Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет телекоммуникаций

Кафедра сетей и устройств телекоммуникаций

С. Б. Саломатин

МОДЕЛИРОВАНИЕ АЛГОРИТМОВ БЫСТРОЙ ОБРАБОТКИ СИГНАЛОВ В ИНФОКОММУНИКАЦИОННЫХ СЕТЯХ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники для направлений специальности 1-45 01 01-02 «Инфокоммуникационные технологии (сети телекоммуникаций)» и 1-45 01 01-05 «Инфокоммуникационные технологии (системы распределения мультимедийной информации)» в качестве учебно-методического пособия

Минск БГУИР 2016

УДК 621.391-047.58:654(076)
ББК 32.811.3я73
С16

Р е ц е н з е н т ы:

кафедра информационных систем и технологий
Белорусского национального технического университета
(протокол №1 от 06.09.2014);

доцент кафедры дискретной математики и алгоритмики
Белорусского государственного университета, кандидат технических наук
Ю. В. Свирид

Саломатин, С. Б.

С16 Моделирование алгоритмов быстрой обработки сигналов в инфокоммуникационных сетях : учеб.-метод. пособие / С. Б. Саломатин. – Минск : БГУИР, 2016. – 132 с. : ил.

ISBN 978-985-543-159-7.

Излагается материал моделирования алгоритмов быстрой обработки сигналов. Алгоритмы рассматриваются с точки зрения теории графов, решеток и временного ряда. Приводятся быстрые алгоритмы ортогонального разложения, пространственной дискретизации, решения задачи поиска кратчайшего вектора в базисе решетка, сингулярного анализа временного ряда трафика. Эффективность алгоритмов оценивается на основе теории сложности.

Предназначено для студентов, изучающих дисциплины «Теория электрической связи» и «Алгоритмы цифровой обработки сигналов».

УДК 621.391-047.58:654(076)
ББК 32.811.3я73

ISBN 978-985-543-159-7

СОДЕРЖАНИЕ

Введение

Главное назначение сетей на сегодняшний момент – передача информации. В настоящее время существует множество сетевых технологий со своими особенностями, критериями применимости и пр. Оценить современное состояние сферы телекоммуникаций невозможно без понимания генезиса, принципов функционирования существующих систем и сетей связи. Объединение компьютеров в сеть изменило парадигму обработки информации.

Сеть связи – это совокупность линий связи и промежуточного оборудования/промежуточных узлов, терминалов/оконечных узлов, предназначенных для передачи информации от отправителя до получателя с заданными параметрами качества обслуживания.

Линия связи представляет собой совокупность физической среды распространения сигналов и оборудования, формирующих специализированные каналы, имеющие определенные стандартные показатели: полосу частот, скорость передачи и т. п.

Каналы связи могут быть непрерывными (аналоговыми) и дискретными (цифровыми).

Учебно-методическое пособие логически разбито на 5 частей.

В первой главе рассматриваются модели инфокоммуникационных сетей. Дается обзор существующих моделей сетей связи. Рассматриваются графовые модели инфокоммуникационной сети, случайные геометрические графы. Приводятся такие базовые геометрические сетевые модели как пуассоновские точечные процессы, мозаика Вороного и граф Делоне, булева модель сети. Определяются общие математические модели и методы управления сетевыми ресурсам и управления ресурсами телекоммуникационной сети.

Во второй главе освещаются вопросы математических моделей сигналов инфокоммуникационных сетей. Приводятся алгебраическо-графовые модели сигналов и фильтры на графах. Рассматриваются модели временных рядов и массивов, модели пакетного трафика и фрактальные свойства пакетного трафика сетей.

В третьей главе рассматриваются методы, оценивающие сложность алгоритмов обработки сигналов в инфокоммуникационных системах. Даются базовые понятия алгоритмов. Определяются понятия алгебраической сложности, асимптотических оценок, сложности в среднем. Рассматриваются рандомизированные и оптимальные алгоритмы. Определяются классы сложности. Оценивается сложность базовых алгоритмов первичной обработки сигналов, таких как свертка, рекурсия, спектральные преобразования. Приводятся поясняющие примеры.

В четвертой главе изучаются быстрые алгоритмы обработки сигналов. Даются основные типы алгоритмов обработки сигналов, алгоритмы дискретизации и квантования на основе теории решеток. Рассматриваются понятия систем обработки сигналов, основные математические операции, связанные с операциями на матрицах. Даются алгоритмы обработки

динамических сигналов. Рассматриваются такие дискретные ортогональные преобразования, как быстрые преобразования Фурье (БПФ), рекуррентные алгоритмы БПФ, преобразование Карунена – Лоэва, полиномиальное представление, преобразование на основе полиномов Чебышева, преобразования на основе системы дискретных базисных функций чисел Фибоначчи. Даются оценки вычислительной сложности алгоритмов. Рассматриваются алгоритмы ортогонального разложения случайных процессов. Приводятся алгоритмы обработки пространственных сигналов на основе теории решеток. Определяются алгоритмы обработки сигналов на основе теории графов и в частности быстрые алгоритмы сегментации графов.

Пятый раздел посвящен алгоритмам анализа трафика инфокоммуникационной системы. Рассматриваются вопросы выделения трендовых и периодических составляющих случайных процессов. Приводится пример анализа с использованием сингулярного спектра сетевого трафика локальной сети.

1. МАТЕМАТИЧЕСКИЕ МОДЕЛИ ИНФОКОММУНИКАЦИОННЫХ СЕТЕЙ СВЯЗИ

Сеть связи – это совокупность линий связи и промежуточного оборудования/промежуточных узлов, терминалов/оконечных узлов, предназначенных для передачи информации от отправителя получателю с заданными параметрами качества обслуживания.

1.1. Модели инфокоммуникационной сети на основе теории графов

Сеть связи (телекоммуникационная сеть) как объект синтеза и анализа представляет собой совокупность пунктов сети и соединяющих их линий. В качестве математической модели такого объекта используют граф [1–4].

Определение. Граф $G(V, E)$ – совокупность двух множеств: вершин V и ребер E, между которыми определено отношение инцидентности. Каждое ребро e из E инцидентно равно двум вершинам v' и v'', которые оно соединяет. При этом вершина v' и ребро e называются инцидентными друг другу, а вершины v' и v'' – смежными.

Вершины можно обозначить строчными буквами (i, j, k, l, s) либо цифрами (1, 2, 3, 4, 5), а дуги соответственно парами: $\{(i, j), (j, k), (k, l) \ldots\}$ либо {(1,2), (2,3), (3,4), ... }, где первый индекс определяет начало, а второй – конец дуги (рис. 1.1).

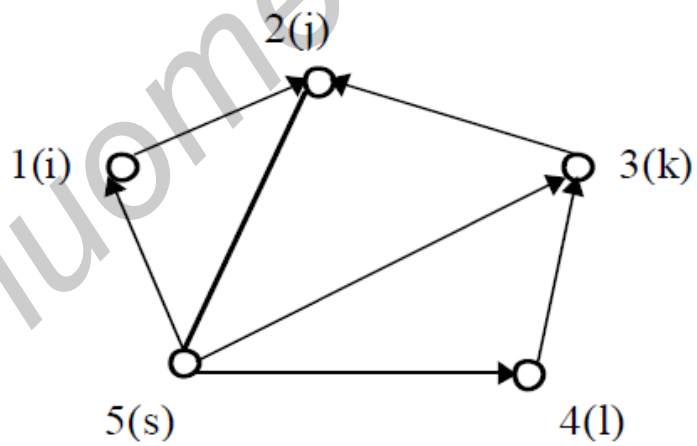


Рис. 1.1. Модель ориентированного графа

Граф, в котором задается направление дуг, называется ориентированным, в противном случае – неориентированным. Неориентированные дуги называются ребрами.

Между двумя вершинами, соединенными дугой (ребром), существует отношение смежности. Для ориентированного графа вершины i и j смежны, только если дуга начинается в i и направлена в j.

Граф, каждой дуге (ребру) которого поставлены в соответствие некоторые числовые характеристики, называемые весами, представляет собой взвешенный граф. При необходимости веса могут быть приписаны также вершинам графа.

Взвешенный граф принято называть сетью (в данном случае имеется в виду сетевая модель, а не сама сеть как объект). В качестве весовых характеристик сети могут выступать расстояния, пропускная способность, стоимость и т. д.

Помимо геометрического изображения в виде точек и линий, граф может быть представлен в дискретной, матричной форме. Именно эта форма используется при вводе графовой модели в ЭВМ.

Модель сети. Сетевая инфокоммуникационная система представляет собой распределенную структуру, размещенную на определенной территории. Схема функционирования ее задается с помощью графа. Ребра r_i графа соответствуют физическим компонентам, таким, например, как каналы связи, проложенные от одной вершины графа (узла) V_i к другому.

Узлы графа соответствуют источникам/приемникам потоков либо осуществляют транзитные функции для существующих потоков. Такие вершины графа носят название транзитных. Совокупность ребер, которую надо пройти потоку из вершины V_i до вершины V_j, называется путем (V_i, V_j). Если для любых двух вершин графа существует путь (V_i, V_j), то граф называется связанным, в противном случае граф не связан. Каждое ребро, входящее в вершину или исходящее из нее, называется *инцидентным* этой вершине. Общее количество ребер $d(i)$, инцидентных вершине, называется степенью вершины. Если граф ориентирован, то различают степень исхода $d^+(i)$ и захода $d^-(i)$.

Сечением по ребрам называют наименьшее количество ребер, удаление которых от графа разбивает последний на два несвязанных между собой компонента.

Разбиение множества V всех вершин графа на два подмножества $V_1 \cup V_2 = V$ называется разрезом по вершинам. Это разбиение определяет разрез по ребрам, $E_1 \cup E_2 = E$, где E – множество всех ребер, выходящих из вершины V_1 и входящих в вершину V_2. Если элементу графа (ребру, вершине) приписана физическая величина (например, длина ребра, пропускная способность, задержка обработки информации), эта величина отмечается числом, называемым весом элемента (ребра, вершины).

Представление графов. Представление сети связи в виде графа является адекватным и позволяет применить математический аппарат теории графов для задач моделирования сетей связи.

Существуют 2 способа представления графа $G = (V, E)$: в виде набора списков смежных вершин и в виде матрицы смежности.

Первый способ предпочтительней для *разреженных* графов, когда $|E|$ много меньше $|V|^2$. Представление с помощью матрицы смежности удобнее использовать для плотных графов, когда $|E|$ сравнимо с $|V|^2$.

Представление графа $G = (V, E)$ в виде списков смежности (рис. 1.2, 1.3) использует массив Adj из $|V|$ списков по одному для каждой вершины. Для каждой вершины список смежных вершин Adj из $[u]$ содержит в произвольном порядке все смежные с ней вершины v, для которых существует ребро $(u, v) \in E$.

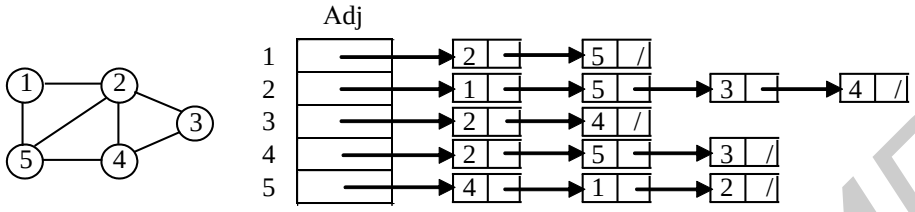


Рис. 1.2. Списки смежности для неориентированного графа

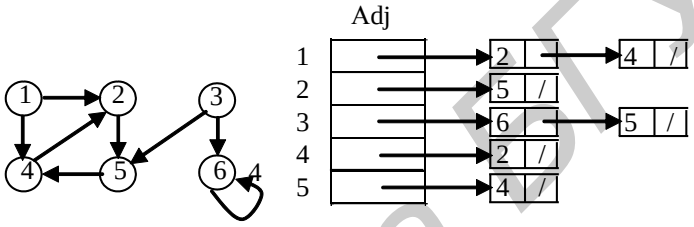


Рис. 1.3. Списки смежности для орграфа

Для неориентированного графа сумма длин всех списков равна удвоенному числу ребер, т. к. ребро (u, v) порождает элемент в списке вершины ***u*** и вершины ***v***. Для орграфа сумма длин списков равна числу ребер. В обоих случаях количество памяти для представления графа оценивается как $O(V + E)$.

Списки смежности удобны, когда наряду с номером вершины нужно хранить дополнительную информацию (например, вес ребра, родителя вершины, метку вершины и т. д.).

Недостатком представлений является необходимость просматривать весь список, чтобы узнать есть ли ребро из вершины *u* и *v*. Этого можно избежать в представлении с помощью матрицы смежности (рис. 1.4).

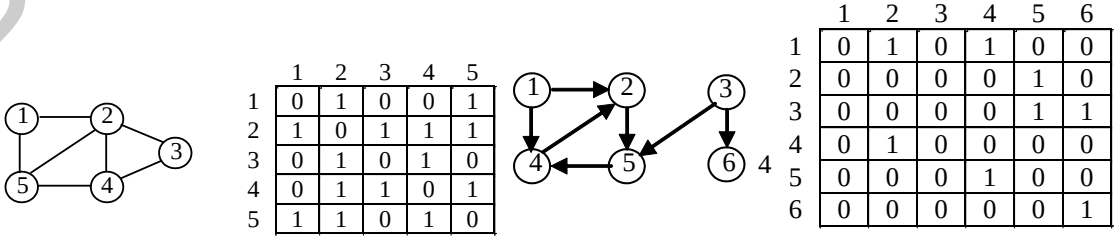


Рис. 1.4. Примеры матриц смежности графов

Элемент матрицы смежности размером $|V| * |V|$ равен единице $a_{i,j} = 1$, если есть ребро или дуга $(i,j) \in E$, и равен нулю в противном случае. Для орграфа и неориентированного графа матрица смежности требует $O(V^2)$ памяти.

Для неориентированного графа матрица смежности симметрична относительно главной диагонали. Поэтому достаточно хранить только элементы от главной диагонали и выше.

В матрице смежности вместо 0 и 1 можно хранить веса ребер $w(u,v)$. Для отсутствующих ребер можно использовать специальные значения *NIL* (0 или ∞).

Матрица смежности удобнее для небольших графов. Если не нужно хранить веса, то для элементов матрицы смежности можно использовать биты и упакованное битовое представление.

Лапласиан графа. Рассматриваемые графы полагаются неориентированными и не имеющими петель; все векторы считаются столбцами размерности $n; n \geq 3;$. Расстояние между векторами понимается в смысле евклидовой нормы – вектор, все компоненты которого единицы.

Лапласианом графа *G* называется матрица $\boldsymbol{Q}$:

$$\boldsymbol{Q} = \boldsymbol{D} - \boldsymbol{A}, \tag{1.1}$$

где $\boldsymbol{D}$ – диагональная матрица степеней вершин.

Собственные значения $\boldsymbol{Q}$ обозначаются как $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Матрица $\boldsymbol{Q}$ – сингулярная М-матрица, поэтому $\lambda_k = 0$ $(1 \leq i \leq n)$ [1]. Собственный вектор, соответствующий λ_k $(1 \leq k \leq n)$, обозначается $\mathbf{x}_k$ и называется k-м собственным вектором Q. Поскольку $\lambda_1 = 0$, первым собственным вектором $\mathbf{x}_1$ можно считать любой ненулевой вектор. Так как G – связный граф, матрица $\boldsymbol{Q}$ – неприводима

1.2. Случайные геометрические графы

В случайных геометрических графах узлы распределены по закону точечного процесса Пуассона в R^d с интенсивностью λ, а ребра расположены между всеми парами узлов, расстояние между которыми не превосходит r. Такие графы, также известные как *непрерывная перколяция*. Булевы модели широко используются в качестве моделей для сетей связи.

Несмотря на свою простоту, эти модели устанавливают важные качественные свойства реальных сетей, такие как фазовый переход в подключении по мере увеличения интенсивности λ пуассоновского точечного процесса. Для захвата мобильности узлов является нормальным моделирование их движения путем независимого броуновского движения. Назовем это моделью мобильного геометрического графа (*Mobile Geometric Graph Model – MGG*).

Сосредоточимся на трех проблемах модели:

– *обнаружение* – время, в течение которого цель – фиксированная или движущаяся – проходит на расстояние r узла MGG;

– *покрытие* – время, в течение которого все точки в большом наборе будут обнаружены;

– *перколяция* – время, в течение которого заданный узел соединен с бесконечным соединенным компонентом.

Пусть $\Pi_0 = \{X_i\}_i \Pi_0 = \{X_i\}_i$ – точечный пуассоновский процесс в R^d с интенсивностью λ. Каждый узел X_i движется в соответствии с законом броуновского движения $(\zeta_i(s))_{s\geq 0}$ независимо от других узлов, и пусть $\Pi_s = \{X_i + \zeta_i(s)\}_i$ – точечный процесс, полученный после движения Π_0 спустя период времени s. Отсюда следует, что Π_s – пуассоновский процесс такой же интенсивностью λ.

В любой момент времени s построим граф G_s путем соединения любых двух узлов Π_s, расстояние между которыми не превосходит r. В дальнейшем будем считать значение r постоянной величиной.

Обнаружение. Пусть u – частица, помещенная в первоначальное состояние. Предположим, что положение u в момент времени s определяется как $g(s)$ и является непрерывным процессом в R^d. Нас интересует время, необходимое для обнаружения процессом частицы u, а именно: за какое время узел процесса Π_0 пройдет расстояние не более r из положения u. Более формально определим

$$T_{det} = \inf\left\{t \geq 0: g(t) \in \bigcup_i B(X_i + \zeta_i(t), r)\right\},$$

где объединение берется по всем узлам $\{X_i\}$ процесса Π_0 и $B(x, r)$ обозначает шар радиусом r с центром в точке x.

Покрытие. Пусть A есть подмножество R^d. Требуемое для обнаружения всех точек A определим как

$$T_{cov} = \inf\left\{t \geq 0: A \subset \bigcup_{s\leq t}\bigcup_i B(X_i + \zeta_i(t), r)\right\}.$$

Перколяция. Пусть u будет дополнительным узлом в нулевом состоянии, движущимся независимо от узлов процесса Π_0 в соответствии с некоторой функцией g. Время T_{perc}, необходимое для того, чтобы u принадлежал бесконечной связанной компоненте, запишется как

$$T_{perc} = \inf\{t \geq 0: \exists y \in C_\infty(t)\ s.t. \|g(t) - y\|_2 \leq r\}.$$

1.3. Базовые геометрические сетевые модели

Пуассоновские точечные процессы. Пуассоновский точечный процесс является основой стохастической геометрии для моделирования сетей связи.

Под стационарным пуассоновским процессом с интенсивностью λ понимают точечный процесс Ф со следующими свойствами:

1) число $\Phi(B)$ точек в множестве B имеет распределение Пуассона с параметром $\lambda|B|$, где $|\cdot|$– мера Лебега:

$$P\{\Phi(B) = k\} = e^{-\lambda|B|}\frac{\lambda|B|^k}{k!};$$

2) количество точек Ф в непересекающихся множествах независимо.

Простейшей и наиболее употребляемой моделью стационарного точечного процесса является стационарный пуассоновский процесс. Он соответствует «чисто случайному» хаотическому расположению точек, что нередко встречается в природе. Пуассоновский процесс образует основу для построения большинства сложных моделей точечных процессов, случайных замкнутых множеств и случайных мер.

Мозаика Вороного и граф Делоне. Математический принцип мозаики Вороного широко используется в качестве простой идеализации множества сложных «настоящих» разделений плоскости. Граф Делоне может быть использован как модель протокола соседних сетей – топология маршрутизации.

Булева модель. Модель покрытия беспроводной сети не учитывает влияния помех и использует булеву алгебру (булева модель). Она может быть использована в качестве базовой модели для изучения непрерывной перколяции и для решения вопросов связи одноранговых сетей в отсутствии помех. Модель использует теорию случайных замкнутых множеств. Булева модель часто применяется для описания сложных структур.

Пусть задан стационарный пуассоновский процесс $\Phi = \{(X_i, G_i)\}$. Булевой моделью называется следующее множество:

$$\Xi = \bigcup_i X_i \oplus G_i,$$

где $x \oplus G = \{x + y : y \in G\}$.

Точки X_i часто называют «ростками», множества G_i – «зернами».

Данная модель является классической моделью стереологии и стохастической геометрии. При малых λ булева модель используется как модель для «случайно распределенных» множеств. При $\lambda \to \infty$ все большее число «зерен» перекрывает друг друга, но чаще «зерна» бывают изолированными друг от друга, и тогда модель в лучшем случае служит

хорошим приближением. Булева модель часто используется в качестве модели для хаотически расположенных геометрических образов, причем в качестве «зерен» для простоты берутся шары и круги.

Булева модель является общей моделью покрытия и связности. Точки указывают положения переменных структурных элементов (устройств) сети, а зерна – независимые области, обслуживаемые этими устройствами.

Модель дробового шума. Математическое представление помех основано на процессах дробового шума (пуассоновских процессах), которые позволяют описывать различные результаты покрытия, связи, емкости (производительности) сетей с ограниченным влиянием помех.

Пусть $\Phi = \{X_i, S_i\}$ – маркированный точечный процесс, где $\{X_i\}$ – точки, а $\{S_i\}$ – независимые одинаково распределенные случайные величины. С учетом переходной функции $L\{y - X_i\}$ расстояние на плоскости определим через поле дробового шума:

$$I_\Phi(y) = \sum_i S_i \, L(y - X_i).$$

Если Ф – маркированный точечный процесс, то I_Φ – пуассоновский дробовой шум.

Для пуассоновского дробового шума преобразование Лапласа вектора $\big(I_\Phi(y_1), \dots, I_\Phi(y_n)\big)$ известно для любого $y_1, \dots, y_n \in \mathbb{R}^2$ (через преобразование Лапласа пуассоновского точечного процесса).

Модель дробового шума является хорошей моделью помех в беспроводных сетях. При этом S_i соответствуют излучаемой энергии, а переходная функция – функции затухания.

Модель покрытия отношения уровня сигнала к уровню шума. Введем следующие обозначения:

$\Phi = \{X_i, (S_i, T_i)\}$ – маркированный точечный процесс (пуассоновский);

$\{X_i\}$ – точки точечного процесса пространства;

R^2 – положения антенн;

$(S_i, T_i) \in (R^+)^2$ – случайная метка точки;

X_i – мощность, позволяющая определить порог.

Процесс покрытия определяется как

$$\Xi(\Phi; W) = \bigcup_{i \in \mathbb{N}} C_i(\Phi, W),$$

где C_i – область, в которой отношение уровня сигнала к уровню шума X_i больше, чем порог T_i.

Пример. Диаграмма Вороного беспроводной сенсорной сети.

Беспроводные сенсорные сети WSN (Wireless Sensor Network) представляют собой самоорганизующиеся сети, состоящие из множества беспроводных сенсорных узлов, распределенных в пространстве и предназначенных для мониторинга характеристик окружающей среды или объектов, расположенных в ней. Пространство, которое покрывается сенсорной сетью, образует сенсорное поле.

Беспроводные сенсорные узлы – это миниатюрные устройства с ограниченными ресурсами: зарядом батареи, объемом памяти, вычислительными возможностями и т. д. Однако объединение большого числа этих элементов в сеть обеспечивает возможность получения реальной картины происходящего в рамках сенсорного поля.

Архитектура беспроводной сенсорной сети изображена на рис. 1.5.

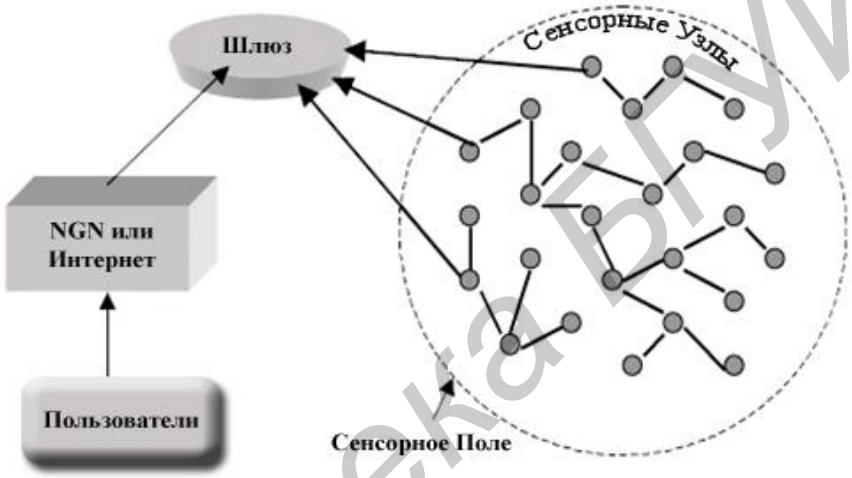


Рис. 1.5. Архитектура беспроводной сенсорной сети

Кластерная организация (см. рис. 1.5) считается эффективной и масштабируемой для решения подобных задач, но лишь при условии рационального выбора головного узла в кластерной сети в конкретный момент времени. Действительно являющийся головным в момент времени t_1 сенсорный узел не обязательно должен быть им же в момент времени t_2, т. к. существующий головной узел уже может затратить достаточно большое количество энергии на передачу сообщений от всех сенсорных узлов кластера к моменту времени t_2. Поэтому в момент времени t_2 головным узлом в кластере может быть назначен и иной сенсорный узел, сохранивший к этому времени наибольший энергетический запас.

1.4. Математические модели и методы управления сетевыми ресурсами

Развитие современных сетей связано со структурно-функциональной декомпозицией территориально-распределенной телекоммуникационной сети (ТКС) на ряд подсетей, которые в IP-технологии именуются автономными системами (Autonomous Systems, AS), а в технологии асинхронного режима

передачи (Asynchronous Transfer Mode, ATM) – кластерами. В первом случае в рамках одной AS маршрутизация организована с использованием протоколов внутреннего шлюза IGP (Interior Gateway Protocols), а между AS – с помощью протоколов внешнего шлюза EGP/BGP (Exterior/Border Gateway Protocols).

В технологии ATM маршрутизация осуществляется с использованием протокола иерархической маршрутизации PNNI (Private Network – to – Network Interface) [2].

В основу современных протоколов маршрутизации положенны графовые модели и комбинаторные алгоритмы расчета кратчайшего пути (RIP, IGRP – алгоритм Беллмана – Форда; OSPF, PNNI – алгоритм Дейкстры).

Основные требования, предъявляемые к моделям управления, касаются учета динамичности и стохастичности протекающих в ТКС процессов информационного обмена с ориентацией на реализацию иерархическо-координационной стратегии управления.

Математическая модель управления ресурсами телекоммуникационной сети. В общем случае структура ТКС может быть представлена в виде графа $G(V, D)$, множество вершин которого составляют сетевые узлы-маршрутизаторы $\{V_j, j = \overline{1, N}\}$, где N – число узлов в ТКС, а множество дуг $\{D_{i,j}: i, j = \overline{1, N}; i \neq j\}$ моделирует тракты передачи между маршрутизаторами. В соответствии с иерархическим представлением ТКС в виде совокупности взаимодействующих подсетей (автономных систем, кластеров) представим каждую из них в виде подграфа $G_q(V^q, D^q)$ графа $G(V, D)$, в котором V^q и N_q – соответственно подмножество маршрутизаторов и их общее число в q-й подсети ТКС; V^q – множество трактов передачи ТКС, принадлежащих q-й подсети. Пусть общее число подсетей равно Q. Условимся, что подмножества V^q не пересекаются, а различные подмножества D^q $(q = \overline{1, Q})$ могут иметь общие элементы.

С функциональной точки зрения в соответствии с вышеперечисленными требованиями заслуживает внимания математическая модель маршрутизации и распределения канальных и буферных ресурсов, представленная системой разностных стохастических уравнений состояния ТКС:

$$x_{i(q),j(g)}(k+1) = x_{i(q),j(g)}(k) - \sum_{\substack{V_i \in V^q \\ l \neq i(q)}}^{N_q} b_{i(q),l}(k) u_{i(q),l}^{j(g)}(k) +$$

$$+ \sum_{\substack{V_i \in V^q \\ m \neq i(q),j(g)}}^{N_q} b_{m,i(q)}(k) u_{m,i(q)}^{j(g)}(k) +$$

$$+\sum_{\substack{p=1\\p\neq q}}^{Q}\left[\sum_{\substack{V_i\in V^q\\n\neq j(g)}}^{N_q} b_{n,i(q)}(k)u_{n,i(q)}^{j(g)}(k)-\sum_{V_i\in V^q}^{N_q} b_{i(q),l}(k)u_{i(q),l}^{j(g)}(k)\right]+y_{i(q),j(g)}(k),$$

где $b_{m,i(q)}(k)=c_{m,i(q)}(k)\Delta t$; $c_{i(q),i(q)}(k)\Delta t$ – скорость передачи данных от узла V_i^q к узлу V_j^p в момент времени t_k; $x_{i(q),j(g)}(k)$ – объем данных, находящихся на узле $x_{i(q),j(g)}(k)$ и предназначенных для передачи узлу V_j^g в момент времени t_k; $u_{i(q),l}^{j(g)}(k)$ – доля пропускной способности тракта передачи $D_{i(q),l(g)}$, выделенная информационному потоку с адресом V_j^p в момент времени t_k и трактуемая как управляющая переменная; $y_{i(q),j(g)}(k)=\xi_{i(q),j(g)}(k)\Delta t$, $\xi_{i(q),j(g)}(k)$ – интенсивность поступления данных от абонентов ТКС на узел V_i^q в момент времени t_k с адресатом узла V_j^g; $\Delta t=t_{k+1}-t_k$ – период перерасчета управляющих переменных.

Характерной особенностью данной модели является то, что именно численные значения управляющих переменных $u_{i(q),l}^{j(g)}(k)$ обусловливают конечный результат решения задач маршрутизации, распределения канального ресурса, а также непосредственно определяют порядок решения задач управления очередями.

Контрольные вопросы

1. Поясните, как можно использовать теорию графов для построения модели инфокоммуникационной сети.
2. Дайте определение графа.
3. Какие существуют методы определения графа?
4. Дайте определение лапласиана графа.
5. Какую информацию несут собственные значения и функции лапласиана графа?
6. Дайте определение булевой модели сети связи.
7. Поясните модель покрытия и связности сети.
8. Как оценить отношение уровня сигнала к уровню шума с помощью геометрической сетевой модели?
9. Поясните суть математической модели управления ресурсами сети.

2. МАТЕМАТИЧЕСКИЕ МОДЕЛИ СИГНАЛОВ ИНФОКОММУНИКАЦИОННЫХ СЕТЕЙ

2.1. Формы спектрального описания сигналов

В современном описании сигналов наиболее часто используют спектральные и алгебраические формы представления [5–7]. Вначале определим спектральную форму для непрерывных одномерных сигналов общего вида $x(\xi)$, где ξ – некоторый аргумент, в частном случае время *t*.

При этом сигнал на заданном интервале его определения $[\xi_{\min}, \xi_{\max}]$ рассматривается как совокупность элементарных сигналов $\varphi_a(\xi)$, умноженных на коэффициенты c_a и составляющих систему функций $\{\varphi_a(\xi)\}$ определенного типа:

$$x(\xi) = \sum_{a=0}^{\infty} c_a \varphi_a(\xi). \qquad (2.1)$$

При этом система функций $\{\varphi_a(\xi)\}$ называется базисной, а представление сигнала в виде (2.1) – это его разложение по системе базисных функций, или обобщенный ряд (многочлен). Если сигнал $x(\xi)$ является комплексным, то и коэффициенты c_a и система базисных функций $\{\varphi_\alpha(\xi)\}$ также будут являться комплексными.

Если система функций выбрана, то сигнал полностью характеризуется набором (вектором) спектральных коэффициентов $\{c_a\}$ – его спектром.

В общем случае ряд (2.1) для непрерывных сигналов содержит бесконечное число членов. При практических расчетах такой ряд обычно ограничивают (усекают). В этом случае представление сигнала будет приближенным:

$$x(\xi) \approx x[\xi] = \sum_{a=0}^{N-1} c_a \varphi_a(\xi) \qquad (2.2)$$

и имеет место аппроксимация сигнала $x(\xi)$ конечным рядом (2.2).

Выбирая приближенное описание сигнала, естественно, стремятся к тому, чтобы оно наилучшим образом соответствовало оригиналу. При этом каждый раз необходимо формулировать критерий приближения, т. к. в выражение «наилучшее приближение» можно вкладывать различный смысл.

Приведем наиболее широко применяемые критерии приближения (сходимости).

1. Можно потребовать, чтобы максимальное значение погрешности аппроксимации $\Delta(\xi) = x(\xi) - x[\xi]$ было минимальным на заданном интервале определения функции $x(\xi)$. Этот вид аппроксимации, при котором минимизируется величина $|\Delta(\xi)|_{\max}$, называется *равномерным приближением*.

2. В качестве критерия приближения можно выбрать среднюю погрешность

$$\Delta_{\text{ср}} = \frac{1}{T}\int_{\xi_{\min}}^{\xi_{\max}} |x(\xi) - x[\xi]|d\xi, \tag{2.3}$$

где $T = \xi_{\max} - \xi_{\min}$.

Такая аппроксимация называется приближением в среднем.

3. Если в качестве меры представления принимается минимум среднеквадратичной погрешности

$$\sigma = \sqrt{\frac{1}{T}\int_{\xi_{\min}}^{\xi_{\max}} \left[x(\xi) - x[\xi]\right]^2 d\xi}, \tag{2.4}$$

то такой вид аппроксимации называется приближением в среднеквадратичном.

Существуют и другие критерии приближения [5–7]. В большинстве технических применений преимущественное распространение получил среднеквадратичный критерий, учитывающий интегральный эффект – ошибку, накопленную на всем интервале определения сигнала, – и в большинстве случаев лучше соответствующий физическому смыслу исследуемых явлений. Кроме того, что тоже немаловажно, теория, основанная на этом критерии, имеет наиболее простой и удобный для практики вид.

Все рассмотренные критерии приближения взаимосвязаны. Если ряд (2.2) сходится к $x(\xi)$ равномерно, то он тем более сходится среднеквадратично. Из среднеквадратичной сходимости вытекает сходимость в среднем.

Для того чтобы разложение сигнала в виде (2.1) было возможным, система базисных функций (СБФ) должна удовлетворять следующему ряду требований:

1. Быть упорядоченной системой линейно независимых функций.

2. Быть полной, для того чтобы по выбранной системе функций можно было разложить любой сигнал из заданного множества.

3. Число линейно независимых функций в полной системе должно быть равным размерности рассматриваемого множества сигналов, т. е. количеству чисел, с помощью которых можно выбрать любой сигнал из этого множества. Когда рассматривается множество непрерывных сигналов произвольной

формы, то их размерность бесконечно велика и в этом случае СБФ должна содержать также бесконечно большое число линейно независимых функций.

Наиболее удобно производить разложение сигналов, если базисная система $\{\varphi_a(\xi)\}$ является ортогональной на интервале определения сигнала $[\xi_{\min}, \xi_{\max}]$. Условие ортогональности двух различных базисных функций заключается в равенстве нулю их взаимной мощности:

$$\frac{1}{T}\int_{\xi_{\min}}^{\xi_{\max}} \varphi_a(\xi)\varphi_k(\xi)d\xi = Q_a\delta_{a,k}, \tag{2.5}$$

где символ Кронекера и мощность a-й базисной функции

$$\delta_{a,k} = \begin{cases} 0 & \text{при} \quad a \neq k, \\ 1 & \text{при} \quad a = k, \end{cases} \tag{2.6}$$

$$Q_a = \frac{1}{T}\int_{\xi_{\min}}^{\xi_{\max}} \varphi_a^2(\xi)d\xi. \tag{2.7}$$

Интервал определения ортогональных базисных функций называется также интервалом ортогональности.

Если система ортогональных функций полная, то к ней нельзя добавить ни одной новой функции, которая была бы ортогональна одновременно ко всем другим функциям данной системы. Любую систему линейно независимых функций можно ортогонализировать, т. е. преобразовать в ортогональную систему.

Представление сигналов с помощью ортогональных СБФ обладает важным свойством: повышение порядка аппроксимирующего многочлена всегда улучшает аппроксимацию по сравнению с представлением сигналов не ортогональными СБФ. Если при $N \to \infty$ многочлен $x[\xi]$ (см. (2.2)) сходится к $x(\xi)$, то $x[\xi]$ совпадает с $x(\xi)$ в рамках выбранного критерия приближения.

Система ортогональных функций называется также нормированной, если мощности всех базисных функций равны единице (в этом случае СБФ называется еще ортонормированной):

$$\frac{1}{T}\int_{\xi_{\min}}^{\xi_{\max}} \varphi_a^2(\xi)d\xi = 1.$$

Любую систему ортогональных функций можно нормировать, если разделить каждую базисную функцию на ее мощность.

При представлении сигналов в форме (2.2) необходимо решать вопрос о способе вычисления спектральных коэффициентов. Он во многом будет зависеть от используемого метода аппроксимации (вида принятого критерия сходимости).

В случае применения среднеквадратичного критерия коэффициенты c_a выбирают таким образом, чтобы среднеквадратичная ошибка σ была минимальной. Это достигается с помощью обобщенной формулы Фурье расчета спектра:

$$c_a = \frac{1}{T}\int_{\xi_{min}}^{\xi_{max}} x(\xi)\varphi_a(\xi)d\xi. \qquad (2.8)$$

Очевидно, что среднеквадратичная аппроксимация имеет смысл тогда, когда мощность сигнала $x(\xi)$ и функций $\varphi_a(\xi)$ на интервале аппроксимации имеет конечное значение. В случае комплексных базисных систем в формуле (2.8) расчета спектра должна стоять комплексно-сопряженная функция $\varphi_a^*(\xi)$.

Увеличивая неограниченно число членов в аппроксимирующем многочлене с коэффициентами в форме (2.8), получим в пределе равенство $x^*(\xi) = x(\xi)$, выполняемое при $\sigma \to \infty$. В этом случае аппроксимирующий многочлен примет вид бесконечного ряда, называемого обобщенным рядом Фурье.

В спектральном представлении (2.1) и в формуле расчета спектра (2.8) базисные функции являются функциями двух переменных ξ и a, а спектральные коэффициенты – функциями переменной a. Это приводит к симметрии выражений (2.1) и (2.8), называемых соответственно прямым и обратным преобразованиями Фурье, из которой следует математическое равноправие функций $x(\xi)$ и c_a как различных форм представления сигнала. Для рядов Фурье справедливо равенство Парсеваля:

$$\frac{1}{T}\int_{\xi_{min}}^{\xi_{max}} \varphi_a^2(\xi)d\xi = \sum_{a=0}^{\infty} Q_a c_a^2. \qquad (2.9)$$

Так как правая часть этого равенства определяет мощность сигнала при его представлении с помощью спектров, а левая – его мощность при записи в виде математической функции, то равенство Парсеваля отражает эквивалентность двух форм представления сигналов с физической (энергетической) точки зрения. Выполнение равенства Парсеваля свидетельствует также о полноте ортогональной СБФ.

Рассмотрим представление с помощью спектров дискретных сигналов. Решетчатая функция $x[i]$, $i \in [0, N]$ записывается в виде обобщенного дискретного ряда

$$x[i] = \frac{1}{N}\sum_{i=9}^{N-1} \mathrm{c}_a \varphi_k(i) \tag{2.10}$$

по любым полным и ортогональным системам решетчатых базисных функций $\{\varphi_a(i)\}$. При этом моменты отсчетов базисных функций должны совпадать с моментами отсчетов раскладываемых сигналов.

Условия ортогональности и нормированности дискретных СБФ определяются уравнениями

$$\frac{1}{N}\sum_{i=9}^{N-1} \varphi_a(i)\varphi_k(i) = Q_a \delta_{a,k};$$

$$Q_a = \frac{1}{N}\sum_{i=0}^{N-1} \varphi_a^2(i) = 1, \tag{2.11}$$

а равенство Парсеваля для дискретных сигналов имеет вид

$$\frac{1}{N}\sum_{i=0}^{N-1} x^2[i] = \frac{1}{N}\sum_{a=0}^{N-1} Q_a c_a^2. \tag{2.12}$$

Для дискретных функций, удовлетворяющих условию

$$\frac{1}{N}\sum_{i=0}^{N-1} x^2[i] < \infty,$$

справедлива следующая формула для определения спектра:

$$c_a = \frac{1}{Q_a N}\sum_{i=0}^{N-1} x[i]\varphi_a(i). \tag{2.13}$$

Формулы (2.10) и (2.13) представляют собой дискретные преобразования Фурье.

Системы базисных функций. Один и тот же сигнал может быть разложен по различным СБФ или, что одно и то же, рассмотрен в различных системах координат. При этом внутренние закономерности сигналов не могут нарушаться при изменении системы координат. Однако спектральному анализу в различных СБФ соответствует различная физическая интерпретация и, что особенно важно, различная практическая реализация. Так, например, технические характеристики (точность, быстродействие, затраты памяти и оборудования) цифровых фильтров, построенных в спектральной области, зависят от применяемых СБФ и для различных систем существенно различны. В соответствии с этим при решении практических задач целесообразно подбирать наиболее подходящие СБФ. Выбор базиса во многом обусловлен спецификой решаемых задач и требованиями, предъявляемыми при их решении.

Полных и ортогональных СБФ существует бесчисленное множество. Дадим краткий обзор некоторых известных СБФ, применяемых в настоящее время в теории и практике обработки сигналов.

Системы единичных функций. Два прямоугольных импульса, не перекрывающие друг друга, ортогональны. Поэтому система прямоугольных импульсов, приставленных друг к другу и заполняющих интервал $[t_0, t_N]$, будет ортогональной системой.

Такая система полна только для подмножества ступенчатых сигналов с шириной ступени Δt, где Δt – длительность импульсов, $N = T/\Delta t$ – число импульсов на рассматриваемом интервале. Система таких функций будет полна для любого непрерывного сигнала при $\Delta t \to 0$ и $N \to \infty$. В этом случае она превращается в систему единичных импульсов $\{u_a\,(t)\}$, имеющих единичную амплитуду и бесконечно малую длительность, положение которых определяется сдвигом по оси $a\Delta t = t$ при $\Delta t \to 0, a \to \infty$. Система функций $\{u_a(t)\}$ является полной ортогональной системой.

Из нее дискретизацией можно получить систему дискретных единичных функций $\{u_a(t)\}$, каждая из которых имеет вид единичного импульса бесконечно малой длительности и аналитически записывается в виде

$$u_a(t) = \begin{cases} 0 & \text{при } i \neq a, \\ 1 & \text{при } i = a. \end{cases}$$

Такая система определена на целочисленном интервале $[0, N)$.

Система $\{u_a(i)\}$ в такой форме является ненормированной, и ее норма (корень квадратный из мощности)

$$\|u_a\| = \frac{1}{\sqrt{N}}.$$

Эта система представляет собой полную СБФ, служащую для разложения дискретных сигналов произвольной формы.

Система дискретных единичных функций обладает тем свойством, что ее спектральный коэффициент с номером a совпадает со значением сигнала в точке $i = a$ его интервала определения, т. е. $c_a = x(a)$.

Подобным свойством обладает и непрерывная система $\{u_a(t)\}$. Это свойство единичной системы позволяет проиллюстрировать взаимосвязь между представлением сигнала в области аргументов и спектральной области. В соответствии с ним представление в области аргумента можно рассматривать как частный случай спектрального представления в единичном базисе. Это позволяет получать результаты в области аргументов, используя более общие результаты в спектральной области.

2.1.1. Системы ортогональных функций

Системы комплексных экспоненциальных функций. Полной ортогональной системой на интервале $[-\pi, \pi]$ или любом другом интервале длительностью 2π является система комплексных экспоненциальных функций $\{\exp(jk\xi)\}$. Это нормированная периодическая система с периодом 2π. Для нее характерно свойство мультипликативности, которое заключается в том, что произведение двух любых ее функций является также функцией этой системы:

$$\{\exp(jk\xi)\}\{\exp(jn\xi)\} = \{\exp(jl\xi)\},$$

где $l = k + m$.

Дискретный аналог этой системы – система дискретных комплексных экспоненциальных функций $\{\exp(j2\pi ki/N)\}$, обладающая свойствами полноты, нормированности, ортогональности и мультипликативности на интервале, содержащем N отсчетов. Зависимости ряда и коэффициентов Фурье при использовании в качестве базиса системы дискретных комплексных экспоненциальных функций называются дискретными преобразованиями Фурье [5]:

$$x[i] = \sum_{k=0}^{N-1} c_k \exp(j\,2\pi ki/N);$$

$$c_k = \frac{1}{N}\sum_{i=0}^{N-1} x[i]\exp(-j\,2\pi ki/N),$$

где $\{\exp(-j2\pi ki/N)\}$ – система комплексно-сопряженных экспоненциальных функций, определенных на интервале в N точках.

Спектр c_α в базисе $\{\exp(j2\pi ki/N)\}$ является комплексной функцией.

Полиномиальные базисные системы. К ним относят системы, построенные на основе ортогональных полиномов [6]. Рассмотрим две такие системы, определенные на конечных интервалах.

Полиномы Чебышева. На интервале [–1, 1] можно построить полную ортонормальную систему

$$\varphi_n(\xi) = 2^n(2\pi)^{-1/2}T_n(\xi), \quad n = 0, 1, 2, ...,$$

где $T_n(\xi)$ – полиномы Чебышева, задаваемые следующим образом:

$$T_0(\xi) = 1, T_n(\xi) = \frac{1}{2^{n-1}}\cos(n\arccos(\xi)), n \geq 1.$$

Полиномы Чебышева обладают тем важным свойством, что из всех полиномов n-й степени, имеющих коэффициент при ξ^n, равный единице, полином Чебышева $T_n(\xi)$ наименее отклоняется от нуля на интервале [–1, 1]. При $n \geq 3$ значение $T_n(\xi)$ можно вычислять по рекуррентной формуле

$$T_n(\xi) = \xi T_{n-1}(\xi) - 0{,}25T_{n-2}(\xi).$$

Полиномы Лежандра. Нормированные и ортогональные функции

$$\varphi_0(\xi) = \frac{1}{\sqrt{2}};\ \varphi_1(\xi) = \sqrt{\frac{3}{2}}\xi,; \varphi_2(\xi) = \sqrt{\frac{5}{2}}\left[\frac{3}{2}\xi^2 - \frac{1}{2}\right], ...; \varphi_n(\xi) = \sqrt{\frac{2n+1}{2}}P_n(\xi)$$

образуют полную систему базисных функций на отрезке $[-1, 1]$.

Здесь $\{P_n(\xi)\}$ – полиномы Лежандра, определяемые по формуле

$$P_n(\xi) = \frac{1}{2^n n!}\frac{d^n}{d\xi^n}(\xi^2 - 1)$$

или по рекуррентной зависимости

$$nP_n(\xi) = (2n - 1)\xi P_{n-1}(\xi) - (n - 1)P_{n-2}(\xi).$$

Непосредственная дискретизация непрерывных СБФ, построенных на основе ортогональных полиномов, образует системы дискретных функций с неравноотстоящими отсчетами. В этом смысле непрерывные полиномы не имеют решетчатых аналогов. Однако в классе полиномиальных функций можно построить решетчатые полиномы, используя непосредственное

представление их аргумента в виде дискретной переменной. Среди таких функций конечный интервал определения имеют функции дискретных систем Чебышева, Кравчука, Шарлье и Мейкснера.

Функции систем этого класса ортогональны на интервале $[0, N)$ и являются ненормированными. Для каждой системы известны рекуррентные соотношения, позволяющие строить аналитические описания функций при различных значениях номера функции α и числа отсчетов N. Данные аналитические описания можно представить в форме обобщенных степенных полиномов

$$pol_{\alpha}(z) = \sum_{k=0}^{\alpha} a_k z^k,$$

где a_k – коэффициенты, зависящие от конкретного типа системы.

От типа системы зависят и нормы базисных функций, поскольку эти системы не являются нормированными. Например, для системы Чебышева коэффициенты и норма записываются как [7]

$$a_k = \frac{(-1)^k (k+\alpha)!}{k!^2\,(\alpha-k)!\,(N-1)(N-2)\ldots(N-k)};$$

$$\|pol_{\alpha}(z)\| = \frac{(N+\alpha)(N+\alpha-1)\ldots N}{N(2\alpha+1)(N-1)(N-2)\ldots(N-\alpha)}.$$

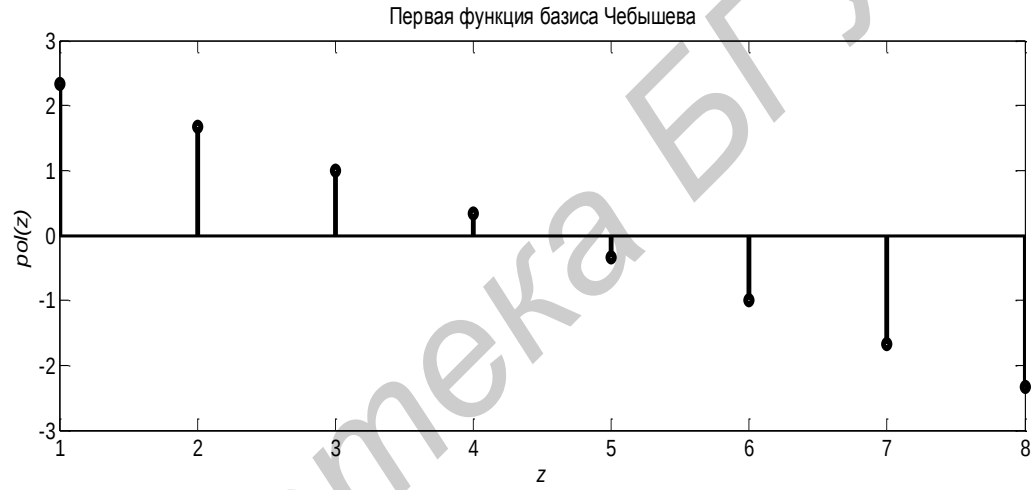
Проведя нормирование, можно построить нормированные системы дискретных полиномов. Для нормированного базиса Чебышева первые три его функции представляются следующим образом:

$$pol_0(z) = 1, \mathrm{pol}_1(z) = 3(N-1-2z)/(N+1),$$

$$pol_2(z) = \frac{5}{(N+2)(N+1)}[(N-1)(N-2) - 6z(N-2) + 6z(z-1)].$$

Для $N = 8$ первые три функции базиса Чебышева приведены на рис. 2.1.

а



б

в

Рис. 2.1. Функции Чебышева для $N = 8$:
а – $pol_0(z)$; б – $pol_1(z)$; в – $pol_2(z)$

2.2. Алгебраические модели представления сигналов

Линейные векторные пространства. Пусть L – множество объектов, которые условно назовем векторами и обозначим $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$,... . Введем на множестве векторов операцию сложения так, чтобы $(L, +)$ была бы коммутативной группой. Введем операцию умножения скаляра на вектор так, чтобы результат был вектором того же множества $L: \alpha\boldsymbol{A} = \boldsymbol{B} \in L$. Множество скаляров α, β, γ,… должно принадлежать числовому кольцу или полю. Если при этом выполняется набор аксиом дистрибутивности:

$$(\alpha + \beta)\boldsymbol{A} = \alpha\boldsymbol{A} + \beta\boldsymbol{A}, \alpha(\boldsymbol{A} + \boldsymbol{B}) = \alpha\boldsymbol{A} + \alpha\boldsymbol{B},$$

кроме того, $0\boldsymbol{A} = 0$, $1\boldsymbol{A} = \boldsymbol{A}$ для всех векторов L, то L является *линейным векторным пространством*. Первым классом объектов является множество векторов. Вторым классом объектов является множество скаляров.

Примеры

1. Возьмем плоскость с декартовой системой координат. Каждой точке плоскости можно сопоставить вектор, выходящий из начала координат и заканчивающийся в данной точке. Каждому вектору можно сопоставить пару чисел, соответствующих его проекциям на координатные оси, причем эти соответствия будут взаимно однозначными. Таким образом, задав пару чисел $(a,\ b)$, мы можем считать, что задали вектор на плоскости, выходящий из начала координат. Обозначим множество таких двумерных векторов через L. Введем на множестве векторов операцию сложения:

$$(a, b) + (c, d) = (a + c, b + d).$$

Легко убедиться в том, что множество векторов L с такой операцией сложения образует коммутативную группу. Введем операцию умножения скаляра на вектор: $\alpha(a,\ b) = (\alpha a,\ \alpha b)$. Аксиомы дистрибутивности выполняются, следовательно, множество таких векторов образует линейное двумерное векторное пространство.

Рассмотрим теперь тройки чисел вида (**a**, **b**, **c**) и назовем их векторами в трехмерном пространстве. Введем операцию сложения трехмерных векторов аналогично двумерным, т. е. компоненты трехмерного вектора будут равны сумме соответствующих компонентов слагаемых векторов. Введем операцию умножения скаляра на трехмерный вектор, при которой компоненты вектора умножаются на скаляр. В этом случае легко проверить, что все аксиомы дистрибутивности выполняются. Следовательно, мы построили трехмерное линейное векторное пространство.

Рассмотрим теперь объекты вида $\mathbf{a} = (a_1, a_2, a_3, \ldots, a_m)$, где все a_i принадлежат полю действительных чисел, а сами объекты назовем m-мерными

векторами. Аналогично рассмотренным случаям введем операции сложения m-мерных векторов и умножения скаляров на m-мерный вектор. Проверив, что такие векторы образуют коммутативную группу относительно операции сложения и выполняются аксиомы дистрибутивности, мы можем считать, что построили линейное m-мерное векторное пространство.

Если считать, что все компоненты m-мерного вектора $\mathbf{c} = (c_1, c_2, c_3, \ldots, c_m)$ есть комплексные числа, то, введя аналогичным образом операции сложения векторов и умножения скаляра на вектор и проверив выполнимость всех аксиом, получим линейное m-мерное комплекснозначное векторное пространство.

2. Рассмотрим всевозможные непрерывные функции с интегрируемым квадратом, заданные на интервале $[a, b]$. Введем операцию сложения двух функций $F_1(x) + F_2(x) = F_3(x)$ так, что значение функции F_3 в каждой точке интервала равно сумме значений функций F_1 и F_2 в этой же точке. Каждую функцию условно можно назвать вектором, тогда относительно операции сложения множество непрерывных функций на интервале $[a, b]$ образует коммутативную группу. Введя операцию масштабирования функции (умножения скаляра на функцию), мы можем убедиться в том, что все аксиомы дистрибутивности будут выполняться. Таким образом, мы получили линейное векторное пространство функций с интегрируемым квадратом, заданных на интервале $[a, b]$. Это пространство будет бесконечномерным (континуальным).

Необходимые определения [10, 14,15]:

1. *Алгебра.* $\boldsymbol{C}$-алгебра A определяется как кольцо в $\boldsymbol{C}$-векторном пространстве, для которого операции сложения в кольце и векторном пространстве совпадают. Дополнительно для $\alpha \in \boldsymbol{C}$, $g, h \in A$ должно выполняться условие $\alpha(gh) = (\alpha g)\, h = g(\alpha h)$.

2. *Модуль.* Пусть A будет $\boldsymbol{C}$-алгебра. Левый A-модуль определяется как $\boldsymbol{C}$-векторное пространство M, в котором разрешены операции

$$A \times M \to M, \qquad (a, m) \to am,$$

причем для $a, b, 1 \in A$ и $m, n \in$ M

$$a(m + n) = am + an; \;\; (a + b)m = am + bm; \;\; (ab)m = a(bm); \;\; 1m = m.$$

Модуль – абелева группа с кольцом операторов. Модуль является обобщением линейного векторного пространства над полем K для случая, когда K заменяется некоторым кольцом.

Примеры модулей:

а) любая абелева группа M является модулем над кольцом целых чисел $\mathbb{Z}$. Для $a \in \mathbb{Z}$ и $m \in \mathbb{Z}$ произведение am определяется как результат сложения ma раз;

б) в случае, когда A – поле, понятие унитарного A-модуля в точности эквивалентно понятию линейного векторного пространства над A.

Алгебраические модели сигналов

Производящая функция дискретного сигнала. Пусть дискретный сигнал имеет вид произвольной (бесконечной) последовательности $a_0, a_1, a_2, \dots$.

Производящей функцией (производящим рядом) для этой последовательности будем называть выражение вида

$$f^*(z) = a_0 + a_1 z + a_2 z^2 + \dots,$$

или в сокращенной записи

$$f^*(z) = \sum_{\substack{n=0 \\ n \in \mathbb{N}}}^{\infty} a_n z^n.$$

Соответствие между последовательностью a_n и $f^*(z)$ взаимно однозначно. Функция $f^*(z)$ называется производящей функцией последовательности a_n.

Последовательность a_n представляет собой функцию от n, $n = 0, 1, 2, \dots$. Обозначим ее через $f(n)$ и будем говорить, что между совокупностями $f(n)$ и $f^*(z)$ существует взаимно однозначное соответствие.

Пример. Экспоненциальная производящая функция имеет вид

$$f^e(z) = a_0 + a_1 z + a_2 \frac{z^2}{2!} + \dots + a_n \frac{z^n}{n!} + \dots.$$

Обобщенная производящая функция определяется как

$$\varphi^*(z) = a_0 \varphi_0(z) + a_1 \varphi_1(z) + \dots + a_n \varphi_n(z) + \dots.$$

Соответствие $a_n \leftrightarrow \varphi^*(z)$ должно быть взаимно однозначным. Это приводит к независимости $\{\varphi_i(z)\}$.

Полиномиальная модель. В качестве моделей сигналов и фильтров преобразований можно использовать полиномы $S(z^{-1})$ и $H(z^{-1})$ степени $N - 1$, а саму операцию фильтрации рассматривать как результат произведения полиномов. В таком представлении результат линейной фильтрации имеет более высокую степень полинома, чем полиномы сигналов и фильтров, и фильтрация как алгебраическая форма не является замкнутой.

Пространство сигналов можно замкнуть, используя операцию умножения по модулю $(z^{-N} - 1)$:

$$H(z^{-1})S(z^{-1})\bmod(z^{-N}-1).$$

В этом случае сигналы и фильтры располагаются в пространстве полиномов в $\mathbb{C}[z^{-1}]$ по модулю $(z^{-N}-1)$, которое обозначается как $\mathbb{C}[z^{-1}]/(z^{-N}-1)$ и называется полиномиальной алгеброй.

Модель сигнала определяется через тройку операторов (*A,* Ф, *M*); где *A* – алгебра фильтров; *M* – модуль сигналов, Ф – обобщенное биективное линейное отображение из векторного пространства (т. е. отсчетов сигнала) в модуль *M*.

Операторы сдвига. Ключевую роль в алгебраической модели играет *оператор сдвига*. Вид сдвига тесно связан с моделью сигнала. Модели конечных и бесконечных интервалов используют разные операторы сдвига. Другие операторы сдвига естественно ведут к новым моделям сигналов и новым спектральным преобразованиям.

Модель временного сдвига (рис. 2.2, а). Модель временного сдвига можно определить как $q\langle\rangle t_n = t_{n+1}$, где q – оператор сдвига; $\langle\rangle$ – операция сдвига; t_n – точка дискретного времени. Модель ассоциируется с бесконечным или конечным *z*-преобразованием.

Модель пространственного сдвига (рис. 2.2, б). Пространственный сдвиг в операторной форме можно записать как $q\langle\rangle t_n = \frac{1}{2}(t_{n-1} + t_{n+1})$. С такой моделью ассоциируется дискретное косинусное преобразование (ДКП).

Покажем, что оператор *k*-го пространственного сдвига q_k может быть задан полиномом Чебышева $T_k(q)$ $q_k = T_k(q)$.

Действительно, можно записать следующее соотношение

$$q_{k+1}\langle\rangle t_n = \frac{t_{n+k+1} + t_{n-k-1}}{2} =$$

$$= (t_{n+k+1} + t_{n+k-1} + t_{n-k+1} + t_{n-k-1})/2 -$$

$$-\frac{t_{n+k-1} + t_{n-k+1}}{2} = 2q\langle\rangle(t_{n+k} + t_{n-k})/2 - (t_{n+k-1} + t_{n-k+1})/2 =$$

$$= (2qq_k - q_k - 1)\langle\rangle t_n.$$

Напомним, что последовательность полиномов $T = (T_n \mid n \in \mathbb{Z})$, удовлетворяющая рекуррентному соотношению $T_{n+1}(v) = 2vT_n(v) - T_{n-1}(v)$, называется последовательностью полиномов Чебышева.

Используя рекуррентное свойство полиномов Чебышева, получаем

$$q_{k+1} = T_{k+1}(q).$$

Инвариантность к сдвигу. Инвариантность к сдвигу является ключевой концепцией обработки сигналов. В алгебраической теории это свойство имеет простую форму представления. А именно, если q – оператор сдвига, s – сигнал, а h – фильтр, то h обладает свойством инвариантности к сдвигу, если для всех сигналов $h(qs) = q(hs)$, что эквивалентно $hq = qh$. Требования инвариантности к сдвигу для всех фильтров запишутся как $qh = hq$ для всех $h \in A$.

Так как сдвиг q формируется A, то необходимо, чтобы A была коммутативна. И наоборот, если A представляет собой коммутативную алгебру, то все фильтры $h \in A$ обладают свойством инвариантности к сдвигу.

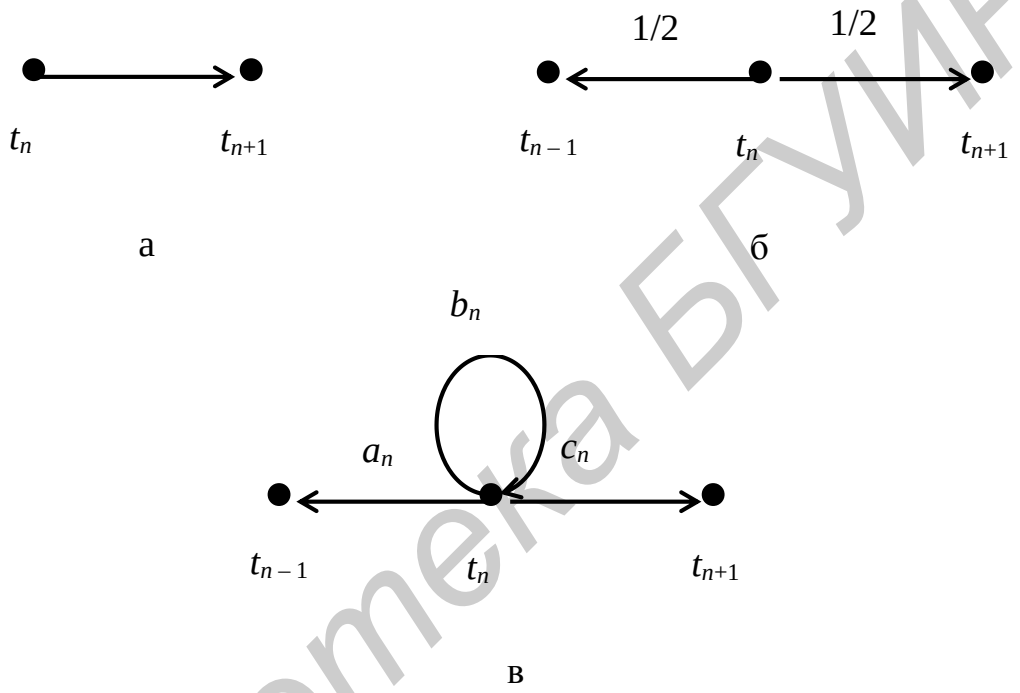


Рис. 2.2. Примеры возможных сдвигов в алгебраической модели:
а – временной; б – пространственный; в – цепь сдвиговых состояний

Зададимся вопросом, какие алгебры обладают свойством инвариантности к сдвигу? В случае одного сдвига это алгебра, образованная одним элементом v. В бесконечномерном случае мы получаем алгебры ряда переменной v или полиномов произвольной степени от v. Если обработка ведется на конечных интервалах, то имеем алгебры полиномов вида $A = \boldsymbol{C}[v]/p(v)$, где p – полином степени n; $\boldsymbol{C}[v]/p(v)$ – множество всех полиномов степени, меньшей чем n с операциями сложения и умножения по модулю $p(v)$. Как векторное пространство алгебра A имеет размерность n.

Пример. Обозначим через $\mathbb{C}_N$ множество комплекснозначных периодических функций целочисленного аргумента $x = x(j), j \in \mathbb{Z}$. Элементы этого множества будем называть сигналами. В $\mathbb{C}_N$ обычным способом вводятся

операции умножения на комплексное число, сложения и умножения сигналов. А именно

$$y = cx \Leftrightarrow y(j) = cx(j), j \in \mathbb{Z};\ y = x_1 + x_2 \Leftrightarrow y(j) = x_1(j) + x_2(j), j \in \mathbb{Z};$$

$$y = x_1 x_2 \Leftrightarrow y(j) = x_1(j)x_2(j), j \in \mathbb{Z}.$$

В результате $\mathbb{C}_N$ становится коммутативной алгеброй с единицей. Единицей является сигнал **1**, все отсчеты которого равны единице.

Обратный к x сигнал x^{-1} определяется из условия $x\,x^{-1} = \mathbf{1}$. Очевидно, что сигнал x обратим тогда и только тогда, когда все его отсчеты $x(j)$ отличны от нуля. В этом случае

$$x^{-1}(j) = [x(j)]^{-1}, j \in \mathbb{Z}.$$

Модель сигнала на конечном интервале может быть построена с помощью алгебры $A = M = \mathbb{C}[v]/(v^N - 1)$, где $v = z^{-1}$, и отображения

$$\Phi: \mathbf{s} \to \sum_{0 \le n < N} s_n v^n \in M,$$

где сигнал $\mathbf{s} = (s_0, \ldots s_{N-1}) \in V = \mathbb{C}^n$ допускает комплексные значения.

Пространственная модель сигнала может быть построена с помощью полиномиальной алгебры вида $A = M = \mathbb{C}[v]/T_N(v)$ и отображения

$$\Phi: \mathbf{s} \to \sum_{0 \le n < N} s_n T_n(v) \in M,$$

где T_n – полиномы Чебышева первого рода. Соответствующее Фурье-преобразование для этой модели носит название *дискретного косинусного преобразования*.

2.3. Модели сигналов на основе теории решеток

Решетка представляет собой регулярный пространственный массив точек [16–19]

$$L = \{\boldsymbol{B}\mathbf{x} | \mathbf{x} \in \mathbb{Z}^n\},$$

где $\boldsymbol{B} \in \mathbb{R}^{n \times n}$ – базис пространства.

В теории связи решетки используются для описания сигналов с квадратурной амплитудной модуляцией представления линейной модели канала связи.

При этом решетка комплексных значений

$$L = \{\boldsymbol{B}\mathbf{x} | \mathbf{x} \in \mathbb{Z}^n + j\mathbb{Z}^n\}, \boldsymbol{B} \in \mathbb{C}^{n\times n}$$

может быть преобразована в решетку действительных значений.

На приемной стороне модель решетки используется для построения алгоритмов декодирования.

Групповые решетки. Напомним некоторые результаты из теории групп. Элементы $\alpha_1, \ldots, \alpha_n$ аддитивной абелевой группы M называются системой образующих группы, рассматриваемой как $\mathbb{Z}$-модуль, если любой элемент в M можно представить в виде

$$\alpha = c_1\alpha_1 + \cdots + c_n\alpha_n, \text{ где } c_i \in \mathbb{Z}.$$

Система образующих называется базисом, если такое представление единственно.

Элемент α аддитивной абелевой группы M называется элементом конечного порядка, если $c\alpha = 0$ при некотором $c_i \in \mathbb{Z}$.

Если абелева группа без элементов конечного порядка имеет конечную систему образующих, то она имеет и базис. Число элементов базиса является инвариантом группы.

Определения:

1. Решеткой называется конечно порожденная подгруппа группы $\mathbb{R}^n$. Если ранг группы равен n, то решетка называется полной, в противном случае – неполной. Базис группы называется в этом случае базисом решетки. В векторном пространстве $\mathbb{R}^n$ определены скалярное произведение и норма.

2. Подгруппа G группы $\mathbb{R}^n$ называется дискретной, если в шаре

$$U(r) = \{x \in \mathbb{R}^n | \, \|x\| < r\}$$

радиусом r имеется только конечное число элементов группы G. Решетка является дискретной группой.

3. Пусть $\mathbf{b}_1, \ldots, \mathbf{b}_k$ – базис решетки. Основным параллелепипедом решетки называется множество

$$T = \{x \in \mathbb{R}^n | \, x = c_1\mathbf{b}_1 + \cdots + c_k\mathbf{b}_k, \ 0 \le c_i < 1\}.$$

Объем основного параллелепипеда называется детерминантом решетки. Детерминант решетки не зависит от базиса.

Если T – основной параллелепипед полной решетки M, то имеется разбиение

$$\mathbb{R}^n = \bigcup_{z\in M} z + T.$$

Причем $z + T \cap w + T = \emptyset$ при $z \neq w$.

Подгруппа $M \subset \mathbb{R}^n$, множество элементов которой дискретно, является решеткой. Пусть $\mathbf{b}_1, \ldots, \mathbf{b}_m$ – базис решетки M. Тогда ее детерминант равен квадратному корню из определителя

$$\begin{vmatrix} (\mathbf{b}_1, \mathbf{b}_1) & \ldots & (\mathbf{b}_1, \mathbf{b}_m) \\ \cdots & \cdots & \cdots \\ (\mathbf{b}_m, \mathbf{b}_1) & \cdots & (\mathbf{b}_m, \mathbf{b}_m) \end{vmatrix},$$

где $(\mathbf{b}_l, \mathbf{b}_m)$ – скалярное произведение базисных векторов $\mathbf{b}_l$ и $\mathbf{b}_m$.

Решетка M в линейном пространстве L полна тогда и только тогда, когда в L существует ограниченное множество U, сдвиги которого на векторы из M полностью заполняют все пространство L.

2.4. Модели сигналов на графах

Пусть информация имеет вид множества из N элементов. Сетевая информация представляется в виде направленного графа $G = (V, A)$, где $V = \{v_0, \ldots, v_{N-1}\}$ – множество узлов; A – весовая матрица смежности графа.

Каждому информационному элементу поставим в соответствие узел v_n и положим, что каждый вес $A_{n,m}$ ребра графа, соединяющего узлы v_m и v_n, отражает степень отношений, существующих между этими элементами. Значение $A_{n,m}$ может принимать произвольные действительные или комплексные значения (например, если элементы имеют отрицательную корреляцию). Множество индексов узлов, соединенных с v_n, обозначим как $\mathcal{N}_n = \{m | A_{n,m} \neq 0\}$ и будем называть множеством ближайших соседей узла v_n.

Без ограничения общности положим, что информационные элементы являются комплексными скалярами. Тогда сигнальный граф можно представить в виде отображения из узлов множества V в множество комплексных чисел $\mathbb{C}$

$$\mathbf{s}: V \to \mathbb{C},$$
$$v_n \longmapsto s_n.$$

Существует изоморфизм между сигналом и вектором из N комплексных элементов. Для упрощения записи сигнальный граф будем ассоциировать с вектором $\mathbf{s} = [s_0, s_1, \ldots, s_{N-1}]^T$, не забывая, что каждый элемент s_n является индексом узла v_n графа $G = (V, A)$.

Пространство $\boldsymbol{S}$ сигнального графа идентично $\mathbb{C}^N$.

Примеры графов показаны на рис. 2.3.

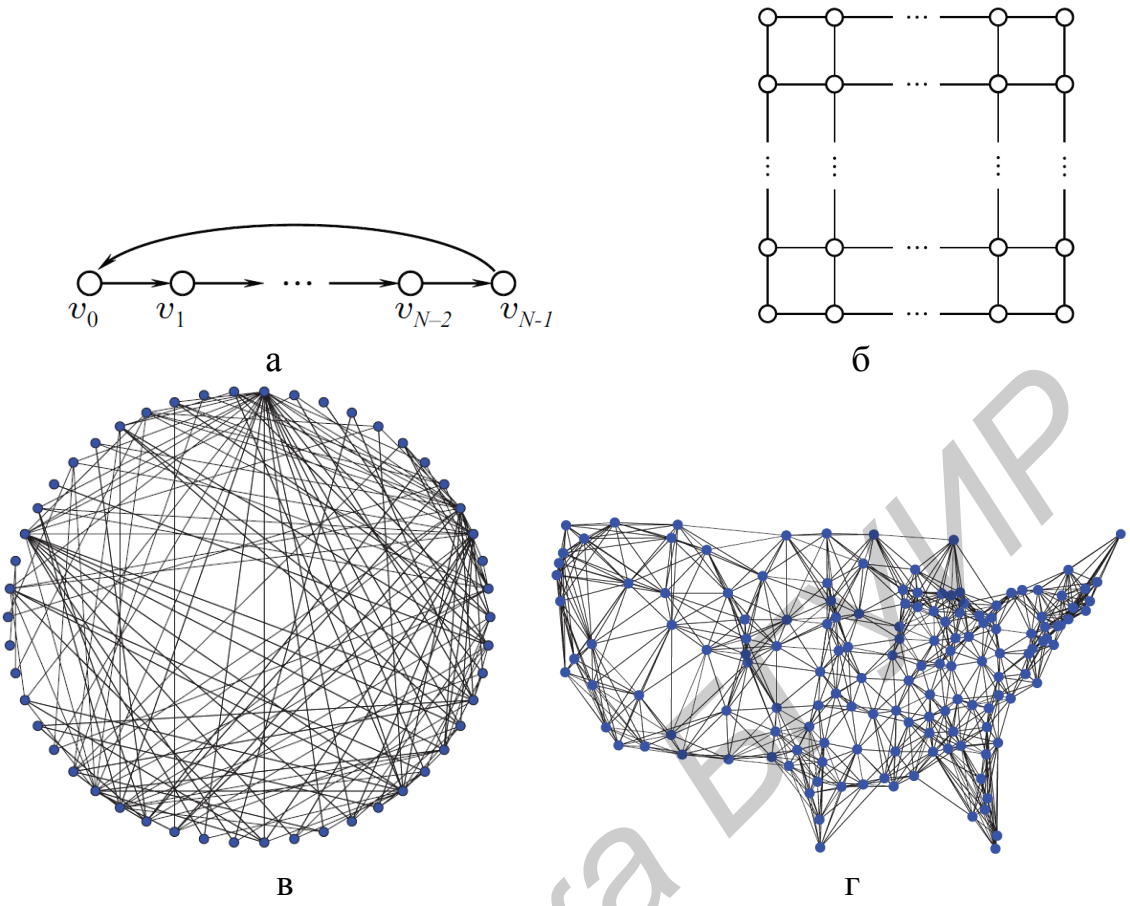


Рис. 2.3. Сигнальные графы:
а – временного ряда; б – цифрового изображения; в – электронного документа; г – сенсорного поля

Ограничимся случаем линейных, инвариантных к сдвигу фильтров, аналогами которых являются линейные, инвариантные по времени фильтры обработки сигналов для временных рядов.

Операция сдвига на графе. В цифровой обработке сигналов фильтр, выполняющий задержку сигнала, имеет передаточную функцию z^{-1}. Сигнал задерживается на один отсчет $\tilde{s}[n] = s[n - 1 \bmod N]$.

Матрица смежности A графа, соответствующего периодическому временному ряду, имеет форму матрицы-циркулянта $A = C_N$ размером $N \times N$ с элементами вида

$$A_{n,m} = \begin{cases} 1, & \text{если } n - m = 1 \bmod N, \\ 0, & \text{в остальных случаях.} \end{cases}$$

Тогда можно записать операцию сдвига по времени как

$$\tilde{\mathbf{s}} = C_N \mathbf{s} = A\mathbf{s}.$$

Операцию сдвига можно обобщить на произвольный граф $G = (V, A)$. Данная операция должна изменять координаты отсчета $s[n]$ узла v_n относительно взвешенной линейной комбинации отсчетов сигнала ближайших соседей

$$\tilde{s}[n] = \sum_{m=0}^{N-1} A_{n,m} s[m] = \sum_{m \in \mathcal{N}_n} A_{m,n} s[m].$$

Операция сдвига требует задания неких граничных условий, которые определяются структурой графа $G = (V, A)$.

Рассмотрим граф $G = (V, A)$, для которого минимальный $m_A(x)$ и характеристический $p_A(x)$ полиномы матрицы смежности графа ко-инцидентны: $p_A(x) = m_A(x)$.

Отображение $A \to x$ матрицы смежности A в x может быть представлено как результат действия фильтра $H = h(A)$ в пространстве $\mathcal{F}$ с полиномом $h(x)$. Фильтровое пространство $\mathcal{F}$ при этом можно рассматривать как полиномиальную алгебру

$$\mathcal{A} = \mathbb{C}[x]/m_A(x).$$

Пространство $\mathcal{F}$ – пространство полиномов степени меньше чем $\deg\{m_A(x)\}$ с комплексными полиномиальными коэффициентами, замкнутое относительно операций сложения и умножения по модулю полинома $m_A(x)$.

Отображения $\mathcal{F} \to \mathcal{A}$ и $h(A) \to h(x)$ являются изоморфизмами $\mathbb{C}$-алгебр, $\mathcal{F} \cong \mathcal{A}$ [15]. Определим такое преобразование как z-преобразование фильтров на графе $G = (V, A)$.

Сигнальное пространство $\mathcal{S}$ является векторным пространством, которое также замкнуто относительно операции фильтрации на графе в $\mathcal{F}$: для любого сигнала $s \in \mathcal{S}$ и фильтра $h(A)$ выход последнего представляет собой сигнал, который принадлежит пространству $\mathcal{F}$. Пространство $\mathcal{S}$ можно рассматривать как модуль в $\mathcal{F}$. Производящая функция z-преобразования сигналов на графе может быть определена как изоморфизм $\mathcal{S}$ в $\mathcal{M}$-модуль, который имеет вид

$$\mathcal{M} = \mathbb{C}[x]/p_A(x) = \left\{ s(x) = \sum_{i=0}^{N-1} s_i b_i(x) \right\}$$

относительно отображения

$$\mathbf{s} = [s_0, s_1, \dots, s_{N-1}]^T \mapsto s(x) = \sum_{i=0}^{N-1} s_i b_i(x).$$

Полиномы $b_0(x), b_1(x), \dots, b_{N-1}(x)$ являются линейно независимыми полиномами степени не ниже $(N-1)$. Если записать вектор

$$\mathbf{b}(x) = [\, b_0(x), b_1(x), \dots, b_{N-1}(x)]^T,$$

то производные полиномов удовлетворяют соотношению

$$\mathbf{b}^{(r)}(x) = \left[\, {b_0}^{(r)}(\lambda_m), {b_1}^{(r)}(\lambda_m), \dots, {b_{N-1}}^{(r)}(\lambda_m)\right]^T, = r!\, \tilde{\mathbf{v}}_{m,0,r}.$$

Для $0 \le r < R_{m,0}$ и $0 \le m < M$, где λ_m и $\tilde{\mathbf{v}}_{m,0,r}$ – обобщенные собственные векторы матрицы A^T, ${b_i}^{(r)}(\lambda_m)$ – r-я производная $b_i(x)$, взятая при $x = \lambda_m$.

Задачи теории решеток [18–19]:

1. По базису решетки найти кратчайший ненулевой вектор (Shortest Vector Problem, SVP; поиск кратчайшего вектора) (рис. 2.4).

2. По базису решетки $B \in \mathbb{Z}^{m\times n}$ и вещественному $\gamma > 0$ найти ненулевой вектор $\mathbf{b} \in B\mathbb{Z}^n \backslash \{0\}$: $\|\mathbf{b}\|_p \le \gamma\lambda_1^p(L)$ с p-нормой (γ-approximation Shortest Vector Problem, SVP_γ^p; приближенный поиск кратчайшего вектора) (рис. 2.5).

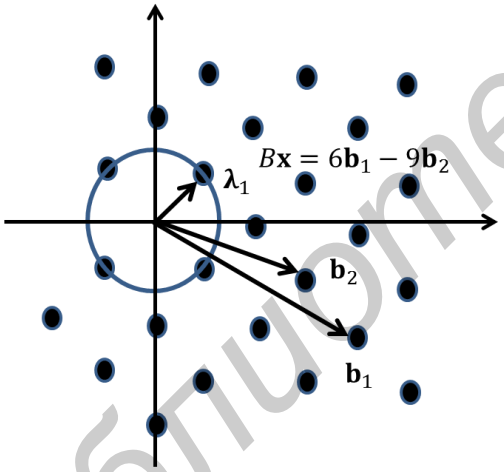


Рис. 2.4. Пример SVP-задачи в $\mathbb{R}^2$

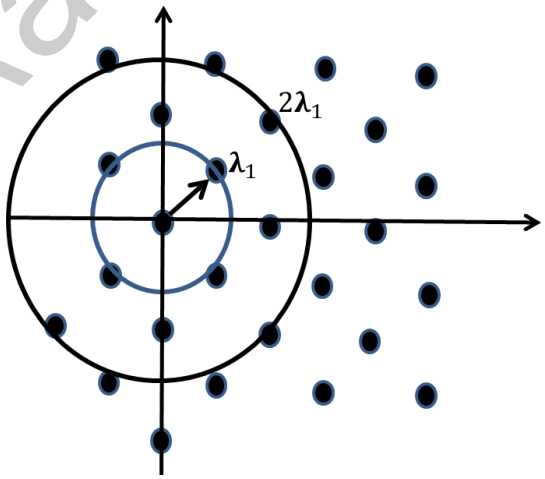


Рис. 2.5. Пример SVP_γ -задачи в $\mathbb{R}^2$

3. По базису решетки B и заданному вектору $\bar{\jmath} \notin L(B)$ найти ближайший вектор $\mathbf{b} \in L(B)$ (Closes Vector Problem, CVP; поиск ближайшего вектора) (рис. 2.6).

4. По базису решетки $B \in \mathbb{Z}^{m\times n}$, вещественному $\gamma > 0$ и заданному вектору $\boldsymbol{\gamma} \in L\mathbb{R}^n$ найти ненулевой вектор $\mathbf{b} \in B\mathbb{Z}^n$: $\|\boldsymbol{\gamma} - \mathbf{b}\|_p \le \gamma\lambda_1^p(L)$ с нормой (γ-approximate Closes Vector Problem, CVP_γ^p).

5. Пусть дана n-мерная решетка L. Найти линейно независимые векторы $\mathbf{b}_1, \dots, \mathbf{b}_n \in L$, для которых $\max_i \|\mathbf{b}_i\|_p \le \gamma\lambda_n^p(L)$, где $\lambda_n^p(L)$ – n-й последователь-

ный минимум в решетке с *p*-нормой (γ-approximate Shortest Independent Vector Problem, $\mathrm{SIVP}_\gamma^p(n,\gamma)$; приближенный поиск кратчайших линейно независимых векторов) (рис. 2.7).

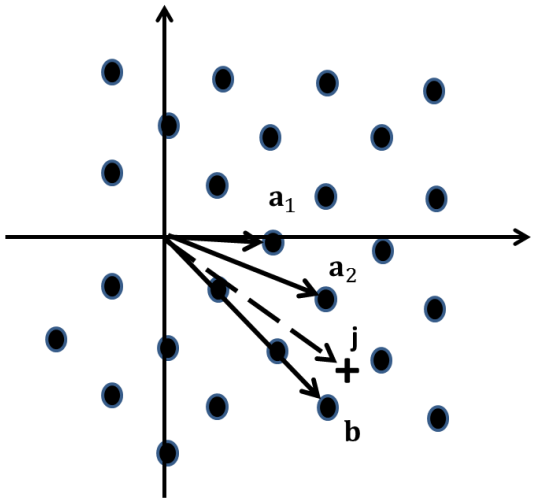


Рис. 2.6. Пример CVP-задачи в $\mathbb{R}^2$

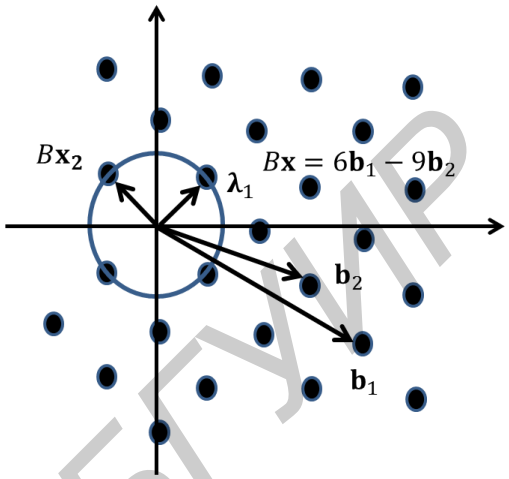


Рис. 2.7. Пример SIVP-задачи в $\mathbb{R}^2$

6. Пусть дана *n*-мерная решетка L. Найти вектор

$$\mathbf{u} \in L\backslash\{0\}\colon \|\mathbf{u}\|_p \le \gamma\lambda_1^p(L),$$

где $\lambda_1^p(L)$ – длина кратчайшего вектора в решетке с *p*-нормой и $\mathbf{u}$ – кратчайший γ-уникальный вектор, т. е.

$$\forall w \in L\colon\ \lambda_1^p(L) \le \|\mathbf{w}\|_p \le \gamma\lambda_1^p(L),$$

где $\mathbf{w} = z\mathbf{u}$ для некоторых $z \in \mathbb{Z}$ (γ-approximate Unique Shortest Vector Problem, $u\mathrm{SVP}_\gamma^p(n,\gamma)$ – поиск уникального кратчайшего вектора) (рис. 2.8).

7. Пусть дан базис B q-нарной (модулярной) *n*-мерной решетки $L_q^{m\times n}$, для которой принадлежность вектора к решетке L определяется как

$$L(B) = \{B^T\mathbf{s}\ \mathrm{mod} q \subseteq \mathbb{Z}^m, \mathbf{s} \in \mathbb{Z}^n\},$$

где q – простое число.

На решетке равномерно распределен шум $\mathbf{e}$ (обычно с нулевым математическим ожиданием и дисперсией $\sqrt{q}$); $\mathbf{s} \in \mathbb{Z}_q^n$ – некоторый исходный вектор без шума; известно значение $(B\mathbf{s}+\mathbf{e})$. Найти исходную точку в решетке (исключить шум) по некоторому множеству известных $(B\mathbf{s}+\mathbf{e})$. Задача

обучения с ошибками (Learning With Errors, LWE) является обобщением задачи обучения контролю целостности (четности) данных с шумами (рис. 2.9).

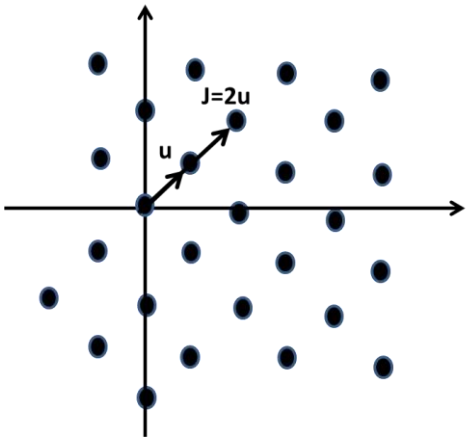


Рис. 2.8. Пример uSVP-задачи

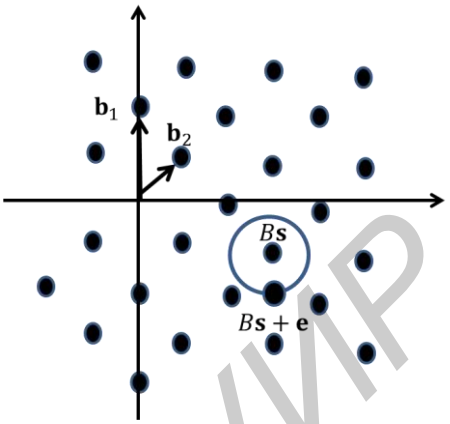


Рис. 2.9. Пример LWE-задачи

8. Пусть дана m-мерная модулярная решетка

$$L_q^{\perp}(B) = \{\mathbf{x} \in \mathbb{Z}^m : B\mathbf{x} \equiv \bar{0} \in \mathbb{Z}^n \bmod q\},$$

образованная базисом $B \in \mathbb{Z}_q^{n\times m}$, взятым случайно из равномерного распределения над $\mathbb{Z}_q^{n\times m}$.

Найти кратчайший вектор $\mathbf{v} \in L_q^{\perp}(A)$: $\|\mathbf{v}\|_p \leq \beta$ с p-нормой (Short Integer Solution Problem, $\mathrm{SIS}_{\beta}^{p}(n,m)$; задача поиска вектора по норме в модулярной решетке).

9. Пусть дана m-мерная модулярная решетка

$$L_q^{\perp}(B) = \{\mathbf{x} \in \mathbb{Z}^m : B\mathbf{x} \equiv \mathbf{0} \in \mathbb{Z}^n \bmod q\},$$

образованная базисом $B \in \mathbb{Z}_q^{n\times m}$ и вектором $\mathbf{y} \in \mathbb{Z}^n$, взятыми случайно из равномерного распределения над $\mathbb{Z}_q^{n\times m}$, $\mathbb{Z}^n$ соответственно. Найти вектор $\mathbf{v} \in \{\mathbf{x} \in \mathbb{Z}^m : A\mathbf{x} \equiv \mathbf{y} \bmod q\}$: $\|\mathbf{v}\|_p \leq \beta$ (Independent Short Integer Solution Problem, $\mathrm{ISIS}_{\beta}^{p}(n,m)$; гетерогенная задача поиска вектора по норме в модулярной решетке).

10. По базису решетки $B \in \mathbb{Z}^{m\times n}$ и вещественному $\gamma \geq 1$ найти ненулевой вектор $\mathbf{b} \in B\mathbb{Z}^n \backslash \{0\}$: $\|\mathbf{b}\|_p \leq \gamma \det\big(L(B)\big)^{1/n}$ с p-нормой (γ-approximate Hermite Shortest Vector Problem, $\mathrm{hermitSVP}_{\gamma}^{p}$; задача приближенного поиска кратчайшего вектора по Эрмиту).

11. Пусть дана n-мерная решетка $L^n \subseteq \mathbb{R}^k$. Найти базис $B: \forall B' \in \left\{B \in \mathbb{Q}^{n\times k}: L = L(B), \max_i\{\|\mathbf{b}_i\|_p\}\right\} \leq \gamma \max_i\{\|\mathbf{b'}_i\|_p\}$ (γ-approximate Shortest Basis Problem, $\mathrm{SBP}_\gamma^p(n)$; приближенный поиск кратчайшего базиса).

12. Пусть дана n-мерная решетка L^n и задана некоторая p-норма. Найти длину $l^{(p)} \in \mathbb{R}: l^{(p)} \leq \lambda_1^{(p)}(L) \leq \gamma l^{(p)}$, где $\lambda_1^{(p)}(L)$ – длина кратчайшего вектора, или первый последовательный минимум в решетке L (γ-approximate Shortest Length Problem, $\mathrm{SLP}_\gamma^p(n)$; задача приближенного поиска длины кратчайшего вектора в решетке).

13. По базису решетки $B \in \mathbb{Z}^{m\times n}, m \geq n$ и радиусу $r \in \mathbb{R}$ ответить на вопрос: «да» – если все точки решетки можно покрыть радиусом r, $r \geq \rho\big(L(B)\big)$; «нет» – если $\gamma r < \rho\big(L(B)\big)$ (γ-approximate Covering Radius Problem, $\mathrm{CRP}_\gamma^\rho(n,r)$; приближенная задача покрытия решетки радиусами).

14. По базису решетки $B \in \mathbb{Z}^{m\times n}$, вещественному $\alpha \geq 1$ и вектору $\mathbf{u}: \exists \mathbf{t} \in L(B), \|\mathbf{u}-\mathbf{t}\|_p < \alpha\lambda_1^p(L)$ найти вектор $\mathbf{v} \in L(B): [\![\mathbf{u}-\mathbf{v}]\!]_p = \min$ (α-approximate Bounded Distance Decoding, $\mathrm{BDD}_\alpha^p(u)$; приближенная задача о декодировании с ограниченным расстоянием).

15. По базису $B \in \mathbb{Z}^{m\times n}$, вектору решетки $\mathbf{t} \in B\mathbb{Z}^n$ и положительным действительным числам $d, \gamma > 0$ ответить на вопрос:

– «да» – если $\min\{\|\mathbf{t}-\mathbf{v}\|_p: \mathbf{v} \in B\mathbb{Z}^n\} \leq d$;

– «нет» – если $\min\{\|\mathbf{t}-\mathbf{v}\|_p: \mathbf{v} \in B\mathbb{Z}^n\} > \gamma d$ (Decisional Closest Vector Problem, GapCVP_γ^p; булева задача поиска вектора, близкого к вектору в решетке).

16. Пусть дан идеал $I \in \mathbb{Z}[x]/f(x)$. Найти многочлен $g(x) \in I \setminus \{0\}: \|g \bmod f(x)\|_p \leq \gamma\lambda_1^p(I)$ (Approximate Ideal Shortest Vector Problem/Shortest Polynomial Problem, $\mathrm{IdealSVP}_\gamma^p(f)$ приближенная задача поиска кратчайшего вектора в идеальной решетке/задача поиска кратчайшего полинома).

17. Пусть дано m многочленов $g_1(x), \dots, g_m(x)$, выбранных случайно из равномерного распределения, заданного на $\mathbb{Z}_q[x]/f(x)$, и n – степень многочлена $f(x)$. Найти целые $e_1, \dots, e_m \in \mathbb{Z}[x]: \sum_{i\leq m} e_i g_i = 0 \bmod q, \|\mathbf{e}\|_p \leq \beta$, где вектор $\mathbf{e}$ получается путем конкатенации (объединения) коэффициентов при всех e_i; (Ideal Small Integer Solution Problem, $\mathrm{Ideal\,SIS}_\beta^p(g,n,m)$; задача поиска вектора по норме в идеальной решетке).

2.5. Модели временных рядов и массивов

Одними из наиболее часто используемых моделей обрабатываемых сигналов являются временные ряды и двумерные массивы данных [20].

Под вещественнозначным (комплекснозначным) одномерным временным рядом длиной $N \geq 2$ понимается упорядоченная последовательность вещественных (комплексных) чисел $F = (f_0, \dots, f_{N-1})$, $f_n \in \mathbb{R}$ ($f_n \in \mathbb{C}$).

Чаще всего временной ряд является результатом измерения некоторого показателя в равноотстоящие моменты времени с шагом $\Delta > 0$, т. е. $f_n = f(\Delta n)$, где $f(t)$ – некоторая функция, заданная при $t > 0$; однако переменная t не обязательно имеет смысл времени и f может быть функцией от другого физического параметра, например пространственной координаты. Последовательности $F = (f_0, \ldots, f_{N-1})$, $f_n \in \mathcal{K}$, где $\mathcal{K}$ – конечное поле, описывающие изменение категориальных показателей, будем также называть временными рядами (над конечным полем). Мы будем считать временной ряд вектором-строкой, где это необходимо.

Под двумерным массивом (вещественнозначным или комлекснозначным) понимается таблица $f_{m,n} = f(\Delta_x m, \Delta_y n)$:

$$\mathbf{F} = \left(f_{m,n}\right)_{m,n=0}^{N_x-1,N_y-1}, \left(f_{m,n} \in \mathbb{R} \text{ или } f_{m,n} \in \mathbb{C}\right)$$

значений некоторой функции двух переменных $f(x, y), x, y > 0$ на дискретной прямоугольной сетке с шагом $x > 0$ и $y > 0$.

Данные могут иметь разную природу: цифровые изображения, измерения некоторого показателя в пространстве; в качестве одной из координат может выступать время, в этом случае массив можно рассматривать как совокупность рядов или как многомерный временной ряд. Двумерный массив везде далее мы считаем $N_x \times N_y$ матрицей.

Будем говорить, что ряд F (массив $\mathbf{F}$) является суммой аддитивных составляющих $F^{(1)}, \ldots, F^{(r)}$ ($\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(r)}$ соответственно), если он является их суммой как векторов (или матриц): $F = F^{(1)} + \ldots + F^{(r)}$ или $\mathbf{F} = \mathbf{F}^{(1)} + \ldots + \mathbf{F}^{(r)}$, т. е. в смысле покомпонентного сложения.

Во многих инфокоммуникационных приложениях сложилось представление о возможности описания природных процессов в виде суммы $f(t) = f^{(T)}(t) + f^{P}(t) + f^{(\varepsilon)}(t)$, где $f^{(T)}(t)$ – медленно меняющаяся составляющая, называемая трендом, которую часто описывают некоторой параметрической составляющей, например полиномами невысоких порядков или функциями с ограничениями на их гладкость; $f^{(P)}(t)$ – периодическая составляющая; $f^{(\varepsilon)}(t)$ – шумовая составляющая, которая обычно предполагается реализацией некоторого случайного процесса.

2.5.1. Модели пакетного трафика

Под трафиком сети с коммутацией пакетов понимают совокупность данных, передаваемых по сети. Анализ трафика – сбор и последующая оценка разнообразной статистики (скорость, объемы переданных данных, используемые протоколы, адреса узлов сети, отправляющих и/или принимающих трафик и т. д.) в действующей сети.

Модели трафика условно можно разделить на две группы: традиционные и нетрадиционные [20–26]. К традиционным моделям относят модели, основанные на состоянии и на временных рядах, к нетрадиционным – модели, основанные на данных наблюдения.

Вейвлет-модели (Wavelet Models) строятся на основе обратного дискретного вейвлет-преобразования, суть которого состоит в формировании с помощью масштабных и вейвлет-коэффициентов дискретного временного ряда, используя функции детализации различного масштаба на основе прототипа полосовой вейвлет-функции и низкочастотной скейлинг-функции. Вейвлет-модели могут иметь различное количество параметров (три и более) и достаточно эффективны для моделирования самоподобного трафика.

Мультифрактальные модели (Multifractional – MF) удачно воспроизводят трафик, агрегированный от нескольких существенно отличающихся источников. Мультифрактальность трафика проявляется в изменении статистических свойств реализации трафика при изменении масштаба агрегирования. Для описания таких свойств вводятся дополнительные масштабная функция и моментный коэффициент.

Авторегрессионные модели (Autoregressive Models – AR) широко применяются для моделирования и предсказания благодаря свойству длительной памяти самоподобных процессов. В этих моделях текущее значение генерируемой величины рассчитывается как взвешенная сумма N предыдущих отсчетов плюс случайная переменная.

Фрактальный гауссовский шум (Fractional Gaussian Noise – FGN) [5, 6]. FGN – стационарный в широком смысле стохастический процесс с определенными параметрами (средним значением, дисперсией, параметром Херста) и автокорреляционной функцией заданного вида. По сравнению с обычным гауссовым шумом, FGN имеет дополнительный параметр Херста, который количественно определяет степень фрактального масштабирования.

Сложные модели трафика требуют большого числа параметров, но обеспечивают малое проникновение в динамику трафика, наблюдаемого на реальных сетях.

К идентификационным характеристикам трафика относятся долговременная зависимость (LRD), медленно спадающая дисперсия, распределения с утяжеленными хвостами, фрактальные характеристики.

Другим методом является моделирование процесса *частичного броуновского движения* (Fractional Brownian Motion Process – fBm) с Херст-параметром $H \in [½, 1]$. Это гауссовский процесс с нулевым средним и стационарными инкрементами и ковариационной структурой:

$$\mathrm{Cov}[Z(t), Z(s)] = (\sigma^2/2)(t^2H + s^2H - |t-s|^2H).$$

В основе модели fBm лежит случайный процесс, начинающийся в начале координат с независимыми бесконечно малыми гауссовскими приращениями. В особом случае, когда $H = 0{,}5$, то $Z(t)$ – стандартное броуновское движение.

Самоподобные свойства $Z(t)$ основываются на факте, что $Z(\alpha t)$ идентичен по распределению с $\alpha HZ(t)$.

Инкрементный процесс $X(k) = Z(k+1) - Z(k), k \geq 0$ называется частичным гауссовым шумом (Fractional Gaussian Noise–fGn) и является стационарным (дискретно-временным) гауссовым процессом с автокорреляционной функцией

$$r(k) = (1/2)(|k+1|^{2H} - 2|k|^{2H} + |k-1|^{2H}),$$

где $k \geq 1$.

Асимптотически $r(k) \approx H^{(2H)}|k|^{-2H-2}$, т. е. демонстрирует долговременную зависимость (LRD). Экспериментальным анализом трасс было показано, что fBm-модель является разумным представлением «строго самоподобного трафика», например трафика данных, который имел ту же самую корреляционную структуру в сетях Ethernet в целом ряде диапазонов временных шкал (~10 мс через ~1000 с). Наоборот, трафик VBR должен иметь коротковременную зависимость (SRD), а корреляционная структура долговременной зависимости (LRD) наблюдается только на временных шкалах после нескольких секунд («асимптотическое самоподобие»).

Контрольные вопросы

1. Дайте определение системе ортогональных функций.
2. Поясните суть модели сигналов на основе теории решеток.
3. Поясните суть алгебраической модели сигналов.
4. Решетку можно определить как конечно порожденную подгруппу группы $\mathbb{R}^n$. Если ранг группы равен n, то решетка называется полной, в противном случае – неполной. Определите, как базис группы связан с базисом решетки.
5. Пусть $\mathbf{b}_1, \ldots, \mathbf{b}_k$ – базис решетки. Основным параллелепипедом решетки называется множество $T = \{x \in \mathbb{R}^n | x = c_1\mathbf{b}_1 + \cdots + c_k\mathbf{b}_k, 0 \leq c_i < 1\}$.

Определите как объем основного параллелепипеда связан с детерминантом решетки.

6. Пусть $\det(\Lambda)$ детерминант решетки и $\mathbf{b}_1, \ldots \mathbf{b}_n$ – ее базис. Докажите, что справедливо неравенство

$$det(\Lambda) \leq \|\mathbf{b}_1\| \cdot \ldots \cdot \|\mathbf{b}_n\|,$$

где $\|\mathbf{x}\|$ – евклидова норма вектора $\boldsymbol{x}$, т. е. $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}}$.

3. СЛОЖНОСТЬ АЛГОРИТМОВ ОБРАБОТКИ СИГНАЛОВ В ИНФОКОММУНИКАЦИОННЫХ СИСТЕМАХ

3.1. Базовые понятия

Главная задача теории сложности состоит в том, чтобы предоставить механизмы для классификации вычислительных алгоритмов согласно имеющимся ресурсам, необходимым для их решения. Классификация не должна зависеть от специфики вычислительной модели и должна учитывать трудность решения поставленной задачи. Оцениваемые ресурсы могут быть разными (время, требуемая память, разрядность, необходимое число процессоров и т. д.), но обычно это все-таки время и иногда память [8].

Определение. Алгоритмом называется вычислительная процедура, которая обрабатывает входные переменные, превращая их в процессе работы алгоритма в выходные переменные.

Термин «вычислительная процедура» можно представить, используя формальные вычислительные модели типа машин Тьюринга (машины со случайным доступом и булевскими циклами).

Представляет интерес найти наиболее эффективный (т. е. самый быстрый) алгоритм для решения вычислительной проблемы. Время, которое требуется алгоритму до своего завершения, зависит от входных данных, вытекающих из постановки задачи. Оценка используемого времени должна быть точной, особенно когда сравнивается выполнение двух алгоритмов.

Сравнительный анализ сложности нескольких алгоритмов решения одной и той же задачи позволяет делать обоснованный выбор лучшего из них. Если для решения задачи предлагается новый алгоритм, то необходимы доводы, говорящие о его преимуществах в сравнении с известными ранее алгоритмами, и анализ сложности может предоставить такие доводы.

С понятием сложности связываются затраты времени или памяти, соответствующие худшему случаю, либо затраты в среднем; при этом, чтобы обсуждать худший или «средний» случай, нужно прежде всего договориться, как определяется размер входа (размер входных данных) алгоритма и в чем измеряются затраты при работе алгоритма над фиксированным входом.

Размер входа. Выбор размера входа зависит от характера задачи. Наиболее часто его определяют как общее число символов в представлении входа.

В задачах сортировки и поиска размер входа – это количество элементов n входного массива.

В задачах на графах – число вершин $|V|$ или число ребер $|E|$ входного графа $G = (V, E)$, но, с другой стороны, $|V|$ и $|E|$ могут рассматриваться и совместно как два параметра размера входа.

В арифметических задачах размером входа может быть максимум абсолютных величин входных целых чисел или количество цифр в двоичной

записи этого максимума, или же суммарное количество двоичных цифр всех входных чисел.

Если размер входа является целым положительным числом, то возникающие функции являются последовательностями.

Пример. Размеры некоторых объектов.

1. Число битов в двоичном представлении n положительных целых чисел равно $1 + \lfloor \log n \rfloor$. Для простоты n будет занимать $\log n$ бит.

2. Если функция f является полиномиальной с наибольшим показателем k, каждый коэффициент которой является положительным целым числом не больше n, то для своего представления функция f потребует $(1 + k) \log n$ бит.

3. Если A – матрица с r строками и s столбцами и элементами, которые являются целыми положительными числами не больше чем n, то для своего представления A потребует $rs \log n$ бит.

Меры затрат. Для алгоритмов сортировки соответствующие затраты времени характеризуются количеством сравнений и перемещений элементов массива. Для алгоритмов на графах затраты могут складываться из операций над матрицей смежности или над массивом списков смежных вершин данного графа и из некоторых дополнительных вычислительных операций.

Для арифметических алгоритмов в качестве меры затрат может быть взято количество всех выполняемых арифметических операций или как альтернатива лишь наиболее дорогих операций (например мультипликативных); более тщательный анализ требует рассмотрения количества битовых операций. Если алгоритм является рандомизированным (содержит обращения к генератору случайных чисел с известным распределением), то затраты, вообще говоря, не определяются однозначно входом алгоритма, но зависят от полученных случайных чисел.

Можно рассматривать *усредненные затраты* для каждого конкретного входа; такие затраты уже будут функцией входа.

При фиксированном значении размера входа сами входы алгоритма могут варьироваться, при этом меняются и затраты; алгоритм может быть охарактеризован наибольшими или средними (в соответствии с распределением вероятностей на входах данного размера) затратами. В обоих случаях сложность алгоритма – это функция размера входа или соответственно нескольких параметров размера входа. Для этого понятия иногда используется термин «вычислительная сложность».

Описательная сложность определяется исходя из самой записи (текста) алгоритма. Одним из видов описательной сложности является длина записи алгоритма.

Сложность является функцией числового чаще всего целого аргумента, иногда – нескольких таких аргументов. Теория сложности изучает эти функции, сопоставленные как отдельным алгоритмам, так и классам алгоритмов решения некоторой задачи. В последнем случае возникает семейство функций, для которого могут обсуждаться вопросы о нахождении нижней границы, о

существовании минимальной функции этого семейства (если такая функция существует, то соответствующий ей алгоритм – *оптимальный* в рассматриваемом классе) и т. д.

Важным является также вопрос о существовании в данном классе алгоритмов, нацеленных на решение конкретной задачи, а именно такого алгоритма, сложность которого растет с увеличением размера входа не слишком быстро, например остается ограниченной некоторым полиномом, вид которого не оговаривается (такой алгоритм принято называть полиномиальным).

Сложности многих алгоритмов трудно или вообще нельзя представить в простом «замкнутом» виде. Точное значение сложности алгоритма для каждого конкретного значения размера входа часто не представляет особого интереса. Актуальным является исследование роста сложности при возрастании размера входа. Поэтому в теории сложности широко используется асимптотическое оценивание. Однако сравнение сложностей различных алгоритмов на основе асимптотических оценок этих сложностей возможно не для всех типов таких оценок.

3.2. Алгебраическая сложность

Пусть по входным данным (входу) x алгоритм A вычисляет результат (выход) y. Такое вычисление связано с затратами времени и памяти. Можно рассмотреть функции затрат

$$C_A^T(x),\ C_A^S(x),$$

где верхний индекс T указывает на временные затраты, S – на пространственные затраты.

Вид этих функций может быть очень непростым; их трудно исследовать методами математического анализа. Можно ввести некоторую неотрицательную числовую характеристику $\|x\|$ возможных входов алгоритма и оценивать функции затрат с помощью функций, зависящих не от **x**, а от $\|x\|$, которая характеризует «громоздкость» исходных данных:

$$C_A^T(x) \le \varphi(\|x\|),\ C_A^S(x) \le \psi(\|x\|). \qquad (3.1)$$

Если φ и ψ не слишком быстро растут при возрастании числового аргумента, то эти оценки могут свидетельствовать о некой эффективности алгоритма A.

Пусть функция $C_A^T(x)$ отражает затраты на выполнение лишь какого-то одного типа операций в предположении, что все эти операции требуют одинакового времени выполнения, не зависящего от вида и размера операндов, и это время равно единице. Тогда соответствующая функция $T_A(n)$ – это

сложность по числу операций рассматриваемого типа. Привлечение в качестве $C_A^S(x)$ функции, отражающей только количество хранимых дополнительных величин того или иного фиксированного типа, приводит к пространственной сложности $S_A(n)$ по числу величин рассматриваемого типа.

Такую сложность называют алгебраической сложностью по числу операций или числу величин рассматриваемого типа.

Алгебраическую сложность арифметических алгоритмов по числу умножений и делений называют мультипликативной сложностью.

Используется также понятие *битовой сложности*, которая позволяет принимать в расчет различие в размерах операндов и рассматривать при нахождении сложности непохожие операции, сопоставляя требуемые затраты на битовом уровне.

В некоторых случаях алгоритму сопоставляют несколько сложностей. Так, итоговые временные затраты сортировки массива с помощью сравнений складываются из затрат на сравнение и на перемещение элементов. Без учета типа элементов массива нельзя сказать, какая из операций является более дорогой. Поэтому для алгоритмов сортировки используются две сложности: с одной стороны – по числу сравнений, а с другой – по числу перемещений элементов, т. е. присваиваний (:=) или обменов (↔).

Если же рассматривается некоторый арифметический алгоритм, то, исследовав его мультипликативную сложность, можно дополнительно интересоваться в качестве уточнения **аддитивной сложностью** (числом сложений и вычитаний в худшем случае).

Асимптотические оценки. Часто трудно получить точную оценку времени выполнения алгоритма. В таких случаях прибегают к приближенным оценкам. Как правило, это бывает асимптотическая оценка продолжительности, т. е. оценка изменения времени выполнения алгоритма без линейной связи с изменениями размеров входных данных. Отсюда следует, что рассматриваются только функции, определенные для положительных целых чисел и всегда принимающих действительные значения при любых значениях аргументов. Пусть f и g – две такие функции.

Определения:

1. *Асимптотическая верхняя граница* $f(n) = O\big(g(n)\big)$ существует, если существует положительная константа c и положительное целое n_0, такие что $0 \le f(n) \le cg(n)$ для всех $n \ge n_0$.

2. *Асимптотическая нижняя граница* $f(n) = \Omega\big(g(n)\big)$ существует, если существует положительная константа c и положительное целое n_0, такие что $0 \le cf(n) \le g(n)$ для всех $n \ge n_0$.

3. *Асимптотическая средняя оценка* $f(n) = \Theta\big(g(n)\big)$ существует, если существуют положительные константы c_1 и c_2 и положительное целое n_0, такие что $c_1 g(n) \le f(n) \le c_2 g(n)$ для всех $n \ge n_0$.

4. *Нотация o* (*зависимость*) $f(n) = o\big(g(n)\big)$ существует, если для любых положительных $c > 0$ всегда существует положительное целое n_0, такое что $0 \le f(n) \le cg(n)$ для всех $n \ge n_0$.

Другими словами, выражение $f(n) = O\big(g(n)\big)$ означает, что функция $f(n)$ растет асимптотически не быстрее чем $g(n)$ в пределах ее произведения на константу. Выражение $f(n) = \Omega\big(g(n)\big)$ означает, что функция $f(n)$ растет асимптотически так же быстро, как растет $g(n)$ в пределах его произведения на константу. Выражение $f(n) = o(g(n))$ означает, что функция $g(n)$ является верхней границей для функции $f(n)$. Или, другими словами, $f(n)$ принимает всегда меньшие значения по сравнению с функцией $g(n)$ для любых значений n. Выражение $o(1)$ часто используют для обозначения функции $f(n)$, чей предел стремится к нулю при n, стремящейся к бесконечности.

Асимптотическая формула $f\ (n) = O(g(n))$ является удобным средством оценивания нетривиально устроенной функции $f(n)$ с помощью более простой функции $g(n)$; столь же полезными оказываются и оценки вида $f\ (n) = o(g(n))$.

Если алгоритмы имеют квадратичные сложности, то соответствующие сложности допускают оценку $O(n^2)$, но эти сложности являются величинами порядка n^2.

В математическом анализе это иногда записывается как $T(n) \asymp n^2$, где $T(n)$ – рассматриваемая функция, в данном случае – сложность. В последние годы в теории сложности алгоритмов вместо $f\ (n) \asymp g(n)$ стали писать $f(n) = \Theta(g(n))$.

Для сложности $T(n)$ мы имеем $T(n) = \Theta(n^2)$. Это более сильное утверждение, чем $T(n) = O(n^2)$, т. к. $T(n) = O(n^2)$ является лишь асимптотической верхней оценкой

Оценка $f\ (n) = O(g(n))$ есть *асимптотическая верхняя* оценка, равно как оценка $f\ (n) = \Omega(g(n))$ – *асимптотическая нижняя*.

Приближения некоторых часто встречающихся функций:

1. Полиномиальная функция. Если $f(n)$ – полином степени k с положительными коэффициентами, то $f(n) = \Theta(n^k)$.

2. Для любого $c > 0$ $\log_c n = \Theta(\log n)$.

3. Формула Стирлинга. Для любых целых $n \ge 1$

$$2\pi n \left(\frac{n}{e}\right)^n \le n! \le 2\pi n \left(\frac{n}{e}\right)^{n+(1/12n)}.$$

Таким образом,

$$n! = 2\pi n \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right),$$

$$n! = o(n^n) \text{ и } n! = \Omega(2^n).$$

4. $\log(n!) = \Theta(n \log n)$.

Пример. Сравнительные оценки роста некоторых функций. Пусть ε и c – произвольные константы, такие что $0 < \varepsilon < 1 < c$. Следующие функции внесены в список в увеличивающемся порядке их асимптотического роста:

$$1 < \ln\ln n < \ln n < exp(\ln n \ln\ln n) < n^{\varepsilon} < n^{c} < n^{\ln n} < c^{n} < n^{n} < c^{c^{n}}.$$

Получение оценок сложности

Пример. *Алгоритм Грэхема*. Задача построения выпуклой оболочки конечного множества M точек координатной плоскости, т. е. выпуклого многоугольника H, содержащего все множество M (рис. 3.1)

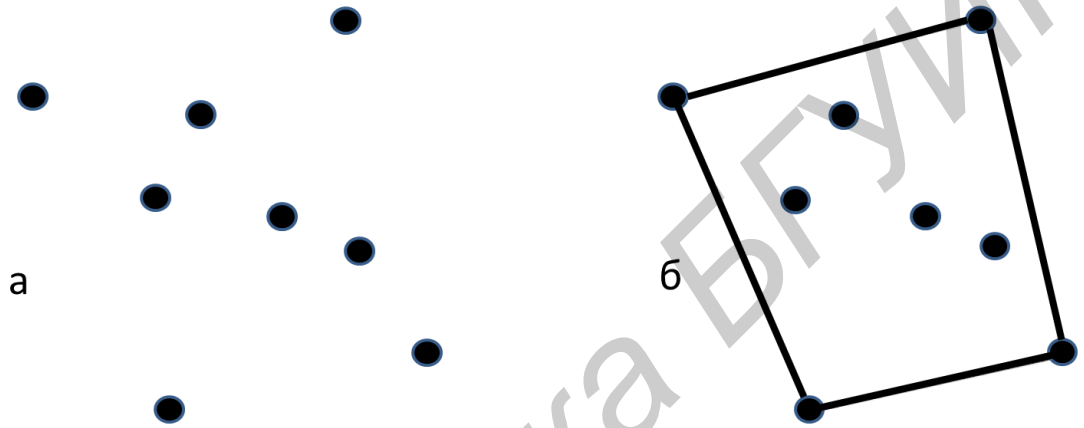


Рис. 3.1. Множество M:
а) конечное множество M точек плоскости; б) выпуклая оболочка множества M

Множество M задается массивом координат принадлежащих ему точек. Требуется построить массив координат вершин многоугольника H при обходе этого многоугольника, начиная с какой-нибудь его вершины, против часовой стрелки 1 (считаем, что это направление совпадает с направлением обхода точек (0, 0), (1, 0), (0, 1), (0, 0)).

Пусть n – число элементов множества M, будем считать это число размером входа. Алгоритм, основанный на переборе всех подмножеств множества M с проверкой для каждого из них, является ли оно множеством вершин искомого многоугольника H, имеет очень высокую сложность $\Omega(2^n)$.

Можно довольно быстро найти среди точек множества M такую, которая обязательно будет одной из вершин многоугольника H: достаточно выбрать в M точку P с наименьшей ординатой, а если таких точек несколько, то из этих нескольких взять ту, которая имеет наименьшую абсциссу. Дополнительно найдем точку O, которая принадлежит многоугольнику H, но не совпадает ни с одной из точек множества M: возьмем для этого какие-нибудь две точки из M и найдем середину соединяющего их отрезка (если впоследствии вдруг окажется, что эта точка принадлежит M, то можно будет удалить ее из M, т. к. она заведомо не является вершиной H).

Используя какую-нибудь сортировку с помощью сравнений, все точки множества M можно упорядочить по возрастанию углов между отрезком OP и отрезками, соединяющими O с точками множества M, при этом мы считаем, что величина каждого угла принадлежит полуинтервалу $[0, 2\pi)$. Если вдруг обнаружится, что два каких-то угла равны, то упорядочим соответствующие точки по удаленности от O, но для краткости будем говорить просто о сортировке по величине угла. Соединив точки в этом порядке (будем обозначать их $P_1, P_2, \ldots, P_n$, при этом $P_1 = P$) и соединив дополнительно P_n с P_1, мы получим замкнутую несамопересекающуюся ломаную, но ограниченный этой ломаной многоугольник может не быть выпуклым (рис. 3.2). Тогда среди вершин $P_2, P_3, \ldots, P_n$ найдется хотя бы одна, скажем P_k, вдавленная, которая принадлежит треугольнику $P_{k-1}OP_{k+1}$ при $k < n$ и треугольнику $P_{n-1}OP_1$ при $k = n$. Вдавленную вершину можно исключить из дальнейшего рассмотрения, соединив напрямую P_{k-1} с P_{k+1} или соответственно P_1 с P_{n-1}. Удалив все вдавленные вершины, мы получим требуемый многоугольник. Такова общая идея алгоритма.

Для любой сортировки массивов длиной n, имеющей некоторую сложность $s(n)$ по общему числу сравнений и перемещений элементов, существует алгоритм построения выпуклой оболочки n точек, заданных массивом своих координат на плоскости, сложность которого по общему числу арифметических операций, сравнений и перемещений есть $O(s(n))$. Пространственная сложность алгоритма Грэхема равна $O(n)$.

Пример. Пусть $G = (V, E)$ – ориентированный граф без кратных ребер и $v \in V$. Вояжем по G, выходящим из вершины v, будем называть любой путь со следующими характеристиками:

• начинается в вершине v;

• не проходит ни по одному из ребер дважды;

• завершается в вершине, из которой не выходит ни одно непройденное ребро (вояж не обязательно охватывает все ребра G).

Примером выходящего из вершины 1 вояжа в изображенном на рис. 3.2 графе служит (1, 2, 2, 3, 1, 4).

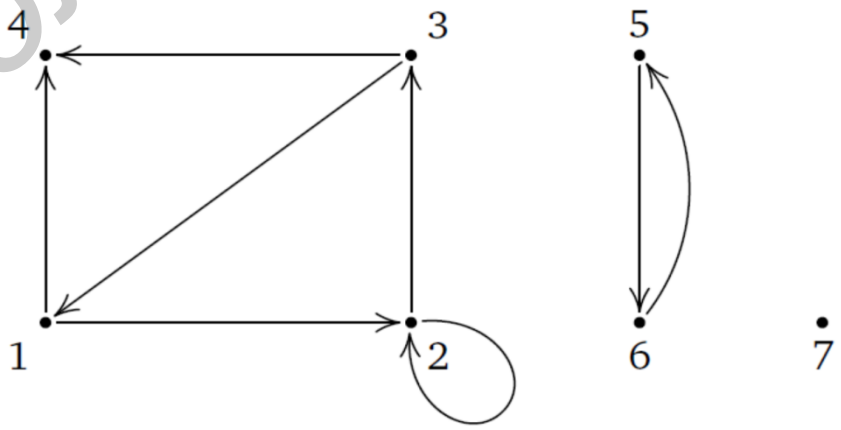


Рис. 3.2. Ориентированный граф

Построение какого-то одного выходящего из данной вершины v вояжа может быть выполнено произвольным блужданием по непройденным ребрам графа, завершающимся в вершине, из которой не выходят непройденные ребра. Систематизация блужданий может состоять в том, что, попав в некоторую вершину, мы выбираем для продолжения пути ребро, ведущее в вершину с наименьшим доступным номером.

Определяемый этим алгоритмом вояж может захватить и все ребра, например, когда граф имеет вид кольца (на рис. 3.3 изображен такой граф, для которого $|E| = |V| = 5$). Входом алгоритма является граф G и вершина v. Если мы рассматриваем $|E|$ как размер входа, то для сложности любого алгоритма построения вояжа обязательно имеет место нижняя оценка $\Omega(|E|)$. Возникает вопрос, можно ли обосновать верхнюю оценку $O(|E|)$? Если да, то в итоге это дало бы оценку $(\Theta(|E|))$.

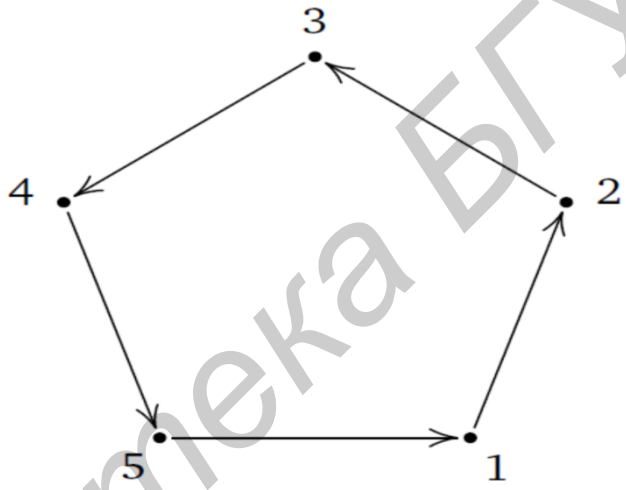


Рис. 3.3. Граф в виде кольца (любой вояж охватывает все ребра)

Когда речь идет о блужданиях по графу, при которых однажды пройденное ребро изымается из дальнейшего рассмотрения, более удобным оказывается представление графа в виде массива $a_1, a_2, \ldots, a_{|V|}$ списков смежных вершин.

В списке a_i, $i = 1, 2, \ldots, |V|$ в порядке возрастания содержатся номера вершин, в которые входят ребра, выходящие из вершины с номером i.

Для графа, приведенного на рис. 3.2, имеем

$$a_1 = (2,4), a_2 = (2,3), a_3 = (1,4), a_4 = (\ \ ), a_5 = (6), a_6 = (5), a_7 = (\ \ ).$$

Представление в виде массива списков смежных вершин возможно и для графов, имеющих кратные ребра; в этом случае номера вершин в каждом списке располагаются в порядке неубывания.

Если массивы списков смежных вершин используются в качестве представления ориентированных графов, то построение начинающегося с заданной вершины v вояжа в графе $G = (V, E)$ возможно с помощью

алгоритма, который при выборе числа $|E|$ ребер в качестве размера входа имеет сложность ($|E|$) по числу операций над списками. Эта же оценка имеет место для пространственной сложности, если считать, что пространственные затраты на хранение списка с числовыми элементами пропорциональны длине этого списка.

Сложность в среднем. Примем, что при каждом фиксированном значении s размера входа сами соответствующие входы образуют конечное множество $X_s = \{x : \|x\| = s\}$.

Будем предполагать, что каждому $x \in X_s$ приписана некоторая вероятность $P_s(x)$, $0 \le P_s(x) \le 1$ и при этом

$$\sum_{x \in X_s} P_s(x) = 1.$$

Таким образом, при каждом допустимом значении s размера входа множество X_s превращается в вероятностное пространство; по умолчанию считается, что вероятность распределена равномерно:

$$P_s(x) = \frac{1}{N_s},$$

где N_s – количество элементов множества X_s. Функции от s

$$\sum_{x \in X_s} C_A^T(x) P_s(x) = 1, \ \sum_{x \in X_s} C_A^S(x) P_s(x).$$

называются временной и соответственно пространственной сложностью в среднем алгоритма A.

Пусть при любом допустимом значении s множество X_s всех входов размером s является вероятностным пространством, в силу чего временные и пространственные затраты алгоритма A для входов x (т. е. $C_A^T(x)$, $C_A^S(x)$) размером s являются случайными величинами на X_s. Сложностью в среднем называется математическое ожидание соответствующей случайной величины.

Для временной и пространственной сложности в среднем алгоритма A используются обозначения $\bar{T}_A(\cdot)$, $\bar{S}_A(\cdot)$.

Анализ сложности в среднем для широкого ряда арифметических алгоритмов опирается на асимптотический закон распределения простых чисел.

Асимптотический закон распределения простых чисел. Для функции $\pi(n)$, значение которой равно количеству простых чисел, меньших данного натурального n, справедливо асимптотическое равенство

$$\pi(n) \sim \frac{n}{\ln n}.$$

Как следствие, вероятность того, что при выборе «наугад» одного из целых чисел $1, 2, \ldots, n$ попадется простое число, асимптотически равна $1/\ln n$.

3.3. Рандомизированные алгоритмы

Алгоритмы с элементами случайности, реализуемые обращениями к генераторам случайных чисел, называются рандомизированными.

Рандомизированные алгоритмы можно разделить на вероятностные, или, что то же самое, алгоритмы типа Монте-Карло и Лас-Вегас. Первый тип допускает, что ответ, который дает алгоритм для поставленной задачи с некоторым конкретным входом, может быть неправильным, хотя бы и с малой вероятностью; второй тип – Лас-Вегас – гарантирует правильный ответ, но (как и для алгоритмов типа Монте-Карло) время получения ответа для конкретного входа не определяется однозначно этим входом. Исключая специально оговариваемые редкие случаи, мы будем рассматривать только алгоритмы типа Лас-Вегас, не упоминая этого каждый раз.

Анализ сложности рандомизированного алгоритма сводится к нахождению математических ожиданий некоторых случайных величин.

Но ситуация здесь отличается от той, когда множество входов фиксированного размера рассматривается как вероятностное пространство и затраты алгоритма становятся случайными величинами, однозначно определенными для каждого конкретного входа. При исследовании рандомизированных алгоритмов вероятностное пространство, на котором рассматриваются случайные величины, состоит из сценариев выполнения алгоритма для фиксированного входа, и каждый сценарий определяется случайными числами, которые будут сгенерированы в соответствующие моменты выполнения алгоритма; за каждым таким сценарием закрепляется некоторая вероятность. В нашем контексте генератор случайных чисел можно представлять себе как стандартную функцию $\mathrm{random}(N)$ целого положительного аргумента N, результатом выполнения которой является элемент множества $\{0, 1, \ldots, N-1\}$, но невозможно предсказать точно, каким именно будет значение этой функции, т. к. любой из элементов указанного множества может появиться с вероятностью $1/N$.

Таким образом, затраты рандомизированного алгоритма при фиксированном входе не определяются однозначно, но зависят от сценария вычисления. При фиксированном входе мы можем рассмотреть множество всех сценариев и, приписав адекватным образом каждому из сценариев некоторую вероятность, ввести на полученном вероятностном пространстве случайную величину, значение которой для данного сценария равно соответствующим вычислительным затратам. Значение функции затрат на данном входе можно положить равным математическому ожиданию этой случайной величины (усредненным затратам для данного входа). А после того как определена функция затрат и принято соглашение о том, что такое размер входа, мы

можем, как обычно, рассматривать сложность алгоритма – в худшем случае или же в среднем.

3.4. Оценивание числа шагов (итераций) алгоритма

Выполнение алгоритма часто является последовательностью однотипных шагов (итераций). Если все шаги равнозатратны по времени, то общее их число с точностью до постоянного множителя эквивалентно временным затратам рассматриваемого алгоритма для данного входа и важной задачей является определение или хотя бы оценивание числа этих шагов.

Пример. Алгоритм Евклида. Пусть a_0, a_1 – натуральные числа, $a_0 \geq a_1$. Поиск наибольшего общего делителя (НОД) чисел a_0, a_1 по алгоритму Евклида требует выполнения серии однотипных шагов, каждый из которых – деление с остатком:

$$\begin{gathered} a_0 = q_1 a_1 + a_2, \\ a_1 = q_2 a_2 + a_3, \\ \cdots\cdots\cdots\cdots \\ a_{n-3} = q_{n-2} a_{n-2} + a_{n-1}, \\ a_{n-2} = q_{n-1} a_{n-1} + a_n, \\ a_{n-1} = q_n a_n. \end{gathered}$$

В этом случае $a_n = \text{НОД}(a_0, a_1)$, т. к.

$$\text{НОД}(a_0, a_1) = \text{НОД}(a_1, a_2) = \cdots = \text{НОД}(a_n, 0) = a_n \ .$$

Последовательность получаемых остатков убывает (остаток меньше делителя), при этом все остатки – неотрицательные целые. Не существует убывающей бесконечной последовательности, элементами которой являются неотрицательные целые числа, поэтому выполнение алгоритма Евклида (будем обозначать его буквой E) завершается для любых натуральных a_0, a_1 и число делений с остатком не превосходит a_1.

Обозначив через $C_E(a_0, a_1)$ исследуемое число делений с остатком, получаем

$$C_E(a_0, a_1) \leq a_1.$$

Это говорит о том, что если a_1 рассматривается как размер входа или если a_0, a_1 рассматриваются как два параметра размера входа, то сложность алгоритма Евклида по числу делений не превосходит a_1. Эта оценка является весьма грубой. Более точная оценка формулируется следующим образом.

Если рассматривать a_1 как размер входа a_0 алгоритма Евклида, то для сложности $T_E(a_1)$ по числу делений выполняется неравенство

$$T_E(a_1) \leq 2\log_2 a_1 + 1.$$

Эта же оценка имеет место и при рассмотрении двух параметров a_0, a_1 размера входа.

Рассматривая далее оценки сложности по числу делений алгоритма Евклида, мы будем использовать значение a_1 в качестве размера входа (значение a_0 может быть очень большим в сравнении с a_1, но первое же деление с остатком приведет к a_0, a_1, где $a_2 < a_1$).

Известно, что алгоритм Евклида допускает разнообразные обобщения и расширения. Прежде всего вместе с НОД(a_0, a_1) можно находить целые x, y, такие что

$$a_0 x + a_1 y = d = \text{НОД}(a_0, a_1).$$

Этот алгоритм мы назовем расширенным алгоритмом Евклида и будем его обозначать буквами EE от его английского названия $extended\ Euclidean$ – расширенный евклидов. Каждый шаг расширенного алгоритма Евклида содержит три мультипликативные операции – одно деление с остатком и два умножения.

Если рассматривать a_1 как размер входа a_0, a_1 расширенного алгоритма Евклида, то для мультипликативной сложности $T_{EE}(a_1)$ этого алгоритма имеем

$$T_{EE}(a_1) \leq 6\log_2 a_1 + 3.$$

Расширенный алгоритм Евклида дает возможность решать в целых числах линейные уравнения с целыми коэффициентами, он также играет существенную роль в модулярной арифметике.

Пример. Алгоритм Евклида.

ВХОДНЫЕ ДАННЫЕ: $a \triangleq a_0$ и $b \triangleq a_1$ – неотрицательные целые и $a > b$.

ВЫХОДНЫЕ ДАННЫЕ: НОД(a, b).

Пока $b \neq 0$ делаем следующее:

1) устанавливаем $r = a \bmod b$, $a = b$, $b = r$;

2) возвращаем (a).

Алгоритм Евклида выполняется за $O(\log n^2)$ битовых операций.

Пример. Вычислим по шагам НОД$(4864,3458) = 38$:

$$\begin{gathered} 4864 = 1 \cdot 3458 + 1406, \\ 3458 = 2 \cdot 1406 + 646, \\ 1406 = 2 \cdot 64 + 114, \\ 646 = 5 \cdot 114 + 76, \\ 114 = 1 \cdot 76 + 38, \\ 76 = 2 \cdot 38 + 0. \end{gathered}$$

Алгоритм может быть расширен так, чтобы он вычислял не только НОД d двух целых чисел a и b, но также x и y, такие что $ax + by = d$.

Пример. Расширенный алгоритм Евклида.

ВХОДНЫЕ ДАННЫЕ: a и b – неотрицательные целые и $a > b$.

ВЫХОДНЫЕ ДАННЫЕ: $d = \text{НОД}(a, b)$ и целые x и y, удовлетворяющие условию $ax + by = d$.

Если $b = 0$, установить $d = a, x = 1, y = 0$ и вернуть (d, x, y).

Установить $x_2 = 1, x_1 = 0, y_2 = 0, y_1 = 1$.

Пока $b \neq 0$ делаем следующее:

1) $q = \lfloor a/b \rfloor, r = a - qb, x = x_2 - qx_1, \ y = y_2 - qy_1$;

2) $a = b, b = r, x_2 = x_1, x_1 = x, y_1 = y_1, y_1 = y$.

Установить $d = a, x = x_2, y = y_2$ и вернуть (d, x, y).

Расширенный алгоритм Евклида выполняется за $O((\log n)^2)$ битовых операций.

Вычислим по шагам $\text{НОД}(4864, 3458) = 38$ и расширение $(4864)(32) + +(3458)(-45) = 38$.

Таблица 3.1

q	r	x	y	a	b	x_2	x_1	y_2	y_1
-	-	-	-	4864	3458	1	0	0	1
1	1406	1	–1	3458	1406	0	1	1	–1
2	646	–2	3	1406	646	1	–2	–1	3
2	114	5	–7	646	114	–2	5	3	–7
5	76	–27	38	114	76	5	–27	–7	38
1	38	32	–45	76	38	–27	32	38	–45
2	0	–91	128	38	0	32	–91	–45	128

Пусть $d = \text{НОД}(a, n)$. Уравнение $ax \equiv b \pmod n$ имеет решение тогда и только тогда, когда d делит b, в некоторых случаях получается точно d решений в интервале от 0 до $n - 1$, которые являются вычетами по модулю n/d.

Пример. Китайская теорема об остатках (Chinese Remainder Theorem – CRT). Если $n_1, n_2, \ldots, n_k$ взаимно простые числа, тогда выполняются следующие уравнения:

$$x \equiv a_1 (mod\ n_1);$$
$$x \equiv a_2 (mod\ n_2);$$
$$\vdots$$

$$x \equiv a_k (mod\ n_k),$$

которые имеют единственное решение при $n = n_1 \cdot n_2 \cdot \ldots \cdot n_k$.

Алгоритм Гаусса. Решение уравнений из китайской теоремы может быть получено как

$$x = \sum_{i=1}^{k} a_i N_i M_i,$$

где $N_i = n/n_i$, а $M_i = N_i^{-1} (\mathrm{mod}\, n_i)$. Сложность вычислений составляет $O((\ln n)^2)$ битовых операций.

Пример. Бинарный поиск места (т. е. значения p, $1 \le p \le n+1$) элемента y в упорядоченном массиве из n элементов $x_1 < x_2 < \cdots < x_n$:

$$p \coloneqq 1; q \coloneqq n+1;$$
$$while\ p < q$$
$$do$$
$$r \coloneqq \left\lfloor \frac{p+q}{2} \right\rfloor;$$
$$if\ x_r < y\ then$$
$$p \coloneqq r+1$$
$$else$$
$$q \coloneqq r$$
$$fi;$$
$$od.$$

Обозначим этот алгоритм буквами BS от его английского названия *Binary Search* – бинарный поиск. Будем считать число элементов сегмента массива длиной этого сегмента. При рассмотрении задач сортировки и поиска сегментом массива является любая упорядоченная часть следующего массива:

$$x_s < x_{s+1} < \cdots < x_{t-1} < x_t.$$

Легко видеть, что от сегмента длиной k мы переходим к сегменту длиной $\left\lfloor \frac{k}{2} \right\rfloor$ или $\left\lfloor \frac{k}{2} \right\rfloor - 1$.

Это говорит о том, что с каждым шагом алгоритма функция $L(k) = \lambda(k)$, где положительное k является длиной сегмента, рассматриваемого на данном шаге, убывает по крайней мере на единицу, пока не приходим к сегменту, содержащему один элемент, после чего еще одно сравнение решает задачу. Отсюда сложность бинарного поиска не превосходит $\lambda(n) = \lfloor \log_2 n \rfloor + 1$.

Свойство бинарного поиска. Сложность $T_{BS}(n)$ бинарного поиска места элемента в массиве длиной n по числу сравнений равна $\lfloor \log_2 n \rfloor + 1$, или, что то же самое, $\lceil \log_2 (n+1) \rceil$.

Бинарный поиск находит широчайшее применение при поиске информации в разнообразных таблицах. Укажем здесь еще одно его применение, касающееся вычислительной геометрии: он позволяет быстро определять, принадлежит ли произвольная точка Q выпуклому n-угольнику, заданному вершинами $P_1, P_2, \dots, P_n$. Можно легко построить какую-нибудь внутреннюю точку O данного n-угольника.

В силу его выпуклости точка Q – внутренняя, если и только если Q и O лежат в одной полуплоскости относительно любой из прямых $P_1P_2, \dots, P_{n-1}P_n, P_nP_1$.

Это соображение приводит к имеющему временную сложность $\Theta(n)$ алгоритму.

Но допустим, что проведены n добавочных лучей (рис. 3.4), исходящих из точки O и проходящих через вершины (считаем, что $O \neq Q$, иначе мы сразу бы заключили, что Q принадлежит многоугольнику).

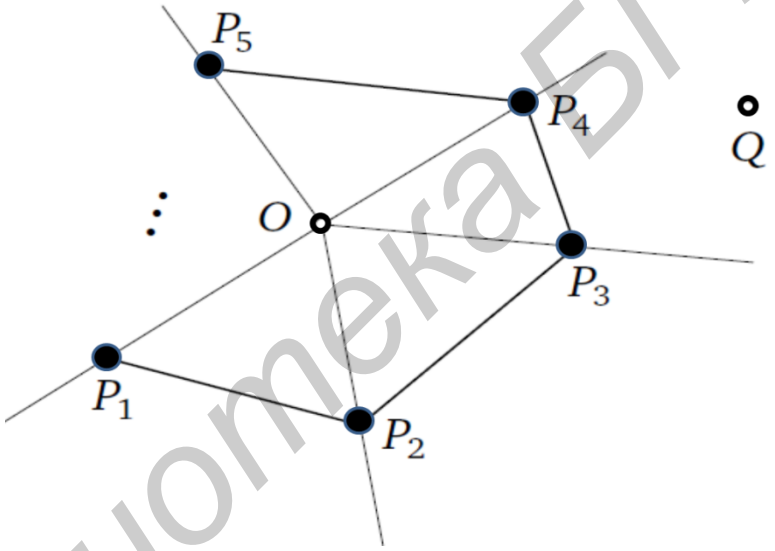


Рис. 3.4. Точка Q лежит между двумя лучами, проведенными из внутренней точки O многоугольника через его вершины

Можно установить, какому из углов $\angle P_1OP_2, \dots, \angle P_{n-1}OP_n, \angle P_nOP_1$ принадлежит точка Q: если углы пронумерованы в указанном порядке, то бинарным поиском определяется номер m угла, $1 \leq m \leq n$; при этом если Q лежит на одном из проведенных лучей, то из двух значений m берется любое. Узнав m, мы проверяем согласованность расположения точек O и Q по отношению к прямой, являющейся продолжением той стороны многоугольника, концы которой лежат на сторонах m-го угла.

Теперь заметим, что в самом проведении лучей $OP_1, OP_2, \dots, OP_n$ нет необходимости: сравнение $\angle P_1OQ$ с $\angle P_1OP_iQ$ требует ограниченного числа операций и в том случае, когда нам лишь известны координаты точек O, Q, P_1, P_i.

Основывающийся на бинарном поиске *алгоритм распознавания* принадлежности точки выпуклому n-угольнику имеет сложность $O(log\ n)$ по общему числу операций и пространственную сложность $O(1)$.

Полезным для решения ряда задач является то обстоятельство, что если точка не принадлежит данному выпуклому n-угольнику, то с помощью этого алгоритма мы дополнительно определяем одну из сторон n-угольника, которая из этой точки видна целиком.

Пример. Установим число этапов слияния при сортировке, предложенной Дж. фон Нейманом, которая является одним из вариантов сортировки слияниями. При сортировке фон Неймана шаг за шагом происходят переброски элементов исходного массива в дополнительный массив и наоборот и каждая переброска сопровождается слиянием соседних сегментов массива (рис. 3.5).

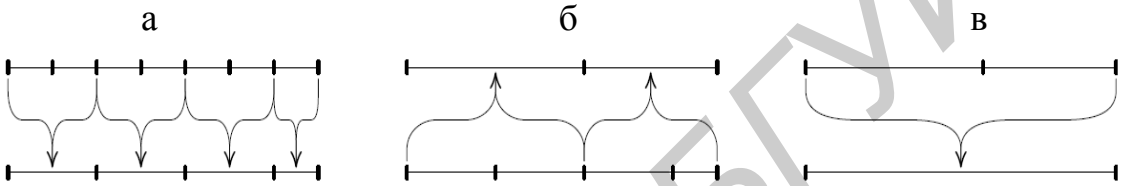


Рис. 3.5. Три последовательных шага сортировки фон Неймана, удлиняющих упорядоченные участки (сегменты):
а) переход от семи сегментов к четырем; б) переход от четырех сегментов к двум; в) переход от двух сегментов к одному (массив полностью упорядочен)

В качестве вспомогательного размера массива удобно рассмотреть число сегментов k (первоначально $k = n$, затем шаг за шагом k довольно быстро убывает).

Если на каком-то шаге имеем $k > 1$ уже построенных сегментов и $2^{m-1} < k \leq 2^m$ (это соответствует тому, что $\lceil \log_2 k \rceil = m$), то на следующем шаге сегментов будет $k' < k$ и $2^{m-2} < k' \leq 2^{m-1}$ (это соответствует тому, что $\lceil \log_2 k' \rceil = m - 1$). Поэтому функция $L(k) = \lceil \log_2 k \rceil$, где k – текущее число построенных сегментов, убывает с каждым шагом ровно на единицу. Таким образом, число этапов слияния, или число переброс сортируемых элементов из массива в массив, равно $\lceil \log_2 n \rceil$.

Сложность сортировки фон Неймана по числу сравнений меньше, чем $n\lceil \log_2 n \rceil$; сложность по числу присваиваний равна $n\lceil \log_2 n \rceil$.

Основываясь на сортировке фон Неймана, можно получить алгоритм построения выпуклой оболочки данных n точек координатной плоскости, имеющий сложность $O(n\ log\ n)$ в худшем случае по общему числу операций.

3.5. Качество оценок

После вывода оценки сложности алгоритма часто возникает вопрос о том, насколько эта оценка хороша и нельзя ли входящие в оценку константы и функции заменить другими так, чтобы оценка стала более точной и, как следствие, несущей больше информации об исследуемом алгоритме.

Пример. Докажем точность оценки сложности алгоритма Евклида

$$T_E(a_1) = O(\log(a_1)).$$

Для этого достаточно подобрать для алгоритма последовательность входов

$$\left(a_0^{(1)}, a_1^{(1)}\right), \left(a_0^{(2)}, a_1^{(2)}\right), \dots,$$

таких что последовательность $a_1^{(n)}$ (последовательность размеров входа) неограниченно возрастает и $C_E\left(a_0^{(n)}, a_1^{(n)}\right) = \Omega\left(\log a_1^{(n)}\right)$ при $n \to \infty$; тогда будем иметь $T_E\left(a_1^{(n)}\right) = \Omega\left(\log a_1^{(n)}\right)$.

Фактически нам нужна последовательность таких входов возрастающего размера, для каждого из которых алгоритм Евклида работает медленно. Подойдет, например, последовательность входов (F_{n+1}, F_n), $n = 0, 1, \dots$, где $F_0, F_1, F_2, \dots$ – числа Фибоначчи, определяемые равенствами $F_0 = 0$, $F_1 = 1$ и правилом «каждое следующее число равно сумме двух предыдущих», т. е. рекуррентным соотношением

$$F_{n+2} = F_{n+1} + F_n.$$

Из определения чисел Фибоначчи следует, что применение алгоритма Евклида к F_{n+1}, F_n при $n \geq 1$ требует n делений (все частные будут равны единице), поэтому $C_E(F_{n+1}, F_n) = n$.

По индукции доказывается, что для всех неотрицательных n справедлива формула Бине:

$$F_n = \frac{1}{\sqrt{5}}(\phi^n - (-\phi)^{-n}),$$

где

$$\phi = \frac{1+\sqrt{5}}{2} = 1{,}61803\dots$$

Деление отрезка в отношении $1:\phi$ называют *золотым сечением*.

Так как $\phi > 1$, то из формулы для F_n получаем $\phi^n = \sqrt{5}F_n\big(1 + o(1)\big)$ и, поскольку $\log_\phi\big(1 + o(1)\big) = o(1)$), с помощью логарифмирования по основанию ф имеем

$$C_E(F_{n+1}, F_n) = n = \log_\phi F_n + O(1) = \Omega(\log F_n).$$

Откуда следует точность оценки.

С точки зрения программирования более приемлемой будет запись алгоритма Евклида в виде

$$a := a_0; b = a_1;$$
$$while\ b > 0$$
$$do$$
$$d := rem(a, b);\ a := b;$$
$$b := d$$
$$od,$$

где rem – знак операции нахождения остатка от деления.

После выполнения алгоритма имеем $a = \text{НОД}(a_0, a_1)$. Использованы только три дополнительные переменные (a, b и d), и мы получаем

$$S_E(a_1) = 3.$$

Завершимость работы алгоритма. Если выполнение циклического (итерационного) алгоритма не завершается для некоторого входа v, то сложность этого алгоритма не определена для значения аргумента, равного $\|v\|$. Поэтому анализ сложности алгоритма прямо или косвенно включает исследование завершимости.

Установление завершимости может быть рассмотрено и как самостоятельная задача.

В случае рандомизированных алгоритмов мы сталкиваемся с возможностью того, что алгоритм завершается с вероятностью 1, но математическое ожидание времени от начала выполнения до полного завершения алгоритма бесконечно. Поэтому утверждения о завершимости возможны в двух формах: завершимость с вероятностью 1 и конечность ожидаемого времени выполнения. Вторая форма является более сильной.

Оптимальные алгоритмы. *Определение.* Пусть A – класс алгоритмов решения некоторой задачи. Пусть принято соглашение о том, в чем измеряются затраты алгоритмов и что считается размером входа, и пусть n – обозначение размера входа. Алгоритм $A \in A$ называется оптимальным в A, если $T_A(n)$ является нижней границей сложности алгоритмов из A.

Оптимальность схемы Горнера. Пусть n – произвольное неотрицательное целое число. Не существует алгоритма, который, используя только сложение, вычитание и умножение, позволяет вычислять значение

$$f(x) = a_n x^n + \cdots + a_1 x + a_0$$

по числам $x, a_0, a_1, \ldots, a_n$ так, что количество сложений или умножений всегда оказывается меньше n.

В терминах нижних границ это утверждение можно, очевидно, переформулировать следующим образом.

Пусть P – класс алгоритмов, вычисляющих значение $f(x)$ с помощью операций сложения, вычитания и умножения. Будем рассматривать число n как размер входа $x, a_0, a_1, \ldots, a_n$. Тогда n является нижней границей как аддитивной сложности (измеряемой числом сложений и вычитаний в худшем случае), так и мультипликативной сложности (измеряемой числом умножений в худшем случае) алгоритмов класса P .

3.6. Классы сложности

Определения:

1. *Полиномиальный алгоритм* – алгоритм, в самом плохом случае ограниченный асимптотической функцией $O(n^k)$, где n – размерность, а k – константа. Любой другой алгоритм, чья продолжительность не может быть ограничена такой функцией, называется *экспоненциальным*.

При рассмотрении полиномиальной сложности степень полинома имеет существенное значение. Например, алгоритмы с оценкой $O\left(n^{\ln \ln n}\right)$, где n – размерность входа, будут асимптотически медленнее алгоритмов с оценкой $O(n^{100})$. Вторые алгоритмы могут быть быстрее на практике при маленьких значениях n, особенно, если оценка производится по большому O.

2. *Подэкспоненциальный алгоритм* – алгоритм, в самом плохом случае ограниченный асимптотической функцией $e^{O(n)}$, где n – размерность входа.

Этот алгоритм асимптотически быстрее, чем экспоненциальный алгоритм, но медленнее полиномиального.

Пример. Подэкспоненциальный алгоритм. Пусть A – алгоритм, на вход которому подаются элементы из конечного поля F_q или целые q. Ожидаемое время работы алгоритма вычисляется по формуле

$$L_q[\alpha, c] = O\left(exp\left(c + o(1)\right)(\ln q)^{\alpha}(\ln \ln q)^{1-\alpha}\right),$$

где c и α – положительные константы.

Если $0 < \alpha < 1$, тогда A – подэкспоненциальный алгоритм. Наблюдения за константой:

1) $\alpha = 0$, $L_q(0, c)$ – алгоритмы полиномиальные со сложностью $\ln q$;

2) пока $\alpha = 1$, $L_q(1, c)$ – алгоритмы полиномиальные со сложностью q или полностью экспоненциальные с $\ln q$.

Для простоты теория вычислительной сложности ограничивается решением задач выбора, т. е. задач, которые имеют в качестве ответа или ДА, или НЕТ.

Определения:

1. Класс сложности ***P*** объединяет все задачи выбора, решаемые за полиномиальное время.

2. Класс сложности ***NP*** объединяет все задачи выбора, для которых ответ «ДА» можно получить за полиномиальное время при наличии дополнительной информации, называемой свидетельством.

3. Класс сложности ***co-NP*** объединяет все задачи выбора, для которых ответ «НЕТ» можно получить за полиномиальное время при наличии свидетельства.

Необходимо подчеркнуть, что, если задача выбора находится в классе ***NP***, это не означает, что при отсутствии свидетельства ответ «ДА» может быть легко получен; также трудно вычислимой задачей является получение этого свидетельства, если оно известно, то его можно использовать для эффективного решения задачи. Все выше перечисленное справедливо по отношению к ***co-NP***-классам сложности.

Пример. ***NP-проблема***. Задача выбора: факторизация.

ВХОД: положительное целое n.

ВОПРОС: n факторизуется? Есть ли целые числа, такие что $a, b > 1$, $n = ab$?

Решение. Факторизация принадлежит к ***NP***-классу, потому что, если n может быть разложено на два сомножителя, этот факт может быть проверен за полиномиальное время, достаточно поделить без остатка целое число n на известное целое число a, где $1 < a < n$. Обратная задача принадлежит к классу ***co-NP*** по аналогии. До сих пор не доказано, что факторизация принадлежит классу ***P***.

С другой стороны, $\boldsymbol{P} \subseteq \boldsymbol{NP}$ и $\boldsymbol{P} \subseteq$ ***co-NP***. Следующие вопросы являются наиболее значимыми (хотя строго не доказанными) в теории сложности:

1. $\boldsymbol{P} = \boldsymbol{NP}$?
2. ***NP***=***co-NP***?
3. $\boldsymbol{P} = \boldsymbol{NP} \cap$ ***co-NP***?

Пусть L_1 и L_2 – две решаемые задачи. Если выполнение L_1 занимает меньше времени, чем L_2, то будем писать $L_1 \leq_P L_2$. Если выполнение L_1 является частью L_2 и оно выполняется за полиномиальное время, то задача L_2 может быть выполнена. Если $L_1 \leq_P L_2$, то L_2 является столь же трудной (или не менее трудной), как и L_1.

Определение. Если L_1 и L_2 – две решаемые задачи и выполняются два условия $L_1 \leq_P L_2$, и $L_2 \leq_P L_1$, то задачи *эквивалентны в вычислительном смысле*.

Пусть L_1, L_2 и L_3 – три решаемые задачи, тогда:

а) если $L_1 \leq_P L_2$ и $L_2 \leq_P L_3$, то $L_1 \leq_P L_3$ (*транзитивность*);

б) если $L_1 \leq_P L_2$ $L_1 \leq_P L_2$ и $L_1 \in P$, то $L_2 \in P$.

Определение. Задача L является ***NP***-сложной, если:

1) $L \in \boldsymbol{NP}$;

2) $L_1 \leq_P L_2$ для любого $L_1 \in \boldsymbol{NP}$.

Класс всех ***NP***-сложных задач обозначается ***NPC*** (рис. 3.6). Существуют тысячи задач из различных областей математики, таких как комбинаторика, числовая теория, логика и других, которые относятся к ***NP***-классу.

Пример. Задача рюкзака. Дано множество $\{a_1, a_2, \dots, a_n\}$ и положительное целое s, требуется определить, является ли s суммой поднабора из a_i. Эта задача ***NP***-сложная.

Пусть L_1 и L_2 две решаемые задачи, тогда:

а) если L_1 оказалась ***N***-сложная и $L_1 \in \boldsymbol{P}$, то ***P=NP***;

б) если $L_1 \in \boldsymbol{NP}$, L_2 – ***NP***-сложная задача и $L_2 \leq_P L_1$, то L_1 тоже ***NP***-сложная;

в) если L_1 оказалась ***co-NP***-сложная и $L_1 \in \boldsymbol{NP}$, то ***NP=co-NP***.

Если некая задача, которая считалась полиномиальной оказалась ***NP***-сложной, то и другие задачи, отнесенные к этому классу, должны быть подвергнуты сомнению в их полиномиальности.

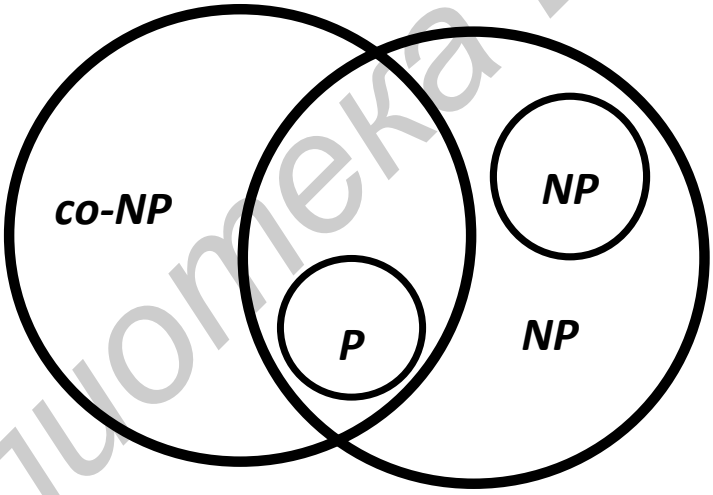


Рис. 3.6. Отношение классов

Доказательство ***NP***-сложности любой L_1 задачи состоит из выбора другой L_2 задачи, о которой известно, что она ***NP***-сложная, и доказательства факта $L_2 \leq_P L_1$.

Определение. Задача ***NP*** – *тяжелая*, если в классе существует ***NP***-сложная задача, которая полиноминально уменьшает ее трудоемкость.

Выявление ***NP***-тяжести не ограничивается только необходимостью в решении задачи. Существуют ***NP***-сложные задачи, которые одновременно является ***NP***-тяжелыми.

Пример. Вышеупомянутая задача о рюкзаке – ***NP***-тяжелая, если надо найти конкретное подмножество, состоящее из элементов a_i, которые в сумме дают s.

Классификация алгоритмов по вычислительной сложности. В зависимости от порядка сложности и вида результирующих данных алгоритмы обработки можно отнести к четырем уровням.

1. *Базовые арифметические операции.* Считается, что их сложность $O(1)$, хотя они отличаются по сложности битовых операций. Например, умножение двух m-разрядных чисел с фиксированной точкой требует $O(m^2)$ битовых операций, а сложение – $O(m)$.

2. *Скалярные алгоритмы* с вычислительной сложностью $O(N)$. Уровень включает в себя вычисление скалярного произведения N-компонентных векторов, вычисление значения полинома по схеме Горнера, фильтрацию (в частности, численное интегрирование и дифференцирование).

3. *Векторные алгоритмы* сложности $O(N^2)$. Примерами являются алгоритмы умножения матрицы на вектор для вычисления линейных преобразований (Фурье, Ганкеля – Теплица и др.), вычисления свертки векторов (полиномов) (их спектров и корреляций), фильтрации с КИХ и т. д. Некоторые из этих алгоритмов (для случая специальных матриц) можно улучшить до оценки $O(N\ log\ N)$. Примером является быстрое преобразование Фурье.

4. Алгоритмы сложности $O(N^3)$ составляют четвертый уровень. Это матричное произведение, вычисление собственных значений и векторов, прямые методы решения систем уравнений, сингулярное разложение и обращение матрицы, факторизация матрицы, метод наименьших квадратов, решение задач математического программирования, нахождение путей в графе, фильтрация по Калману и т. д.

Контрольные вопросы

1. Дайте определение сложности алгоритма.
2. Поясните суть асимптотических оценок сложности алгоритмов.
3. Как можно оценить сложность алгоритма Евклида?
4. Поясните итеративный алгоритм Гаусса и оцените сложность данного алгоритма.
5. Как можно построить алгоритм распознавания с использованием процедуры бинарного поиска?
6. Дайте определение качества оценок сложности алгоритмов.
7. Поясните на примере оптимальность алгоритма Горнера.
8. Какие существуют классы сложности?

4. БЫСТРЫЕ АЛГОРИТМЫ ОБРАБОТКИ СИГНАЛОВ

4.1. Основные типы алгоритмов обработки сигналов

Математические методы обработки сигналов можно подразделить на три группы, если в основу классификации положить принцип формирования отдельного элемента (отсчета) результата по некоторой совокупности элементов (отсчетов) исходного сигнала.

Точечные преобразования. В таких преобразованиях обработка каждого элемента исходных данных производится независимо от соседнего. Иначе говоря, значение каждого отсчета результата определяется как функция от одного отсчета исходного сигнала, причем номера отсчетов сигнала и результата одинаковы.

Пусть требуется обработать вектор из n отсчетов сигнала:

$$X = [x_0\ x_1\ x_2 \dots x_n]$$

и получить последовательность чисел $y = \{y_0\ \ y_1 \dots y_n\}$. Причем $y_i = f(x_i)$.

Точечные преобразования достаточно просты и наименее громоздки с точки зрения вычислительных затрат. Если обрабатывается матрица размером $N \times N$ элементов отсчетов исходного двумерного сигнала, то вычислительная сложность процедуры точечных преобразований составит $O_m = N^2$ (БО), где под базовой операцией (БО) понимается операция вида $y_i = f(x_i)$.

Локальные преобразования. При локальных преобразованиях обеспечивается формирование каждого элемента матрицы или вектора результата как функции от некоторого множества соседних элементов матрицы или вектора отсчетов исходного сигнала, составляющих некоторую локальную окрестность. При этом полагается, что местоположение вычисляемого отсчета результата (или текущий индекс элемента) задается координатами (или текущими индексами) центрального элемента локальной окрестности. Для формирования следующего элемента матрицы результата выполняется смещение окрестности вдоль строки матрицы исходных данных или вдоль исходного вектора. Такая перемещаемая окрестность часто носит название окна сканирования. При обработке матрицы исходных данных после прохождения всей строки матрицы исходных данных окно сканирования смещается на одну строку и возвращается в начало следующей строки, после чего продолжается обработка. Просматриваемая при перемещении окна сканирования полоса строк матрицы носит название полосы сканирования. Иначе говоря, при такой обработке

$$y_i = F(\hat{X})\ ;\ \hat{X} = \{x_{i-m/2}, x_{i-m/2+1}, \dots, x_i, \dots, x_{i+m/2-1}, x_{i+m/2}\},$$

где $i = 0, N-1$ – индекс отсчета результата; m – размер окна сканирования.

Если $i < m/2$ или $i > N - m/2$, что имеет место на практике при обработке начальных и конечных отсчетов вектора исходного сигнала, то элементы вектора исходных данных с «недостающими» индексами полагаются равными нулю.

Вычислительная сложность локального преобразования составляет $O_{\text{л}} = N^2 m^2$ (БО), где под базовой операцией понимается выполнение заданного преобразования для отдельного отсчета исходных сигналов. Примером локальных преобразований могут служить апериодическая свертка, корреляция, а также процедуры ранговой фильтрации.

Глобальное преобразование предусматривает формирование каждого отсчета результата как функции от всей совокупности отсчетов исходного сигнала и некоторого множества, меняющихся от одного отсчета результата к другому по определенному правилу коэффициентов, составляющих так называемое ядро преобразования. В случае обработки одномерного исходного сигнала глобальное преобразование можно определить как

$$Y_i = F(G_i, X); X = [x_0, x_1, \dots, x_n],\ i = 0, \dots, N-1,$$

где G_i – изменяемое ядро преобразования.

Вычислительная сложность глобального преобразования в общем случае для случая обработки двумерного сигнала составляет $O_{\text{г}} = N^4$ (БО), где под базовой операцией понимается выполнение заданного преобразования для отдельного элемента исходных данных. Примером подобных преобразований могут служить дискретные ортогональные преобразования типа преобразования Фурье, Хартли, Адамара.

Модели преобразований сигналов

Матрично-векторная модель. Многие алгоритмы цифровой обработки сигналов основываются на линейном преобразовании дискретного сигнала. Математически такое преобразование может быть записано с помощью матрично-векторного произведения

$$\boldsymbol{X} = \boldsymbol{M}\mathbf{x},$$

где $\mathbf{x} \in F^N$ – вектор дискретного сигнала; $\boldsymbol{M} \in F^{N \times N}$ – матрица преобразования над полем F; $\boldsymbol{X}$ – спектр сигнала.

Полиномиальная модель. В качестве моделей сигналов и фильтров преобразований можно использовать полиномы $S(z^{-1})$ и $H(z^{-1})$ степени $N - 1$, а саму операцию фильтрации рассматривать как результат произведения полиномов. В таком представлении результат линейной фильтрации имеет более высокую степень полинома, чем полиномы сигналов и фильтров, и фильтрация как алгебраическая форма не является замкнутой. Пространство сигналов можно замкнуть, используя операцию умножения по модулю $(z^{-N} - 1)$

$$H(z^{-1})S(z^{-1})\bmod(z^{-N}-1).$$

В этом случае сигналы и фильтры располагаются в пространстве полиномов в z^{-1} по модулю $z^{-N}-1$, которое обозначается как $\mathbf{C}[z^{-1}]/(z^{-N}-1)$ и называется полиномиальной алгеброй.

4.2. Модели спектральных преобразований на группах

Рассмотрим алгоритм дискретного преобразования Фурье F. Действие алгоритма F эквивалентно декомпозиции сигнального модуля в неприводимые компоненты, с помощью которых формируется спектр сигнала. Это созвучно декомпозиции векторного пространства в инвариантные подпространства с соответствующим линейным отображением. Алгебраическая структура дискретного преобразования Фурье (ДПФ) декомпозиции матрицы для групповой алгебры $\mathbb{C}[\mathbb{Z}]$ циклической группы $\mathbb{Z}_N$, состоящей из N элементов, запишется как

$$F_N : \mathbb{C}[\mathbb{Z}_N] \to \mathbb{C} \oplus \ldots \oplus \mathbb{C}, \tag{4.1}$$

где $\oplus$ – оператор прямой суммы.

В случае полиномиальной алгебры и сигнальной модели

$$a = m = \mathbb{C}[v]/p(v)$$

декомпозиция может быть описана с помощью китайской теоремы об остатках следующим образом:

$$F: \mathbb{C}[v]/p(v) \to \mathbb{C}[v]/(v-\alpha_0) \oplus \ldots \oplus \mathbb{C}[v]/(v-\alpha_{N-1}).$$

Полином $p(v)$ имеет степень $\deg(p) = N$ и нули $\{\alpha_n\}$ (в общем случае различные). Предполагается, что $\mathbb{C}[v]/p(v)$ представляет собой полную декомпозицию. Если $p(v) = q(r(v))$, то возможна двухэтапная декомпозиция:

$$\mathbb{C}[v]/p(v) \to \mathbb{C}[v]/(r(v)-\beta_0) \oplus \ldots \oplus \mathbb{C}[v]/(r(v)-\beta_{K-1}) \to$$

$$\to \mathbb{C}[v]/(v-\alpha_0) \oplus \ldots \oplus \mathbb{C}[v]/(v-\alpha_{N-1}),$$

где β_k – нули полинома $r(v)$, $deg\{r(v)\} = K$.

Если $N = KM$, то получаем алгоритм Кули – Тьюки преобразования Фурье, который производит вычисления по модулю

$$p(v) = v^N - 1 = (v^M)^K.$$

Такая структура позволяет обобщить понятие спектрального преобразования на другие классы сигналов.

1. Обобщение (4.1) на произвольную конечную группу $G \neq \mathbb{Z}_N$. Такой подход приводит к Фурье-анализу на группах. Использование $\mathbb{C}[G]$-модулей позволяет построить матрицы преобразований, обладающие свойствами симметрии, что удобно для дискретных преобразований, инвариантных к определенным классам сдвигов. Примером таких преобразований служит дискретное тригонометрическое преобразование.

2. Второе обобщение связано с выбором произвольного полинома $p(v) \neq v^N - 1$ и произвольного базиса $\mathbb{C}[v]/p(v)$. Такой подход приводит к полиномиальным преобразованиям. Так, если выбрать произвольный $p(v)$ и базис вида $(1, v, \dots, v^N)$, то приходим к матричным структурам Вандермонда, имеющим большое значение для пространственно-временных преобразований.

Фурье-анализ на группах. Пусть G будет конечной группой. Построим из элементов группы базис следующего векторного пространства:

$$\mathbb{C}[G] = \left\{ \sum_{g \in \mathbb{C}} a_g g | a_g \in \mathbb{C} \right\}.$$

Очевидно, что $\mathbb{C}[G]$ является векторным пространством, натянутым на элементы группы. Можно определить стандартный вид операции умножения в $\mathbb{C}[G]$, используя дистрибутивный закон и умножение элементов группы. Следовательно, $\mathbb{C}[G]$ представляет собой алгебру. С другой точки зрения, можно рассматривать $\mathbf{C}[G]$ как множество комплексных функций $g \to a_g$ на группе G. Регулярный модуль $M = A = \mathbb{C}[G]$ определяет сигнальную модель следующим образом. Если G состоит из n элементов, мы имеем множество $M = A = \mathbb{C}[G]$ и отображение

$$\Phi: \mathbb{C}^n \to \mathbb{C}[G], \quad s \to \sum_{g \in G} s_g g.$$

В данном случае сигнал и фильтры являются элементами алгебры групп. Операторами сдвига в группе G являются элементы, выбранные как образующие множества элементов группы. Если группа не циклическая, то G, а следовательно, и $\mathbb{C}[G]$ требуют как минимум двух образующих – генераторов или сдвигов. Если G не коммутативная группа, тогда не коммутативны и пары генераторов группы. Следовательно, определенная выше модель становится зависимой от сдвига.

Попробуем ответить на вопрос, как соотносятся полиномиальные и групповые алгебры? Полиномиальные алгебры всегда коммутативны и ясно, что некоммутативная группа, ассоциированная с групповой алгеброй, не может быть полиномиальной алгеброй. С другой стороны, известно, что каждая групповая алгебра для коммутативной группы является полиномиальной алгеброй с очень специфической структурой.

Рассмотрим пример, когда коммутативная группа G представлена в виде прямого произведения циклических групп

$$G = \mathbb{Z}_{n_1} \times \ldots \times \mathbb{Z}_{n_m},$$

где $\mathbb{Z}_{n_i}$ имеет размерность n_i; образующим элементом является x_i.

Тогда можно записать

$$\mathbb{C}[G] = \mathbb{C}[x_1, \ldots, x_m]/\langle\{x_i^{n_i} - 1 | 1 \le i \le m\}\rangle.$$

В случае одной переменной (одномерный сигнал) G должна быть циклической $G = \mathbb{Z}_n$ и мы получаем

$$\mathbb{C}[\mathbb{Z}_n] = \mathbb{C}[v]/(v^n - 1).$$

Полученная алгебра ассоциирована с дискретным преобразованием Фурье размером n.

Тот факт, что ДПФ ассоциировано с $\mathbb{C}[\mathbb{Z}_n]$ породил значительные усилия по развитию Фурье-анализа для других, некоммутативных групп и по поиску алгоритмов быстрых вычислений. С другой стороны, понимание того, что некоммутативные группы создают зависимые от сдвига модели, сдерживает применение таких групп при обработке сигналов.

На рис. 4.1 показана схема пересечения областей существования алгебр.

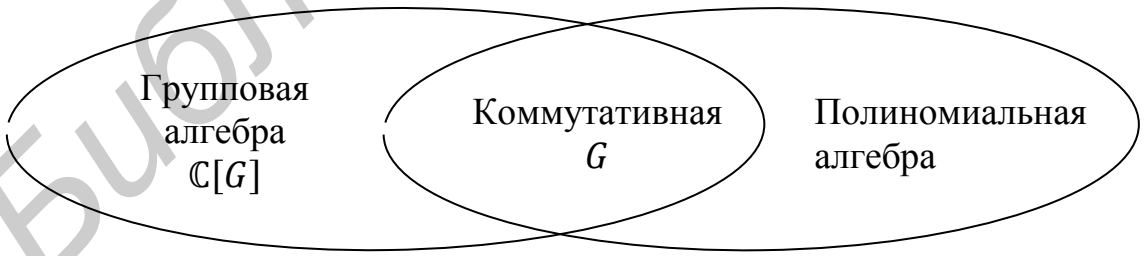


Рис. 4.1. Структурное расположение алгебр и коммутативной группы

Модели на основе полиномиальной алгебры одной переменной

Пусть $p(v)$ – полином степени $\deg\{p\} = n$. Тогда класс вычетов по модулю p

$$A = \mathbb{C}[v]/p(v) = \{h(v)|\deg\{h\} < n\}$$

является алгеброй со сложением полиномов и умножением полиномов по модулю p. Назовем такую алгебру полиномиальной алгеброй одной переменной. Полиномиальная алгебра всегда циклическая, т. е. формируется одним подходящим для этого элементом, обычно это v, который является и оператором сдвига. Это означает, что все элементы в A получены путем суммирования и умножения на скаляр, последовательного возведения v в степени. Другими словами, элементы алгебры – это полиномы переменной v.

Пример. Предположим, что

$$p(v) = (v-1)(v+1) = v^2 - 1.$$

Умножая два элемента v и $v + 1 \in A$ получаем

$$v(v+1) = v^2 + v \equiv v + 1 \bmod (v^2 - 1). \qquad (4.2)$$

Выражение (4.2) читается как «$v^2 + v$ конгруэнтно (или равно) $v + 1$ по модулю $(v^2 - 1)$». Таким образом, мы определяем равенство двух полиномов по модулю третьего полинома.

Модель сигналов. Выберем векторное пространство $\boldsymbol{V} = \mathbb{C}^n$, полиномиальную алгебру $A = \mathbb{C}[v]/p(v)$, $\deg\{p\} = n$ и модуль $M = A$. Далее выберем базис $\boldsymbol{b} = (p_0, \dots, p_{n-1})$. Этот выбор позволяет определить конечную n-мерную сигнальную модель (A, M, Φ). Биективное линейное отображение Ф при этом будет иметь вид

$$\Phi: \mathbb{C}^n \to M, \quad s \to \sum_{0 \le l < n} s_l p_l,$$

где $\boldsymbol{s} = \left(s_0, \dots, s_{n-1}\right)$ – сигнальный вектор.

Заметим, что Ф играет роль Z-преобразования и зависит от выбора базиса **b**. Базисный элемент p_i образует в модуле M единичный импульс, т. е. в векторе **s** компонента $s_i = 1$, а остальные компоненты $s_l = 0$ для $l \neq i$.

Пример. Для рассмотренного выше примера зададим базис $\boldsymbol{b} = \{1, v\}$ в алгебре $M = A = \mathbb{C}[v]/(v^2 - 1)$. Сигнальная модель для векторного пространства $\mathbb{C}^2$ определится как

$$\Phi: \mathbb{C}^n \to M, \quad \{s_0, s_1\} \to s_0 + s_1 v.$$

Фильтрация. Сигнальная модель определяет фильтрацию на сигнальном пространстве M как операцию A на M. Алгебра A описывает пространство

фильтров, а *A*-модуль *M*-пространство сигналов. Заметим, что даже если множества *A* и *M* равны, их алгебраические структуры не равны. Например, *A* воздействует на *M*, но не наоборот, фильтры (элементы *A*) могут каскадироваться, т. е. умножаться, в то время как сигналы (элементы *M*) – нет.

Определим сигнал

$$s = \Phi(\mathbf{s}) = \sum_{0 \le l < n} s_l p_l \in M$$

и фильтр $h \in A$.

Тогда фильтрация сигнала s с помощью фильтра h запишется как $h \cdot s \in M$, т. е. произведение полиномов h и s по модулю p. Заметим, что так как результат принадлежит *M*, его тоже можно рассматривать как сигнал. Операция фильтрации или умножения на h – это линейное отображение, которое имеет матричное представление в базисе **b**: $\varphi(h) \cdot s \in \mathbb{C}^N$.

Пример. Пусть $h = h_0 + h_1 v \in A = \mathbb{C}[v]/(v^2 - 1)$ – произвольный фильтр. Вычислим матрицу $\varphi(h)$ его преобразования в базисе b = (1, *v*). Столбцы матрицы представляют собой отклики фильтра на воздействия базисных компонент:

$$h \cdot 1 = h_0 + h_1 v \in M,$$

$$h \cdot v = h_0 v + h_1 v^2 \equiv h_1 + h_0 v \; mod \; (v^2) - 1.$$

Следовательно,

$$\varphi(h) = \begin{bmatrix} h_0 & h_1 \\ h_1 & h_0 \end{bmatrix} \text{ и } s \Leftrightarrow \varphi(h) \cdot s.$$

Спектры и Фурье-преобразование

Предположим, что *p*(*v*) – сепарабельный полином вида

$$p(v) = \sum_{k=0}^{N-1} (v - \alpha_k), \qquad \alpha_k \neq \alpha_l, \qquad \text{для } k \neq l.$$

Преобразование Фурье сигнала s можно записать в виде отображения или регулярного модуля *M*:

$$\Delta: \mathbb{C}[v]/p(v) \to \mathbb{C}[v]/(v - \alpha_0) \oplus \; \ldots \oplus \mathbb{C}[v]/(v - \alpha_{N-1}),$$

$$s = s(v) \to \big(s(\alpha_0), \ldots, s(\alpha_{N-1})\big).$$

Компонента регулярного модуля $M_k = \mathbb{C}[v]/(v - \alpha_k)$ имеет размерность один. Следовательно, элементы (векторы) $\mathbb{C}[v]/(v - \alpha_k)$ являются полиномами степени 0 или скалярами $c \in \mathbb{C}$. Далее M_k представляет собой A-модуль, т. к.

$$h = h(v) \in A, c \in M_k \text{ и } h(v) \cdot c \equiv h(\alpha_k) \cdot c \bmod(v - \alpha_k) \in M_k.$$

Спектральное преобразование можно рассматривать как проекцию сигнала $s \in \mathbb{C}[v]/(v - \alpha_k)$ на модуль

$$\mathbb{C}[v]/(v - \alpha_k) : s(v) \equiv s(\alpha_k) \bmod(v - \alpha_k).$$

Множество одномерных, неприводимых подмодулей $M_k = \mathbb{C}[v]/(v - \alpha_k)$ составляет спектр сигнального пространства M. Каждый подмодуль M_k одновременно формирует собственное пространство всех фильтров (или линейных систем) в A.

Спектр сигнала $s \in M$ можно рассматривать как вектор

$$\Delta(s) = \big(s(\alpha_0), \dots, s(\alpha_{N-1})\big),$$

а скаляр $s(\alpha_k)$ – как его компоненту.

Пример. Найдем спектр Фурье сигнала $s(v) = 1 + 3v$. Запишем преобразование Фурье в полиномиальной форме:

$$\Delta: \mathbb{C}[v]/(v^2 - 1) \to \mathbb{C}[v]/(v - 1) \oplus \mathbb{C}[v]/(v + 1).$$

Корни полинома $p(v)$ равны $\alpha_0 = 1$, $\alpha_1 = -1$. Спектр сигнала равен

$$s = s(v) \to S = \big(s(1), s(-1)\big) = (4, -2).$$

Матричная форма Фурье-преобразования. Преобразование Фурье – это линейное отображение, которое имеет матрицу преобразования $\boldsymbol{F}$, соответствующую выбранному базису

$$\mathbf{b} = (\mathbf{p}_0, \dots, \mathbf{p}_{N-1}).$$

Столбцы матрицы $\boldsymbol{F}$ определяются как коэффициенты полинома $p_l(v) \equiv p_l(\alpha_k)\, mod(v - \alpha_k)$. Соответственно l-й столбец матрицы равен вектору $[p_l(\alpha_0), \dots, p_l(\alpha_{N-1})]^T$.

Диагонализирующие свойства преобразования Фурье. Пусть $\boldsymbol{F}$ будет матрица Фурье-преобразования для регулярного A-модуля $\mathbb{C}[v]/p(v)$ в базисе **b** и соответствующая матрица фильтрового преобразования φ. Тогда преобразование

$$\boldsymbol{FAF}^{-1} = \mathrm{diag}(a_0, \dots, a_{N-1})$$

диагонализирует матрицу $\boldsymbol{A}$, если $\boldsymbol{A} = \varphi(h),\ h \in \boldsymbol{A}$.

В частности, $\boldsymbol{F}$ диагонализирует матрицу преобразования сдвига $\varphi(\tau)$. Оператор сдвига τ имеет частотный отклик, равный $(\alpha_0, \dots, \alpha_{N-1})$.

Рассмотренное свойство позволяет найти механизм фильтрации в частотной области. Действительно, рассмотрим соотношение

$$\varphi(h) \cdot s = \boldsymbol{F}^{-1}\boldsymbol{F}\varphi(h)\boldsymbol{F}^{-1}\boldsymbol{F} \cdot \mathbf{s} = \boldsymbol{F}^{-1}\mathrm{diag}\big(h(\alpha_0), \dots, h(\alpha_{N-1})\big)\boldsymbol{F} \cdot \mathbf{s},$$

которое показывает, что умножение матрицы преобразования φ на вектор $\mathbf{s}$ во временной области равносильно умножению в частотной области его спектра $\boldsymbol{F}\mathbf{s}$ на диагональную матрицу $\mathrm{diag}\big(h(\alpha_0), \dots, h(\alpha_{N-1})\big)$.

4.2.1. Алгоритмы дискретных ортогональных преобразований

Базисные концепции. Пусть H_N – унитарная $(N \times N)$ матрица, для которой справедливо соотношение

$$H_N H_N^* = H_N^* H_N = I_N,$$

где * – знак комплексного сопряжения; I_N – единичная диагональная матрица.

Для $(N \times 1)$ обрабатываемого входного вектора $\mathbf{x} = [x_0, x_1, \dots, x_{N-1}]^T$ вектор $\mathbf{y} = [y_0, y_1, \dots, y_{N-1}]^T$ записывается как

$$\mathbf{y} = H_N \mathbf{x} \tag{4.3}$$

и определяется как результат дискретного ортогонального преобразования вектора $\boldsymbol{x}$. В спектральном анализе $\boldsymbol{y}$ определяет спектр $\mathbf{x}$.

Уравнение (4.3) представляет произвольный вектор $\mathbf{y}$ в виде линейной комбинации столбцов матрицы H_N, которая в этом случае рассматривается как матрица множества базисных функций новой координатной системы.

Обратное дискретное ортогональное преобразование записывается как

$$\mathbf{x} = H_N^{-1}\mathbf{y} = H_N^*\mathbf{y}. \tag{4.4}$$

Здесь входной вектор $\mathbf{x}$ представляется в виде линейной комбинации столбцов матрицы H_N^* с весовыми элементами вектора $\mathbf{y}$.

Уравнения (4.3) и (4.4) соответствуют одномерному 1D-преобразованию. Разделимое двумерное 2D-преобразование может быть представлено в виде композиции одномерных преобразований. Пусть $\boldsymbol{X}$ – входная матрица размером $(N \times M)$. Разделимое 2D-преобразование матрицы $\boldsymbol{X}$ определится как

$$\boldsymbol{Y} = H_N \boldsymbol{X} Q_M, \tag{4.5}$$

где H_N и Q_M – унитарные матрицы размерами $(N \times N)$ и $(M \times M)$ соответственно.

Для ряда практических случаев $N = M$ и $Q_M = H_N^T$.

Обратное разделимое 2D-преобразование над матрицей $\boldsymbol{Y}$ размером $(N \times M)$ задается выражением

$$\boldsymbol{X} = H_N^* \boldsymbol{Y} Q_M^*. \tag{4.6}$$

Уравнения (4.5) и (4.6) матрицы $\boldsymbol{Y}$ или $\boldsymbol{X}$ представляют в виде линейной комбинации так называемых базисных изображений, которые формируются через произведение элементов столбцов левой матрицы (H_N или H_N^*) и элементов строк правой матрицы (Q_M или Q_M^*) соответственно.

Дискретные ортогональные преобразования широко используются во многих инфокоммуникационных приложениях, таких как цифровая обработка сигналов физического уровня, обработка речи и изображений, спектральная обработка процессов траффика и т. п.

Основной подход к применению дискретного ортогонального преобразования состоит в отображении входного сигнала в новую координатную систему с другими требуемыми, желаемыми распределениями энергии сигнала.

Дискретные ортогональные преобразования являются линейными, непараметрическими, обратимыми и обладают свойствами консервации энергии сигнала в исходной и спектральной областях. Это означает, что для входного сигнала $\mathbf{x} = [x_0, x_1, \dots, x_{N-1}]^T$ и его спектра $\mathbf{y} = [y_0, y_1, \dots, y_{N-1}]^T$ справедливо соотношение Парсеваля:

$$\sum_{i=0}^{N-1} |x_i|^2 = \sum_{i=0}^{N-1} |y_i|^2.$$

4.2.2. Дискретное преобразование Фурье

Прямое и обратное дискретное преобразование Фурье устанавливает взаимно однозначное соответствие между N отсчетами во временной области и N отсчетами в спектральной.

Положим, что последовательность $\{x[n] = x(nT_\partial),\ n = \{\ 0,1, \dots\ M - 1\}\}$ получена в результате дискретизации непрерывного сигнала с интервалом дискретизации, равным T_∂ с. Число $(N \geq M)$ определим как размер (число точек, порядок) дискретного преобразования Фурье. Пара прямого и обратного преобразований Фурье запишется в следующем виде:

$$\text{ДПФ:}\quad X(k) = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N}, \quad k = 0{:}\,N-1,$$

$$\text{ОДПФ:}\quad x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi nk/N}, \quad n = 0{:}\,N-1.$$

Здесь $e^{-j2\pi nk/N} \triangleq W_N^{nk}$ – базисные функции ДПФ.

В общем случае коэффициенты ДПФ $X(k)$ представляют собой комплексные числа, несут информацию об амплитуде и фазе сигнала.

Пара ДПФ справедлива для последовательности $x(n)$ конечной длины из N отсчетов. Однако при этом следует помнить, что обратное ДПФ дает N-периодическую функцию, совпадающую с исходной последовательностью $x(n)$ на интервале $n = 0, 1, \ldots, N-1$.

Алгоритмы быстрого преобразования Фурье

Быстрое преобразование Фурье (БПФ) включает набор эффективных алгоритмов, предназначенных для вычисления ДПФ. В основе алгоритмов лежит стратегия divide et impera (разделяй и властвуй). Идея БПФ по своей природе является алгебраической и заключается в следующем. Величина N, определяющая длину входной последовательности отсчетов, раскладывается на сомножители, затем вычисляются отдельные ДПФ меньших длин, чем N, из которых потом формируется выходная последовательность. Происходит так называемое *расщепление* исходного алгоритма на комбинацию подобных алгоритмов меньшего размера.

Положим, что для N-точечной последовательности нужно вычислить ее ДПФ и что N представляет собой составное число $N = n_1 n_2 \ldots n_\mu$, где $\{\mu_\mu\}$ – это набор множителей числа N, которые необязательно являются простыми. Алгоритм расщепления для вычисления такого ДПФ через комбинацию преобразований меньших размеров потребует число операций, пропорциональное

$$V_{\text{сл}} \approx N \sum_{j=1}^{\mu} n_i.$$

Такие алгоритмы называются алгоритмами быстрого преобразования Фурье. Наиболее важный частный случай имеет место, когда

$$n_1 = n_2 = \cdots = n_\mu = 2.$$

Для БПФ число операций в этом случае пропорционально

$$V_{\text{сл}} \approx N \log_2 N.$$

Быстрое полиномиальное преобразование Фурье. ДПФ вектора-реализации **a** по прямому алгоритму имеет оценку сложности вычислений $O(N^2)$ в предположении, что каждая арифметическая операция выполняется за один шаг.

Рассмотрим алгебраический подход к построению быстрого алгоритма ДПФ. Воспользуемся возможным подобием между частями тех сумм, которые порождаются процессом умножения матрицы дискретно-экспоненциальных функций на вектор.

Определение. Элемент W из произвольного коммутативного кольца $K =< K, +, \times, 0, 1 >$ является примитивным корнем N-й степени из единицы и обладает следующими свойствами

$$W \neq 1, \quad W^N = 1,$$

$$\sum_{i=0}^{N-1} W^{ip} = 0, \quad 1 \leq p < N.$$

Пример. $W = \exp(j\,2\pi/N)$, $j = \sqrt{-1}$ является примитивным корнем из единицы в поле (кольце) $\mathbb{C}$ (комплексных чисел).

Значения $W^0, W^1, \ldots, W^{N-1}$ – характеры циклической абелевой группы порядка N, на которой определено преобразование Фурье, принимающие значения в мультипликативной группе кольца K. Отсюда вытекают следующие возможности представления процесса вычисления ДПФ.

1. Вычисление ДПФ вектора отсчетов **x**

$$\mathbf{x} = \left[x[0], x[1], \ldots, x[N-1]\right]^T$$

эквивалентно вычислению полинома

$$p(z) = \sum_{i=0}^{N-1} x[i] z^i$$

в точках $z = W^0, W^1, \ldots, W^{N-1}$.

2. Вычисление $p(z)$ в точке $z = a$ равносильно нахождению остатка от деления $p(z)$ на $(z - a)$.

Доказательство. Запишем полином в виде

$$p(z) = (z - a)q(z) + c,$$

где c – постоянная. Тогда $p(a) = c$.

Таким образом, вычисление ДПФ сводится к нахождению остатка от деления полинома

$$p(z) = \sum_{i=0}^{N-1} x[i]z^i$$

степени $(N - 1)$ на каждый из полиномов:

$$(z - W^0), (z - W^1), \dots, (z - W^{N-1}).$$

Метод последовательного деления полинома $p(z)$ на каждый полином $(z - W^i)$ имеет сложность $O(N^2)$. Полиномы $(z - W^i)$ попарно взаимно просты, а их произведение равно $z^N - 1$.

В случае $N = 2^m$ можно воспользоваться следующим приемом.

Поскольку

$$W^{N/2} \equiv -1 \bmod M,$$

имеем

$$z^N - 1 = (z^{N/2} - 1)(z^{N/2} + 1) \equiv (z^{N/2-1})(z^{N/2} - W^{N/2}) \bmod M.$$

Аналогично

$$z^{N/2} - 1 = (z^{N/4} - 1)(z^{N/4} + 1) \equiv (z^{N/4} - 1)(z^{N/4} - W^{N/2}) \bmod M;$$

$$(z^{N/4} - W^{N/2}) \equiv (z^{N/4} - W^{N/4})(z^{N/4} - W^{3N/4}) \bmod M$$

и т. д., пока в правой части не получатся полиномы первой степени $(z - W^i)$.

Этот процесс можно описать деревом высоты n, где полином $(z^N - 1)$ является корнем, а линейные полиномы $(z - W^i)$ – листьями. Из каждой вершины, не являющейся листом, выходят два ребра, соответствующие разложению на два сомножителя.

Теперь для вычисления остатка от деления полинома $p(z)$ на каждый из полиномов $(z - W^0), (z - W^1), \dots, (z - W^{N-1})$ сначала делим $p(z)$ с остатком на полиномы $(z^{N/2} - 1)$ и $(z^{N/2} - W^{N/2})$, каждый из полученных остатков делим с остатком на полиномы вида $(z^{N/4} - W^{N/4})$ и т. д.

Для вычисления остатка от деления произвольного полинома степени $(2t - 1)$ на полином $(z^t - C)$, где C – некоторая константа, требуется $O(t)$ элементарных операций.

Пусть $N = 2^n$, W – примитивный корень степени N из единицы. Определим

$$(d_{n-1}, d_{n-2}, \dots, d_0)_2,$$

двоичное представление целого числа $j, 0 \le j < 2^n$

$$j = \sum_{i=0}^{n-1} d_i 2^i,$$

а также $rev(j)$ – целое число с инверсным двоичным представлением $(d_0, d_1, \dots, d_{n-1})_2$.

Тогда справедливо соотношение

$$\prod_{j=l}^{l+2^s-1} \left(z - W^{rev(j)}\right) \equiv z^{2^s} - W^{(l/2^s)} \bmod M,$$

где $0 \le s \le n$, число l кратно 2^s , $0 \le l < n - 1$.

Последнее соотношение задает порядок появления степеней i в полиномах вида $(z^t - \omega^i)$ при составлении попарных произведений так, чтобы все произведения имели вид $(z^{2^s} - \omega^i)$.

Алгоритм быстрого преобразования Фурье

Вход: Вектор

$$\mathbf{x} = [x[0], x[1], \dots, x[N - 1]]^T, N = 2^n.$$

Выход: Вектор

$$\boldsymbol{X} = [X(0), X(1), \dots, X(N - 1)]^T,$$

где

$$X(i) = \sum_{j=0}^{N-1} x[i] W^{ij}.$$

Алгоритм:

1. Положить

$$r_{0,n} \leftarrow \sum_{j=0}^{N-1} x[j] z^j.$$

2. Для $s = n - 1, n - 2, \ldots, 0$ выполнить следующие действия:

а) положить $t \leftarrow 0$;

б) пока $t < N - 1$:

– представить полином $r_{t,s+1}(z)$ в виде

$$\sum_{j=0}^{2^{s+1}-1} x[j] z^j;$$

– положить $e \leftarrow rev(t/2^s)?$;

– положить

$$r_{t,s}(z) \leftarrow \sum_{j=0}^{2^s-1} (x[j] + W^e x[j + 2^s]) z^j;$$

– положить

$$r_{t+2^s,s}(z) \leftarrow \sum_{j=0}^{2^s-1} \left(x[j] + W^{e+\frac{N}{2}} x[j + 2^s]\right) z^j;$$

в) положить $t \leftarrow t + 2^{s+1}$ и вернуться на шаг «б».

3. Для $i = 0, 1, \ldots, N - 1$ положить $X(rev(i)) \leftarrow r_{i,0}$.

4. Результат $\boldsymbol{X} = [X(0), X(1), \ldots, X(N - 1)]^T$.

Сложность алгоритма равна $O(N \log_2 N)$.

Рекуррентные алгоритмы БПФ. Наиболее быстродействующим при вычислении комплексных коэффициентов Фурье или других спектральных параметров алгоритмом является рекуррентное преобразование Фурье (РПФ), позволяющее получать оценку текущего спектра сигнала в реальном масштабе времени.

Нестационарный сигнал $f(n)$ задается дискретно в соответствии с теоремой Котельникова.

Рекуррентная формула вычисления коэффициента Фурье спектра сигнала имеет вид

$$F_{p+1}(n) = F_p(n) + e^{-j(2\pi n/N)}\Delta f, \qquad (4.7)$$

где p – номер выборки; n – номер гармоники; Δf – поправочный коэффициент, соответствующий разности между значениями сигнала в начале и конце выборки; N – размер выборки.

Для вычисления коэффициента Фурье $F_{p+1}(n)$ по формуле (4.7) необходимо произвести N операций умножения значений Δf на $e^{-j(2\pi n/N)}$

Вычисления действительной и мнимой частей коэффициента Фурье (4.7) можно представить соответственно как

$$\mathrm{Re}_{p+1}(n) = \mathrm{Re}_p(n) - \Delta f \cos\omega, \qquad \mathrm{Im}_{p+1}(n) = \mathrm{Im}_\mathrm{p}(n) - \Delta f \sin\omega,$$

где ω – частота среза спектра.

Фактически РПФ вычисляет оценки спектральных линий (гармоник) дискретного преобразования Фурье в скользящей системе отсчета, связанной с последними N отсчетами, размещаемыми в регистровой памяти анализатора, объемом N ячеек. Практический интерес представляет динамический спектр амплитуд:

$$A(n) = \sqrt{\left(\mathrm{Re}_{p+1}(n)\right)^2 + \left(\mathrm{Im}_{p+1}(n)\right)^2}.$$

При вычислении спектра в ячейки памяти, из которых были считаны $\mathrm{Re}_p(n)$ и $\mathrm{Im}_p(n)$), записываются полученные результаты $\mathrm{Re}_{p+1}(n)$ и $\mathrm{Im}_{p+1}(n)$. Они будут использованы для вычисления на следующем шаге. Процесс уточнения n-й гармоники продолжается до выполнения условий

$$\mathrm{Re}_{p+1}(n) = \mathrm{Re}_p(n),$$

$$\mathrm{Im}_{p+1}(n) = \mathrm{Im}_p(n).$$

Граф рекурсивного алгоритма для $N = 8$ показан на рис. 4.2.

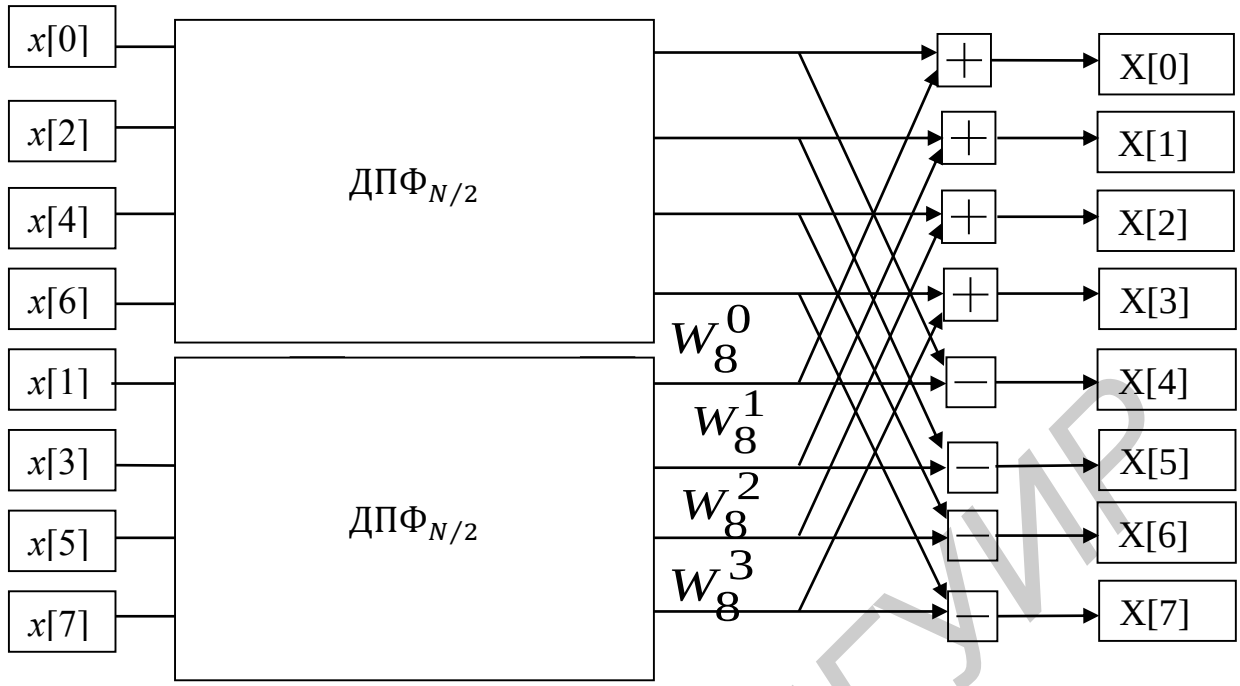


Рис. 4.2. Граф рекурсивного 8-точечного быстрого преобразования Фурье

Анализ результатов динамических свойств РПФ показывает, что преимущество метода РПФ при точном учете объема выборки N состоит в повышении на порядок быстродействия выполнения спектрального анализа нестационарного сигнала.

4.2.3. Преобразование на основе полиномов Чебышева

Преобразование Чебышева лежит в основе алгоритмов сжатия массивов информации.

Многочленом Чебышева называют многочлен $T_n(x)$, для которого

$$T_n(\cos\varphi) = \cos n\varphi \text{ для всех } \varphi.$$

Формула

$$\cos(n+1)\varphi + \cos(n-1)\varphi = 2\cos n\varphi \cos\varphi$$

показывает, что $T_n(x)$ действительно является многочленом (степени n), причем семейство этих многочленов определяется рекуррентным соотношением

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

и начальными условиями $T_0(x) = 1$ и $T_1(x) = x$.

Следующее утверждение показывает, что многочлены Чебышева образуют ортогональную систему относительно скалярного произведения

$$(f,g)=\int_{-1}^{1}\frac{1}{\sqrt{1-x^2}}f(x)g(x)dx.$$

Можно показать, что

$$\int_{-1}^{1}\frac{1}{\sqrt{1-x^2}}T_n(x)T_m(x)dx=\frac{\pi}{2}\delta_{m.n},$$

если одно из чисел n и m больше нуля, то

$$\int_{-1}^{1}\frac{1}{\sqrt{1-x^2}}T_0(x)T_0(x)dx=\pi.$$

Многочлен Чебышева $T_n(x)$ равен определителю матрицы порядка n

$$\begin{bmatrix} x & 1 & 0 & \cdots & 0 & 0 \\ 1 & 2x & 1 & \cdots & 0 & 0 \\ 0 & 1 & 2x & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2x & 1 \\ 0 & 0 & 0 & \cdots & 1 & 2x \end{bmatrix}.$$

Данное преобразование основано на разложении сигнала по базису ортогональных полиномов Чебышева. Под отсчетами будем понимать значения непрерывной функции на конечном множестве точек. Будем рассматривать двумерный непрерывный сигнал $s(x,y)$, для которого известны только отсчеты в узлах равномерной сетки.

Формулы прямого и обратного преобразований по базису ортогональных полиномов Чебышева для одномерного случая имеют вид:

$$C_m=\sqrt{\frac{2}{N}}\sum_{n=0}^{N-1}s(z_n)\cos\left(\pi m\frac{n+0{,}5}{N}\right),$$

$$C_0=\sqrt{\frac{1}{N}}\sum_{n=0}^{N-1}s(z_n),$$

$$s_M(z) = \sqrt{\frac{2}{N}} \sum_{m=0}^{M-1} g_m C_m T_m(z) = \sqrt{\frac{2}{N}} \sum_{m=0}^{M-1} g_m C_m \cos\big(m \arccos(z)\big),$$

$$g_m = \begin{cases} 0{,}5, & m = 0, \\ 1, & m > 0, \end{cases}$$

где $T_m(z)$ обозначает полином Чебышева степени m; $z_n = \cos(\pi\,(n + 0{,}5)/N)$ – значение n-го нуля полинома Чебышева степени N.

Прямое преобразование по виду совпадает с прямым дискретным косинусным преобразованием (ДКП) с точностью до узлов, в которых вычисляются значения сигнала $s(z)$. Для ДКП эти узлы представляют собой сетку равномерных отсчетов.

В случае преобразования с использованием ортогональных полиномов Чебышева $s(z_n)$ – это значения сигнала, вычисленные на неравномерной сетке, узлы которой соответствуют нулям полиномов Чебышева. Так как на практике сигнал $s(z)$ обычно задается отсчетами, вычисленными на равномерной сетке (например значение пикселей изображений), значения $s(z_n)$ можно вычислять, например, применяя методы интерполяции к значениям в узлах равномерной сетки, ближайшим к точке z_n.

При выполнении обратного преобразования восстановленный сигнал можно вычислять на произвольной сетке аргументов. Если взять такую сетку, узлы которой задаются по правилу $z_j = \cos(\pi\,(j + 0{,}5)/L)$, то обратное преобразование совпадет с обратным дискретным косинусным преобразованием.

Распространяя приведенные формулы на двумерный случай, можно выписать выражения для прямого C и обратного R преобразований в матричном виде:

$$C = \Phi S \Phi^T, \quad R = \Psi C \Psi^T,$$

где матрица прямого преобразования Φ вычисляется по формуле

$$\Phi = \sqrt{\frac{2}{N}} \begin{bmatrix} \sqrt{0{,}5} \\ \cos(\pi m\,(i + 0{,}5)/N) \end{bmatrix}, \quad \begin{matrix} m = 0, \ldots, M-1, \\ i = 0, \ldots, N-1. \end{matrix}$$

Матрица обратного преобразования Ψ вычисляется по формуле

$$\Psi = \sqrt{\frac{2}{N}} \begin{bmatrix} \sqrt{0{,}5} \\ \cos\left(m \arccos\left(\frac{2n}{L-1} - 1\right)\right) \end{bmatrix}, \quad \begin{matrix} m = 0, \ldots, M-1, \\ i = 0, \ldots, N-1. \end{matrix}$$

Таким образом, процедуру преобразования формально можно описать последовательностью следующих шагов:

1. Выделение блока изображения размером $N_1 \times N_1$.

2. Вычисление блока $N \times N$, соответствующего значениям сигнала в нулях полинома Чебышева.

3. Нахождение блока спектральных коэффициентов размером $M \times M$ путем вычисления прямого преобразования.

4. Применение обратного преобразования для нахождения блока восстановленного изображения размером $L \times L$.

Значения N_1, N, M, L могут быть различны, что является преимуществом преобразования, т. к. в случае необходимости изменения размера изображения достаточно просто выполнить обратное преобразование с другим значением L. Для изменения размера восстановленного изображения при использовании ДКП придется прибегать к методам интерполяции.

Разберем более подробно этап вычисления значений сигнала в нулях полиномов Чебышева. На рис. 4.3 изображен график сигнала $s(x)$ и его значений в узлах равномерной сетки, а также истинных значений сигнала в нулях полиномов Чебышева и их оценок. Для выполнения прямого преобразования требуется посчитать значение $s(x)$ в узлах $\{z_0, z_1, z_2, z_3\}$, соответствующих позициям нулей полинома Чебышева третей степени.

Для вычисления $s(z_i)$ можно воспользоваться методами интерполяции. На рис. 4.3 сплошные кресты обозначают истинные значения $s(z_i)$, а полыми крестами обозначены оценочные значения $\hat{s}(z_i)$, вычисленные методом линейной интерполяции по двум ближайшим точкам $s(x_{j-1})$ и $s(x_j)$. Различие $\hat{s}(z_i)$ и истинного значения $s(z_i)$ приводит к ошибке вычисления спектральных коэффициентов, что повлечет ошибку восстановления изображения. Таким образом, неточность вычисления истинного значения $s(z_i)$, определяет ошибку E_{z_n}.

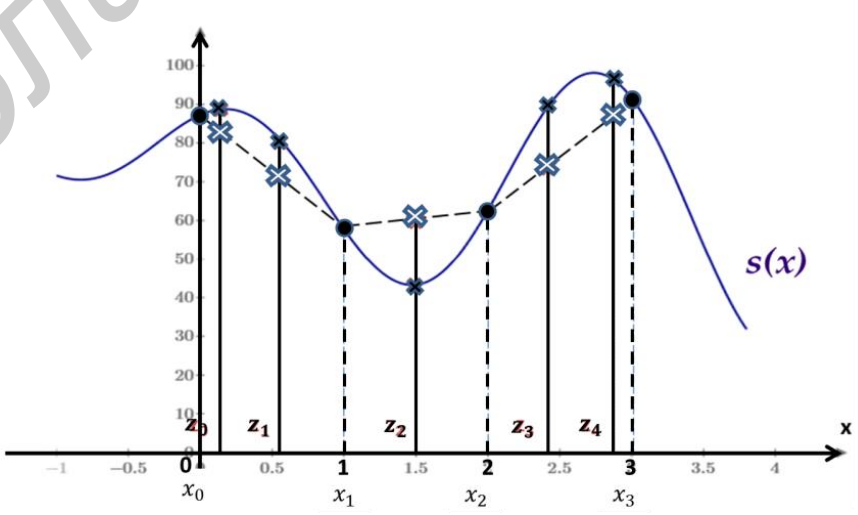


Рис. 4.3. График сигнала $s(x)$

Будем обозначать N_1 – размер исходного блока отсчетов сигнала на равномерной сетке, а N размер блока отсчетов сигнала, вычисленных в нулях полинома Чебышева. В соответствии со схемой преобразования значения N_1 и N можно выбирать произвольным образом. Для различных пар (N_1, N) ошибка E_{z_n} получается различной. Возникает вопрос о выборе таких параметров N_1, N, для которых E_{z_n} будет минимальна.

4.2.4. Преобразования на основе системы дискретных базисных функций на основе чисел Фибоначчи

Целочисленные последовательности Фибоначчи относятся к классу рекуррентных (возвратных) числовых последовательностей, в которых текущее число является линейной комбинацией предыдущих чисел. Для последовательности Фибоначчи каждое текущее число f_k равно сумме двух предыдущих чисел, поэтому рекуррентное соотношение в этом случае имеет простейший вид $f_k = f_{k-1} + f_{k-2}$ и выполняется для всех $k > 2$ при начальных значениях $f_1 = f_2 = 1$.

Пример. Записать первые шесть чисел Фибоначчи.

Решение. Используя рекуррентное соотношение, получим

$$\{f_k\} = \{1, 1, 2, 3, 5, 8\}.$$

Из данных чисел сформируем конечную базисную систему (назовем ее системой Фибоначчи), каждую функцию $fib(k, i)$ которой свяжем с числами Фибоначчи следующим правилом:

$$fib(k,i) = \begin{cases} f_{i+1}, & 0 \le i \le k, \\ -f_i, & l = k+1, \\ 0, & k+1 < i \ll N-1. \end{cases}$$

В соответствии с ним при $0 \le k \le N-2$ все функции

$$fib(k,i) = \{f_1, f_2, \dots, f_{k+1}, -f_{k+1}, 0, \dots, 0\},$$

а при $k = N-1$ функция $fib(N-1, i) = \{f_1, f_2, \dots, f_N\}$.

Пример. Записать систему Фибоначчи из шести базисных функций.

Решение. В этом случае $N = 6$ и

$$fib(0,i) = \{f_1, -f_2, 0, 0, 0, 0\},$$
$$fib(1,i) = \{f_1, f_2, -f_2, 0, 0, 0\},$$

$$fib(2,i) = \{f_1, f_2, f_3, -f_3, 0, 0\},$$
$$fib(3,i) = \{f_1, f_2, f_3, f_4, -f_4, 0\},$$
$$fib(4,i) = \{f_1, f_2, f_3, f_4, f_5, -f_5\},$$
$$fib(5,i) = \{f_1, f_2, f_3, f_4, f_5, f_6\}.$$

Используя значения чисел Фибоначчи предыдущего примера, запишем полученную числовую систему в следующем матричном виде:

$$\{fib(k,i)\} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & 2 & -2 & 0 & 0 \\ 1 & 1 & 2 & 3 & -3 & 0 \\ 1 & 1 & 2 & 3 & 5 & -5 \\ 1 & 1 & 2 & 3 & 5 & 6 \end{bmatrix}.$$

Система функций Фибоначчи является ортогональной. Все функции базиса Фибоначчи являются ненормированными. Их мощность равна

$$P_k = \frac{1}{N}\sum_{i=0}^{N-1} fib^2(k,i) = \frac{1}{N}\left(\sum_{i=0}^{k} f_{i+1}^2 + f_{k+1}^2\right) =$$

$$= \frac{1}{N} f_{k+1}(f_{k+1} + f_{k+2}) = f_{k+1} f_{k+3}/N \qquad (4.8)$$

для $0 \le k \le N-2$, а для $k = N-1$

$$P_k = \frac{1}{N}\sum_{i=0}^{N-1} f_{i+1}^2 = f_N f_{N+1}/N. \qquad (4.9)$$

Пример. Найти мощности базисных функций системы Фибоначчи для $N = 6$

Решение. Выполнив вычисления по формулам (4.8) и (4.9), получим

$$P_0 = f_1 f_3/N = 2/N = 1/3,$$
$$P_1 = f_2 f_3/N = 3/N = 1/2,$$
$$P_2 = f_3 f_5/N = 10/N = 5/3,$$
$$P_3 = f_4 f_6/N = 24/N = 4,$$
$$P_4 = f_5 f_7/N = 65/N = 65/6,$$
$$P_5 = f_6 f_{73}/N = 104/N = 52/3.$$

Базис Фибоначчи является полным базисом и может быть использован для спектрального представления любых решетчатых сигналов и функций ограниченной мощности. Дискретные преобразования Фурье и равенство Парсеваля в нем имеют типичный для дискретных базисов вид записи. Прямое преобразование Фурье – Фибоначчи может быть представлено в более удобной для вычислений форме:

$$X(k) = \frac{1}{NP_k}\sum_{i=0}^{N-1} x[i]fib(k,i) =$$

$$= \begin{cases} \left[\sum_{i=0}^{k} x[i]f_{i+1} - x[k+1]f_{k+1}\right] \Big/ (f_{k+1}f_{k+3}), & k \neq N-1, \\ \left[\sum_{i=0}^{k} x[i]f_{i+1}\right] \Big/ (f_N f_{N+1}), & k = N-1. \end{cases} \quad (4.10)$$

Как следует из приведенной зависимости, спектр сигнала в базисе Фибоначчи тесно связан с самими числами Фибоначчи. Приведем два примера спектров Фибоначчи конкретных сигналов.

Пример. Найти спектр Фибоначчи постоянного сигнала $x(i) = a$.

Решение. Используя формулу преобразования Фибоначчи (4.10), получим

$$X(k \neq N-1) = \left(a\sum_{i=0}^{k-1} f_{i+1}\right) \Big/ (f_{k+1}f_{k+3}),$$

$$X(N-1) = \left(a\sum_{i=0}^{k-1} f_{i+1}\right) \Big/ (f_N f_{N+1}).$$

Но

$$\sum_{i=0}^{n-1} f_{i+1} = \sum_{j=1}^{n} f_j = f_{n+2} - 1,$$

поэтому

$$X(k) = \begin{cases} \dfrac{a(f_{k+2}-1)}{f_{k+1}f_{k+3}}, & 0 \leq k \leq N-2, \\ \dfrac{a(f_{N+2}-1)}{f_N f_{N+1}}, & k = N-1. \end{cases}$$

Следовательно, для постоянного сигнала только одна нулевая спектральная составляющая равна нулю (в этом случае $f_{k+2} = f_2 = 1$). Все остальные $X(k)$ имеют ненулевые значения, связанные со значениями конкретных чисел Фибоначчи.

Пример. Найти спектр Фибоначчи линейного сигнала $x(i) = ai$.

Решение. При $0 \le k \le N-1$ в соответствии с формулой (4.10) имеем

$$X(k) = \frac{a}{NP_k} = \sum_{i=0}^{N-1} i\, fib(k,i) = \frac{a}{f_{k+1}f_{k+3}}\left(\sum_{i=0}^{k} i\, f_{i+1} - (k+1)f_{k+1}\right) =$$

$$= \frac{a}{f_{k+1}f_{k+3}}\left(\sum_{i=9}^{k-1} i\, f_{i+1} + kf_{k+1} - kf_{k+1} - f_{k+1}\right) =$$

$$= \frac{a}{f_{k+1}f_{k+3}}\left(\sum_{i=0}^{k-1} i\, f_{i+1} - f_{k+1}\right). \qquad (4.11)$$

Если сделать подстановку $i = j-1, j = 1,2,\dots,k$, то из выражения (4.11) можно получить

$$X(k) = \frac{a}{f_{k+1}f_{k+3}}\left(\sum_{j=1}^{k} j\, f_j - \sum_{j=1}^{k} f_j - f_{k+1}\right) = \frac{a}{f_{k+1}f_{k+3}}\left(\sum_{j=1}^{k} j\, f_j - \sum_{j=1}^{k+1} f_j\right).$$

Для чисел Фибоначчи справедливо соотношение

$$\sum_{j=1}^{k} j\, f_j = kf_{k+2} - f_{k+3} + 2,$$

поэтому окончательно получаем

$$X(k) = \frac{a}{f_{k+1}f_{k+3}}(kf_{k+2} - 2f_{k+3} + 3).$$

При $k = N-1$

$$X(N-1)=\frac{a}{f_N f_{N+1}}\sum_{i=0}^{N-1} i\, f_{i+1}=\frac{a}{f_N f_{N+1}}\left(\sum_{j}^{N} j\, f_j-\sum_{j=1}^{N} f_j\right)=$$

$$=\frac{a}{f_N f_{N+1}}(N f_{N+2}-f_{N+3}+2-f_{N+2}+1)=\frac{a}{f_N f_{N+1}}\big((N-1)f_{N+2}-f_{N+3}+3\big).$$

В спектре Фибоначчи линейного сигнала все составляющие $X(k)$ не равны нулю и зависят от значений соответствующих чисел Фибоначчи.

Базисные функции можно записать и в виде функции дискретного времени i, а их спектральные коэффициенты – в виде функций их номера k. Для этого нужно воспользоваться формулой Бине для чисел Фибоначчи:

$$f_i=\big(\alpha^i-\beta^i\big)/\sqrt{5},$$

где

$$\alpha=\big(1+\sqrt{5}\big)/2\approx 1{,}618;\quad \beta=\big(1-\sqrt{5}\big)/2\approx -0{,}618.$$

Из формулы Бине следует, что числа Фибоначчи имеют экспоненциальный характер изменения. Поэтому такой же характер изменения будут иметь и сами базисные функции Фибоначчи, и их спектры различных сигналов. Для этих базисных функций адекватными в смысле обеспечения концентрации энергии в минимальном числе спектральных коэффициентов будут экспоненциальные сигналы вида α^i, поскольку для них спектр Фибоначчи близок к дельта-функции. Поэтому функции Фибоначчи могут оказаться весьма полезными при обработке сигналов с экспоненциальным законом изменения во времени

4.2.5. Ортогональное разложение случайных процессов

Разложение случайных процессов основано на возможности представления корреляционной функции $R(t,\ \tau)$ случайного процесса в виде равномерно сходящегося ряда Карунена – Лоэва:

$$R(t,\tau)=\sum_{k=1}^{\infty}\lambda_k\varphi_k(t)\varphi_k(\tau),\qquad a\le t\le b,\qquad a\le\tau\le b,\qquad (4.12)$$

где $\{\varphi_k(t)\}$ – собственные функции; λ_k – собственные значения однородного интегрального уравнения Фредгольма второго рода

$$\int_a^b R(t,\tau)\varphi(\tau)d\tau=\lambda\varphi(t).\qquad (4.13)$$

Функция $R(t,\tau)$ называется ядром интегрального уравнения.

Если $R(t, \tau)$ – вещественная функция, то и собственные функции $\varphi_k(t)$, и собственные значения λ_k также вещественны, причем если $R(t,\tau)$ – положительно определенная функция, то система функций $\{\varphi_k(t)\}$ ортогональна на интервале $[a, b]$, т. е.

$$\int_a^b \varphi_m(t)\varphi_n(t)dt = \delta_{m,n} = \begin{cases} 1, & \text{при } m = n, \\ 0, & \text{при } m \neq n, \end{cases} \tag{4.14}$$

где $\delta_{m,n}$ – символ Кронекера.

Таким образом, любой непрерывный в среднеквадратичном смысле случайный процесс $x(t)$ с автокорреляционной функцией $R(t, \tau)$ может быть представлен в среднеквадратичном смысле на интервале $[a, b]$ ортогональным разложением

$$x(t) = \sum_{k=1}^{\infty} X(k)\varphi_k(t), \tag{4.15}$$

где

$$X(k) = \int_a^b x(t)(t)\varphi_n(t)dt\ . \tag{4.16}$$

Математическое ожидание процесса $X(k)$ определить как

$$M\{X(m)X(k)\} = M\left\{\int_a^b x(t)\varphi_m(t)dt \int_a^b x(\tau)\varphi_m(\tau)d\tau\right\} =$$

$$= \int_a^b \int_a^b R(t,\tau)\varphi_m(t)\varphi_k(\tau)dtd\tau.$$

С учетом условий (4.15) и (4.16) получим

$$M\{X(m)X(k)\} = \lambda_k \int_a^b \varphi_m(t)\varphi_k(t)dt = \lambda_k\delta_{m,k},$$

что позволяет сделать вывод о некоррелированности (ортогональности) коэффициентов ряда $\{X(l)\}$. Причем второй начальный момент равен $M\{X^2(m)\} = \lambda_m$.

Если ряд записать в форме

$$x(t) = \sum_{m=1}^{\infty} \sigma_m X'(m)\varphi_m(t)$$

и принять $\sigma_m^2 = \lambda_m$, то выполняется условие ортонормированности для случайных величин $X'(m)$, а именно $M\{X'(m)X'(k)\} = \delta_{m,k}$.

Пример. Пусть случайный процесс $x(t)$ имеет вид гармонического колебания со случайной начальной фазой

$$x(t) = A\cos(\omega_0 t + \theta),$$

где A, ω_0 – постоянные величины; θ – случайная величина, равномерно распределенная в интервале $(-\pi;\ \pi)$.

Случайный процесс периодический с периодом $T = (2\pi/\omega_0)$. Корреляционная функция такого процесса имеет вид

$$R(\tau) = \frac{A^2}{2}\cos(\omega_0\tau).$$

Уравнение (4.13) имеет вид

$$\int_{-T/2}^{T/2} \frac{A^2}{2}\cos[\omega_0(t-\tau)]\,\varphi(\tau)d\tau = \lambda\varphi(t), \quad -T/2 \le t \le T/2.$$

Разложение корреляционной функции запишем как

$$R(t-\tau) = \sum_{i=1}^{\infty} \lambda_i\varphi_i(t)\varphi_i(\tau) = \frac{A^2}{2}\cos[\omega_0(t-\tau)] =$$

$$= \frac{A^2}{2}\cos\omega_0 t\cos\omega_0\tau + \frac{A^2}{2}\sin\omega_0 t\sin\omega_0\tau.$$

Собственные функции и собственные значения имеют вид

$$\varphi_1(t) = \sqrt{\frac{2}{T}}\cos\omega_0 t\,,\ \lambda_1 = \frac{A^2T}{4}\,;$$

$$\varphi_2(t) = \sqrt{\frac{2}{T}}\sin\omega_0 t\,,\ \lambda_2 = \frac{A^2T}{4}\,.$$

Ортогональное разложение случайного процесса принимает вид

$$x(t) = A\cos(\omega_0 t + \theta) = X(1)\sqrt{\frac{2}{T}}\cos\omega_0 t + X(2)\sqrt{\frac{2}{T}}\sin\omega_0 t,$$

где $X(1) = A\sqrt{\frac{T}{2}}\cos\theta$, $X(2) = -A\sqrt{\frac{T}{2}}\sin\theta$.

Дискретное разложение периодической случайной последовательности

Известно, что корреляционная матрица стационарного процесса является теплицевой. Если корреляционная последовательность случайного процесса является периодической с периодом M, то корреляционная матрица становится циркулянтной. Общий вид циркулянтной матрицы

$$\boldsymbol{A} = \begin{bmatrix} a_1 & a_2 & \dots & \dots & a_n \\ a_n & a_1 & a_2 & \cdots & a_{n-1} \\ a_{n-1} & a_n & a_1 & \dots & a_{n-2} \\ \dots & \dots & \dots & \ddots & \dots \\ a_2 & a_3 & \dots & a_n & a_1 \end{bmatrix}.$$

Любая строка такой матрицы получается из предыдущей путем циклического сдвига на одну позицию вправо.

Циркулянтная корреляционная матрица имеет вид

$$\boldsymbol{R}_x = \begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \dots & r_x(M-1) \\ r_x(M-1) & r_x(0) & r_x(1) & \dots & r_x(M-2) \\ r_x(M-2) & r_x(M-1) & r_x(0) & \dots & r_x(M-3) \\ \dots & \dots & \dots & \ddots & \dots \\ r_x(1) & r_x(2) & \dots & r_x(M-1) & r_x(0) \end{bmatrix}.$$

Введем M – точечное ДПФ последовательности отсчетов корреляционной функции $r_x(l)$:

$$\tilde{R}_x(k) = \sum_{l=0}^{M-1} r_x(l) W_M^{kl},$$

где $W_M = \exp(-j2\pi/M)$.

Рассмотрим вектор

$$\mathbf{w}_k = \frac{1}{\sqrt{M}}\left[1, W_M^k, W_M^{2k}, \ldots . W_M^{(M-1)k}\right]^T, \quad 0 \le k \le M-1.$$

Нетрудно установить, что справедливо следующее соотношение

$$\boldsymbol{R}_x \mathbf{w}_k = \tilde{R}_x(k)\mathbf{w}_k, \quad 0 \le k \le M-1.$$

Отсюда следует, что вектор $\mathbf{w}_k$ ДПФ является собственным вектором циркулянтной корреляционной матрицы $\boldsymbol{R}_x$, а величина $\tilde{R}_x(k)$ – собственным значением этой матрицы. Таким образом, ДПФ эквивалентно дискретному преобразованию Карунена – Лоэва (ДПКЛ) периодической импульсной последовательности.

Справедливо еще одно полезное соотношение, которое говорит о том, что ДПФ диагонализирует циркулянтную корреляционную матрицу

$$\boldsymbol{W}^H \boldsymbol{R}_x \boldsymbol{W} = diag\left(\tilde{R}_x(0), \tilde{R}_x(1), \ldots, \tilde{R}_x(M-1)\right),$$

где $\boldsymbol{W}$ – матрица ДЭФ.

Пример. Дано разностное уравнение

$$x[n] = v[n] + bv[n-1],$$

где $v[n]$ – белый гауссовский шум с нулевым средним значением и единичной дисперсией.

Найти ДПКЛ при $M = 3$.

Решение. Найдем матрицу $\boldsymbol{R}_x$:

$$\boldsymbol{R}_x = \begin{bmatrix} 1+b^2 & b & 0 \\ b & 1+b^2 & b \\ 0 & b & 1+b^2 \end{bmatrix}.$$

Вычислим собственные значения матрицы $\boldsymbol{R}_x$:

$$|\boldsymbol{R}\mathbf{x} - \lambda \boldsymbol{I}| = (1 + b^2 - \lambda)[(1 + b^2 - \lambda)^2 - 2b^2] = 0,$$

где $\mathbf{x} = \left[x[0], \ldots, x[N-1]\right]^T$ – вектор обрабатываемых значений.

Отсюда находим собственные значения

$$\lambda^{(1)} = 1 + b^2; \quad \lambda^{(2,3)} = 1 + b^2 \pm b\sqrt{2}.$$

Матрица **Q** состоит из ортонормированных собственных векторов:

$$\boldsymbol{Q} = [\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3]$$

матрицы $\boldsymbol{R}_x$:

$$\boldsymbol{Q} = \begin{bmatrix} 1/2 & 1/\sqrt{2} & 1/2 \\ \sqrt{2}/2 & 0 & -\sqrt{2}/2 \\ 1/2 & -1/\sqrt{2} & 1/2 \end{bmatrix}.$$

ДПКЛ принимает вид

$$\mathbf{x}(n) = w_1(n)\mathbf{q}_1 + w_2(n)\mathbf{q}_2 + w_3\,\mathbf{q}_3,$$

где $M\{w_1^2\} = \lambda^{(1)}, M\{w_2^2\} = \lambda^{(2)}, M\{w_3^2\} = \lambda^{(3)}$.

4.3. Алгоритмы обработки пространственных сигналов на основе теории решеток

Дискретизация на основе решеток. Функция $\psi(\mathbf{x})$ является периодической с несингулярной периодической матрицей $\boldsymbol{V}$, если $\psi(\mathbf{x}) = \psi(\mathbf{x} + \boldsymbol{V}\mathbf{n}), \forall \mathbf{n} \in \mathbb{Z}^N$.

В том случае, если решетка применяется для периодической функции, ее обычно определяют как периодическую решетку, а единичная ячейка решетки определяет основной период функции.

Задавая непрерывный сигнал $s_c(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^N$, дискретный сигнал над решеткой Λ с генераторной матрицей $\boldsymbol{V}$ определяется как

$$s(\mathbf{x}) = \sum_{\mathbf{n} \in \mathbb{Z}^N} s_c(\mathbf{x})\delta(\mathbf{x} - \boldsymbol{V}\mathbf{n}), \mathbf{x} \in \mathbb{R}^N.$$

Преобразование Фурье сигналов с решетчатой дискретизацией. Преобразование Фурье сигнала $s(\mathbf{x})$ над решеткой $\boldsymbol{V}\mathbf{n}$ имеет вид

$$S(\mathbf{u}) = \sum_{\mathbf{x} \in \Lambda} s(\mathbf{x}) \exp(-2\pi j \mathbf{u}^T \mathbf{x}) = \sum_{\boldsymbol{n} \in \mathbb{Z}^N} s(\boldsymbol{V}\mathbf{n}) \exp(-2\pi j \mathbf{u}^T \boldsymbol{V}\mathbf{n}),$$

где $\mathbf{u} = [u_1, \ldots, u_N]^T$ – вектор координат в частотной области.

Пусть $\mathbf{r}$ будет вектор-точка в Λ^*. Спектр $S(\mathbf{u}+\mathbf{r})$ может быть записан в следующем виде:

$$S(\mathbf{u}+\mathbf{r}) = \sum_{n\in\mathbb{Z}^N} s(\boldsymbol{V}\mathbf{n})\exp(-2\pi j(\mathbf{u}+\mathbf{r})^T\boldsymbol{V}\mathbf{n}) =$$

$$= \sum_{\boldsymbol{n}\in\mathbb{Z}^N} s(\boldsymbol{V}\mathbf{n})\exp(-2\pi j(\mathbf{u})^T\boldsymbol{V}\mathbf{n})\exp(-2\pi j(\mathbf{r})^T\boldsymbol{V}\mathbf{n}).$$

Из определения взаимности следует, что $\mathbf{r}^T\mathbf{x}$ является для $\forall\mathbf{x}\in\Lambda,\ \mathbf{r}\in\ \Lambda^*$. Следовательно, Фурье-преобразование является периодическим с генераторной матрицей $\boldsymbol{W} = \boldsymbol{V}^{-T}$ для взаимной решетки Λ.

Обратное преобразование Фурье задается выражением

$$s(\mathbf{x}) = d(\Lambda)\int_{\boldsymbol{u}(\Lambda^*)} S(\mathbf{u})\exp(2\pi j\mathbf{u}^T\mathbf{x})\,d\mathbf{u}\,,\mathbf{x}\in\Lambda.$$

Дискретизация на смежных классах. Пусть Ψ представляет собой дискретное множество точек в $\mathbb{R}^N$, такое что оно является объединением смежных классов подрешеток Λ в решетке $\mathbf{\Gamma}$.

Можно записать

$$\Psi = \bigcup_{i=1}^{P}(c_i+\Lambda),$$

где $\mathbf{c}_i$ – вектор в $\mathbf{\Gamma}$.

Дискретный сигнал запишем как

$$s(\mathbf{u}) = \sum_{\mathbf{x}\in\Psi} s_c(\mathbf{x})\exp(-2\pi j\mathbf{u}^T\mathbf{x}).$$

Дискретное преобразование Фурье для $s(\mathbf{u})$ имеет вид

$$S(\mathbf{u}) = \sum_{i}^{P}\sum_{\boldsymbol{x}\in\Lambda} s(\mathbf{c}_i+\mathbf{x})\exp\left(-2\pi j\mathbf{u}^T(\mathbf{c}_i+\mathbf{x})\right) =$$

$$= \sum_{i=1}^{P}\exp(-2\pi j\mathbf{u}^T\mathbf{c}_i)\sum_{\mathbf{x}\in\Lambda} s(\mathbf{c}_i+\mathbf{x})\exp(-2\pi j\mathbf{u}^T\mathbf{x}) =$$

$$= \sum_{i=1}^{P}\exp(-2\pi j\mathbf{u}^T\mathbf{c}_i)S_i(\mathbf{x}),$$

периодичность которого определяет решетка $\mathbf{\Gamma}^*$.

Обобщенная теорема дискретизации. Запишем сигнал

$$S(\mathbf{u}) = \frac{1}{d(\Lambda)} \sum_{\mathbf{r} \in \Lambda^*} S_c(\mathbf{u} - \mathbf{r}),$$

где $S_c(\boldsymbol{u})$ – непрерывное преобразование Фурье сигнала $s_c(\mathbf{x})$.

Восстановление с малой погрешностью непрерывного исходного сигнала после его дискретизации возможно только в том случае, если область размещения $S_c(\mathbf{u})$ ограничена основным периодом взаимной решетки Λ^*.

Для восстановления используем реконструирующий фильтр, который осуществляет интерполяцию между отсчетами дискретного сигнала, с передаточной функцией

$$H_r(u) = \begin{cases} d(\Lambda); u \in \mathcal{U}(\Lambda^*), \\ 0; o.w. \end{cases}$$

Для исключения эффекта наложения применим антиэлайзинговый фильтр:

$$H_{AA}(u) = \begin{cases} 1; u \in \mathcal{U}(\Lambda^*), \\ 0; u \notin \mathcal{U}(\Lambda^*), \end{cases}$$

где $u \in \mathcal{U}(\Lambda^*)$ – должна иметь размерность как минимум $2 \times BW\big(S_c(\mathbf{u})\big)$; $BW\big(S_c(\mathbf{u})\big)$ – область размещения непрерывного сигнала.

4.3.1. Быстрый алгоритм поиска кратчайшего вектора

Обозначим скалярное произведение как $\langle \mathbf{x}, \mathbf{y} \rangle$. Пусть $\boldsymbol{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$ – набор линейно независимых векторов в $\mathbb{R}^n$. Тогда решетка с размерностью m – это набор линейных комбинаций $\mathbf{b}_i$ с целыми коэффициентами:

$$L = \mathbb{Z}\mathbf{b}_1 + \mathbb{Z}\mathbf{b}_2 + \ldots + \mathbb{Z}\mathbf{b}_m.$$

Базис решетки – это множество линейно независимых векторов, ее порождающих. Базис может состоять как из очень длинных векторов, так и из коротких (рис. 4.4). Найти короткий вектор – сложная задача.

Обозначим кратчайший вектор через $\lambda_1(L)$. Многомерное обобщение: $\lambda_k(L)$ – минимальное r, для которого размерность решетки внутри шара радиусом r больше либо равна k.

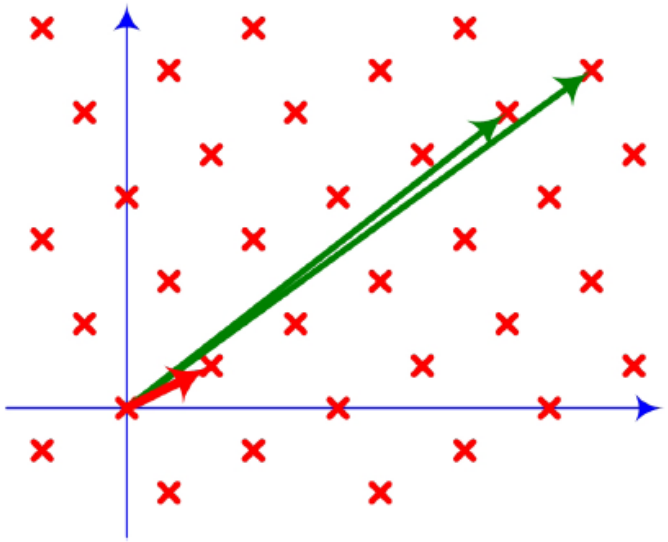


Рис. 4.4. Базисные векторы решетки

Фундаментальный параллелепипед образован базисом (рис. 4.5). Важный инвариант – его объем $\det(L)$, т. е. определитель матрицы, составленный из векторов базиса.

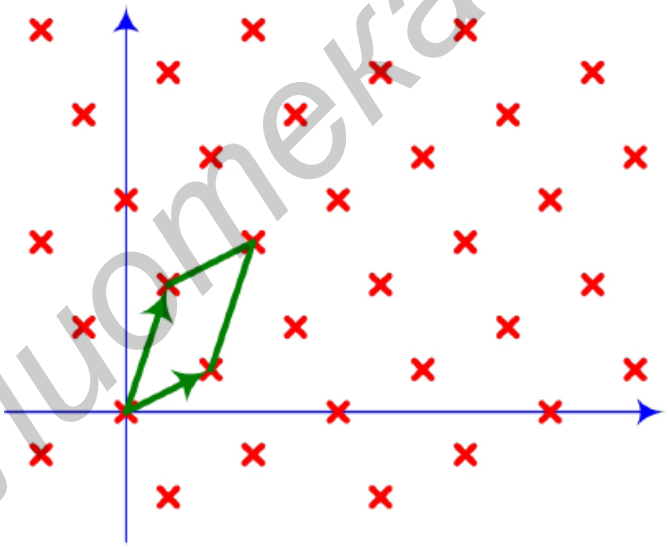


Рис. 4.5. Фундаментальный параллелепипед

Он от базиса не зависит, т. к. разные базисы решетки эквивалентны, только если они преобразуются друг в друга не меняющими $\det(L)$ преобразованиями.

Ортогонализация по методу Грама – Шмидта из исходного базиса (пространства) определяет ортогональный к исходному базис. При этом $\mathbf{b}_1^* = \mathbf{b}_1$ и каждый последующий базис ортогонален пространству предыдущих базисов. Для этого необходимо проводить вычитание проекции предыдущих векторов (рис. 4.6)

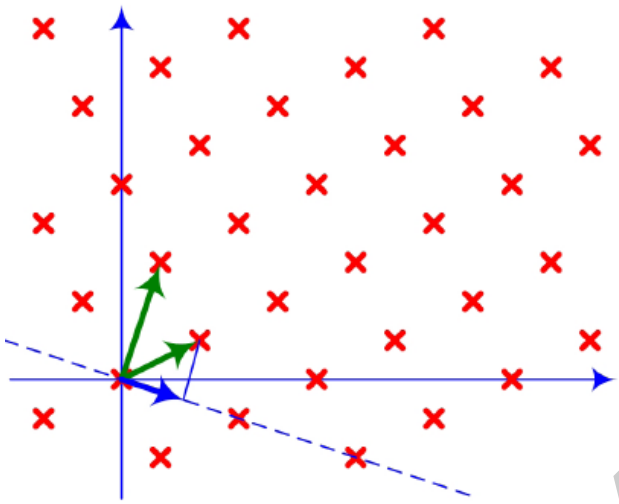


Рис. 4.6. Вычитание проекций предыдущих векторов

В частности определитель $\det(L)$ не поменяется. В результате получится ортогональный базис:

$$\det(L) = \|\mathbf{b}_1^*\|\|\mathbf{b}_2^*\| \dots \|\mathbf{b}_n^*\|.$$

Рассмотрим решетку L с базисом $\boldsymbol{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_m\}$. Определим по методу ортогонализации Грама – Шмидта векторы ортогонального базиса:

$$\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}, \quad 1 \le j < i \le n,$$

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*, \quad 1 \le i \le m.$$

Тогда $\langle \mathbf{b}_i^*, \mathbf{b}_j \rangle = 0$ для всех $j < i$ и $\langle \mathbf{b}_i^*, \mathbf{b}_i \rangle = \langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle$.

Матрицу ортонормированного базиса $\boldsymbol{B}$ можно представить в виде

$$\begin{bmatrix} \|\mathbf{b}_1^*\| & \mu_{21}\|\mathbf{b}_1^*\| & \mu_{3,1}\|\mathbf{b}_1^*\| & \cdots & \mu_{n,1}\|\mathbf{b}_1^*\| \\ 0 & \|\mathbf{b}_2^*\| & \mu_{3,2}\|\mathbf{b}_2^*\| & \cdots & \mu_{n,2}\|\mathbf{b}_2^*\| \\ 0 & 0 & \|\mathbf{b}_3^*\| & \cdots & \mu_{n,3}\|\mathbf{b}_3^*\| \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \|\mathbf{b}_n^*\| \end{bmatrix}.$$

Например, для решетки с базисом $\mathbf{b}_1 = (2, 1)$ и $\mathbf{b}_2 = (1, 3)$ имеем $\mathbf{b}_1^* = \mathbf{b}_1$ и

$$\mu_{2,1} = \frac{\langle \mathbf{b}_2, \mathbf{b}_1^* \rangle}{\langle \mathbf{b}_1^*, \mathbf{b}_1^* \rangle} = \frac{5}{5} = 1,$$

$$\mathbf{b}_2^* = \mathbf{b}_2 - \mu_{2,1}\mathbf{b}_1^* = \mathbf{b}_2 - \mathbf{b}_1^* = (-1, 2).$$

А для решетки с базисом $\mathbf{b}_1 = (9, 7)$ и $\mathbf{b_2} = (11, 9)$ (та же решетка) имеем $\mathbf{b}_1^* = \mathbf{b}_1$ и

$$\mu_{2,1} = \frac{\langle \mathbf{b}_2, \mathbf{b}_1^* \rangle}{\langle \mathbf{b}_1^*, \mathbf{b}_1^* \rangle} = \frac{155}{130} = \frac{31}{26},$$

$$\mathbf{b}_2^* = \mathbf{b}_2 - \mu_{2,1}\mathbf{b}_1^* = \mathbf{b}_2 - \frac{31}{26}\mathbf{b}_1^* = \left(\frac{7}{26}, -\frac{9}{26}\right).$$

Вектор $\mathbf{b}_2^*$ получился короче обоих исходных векторов

$$\|\mathbf{b}_2^*\| \approx 0{,}4385.$$

Задача поиска кратчайшего вектора. Рассмотрим базис B и произвольный вектор $\mathbf{x} \in \mathbb{Z}^n$ в L, который представляется как $B\mathbf{x}$. Пусть $j \in 1, \dots, n$ – наибольшее такое j, что $x_j \neq 0$. Тогда

$$\left|\langle B\mathbf{x}, \mathbf{b}_j^* \rangle\right| = \left|\langle \sum_{i=1}^{j} x_i \mathbf{b}_i, \mathbf{b}_j^* \rangle\right| = \left|x_j\right| \langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle = \left|x_j\right| \left\|\mathbf{b}_j^*\right\|^2.$$

С другой стороны, $\left|\langle B\mathbf{x}, \mathbf{b}_j^* \rangle\right| \leq \|B\mathbf{x}\| \cdot \left\|\mathbf{b}_j^*\right\|$ и справедливо

$$\|B\mathbf{x}\| \geq \left|x_j\right| \left\|\mathbf{b}_j^*\right\| \geq \left\|\mathbf{b}_j^*\right\| \geq \min_j \left\|\mathbf{b}_j^*\right\|.$$

Таким образом, нижняя оценка на размер кратчайшего вектора решетки равна

$$\lambda_1(L) \geq \min_j \left\|\mathbf{b}_j^*\right\|.$$

В рассмотренном примере кратчайший вектор был (2,1) с длиной $\sqrt{5}$, что больше $\|\mathbf{b}_2^*\|$.

Определение. Базис ***B*** называется δ-*LLL*-редуцированным, если

$$\left|\mu_{i,j}\right| \leq \frac{1}{2} \quad \text{для всех } 1 \leq j < i \leq n$$

и

$$\delta\|\mathbf{b}_i^*\|^2 \le \left\|\mu_{i+1,j}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*\right\|^2 \quad \text{для всех } 1 \le i < n.$$

Обычно $\delta = 3/4$.

Для последнего неравенства справедливы свойства

$$\delta\|\mathbf{b}_i^*\|^2 \le \left\|\mu_{i+1,j}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*\right\|^2 = \mu_{i+1,j}^2 \|\mathbf{b}_i^*\|^2 + \|\mathbf{b}_{i+1}^*\|^2$$

и

$$\|\mathbf{b}_i^*\|^2 \ge \left(\delta - \frac{1}{4}\right)\|\mathbf{b}_i^*\|^2.$$

Запишем базис $\boldsymbol{B}$ в ортонормированном виде, получаемом из $\boldsymbol{B}^*$ (i-й столбец $\mathbf{b}_i$ матрицы в ортонормированном базисе):

$$\begin{bmatrix} \|\mathbf{b}_1^*\| & * & * & \cdots & * \\ 0 & \|\mathbf{b}_2^*\| & * & \cdots & * \\ 0 & 0 & \|\mathbf{b}_3^*\| & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \|\mathbf{b}_n^*\| \end{bmatrix}.$$

Условие выполнения неравенства $\left(\left|\mu_{i,j}\right| \le 1/2\right)$ приводит к матрице вида

$$\begin{bmatrix} \|\mathbf{b}_1^*\| & \le \frac{1}{2}\|\mathbf{b}_1^*\| & \le \frac{1}{2}\|\mathbf{b}_1^*\| & \cdots & \le \frac{1}{2}\|\mathbf{b}_1^*\| \\ 0 & \|\mathbf{b}_2^*\| & \le \frac{1}{2}\|\mathbf{b}_2^*\| & \cdots & \le \frac{1}{2}\|\mathbf{b}_2^*\| \\ 0 & 0 & \|\mathbf{b}_3^*\| & \cdots & \le \frac{1}{2}\|\mathbf{b}_3^*\| \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \|\mathbf{b}_n^*\| \end{bmatrix}.$$

Заметим, что в подматрице

$$\begin{bmatrix} \|\mathbf{b}_i^*\| & \mu_{i+1,j}\mathbf{b}_i^* \\ 0 & \|\mathbf{b}_{i+1}^*\| \end{bmatrix}$$

второй столбец почти такой же длины, как первый

$$\delta\|\mathbf{b}_i^*\|^2 \le \mu_{i+1,j}^2\|\mathbf{b}_i^*\|^2 + \|\mathbf{b}_{i+1}^*\|^2.$$

В общем случае такое требование обобщается на подматрицы размером $(k \times k)$.

Если δ-*LLL*-редуцированный базис найден, то

$$\|\mathbf{b}_n^*\|^2 \geq \left(\delta - \frac{1}{4}\right)\|\mathbf{b}_{n-1}^*\|^2 \geq \cdots \geq \left(\delta - \frac{1}{4}\right)^{n-1}\|\mathbf{b}_1^*\|^2 = \left(\delta - \frac{1}{4}\right)^{n-1}\|\mathbf{b}_1\|^2.$$

Следовательно, для любого i имеем

$$\|\mathbf{b}_1\| \leq \left(\delta - \frac{1}{4}\right)^{-\frac{i-1}{2}}\|\mathbf{b}_1^*\| \quad \leq \left(\delta - \frac{1}{4}\right)^{-\frac{n-1}{2}}\|\mathbf{b}_1^*\|$$

и

$$\|\mathbf{b}_1\| \leq \left(\delta - \frac{1}{4}\right)^{-\frac{n-1}{2}}\min_i\|\mathbf{b}_1^*\| \leq \left(\delta - \frac{1}{4}\right)^{-\frac{n-1}{2}}\lambda_1(L).$$

Например, для $\delta = 3/4$

$$\|\mathbf{b}_1\| \leq \left(2^{(n-1)/2}\right)\|\mathbf{x}\|.$$

Таким образом, задача поиска кратчайшего вектора решается с точностью до $\left(\delta - \frac{1}{4}\right)^{-\frac{n-1}{2}}$, если найден δ-*LLL*-редуцированный базис.

Задача, связанная с вычислением δ-*LLL*-редуцированного базиса, решается с помощью алгоритм L^3 или *LLL*: Lenstra, Lenstra, Lovasz .

Алгоритм:

1. Вычислить $\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*$.
2. Редукция. Для всех $2 \leq i \leq n$ и всех $i - 1 \geq j \geq 1$

$$\mathbf{b}_i := \mathbf{b}_i - c_{i,j}\mathbf{b}_j, \quad \text{где} \quad c_{i,j} = \left\lceil \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \right\rfloor.$$

3. Обмен значениями. Если для какого-то i

$$\delta\|\mathbf{b}_i^*\|^2 > \|\mu_{i+1}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*\|^2,$$

то поменять местами $\mathbf{b}_i$ с $\mathbf{b}_{i+1}$ и вернуться к шагу 1.

4. Выдать $\mathbf{b}_1, \ldots, \mathbf{b}_n$.

Если обмен не произошел, то второе требование выполняется. А редукция обеспечивает первое условие: она вычитает столбец за столбцом так, чтобы оставалось $\leq 1/2\,\|\mathbf{b}_i^*\|$.

Рассмотрим действие алгоритма по шагам. Перед началом редукции имеем

$$\begin{bmatrix} \|\mathbf{b}_1^*\| & * & * & \cdots & * \\ 0 & \|\mathbf{b}_2^*\| & * & \cdots & * \\ 0 & 0 & \|\mathbf{b}_3^*\| & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \|\mathbf{b}_n^*\| \end{bmatrix}.$$

Первый шаг: $i = 2,\ j = 1$:

$$\begin{bmatrix} \|\mathbf{b}_1^*\| & \leq \frac{1}{2}\|\mathbf{b}_1^*\| & * & \cdots & * \\ 0 & \|\mathbf{b}_2^*\| & * & \cdots & * \\ 0 & 0 & \|\mathbf{b}_3^*\| & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \|\mathbf{b}_n^*\| \end{bmatrix}.$$

Второй шаг: $i = 3,\ j = 2$:

$$\begin{bmatrix} \|\mathbf{b}_1^*\| & \leq \frac{1}{2}\|\mathbf{b}_1^*\| & * & \cdots & * \\ 0 & \|\mathbf{b}_2^*\| & \leq \frac{1}{2}\|\mathbf{b}_2^*\| & \cdots & * \\ 0 & 0 & \|\mathbf{b}_3^*\| & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \|\mathbf{b}_n^*\| \end{bmatrix}.$$

Третий шаг: $i = 3,\ j = 1$:

$$\begin{bmatrix} \|\mathbf{b}_1^*\| & \leq \frac{1}{2}\|\mathbf{b}_1^*\| & \leq \frac{1}{2}\|\mathbf{b}_1^*\| & \cdots & * \\ 0 & \|\mathbf{b}_2^*\| & \leq \frac{1}{2}\|\mathbf{b}_2^*\| & \cdots & * \\ 0 & 0 & \|\mathbf{b}_3^*\| & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \|\mathbf{b}_n^*\| \end{bmatrix}.$$

И так далее. После всех шагов

$$\begin{bmatrix} \|\mathbf{b}_1^*\| & \le \frac{1}{2}\|\mathbf{b}_1^*\| & \le \frac{1}{2}\|\mathbf{b}_1^*\| & \cdots & \le \frac{1}{2}\|\mathbf{b}_1^*\| \\ 0 & \|\mathbf{b}_2^*\| & \le \frac{1}{2}\|\mathbf{b}_2^*\| & \cdots & \le \frac{1}{2}\|\mathbf{b}_2^*\| \\ 0 & 0 & \|\mathbf{b}_3^*\| & \cdots & \le \frac{1}{2}\|\mathbf{b}_3^*\| \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \|\mathbf{b}_n^*\| \end{bmatrix}.$$

Если обмен не произошел, то второе требование выполняется. А редукция обеспечивает первое условие: она вычитает столбец за столбцом так, чтобы оставалось $\le 1/2\,\|\mathbf{b}_i^*\|$. После редукции имеем $|\mu_{i,j}| \le 1/2$.

Покажем, что *LLL* – полиномиальный алгоритм. Рассмотрим потенциал базиса решетки. Для базиса $\boldsymbol{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_m\}$ под потенциалом $\mathcal{D}_B$ будем понимать произведение

$$\mathcal{D}_B = \prod_{i=1}^{n} \mathcal{D}_{B,j} = \prod_{i=1}^{n} \det(L_i) = \prod_{i=1}^{n} \|\mathbf{b}_1^*\|\|\mathbf{b}_2^*\| \dots \|\mathbf{b}_i^*\|,$$

где L_i – решетка, порождаемая $\{\mathbf{b}_1, \dots, \mathbf{b}_i\}$.

Потенциал можно оценить как

$$\mathcal{D}_B \le \left(\max_i \|\mathbf{b}_i\|\right)^{\frac{n(n-1)}{2}},$$

поскольку $[\![\mathbf{b}_i^*]\!] \le \|\mathbf{b}_i\|$.

От редукции $\mathcal{D}_B$ не убывает, т. к. $\mathbf{b}_i^*$ не меняются. Когда $\mathbf{b}_i$ и $\mathbf{b}_{i+1}$ меняются местами, из всех определителей $\det(L_k)$ меняется только определитель $\det(L_i)$ и отношение

$$\frac{\mathcal{D}_B'}{\mathcal{D}_B} = \frac{\det(L_i')}{\det(L_i)} = \frac{\det(L)\,(\mathbf{b}_1, \dots, \mathbf{b}_{i-1}, \mathbf{b}_{i+1})}{\det(L)\,(\mathbf{b}_1, \dots, \mathbf{b}_i)} = \frac{\left\|\mu_{i+1,j}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*\right\|}{\left\|\mathbf{b}_i^*\right\|} \le \sqrt{\delta}.$$

Следовательно, число итераций может быть не больше

$$\log_{\frac{1}{\sqrt{\delta}}} \mathcal{D}_B \le \frac{1}{\log\frac{1}{\sqrt{\delta}}} \frac{n(n-1)}{2} \log\left(\max_i \|\mathbf{b}_i\|\right).$$

Таким образом, *LLL*-алгоритм с параметром δ позволяет решить задачу поиска кратчайшего вектора с точностью $(\delta - 1/4)^{-(n-1)/2}$. Например, задавая $\beta = 1/4 + (3/4)^{n/(n-1)}$, получим точность $\left(2/\sqrt{3}\right)^n$.

Одно из важных условий применения алгоритма – поиск корней многочленов. Рассмотрим целое число N и унитарный многочлен $f \in \mathbb{Z}_N[x]$ степени d. Требуется при помощи *LLL*-алгоритма найти все корни f в $\mathbb{Z}_N[x]$ по модулю, не превосходящие $B = N^{1/d}$. Примем, что $B = cN^{2/[d(d+1)]}$.

Рассмотрим многочлен

$$f(x) = x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + a_0.$$

Если все коэффициенты многочлена сильно ограничены, то можно искать просто обычные корни, что является достаточно простой задачей.

Таким образом, необходимо найти такой многочлен $g(x)$, который имел бы те же корни, что и $f(x)$, но при этом имел достаточно малые значения коэффициентов или короткий вектор коэффициентов. Для решения этой задачи можно использовать структуру решетки.

Рассмотрим множество $Z = \{N, Nx, Nx^2, \ldots, Nx^{d-1}, f(x)\}$. Все корни $f(x)$ также являются корнями любой их линейной комбинации по модулю N.

Определим решетку из столбцов матрицы

$$L = \begin{bmatrix} N & 0 & 0 & \cdots & 0 & a_0 \\ 0 & BN & 0 & \cdots & 0 & Ba_1 \\ 0 & 0 & B^2N & \cdots & 0 & B^2a_2 \\ \vdots & \vdots & & & \vdots & \\ 0 & 0 & 0 & \cdots & B^{d-1}N & B^{d-1}a_{d-1} \\ & & & & 0 & B^d \end{bmatrix}.$$

На решетке L при помощи *LLL*-алгоритма найдем короткий вектор. Определитель решетки равен

$$\det L = N \cdot BN \cdot \ldots \cdot B^{d-1}N \cdot B^d = N^d B^{\frac{d(d+1)}{2}}.$$

После применения *LLL*-алгоритма найдем такой v, что

$$\|v\| \le O\left(\lambda_1(L)\right) \le O\left((\det L)^{\frac{1}{d+1}}\right) = O\left(N\frac{B^{d/2}}{N^{1/(d+1)}}\right),$$

где константы под O зависят только от размерности d решетки.

Для $B < c_1(\text{в})N^{\frac{2}{d(d+1)}}$ получаем оценку меньше, чем $N/(d+1)$.

Теорема Айтая. Определим для любого натурального $n \in N$ класс случайных решеток L^n, порождаемых с полиномиальной временной сложностью. Предположим, что существует полиномиальный по временной сложности алгоритм, такой что для любой случайно выбранной решетки $L \in L^n$, найдется нетривиальный вектор $\boldsymbol{v}$, длина которого не превосходит n. Значит, существует вероятностный полиномиальный по временной сложности алгоритм B, который для любой решетки $L \in R^n$ и некоторых констант c_0, c_1, c_2 с высокой вероятностью способен решить любую из следующих задач:

1) по базису решетки найти кратчайший ненулевой вектор (SVP; поиск кратчайшего вектора), с точностью $\gamma = n^{c_2}$;

2) приближенный поиск кратчайших линейно независимых векторов (SIVP) с точностью $\gamma = n^{c_0}$;

3) приближенный поиск кратчайшего базиса (SBP) с точностью $\gamma = n^{c_1}$.

Этой теоремой Айтай установил связь между сложностью в худшем и среднем случаях вышеперечисленных задач.

Каждая из задач теории решеток в зависимости от параметров (например, точности решения) принадлежит к некоторому классу временной/емкостной сложности для детерминированной и недетерминированной машин Тьюринга. При анализе задач внимание концентрируется именно на временной сложности, (собственно емкостная сложность заведомо не превышает временную). Задачи допускают взаимные редукции, что упрощает их исследование, например:

$$\mathrm{SBP}_\gamma^p \leq_p \mathrm{SIVP}_\gamma^p(n), \ldots, \mathrm{uSVP}_\gamma \leq \mathrm{BDD}_{1/\gamma} \leq \mathrm{uSVP}_{2/\boldsymbol{\gamma}},$$

$$\mathrm{GapCVP}_\gamma^p \equiv_p \mathrm{CVP}_\gamma^p, \mathrm{SVP}_\gamma^2 \leq_p \mathrm{CVP}_\gamma^2.$$

Конструкция Айтая. Ортогонализация по методу Грама – Шмидта: из базиса (пространства) делают ортогональный базис. $\lambda_k(L)$ – минимальное r, для которого размерность решетки внутри шара радиусом r больше либо равна k. Важный инвариант – его объем $\det(L)$, т. е. определитель матрицы, составленный из векторов базиса.

Если разные базисы эквивалентны, то объем решетки не зависит от базиса. В алгоритме ортогонализации $\mathbf{b}_1^* = \mathbf{b}_1$ и каждый последующий вектор ортогонален пространству предыдущих векторов, при этом определитель $\det(L)$ не меняется:

$$\det L = \|\mathbf{b}_1^*\| \, \|\mathbf{b}_2^*\| \ldots \|\mathbf{b}_\mathbf{n}^*\|.$$

Задача unique-SVP – поиск кратчайшего вектора $\boldsymbol{v}$ в решетке с размерностью n при условии, что нет непараллельных ему векторов длиной меньше $n^8\|v\|$. Рассмотрим m векторов $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}_q^n$. Определим векторную функцию $f_{\mathbf{a}_1 \ldots \mathbf{a}_m}: \{0,1\}^{n \log q}$ как

$$f_{\mathbf{a}_1 \dots \mathbf{a}_m}(\mathbf{b}_1, \dots, \mathbf{b}_m) = \sum_{i=1}^{m} \mathbf{b}_i \mathbf{a}_i (\mathrm{mod}\, q).$$

Семейство *хеш-функций* – это набор функций:

$$\mathcal{F} = \{f_{\mathbf{a}_1 \dots \mathbf{a}_m} | \mathbf{a}_1, \dots, \mathbf{a}_m \in \mathbb{Z}_q^n\}.$$

Чтобы семейство было стойким, нужно, чтобы для случайной функции $f \in \mathcal{F}$ никакой полиномиальный алгоритм не мог бы со значительной вероятностью найти коллизию, т. е. такие $x \neq y$, что $f(x) = f(y)$. Если такой алгоритм существует, то используя его как оракула можно найти кратчайший вектор в любой решетке. Идея сведения – рассмотреть решетку как распределение вероятностей и добавлять в нее нормально распределенный шум. Если добавить достаточно много шума, решетка станет неотличима от равномерного распределения (рис. 4.7).

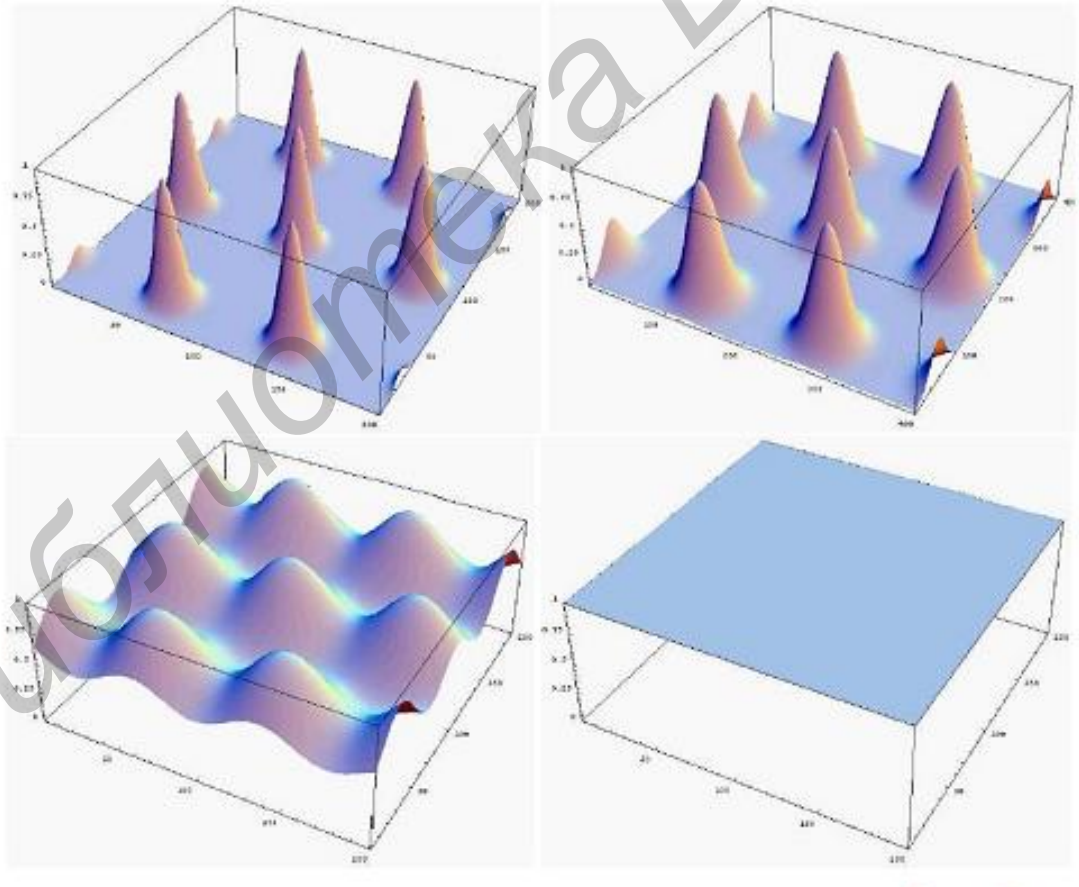


Рис. 4.7. Шум в решетке

4.4. Алгоритмы обработки сигналов на основе теории графов

Фильтры на графах. Операцию фильтрации на графе определим с помощью векторно-матричного произведения. Система $\boldsymbol{H} \in \mathbb{C}^{N\times N}$ выполняет операцию линейной фильтрации на графе, если для входного воздействия $\mathbf{s} \in \mathcal{S}$ возвращает на свой выход $\boldsymbol{Hs}$ и удовлетворяет требованиям, предъявляемым к линейным системам. Для линейной комбинации входных сигналов на выходе фильтра формируется сумма линейных комбинаций выходов для отдельных сигналов

$$\boldsymbol{H}(\alpha\mathbf{s}_1 + \beta\mathbf{s}_2) = \alpha\boldsymbol{H}\mathbf{s}_1 + \beta\boldsymbol{H}\mathbf{s}_2.$$

Инвариантность к сдвигу. Будем считать, что фильтр на графе обладает свойством инвариантности к сдвигу, если применение сдвига на графе к его выходу эквивалентно применению сдвига на графе к его входу:

$$\boldsymbol{A}(\boldsymbol{H}\mathbf{s}) = \boldsymbol{H}(\boldsymbol{A}\mathbf{s}).$$

Следующая теорема утверждает, что все линейные, инвариантные к сдвигу фильтры на графе задаются полиномами в сдвигах $\boldsymbol{A}$.

Пусть $\boldsymbol{A}$ будет матрицей смежности графа и предполагается, что ее характеристический и минимальный полиномы равны:

$$p_{\boldsymbol{A}}(x) = m_{\boldsymbol{A}}(x).$$

Тогда фильтр $\boldsymbol{H}$ на графе G относится к линейной и инвариантной к сдвигу системе, если и только если $\boldsymbol{H}$ есть полином сдвига $\boldsymbol{A}$ на графе. Иными словами, существует полином

$$h(x) = h_0 + h_1 x + \cdots + h_L x^L$$

с возможно комплексными коэффициентами $h_l \in \mathbb{C}$, такими что

$$\boldsymbol{H} = h(\boldsymbol{A}) = h_0\mathbf{I} + h_1\boldsymbol{A} + \cdots + h_L\boldsymbol{A}^L.$$

Доказательство. Так как условие инвариантности к сдвигу сохраняется для всех сигнальных графов $\mathbf{s} \in \mathcal{S} = \mathbb{C}^N$, то матрицы $\boldsymbol{A}$ и $\boldsymbol{H}$ коммутативны: $\boldsymbol{AH} = \boldsymbol{HA}$. Поскольку $p_{\boldsymbol{A}}(x) = m_{\boldsymbol{A}}(x)$, то все собственные значения матрицы $\boldsymbol{A}$ имеют точно одни ассоциированные с ними собственные векторы. Следовательно, граф матрицы $\boldsymbol{H}$ коммутативен с матрицей сдвига $\boldsymbol{A}$ тогда и только тогда, когда существует полином в $\boldsymbol{A}$.

Свойства фильтров на графах. Пусть $\boldsymbol{A}$ **и** $\widetilde{\boldsymbol{A}}$ – матрицы сдвига. Фильтры $h(\boldsymbol{A})$ и $g(\widetilde{\boldsymbol{A}})$ эквивалентны, если для всех входных сигналов $\mathbf{s} \in \mathcal{S}$ они имеют равные выходы: $h(\boldsymbol{A})\mathbf{s} = g(\widetilde{\boldsymbol{A}})\mathbf{s}$.

Любой фильтр графа имеет единственный эквивалентный фильтр с не менее $\deg\{m_{\boldsymbol{A}}(x)\} = N_{\boldsymbol{A}}$ элементами сдвига на данном графе

Все линейные, инвариантные к сдвигу фильтры на графе $G = (V, \boldsymbol{A})$ формируют векторное пространство

$$\mathcal{F} = \left\{ \boldsymbol{H}: \boldsymbol{H} = \sum_{l=0}^{N_A-1} h_l \boldsymbol{A}^l \,|h_l \in \mathbb{C} \right\}.$$

Действия, связанные со сложением и перемножением фильтров в $\mathcal{F}$, позволяют получить новые фильтры, эквивалентные фильтрам в $\mathcal{F}$. Таким образом, $\mathcal{F}$ является замкнутым относительно такого вида операций и имеет структуру алгебры.

Фильтр $\boldsymbol{H} = h(\boldsymbol{A}) \in \mathcal{F}$ является инвертируемым, если полином $h(x)$ удовлетворяет условию $h(\lambda_m) \neq 0$ для всех различных собственных значений $\lambda_0, \lambda_1, \ldots, \lambda_{M-1}$ матрицы $\boldsymbol{A}$. Следовательно, существует единственный полином $g(x)$ степени $\deg\{g(x)\} < N_{\boldsymbol{A}}$, который удовлетворяет условию

$$h(\boldsymbol{A})^{-1} = g(\boldsymbol{A}) \in \mathcal{F}.$$

Коэффициенты $h_0, \ldots, h_{N_A-1}$ фильтра $h(\boldsymbol{A})$ однозначно определяют его импульсную характеристику $\mathbf{u}$. С другой стороны, импульсная характеристика фильтра однозначно определяет коэффициенты фильтра, причем $\mathrm{rank}\widehat{\boldsymbol{A}} = N_{\boldsymbol{A}}$, где $\widehat{\boldsymbol{A}} = (\boldsymbol{A}^0\delta, \ldots, \boldsymbol{A}^{N_A-1}\delta)$.

Заметим, что изменение порядка следования узлов графа $v_0, \ldots, v_{N-1}$ не меняет импульсную характеристику. Если $\boldsymbol{P}$ – перестановочная матрица, то единичный импульс равен $\boldsymbol{P}\delta$, матрица смежности имеет вид $\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^T$, коэффициенты фильтра принимают форму $h(\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^T) = \boldsymbol{P}h(\boldsymbol{A})\mathbf{P}^{\mathrm{T}}$. Следовательно, форма импульсной характеристики просто переупорядочивается в соответствии с перестановкой $\boldsymbol{P}h(\boldsymbol{A})\boldsymbol{P}^T\boldsymbol{P}\delta = \boldsymbol{P}\mathbf{u}$.

4.4.1. Быстрые алгоритмы сегментации графов

Задача сегментации графов на непересекающиеся подмножества вершин актуальна во многих областях современной науки и техники.

Разбиение графа используется при сегментации изображений для их последующего анализа и интеллектуальной обработки. Широкое применение задача сегментации нашла также в социальных, информационных и биохимических сетях, где часто требуется выявлять в структуре сети сильно связанные сообщества.

Для улучшения маршрутизации и разбиения на широковещательные домены сеть разделяется на отдельные непересекающиеся подмножества элементов (компьютеров или роутеров), причем количество связей внутри этих подмножеств должно превышать количество внешних связей с другими подмножествами. Оптимальность разбиения зависит от выбора алгоритма сегментации. Особый интерес представляют комбинаторные алгоритмы сегментации структур компьютерных сетей, т. к. они обеспечивают более сбалансированное разбиение и меньшее информационное взаимодействие

полученных подсетей по сравнению с другими алгоритмами и используют информацию о связности вершин в графе.

Постановку задачи сегментации графов можно сформулировать следующим образом. Представим компьютерную сеть в виде неориентированного взвешенного связанного графа $G = (V, E)$, каждой вершине $v \in V$ и каждому ребру $e \in E$ которого приписан вес. Задача оптимального разделения графа состоит в разбиении его вершин на непересекающиеся подмножества с максимально близкими суммарными весами вершин и минимальным суммарным весом ребер, проходящих между полученными подмножествами вершин. Следует отметить возможную противоречивость указанных критериев разбиения графа: равновесность подмножеств вершин может не соответствовать минимальности весов граничных ребер и наоборот. В большинстве случаев необходимым является выбор того или иного компромиссного решения.

Задача разбиения графа относится к классу *NP*-полных, верхняя оценка числа разбиений определяется числом Белла:

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k.$$

Рассмотрим следующие комбинаторные алгоритмы сегментации структур компьютерных сетей:

– алгоритм рекурсивного деления пополам;

– «жадный» алгоритм разбиения графов;

– спектральный алгоритм Ньюмана;

– алгоритм Гирвана – Ньюмана.

Алгоритм рекурсивного деления пополам относится к числу самых простых и наименее трудоемких алгоритмов сегментации. В его основе лежит метод бинарного деления, при котором на первой итерации граф разделяется на две равные части, на втором шаге каждая из полученных частей также разбивается на две части, аналогичное разбиение производится и на всех последующих итерациях. Применительно к данному алгоритму сначала находится диаметр графа, определяемый как наибольшее расстояние между вершинами графа. Расстояние между двумя вершинами графа определяется как наименьшее число граней, которые нужно пройти от узла i к узлу j. Начиная с одной из вершин, половина вершин присваивается одной из подобластей, а половина – другой подобласти. Затем рекурсивная процедура применяется к каждой из подобластей. Время работы алгоритма оценивается как $O\,(N)$, где N – число вершин графа или маршрутизаторов в сети.

Для нахождения диаметра графа, начиная с исходной вершины v_s, принятой за потенциальный корень, каждой соседней вершине присваивается метка 1, а соседям соседей присваивается метка 2. Последняя из помеченных вершин имеет метку m, которая обозначает расстояние от корневой вершины.

Далее следующая вершина v_i обозначается как исходная, процесс повторяется. Вершина v_i с максимальным значением m считается корнем, при этом m является диаметром графа. Реализация алгоритма нахождения диаметра графа также требует $O\ (N)$ операций.

Пример работы алгоритма рекурсивного деления пополам для определения диаметра показан на рис. 4.8–4.10. Разобьем исходный граф (см. рис. 4.8) на p = 3 части. Диаметр графа будет равняться m = 3, вершина v_1 будет являться корнем графа.

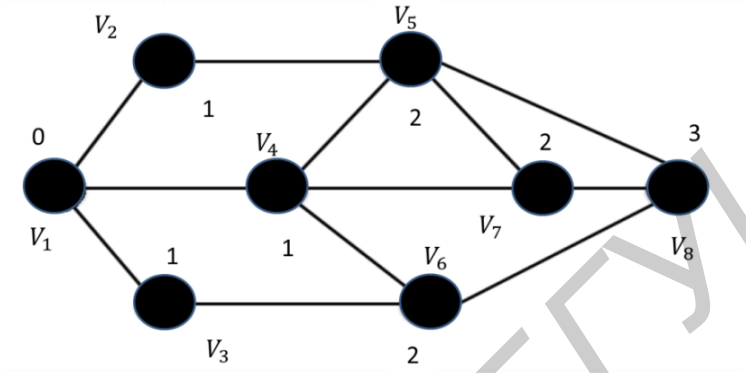


Рис. 4.8. Алгоритм рекурсивного деления пополам

Вершина v_4 с меткой 1 будет являться границей раздела исходного графа на 2 части (см. рис. 4.9). Произведем разбиение графа (см. рис. 4.9). Далее перейдем к первому подграфу и применим к нему процедуру разбиения. Диаметр подграфа будет равен m = 2, вершина v_2 будет являться корнем подграфа. В соответствии с вычисленными метками вершин данный подграф разделится еще на 2 области: в первой области будут расположены вершины v_1 и v_2, во второй – вершины v_3 и v_4. В результате работы алгоритма исходный граф разобьется на 3 подграфа (см. рис. 4.10).

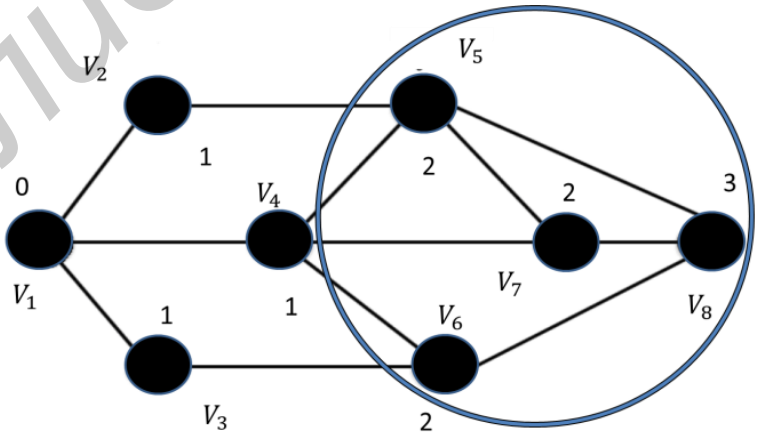


Рис. 4.9. Разбиение графа на 2 части

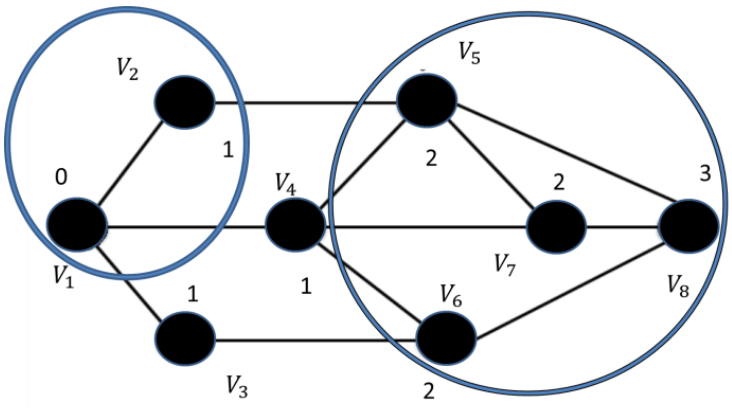


Рис. 4.10. Результат разбиения исходного графа на 3 части

Однако алгоритм рекурсивного деления пополам имеет существенный недостаток: в нем мало учитывается связность вершин и получаемое в итоге разбиение оказывается неоптимальным.

«Жадный» алгоритм разбиения заключается в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным [6].

Процедура начинается с вершины, имеющей наименьшую степень. Маркируются все соседние вершины, а затем вершины, соседние с соседями. Первые N/p маркированных вершин присваиваются одной подобласти, и процедура применяется к оставшейся части графа до тех пор, пока все вершины не станут маркированными. Данный алгоритм имеет схожие черты с алгоритмом рекурсивного деления пополам, хотя и не является таковым. Трудоемкость алгоритма оценивается как $O\ (N)$ [5].

Пример работы «жадного» алгоритма разбиения показан на рис. 4.11. Разобьем исходный граф на p = 3 части. Разбиение будем начинать с вершины v_3, т. к. она имеет самую низкую степень, равную 2.

Произведем маркировку всех вершин графа и выделим 8/3 = 2 вершины с наименьшими пометками в отдельное подмножество (см. рис. 4.11).

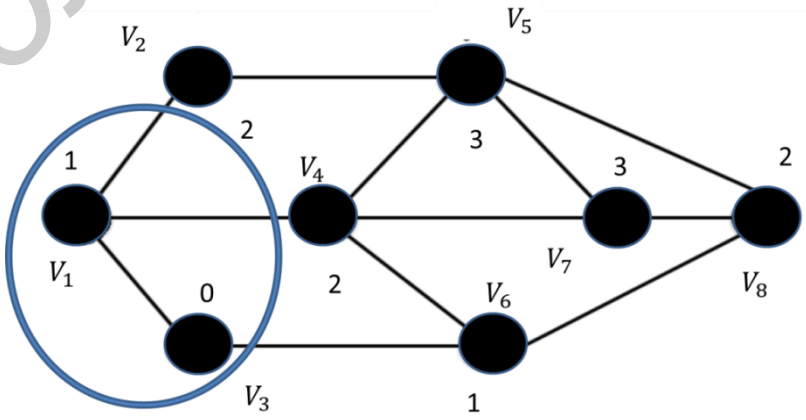


Рис. 4.11. Разбиение графа на 2 части с помощью «жадного» алгоритма

Повторим процедуру для оставшегося подграфа. Разбиение начнем с вершины v_2 с самой низкой степенью, равной 2. Произведем маркировку всех вершин графа и выделим 6/2 = 3 вершины с наименьшими пометками в отдельное подмножество. Результат разбиения графа показан на рис. 4.12.

Преимуществом данного алгоритма является простота и высокая скорость работы, однако итоговое разбиение графа получается неоптимальным, что является весомым недостатком при сегментации компьютерных сетей.

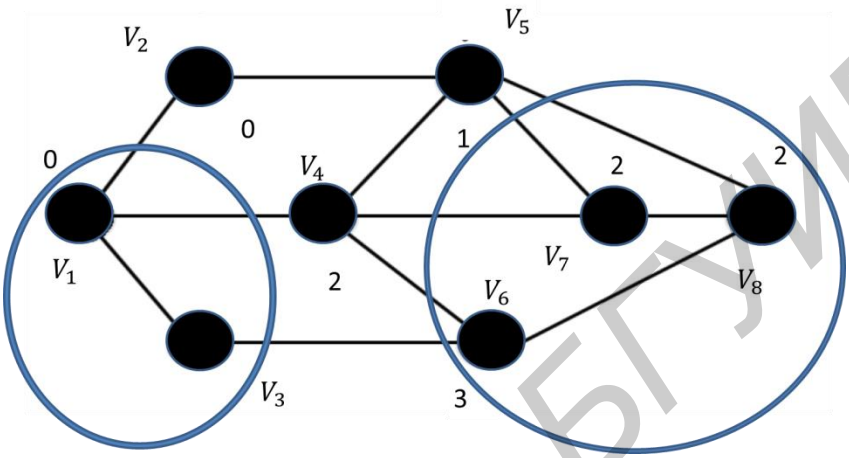


Рис. 4.12. Результат разбиения исходного графа на 3 части

Спектральный алгоритм Ньюмана, основанный на использовании собственных векторов и собственных значений, производит разбиение исходного графа, минимизировав при этом количество внешних ребер.

В алгоритме Ньюмана размер разреза определяется как

$$R = 0{,}5 \sum_{i \in \gamma_1, j \in \gamma_2} A_{i,j}.$$

Если ввести индексный вектор $s_i \in \{-1; 1\}$, то

$$R = 0{,}5 \sum_{i,j} (1 - s_i s_j) A_{i,j}.$$

Далее если k_i – мощность вершины i, $\delta_{ij} = 1$, если $i = j$, и $\delta_{ij} = 0$ – в противном случае, то

$$R = 0{,}5 \sum_{i,j} s_i s_j (k_i \delta_{i,j} - A_{i,j}).$$

В матричной форме

$$R = \frac{1}{4}\boldsymbol{s}^T L\boldsymbol{s}.$$

Матрица Лапласа представляется в виде

$$\boldsymbol{L} = \boldsymbol{A} - \boldsymbol{D},$$

где $\boldsymbol{A}$ – матрица смежности графа; $\boldsymbol{D}$ – диагональная матрица, состоящая из степеней вершин графа.

Матрица $\boldsymbol{L}$ является положительно полуопределенной и имеет наименьшее собственное значение $\lambda_1 = 0$ и соответствующий ему собственный вектор $\boldsymbol{e}_1 = \{1, 1, \ldots, 1\}$. Вектор $\mathbf{e}_1$ не является решением задачи, поскольку найденное решение означает, что все вершины при разбиении попадут в одну группу. Для связного графа следующее наименьшее собственное значение матрицы Лапласа λ_2 является положительным.

Собственный вектор $\mathbf{e}_2$, связанный с собственным значением λ_2, является искомым решением задачи минимизации. Этот собственный вектор (*вектор Фидлера*) состоит из весовых множителей, связанных с вершинами графа, что позволяет применить метод деления пополам, разделяя вершины по различным подобластям в соответствии с их весовыми множителями. Собственный вектор матрицы Лапласа, соответствующий ее наименьшему нетривиальному собственному значению, определяет связность графа и его протяженность, а также выступает в качестве метрики.

Для нахождения собственного вектора, связанного со вторым наименьшим собственным значением, используются алгоритм Ланцоша или метод сопряженных градиентов.

На рис. 4.13 изображен исходный граф, который необходимо разбить на 2 части.

Для данного графа матрицы $\boldsymbol{A}$, $\boldsymbol{D}$ и $\boldsymbol{L}$ выглядят следующим образом:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix},$$

$$\boldsymbol{L} = \begin{bmatrix} 2 & 0 & 1 & 0 & 1 & 0 \\ 0 & -3 & 0 & 1 & 1 & 1 \\ 1 & 0 & -3 & 0 & 1 & 1 \\ 0 & 1 & 0 & -2 & 0 & 1 \\ 1 & 1 & 1 & 0 & -3 & 0 \\ 0 & 1 & 1 & 1 & 0 & -3 \end{bmatrix}.$$

Вектор Фидлера будет иметь вид $\boldsymbol{e}_2 = [2 - 1\ 1 - 2\ 1 - 1]$. Итоговое разбиение графа изображено на рис. 4.14. Величина разреза, т. е. количество внешних ребер примет минимальное значение и будет равно $R = 2$.

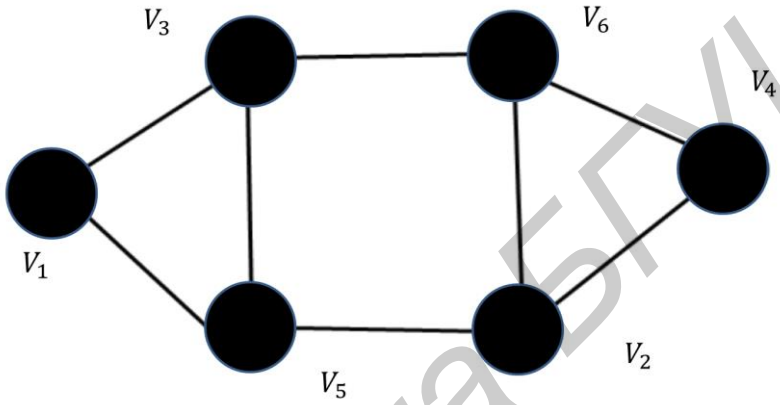


Рис. 4.13. Исходный граф для разбиения

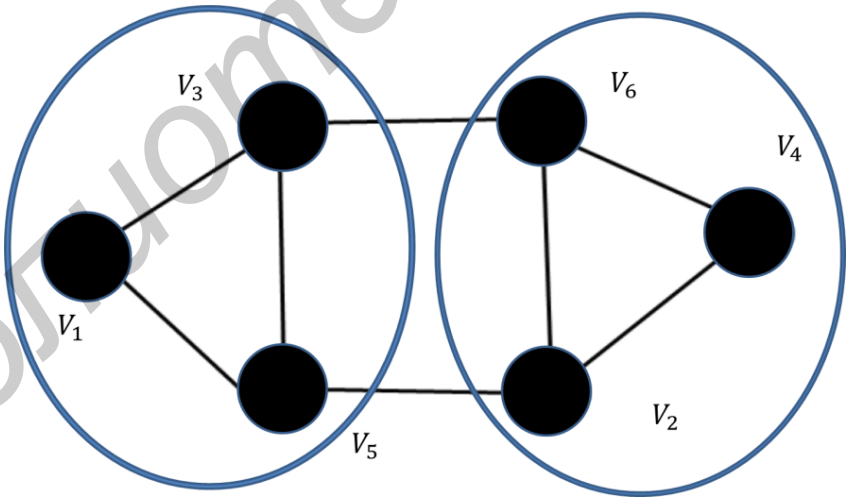


Рис. 4.14. Итоговое разбиение графа

В отличие от ранее рассмотренных алгоритмов спектральный алгоритм Ньюмана приводит к связным подобластям при условии, что исходный граф также является связным, при этом число ребер графа, разрезаемых при сегментации области, оказывается примерно в два раза меньшим по сравнению

с остальными алгоритмами. Трудоемкость нахождения собственных векторов оценивается как $O\ (N\ log\ N)$.

Алгоритм Гирвана – Ньюмана основывается на понятии центральности вершин и производит разбиение графа в соответствии с данной величиной. Центральность – мера важности вершины в графе. Для определения данной величины рассчитывается смежность ребер графа – мера важности, пропорциональная числу кратчайших путей от всех вершин до остальных, проходящих через ребро:

$$C_{B(V)} = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(V)}{\sigma_{st}}.$$

Рассмотрим основные шаги алгоритма Гирвана – Ньюмана.

Шаг 1. Вычислить смежность для всех ребер графа. Вычисление можно произвести по алгоритму Ньюмана с трудоемкостью $O(|V||E|^2)$ или по алгоритму Джонсона с трудоемкостью $O(V^2 \log V + VE))$.

Шаг 2. Удалить ребро с наивысшим показателем.

Шаг 3. Пересчитать величину смежности с учетом изменений.

Шаг 4. Выполнять шаги 2 и 3 до тех пор, пока ребра не закончатся.

Алгоритм Гирвана – Ньюмана выполняет иерархическую кластеризацию дивизионным методом. Результаты разбиения структуры сети при помощи данного алгоритма показаны на рис. 4.15. Однако существенным недостатком данного способа разбиения является высокая вычислительная сложность.

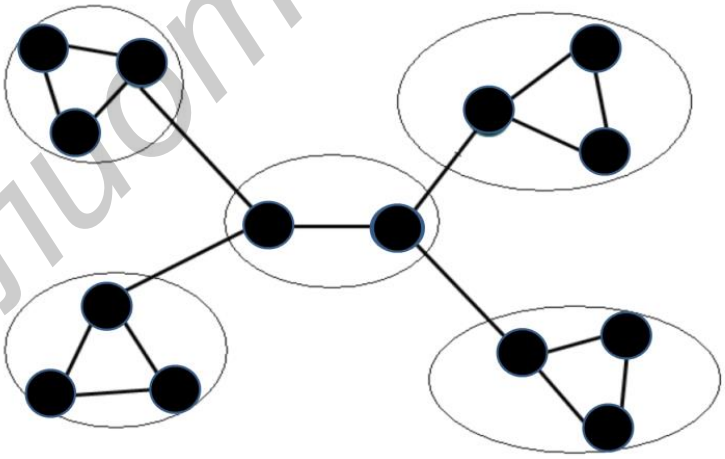


Рис. 4.15. Разбиение графа алгоритмом Гирвана – Ньюмана

Существует еще несколько комбинаторных алгоритмов сегментации графов, в том числе алгоритм Кернигана – Лина, алгоритм распространения меток, алгоритм k-кластеризации. Однако все эти алгоритмы редко применяются в сетях.

Для сравнения представленных алгоритмов сегментации по критерию оптимальности итогового разбиения будем использовать меру модулярности.

Мера модулярности – показатель связности сообщества в виде отношения количества связей внутри сообщества ко внешним связям:

$$Q = \frac{1}{4m}\sum_{i,j}\left[A_{i,j} - \frac{k_i k_j}{2m}\right](s_i s_j + 1),$$

где $m = |E|$, $A_{i,j}$ – элемент матрицы смежности; s_j – показатель вхождения в сообщество $(+1/-1)$; k_i – мощность вершины i.

Смысл данной меры – разница между долей ребер, входящих в первое или во второе сообщество, и ожидаемой долей ребер в первом либо втором сообществе для случайного графа с распределением мощностей вершин по заданному закону.

Алгоритм рекурсивного деления пополам и «жадный» алгоритм имеют низкую вычислительную сложность, однако итоговое разбиение оказывается неоптимальным, что доказывает низкая мера модулярности. Алгоритм Гирвана – Ньюмана обладает слишком высокой трудоемкостью. Исходя из этого оптимальным среди рассмотренных является спектральный алгоритм Ньюмана, т. к. он осуществляет сегментацию исходного графа на подмножества с самой высокой мерой модулярности. Его трудоемкость составляет $O((k-1)\,N\,log\,N)$, где k – число подмножеств, на которые разбивается исходный граф.

Контрольные вопросы

1. Вычислите спектры последовательностей **x**=[1, 2, –1]^T и **h**=[1, –1, 1, 1]^T с помощью алгоритмов различных спектральных преобразований. Проведите сравнительный анализ результатов вычислений.
2. Запишите матрицы прямых и обратных дискретных преобразований Фурье, Чебышева, Фибоначчи. Как данные преобразования соотносятся с операторами сдвигов?
3. Сопоставьте вычислительную сложность преобразований Фурье, Чебышева, Фибоначчи.
4. Проведите сравнительный анализ алгоритмов БПФ в полиномиальных и обычных представлениях.
5. Поясните, как структура решетки связана с дискретной группой.
6. Как можно использовать теорию решеток для обработки и анализа данных?
7. Приведите пример построения квантователя сигналов с использованием теории решеток.
8. В чем суть ортогонализации Грамма – Шмидта?

5. АНАЛИЗ ТРАФИКА ИНФОКОММУНИКАЦИОННОЙ СИСТЕМЫ

5.1. Выделение трендовых и периодических составляющих временного ряда

SSA (Singular Spectrum Analysis, или анализ сингулярного спектра) – метод анализа временных рядов, основанный на преобразовании одномерного временного ряда в многомерный ряд с последующим применением к полученному многомерному временному ряду *метода главных компонент*.

Метод SSA используется для исследования временных рядов, транспортных потоков – везде, где естественно предположить существование тренда и/или периодическое поведение. Разработанные методы идентификации могут быть применены для автоматизации исследования таких рядов [20 – 26].

Рассмотрим общую задачу исследования структуры вещественнозначного временного ряда $(f_0, \ldots, f_{N-1})$ длиной N. Здесь под структурой понимаются некоторые характеристики или свойства ряда, которые сохраняются с течением времени. Например, может стоять задача нахождения тренда или выделения периодических составляющих. Как правило, для нахождения структуры какого-либо явления необходима повторяемость этого явления (например, повторные выборки в статистике для нахождения характеристик генерального распределения). Если же мы имеем дело с временным рядом, который обычно существует в единственном экземпляре, то повторяемости в общем случае нет. В качестве способа выхода из этой ситуации в стандартных методах анализа временных рядов часто предполагается справедливость параметрической модели ряда (например, ряд состоит из линейного тренда и белого шума) или же стационарность ряда (т. е. априори предполагается повторяемость на уровне первых двух моментов). Однако эти ограничения нередко оказываются слишком жесткими.

Важным является вопрос: как получить повторяемость результатов обработки, не накладывая на ряд предварительных ограничений? Определим множество отрезков временного ряда, имеющих достаточно большую длину L (назовем ее длиной окна). Будем рассматривать эти отрезки последовательно с первой по L-ю точку, со второй по L+1-ю и т. д. Рассмотренные отрезки (назовем их векторами L-вложения) будут *наследовать* свойства ряда. Если ряд содержит тренд (здесь под трендом мы понимаем медленно меняющуюся составляющую ряда), то и векторы вложения будут его содержать; если исходный ряд содержит периодическую составляющую, то такими же будут и векторы вложения. Таким образом, мы приходим к идее исследования всей совокупности векторов вложения для выявления их общей структуры.

Составим из векторов вложения так называемую траекторную матрицу с размерностью $L \times K$, где $K = N - L + 1$ есть число векторов вложения. Она будет иметь вид

$$X = \begin{bmatrix} f_0 & f_1 & \cdots & f_{K-1} \\ f_1 & f_2 & \cdots & f_k \\ \vdots & \vdots & \ddots & \vdots \\ f_{L-1} & f_L & \cdots & f_{N-1} \end{bmatrix} = [X_1, \ldots, X_K]. \quad (5.1)$$

Теперь у нас есть повторяемость и мы можем попытаться увидеть структуру векторов вложения. Воспользуемся следующим приемом: разложим всю траекторную матрицу на элементарные части (сумму элементарных матриц), в некотором смысле независимые (ортогональные) и упорядоченные по их вкладу в разложение. Если разложение окажется «удачным», то мы сможем сгруппировать эти элементарные матрицы так, чтобы, например, одна группа соответствовала трендовой составляющей ряда, другая группа – циклической и т. д. Затем просуммируем матрицы внутри каждой группы и вернемся от разложения матриц к разложению ряда на тренд и циклическую составляющую.

Способом разложения траекторной матрицы, который оказывается очень хорошо согласованным с описанной выше задачей, является сингулярное разложение.

Техника сингулярного спектрального разложения. Рассмотрим временной ряд $\mathbf{x} = [x[1], x[2], \ldots, x[n]]^T$, заданный в виде вектора **x**. Образуем траекторную матрицу Ганкеля вида

$$\mathbf{\Gamma} = [\gamma_{i,j}] = \begin{bmatrix} x[1] & x[2] & \cdots & x[m] \\ x[2] & x[3] & \cdots & x[m+1] \\ \vdots & \vdots & \ddots & \vdots \\ x[n-m+1] & x[n-2+2] & \cdots & x[n] \end{bmatrix},$$

где $\gamma_{i,j} = x[i + j - 1]$.

Сингулярная декомпозиция позволяет представить траекторную матрицу в виде

$$\mathbf{\Gamma} = \boldsymbol{USV}^T,$$

где $\boldsymbol{U}$ – вещественная ортогональная матрица размером $(n - m + 1)\times(n - m + 1)$; $\boldsymbol{V}$ – вещественная ортогональная матрица размером $m \times m$; $\boldsymbol{S}$ – диагональная вещественная матрица размером $(n - m + 1)\times m$, диагональные элементы которой представляют собой сингулярные значения матрицы **Γ**.

Матрицу $\boldsymbol{C}$ оценок ковариационных значений можно записать как

$$\boldsymbol{C} = \mathbf{\Gamma}^T\mathbf{\Gamma}.$$

Если $\boldsymbol{C}$ – симметричная, полуопределенная матрица положительных значений, то существует единственное спектральное разложение

$$\boldsymbol{C} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{T},$$

где $\boldsymbol{\Phi}$ – вещественная ортонормальная матрица, столбцы которой представляют собой собственные вектора матрицы $\boldsymbol{C}$; $\boldsymbol{\Lambda}$ – вещественная диагональная матрица, вдоль главной диагонали которой располагаются в порядке убывания элементы σ_i^2, равные собственным значениям матрицы $\boldsymbol{C}$.

Можно заметить, что

$$\boldsymbol{C} = \boldsymbol{\Gamma}^{T}\boldsymbol{\Gamma} = (\boldsymbol{USV}^{T})^{\boldsymbol{T}}(\boldsymbol{USV}^{T}) = \boldsymbol{SV}^{T}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{USV}^{T},$$

где $\boldsymbol{U}$ – является ортонормальной матрицей; $\boldsymbol{S}$ – диагональная матрица.

Таким образом,

$$\boldsymbol{C} = \boldsymbol{VS}^{2}\boldsymbol{V}^{T}.$$

Сравнивая приведенные выше уравнения, получаем

$$\boldsymbol{\Phi} = \boldsymbol{V}, \;\; \boldsymbol{\Lambda} = \boldsymbol{S}^{2}.$$

Заметим, что если использовать статистические аналогии и рассматривать набор векторов вложения как выборку, то сингулярное разложение с точностью до центрирования эквивалентно анализу главных компонент этой многомерной выборки.

Метод анализа временных рядов, основанный на описанном выше приеме, известен под названием SSA (Singular Spectrum Analysis). Он возник из анализа хаотического поведения ряда и аттракторов. Сингулярный спектр здесь – это набор собственных чисел сингулярного разложения траекторной матрицы, понимаемый как сингулярный спектр соответствующего матрице оператора. Метод носит название «гусеница» из-за скользящей процедуры нарезания векторов вложения из исходного ряда (подобно движению гусеницы) и возник из статистических аналогий с методом главных компонент.

Кроме описанных выше идей к методу SSA приходили со стороны анализа рядов, управляемых линейными рекуррентными формулами. Эти ряды обладают следующей замечательной особенностью: в сингулярном разложении их траекторной матрицы оказывается только небольшое число ненулевых компонент, причем это число (размерность ряда) не зависит от длины окна L (если L и N достаточно большие). Структура траекторной матрицы порождается рядами конечной размерности, а точнее, структура пространства, порожденного столбцами траекторной матрицы, т. е. векторами вложения.

На основе свойств разложения и теоретического аппарата метода «Гусеница»-SSA были разработаны методы идентификации трендовых и гармонических (возможно, модулированных) компонент. Методы позволяют

автоматизировать процесс идентификации, а также предоставляют дополнительную информацию, которая может быть использована при интерактивной идентификации в анализе рядов со сложной структурой.

SSA-алгоритм. Рассмотрим вещественнозначный временной ряд длиной N $F_N = (f_0, \ldots, f_{N-1})$.

Алгоритм можно разбить на четыре шага: вложение, сингулярное разложение, группировка и диагональное усреднение. Первые два в совокупности называются разложением, последние – восстановлением. Основным параметром алгоритма является так называемая длина окна L, $1 < L < N$. Результатом алгоритма является разбиение временного ряда на аддитивные составляющие.

Сингулярное разложение. Состоит в формировании из ряда траекторной матрицы $\boldsymbol{X}$ размером $L \times K$, где $K = N - L + 1$. Далее проводится сингулярное разложение матрицы X:

$$\boldsymbol{X} = \boldsymbol{X}_1 + \boldsymbol{X}_2 + \ldots + \boldsymbol{X}_d, \ \boldsymbol{X}_i = \sqrt{\lambda_i}\boldsymbol{U}_i\boldsymbol{V}_i^T, \tag{5.2}$$

где $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d > 0$ – упорядоченные ненулевые собственные числа матрицы $\boldsymbol{X}\boldsymbol{X}^T$; $\{\boldsymbol{U}_i\}_{i=1}^d : \boldsymbol{U}_i \in \mathbb{R}^L$ – множество векторов факторизации траекторной матрицы.

Соответствующие этим собственным числам собственные векторы

$$\{\boldsymbol{V}_i\}_{i=1}^d : \boldsymbol{V}_i = \sqrt{\lambda_i}^{-1}\boldsymbol{X}^T\boldsymbol{U}_i \in \mathbb{R}^K$$

будем называть факторными векторами.

Группировка. На этом шаге проводится группировка компонент разложения. Разбив $\{1, \ldots, d\}$ на m непересекающихся подмножеств I_j, получим

$$\boldsymbol{X} = \boldsymbol{X}_{I_1} + \boldsymbol{X}_{I_2} + \ldots + \boldsymbol{X}_{I_m}, \qquad \boldsymbol{X}_{I_j} = \sum_{k \in I_j} \boldsymbol{X}_k. \tag{5.3}$$

Последним шагом является восстановление рядов F_N по сгруппированным матрицам $\boldsymbol{X}_{I_j}$.

Самым неформализуемым шагом является шаг группировки. Вся информация о каждой из компонент $\boldsymbol{X}_i$ содержится в собственном числе λ_i, а также в i-х собственном $\boldsymbol{U}_i$ и факторном $\boldsymbol{V}_i$ векторах.

Собственный и факторный векторы называют сингулярными векторами, а совокупность $(\lambda_i, \boldsymbol{U}_i, \boldsymbol{V}_i)$ – собственной тройкой. Поиск компонент для требуемой группировки, главным образом на основе анализа собственных троек, будем называть процедурой идентификации.

Целью метода является разложение ряда на трендовую составляющую, периодические (циклические) составляющие и шум.

Заметим, что метод может решать также задачу разбиения ряда на низкочастотную и высокочастотную составляющие.

Рассмотрим случай, когда выделение тренда и периодик возможно. Для ответа на вопрос о возможности построения разложения было введено понятие разделимости. Два ряда – $F_N^{(1)}$ и $F_N^{(2)}$ являются (слабо) разделимыми при длине окна L, если для их суммы $F_N = F_N^{(1)} + F_N^{(2)}$ существует сингулярное разложение (неединственность разложения может быть вызвана совпадающими собственными числами), для которого может быть выбрана группировка с $m = 2$.

Понятие такой точной разделимости накладывает слишком жесткие условия на ряды $F_N^{(1)}$ и $F_N^{(2)}$. От полиномиального ряда, например, никакой другой ряд не отделяется.

При наличии шума (который можно понимать как сумму большого числа гармоник с разными периодами и близкими небольшими амплитудами) точная разделимость вообще оказывается неосуществима. Поэтому более реальным оказывается понятие *приближенной разделимости*. Приближенная разделимость чаще всего появляется из асимптотической по N ($N \to \infty$) разделимости при фиксированной длине ряда N.

Отметим два наиболее важных следствия асимптотической разделимости при $L \to \infty, K = N - L + 1 \to \infty$:

1) любой ряд, являющийся суммой экспонент и полиномов, асимптотически отделим от периодического ряда;

2) любой гармонический ряд с частотой ω_1 асимптотически отделим от другого гармонического ряда с частотой $\omega_2 = \omega_1$; это же верно и для экспоненциально-модулированных гармоник.

Эти два результата дают возможность, в частности, приближенно отделять тренд (понимаемый как медленно меняющаяся аддитивная составляющая ряда) от периодических компонент и от шума, а также разделять периодические компоненты на их гармонические составляющие.

Качество разделимости определяется величиной $\min(L, K)$. В соответствии с этим длина окна должна быть достаточно большой, но не больше, чем половина длины ряда. Заметим также, что кратность длины окна L периоду улучшает качество отделимости соответствующей периодической компоненты.

Группировка ряда. Разделимость дает нам возможность выделить аддитивную составляющую. Но для того, чтобы проводить идентификацию компонент, нам необходимо знать, сколько компонент соответствует искомой составляющей и какими свойствами они должны обладать.

Для ответа на этот вопрос вводится понятие рядов конечного ранга. К таким рядам относится любой ряд, являющийся суммой произведений полиномов, экспонент и гармоник. Ряд конечного ранга обладает тем свойством, что число ненулевых компонент сингулярного разложения его траекторной матрицы конечно, причем оно не зависит от L (при достаточно больших L и N).

Примеры рядов конечного ранга

1. *Экспоненциально-модулированный гармонический ряд* с общим членом

$$f_n = A\exp(\alpha n)\cos(2\pi\omega n + \varphi), A = 0, \omega \in (0;\ 0{,}5].$$

Ранг этого ряда зависит от частоты ω.

Если $\omega \in (0;\ 0{,}5)$, то такой ряд имеет ранг 2. Оба собственных (факторных) вектора тоже имеют экспоненциально-модулированный гармонический вид с теми же экспоненциальным показателем и частотой, как у исходного ряда.

Если $L\omega \in \mathbb{Z}\,(K\omega \in \mathbb{Z}), \alpha = 0$ (случай обычной гармонической составляющей), то фазы сингулярных векторов различаются точно на π/2 (если один записать в виде синуса, то другой запишется косинусом от того же аргумента).

Если ω = 0,5, то ряд имеет ранг 1. Собственный (факторный) вектор имеет вид исходного ряда с теми же α и ω, а именно вид модулированной гармоники с периодом 2.

2. *Экспоненциальный ряд* с общим членом, заданным в виде

$$f_n = A\exp(\alpha n),$$

имеет размерность 1. Собственный (факторный) вектор тоже имеют экспоненциальный вид с таким же экспоненциальным показателем, как у исходного ряда.

3. *Полиномиальный ряд*. Перечисленные выше результаты приводят к тому, что, во-первых, при выделении тренда нужно сгруппировать те компоненты, чьи сингулярные вектора медленно меняются. Их количество неизвестно, т. к. в реальных временных рядах тренд скорее всего только аппроксимируется некоторой суммой экспонент, полиномов и/или гармоник с большими периодами. Во-вторых, при выделении периодической компоненты с периодом T надо искать пары компонент сингулярного разложения, соответствующие периодам P, таким что T/P – целое. Каждая пара отличается тем, что ее изображение на двумерной диаграмме имеет вид «кружка» или «спирали». Исключение составляет случай $P = 2$, которому соответствует единственная собственная тройка с пилообразными сингулярными векторами.

Использование собственных чисел. Рассмотрим, какую информацию несут в себе собственные числа и как ими пользоваться при исследовании ряда и идентификации.

Во-первых, отношение

$$\sum_{k\in I}\lambda_k \Big/ \sum_{i=1}^{d}\lambda_i$$

отражает вклад составляющей, восстановленной по группе компонент I в разложение исходного ряда, $I \subset \{1, \dots, d\}$. Так как мы упорядочили компоненты в порядке убывания собственных чисел, компонента с меньшим номером вносит больший вклад в вид ряда, а незначительные компоненты имеют большие номера. Этим можно пользоваться, в частности, для ограничения количества исследуемых компонент.

Известно, что при целых $L\omega$ и $K\omega$ собственные числа обоих компонент, соответствующих гармонической составляющей ряда, будут равны между собой. Для экспоненциально-модулированного гармонического ряда с невозрастающей амплитудой это свойство будет асимптотическим при $\min(L, K) \to \infty$. Поэтому компоненты, соответствующие гармоническим составляющим, будут соседними, а на графике собственных чисел им будет соответствовать «ступенька».

5.2. Анализ сингулярного спектра сетевого трафика

Способ преобразования одномерного ряда в многомерный представляет собой «свертку» временного ряда в матрицу, содержащую фрагменты временного ряда, полученные с некоторым сдвигом. Заметим, что длина фрагмента называется длиной «гусеницы», а величина сдвига одного фрагмента относительно другого – шагом «гусеницы». Обычно используется шаг 1.

Рассмотрим временной ряд $\{x_i\}_{i=1}^{N}$, образованный последовательностью N равноотстоящих значений некоторой функции. Базовый алгоритм метода «гусеница» можно разбить на четыре этапа.

Этап 1. *Развертка одномерного ряда в многомерный*

Выберем некоторое число $L<N$, называемое *длиной «гусеницы»*, и представим первые L значений последовательности $\{x_i\}_{i=1}^{N}$ в качестве первой строки матрицы $\boldsymbol{X}$. В качестве второй строки матрицы берем значения последовательности с x_2 по x_{L+1}.

Последней строкой с номером $k = N - L + 1$ будут последние L элементов последовательности. Матрицу

$$\boldsymbol{X} = [\boldsymbol{X}_1, \dots, \boldsymbol{X}_K] = \left[x_{ij}\right]_{i,j=1}^{L,K}$$

можно рассматривать как L-мерную выборку объема K или L-мерный временной ряд, которому соответствует L-мерная траектория (ломаная в L-мерном пространстве из $K-1$ звена). $\boldsymbol{X}_i = (x_i, \dots, x_{i+L-1})^T$, $(1 \le i \le K)$ – вектора вложения длины L.

Далее по обычной схеме (за исключением стандартизации признаков) проводится анализ главных компонент (АГК).

Этап 2. С*ингулярное разложение выборочной ковариационной матрицы* Производится сингулярное разложение (SVD) траекторной матрицы $\boldsymbol{X}$. Положим $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^T$ и обозначим $\lambda_1, \dots, \lambda_L$ – собственные числа $\boldsymbol{S}$, взятые в невозрастающем порядке $\lambda_1 \geq \cdots \geq \lambda_L \geq 0$, и $\boldsymbol{U}_1, \dots, \boldsymbol{U}_L$ – ортонормированная система собственных векторов матрицы $\boldsymbol{S}$, соответствующих собственным числам.

Положим $d = \mathrm{rank}\boldsymbol{X} = \max\{i: \lambda_i > 0\}$ (заметим, что в реальности, как правило, $d = L$) и

$$V_i = \boldsymbol{X}^T \boldsymbol{U}_i / \sqrt{\lambda_i}, \quad (i = 1, \dots d).$$

В этих обозначениях сингулярное разложение траекторной матрицы $\boldsymbol{X}$ может быть записано как

$$\boldsymbol{X} = \boldsymbol{X}_1 + \cdots + \boldsymbol{X}_d,$$

где матрицы $\boldsymbol{X}_i = \sqrt{\lambda_i}\boldsymbol{U}_i\boldsymbol{V}_i^T$ имеют ранг 1 и называются элементарными матрицами.

Набор ($\boldsymbol{X}_i = \sqrt{\lambda_i}\boldsymbol{U}_i\boldsymbol{V}_i^T$) называется i-й собственной тройкой (коротко – ET от eigentriple) сингулярного разложения. Векторы $\boldsymbol{U}_i$ и $\boldsymbol{V}_i$ называются левыми и правыми соответственно сингулярными векторами матрицы **X**, числа $\sqrt{\lambda_i}$ – сингулярные числа (они составляют сингулярный спектр X, что и дало название Singular Spectrum Analysis методу), векторы $\sqrt{\lambda_i}\boldsymbol{V}_i = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{U}_{\boldsymbol{i}}$ по аналогии с анализом главных векторов называются векторами главных компонент.

Этап 3. Группировка собственных троек (*отбор главных компонент*)

Множество всех индексов $\{1, \dots, d\}$ разбивается на m непересекающихся подмножеств $I_1, \dots, I_m$.

Пусть I=$\{i_1, \dots i_p\}$. Тогда результирующая матрица $\boldsymbol{X}_I$, соответствующая группе I, определяется как $\boldsymbol{X}_I = \boldsymbol{X}_{i1} + \cdots + \boldsymbol{X}_{ip}$. Результирующие матрицы вычисляются по группам I=$\{I_1, \dots I_m\}$ и сгруппированное SVD-разложение матрицы $\boldsymbol{X}$ может быть записано как

$$\boldsymbol{X} = \boldsymbol{X}_{I_1} + \cdots + \boldsymbol{X}_{I_m}.$$

Этап 4. *Восстановление одномерного ряда*

Каждая матрица $\boldsymbol{X}_{Ij}$ сгруппированного разложения ганкелизуется (усредняется по антидиагоналям) и затем полученная ганкелева матрица трансформируется в новый временной ряд длиной N на основе взаимно однозначного соответствия между ганкелевыми матрицами и временными рядами. Диагональное усреднение, примененное к каждой результирующей матрице $\boldsymbol{X}_{Ik}$, производит восстановленные ряды $\tilde{\boldsymbol{X}}^{(k)} = (\tilde{x}_1^{(k)}, \dots, \tilde{x}_N^{(k)})$. Таким

образом, исходный ряд $x_1, \ldots, x_N$ раскладывается в сумму m восстановленных рядов:

$$x_n = \sum_{k=1}^{m} \tilde{x}_n^{(k)} \ (n = 1,2,..,N).$$

Данное разложение является главным результатом алгоритма SSA для анализа временного ряда. Это разложение имеет смысл, если каждая из его компонент может быть интерпретируема как тренд либо колебания (периодики), либо шум.

Программа алгоритма SSA в среде Matlab запишется следующим образом.

```
function [y,r,vr]=ssa(x1, L)
% x1 – вектор-столбец исходного временного ряда
% L – длина окна
% y – восстановленный временной ряд
% r – временной ряд остатков r=x1-y
% vr – нормирующий коэффициент для приближенной траекторной матрицы
% Программа возвращает сингулярный спектр временного ряда
% Шаг1: построение траекторной матрицы
  N=length(x1);
  if L>N/2; L=N-L; end
 K=N-L+1;
  X=zeros(L,K);
 for i=1:K
   X(1:L,i)=x1(i:L+i-1); end
% Шаг 2: сингулярное SVD-преобразование
  S=X*X';
 [U,autoval]=eig(S); % вычисление собственных значений
[d,i]=sort(-diag(autoval));  % сортировка собственных значений
 d=-d;
 U=U(:,i);sev=sum(d); % вычисление суммы
% Визуализация результатов
plot((d./sev)*100),
hold on,
plot((d./sev)*100,'rx');
title ('Сингулярный спектр'); xlabel ('Номер собственного значения);
ylabel('Собственные значения (% Norm of trajectory matrix retained)')
  V=(X')*U;
  rc=U*V';
```

```
% Шаг 3: группировка
  I=input ('Выбор аргументов компонент реконструкции временного ряда в форме I=[i1,i2:ik,...,iL]  ')
  Vt=V';
  rca=U(:,I)*Vt(I,:);
% Шаг 4: реконструкция
  y=zeros(N,1);
  Lp=min(L,K);
  Kp=max(L,K);
  for k=0:Lp-2
    for m=1:k+1;
     y(k+1)=y(k+1)+(1/(k+1))*rca(m,k-m+2);
    end
  end
  for k=Lp-1:Kp-1
    for m=1:Lp;
     y(k+1)=y(k+1)+(1/(Lp))*rca(m,k-m+2);
    end
  end
  for k=Kp:N
     for m=k-Kp+2:N-Kp+1;
      y(k+1)=y(k+1)+(1/(N-k))*rca(m,k-m+2);
    end
  end
%Визуализация результатов
  figure;
subplot(2,1,1); hold on;
xlabel ('Data poit'); ylabel ('Исходный и восстановленный ряды');
 plot(x1); grid on;
 plot(y,'r')
  r=x1-y;
  subplot(2,1,2); plot(r,'g');
xlabel('Data poit');ylabel('Ряд ошибок восстановления'); grid on
  vr=(sum(d(I))/sev)*100;
```

Был проведен анализ сингулярного спектра модели временной реализации трафика сети в соответствии с алгоритмом SSA. Результаты анализа представлены на рис. 5.1 и 5.2. Визуальный анализ графиков позволяет оценить трафик как самоподобный процесс.

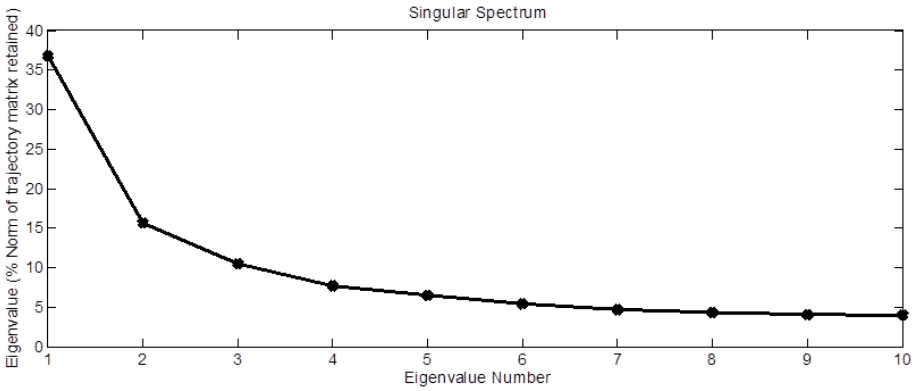
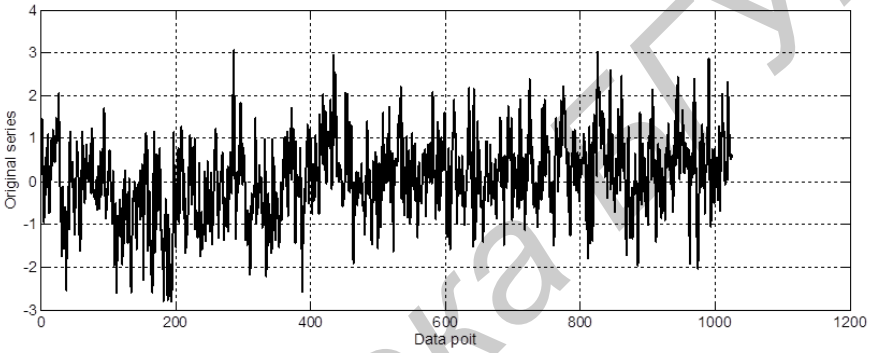
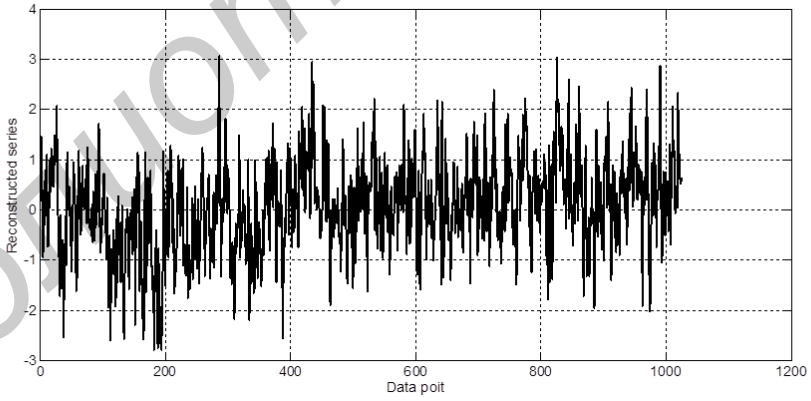


Рис. 5.1. Сингулярный спектр временной реализации модели трафика в виде фрактального гауссовского шума с показателем Херста H=0,8 при длине «гусеницы» L=10, I=10



а



б

Рис. 5.2. Исходный и восстановленный временные ряды модели трафика в виде фрактального гауссовского шума с показателем Херста H=0,8 при длине «гусеницы» L=10, I=10, ошибка реконструкции не превышает 10^{-15}:
а – диаграмма исходного ряда; б – диаграмма восстановленного ряда

На рис. 5.3 и 5.4 показаны результаты анализа временного ряда данных, полученных с помощью генератора случайных чисел rand(1, 1024) с равномерным распределением чисел в интервале 0,1.

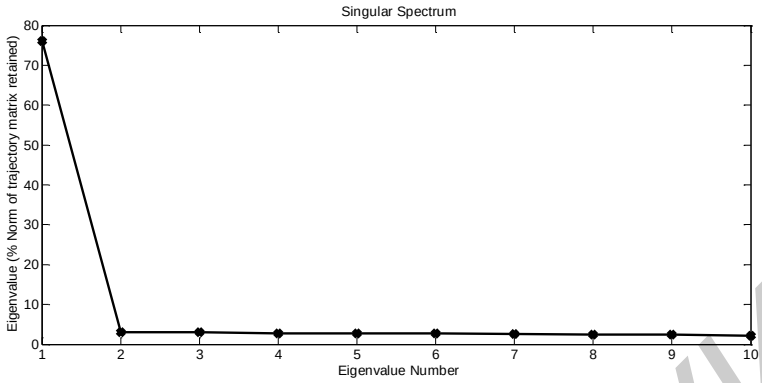
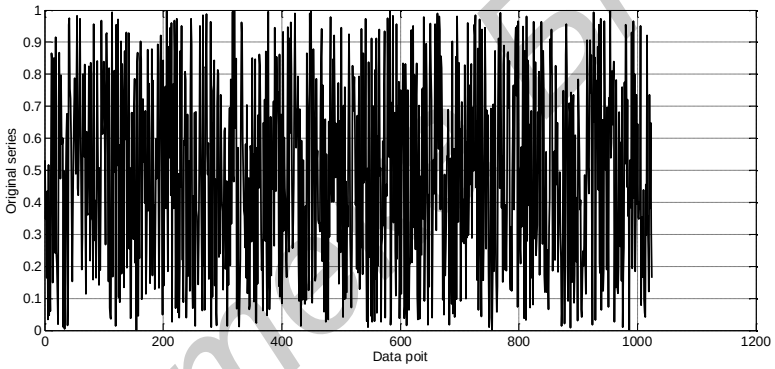
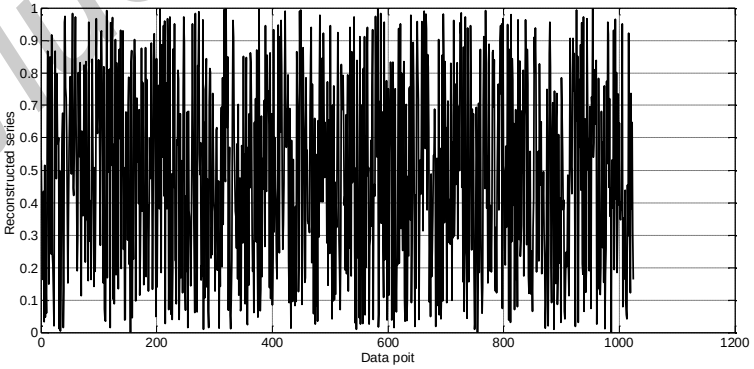


Рис. 5.3. Сингулярный спектр временной реализации случайного процесса rand при длине «гусеницы» L=10, I=10



а



б

Рис. 5.4. Исходный и восстановленный временные ряды случайного процесса типа rand при длине «гусеницы» L=10, I=10, ошибка реконструкции не превышает $6 \cdot 10^{-16}$: а – диаграмма исходного ряда; б – диаграмма восстановленного ряда

Как видно из приведенных результатов ошибки, применение алгоритма SSA позволяет получить достаточно высокую точность реконструкции случайных временных рядов. Фрактальные и случайные процессы имеют различные сингулярные спектры, что позволяет использовать алгоритм SSA для идентификации вида трафика.

Контрольные вопросы

1. Поясните, как метод SSA позволяет решать вопросы анализа и прогноза временных рядов модели трафика инфокоммуникационной системы.
2. Какую роль в алгоритме SSA играют собственные векторы автоковариационной матрицы?
3. Постройте алгоритм выделения аддитивных составляющих временного ряда с помощью метода «гусеница»-SSA.
4. Постройте алгоритмы выделения трендовых и периодических компонент с помощью метода «гусеница»-SSA.
5. Получите временной ряд трафика телекоммуникационной сети с помощью сетевого анализатора или воспользуйтесь генератором модели трафика. Примените к временному ряду алгоритм сингулярного анализа. Прокомментируйте полученный результат.
6. Есть ли связь между преобразованием Карунена – Лоэва и сингулярным преобразованием сигнала?

ЛИТЕРАТУРА

1. Олифер, В. Г. Компьютерные сети. Принципы, технологии, протоколы : учебник для вузов / В. Г. Олифер. – 4-е изд. – СПб. : Питер, 2010. – 944 с.

2. Кулябов, Д. С. Архитектура и принципы построения современных сетей и систем телекоммуникаций : учеб. пособие / Д. С. Кулябов, А. В. Королькова. – М. : РУДН, 2008. – 281 с.

3. Заборовский, В. С. Сети ЭВМ и телекоммуникации: анализ трафика в сетях коммутации пакетов : учеб. пособие / В. С. Заборовский, В. А. Мулюха, А. Г. Новопашенный. – СПб. : СПбГПУ, 2010. – 87 с.

4. Бестугин, А. Р. Контроль и диагностирование телекоммуникационных сетей / А. Р. Бестугин, А. Ф. Богданова, Г. В. Стогов. – СПб. : Политехника, 2003. – 174 с.

5. Бахарева, Н. Ф. Аппроксимативные методы и модели массового обслуживания. Исследование компьютерных сетей / Н. Ф. Бахарева, В. Н. Тарасов. – Самара : СНЦ РАН, 2011. – 327 с.

6. Кроновер, Р. М. Фракталы и хаос в динамических системах. Основы теории / Р. М. Кроновер. – М. : Постмаркет, 2000. – 352 с.

7. Управление качеством и вероятностные модели функционирования сетей связи следующего поколения : учеб. пособие / Г. П. Башарин [и др.]. – М. : РУДН, 2008. – 157 с.

8. Абрамов, С. А. Лекции о сложности алгоритмов / С. А. Абрамов.– М. : МЦНМО, 2009. – 256 с.

9. Оппенгейм, А. Цифровая обработка сигналов / А. Оппенгейм, Р. Шафер. – М. : Техносфера, 2006. – 856 с.

10. Вариченко, Л. В. Абстрактные алгебраические системы и цифровая обработка сигналов / Л. В. Вариченко, В. Г. Лабунец, М. А. Раков. – Киев : Наукова думка, 1986. – 248 с.

11. Сюзев, В. В. Цифровая обработка сигналов: методы и алгоритмы : учеб. пособие / В. В. Сюзев. – М. : Изд-во МГТУ им. Н.Э. Баумана, 2014. – 749 с.

12. Блейхут, Р. Е. Быстрые алгоритмы цифровой обработки сигналов / Р. Е. Блейхут. – М. : Мир, 1989. – 448 с.

13. Blahut, R. E. Algebraic Codes on Lines, Planes, and Curves. An engineering approach / R. E. Blahut. – Cambridge University Press, 2008. – 567 с.

14. Pueschel, M. The algebraic approach to the discrete cosine and sine transforms and their fast algorithms / M. Pueschel, J. M. F. Moura // SIAM Journal of Computing. – 2003. – Vol. 32, №5. – P. 1280–1316.

15. Pueschel, M. Algebraic Theory of Signal Processing / M. Pueschel, J. M. F. Moura [Electronic resource]. – Mode of access: http: arxiv. org/abs/cs. IT / 0612077, Dec. 2004.

16. Василенко, О. Н. Теоретико-числовые алгоритмы в криптографии / О. Н. Василенко. – М. : МЦНМО, 2003. – 328 с.

17. Юрков, К. В. Эффективность векторного квантования на основе числовых решеток в системах сжатия видеоинформации / К. В. Юрков, Б. Д. Кудряшов // Цифровая обработка сигналов и ее применение : сб. тр. 8-й Междунар. конф. – М., 2006. – С. 417–419.

18. Stinson, D. R. Combinatorial Designs: Constructions and Analysis / D. R. Stinson. – New York : Springer-Verlag, Inc. 2004. – 317 с.

19. Bremner, M. R. Lattice Basis Reduction An Introduction to the LLL Algorithm and Its Applications / M. R. Bremner. – CRC Press Taylor & Francis, 2012. – 334 с.

20. Петров, В. В. Статистический анализ сетевого трафика / В. В. Петров. – М. : Изд-во энергетического ин-та, 2003. – 47 с.

21. Главные компоненты временных рядов: метод «Гусеница» / под ред. Д. Л. Данилова, А. А. Жиглявского. – СПб. : Пресском, 1997. – 308 с.

22. Golyandina, N. E. Analysis of Time Series Structure: SSA and Related Techniques / N. E. Golyandina, V. V. Nekrutkin, A. A. Zhigljavsky. – Boca Raton : Chapmap & Hall/CRC, 2001. – 305 p.

23. Elsner, J. B. Singular Spectrum Analysis: A New Tool in Time Series Analysis / J. B. Elsner, A. A. Tsonis. – New York; London : Plenum Press, 1996. – 164 p.

24. Голяндина, Н. Э. Метод «Гусеница»-SSA: анализ временных рядов : учеб. пособие / Н. Э. Голяндина. – СПб. : ВВМ, 2004. – 76 с.

25. Лемешко, А. В. Алгоритм иерархическо-координационного управления информационным обменом в сети передачи данных / А. В. Лемешко // Открытые информационные и компьютерные интегрированные технологии. – 1998. – Вып. 1. – С. 323–328.

26. Евсеева, О. Ю. Решение задачи иерархическо-координационной маршрутизации в телекоммуникационных сетях методом предсказания взаимодействия / О. Ю. Евсеева // Открытые информационные и компьютерные интегрированные технологии. – 2003. – Вып. 21. – С. 102–111.

Св. план 2014, поз. 56

Учебное издание

Саломатин Сергей Борисович

МОДЕЛИРОВАНИЕ АЛГОРИТМОВ БЫСТРОЙ ОБРАБОТКИ СИГНАЛОВ В ИНФОКОММУНИКАЦИОННЫХ СЕТЯХ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *М. А. Зайцева*
Корректор *Е. И. Герман*
Компьютерная правка, оригинал-макет *В. М. Задоля*

Подписано в печать 16.03.2016. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 7,9. Уч.-изд. л. 7,0. Тираж 100 экз. Заказ 468.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
ЛП №02330/264 от 14.04.2014.
220013, Минск, П. Бровки, 6