; места размещения элементов; число элементов в линейке; число линеек; число магистралей в каналах и их координаты. Все размеры даются в системе координат платы. На рис. 3.79 приведена структурная схема комплекса алгоритмов трассировки МПП. С помощью алгоритмов блока *1* осуществляется ввод обязательной входной информации — матрицы **Т** и векторов **Ш**. Задача размещения цепей по каналам решается в два этапа. Сначала алгоритмами блока *2* строятся кратчайшие связывающие деревья (КСД). Построение кратчайших связывающих деревьев и размещение фрагментов цепей по кана-

лам производятся на упрощенной модели платы — матрице $M_t = \|m^t_{l,r}\|$, где

$$m^t_{l,r} = \begin{cases} 1, \text{ если фрагмент цепи } t \text{ расположен на пересечении вертикального канала } a_r \text{ и горизонтального } b_l; \\ 0 \text{ в противном случае.} \end{cases}$$

Горизонтальный участок q^t_l, расположенный в канале b_l и пересекающий канал $a_{r_1}—a_{r_2}$, в матрице моделируется рядом единиц, расположенных на пересечении l-й строки и $r_1—r_2$ столбцов. Если фрагмент цепи t инцидентен ячейке x_r, расположенной в канале b_l, то двум элементам m_{l_r} и $m_{l(r+1)}$ присваиваются единичные значения.

Матрица $\mathbf{M}_1$, построенная для первой цепи по матрице $\mathbf{T}$, имеет вид

$\mathbf{M}_1 =$

	a_1	a_2	a_3	a_4	a_5
b_1	1	1	0	0	0
b_2	0	1	1	1	0
b_3	1	1	0	0	0

.

Пример для цепи t_1 показан на рис. 3.80. Дерево строится по алгоритму Прима. Для отыскания кратчайшего расстояния используются волновые методы. В качестве шага распространения волны берется расстояние между двумя соседними каналами. Для записи информации о кратчайшем связывающем дереве используется матрица $\mathbf{D}_t = \|d^t_{i,j}\|$, где

$$d^t_{i,j} = \begin{cases} 1, \text{ если участки } q^t_i \text{ и } q^t_j \text{ цепи } t \text{ связаны;} \\ 0 \text{ в противном случае.} \end{cases}$$

Матрица $\mathbf{D}_1$ для цепи t_1 имеет вид

$\mathbf{D}_1 =$

	q^1_1	q^1_2	q^1_3
q^1_1	0	1	0
q^1_2	1	0	1
q^1_3	0	1	0

.

Размещение фрагментов цепей по каналам производится алгоритмами блока *3*. Участки можно соединять несколькими кратчайшими трассами. Из этого множества выбираются две, дающие наименьшее число изгибов. Для каждой из этих трасс в матрице $\mathbf{M}_t$ значения элементов $m^t_{l,r}$, моделирующих трассу, увеличиваются на единицу. После проведения всех возможных трасс выбирается та, для которой сумма значений элементов $m^t_{l,r}$, моделирующих ее в матрице $\mathbf{M}_t$, максимальна. Для записи результатов раз-

мещения фрагментов цепей по каналам матрица **Т** расширяется увеличением числа столбцов. Границы распространения фрагментов цепей в каналах помечаются индексами. Расширенная и проиндексированная матрица **Т*** является моделью платы и содержит всю информацию о размещении фрагментов цепей по каналам. Индексация матрицы цепей **Т*** осуществляется алгоритмом бло-

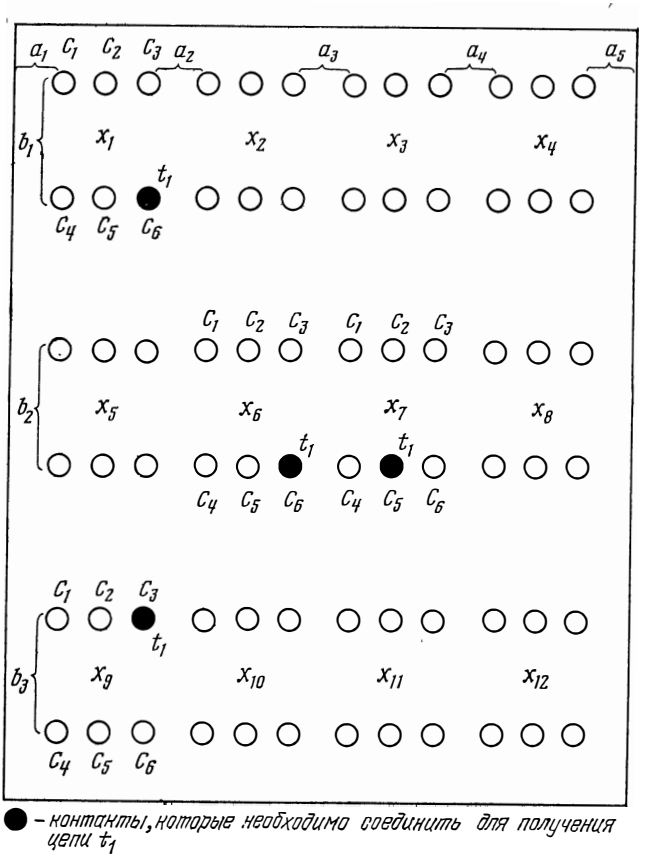


Рис. 3.80. Пример распределения каналов для построения цепи t_1

ка *4*. Из матрицы **Т*** выделяется подматрица $\mathbf{T}^*_l$, составленная из строк, соответствующих элементам, расположенным в l-й линейке. В подматрице $\mathbf{T}^*_l$ последовательно, начиная с первой, просматриваются пары элементов, соответствующие парам мест контактов и элементов.

На следующем этапе в блоке 5 производится упорядочение фрагментов цепей в пределах одного канала по критерию минимизации числа пересечений. Далее алгоритмом блока *6* производится разнесение цепей по слоям с помощью алгоритма раскраски

вершин графа пересечений. После разнесения цепей по слоям в каждом слое алгоритмом блока 7 производятся уплотнение и размещение фрагментов цепей по каналам. На последнем этапе по матрице **Т*** и по векторам **Ш** производятся расчет координат фрагментов цепей и вывод информации на координатограф или другое периферийное устройство.

3.4. КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Что называется компоновкой схемы ЭА?
2. Сформулируйте постановку задачи компоновки как разбиения графа на части.
3. Приведите основные критерии компоновки.
4. Дайте определение упорядоченного и неупорядоченного разбиений.
5. Приведите классификацию алгоритмов разбиения.
6. Опишите алгоритм покрытия схем ЭА.
7. Приведите последовательные алгоритмы разбиения графа схемы на части.
8. Дайте определение минимального и квазиминимального массивов в графе и постройте расплывчатый алгоритм разбиения графа схемы на произвольное число частей.
9. Приведите алгоритм определения клик графа и покажите его использование для разбиения графа.
10. В чем сущность итерационных методов разбиения графа?
11. Приведите формулы для определения перестановочных коэффициентов при разбиении.
12. Опишите дихотомический алгоритм разбиения графа схемы на части.
13. Дайте определение чисел связности.
14. Как используется метод случайных назначений при разбиении графов схем?
15. Постройте схему метода ветвей и границ для разбиения графа схемы на части.
16. Постройте структурные схемы алгоритмов для разбиения графа схемы на части на основе поиска в глубину и ширину.
17. Приведите постановку задачи размещения элементов схем ЭА.
18. Приведите классификацию алгоритмов размещения.
19. Постройте структурную схему алгоритма для определения оценки минимальной суммарной длины ребер графа.
20. Приведите формулы для определения средней длины ребра и координат центра тяжести вершин графа.
21. Опишите последовательно-итерационный алгоритм размещения. Приведите пример.
22. Постройте логическую схему алгоритма для факторизации графа по строкам и столбцам.
23. Опишите схему метода ветвей и границ для линейного размещения графа схемы с минимизацией суммарной длины соединений.
24. Приведите алгоритм подсчета числа пересечений по матрице смежности графа.
25. Опишите метод подсчета числа пересечений при размещении вершин графа в узлах прямоугольной сетки.
26. Постройте граф-схему алгоритма для минимизации пересечений ребер графа схемы.
27. Опишите схему метода ветвей и границ для минимизации внутрисхемных пересечений.
28. Постройте структурную схему алгоритма минимизации суммарной длины соединений на основе поиска в ширину.
29. Постройте структурную схему алгоритма минимизации внутрисхемных пересечений на основе поиска в глубину.
30. Опишите идею алгоритма для совместной минимизации суммарной длины и пересечений ребер графа.

31. Постройте алгоритм Ван Хао для размещения элементов схемы на основе алгоритмов разбиения графа схемы на части.
32. Сформулируйте постановку задачи трассировки.
33. Опишите основные этапы трассировки.
34. Приведите алгоритм Прима. Постройте пример.
35. Опишите задачу Штейнера.
36. Каким образом определяется порядок трассировки соединений?
37. Опишите волновые алгоритмы трассировки.
38. Постройте структурную схему алгоритма лучевой трассировки.
39. Постройте логическую схему алгоритма канальной трассировки.
40. Постройте граф-схему алгоритма гибкой трассировки.
41. Опишите алгоритм разбиения графа схемы на плоские суграфы.
42. Дайте определение числа планарности графа схемы.
43. Приведите пример m-планарного разбиения графа схемы.
44. Опишите методы трассировки двухслойных печатных плат.
45. Опишите методы трассировки многослойных печатных плат.
46. Опишите идею построения алгоритмов совместного размещения элементов и трассировки соединений.

4. ПРОЕКТИРОВАНИЕ ТОПОЛОГИИ И ФОТОШАБЛОНОВ ИНТЕГРАЛЬНЫХ МИКРОСХЕМ

4.1. О ПРОЕКТИРОВАНИИ ТОПОЛОГИИ ИНТЕГРАЛЬНЫХ МИКРОСХЕМ

В настоящее время при разработке устройств на ИМС использование ЭВМ становится необходимым. При машинной разработке ИМС, содержащих десятки и сотни тысяч транзисторов на одном или нескольких кристаллах, должна быть гарантирована возможность вмешательства разработчика для контроля получаемых на различных стадиях решений.

Исходными данными для проектирования топологии ИМС являются: принципиальная схема ИМС; типовой технологический процесс, на базе которого должна быть изготовлена ИМС.

Результатом этапа топологического проектирования является информация, управляющая работой тех или иных технологических автоматов вычерчивания фотошаблонов ИМС.

Необходимыми условиями при разработке варианта топологического решения являются: отсутствие ошибок проектирования; получение заданных электрических характеристик; минимизация площади кристалла.

На современном этапе проектирование топологии ИМС осуществляется в основном тремя методами: проектированием на приборном уровне, т. е. из элементарных компонентов; проектированием из типовых элементов; проектированием на основе базового кристалла.

При разработке ИМС первым методом на кристалле размещают транзисторы, резисторы, конденсаторы, структура которых проектируется отдельно, а затем выполняются соединения элементов. Для МДП ИМС проектирование топологии отдельных элементов и всего кристалла происходит одновременно. Такой подход обеспечивает наиболее качественное проектирование тополо-

гии как с точки зрения электрических характеристик схемы, так и с точки зрения занимаемой площади кристалла. Метод требует больших временных затрат, процесс проектирования трудно формализуем, что затрудняет применение автоматических методов проектирования. Поэтому данная методика применяется при проектировании ИМС, выпускаемых большими партиями.

Проектирование топологии ИМС из типовых элементов (логических схем, типовых динамических ячеек и т. д.) чаще всего применяется при разработке ИМС, выполняемых по заказам и небольшими партиями: на кристалле размещаются типовые элементы и соединительные проводники. При этом значительно снижаются время и стоимость проектирования. Такой процесс проектирования может быть формализован и, следовательно, осуществлен с помощью методов машинного решения задач размещения и трассировки. Однако использование данного метода проектирования ведет к увеличению площади кристалла в 1,5—2 раза.

Третий метод проектирования топологии также применяется при разработке ИМС, выпускаемых небольшими партиями. Базовый кристалл — это полуфабрикат с набором типовых изолированных элементов, причем слой металлизации отсутствует. В связи с требуемой логикой работы схемы осуществляется проектирование слоя металлизации. Такой способ крайне выгоден с точки зрения временных затрат, однако неэкономичен с точки зрения использования площади кристалла.

После разработки топологии выполняются проектирование фотошаблонов и их изготовление на микрофотонаборном оборудовании, для которого управляющей информацией являются большие числовые массивы координат послойной геометрии кристалла. Получить вручную эти числовые массивы уже для ИМС3 весьма затруднительно. Поэтому на этапе изготовления фотошаблонов ИМС необходимо применение автоматизированных методов.

Опыт разработки и внедрения систем автоматизированного проектирования фотошаблонов (САПФ) печатных плат позволил надеяться на создание аналогичных САПФ ИМС. Несмотря на то, что высокая плотность компонентов на кристалле ИМС внесла серьезные затруднения в процесс автоматизации труда разработчика, ЭВМ нашли широкое применение при проектировании их топологии и фотошаблонов, хотя, как правило, ИМС, разработанные с помощью САПФ, имеют несколько бо́льшую площадь кристалла.

Применение ЭВМ в проектировании и изготовлении ИМС пошло по пути создания интерактивных систем, в которых решающая роль отводится конструктору-специалисту. Создание таких систем при изготовлении и разработке фотошаблонов ИМС позволило: сократить сроки проектирования; снизить стоимость проекта; повысить качество проектирования.

В настоящее время САПФ ИМС решают следующие задачи (рис. 4.1):

описание топологического чертежа на языке разработчика;

выявление и исправление ошибок, допущенных в описании топологического чертежа;

трансляция, т. е. перевод описания топологического чертежа на язык конкретной технологической установки, изготовляющей фотошаблоны;

полный контроль топологического чертежа;

связь конструктора с ЭВМ на протяжении всего процесса проектирования фотошаблонов.

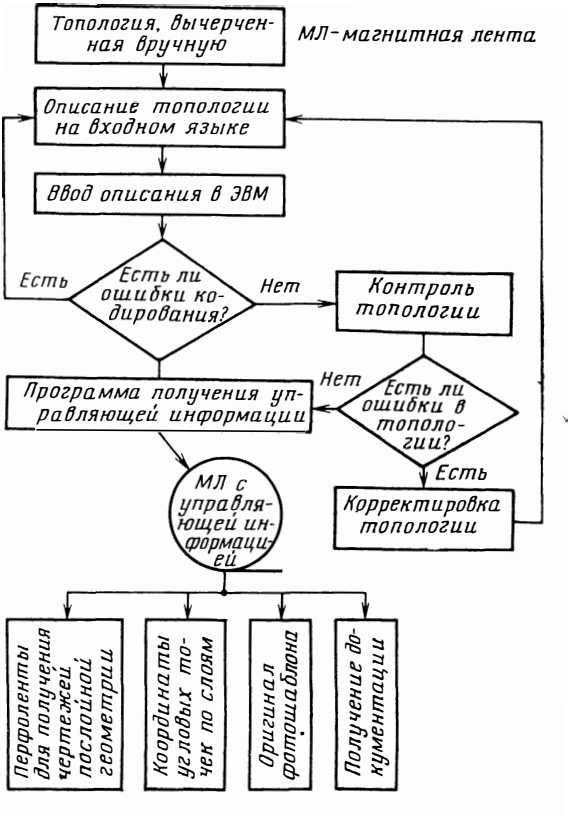


Рис. 4.1. Система автоматизированного проектирования фотошаблонов ИМС

4.2. ЭТАПЫ ПОЛУЧЕНИЯ ФОТОШАБЛОНОВ ИНТЕГРАЛЬНЫХ МИКРОСХЕМ

При изготовлении фотошаблонов ИМС используются координатографы или микрофотонаборные установки, управляющая информация для которых содержит около 10^{i+1} чисел, где i — степень интеграции ИМС. Поэтому подготовка этой информации про-

исходит с помощью ЭВМ. Исходными данными для ЭВМ служит подробное описание топологии ИМС, которое может осуществляться двумя способами: на основе задания координат всех угловых точек областей топологии и с помощью специально разрабатываемых языков.

Описание топологии ИМС координатами всех угловых точек обладает избыточностью, обусловленной следующими причинами:

топология ИМС состоит из наложенных друг на друга многоугольников, все стороны которых параллельны осям координат, в этом случае любые две соседние угловые точки имеют одинаковые координаты;

топология ИМС имеет определенную периодичность, т. е. некоторые ее структуры являются отражением или повторением аналогичных элементов;

для различных ИМС используются одни и те же ранее разработанные компоненты, которые могут быть закодированы определенными символами 1 раз.

Исключая перечисленные избыточные данные, можно, не указывая координат всех угловых точек, однозначно определить топологию ИМС. С уменьшением количества цифровой информации снижаются трудоемкость кодирования и вероятность появления ошибок.

В настоящее время разработаны языки для описания топологии ИМС по имеющемуся чертежу. Разработанный вручную совмещенный чертеж топологии ИМС описывается на специальном языке и с помощью перфокарт (перфолент) вводится в ЭВМ. Здесь информация о топологии схемы проходит синтаксический контроль, и при отсутствии ошибок вырабатывается информация, необходимая для вычерчивания фотошаблона. Кроме фотошаблона вычерчиваются послойная геометрия схемы и необходимая документация.

Рассмотрим структуру языков для описания топологии ИМС на примере языка CADEP, разработанного группой английских фирм. При разработке CADEP, с одной стороны, обращалось внимание на то, чтобы язык легко усваивался, а с другой — на то, что язык должен содержать возможности гибкого и краткого описания всех образцов топологии сложных масок ИМС. Поэтому CADEP был разработан как добавление к алгоритмическому языку ФОРТРАН (что позволило включить в него все возможности ФОРТРАНА) и ориентирован на ЭВМ, имеющие устройства ввода-вывода графической информации. В состав языка включена библиотека образцов фотошаблонов, которая может расширяться. Имеется возможность формировать новые фотошаблоны из библиотечных.

Язык CADEP содержит три группы переменных:

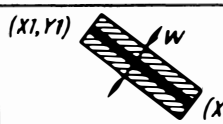
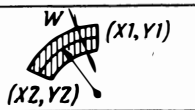
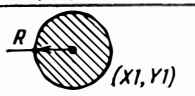
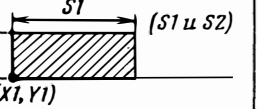
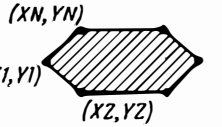
стандартные переменные ФОРТРАНа (логические, целые, действительные, комплексные);

геометрические переменные;

графические переменные.

Все геометрические переменные, определенные в программе, должны быть описаны оператором — GEOMETRIC *A*, *B*, ..., представляющим собой геометрический объект, лежащий в одной плоскости. Геометрические операторы, используемые в CADEP, пока-

Таблица 4.1

Функция	*Геометрический образ*	*Примечание*
LNC (X1, Y1, X2, Y2, W)		
LNP (X1, Y1, RHO, THETA, W)		*Конечные точки дуги располагаются против часовой стрелки по окружности, к которой принадлежит дуга*
ARC (X1, Y1, X2, Y2, R, W)		
CRCL (X1, Y1, R)		
SECT (X1, Y1, R, ALFA1, ALFA2)		
RECT (X1, Y1, X2, Y2)		
RECS		
ROLY (X1, Y1, X2, Y2, ..., XN, YN)		*Вершины должны быть переписаны против часовой стрелки, начиная от произвольной вершины*
FPLANE	*Вся геометрическая плоскость*	
MPLANE (XA, YA, ALFA)		
EMPTY	*Дополнительный*	FPLANE

заны в табл. 4.1. Над геометрическими переменными можно выполнять следующие операции: пересечение (символ /), объединение (символ +), разность (символ —). Последовательность выполнения этих операций такая же, как и в ФОРТРАНе, и ее можно менять с помощью скобок.

Для геометрических переменных существует оператор присвоения. Он дает возможность представить сложный образ топологии с помощью геометрических переменных. Например, образец *P* (рис. 4.2) может быть описан следующими присвоениями:

1. GEOMETRIC *A, B, C, D, E, P*
2. *A*=RECS (1, 1, 1, 1)
3. *B*=LNP (1.7, 1.7, 1.85, 52, 04)
4. *C*=RECT (2.5, 3, 3.5, 4)/MPLANE (3, 3, 218)
5. *D*=LNC (3.5, 3.85, 4.2, 3.85, 0.3)+ARC(4.65, 3.4, 4.2, 3.85, 0, 45, 03)+LNP(4.65, 3.4, 1.25, 270, 0.3).
6. *E*=RECS (4.3, 2.15, 0.7, —0.4)
7. *P*=*A*+*B*+*C*+*D*+*E*

Оператор 1 декларативный. Оператор 2 присваивает имя геометрическому образцу, определенному функцией RECS с аргументами 1,1,1,1 (см. табл. 4.1). Геометрическая переменная *B*, определенная оператором присвоения 3, представляет собой прямоугольник длиной 1,85, шириной 0,4, наклонный к оси *X* под углом 52°, одна из вершин которого имеет координаты (1,7; 1,7). Образец *C* — это пересечение прямоугольника, определенного функцией RECT, с полуплоскостью, определенной ориентированной прямой, проходящей через точку (3,3) и образующей с осью *X* угол, равный 218° (полуплоскость выбирают так, чтобы она была справа от ориентированной линии). Оператор 5 присваивает имя *D* образцу, получившемуся в результате объединения двух прямоугольников, один из которых дан в декартовых координатах (функция LNC), другой — в полярных координатах (LNP) и дугой, радиус кривизны которой 0,45 и ширина которой 0,3 — такая же, как у прямоугольников. Оператор 6 подобен оператору 2. Оператор 7 определяет образец P как объединение геометрических образцов *A, B, C, D, E*.

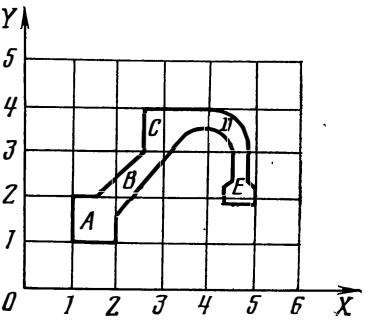


Рис. 4.2. Образец *P*

Для увеличения гибкости CADEP при описании изображения вводятся следующие функции: COPY(*G*, *S*), TRANS(*G*, *X1*, *Y1*, *S*), ROT(*G*, *X1*, *Y1*, ALFA, *S*), где *G* — это геометрическая переменная; *S* — масштабный коэффициент, при котором операция, определенная функцией, должна быть произведена.

Функция COPY дублирует структуру данных, представляющую переменную или выражение G; функция TRANS кроме операций, производимых COPY, переносит новые геометрические объекты, образованные из G, с помощью приращения их координат на X_1 и Y_1 в горизонтальном и вертикальном направлениях соответственно и изменяет масштаб объектов; функция ROT кроме операций, выполняемых COPY, поворачивает геометрический объект, образованный из G, на угол ALFA относительно точки (X_1, Y_1), которая может быть как внутри, так и вне G. Положительные углы соответствуют повороту против часовой стрелки.

Кроме геометрических CADEP содержит графические переменные. Они бывают двух видов: одноуровневые, представляющие собой физические двуразмерные объекты, расположенные на определенной графической плоскости, и многоуровневые переменные, состоящие из нескольких одноуровневых, лежащих в разных плоскостях и принадлежащих одному и тому же физическому объекту. Графические плоскости ограничены краями прямоугольника, противоположные вершины которого (0,0) и ($X_{\max}$ и $Y_{\max}$), где $X_{\max}$ и $Y_{\max}$ должны быть присвоены пользователем.

Графические переменные должны появляться в соответствующем декларативном операторе. Вид этого оператора для одноуровневых графических переменных LEVEL(i)A, B, C, ..., где $i=1$, 2, ... — идентификатор графической плоскости, на которой определены одноуровневые переменные. Многоуровневые переменные описываются оператором GRAPHIC E, F, G, ..., где E, F, G — многоуровневые переменные.

Одноуровневые переменные могут быть объединены в виде одноуровневого выражения с помощью оператора объединения (+). Одноуровневая переменная определяется оператором присваивания, левая часть которого — одноуровневые или многоуровневые переменные, а правая часть определяется геометрическим или одноуровневым выражением, причем в последнем случае уровень переменной в правой части оператора присваивания может отличаться от уровня в левой части. Этим пользуются при изменении уровня графического объекта.

Многоуровневые переменные можно определить посредством многоуровневого присваивания. Это выполняется оператором, у которого левая часть есть имя определяемой многоуровневой переменной, а правая — выражение, содержащее только одноуровневые переменные разных уровней. Например,

LEVEL (2) A

LEVEL (3) B

LEVEL (5) C

GRAPHIC M

$M=A*B*C,$

где * — операция умножения на ФОРТРАНе. Как над многоуровневыми, так и над одноуровневыми переменными могут выполняться функции COPY, TRANS и ROT.

Язык CADEP обладает некоторыми подпрограммами или макрокомандами для определения существования определенных условий среди графических переменных. Общий вид этих макрокоманд следующий:

SYM (аргумент/оператор),

где SYM — это базовый символ CADEP; аргумент — это переменные CADEP, а оператор — любой исполняемый оператор CADEP. Список макрокоманд этого класса приведен в табл. 4.2.

Таблица 4.2

SYM	Аргументы	Значение
BLG	(*X1*, *Y1*, *A*)	Принадлежит ли точка (*X1*, *Y1*) одноуровневому образцу *A*
INT	(*A*/*B*, *C*/(*D*+*E*) ,...)	Будут ли пересечения между образцами *A* и *B*, или *C* и (*D*+*E*), или ... непустыми *A* и *B* (могут быть одноуровневыми выражениями, имеющими один и тот же уровень, или многоуровневыми выражениями)
INC	(*A*/*B*, *C*/(*D*+*E*), ...)	Включен ли образец *A* в *B* или *C* в (*D*+*E*) или ...? (*A* и *B* — одноуровневые выражения, имеющие разные уровни, *A* включен в *B*, если его проекция на плоскости *B* находится внутри *B*)
DIS	(*A*/*B*, DIST)	Будет ли минимальное расстояние между *A* и *B* меньше или равно DIST (*A* и *B* — одноуровневые выражения, имеющие одинаковый уровень, расстояния измеряются по направлениям двух координат)

В качестве примера использования графических переменных рассмотрим задачу определения трехуровневого объекта *U*, составленного из образца, идентичного *P* (см. рис. 4.2) на плоскости первого уровня; образца, подобного *A* (см. рис. 4.2), имеющего центр (1.5, 1.5) и уменьшенного коэффициентом 0.7; компоненты на плоскости третьего уровня. Если *P* уже был описан как геометрическая переменная, мы можем записать

```
LEVEL (1) PP
LEVEL (2) AAP
LEVEL (3) CP
CRAPHIC U
```

PP=P

AAP=TRANS (A, 1.5, 1.5, 0.7)

CP=FPLANE—P

INC (AAR/PP) GO TO 50

STOP

U=PP*AAR*CP.

В CADEP введены некоторые командные операторы, которые приведены в табл. 4.3.

Таблица 4.3

Команда	Значение
DRAW *List*	Строит завершенный образ графических переменных, определенных в списке
a) STORE name, *List* b) STORE name, (*k*) c) STORE name	Хранит в массиве «имя» вспомогательного запоминающего устройства образ: a) одноуровневых и многоуровневых переменных, определенных в списке b) все графические переменные *k*-го уровня или c) все графические переменные
a) SHOW *List* b) SHOW (*k*) c) SHOW	Извлекает массив дисплея из образов: a) графических переменных в списке, b) всех графических переменных *k*-го уровня или c) всех одноуровневых графических переменных и показывает соответствующее изображение на экране дисплея
RECALL name	Переносит массив «имя» в главное запоминающее устройство
ERASE *List*	Стирает все данные геометрических или графических переменных, определенных в списке
DELETE *List*	Стирает последние данные геометрических или графических переменных, определенных в списке

Основными достоинствами языка являются следующие:

связь CADEP с ФОРТРАНом, что позволяет использовать его возможности;

он обладает всеми возможностями для гибкого и краткого описания сложных конфигураций, встречающихся в топологии ИМС;

введение многоуровневых переменных позволяет пользователю связывать элементы, лежащие в разных плоскостях, но представляющие один и тот же объект.

К недостаткам языка можно отнести необходимость иметь выполненный вручную совмещенный чертеж топологии, что является сложной, требующей больших временных затрат, работой.

Описание топологии ИМС вводится в ЭВМ, где получается информация для микрофотонаборного оборудования, вычерчивающего фотошаблоны ИМС. При описании чертежа топологии на специализированном языке необходимо создание сложных программ-трансляторов с этих языков во входные языки микрофотонаборных автоматов, различных устройств, вычерчивающих графическую информацию (типа координатографов).

Следует отметить, что, несмотря на необходимость разработки программ контроля и программ-трансляторов, ввод информации о топологии ИМС на специализированном языке представляется удобным, так как позволяет уменьшить объем вводимой информации, облегчает создание программ полного контроля топологии.

4.3. КОНТРОЛЬ ТОПОЛОГИИ

При автоматизированном проектировании фотошаблонов ИМС важным вопросом, определяющим работоспособность схемы, являются контроль входной информации и оперативное внесение изменений. Ошибки, наиболее часто встречающиеся при описании топологии, можно разбить на ряд групп:

графические, приводящие к искажению чертежа топологии;

ошибки, нарушающие конструкторско-технологические ограничения на топологию, такие, как невыполнение допустимого размера контуров в слое, нарушение допустимых расстояний между контурами в слое, нарушение допустимых расстояний между контурами в двух слоях, нарушение допустимых размеров на контактные окна;

ошибки, возникающие из-за неправильного восстановления принципиальной схемы по полученной топологии.

Для выявления графических ошибок наиболее перспективен оперативный контроль с помощью графического дисплея. При оперативном контроле локализуются ошибки, которые обнаруживаются при анализе отдельных контуров и линий. Указывать выявленные ошибки наиболее целесообразно путем печати в тексте описания топологии номеров ошибочных операторов и типов ошибок.

Контроль топологии на соответствие конструкторско-технологическим ограничениям сводится к проверке геометрических размеров.

Использование переборных методов для решения указанных вопросов практически невозможно из-за необходимости анализа огромных массивов информации, большого времени работы ЭВМ. Поэтому алгоритмы для решения такого рода задач строят на основе операций «упорядочения», «сортировки» и «факторизации», поиска в глубину и других, которые носят последовательный характер и заключаются в последовательной проверке заданных кри-

териев с отбрасыванием рассматриваемых участков и дальнейшей работой с уменьшающимися по размерам элементами топологического чертежа.

Для контроля топологии в интерактивном режиме используются графические дисплеи. Первое время применение дисплеев для синтеза топологии сдерживали малые размеры экранов. В настоящее время наметились тенденции применения мощного процессора для управления несколькими графическими дисплеями в режиме разделения времени и создания системы для одного конструктора на базе мини-ЭВМ.

В первом случае мощный процессор используется для реализации сложных программ, а графический дисплей применяется для вывода и коррекции результатов. Во втором случае минисистемы совместно с графическими дисплеями применяются для оперативного контроля и внесения изменений в топологию. Оперативное внесение изменений позволяет сократить в 2—3 раза цикл разработки топологии. Основные недостатки — высокая стоимость мини-системы и необходимость трудоемкой отладки и стыковки комплекса программного обеспечения этой системы.

Контроль разработанного варианта топологии включается практически во все известные системы проектирования ИМС. Полный контроль топологии ИМС включает:

контроль технологических ограничений;

проверку разработанного варианта топологии на соответствие принципиальной схеме;

анализ электрических характеристик ИМС по разработанной топологии.

Рассмотрим контроль выполнения технологических ограничений. При этом необходимо выполнять проверки на минимальную ширину элементов ИМС, на правильность установки в исходное положение, на правильность вложения одной фигуры в другую и другие геометрические соотношения.

Существующие программы контролируют минимальную ширину элементов топологии, рассчитывают расстояние от каждой вершины фигуры до всех ее сторон и сравнивают подсчитанное расстояние с заданным минимумом, обнаруживают все ситуации недопустимой ширины, если стороны фигуры представляют собой прямые линии, выполняют ряд различных проверок с двумя фигурами одновременно. Сюда относятся следующие тесты:

проверка установки фигур топологии ИМС в исходное положение. Программы считают расстояние от каждой вершины одной фигуры до всех сторон другой и затем изменяют взаимное расположение фигур;

проверка расстояний между двумя соседними фигурами и сравнение их с некоторым предписанным расстоянием;

проверка того, что каждая фигура на данном уровне содержит или должна содержать некоторую фигуру из другого уровня;

проверка нарушений установки фигуры в исходное положение.

Для сообщения конструктору об обнаруженных ошибках программы имеют две формы выходных документов. Первая форма представляет собой файл, состоящий из сведений обо всех фигурах, включаемых в нарушение правил проектирования. В этом же файле выдаются проверочный чертеж этих фигур и координаты места, где произошло нарушение. Вторая форма выходной информации — это распечатка на ЭВМ. Она включает:

перечень всех вопросов пользователя и ответы на них;

содержимое входного файла;

тип нарушения, координаты места, в котором произошло нарушение;

некоторые статистические данные, которые позволяют ускорить процесс работы над схемой топологии ИМС.

Следующим этапом контроля топологии ИМС является установление соответствия топологии принципиальной электрической схеме. Обычно при проектировании топологии ИМС стремятся сохранить информацию о нумерации элементов, что упрощает процесс контроля. Сохранение нумерации элементов на всех этапах оказывается затруднительным при взаимодействии конструктора и ЭВМ через дисплей и для проектирования топологии ИМС по частям. Сравнение принципиальной и топологической схем без учета нумерации элементов выполняется с помощью алгоритма отыскания подстановки изоморфизма графов исходной принципиальной электрической схемы и ее топологического чертежа.

Пусть заданы два неориентированных графа схем: $G=(X, U)$, $G'=(X', U')$, $|X|=|X'|$, $|U|=|U'|$.

Необходимо определить подстановку t, для которой справедливо $t(X)=X'$, $t(U)=U'$, тогда графы G, G' будут изоморфными.

Рассматриваемый алгоритм основан на последовательном разбиении вершин на П-изоморфные (предполагаемо изоморфные) группы. Разбиение прекращается при получении повершинного соответствия, которое проверяется в качестве подстановки изоморфизма.

Алгоритм организован следующим образом. Из X и X' выбираются два П-изоморфных подмножества: W и W'. Для каждой вершины $x\in X$ и $x'\in X'$ определяется число связей с W и W'. Две вершины П-изоморфных подмножеств могут быть изоморфными, когда они имеют одинаковое число связей с W и W'. В каждом подмножестве вершины группируются по числу связей с W и W'. Следовательно, получаются подразбиения подмножеств; П-изоморфными будут те подмножества подразбиений, вершины которых имеют равное число связей с W и W' и принадлежат П-изоморфным подразбиениям. Мощности П-изоморфных подмножеств должны быть равны. Если это не выполняется, то графы неизоморфны. Операция получения подразбиений и установления П-изоморфизма между подмножествами называется разложением. Обычно разложение приводит к подразбиениям некоторых подмножеств. Однако это выполняется не всегда. Для изоморфных графов алгоритм приводит к разбиениям X, X', в которых каждое подмноже-

ство содержит по одной вершине, а разложение неприменимо. Тогда П-изоморфизм между разбиениями определяет подстановку изоморфизма графов. Для учета рассмотрения разбиений вводятся метки запрета. Для разложения используются подмножества без меток. Для ускорения алгоритма при разложении рассматривают подмножества минимальной мощности и в отмеченном подразбиении метка переносится на подмножество максимальной мощности.

Приведем логическую схему алгоритма определения подстановки изоморфизма графов схем:

$$A_0 \overset{1}{\downarrow} A_1 p_1 \overset{5}{\uparrow} A_2 p_2 \overset{3\,2}{\uparrow\downarrow} A_3 \overset{3}{\downarrow} p_3 \overset{4}{\uparrow} A_4 p_4 \overset{2}{\uparrow} \omega \overset{1\,5}{\uparrow\downarrow} A_5 p_5 \overset{1\,4}{\uparrow\downarrow} A_k;$$

A_0, A_k — операторы начала и конца алгоритма соответственно; A_1 — разбиение на подмножества вершин X, X' графов G, G', определяющие $П$-изоморфизм подмножеств; A_2 — выбор из X подмножества W без метки и выбор из X' подмножества W' $П$-изоморфного W; A_3 — отметка W, выполнение разложения X, X' по W, W', перенос метки на подмножества максимальной мощности; A_4 — среди подмножеств из X определяются подмножество W с минимальной мощностью и $П$-изоморфные ему W'.

Выбираются $x \in W$, $x' \in W'$. Далее $W := W \setminus x$, $W' := W' \setminus x'$. Вершина x отмечается и выполняется разложение X, X' по x, x'; A_5 — выбор новых $x \in W$, $x' \in W'$, $W := W \setminus x$, $W' = W' \setminus x'$. Перебор внутри подмножеств, содержащих больше одного элемента; ω, p_1, p_2, p_3, p_4, p_5 — логические условия; p_1 проверяет равенство мощностей $П$-изоморфных подмножеств и смежности элементов внутри них; p_2 — проверка, отмечены ли все подмножества из X или нет; p_3 — проверка, имеет ли хоть одно подмножество из X больше чем одну вершину. Если да, то выполняется оператор A_4, если нет, то графы изоморфны и переход к A_k; p_4 — проверка возникновения подразбиений; p_5 — проверка, разное ли количество связей имеют W, W' с x, x', при $p_5 = 1$ графы неизоморфны и переход к A_k, при $p_5 = 0$ переход к A_1; ω — условие, тождественно равное нулю, т. е. после A_4 невозможно перейти к A_5, при $p_4 = 1$ после A_4 выполняется A_1, при $p_4 = 0$ после A_4 выполняется A_1, при $p_4 = 0$ после A_4 выполняется A_3.

Оценка сложности алгоритма лежит в пределах $0(n^2)$—$0(n!)$.

Работу алгоритма рассмотрим на двух примерах. Пусть заданы два графа G, G' (рис. 4.3): $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$, $X' = \{x'_1, x'_2, x'_3, x'_4, x'_5, x'_6, x'_7\}$.

Предположим, что вершины x_1 и x'_1 П-изоморфны и выполним операцию разложения $(x_1)^*(x_2, x_3, x_4, x_5, x_6, x_7)_1\,(\varnothing)$, $(x'_1)(x'_2, x'_3, x'_4, x'_5, x'_6, x'_7)_1(\varnothing)_0$.

Множества вершин X и X' распались на три подмножества: $X_1 = \{x_1\}$; $X_2 = \{x_2, \ldots, x_7\}$; $X_3 = \varnothing$; $X'_1 = \{x'_1\}$; $X'_2 = \{x'_2, \ldots, x'_7\}$; $X'_3 = \varnothing$. Подмножества X_2, X'_2 включают вершины, связанные с x_1, x'_1 одной связью, а подмножества X_3, X'_3 содержат вершины, не связанные с x_1, x'_1.

Далее согласно алгоритму для П-изоморфизма выбираются вершины из X_2 и X'_2. Пусть это будут вершины x_2 и x'_2. Выполняя операцию разложения, получаем

$$(x_1)^*(x_2)((x^1_3, x_7)_1(x_4, x_5, x_6)_0)_1{}^*,\ (x_1)'(x'_2)((x'^0_3, x'_7)_1(x'_4, x'_5, x'_6)_0)_1.$$

В результате разложения получили подмножества одинаковой мощности, но вершины x_3 и x_7 смежны, а x'_3 и x'_7 не смежны, следовательно, вершина x_2 не может быть П-изоморфна x'_2.

Проводя аналогичный поиск, получаем, что вершина x_2 не может быть изоморфна x'_3: $(x_1)(x_2)((x^1_3, x_7)_1(x_4, x_5, x_6)_0)_1$, $(x'_1)(x'_3)((x^\circ_2, x_4)_1(x_5, x_6, x_7)_0)_1$.

Продолжая аналогично, получаем, что граф G неизоморфен графу G'.

Приведенный пример показывает, что больше времени затрачивается при исследовании неизоморфных графов, так как в этом случае требуется полный перебор внутри классов разбиения.

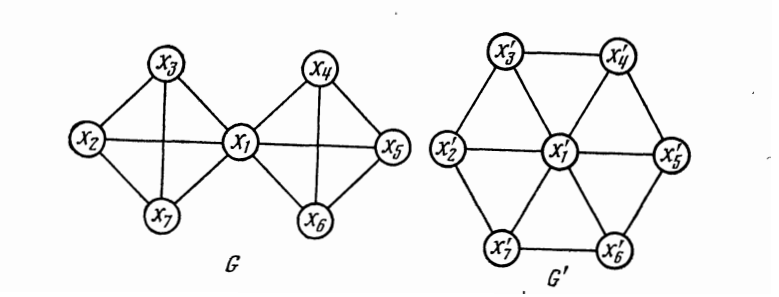


Рис. 4.3. Графы G и G^1

Пусть заданы два графа G, G' (рис. 4.4): $X_1=\{x_1, \ldots, x_{10}\}$; $X'=\{x'_1, \ldots, x'_{10}\}$. Предположим, что вершины x_1 и x'_1 П-изоморфны и выполним разложение по x_1, x'_1:

$$(x_1)^*[(x_2, x_3, x_9)_1(x_4, x_5, x_6, x_7, x_8, x_{10})_0{}^*], \quad (x'_1)[(x'_3, x'_9, x'_8)_1(x_2, x_4, x_5, x_6, x_7, x_{10})_0].$$

Метка перенесена на подмножество большой мощности. Мощности П-изоморфных подмножеств равны. Выбираем для разложения следующие подмножества:

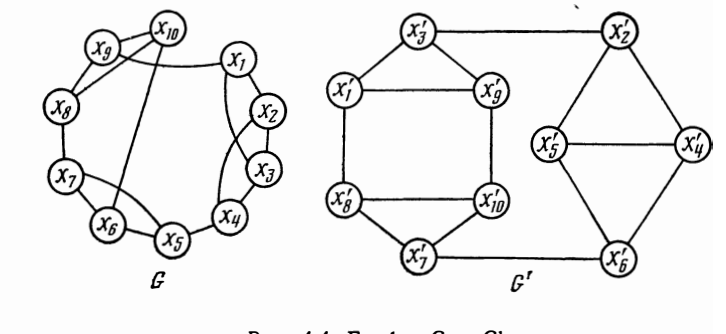


Рис. 4.4. Графы G и G^1

$$(x_1)[(x_2, x_3)^*, (x_9)][(x_4)_2(x^1_8, x_{10})_1(x_{15}, x_6, x_7)_0{}^*],$$
$$(x_1)[(x'_3, x'_9)(x'_8)][(x'_{10})_2(x'^0_2, x'_7)_1(x'_4, x'_5, x'_6)_0].$$

Вершины x_8 и x_{10} смежны, а соответствующие им x'_2 и x'_7 не смежны. Следовательно, вершины x_1 и x'_1 не могут быть $П$-изоморфными.

Выбираем из X' вершину x'_2 и считаем ее $П$-изоморфной x_1:

$$(x_1)^*[(x_2, x_3, x_9)_1(x_4, x_5, x_6, x_7, x_8, x_{10})_0{}^*],$$
$$(x_2)'[(x'_3, x'_4, x'_5)_1(x'_1, x'_6, x'_7, x'_8, x'_9, x'_{10})_0].$$

Процесс продолжаем аналогично до получения разбиений

$$(x_1)*(x_2,\ x_3)*(x_9)*(x_4)*(x_5)*(x_6,\ x_7)*(x_8,\ x_{10})(x'_2)(x'_5,\ x'_4)(x'_3)(x'_6)(x'_7)\times$$
$$\times(x'_{10},\ x'_8)(x'_1,\ x'_9).$$

Производя перебор для подмножеств, содержащих больше одного элемента, получим подстановку изоморфизма графов G и G'

$$t=\begin{pmatrix} x_1, & x_2, & x_3, & x_9, & x_4, & x_5, & x_6, & x_7, & x_8, & x_{10} \\ x'_2, & x'_5, & x'_4, & x'_3, & x'_6, & x'_7, & x'_{10}, & x'_8, & x'_1, & x'_9 \end{pmatrix}.$$

Отметим, что полученная подстановка не единственная. Существует некоторое множество подстановок, переводящих граф G в G'.

Интерпретируя номера вершин графов с учетом их соответствия элементами схем, получаем соответствие между элементами изоморфных схем.

В настоящее время широкое распространение получили интерактивные системы проектирования фотошаблонов ИМС, в которых взаимодействие разработчика и ЭВМ осуществляется с помощью устройств отображения графической и алфавитно-цифровой информации. Как правило, в подобных системах предусматриваются команды вызова из библиотеки и размещения на экране типового элемента, формирования четырехугольников и других фигур, передвижения, поворота и удаления области или группы областей, изменения масштаба изображения, занесения фрагмента топологии в библиотеку или вывода его на графопостроитель. Для ввода команд в ЭВМ применяют световое перо, функциональную кнопочную модель, а также клавиатуру, с помощью которой передается алфавитно-цифровая информация.

С увеличением плотности компонентов на кристалле ИМС резко возросло количество графической информации, выводимой на экран дисплея. Поэтому наиболее перспективными являются системы обработки графической информации в интерактивном режиме на базе мини-ЭВМ. Эти системы входят в состав САПФ ИМС. Краткая характеристика интерактивных мини-систем приведена в табл. 4.4. Такие системы включают: центральный процессор; мини-ЭВМ, дисковый накопитель, кассетный лентопротяжный механизм, большую взаимодействующую поверхность, графический дисплей, преобразователь, построитель трафаретов.

В системах большое внимание уделяется человеческим факторам. Например, конструктор не должен осуществлять преобразование чертежей в программные операторы, он просто вычерчивает или преобразует в цифровую форму соответствующее изображение, наблюдая на экране дисплея результаты.

В системах предлагаются взаимно соотносящиеся средства для диалогового взаимодействия с ЭВМ. Одним из них является поверхность чертежей доски, именуемая «большая взаимодействующая поверхность» (БВП), которая предназначена для выполнения и преобразования в цифровой вид грубых эскизных набросков, на которые ЭВМ реагирует, вычерчивая в ответ результаты на чертежной бумаге в увеличенном масштабе. Кроме того, на БВП графические изображения переводятся в цифровую форму. Графопо-

строитель БВП позволяет вычерчивать достаточно мелкие детали. Вторым средством является графический дисплей, на котором конструктор может увидеть и отредактировать любую часть проектируемой конструкции.

Таблица 4.4

Страна	ЭВМ	Внешняя память	Параметры терминального устройства	
			Съемщик координат	Координатограф
Япония	Мини-ЭВМ	НМЛ	Поле 1,2×1,5 м² Скорость 24 м/мин Точность 0,075 мм	
США	PDP-11/10	НМЛ НМД	Поле 1,2×1,5 м²	Поле 0,9×1,0 м²
Англия	Мини-ЭВМ	НМЛ НМД	Поле 0,9×1,2 м² Скорость 15—25 м/мин Точность 0,127 мм	
Франция	Мини-ЭВМ	НМЛ	Поле 1,5×2,0 м² Точность 0,1 мм	Поле 0,8×1,5 м² Точность 0,05 мм²

Примечание. НМЛ — накопитель на магнитной ленте; НМД — накопитель на магнитных дисках.

Вся работа по любому из перечисленных видов проектирования ИМС может выполняться по частям на дисплее. Затем результаты проектирования могут быть вычерчены на БВП для окончательного контроля качества. Помимо этого, вся работа может выполняться на БВП так, что дисплей может быть использован другим конструктором. Необходимые изменения вносятся с помощью преобразователя и дисплея либо с помощью доски БВП.

Сердцем системы является управляющая мини-ЭВМ, используемая для хранения графических данных и их преобразования. Завершенные чертежи записываются для хранения на магнитную ленту и переписываются с ленты на диск, если они должны подвергнуться редактированию. Оператор может отредактировать любую часть топологии микросхемы на экране дисплея. Преобразователь данных в цифровой вид является устройством, которое представляет эффективные быстродействующие и естественные средства управления электронным лучом в трубке дисплея, позволяя в то же время конструктору вести письменную регистрацию этапов конструирования или следовать эскизному наброску.

Из часто встречающихся фрагментов топологии или чертежей составляется библиотека системы. Эти фрагменты идентифицируются и записываются на магнитную ленту. Конструктору упрощена задача вычерчивания элементарных графических объектов.

С помощью преобразователя луч на экране дисплея переводится в необходимую начальную точку. Затем, используя заданные координаты точек поворота, автоматически получают любую фигуру, представляющую собой набор прямоугольников. Прямые линии строятся под любым углом. После того как луч на экране дисплея переведен в начальную точку, можно, задав один радиус, прорисовать окружность. Текст и заголовки подготавливаются посредством указания начала места, отведенного для текста, с помощью курсора, после чего нужный текст печатается на клавиатуре электрической пишущей машинки или вводится с заранее заготовленной перфоленты.

В результате работы систем изготовляют чертежи в увеличенном масштабе для окончательного контроля качества перфоленты или магнитные ленты для сборочного или сверлильного станка, получают набор необходимой документации и точный фотошаблон на стеклянной фотопластине или фотопленке.

4.4. КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Что является исходными данными для проектирования топологии ИМС?
2. Опишите три основных метода проектирования топологии ИМС.
3. Опишите способ проектирования ИМС на приборном уровне.
4. Опишите способ проектирования ИМС из типовых элементов.
5. Опишите способ проектирования ИМС на основе базовых кристаллов.
6. Какие задачи решает САПФ ИМС?
7. Приведите структурную схему САПФ ИМС.
8. Для каких целей используют языки описания топологии ИМС?
9. Опишите структуру языка описания топологии ИМС.
10. Постройте пример описания фрагмента топологии на известном вам языке.
11. Каким образом описываются одноуровневые переменные?
12. Приведите пример использования графических переменных.
13. Опишите оператора языка.
14. Приведите достоинства и недостатки языков описания топологии ИМС.
15. Каким вы представляете себе язык описания топологии ИМС5 и последующих степеней интеграции?
16. В чем заключается контроль топологии ИМС?
17. Приведите основные ошибки, встречающиеся при описании топологии ИМС.
18. В чем заключается контроль топологии ИМС на соответствие конструкторско-технологическим ограничениям?
19. Какие этапы включает полный контроль топологии ИМС?
20. Опишите процесс контроля выполнения технологических правил проектирования топологии ИМС.
21. Каким образом устанавливается соответствие между топологией ИМС и ее принципиальной электрической схемой?
22. Постройте алгоритм восстановления принципиальной схемы по известной топологии.
23. Постройте структурную схему алгоритма определения изоморфизма ориентированных графов.
24. Постройте логическую схему алгоритма и граф-схему алгоритма для определения изоморфизма графов и гиперграфов.
25. Определите подстановку изоморфизма для графов, заданных матрицами смежности:

$$R_1 = \begin{array}{c|cccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 7 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 8 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 10 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array};$$

$$R_2 = \begin{array}{c|cccccccccc} & 1' & 2' & 3' & 4' & 5' & 6' & 7' & 8' & 9' & 10' \\ 1' & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 2' & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 3' & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 4' & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 5' & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 6' & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 7' & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 8' & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 9' & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 10' & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}.$$

5. ВОПРОСЫ ВЫПУСКА КОНСТРУКТОРСКОЙ И ТЕХНОЛОГИЧЕСКОЙ ДОКУМЕНТАЦИИ

5.1. ОСНОВНЫЕ ТРЕБОВАНИЯ ПРИ ВЫПУСКЕ КОНСТРУКТОРСКОЙ И ТЕХНОЛОГИЧЕСКОЙ ДОКУМЕНТАЦИИ

Максимальный эффект от автоматизации достигается только при комплексном подходе к решению задачи конструирования. В частности, логическим и обязательным завершением работ по проектированию ЭА является этап получения комплекта конструкторской и технологической документации (КТД), обеспечивающий автоматизацию этапов изготовления и контроля. Решение этой задачи связано с преодолением целого ряда трудностей теоретического и прикладного характера. Остановимся на некоторых из них.

Действующий в настоящее время комплекс ГОСТ ЕСКД является обобщением опыта ручных методов проектирования, что не позволяет использовать его полностью в САПР. Поэтому необходимо совершенствовать ЕСКД с учетом специфики САПР.

Ограниченные возможности существующих средств отображения графической и текстовой информации не позволяют получать в готовом виде всю необходимую КТД. Кроме того, разработка программных средств для хранения и воспроизводства графической информации связана с рядом трудностей. Затруднения связаны с тем, что САПР никогда не внедряется сразу целиком, а по частям. Это приводит к тому, что в обращении находятся документы, выполненные как ЭВМ, так и конструктором.

Внедрение САПР и выпуск на ЭВМ КТД сопряжены с изменением функций и взаимоотношений отдельных подразделений. Учитывая вышесказанное, можно сформулировать основные требования к САПР КТД:

1. При выборе форм и форматов выходных документов необходимо максимально учитывать стандарты, принятые в ЕСКД.
2. Система должна по возможности выдавать весь набор КТД, который получается при неавтоматизированном проектировании, плюс КТД для этапов контроля полученного изделия.
3. Машинные формы КТД должны обеспечивать не только автоматизированные, но и неавтоматизированные методы обращения документации и изготовления ЭА.
4. Основные графические документы целесообразно выполнять базовым способом.

На основании этого можно привести примерный перечень программ, входящих в САПР КТД. Эти программы обеспечивают получение: ведомости покупных изделий; спецификации; перфоленты результатов размещения и трассировки для автоматического получения сборочного чертежа на графическом устройстве; перфоленты для автоматического изготовления фотошаблона на генераторе изображений; перфоленты для автоматической сверловки и контроля печатных плат.

5.2. СРЕДСТВА МАШИННОЙ ГРАФИКИ

В настоящее время существует целый ряд устройств машинной графики: графопостроители, координатографы, микрофотонаборные установки, различные кодировщики и т. д.

Координатографы. В СССР выпущена целая серия подобных устройств от «Минск-2000» до «Минск-2006». Первые варианты этих устройств использовались только в режиме получения фотошаблонов печатных плат. Последующие модификации имеют возможность работать в трех режимах:

кодировщик графической информации. Этот режим позволяет кодировать чертеж и получать перфоленту для его прорисовки;

прорисовка закодированных чертежей. Использование координатографа в этом режиме позволяет по единожды полученной перфоленте воспроизводить закодированный чертеж необходимое количество раз;

изготовление фотошаблонов печатных плат.

Все модификации координатографов типа «Минск» управляются перфолентой, максимальный размер стола 500×500 мм. Модификации «Минск-2005», «Минск-2006» снабжены буферной памятью. Основным недостатком координатографов этого типа является низкое быстродействие. Например, один слой двухслойной печатной платы, содержащей до 40 модулей, в среднем вычерчивается на фотопластине около 3 ч. Координатографы типов ЭМ-703, «Картимат-3Е» и КПА-1200 подобны описанному. Но предназначены только для изготовления фотошаблонов и могут работать в двух режимах:

изготовление фотошаблонов методом гравирования (с применением гравировальной головки) на пластине, покрытой специальным слоем;

изготовление фотошаблонов лучом на фотопластине.

Почти все координатографы имеют возможность выводить алфавитно-цифровую информацию.

Рассмотрим принцип действия координатографов этого типа на примере координатографа «Минск-2000». Координатограф «Минск-2000» (рис. 5.1) предназначен для изготовления фотошаблонов пе-

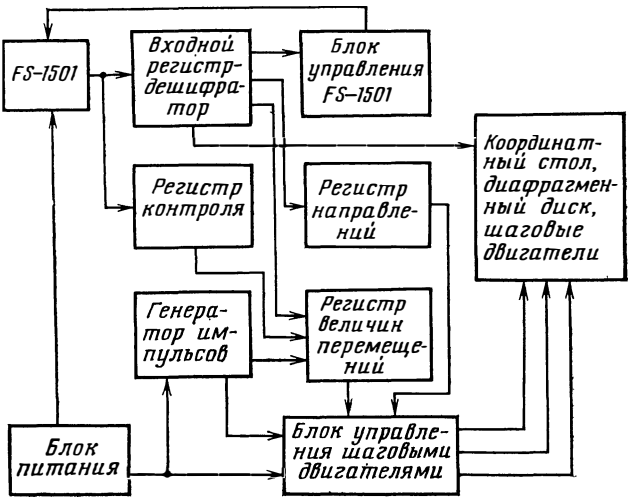


Рис. 5.1. Структурная схема координатографа «Минск-2000»

чатных плат экспонированием по программе, подготовленной на ЭВМ. Он формирует изображение путем нанесения прямолинейных отрезков печатных проводников шириной 0,2 ... 4,0 мм в восьми ортогонально-диагональных направлениях и контактных площадок различной формы. Максимальный размер контактной площадки ограничен кругом диаметром 6 мм. Ширина проводников и форм контактных площадок определяются размерами и формой смен-

ных диафрагм, которые располагаются в 16 гнездах диафрагменного диска. Рабочее поле координатографа имеет размеры 500×500 мм, точность выполнения фотошаблонов 0,05 мм, скорость — не более 10 мм/с. Управление работой координатографа осуществляется с перфоленты, которая вводится с помощью устройства FS-1501.

Наиболее совершенным координатографом этой серии является координатограф «Минск-2005». Это устройство может выполнять все перечисленные операции и, кроме того, реперфорацию, сравнение перфолент, реперфорацию перфолент с одновременной прорисовкой. В «Минск-2005» есть режим, позволяющий выполнить поворот любого символа или всего изображения на угол, кратный 90°. При тех же размерах поля координатограф «Минск-2005» имеет максимальную скорость (50 мм/с) вычерчивания фотошаблонов. Важным достоинством координатографа «Минск-2005» является то, что изготовление фотошаблона производится в незатемненных помещениях.

Основные сведения о кодировщиках. Кодировщики (ЭМ-709, КС-2) служат для кодирования графической информации, т. е. для перевода графической информации в цифровую. Носителем выходной информации является перфолента. Размер стола подобных кодировщиков 1,2×1,2 м. Принцип работы их следующий. Кодируемый чертеж крепится на столе кодировщика, который снабжен движущимся по направлению координатных осей перекрестием и цифровым табло. Оператор совмещает перекрестье с кодируемой точкой, координаты которой высвечиваются на цифровом табло. Нажатием кнопки оператор включает перфоратор, и значения координат выводятся на перфоленту вместе со служебными символами.

Принцип работы кодировщиков рассмотрим на примере координатомера-кодировщика ЭМ-709. Это устройство оптико-механического типа с фотоэлектрическими датчиками отсчета перемещения рабочего органа. Устройство предназначено для полуавтоматического кодирования графической информации с выдачей результатов кодирования на перфоленту и (или) непосредственно в ЭВМ. Рабочее поле имеет размер 900×200 мм, точность снятия координат 0,2 мм, шаг работы устройства может иметь значения 0,1; 1; 2; 2,5 и 5 мм. Устройство обеспечивает поворот и замену координатных осей, вывод на перфоленту алфавитно-цифровых кодов, специальных знаков. Структурная схема устройства приведена на рис. 5.2. Кодирование символов осуществляется путем нажатия на наборном поле площадки с необходимым символом. При этом в кадре информации после значений координат *X, У* будет сформирован номер выбранного символа.

Рассмотрим микрофотонаборные установки (МФНУ) ЭМ-519Б и 549. Они служат только для изготовления фотошаблонов ИМС, увеличенных в 10 раз. Поэтому фотошаблоны, полученные на МФНУ, называют промежуточными. Точность работы МФНУ 1 мкм. Максимальный размер кристалла 6×6 мм. Управляется

МФНУ перфолентой, засветка необходимых мест на фотопластине происходит путем соответствующего перемещения стола установки и управления размерами прямоугольной диафрагменной щели, пропускающей свет.

Мощным средством в руках разработчиков КТД в САПР стали автоматизированные рабочие места (АРМ). В настоящее время отечественная промышленность выпускает АРМ-М и АРМ-Р специально для целей автоматизации проектирования.

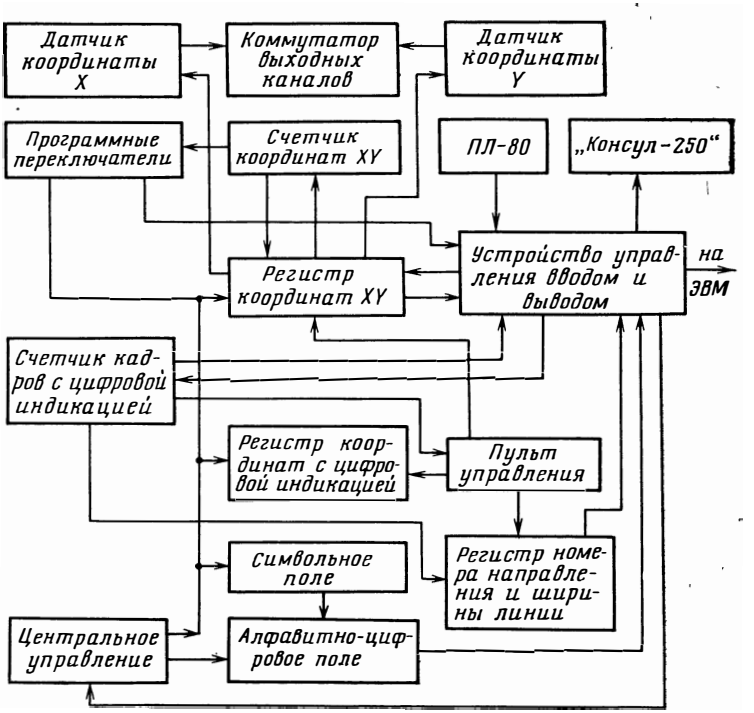


Рис. 5.2. Структурная схема ЭМ-709

Автоматизированное рабочее место является диалоговым графическим комплексом проектирования, предназначенным для автоматизации операций по подготовке, преобразованию и редактированию текстовой и графической информации, решения конструкторских задач, а также операций по взаимодействию конструктора ЭА с САПР. Для проектирования машиностроительных объектов в основном используется АРМ-М. Он содержит графический дисплей с расширенными возможностями чертежного автомата и полуавтомата кодирования. Специализация АРМ-М определяется прикладными программами, введенными в операционную систему, и содержанием информационной базы. В автономном режиме АРМ-М позволяет решать задачи, связанные с вводом и выводом

графической информации, выполнением расчетных работ, кодированием чертежей, выпуском технической документации, подготовкой программ для станков с числовым программным управлением (ЧПУ). В режиме взаимодействия с САПР АРМ-М позволяет решать задачи проектирования современных сложных объектов. Структурная схема АРМ-М показана на рис. 5.3. Автоматизиро-

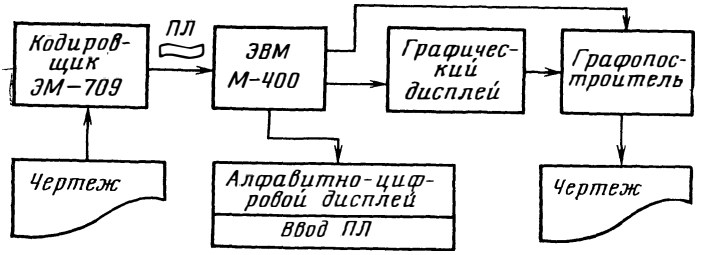


Рис. 5.3. Структурная схема АРМ-М

ванное рабочее место может использоваться для проектирования чертежей печатных плат в интерактивном режиме. Исходный вариант проектируемого чертежа кодируется на кодировщике ЭМ-709, и полученная перфолента (ПЛ) вводится в ЭВМ. В ЭВМ введенная информация обрабатывается программами, введенными разработчиком с дисплея или с перфоленты. Результат проектирования либо выводится на экран дисплея и дорабатывается в интерактивном режиме, а затем прорисовывается на графопостроителе, либо непосредственно выводится для прорисовки на графопостроитель.

Автоматизированное рабочее место используется для:

решения задач проектирования ЭА, связанных с приемом, хранением и переработкой информации;

вывода информации на перфоленту, печать, графопостроитель, графический или текстовой дисплей;

ввода информации в память АРМ с центрального процессора, перфокарт, перфолент, магнитных лент, устройств кодирования графической информации;

ведения диалогового режима проектирования.

Автоматизированное рабочее место имеет два режима работы — автономный и режим взаимодействия с САПР, построенной на базе ЕС ЭВМ. В автономном режиме решаются задачи:

ввода-вывода графической информации;

размещения элементов произвольной конфигурации на поле печатной платы;

разводки и редактирования топологии печатных плат;

выполнения электрических принципиальных схем;

механизированного изготовления управляющих перфолент для получения фотооригиналов слоев печатных плат;

составления спецификаций;

выполнения аналитических и графических инженерных расчетов;

выпуска документации;

корректировки и редактирования графических, текстовых и смешанных документов;

выполнения сборочных чертежей печатных плат, сборок, шкафов;

ввода стандартных конструктивных элементов в библиотеку АРМ.

В режиме взаимодействия с САПР АРМ-Р позволяет:

решать задачи моделирования схем ЭА;

синтезировать схемы ЭА;

выполнять сквозной синтез схемы ЭА с выходом на конструкторско-технологическую документацию;

обращаться в архив САПР;

передавать на хранение в архив информацию.

Комплекс АРМ-Р состоит в основном из следующих технических средств: управляющего вычислительного комплекса (УВК); устройства расширения памяти (УРП); устройства расширения комплекса (УРК); устройства внешней памяти на магнитных дисках (УВП НМД); экранного графического пульта (ЭГП); экранного алфавитно-цифрового пульта (ЭАЦП); мозаичного устройства печати (МУП); устройства ввода информации с перфокарт (УВИП); накопителя на магнитной ленте (НМЛ); графического построителя (ПГ); устройства ввода графической информации (УВГИ).

Все устройства АРМ-Р объединяются в систему с помощью единого комплекта связи, называемого общей шиной.

Основным устройством АРМ-Р является процессор на базе УВК. Процессор служит для выполнения арифметических, логических и операций обмена. Для приема, хранения и выдачи двоичной информации и для расширения оперативной памяти предназначено УРП. Для увеличения функциональных возможностей УВК путем подсоединения дополнительных внешних технических средств предназначено УРК, для работы в качестве внешнего запоминающего устройства УВП НМД. В составе АРМ для ввода, отображения и редактирования на экране графической и буквенно-цифровой информации используется ЭПГ. Для ввода с клавиатуры и отображения на экране алфавитно-цифровой информации предназначен ЭАЦП. Для вывода алфавитно-цифровой информации на печать из УВК применяется МУП. Для восприятия информации, нанесенной на перфорированные карты, преобразования ее в электрические сигналы используется УВИП. Для записи, воспроизведения и длительного хранения информации предназначен НМЛ. Для вычерчивания на обычных бумажных носителях в прямоугольной системе координат схем и других документов по данным в виде кодов или координат точек графика и приказов о положении пишущих устройств применяется ПГ. Для получения кодового опи-

сания чертежа обходом опорных точек с последующим выводом графической информации на перфоленту предназначено УВГИ.

Математическое обеспечение (МО) АРМ-Р призвано обеспечить работу устройств основного комплекса связи с оператором и пользователем, осуществления режима одновременного выполнения прикладных задач и проверки работоспособности АРМ-Р. Оно состоит из базового программного обеспечения (БПО) и пакета прикладных программ (ППП). В свою очередь БПО состоит из следующих программных модулей: дисковой операционной системы (ДОС) АРМ; тест-мониторной системы АРМ; базовой оперативной графической системы; вспомогательных программ обмена и перекодировки; системы МО на перфоленте.

Основным элементом БПО АРМ является ДОС, которая управляет всеми программными модулями.

В качестве ППП используется система графического редактирования и формирования базы данных, которая позволяет использовать АРМ-Р как интеллектуальный диалоговый терминал для реализации различных этапов проектно-конструкторских работ.

Графопостроители или чертежные автоматы (ЧА) позволяют автоматизировать наиболее трудоемкую операцию процесса проектирования — изготовление технической документации. Эти устройства дают возможность получать информацию в привычной форме — в виде графиков, спецификаций, чертежей, таблиц. Графопостроители отличают высокое качество и сравнительно высокая (до 1000 мм/с) скорость вычерчивания графической информации. Чертежные автоматы делятся на три класса по способу программного управления: непосредственно от ЭВМ; от машинных носителей информации; от первых двух вместе.

Графопостроителям с программным управлением присущи следующие недостатки: низкое по сравнению с ЭВМ быстродействие, недостаточная надежность работы, программно-командная несовместимость различных ЧА.

По способу обработки входных сигналов различают аналоговые, цифроаналоговые и цифровые ЧА. Графопостроители аналогового типа построены по принципу компенсационных следящих систем с обратной связью. Высокая точность и полная свобода в направлении перемещения узла записи — вот основные достоинства этих ЧА. Однако из-за сложности согласования их редко используют с цифровыми ЭВМ. Для преобразования электрического сигнала в механическое перемещение ЧА с цифровым управлением используют шаговый двигатель (ШД). Особенностью работы таких ЧА является то, что количество направлений, в которых может перемещаться узел связи, ограничено. Основными недостатками как аналоговых, так и цифровых ЧА являются громоздкость оборудования и динамическая неустойчивость.

В зависимости от формы стола, на котором крепится бумага для вычерчивания, чертежные автоматы делятся на рулонного и планшетного типов.

Работу чертежного автомата рассмотрим на примере ЧА «Вектор 1301». Он относится к классу ЧА с электромеханической системой развертки с управлением от ЭВМ и предназначен для изготовления различной графической информации. Чертеж вычерчивается на обычной бумаге головкой, снабженной двумя перьями. Переключение перьев производится специальной командой, а бумага крепится на планшете с помощью электростатического поля. «Вектор-1301» может управляться или от ЭВМ, или с пульта. В ЧА предусмотрены устройства для удобного подключения к ЭВМ. Основные технические характеристики «Вектор-1301» следующие: размеры рабочего поля 600×850 мм, элементарный шаг перемещения 0,05 мм; скорость вычерчивания линий 25...75 мм/с; точность 0,2 мм; скорость вычерчивания символов 5 симв/с; число символов 77. Схема управления ЧА «Вектор-1301» — цифрового типа и построена по принципу отработки шаговых приращений. Структурная схема показана на рис. 5.4. Управление графопостроителем происходит поступающими от ЭВМ 36-разрядным кодом. Он содержит координаты *X*, *У* перемещения головки записи и команду подъема или опускания пера.

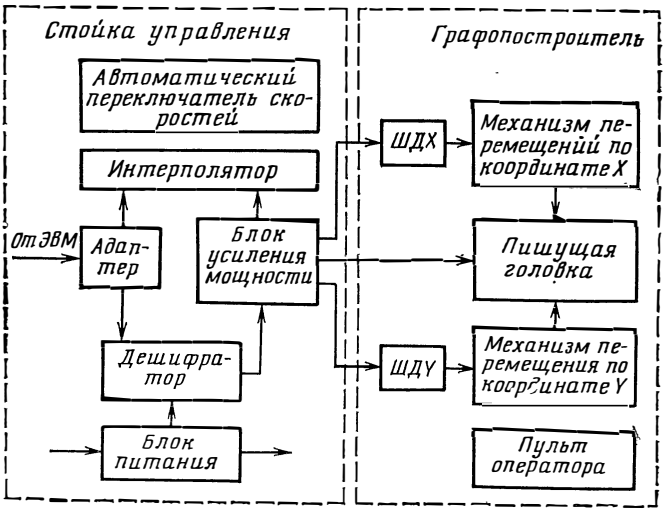


Рис. 5.4. Структурная схема чертежного автомата «Вектор-1301»

Для ЧА «Вектор-1301» построено базовое математическое обеспечение (МО), т. е. набор программ, реализованных на одной из версий алгоритмического языка ФОРТРАН и не ориентированный на конкретную ЭВМ. Комплекс программ разбит на три уровня. Первый уровень — это программы, управляющие построителем независимо от особенностей модели. Программы второго уровня

осуществляют переход от заданной системы координат к любой другой, а программы третьего уровня обеспечивают работу ЧА в декартовых координатах, задаваемых оператором.

5.3. ПРОЕКТИРОВАНИЕ И ВЫПУСК ТЕКСТОВОЙ И ГРАФИЧЕСКОЙ КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКОЙ ДОКУМЕНТАЦИИ

В САПР текстовая документация должна выпускаться согласно ГОСТ «Ведомость покупных изделий» и «Спецификация». При этом выполняются следующие необходимые требования и принятые допущения:

печатаются наименования разделов;

в обозначениях символов «±» печатается «+—»;

все обозначения печатаются прописными буквами, а «мм^2», «см^2» как «кв. мм», «кв. см» и т. д.;

в спецификации указываются позиционные обозначения элементов согласно электрической принципиальной схеме.

Все текстовые документы, кроме титульного листа, должны выпускаться автоматизированным способом в виде распечаток на АЦПУ, причем размеры всех граф, колонок в документах должны соответствовать ГОСТ. Титульный лист имеет некоторую переменную часть (фамилии, подписи, год и т. д.). Поэтому автоматизированным способом выполняется только его постоянная часть, а переменная часть заполняется вручную. При этом часто используют метод базового чертежа, т. е. 1 раз (обычно вручную) выпускают базовый чертеж на однотипную группу конструкторских блоков, который содержит всю постоянную часть. Затем автоматизированными методами добавляют все то, что относится к конкретному устройству.

При изготовлении сборочного чертежа руководствуются ГОСТ и следующими допущениями:

все модули изображают условно. Размеры условных изображений должны соответствовать реальным с учетом масштаба чертежа;

на изображениях элементов указываются их позиционные обозначения в соответствии со схемой;

цифры и буквы выполняются высотой 3 мм.

Результатом работы программ, как правило, является перфолента, управляющая графопостроителем или координатографом, на котором выполняется необходимый чертеж.

Программное обеспечение для выпуска графической документации включает обычно два комплекса программ:

программы подготовки управляющих перфолент для чертежных автоматов;

программы обеспечения графического взаимодействия конструктора с ЭВМ.

Первый комплекс программ выполняет:

преобразование топологической модели печатной платы в графические модели чертежа печатной платы и сборочного чертежа;

формирование условных изображений контактных площадок, переходных отверстий, радиоэлементов;

заполнение таблицы отверстий платы;

формирование постоянной части чертежа;

трансляцию графических процедур;

представление строк символов в виде отрезков линий;

минимизацию холостых пробегов чертежного автомата;

преобразование информации о чертеже в язык чертежного автомата и получение управляющей перфоленты.

Второй комплекс программ обеспечивает графическое взаимодействие конструктора и ЭВМ, формируя команды языка графического диалога в виде дисплейных слов, объединенных в дисплейный файл. Язык графического диалога включает обычно дисплейные слова построения и стирания линий, выделения фрагмента изображения, сдвиг изображения, т. е. в основном используются точечный и векторный режимы работы дисплея.

Обычно конкретная реализация подобного программного обеспечения САПР КТД позволяет выпускать документацию и при ручном изготовлении чертежей. При этом предусматривается возможность ввода информации о чертеже с помощью кодировщика (например ЭМ-709).

Автоматизация этапов изготовления и контроля обеспечивается комплексом технических средств и программ, предназначенных для преобразования информации, сформированной в рабочем архиве системы автоматизированного проектирования и выдачи на носители программ управления отдельными составляющими технологического комплекса.

Графическая информация о плате (или ИМС) получается либо в результате автоматизированного проектирования, либо после корректировки с применением диалоговой подсистемы, либо при трансляции результатов кодирования на кодировщике. Полученная, таким образом, информация о печатной плате или интегральной микросхеме проходит доработку на дисплее (т.е. производятся проверка выполнения технологических ограничений, устранение узких мест, проверка на обрыв, короткое замыкание и т. д.). Полученная информация является исходной для работы технологического автомата, который изготавливает фотошаблоны.

Кроме изготовления фотошаблонов методы автоматизированного проектирования широко используются на этапе сверловки плат. Это вызвано тем, что требования к точности расположения переходных отверстий на печатной плате высоки. В настоящее время, разрабатывая подобные программы, их ориентируют на сверлильные автоматы ОФ-72Б или «Шмоль». При этом обычно решают задачи формирования массивов выходной информации по диаметрам отверстий и получения машинных носителей в кодах управления сверлильным станком.

К другим автоматизированным процессам относится процесс контроля печатных плат, который включает ряд этапов. Первым из них является этап проверки соединений на плате до и после

монтажа, так как во время монтажа возможно появление лишних перемычек из-за припоя. Процесс контроля монтажа заключается в том, что питающее напряжение подается путем коммутаций последовательно через электрические цепи проверяемых соединений. При этом каждая точка монтажа коммутируется с каждой из остальных. Число коммутаций, которое необходимо провести для N точек монтажа, равно $(N—1)\ N/2$, причем если между проводниками имеется короткое замыкание, то могут возникнуть обводные цепи, что не позволит определить обрывы цепей. Поэтому проверки повторяют после устранения коротких замыканий. Программы, используемые при проверке монтажа, выполняют:

выбор в цепях точки для контроля с учетом минимального расстояния между проводниками;

выдачу координат этих точек;

выдачу программ для работы автомата контроля.

Следующим этапом является контроль всего проектируемого блока на правильность функционирования. Получение подобных программ связано с синтезом тестов, моделирующих работу устройства и с разработкой устройств, на которых осуществляются подобные проверки. Обычно испытания проводят на постоянном (статические) и переменном (динамические) токе.

Испытания на постоянном токе включают в себя проверку статических параметров в рабочем режиме и проверку выполнения заданных логических функций без изменения уровня сигнала.

Динамические испытания включают проверку параметров при подаче на вход импульсного сигнала и испытание блока на выполнение заданной функции на рабочей частоте.

Устройства для статических и динамических испытаний отличаются друг от друга как в электронном, так и в программном отношениях, причем устройства для динамических испытаний сложнее и дороже.

Существующие пакеты прикладных программ (ППП), реализованные на ЕС ЭВМ выпуска технической документации, предназначены для формирования, организации, хранения и выпуска текстовых конструкторских документов, а также получения управляющей информации для технологических автоматов.

К основным решаемым задачам ППП относятся:

1. Выпуск комплекта конструкторской документации, который включает спецификации плат и ячеек, ведомости машинных носителей плат и ячеек, таблицу проводного монтажа, сборочный чертеж ячейки, перечень элементов и схему их расположения на плате, таблицы монтажа, частные технические условия, диаметры и координаты отверстий.

2. Выпуск управляющих перфолент для сверления печатных плат на станках, для формирования фотошаблонов слоев печатных плат на координатографах, для вычерчивания слоев и шаблонов печатных плат, для раскладки и пайки микросхем, для контроля печатного монтажа и ячеек.

Выходными результатами ППП являются комплекты КТД и комплект управляющих перфолент.

Для дальнейшего развития автоматизированного выпуска КТД необходимо:

1. Внести коррективы в ЕСКД и ЕСТД с учетом современной специфики получения КТД в САПР.
2. Обучать персоналы организаций работе с КТД, полученной автоматизированными методами.
3. Организовать хранение, обращение, корректировку информации на машинных носителях.
4. Все внешнее оборудование САПР (имеется в виду все графическое оборудование) должно быть соединено в единый комплекс не только информационно, а и аппаратурно, что позволит устранить промежуточные носители информации, непосредственное получение которых требует затрат времени, разработки программ.
5. Разрабатывать графические системы проектирования, позволяющие разработчику оперативно и по желанию вмешиваться в процесс проектирования, влиять на него. Подобные системы будут снабжены широкой сетью внешнего оборудования, задействованного на общий процессор или многопроцессорную вычислительную систему.
6. Уделять внимание разработке аппаратуры контроля монтажа, установочных элементов до и после сборки блока и апаратуре, диагностирующей работоспособность каждого узла и всего устройства в целом.

5.4. КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Сформулируйте основные требования к САПР КТД.
2. Какие случаются трудности при автоматизации выпуска КТД?
3. Постройте структурную схему САПР КТД.
4. Какой набор технических средств по вашему мнению должна содержать современная САПР КТД?
5. Какие алгоритмы и программы должны входить в САПР КТД?
6. Каким образом должно происходить распределение информации в САПР КТД?
7. Какие средства машинной графики вы знаете?
8. Приведите краткую характеристику средств машинной графики, их достоинства и недостатки.
9. Опишите основные принципы работы координатографов.
10. Опишите работу ЧА «Вектор-1301». Приведите его структурную схему.
11. Приведите основные виды и принципы работы кодировщиков.
12. Для каких целей используются микрофотонаборные установки? Дайте их краткую характеристику.
13. Опишите АРМ-М и постройте его структурную схему.
14. Дайте краткую характеристику и опишите структуру АРМ-Р.
15. Приведите технические средства АРМ-Р и опишите их назначение.
16. Опишите два режима работы АРМ-Р.
17. Для каких целей используется МО АРМ-Р?
18. Каким образом производится автоматизированный выпуск КТД?
19. Опишите программное обеспечение для выпуска графической документации.

20. Каким образом осуществляется выпуск КТД для этапов изготовления и контроля ЭА?
21. В чем заключаются статические и динамические испытания?
22. Дайте характеристику ППП для выпуска технической документации.
23. Какие комплексы программ используются для выпуска графической документации?
24. С чем, по вашему мнению, связано развитие автоматизированного выпуска КТД для ЭА на печатных платах и интегральных микросхемах?
25. Какой вы представляете САПР КТД в будущем?

6. АППАРАТНЫЕ СРЕДСТВА СВЯЗИ КОНСТРУКТОРА И ЭВМ

6.1. ОБЩИЕ ПОЛОЖЕНИЯ

Автоматизированное проектирование означает тесное взаимодействие человека с ЭВМ. Это взаимодействие достигается использованием дисплеев, телетайпов и других средств диалога человека с ЭВМ. Очевидно, что любой процесс проектирования носит итерационный характер, т. е. первый, пробный, результат проектирования изменяется, улучшается и т. д. Процесс проектирования и конструирования длится до тех пор, пока результат не будет удовлетворять наперед заданным требованиям. Такой процесс можно условно изобразить с помощью структурной схемы, показанной на рис. 6.1.

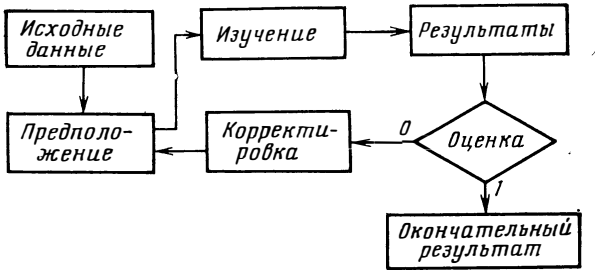


Рис. 6.1. Структурная схема процессора проектирования

Очевидно, что в системах, где ЭВМ используется для выдачи результатов в виде распечаток на АЦПУ или информации, управляющей работой внешнего устройства (например, графопостроителя или координатографа), процесс получения результатов, а следовательно, их оценка и корректировка затягиваются, что увеличивает время проектирования.

Более эффективным является процесс проектирования, при котором результаты выдаются на экран какого-либо устройства, позволяющего сразу же оценить их и внести необходимые изменения

в процесс проектирования. Такими устройствами являются алфавитно-цифровые и графические дисплеи, а системы проектирования, использующие графические дисплеи, называют графическими системами проектирования (ГСП). Они включают человека-оператора, аппаратуру и программное обеспечение. Аппаратура состоит из ЭВМ, графического дисплея (ГД), управляемого ЭВМ, а также устройства ввода, которые позволяют человеку вводить необходимые данные и команды. Программное обеспечение кроме обычного комплекса программ, входящего в состав любой системы проектирования, включает программы взаимодействия ЭВМ с дисплеем.

В настоящее время разработан ряд структур ГСП. Они отличаются друг от друга режимом работы используемых графических дисплеев, средств машинной графики (т. е. устройств ввода-вывода графической информации и программным обеспечением).

6.2. ДИСПЛЕИ

Из рис. 6.1 видно, что основным этапом ГСП является связь человека-оператора с ЭВМ в течение всего процесса проектирования. Эта связь осуществляется с помощью специального внешнего устройства, управляемого ЭВМ, которое называется дисплеем.

Дисплеи являются диалоговыми устройствами для интерактивного взаимодействия конструктора и ЭВМ. Одна из возможных упрощенных схем дисплея изображена на рис. 6.2. Оператор вводит исходные данные на основе клавиатуры. С клавиатуры сигналы проходят через блок управления в ЭВМ или на экран. Ответные сигналы идут через блок управления, где хранятся программы, обеспечивающие различные режимы функционирования терминала. Перемещение изображения на экране осуществляется с помощью координатора.

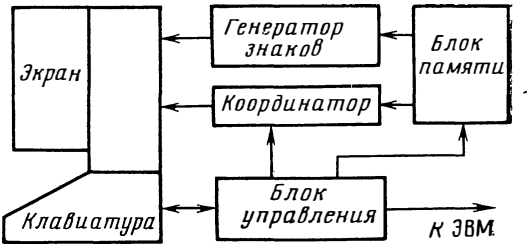


Рис. 6.2. Структурная схема дисплея

Современные дисплеи могут обрабатывать вводимые данные с помощью пакетов программ как без использования мощной центральной ЭВМ, так и при взаимодействии с ней. Наличие у дисплея памяти позволяет осуществлять диалог и графические операции. Такой терминал позволяет вводить данные с одновременным их контролем, редактировать тексты, изменять графические формы. Полный пакет программ вводится в память центральной ЭВМ, а в памяти терминала хранятся подпрограммы, позволяющие строить основные виды изображений и изменять их. Кроме того, такие терминалы позволяют выделять конструктору на экране от-

дельные элементы топологии, вводить стандартные элементы, документировать отображаемую информацию.

При обработке в ЭВМ в процессе проектирования графической информации для связи с ЭВМ используются графические дисплеи, которые представляют собой специализированное устройство с экраном, выполненным на электронно-лучевой трубке. Структура подобной ГСП показана на рис. 6.3. Дисплейный пульт в изображенной системе считается автономным, т. е. сохраняет изображение на экране даже после отключения от него центральной ЭВМ.

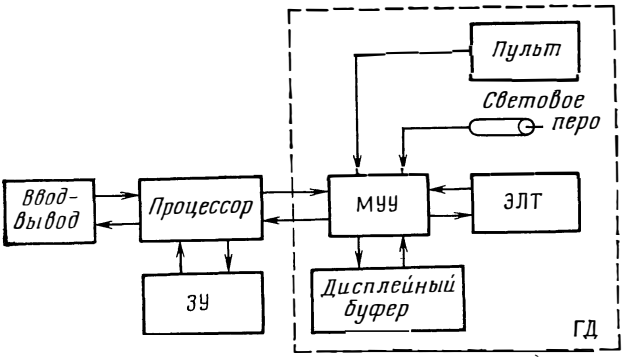


Рис. 6.3. Пример структурной схемы графической системы проектирования:
МУУ — местное устройство управления

Одним из факторов, ограничивающим в настоящее время развитие графических систем, является предельная сложность изображения, которое можно вывести на экран без переполнения памяти, выделенной для изображения. Вообще же ограничения в значительной степени связаны с техникой, используемой для генерации изображения, с самим дисплеем и режимом его работы.

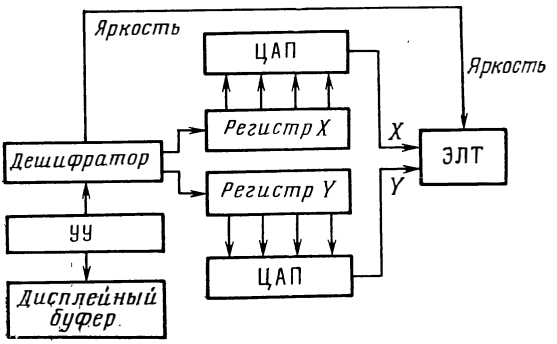


Рис. 6.4. Схема точечного режима

Рассмотрим ряд основных режимов работы дисплея.

Дисплей в точечном режиме действует как устройство отображения точек, т. е. каждое слово дисплейного списка указывает точку на экране (рис. 6.4). Здесь УУ — устройство управления; ЦАП — цифроаналоговый преобразователь; ЭЛТ — электронно-лучевая трубка. Дисплейное слово в этом режиме имеет вид, показанный на рис. 6.5.

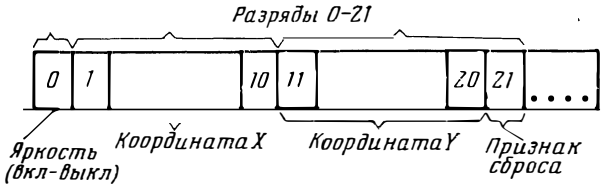


Рис. 6.5. Дисплейное слово при точечном режиме

Простота схемы и отсутствие ошибок накопления (так как координаты каждой точки задаются в абсолютной форме) — основные достоинства точечного дисплея. Однако он имеет много недостатков:

для воспроизведения изображения необходимо много дисплейных слов, что требует большого объема памяти;

для обработки большого числа слов требуется много машинного времени. Если как-то ограничить последнее требование, то автоматически уменьшается сложность изображений, выводимых на экран. Действительно, если время воспроизведения одной точки составляет 40 мкс, а минимальная частота регенерации точек (с учетом отсутствия мерцания) составляет 40 Гц, то изображение не может содержать более 1000 точек. Следовательно, точечный дисплей применим для изображения отдельных точек или прямых отрезков и неэффективен при изображении кривых линий.

При работе в векторном режиме каждое дисплейное слово показывает величину смещения по горизонтали и вертикали от i-й точки изображения к $(i+1)$-й. Вид дисплейного слова показан на рис. 6.6. Схема такого дисплея изображена на рис. 6.7. Очевидно, что данный режим экономичнее в смысле использования памяти, так как для изображения одного вектора не требуется сотен дисплейных слов, описывающих точки. Кроме то-

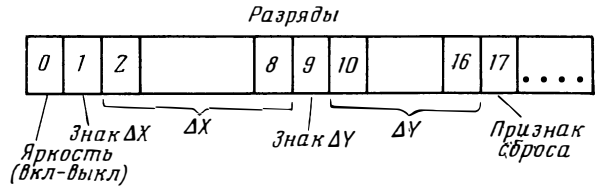


Рис. 6.6. Дисплейное слово при векторном режиме

го, этот режим позволяет перемещать изображение на экране, меняя опорную точку. Однако он менее точен, так как в процессе генерации изображения происходит накапливание ошибки величин перемещений (ΔX и ΔY).

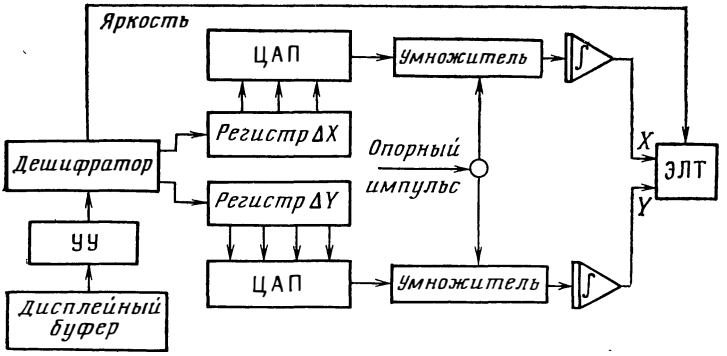


Рис. 6.7. Структурная схема дисплея

Рассмотрим другие режимы работы дисплея. Режим прямолинейных отрезков состоит в том, что линия задается на экране дисплея ее начальной и конечной точками координат X и Y. Вид дисплейного слова показан на рис. 6.8.

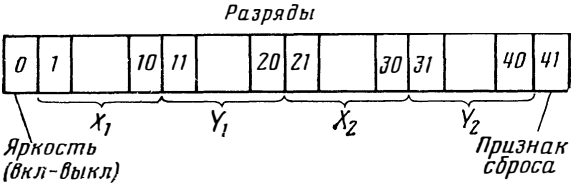


Рис. 6.8. Дисплейное слово при режиме прямолинейных отрезков

Режим коротких векторов используется тогда, когда в изображении преобладают кривые линии, и отличается от векторного режима только тем, что координаты векторов ΔX и ΔY задаются более короткими (менее 7) разрядными числами. Это в несколько раз уменьшает время воспроизведения вектора на экране по сравнению с точечным режимом.

Шаговый режим позволяет больше, нежели предыдущий, экономить память при описании коротких векторов. Так, в одном 12-разрядном дисплейном слове можно поместить описание двух коротких векторов (длина этих векторов колеблется от нуля до трех шагов). На рис. 6.9 приведен пример подобного дисплейного слова.

Из приведенного описания видно, что определение режима работы дисплея сводится к компромиссу между допустимой длиной

вектора, который может быть занесен в дисплейное слово, и объемом памяти буфера дисплея для хранения последовательности слов. Очевидно, что самым «расточительным» по памяти и самым медленным, но самым точным будет точечный режим. На экране с растром 1024×1024 в векторном режиме можно воспроизводить

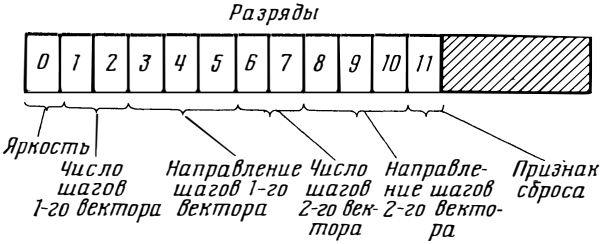


Рис. 6.9. Дисплейное слово при шаговом режиме

отрезки длиной 1/8 всего экрана, но это требует 18-разрядных дисплейных слов. В режиме коротких векторов длина отрезка может составлять 16 элементов растра (1/64 всего экрана), но это требует 12-разрядных слов. Наконец, в шаговом режиме длина отрезка не превышает трех элементов растра, но зато в 12-разрядном слове помещается описание двух векторов. Очевидно, что два последних режима подходят тогда, когда в изображении преобладают кривые линии.

Важными характеристиками работы дисплея являются параметры, определяющие удобство работы с ним. К этим характеристикам относятся: частота мерцания изображения; разрешающая способность; адресуемость точки на экране; форма экрана и полезная площадь.

Изображение на экране должно достаточно быстро восстанавливаться с частотой, позволяющей наблюдателю видеть качественное изображение без мерцания. Устранение мерцания достигается ценой дополнительных схемных затрат, снижения допустимой сложности изображения и других ухудшений параметров. Так как частота мерцания оказывает воздействие на зрение человека, то устранение мерцания представляет значительный интерес при разработке дисплея. Частота мерцания замеряется числом раз в секунду, за которое изображение целиком регенерируется или заново воссоздается на экране ЭЛТ. В результате исследований было установлено, что в нормально освещенном помещении требуется частота регенерации 30 Гц, а частота в 40 Гц практически полностью устраняет мерцание. Необходимо заметить, что при работе в затемненных помещениях или при использовании люминофора с большим послесвечением можно снизить допустимую частоту мерцания.

Под разрешающей способностью понимают возможность воспроизводить на экране дисплея самые мелкие детали чертежа. Мерой разрешающей способности является число различимых линий,

отнесенное к диаметру экрана. Графический дисплей обладает хорошей разрешающей способностью, если по ширине изображения можно провести порядка 1000 линий при малой яркости свечения линий. При этом необходимо помнить, что увеличение яркости ухудшает разрешающую способность.

Разрешающая способность экрана графического дисплея, т. е. электронно-лучевой трубки, всегда выше в области центра, нежели по краям. Это объясняется тем, что при максимальных углах отклонений электронного пучка ЭЛТ ухудшается его разрешающая способность. Например, в дисплеях с большими ЭЛТ по краям экрана понижались разрешающая способность и появлялось мерцание.

Адресуемость точки на экране измеряется числом координатных позиций, в которые может быть помещен центр пятна сфокусированного пучка. Она определяется числом разрядов дисплейного слова. Не нужно путать адресуемость точки на экране с разрешающей способностью. Например, с помощью 12-разрядного дисплейного слова можно разместить пятно в одной из 4096 дискретных позиций экрана ЭЛТ. При этом разрешающая способность дисплея может составлять 1000 линий. Обычно в графическом дисплее адресуемость точки делают примерно равной разрешающей способности.

В применяемых графических дисплеях используются как квадратные, так и прямоугольные экраны, стороны которых относятся как 4 : 3. В ряде случаев используются очень большие экраны, позволяющие изображать чертеж в масштабе 1 : 1. Полезная площадь экрана ЭЛТ выбирается такой, чтобы дискретные позиции адресуемости точки вписывались в круглый или прямоугольный формат экрана.

Люминофор, выбранный для ЭЛТ, играет важную роль в процессе работы с дисплеем. Люминофор определяет послесвечение, цвет, яркость и, что особенно важно, разрешающую способность графического дисплея. Если учесть, что послесвечение влияет на надежность ответного сигнала и в свою очередь зависит от мерцания, то становится ясным, что выбор наиболее удачного люминофора является компромиссом между выбранными характеристиками дисплея.

Дисплей с запоминающей ЭЛТ представляет пользователю возможность доступа к ЭВМ с дистанционных терминалов. В отличие от обычных, дисплеи с запоминающей ЭЛТ не требуют дисплейной буферной памяти, в этих дисплеях не стоит проблема мерцания, так как изображение сохраняется на экране ЭЛТ любое необходимое время. Объем информации, который необходимо вывести на экран, практически не ограничен, так как не ограничено время формирования изображения. В дисплеях с запоминающей ЭЛТ увеличена скорость передачи информации за счет включения в состав дисплея генераторов символов. Например, если для одиночного символа необходим 6-разрядный код, то на строке может быть воспроизведено 200 симв./с при скорости передачи 1200 бит/с

(алфавит состоит из 64 символов). На экране размером 12,5×20 см можно изобразить примерно 4000 символов.

Рассмотрим дисплей с использованием *pnp*-проекции, которая используется в дисплеях тогда, когда на экран ЭЛТ наряду с меняющейся графической информацией необходимо вывести практически не меняющиеся данные. Проекция *pnp* — это наложение на сформированное на экране ЭЛТ электронным лучом изображение со слайдов, пленок и т. д. Метод применяется при выводе на экран географических карт, изображение которых мало меняется, постоянных текстов-комментариев или редко меняющихся частей проектируемого устройства. Особенно удобно применение проекции *pnp* при программируемом обучении конструктора.

Последнее время широкое распространение получили дисплеи с цветными ЭЛТ. С помощью цвета можно выделить слои проектируемой микросхемы или печатной платы, выделить интересующие конструктора блоки проектируемого устройства или схемы и т. д. Несколько замедленное использование дисплеев с цветными ЭЛТ связано с их сложностью, которая приводит к снижению надежности и повышению стоимости.

Стереодисплеи используются в системах проектирования сложных устройств, например ЭВА на ИМС5, так как применение стереоскопии дает эффект глубины и позволяет изображать трехмерные объекты. При этом на экран выводится одно и то же изображение дважды под разными углами зрения, тем самым образуется стереопара. Это и является основным недостатком стереодисплеев, так как при необходимости выводить изображение на экран дважды не снижаются требования к величине частоты мерцания. Кроме того, ровно в 2 раза уменьшается разрешающая способность экрана стереодисплея, так как изображение одной компоненты стереопары занимает только половину экрана.

На экран любого графического дисплея можно вывести алфавитно-цифровую информацию (АЦИ). Однако изображение каждого знака будет представлять собой чертеж. Поэтому гораздо удобнее при работе только с АЦИ использовать специализированные дисплеи. Основным достоинством этих дисплеев являются незначительный объем входных данных и возможность подключения их к ЭВМ с помощью узкополосной линии связи. Как правило, АЦИ на экране дисплея пишется по строкам, менять размеры, наклон и конфигурацию символов нельзя.

6.3. СТРУКТУРЫ ДАННЫХ

В графических системах проектирования часто возникает необходимость идентифицировать, повторно вызывать, изменять элементы изображения на экране и манипулировать с ними. Решение этих задач привело к необходимости разрабатывать информационные модели, т. е. разрабатывать удобные описания графических объектов, выводимых на экран дисплея. Обособленный участок

памяти фиксированной длины, в котором описан конкретный графический элемент или объект, называется блоком данных этого графического элемента или объекта. Рассмотрим блок данных, описывающий различные типы треугольников, которые изображаются на экране в векторном режиме (табл. 6.1).

Здесь 0-строка-идентификатор объекта, описанного в блоке данных. В нашем случае — это «треугольник»; 1-я строка — координаты начальной (опорной) точки изображаемого объекта; 2, 3, 4-я строки — векторное представление 1, 2, 3-й сторон треугольника, заданное в виде приращения по осям X и Y, и параметр луча «включено-выключено»; 5-я строка — дополнительная информация о объекте. Это может быть любая необходимая информация, в том числе и АЦИ.

Каждый блок данных имеет свой адрес, по которому он отыскивается в ЭВМ. Адрес блока данных — это адрес его первого слова в оперативной памяти ЭВМ. Записав в памяти ЭВМ все блоки данных, описывающие графические объекты, зная размер каждого из них и адрес первого блока, отыскивается необходимый блок данных. Однако такой поиск требует больших временных затрат. Зачастую необходимо один и тот же блок вызывать несколько раз в разные моменты работы ГСП.

Таблица 6.1

Вид блока данных

№	Идентификатор блока — треугольник
1	X_0, Y_0
2	I_1, ΔX_1, ΔY_1
3	I_2, ΔX_2, ΔY_2
4	I_3, ΔX_3, ΔY_3
5	Яркость и т. п

Во многих случаях оператор, начиная работу, не знает всех зависимостей и связей между элементами модели, неизвестны ему и размеры самой модели. Эти данные появляются в процессе решения. Поэтому, формируя блоки данных, оператор отводит для них завышенный объем памяти. Стремление более гибко формировать и анализировать модель графического объекта привело к созданию сложных структур данных.

Сложной структурой данных (ССД) называют совокупность объектов, каждый из которых сопровождается своим блоком данных, а соотношения между объектами заданы в явной форме. В ССД сами объекты задаются блоками данных и представляют собой основной предмет моделирования, а основные соотношения между объектами задаются с помощью указателей. Указатель — это слово, находящееся в специальном месте блока данных и содержащее адрес блока данных, связанного с этим блоком.

Наибольшее распространение в ГСП нашли ассоциативные и иерархические формы организации ССД. Ассоциативную форму ССД удобно использовать для указания объектов с похожими свойствами, описанных блоками данных, разбросанными по различным участкам памяти ЭВМ. Тогда ассоциативная форма ССД позволяет быстро отыскивать все объекты с одинаковыми свойствами.

Иерархическая форма ССД используется тогда, когда между объектами существуют отношения иерархического типа, например ячейка ТЭЗ, стойка. Каждый блок данных иерархической ССД содержит три указателя: первый указатель содержит адрес блока данных, связанный с данным и расположенный на более высоком уровне; второй — на более низком; а третий — адрес блока данных, расположенный на том же уровне.

6.4. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ГРАФИЧЕСКОЙ СИСТЕМЫ ПРОЕКТИРОВАНИЯ

Выше были рассмотрены аппаратные средства и структуры данных ГСП. Рассмотрим программное обеспечение (ПО) этих систем.

Важной задачей ПО ГСП является передача данных между прикладными программами и аппаратурой дисплея. Прикладные программы могут формировать выходную информацию или непосредственно в форме команд для дисплея, или созданием некоторых наборов данных. Та часть ПО ГСП, которая предназначена для работы с аппаратурой дисплея, должна приводить описание изображения к виду, удобному для обработки схемой управления дисплеем. Программное обеспечение ГСП должно обладать способностью определять вид и характер отображаемой информации.

При разработке ПО ГСП решаются две основные задачи: разработка и использование процедур отображения и выбор языка программирования. Смысл процедур (программ) отображения разберем на примерах.

Предположим, что программа анализирует схемы, которые содержат резисторы. Она будет содержать процедуру RESISTOR, которая рисует на экране дисплея резистор. Процедура описывает резистор концами линий. Описание зависит от выбранного языка программирования и подобно следующему примеру:

```
DEFINE RESISTOR
LINE FROM (0, 0) TO (20, 0) TO (30, 10) TO (50, 10), TO (70, 10)
TO (10, 10) TO (100, 0) TO (120, 0)
END RESISTOR
```

После выполнения подобной процедуры на экране дисплея будет получен рисунок резистора.

Описание более сложного изображения, например фильтра, содержащего в своем составе резисторы и конденсаторы, составляет процедура FILTRA, которая по мере надобности вызывает процедуры RESISTOR и CAPACITOR. (Процедура CAPACITOR аналогична процедуре RESISTOR и служит для вывода на экран дисплея изображения конденсатора.)

Пример процедуры FILTRA:

```
DEFINE FILTRA
CAPACITOR AT (0, 120)
```

```
CAPACITOR AT (120, 120)
RESISTOR—ROTATED (Pi/2) AT (120, 0)
LINE FROM (0, 0) TO (240, 0)
CIRCLE SCALE (0, 1) AT (0, 120)
CIRCLE SCALE (0, 1) AT (0, 0)
CIRCLE SCALE (0, 1) AT (240, 120)
CIRCLE SCALE (0, 1) AT (240, 0)
END FILTRA
```

Язык программирования, на котором пишется прикладная программа, состоящая из процедур, должен быть пригодным как для программирования прикладных задач, так и для управления графическими устройствами. Это означает, что в языке должны быть развитые средства общего назначения, что обеспечивается языками программирования высокого уровня общего назначения, которые расширяют средствами для работы с графическими устройствами. Вид расширений и способы их реализации и использования зависят от вида графических устройств и способа использования их в ГСП. В простейших случаях ограничиваются введением пакета процедур работы с графическими устройствами, написанных по правилам синтаксиса выбранного языка программирования. Вызов процедур осуществляется так, как это принято в выбранном языке программирования.

Работа ПО ГСП основана на проведении диалога между программами и оператором посредством дисплея. Поэтому при составлении ПО ГСП необходимо описать форму диалога, а именно составить входной язык и обеспечить в программе обработку его инструкций. Эту задачу можно решить как с помощью любых языков высокого уровня общего назначения, так и с помощью специальных диалоговых языков.

Процесс проектирования ГСП состоит из четырех этапов:
спецификация объектов проектирования;
системное проектирование (выбор компонентов системы);
детальная разработка компонентов и связей;
оценка возможностей системы.

Обычно эти этапы выполняются последовательно, но результат, полученный на любом этапе, может потребовать изменения предыдущих решений. Процесс проектирования ГСП состоит из иерархических уровней. На верхних уровнях иерархии рассматривают требования пользователя, определяют внешние параметры системы. На следующем этапе выбирают компоненты системы, которые смогут обеспечить заданные внешние параметры. При этом составляют более детальную схему ГСП. После этого проектируются все специальные устройства системы. В результате выполнения этого этапа формируется структура системы, определяется перечень используемых аппаратных и программных средств, включая выбранную ЭВМ, устройства ввода-вывода графической

информации, структуры данных и описание дисплея. После того как установлена конфигурация системы, выполняется проектирование компонентов и связей, т. е. выполняется детальный проект системы. На последнем этапе процесса проектирования выполняется оценка всей системы. На этом этапе конструктор оценивает общие параметры системы, увязывает оставшиеся нерешенные проблемы согласования различных частей системы. На этом этапе конструктор решает следующие три основные задачи: определяет способность аппаратных средств и программного обеспечения выполнять поставленную задачу; оценивает время, занимаемое аппаратными и программными средствами ГСП на выполнение поставленных задач в интерактивном режиме; оценивает стоимость системы.

Рассмотрим, например, работу типичной ГСП. Оператор вводит световым пером в ЭВМ данные об элементах и подложке проектируемого устройства, принципиальная схема которого находится перед ним. Подложка изображается на экране, и на ней размещаются чертежи стандартных элементов, которые вызываются из библиотеки. Затем, используя программные средства системы, оператор выполняет соединение элементов. При этом оператор может стирать и переносить элементы, уменьшать и увеличивать изображение, записывать новые фрагменты схемы в библиотеку. Результаты проектирования записываются на магнитную ленту (МЛ). Информация на МЛ обрабатывается на ЭВМ, и формируется перфолента для чертежной установки, на которой можно получать и трафареты, и необходимые чертежи.

В ГСП используется дисплей с круглым экраном. Его центральная часть — это квадрат. Область слева используется для изображения знаков совмещения и установки. Область экрана справа используется для ввода кодов операций двух основных классов: вспомогательных и основных. (К этим операциям относятся: подсчет общего числа отрезков прямых, перемещение контура, стирание на экране фрагмента и ввод в ЭВМ идентификаторов элементов схемы, ввод в ЭВМ размеров и параметров подложки, операции точной доводки резистора и т. д. соответственно.) В нижней части экрана при необходимости воспроизводится необходимая алфавитно-цифровая информация, а в верхней части экрана изображаются таблицы, списки элементов схем и т. д. Работа с дисплеем осуществляется с помощью светового пера и клавиатуры. Все программное обеспечение системы, в том числе и программы, реализующие «графическую» часть, т. е. работу с дисплеем, хранятся на магнитных дисках и при необходимости вызываются в оперативную память. На магнитных дисках хранится и библиотека стандартных решений.

В системе проектирования трафаретов обычно используются сложные структуры данных списочного типа, как, например, показано на рис. 6.10. Каждый блок на рисунке соответствует метке или линии. Стрелки, идущие слева направо,— прямые указатели— показывают связь меток и линий между собой. Стрелки, идущие в обратном направлении,— обратные указатели—служат для объе-

динения линий в один элемент. Линии *L1—L4* образуют резистор, а *L5—L6* — связи резистора, *R5* — идентификатор резистора. Каждый блок данных имеет свой код. По этому коду в ЭВМ отыскивается блок данных.

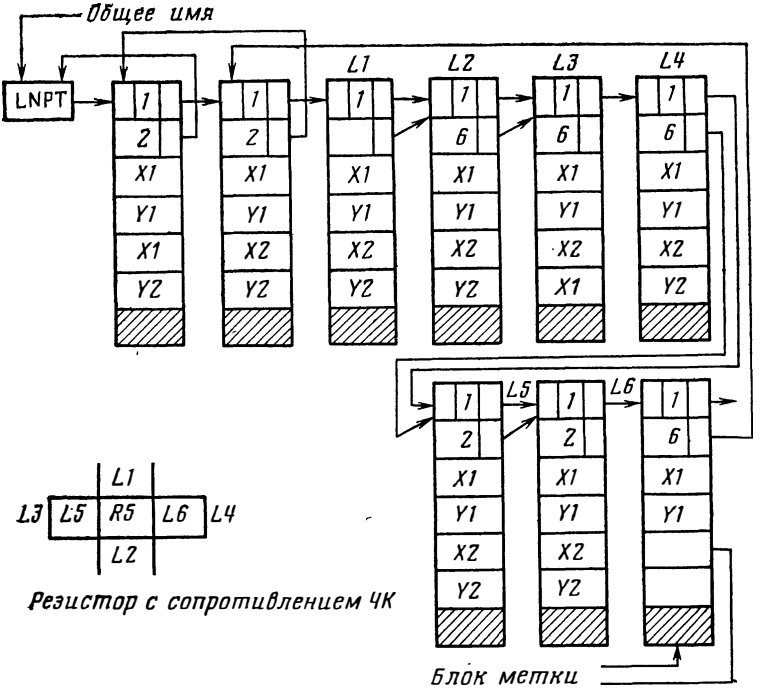


Рис. 6.10. Пример ССД списочного типа

Описанная ГСП предназначается для изготовления гибридных микросхем. В результате работы системы получается полный набор трафаретов для изготовления микросхем, изготовляемых с помощью графического дисплея. Система позволяет как проводить процесс проектирования, так и вводить с помощью пульта ввода-вывода графическую информацию о принципиальной схеме в ЭВМ.

Можно отметить, что ГСП — это комплекс средств ввода-вывода графической информации, а устройство отображения графической информации — это графический дисплей. Комплекс технических средств объединен устройствами связи и управляется ЭВМ в соответствии с ПО ГСП. Широкие возможности при проектировании ГСП предоставляет комплекс технических средств АРМ. Наличие в АРМ ГД, УВВГИ, управляемых мини-ЭВМ, возможность стыковки АРМ и ЕС ЭВМ, позволяют конструктору уделить максимум внимания на разработку программного обеспечения ГСП.

В настоящее время известно большое количество УВВГИ, ГД и ЭВМ, которые можно применять в ГСП. При выборе конкретного устройства необходимо руководствоваться не только техническими возможностями аппаратуры, но и ее стоимостью. При проектировании ГСП необходимо помнить, что, какой бы «хорошей» не оказалась система, она будет лишь периферийным устройством, используемым ЭВМ и предназначенным для ввода-вывода графической информации. Так будет до тех пор, пока ГСП не станут главным фактором при проектировании структур самих ЭВМ. Наиболее сложным при проектировании ГСП является выбор той части решаемой задачи, которую наиболее эффективно решать в интерактивной системе.

6.5. КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Приведите структурную схему итерационного процесса проектирования.
2. Какие устройства используются для внесения изменений в процесс проектирования?
3. Какие системы проектирования называются графическими?
4. Перечислите аппаратуру, входящую в состав ГСП.
5. По каким путям может пойти развитие ГСП в будущем?
6. Приведите структурную схему дисплея.
7. Приведите структурную схему ГСП, использующую графические дисплеи.
8. Опишите точечный режим работы дисплея и приведите схему такого режима.
9. Какой вид имеет дисплейное слово при точечном режиме работы дисплея?
10. Приведите структурную схему, вид дисплейного слова при векторном режиме работы дисплея.
11. Какие другие режимы работы дисплея вы знаете? Какой вид имеют дисплейные слова в соответствующих режимах?
12. Приведите основные характеристики работы дисплеев.
13. Какие специальные виды дисплеев вы знаете?
14. Что называется блоком данных графического элемента?
15. Приведите пример блока данных.
16. Опишите структуры данных в ГСП.
17. Что представляет собой сложная структура данных?
18. Опишите применение ассоциативной и иерархической формы организации ССД.
19. Что представляет собой ПО ГСП?
20. Приведите примеры, иллюстрирующие работу ПО ГСП.
21. Опишите этапы проектирования ГСП.
22. Приведите примеры работы типичной ГСП.
23. Дайте описание ССД списочного типа. Приведите примеры.
24. Приведите структурную схему ГСП, работающую в интерактивном режиме, которую можно будет применять для создания ЭА пятого и последующих поколений.

ЗАКЛЮЧЕНИЕ

В связи с совершенствованием электронной аппаратуры происходит непрерывное совершенствование теории и практики автоматизации конструирования. В этой связи рассмотрим некоторые тенденции развития автоматизации конструирования электронной аппаратуры.

Предполагается, что в будущем еще более возрастут темпы автоматизации разработки новых средств РЭА и ЭВА на основе комплексных САПР. Причем основное внимание будет уделено разработке модульных САПР, способных настраиваться на решение любых задач, принимать решения в расплывчатых условиях, анализировать техническое задание, производить выбор из нескольких альтернативных вариантов, производить оптимизацию конструкции ЭА по комплексным критериям, учитывать требования на механическую прочность, тепловые режимы, электромагнитную совместимость, оптимально анализировать электронные схемы и синтезировать конструкции, производить совместное проектирование, учитывая требования логического, конструкторского и технологического проектирования.

Основными теоретическими вопросами при разработке математического обеспечения САПР ЭА, по мнению авторов, явятся: исследование оптимальных графо-теоретических моделей конструкций ЭА; назначение элементов, контактов схем; совместная компоновка с размещением на основе комплексных критериев; определение планарности и эффективная плоская укладка соединений; оптимальное расслоение соединений; параллельная трассировка соединений.

В настоящее время наблюдается стремление к универсализации алгоритмов конструирования, разработке совместных контролепригодных алгоритмов компоновки, размещения и трассировки, построению точных алгоритмов конструирования на основе поисковых методов, одновременному учету физических и топологических особенностей схем, построению алгоритмов конструирования на основе комплекса базовых подалгоритмов, разработке специализированных приставок для быстрой реализации наиболее трудоемких алгоритмов конструирования, разработке новых формальных и адекватных математических моделей схем ЭА.

Конструктор ЭА будущих поколений должен обладать знаниями в области вычислительной техники, программирования, теории алгоритмов, графов, множеств, быть способным разрабатывать и эксплуатировать системы автоматизированного проектирования.

Перечисленные в учебном пособии проблемы конструирования ЭА, конечно, не охватывают весь комплекс задач, стоящий перед современным конструктором, использующим вычислительную технику, но они показывают, что возможности повышения эффективности и качества ЭА на основе автоматизации неисчерпаемы.

Конструирование и производство ЭА будущих поколений на основе автоматизации является универсальной специальностью будущего.

СПИСОК ЛИТЕРАТУРЫ

Основная литература

1. **Деньдобренко Б. Н., Малика А. С.** Автоматизация конструирования РЭА. — М.: Высшая школа, 1980. — 384 с.
2. **Кузнецов О. П., Адельсон-Вельский К. М.** Дискретная математика для инженера. — М.: Энергия, 1980. — 344 с.
3. **Мелихов А. Н., Берштейн Л. С., Курейчик В. М.** Применение графов для проектирования дискретных устройств. — М.: Наука, 1974. — 304 с.
4. **Морозов К. К.** и др. Методы разбиения схем РЭА на конструктивно законченные части. — М.: Сов. радио, 1978. — 136 с.
5. **Норенков И. П.** Введение в автоматизированное проектирование технических устройств и систем. — М.: Высшая школа, 1980. — 308 с.
6. **Основы** проектирования микроэлектронной аппаратуры/Под ред. Б. Ф. Высоцкого. — М.: Советское радио, 1977. — 351 с.
7. **Петренко А. И., Тетельбаум А. Я.** Формальное конструирование электронно-вычислительной аппаратуры. — М.: Сов. радио, 1979. — 256 с.
8. **Селютин В. А.** Машинное конструирование электронных устройств. — М.: Сов. радио, 1977. — 384 с.
9. **Сигорский В. П.** Математический аппарат инженера. — Киев: Техніка, 1977. — 766 с.
10. **Справочник** конструктора РЭА, общие принципы конструирования/Под ред. Р. Г. Варламова. — М.: Сов. радио, 1980. — 478 с.
11. **Теория** и методы автоматизации проектирования вычислительных систем/Под ред. М. Брейера. — М.: Мир, 1977. — 285 с.
12. **Томашевский Д. И., Масютин Г. Г., Явич А. А., Преснухин В. В.** Графические средства автоматизации проектирования РЭА. — М.: Сов. радио, 1980. — 244 с.

Дополнительная литература

13. **Абрайтис Л. Б., Шейнаускас Р. И., Жилевичус В. А.** Автоматизация проектирования ЭВМ. — М.: Сов. радио, 1978. — 272 с.
14. **Автоматизация** поискового конструирования/Под ред. А. И. Половинкина. — М.: Радио и связь, 1981. — 344 с.
15. **Алферова З. В.** Теория алгоритмов. — М.: Статистика, 1973. — 164 с.
16. **Ахо А., Хопкрофт Дж., Ульман Дж.** Построение и анализ вычислительных алгоритмов. — М.: Мир, 1979. — 536 с.
17. **Базилевич Р. П.** Декомпозиционные и топологические методы автоматизированного конструирования электронных устройств. — Львов: Вища школа, изд-во при Львовском университете, 1981. — 168 с.
18. **Баранов С. И., Майоров С. А., Сахаров Ю. П., Селютин В. А.** Автоматизация проектирования цифровых устройств. — Л.: Судостроение, 1979. — 264 с.
19. **Батищев Д. И.** Поисковые методы оптимального проектирования. — М.: Сов. радио, 1975. — 216 с.
20. **Бахтин Б. И.** Автоматизация в проектировании и производстве печатных плат радиоэлектронной аппаратуры. — Л.: Энергия, 1979. — 120 с.
21. **Вентцель Е. С.** Исследование операций. М.: Наука, 1980. — 208 с.
22. **Автоматизация** проектирования печатных блоков с модулями произвольной формы/Е. П. Герасименко и др. — М.: Машиностроение, 1979. — 167 с.
23. **Гудман С., Хидетниеми С.** Введение в разработку и анализ алгоритмов. — М.: Мир, 1981. — 366 с.
24. **Жимерин Д. Г., Мясников Д. А.** Автоматизированные и автоматические системы управления. — М.: Энергия, 1979. — 592 с.
25. **Карапетян А. М.** Автоматизация оптимального конструирования ЭВМ. — М.: Сов. радио, 1973. — 152 с.
26. **Комплекс** общеотраслевых руководящих методических материалов по созданию АСУ и САПР. Государственный комитет СССР по науке и технике. — М.: Статистика, 1980. — 119 с.

27. **Конструирование** и расчет БГИС, микросборок и аппаратуры на их основе/Под ред. Б. Ф. Высоцкого. — М.: Радио и связь, 1981. — 215 с.
28. **Кристофидес Н.** Теория графов. — М.: Мир, 1978. — 432 с.
29. **Кузин Л. Т.** Основы кибернетики. Т. 2. — М.: Энергия, 1979. — 584 с.
30. **Курейчик В. М.** Автоматизация технического проектирования вычислительных структур. — Электронная промышленность, 1979, вып. 4.
31. **Майоров С. А., Смирнов А. А.** ЭВМ — справочник по конструированию. — — М.: Сов. радио, 1975. — 504 с.
32. **Морозов К. К., Одиноков В. Г.** Использование ЭЦВМ при конструировании некоторых узлов РЭА. — М.: Сов. радио, 1972. — 104 с.
33. **Проектирование** монтажных плат на ЭВМ/К. К. Морозов, А. Н. Мелихов, В. Г. Одиноков и др. — М.: Сов. радио, 1979. — 224 с.
34. **Нилсон Н.** Искусственный интеллект. — М.: Мир, 1973. — 270 с.
35. **Петренко А. И., Курейчик В. М.** и др. Автоматизация проектирования больших и сверхбольших интегральных схем. — Зарубежная радиоэлектроника, 1981, № 6.
36. **Петренко А. И., Тетельбаум А. Я., Шрамченко Б. Л.** Автоматизация конструирования электронной аппаратуры. — Киев: Вища школа, 1980. — 176 с.
37. **Перснухин Л. Н., Шахнов В. А., Кустов В. А.** Основы конструирования микроэлектронных вычислительных машин. — М.: Высшая школа, 1976. — 408 с.
38. **Принс М. Д.** Машинная графика и автоматизация проектирования. — М.: Сов. радио, 1975. — 232 с.
39. **Рейнгольд Э., Нивергельт Ю., Део Н.** Комбинаторные алгоритмы. Теория и практика. — М.: Мир, 1980. — 478 с.
40. **Рубцов В. П., Захаров В. П., Жижко В. А.** Автоматизация проектирования больших интегральных схем. — Киев: Техніка, 1980. — 232 с.
41. **Уокер Б. С., Гурд Дж. Р., Дроник Е. А.** Интерактивная машинная графика.— М.: Машиностроение, 1980. — 168 с.
42. **Шиханович Ю. А.** Введение в современную математику. — М.: Наука, 1965. — 376 с.

ПРИЛОЖЕНИЕ

СПИСОК ГОСТ, РЕКОМЕНДУЕМЫХ ДЛЯ ИСПОЛЬЗОВАНИЯ ПРИ ИЗУЧЕНИИ КУРСА «АВТОМАТИЗАЦИЯ КОНСТРУИРОВАНИЯ»

1. **ГОСТ 18682—73.** Микросхемы интегральные. Классификация и система условных обозначений.
2. **ГОСТ 17021—75.** Микросхемы интегральные. Термины и определения.
3. **ГОСТ 2.113—75.** Групповые и базовые конструкторские документы.
4. **ГОСТ 20406—75.** Платы печатные. Термины и определения.
5. **ГОСТ 2.105—68.** Общие требования к текстовым документам.
6. **ГОСТ 2.108—68.** Спецификация.
7. **ГОСТ 2.117—71.** Согласование применения покупных изделий.
8. **ГОСТ 21033—75.** Система «Человек—машина».
9. **ГОСТ 22487—77.** Проектирование автоматизированное. Термины и определения.
10. **ГОСТ 23501.0—79.** Системы автоматизированного проектирования. Основные положения.
11. **ГОСТ 23501.7—80.** САПР. Предпроектные исследования.
12. **ГОСТ 23501.2—79.** Системы автоматизированного проектирования. Разработка, согласование и утверждение технического задания.
13. **ГОСТ 23501.5—80.** Системы автоматизированного проектирования. Эскизный проект.
14. **ГОСТ 23501.6—80.** Системы автоматизированного проектирования. Технический проект.
15. **ГОСТ 23501.4—79.** Системы автоматизированного проектирования. Общие требования к программному обеспечению.
16. **ГОСТ 23501.9—80.** Системы автоматизированного проектирования. Общие требования к банкам данных.
17. **ГОСТ 23501.8—80.** Системы автоматизированного проектирования. Классификация и обозначения.
18. **ГОСТ 2.413—72.** Единая система конструкторской документации. Правила выполнения конструкторской документации изделий, изготовляемых с применением электрического монтажа.
19. **ГОСТ 2.708—72.** ЕСКД. Правила выполнения электрических схем цифровой вычислительной техники.
20. **ГОСТ 2.031.—77.** ЕСКД. Документы на перфокартах и перфолентах. Основные надписи.
21. **ГОСТ 19001—77.** Единая система программной документации. Назначение, состав, классификация.
22. **ГОСТ 19002—80.** Схемы алгоритмов и программ. Правила выполнения.
23. **ГОСТ 19003—80.** Схемы алгоритмов и программ. Обозначения условные, графические.
24. **ГОСТ 23501.15—81.** Системы автоматизированного проектирования. Ввод в действие.
25. **ГОСТ 23501.16—81.** Системы автоматизированного проектирования. Диалоговые средства. Общие требования.
26. **ГОСТ 19105—78.** ЕСПД. Общие требования к программным документам.
27. **ГОСТ 19106—78.** ЕСПД. Требования к программным документам, выполненным печатным способом.
28. **ГОСТ 23501.3—79.** Системы автоматизированного проектирования. Разработка, согласование, утверждение технического предложения.
29. **ГОСТ 2501.1—79.** Системы автоматизированного проектирования. Стадии создания.

ОГЛАВЛЕНИЕ