Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

О.И. КУТУЗОВ, Т.М. ТАТАРНИКОВА

МОДЕЛИРОВАНИЕ СИСТЕМ И СЕТЕЙ ТЕЛЕКОММУНИКАЦИЙ

Учебное пособие

РГГМУ

Санкт-Петербург
2012

УДК 519.718

Кутузов О.И., Татарникова Т.М. Моделирование систем и сетей телекоммуникаций. Учебное пособие. – СПб, изд. РГГМУ, 2012. – 136 с.

ISBN 978-5-86813-325-1

Рецензент: Сикарев А.Д., д-р техн. наук, проф., зав. кафедрой Санкт-Петербургского университета водных коммуникаций

Учебное пособие содержит материал, связанный с имитационным моделированием телекоммуникационных сетей и их элементов, формально представляемых в виде случайных графов и систем массового обслуживания. Каждая глава содержит вопросы и задания для закрепления пройденного материала.

Предназначено для студентов, обучающихся по специальности «Информационная безопасность».

Kutuzov, O.I. and Tatarnikova, T.M. Simulation of systems and telecommunications networks: мmathematical schemes and algorithms. A manual. - St. Petersburg, RSHU Publishers, 2012. – 136 pp.

The book contains material related to simulations of telecommunications networks and their elements formally represented as random graphs and queuing systems. Each chapters includes questions and assignments to consolidate the material covered.

The manual is designed for training experts specializing in information security.

ISBN 978-5-86813-325-1

ПРЕДИСЛОВИЕ

«Моделирование систем и сетей телекоммуникаций» изучается студентами, обучающимися по специальности «Информационная безопасность» с целью усвоения теории и подготовки к овладению технологиями компьютерного моделирования сложных систем.

В пособии излагается материал, связанный с имитационным моделированием информационных сетей и их элементов, формально представляемых в виде случайных графов и систем массового обслуживания. Такой формализм широко используется в качестве математических моделей сетей.Пособие включает три раздела.

В первом разделе излагается ряд общих сведений, обеспечивающих, по мнению авторов, терминологичес-кое и понятийное содержание излагаемого материала.

Важным направлением при анализе и синтезе сетей с очень большим количеством узлов, структура которых нерегулярна, сложна и динамически эволюционирует во времени, является моделирование на основе аппарата теории графов. Другой широко распространенной формализацией сетей и их элементов (узлов, каналов связи) являются сети и системы массового обслуживания. Материал, связанный с современными подходами к описанию графов и моделей сетей, представлен во втором разделе.

В третьем разделе представлен материал по принципам и схемам построения моделирующих алгоритмов, обеспечивающих отображение функционирования объекта во времени. При имитационном моделировании воспроизводятся траектории «движения» моделируемого объекта в пространстве смены его состояний. При наличии в модели случайных факторов имитационные эксперименты выполняются с привлечением метода статистических испытаний или, иначе, метода Монте-Карло. Суть метода в генерации значений случайной базовой величины и нахождении по ним значений «произвольной» случайной величины. В разделе также приводится материал, связанный с генетическими алгоритмами, которые все чаще используются при оптимизации в сочетании с имитационным моделированием.

Список литературы представлен небольшим числом книг, в которых можно при желании получить сведения по теме пособия, в том числе и библиографические.

ВВЕДЕНИЕ

Информационные сети (ИС) есть результат слияния средств обработки и обмена информацией. В роли элементов ИС выступают информационные и прикладные процессы.

Информационные процессы (ИП) представляют собой совокупность взаимосвязанных процессов выявления, отбора, формирования информации, ее ввода в техническую систему, обработки, хранения и передачи.

Прикладные процессы (ПП) – типы ИП, ориентированные на выполнение конкретной прикладной задачи.

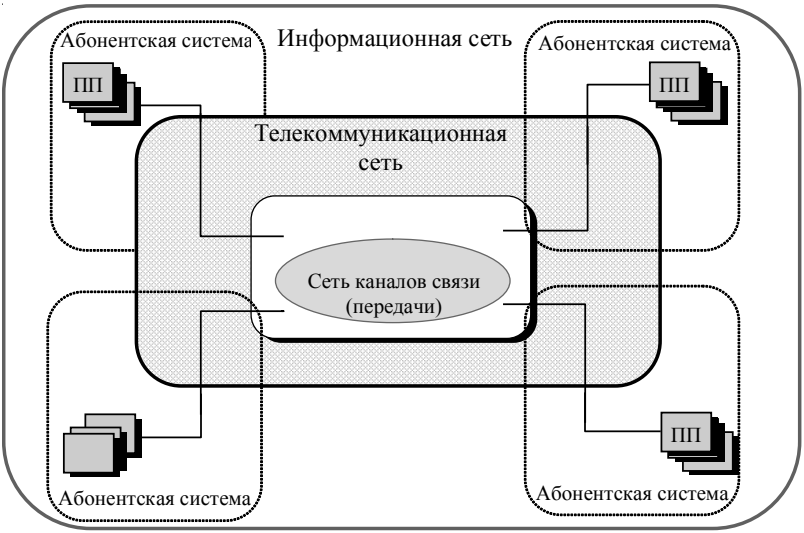


Рис. 1.1. Концептуальная модель информационной сети

Обобщенно функциональную архитектуру ИС можно представить в виде *трехуровневой концептуальной модели* (рис. 1.1):

первый уровень (внутренний) описывает функции и правила взаимосвязи при передаче различных видов информации между территориально удаленными абонентскими системами через физические каналы связи (передачи) и реализуется *транспортной сетью*;

второй уровень (промежуточный) описывает функции и правила обмена информацией в интересах взаимосвязи прикладных процессов различных абонентских систем и реализуется *телекоммуникационной сетью*. Телекоммуникационная сеть представляет собой единую инфраструктуру для обмена различными видами информации в интересах пользователей информационной сети;

третий уровень (внешний) образуется совокупностью *прикладных процессов*, размещенных в территориально удаленных *абонентских системах*. Абонентские системы (АС) являются потребителями информации и выполняют её содержательную обработку. Третий уровень, дополняя первый и второй указанными функциями обработки информации, образует внешний облик *информационной сети*.

Информационные сообщения на входы ИС поступают случайным образом и с разной степенью интенсивности. Разнородность сообщений, их разный приоритет, содержательная обработка сообщений и/или пакетов в узлах сети определяют случайную длительность их пребывания как в отдельной части сети, так и в сети в целом. Большое количество взаимодействующих элементов выполняют сложные функции для достижения заданной цели в условиях взаимодействия внешней среды со множеством случайных факторов. Неопределенность обстоятельств, с которыми придется столкнуться в реальности - всё это позволяет отнести ИС к классу сложных стохастических систем. Поэтому на этапах проектирования, модернизации, эксплуатации ИС возникает большое количество проблем, связанных с исследованием механизма внутрисетевых взаимодействий, формулировкой соответствующих требований к сети и её отдельным компонентами и разработкой алгоритмов управления сетью.

Любая система управления нуждается в информации об управляемом объекте и модели управляемого объекта, в моделировании тех или иных управляющих решений.

1. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Моделирование - это замещение одного объекта (оригинала) другим (моделью) и фиксация и изучение свойств модели. Замещение производится с целью упрощения, удешевления, ускорения изучения свойств оригинала. Объектом-оригиналом может быть естественная или искусственная, реальная или воображаемая система *S*.Любая система может быть описана количественно с помощью определённой совокупности величин. Эту совокупность можно разде-лить на два класса:

- Параметры – величины, описывающие первичные свойства системы, её элементов и не зависимые от других величин;

- Характеристики – величины (зачастую, векторные), описывающие качественные свойства системы и являющиеся вторичными по отношению к параметрам, т.е. зависящие от параметров.

Система проявляет свои свойства под влиянием внешних воздействий. Множество параметров системы *S* и их значений отражает её внутреннее содержание: структуру и принципы функционирования. Характеристики *S* – это в основном её внешние признаки, которые важны при взаимодействии с другими системами. Характеристики *S* находятся в функциональной зависимости от её параметров. Каждая характеристика системы определяется в основном ограниченным числом параметров. Остальные параметры не влияют на значение данной характеристики *S*. Исследователя интересуют, как правило, только некоторые характеристики при конкретных воздействиях на систему.

Модель - это тоже система со своими множествами параметров и характеристик. В модель включаются только существенные аспекты, представляющие оригинал, и отбрасываются *все остальные* (которых бесконечное большинство). Существенный или несущественный аспект описания определяют согласно цели исследования. Модель *адекватна* реальному объекту в рамках поставленной задачи. На практике двигаются от простых моделей к более сложным.

Структурно модель *M* можно представить рис. 1.2.

В математическом виде модель $Y = M(X)$ - закономерность, преобразующая входные значения в выходные.

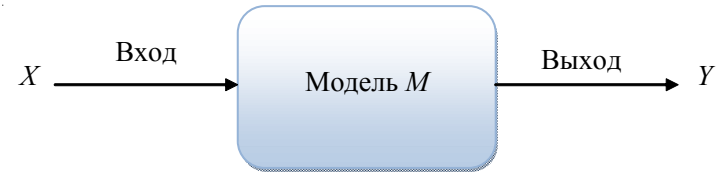


Рисунок 1.2. Структурное изображение модели

С выражением $Y = M(X)$ можно решить три вида задач, которые приведены в табл. 1.1.

Условия, которые должны быть, безусловно, выполнены для решения задач, называются *ограничениями*. А часть ограничений, относитель-

Таблица 1.1

Виды задач моделирования

	Известно	Неизвестно	Решение
Прямая задача	X, M	Y	$Y = M(X)$
Обратная задача	Y, M	X	$X = M^{-1}(Y)$
Задача настройки модели	X, Y	M	$M = f(X, Y)$

но которых высказывают только пожелания («быть как можно больше или меньше»), называются *критериями*.

Решений обычно бывает много. Для нахождения лучшего следует сузить область решений, накладывая определённые ограничения, чтобы отсеять неудовлетворяющие этим ограничениям. Такие задачи часто называют *задачами управления*. В целом получается обратная задача. А то, что надо определить - управляемая переменная. Другими словами как следует изменить входной параметр (управление), чтобы обеспечить выполнение критерия, не выйти за ограничения, чтобы при этом критерий принял наилучшее значение?

Цель моделирования – получение новых сведений об изучаемом объекте или явлении. Познание любой системы сводится по существу к созданию её модели. Моделирование целесообразно, когда у модели отсутствуют те признаки оригинала, которые препятствуют его исследованию.

Достижения математики привели к распространению математических моделей (ММ) различных объектов и процессов. ММ представляют собой формализованное представление системы с помощью математических соотношений, отражающих процесс функционирования системы. По принципам построения математическое моделирование делится на аналитическое и имитационное.

Аналитическое моделирование - формализованное описание системы, которое позволяет получить решение уравнения в явном виде, используя известный математический аппарат. Однако аналитическое представление подходит лишь для очень простых и сильно идеализированных задач и объектов. Сложные объекты редко удаётся описать аналитически, к их описанию привлекаются имитационные модели.

Имитационное моделирование - это частный случай математического моделирования, а имитационная модель - логико-математическое описание объекта.

Имитационная модель (ИМ) - это совокупность описания системы и внешних воздействий, алгоритмов функционирования системы или правил изменения состояния системы под влиянием внешних и внутренних возмущений. Эти алгоритмы и правила позволяют имитировать процесс функционирования системы и производить нахождение оценок интересующих характеристик в режиме вариантных расчётов путем выполнения имитационных экспериментов с целью получения информации о моделируемой системе. Экспериментирование с моделью называют имитацией.

Метод имитационного моделирования (ИМ) заключается в создании логико-аналитической (математической) модели системы и внешних воздействий, имитации функционирования системы, т.е. в определении временных изменений состояния системы под влиянием внешних воздействий и в получении выборок значений выходных параметров, по которым определяются их основные вероятностные характеристики. Данное определение справедливо для стохастических систем. При исследовании детерминированных систем отпадает необходимость изучения выборочных значений выходных параметров.

Процесс моделирования есть процесс перехода из реальной области в виртуальную (модельную) посредством *формализации*. Далее происходит изучение модели (собственно *моделирование*) и, наконец, *интерпретация* результатов как обратный переход из виртуальной области в реальную. Этот путь заменяет прямое исследование объекта в реальной области.

Формализация обеспечивает построение *математической схемы*. Математическую схему можно рассматривать как звено при переходе от содержательного к формализованному описанию процесса функционирования системы с учётом воздействия внешней среды, т.е. имеет место цепочка: описательная (концептуальная) модель - математическая схема - аналитическая или имитационная модель.

При имитационном моделировании сетей понятие «математическая схема» позволяет рассматривать математику не как метод расчёта, а как метод мышления, средство формулирования понятий, что является наиболее важным при переходе от словесного описания к формализованному представлению процесса функционирования системы в виде некоторой математической модели (ММ).

Важным направлением при анализе и синтезе сетей с очень большим количеством узлов, структура которых нерегулярна, сложна и динамически эволюционирует во времени, является моделирование на основе аппарата теории графов. Примерами таких сетей служат технологические (Интернет как сеть компьютеров, корпоративные компьютерные сети, транспортные и электрические сети), информационные (цитиро-

вания в научных статьях, ссылок WWW), биологические (сети нейронов в мозге, взаимодействующих протеинов, генетические сети), ячейки химических реакций, социальные системы (люди и связи между ними).

В последнее десятилетие возрос интерес исследователей к сетям с очень большим количеством узлов, структура которых нерегулярна, сложна и динамически эволюционирует во времени. В пособии рассматриваются основные особенности и типы случайных графов и подходы к их моделированию.

Другой широко распространенной формализацией сетей и их элементов (узлов, каналов связи) являются сети и системы массового обслуживания. Применение этих математических схем широко используется для оценивания необходимых сетевых ресурсов в зависимости от нагрузки, поступающей в сеть. Однако представление процесса функционирования распределенной информационной системы управления в виде сети схем массового обслуживания позволяет хорошо описать процессы, происходящие в системе, но при сложных законах входящих потоков и потоков обслуживания не даёт возможности получения результатов в явном виде, что определяет необходимость имитационного моделирования таких математических схем.

Особенностью имитационного моделирования является то, что имитационная модель позволяет воспроизводить моделируемые объекты с сохранением их логической структуры, с сохранением поведенческих свойств (последовательности чередования во времени событий, происходящих в системе), т.е. динамики взаимодействий.

При наличии в модели случайных факторов (если изучаемый процесс достаточно сложен, то почти неизбежно он случаен) имитационные эксперименты выполняются с привлечением метода статистических испытаний или, иначе, метода Монте-Карло [7].

Метод Монте-Карло есть метод математического моделирования случайных явлений, в которых сама случайность непосредственно включается в процесс моделирования и представляет собой его существенный элемент. Каждый раз, когда в ход операции вмешивается тот или другой случайный фактор, его влияние имитируется с помощью специально организованного «розыгрыша» или жребия. В результате «розыгрыша» получается один экземпляр – одна реализация случайного явления. Произведя такой «розыгрыш» очень большое число раз, получают статистический материал – множество реализаций случайного явления, который можно обработать обычными методами математической статистики. При этом применяется закон больших чисел (теорема Чебышева). Согласно этой теореме, при большом числе опытов среднее арифметическое наблюденных значений случайной величины почти наверняка мало отличается от её математического ожидания. Аналогичным образом

могут быть найдены и другие выборочные характеристики случайных величин.

Отметим ещё раз. Основными элементами, из совокупности которых складывается монте-карловская модель, являются отдельные реализации моделируемого случайного явления. Реализация представляет собой как бы один случай осуществления моделируемого случайного явления (процесса) со всеми присущими ему случайностями. Она разыгрывается с помощью специально разработанной процедуры или алгоритма, в котором важную роль играет собственно «розыгрыш» или бросание жребия». Каждый раз, когда в ход моделируемого процесса вмешивается случайность, её влияние учитывается не расчётом, а бросанием жребия.

В англоязычной литературе терминам «имитация», «имитационная модель», «имитационный эксперимент» приблизительно соответствует термин «simulation». Наиболее известна трактовка термина «имитация», которую дал Р. Шеннон, определив её как «процесс конструирования модели реальной системы и постановки экспериментов на этой модели с целью понять поведение системы либо оценить (в рамках ограничений, накладываемых некоторым критерием или совокупностью критериев) различные стратегии, обеспечивающие функционирование данной системы». Аналогичное определение этому термину дает Т. Нейлор: «численный метод проведения на цифровых вычислительных машинах экспериментов с математическими моделями, описывающими поведение сложных систем в течение продолжительных периодов времени». Обе приведённые трактовки термина «simulation» укладываются в то содержание терминов «имитация», «имитационная модель», «имитационный эксперимент», которое рассмотрено выше.

При создании имитационных моделей в настоящее время используется два подхода: дискретный и непрерывный. Выбор подхода в значительной мере определяется свойствами объекта-оригинала и характером воздействия на него внешней среды. Метод статистического моделирования можно рассматривать как частный случай дискретных вероятностных имитационных моделей.

В основе дискретных вероятностных имитационных моделей лежит понятие «событие». *Событие* определяется как точка во времени, в которой происходят скачкообразные изменения состояний системы. Для получения требуемых результатов моделирования достаточно наблюдать систему в те моменты, когда происходят события. Резкие переходы (скачки), совершаемые моделью при переходе от одного события к другому, указывают на то, что процесс протекает в дискретном времени, откуда появилось название «дискретное моделирование».

Современными технологиями дискретного имитационного моделирования (ИМ) охватываются два самодостаточных (вместе и порознь)-

подхода: дискретно-событийное моделирование и мультиагентный подход.

Дискретно-событийное моделирование есть способ описания и моделирования процессов, присущих системам массового обслуживания, системам с отказами и восстановлением элементов, дискретным производствам и т.п. В дискретно-событийном моделировании функционирование системы представляется как хронологическая последовательность событий. Событие происходит в определенный момент времени и знаменует собой изменение состояния системы. Продвижение системного времени реализуется посредством программирования симулятора – «движителя», который с минимальными затратами машинного ресурса воспроизводит во времени движение (смену состояний) объекта моделирования, отображая действие механизма причинно-следственных связей на смену состояний.

Мультиагентное моделирование - метод, исследующий поведение децентрализованных *агентов* и то, как такое поведение определяет поведение всей системы в целом. Другими словами, поведение агентов определяется на индивидуальном уровне, а глобальное поведение возникает как результат деятельности множества агентов (моделирование «снизу вверх») и их взаимодействия. Агент – это, по сути, процесс, последовательность событий и работ, описывающая поведение во времени какого-либо объекта в моделируемой системе. С формально-логической точки зрения выбор, осуществляемый любым агентом, предопределён достигнутым на момент выбора состоянием системы и значениями параметров, известных агенту. Поэтому действия агентов имитируются в модели точно так же, как и любые другие события – как прямые следствия из достигнутого состояния системы. И модельное время продвигается симулятором строго вперед, в точном соответствии с механизмом причин и следствий.

В семантическом плане концепция агентов как самостоятельных, активно действующих сущностей, оказывается чрезвычайно полезной при моделировании систем, в которых совокупное действие подобных сущностей приводит к эволюции, непредсказуемой с точки зрения отдельных агентов, и иногда более «разумной», чем поведение любого отдельно взятого агента. Эта концепция существенно расширяет сферу практического использования ИМ, поскольку приводит к новым смыслам при интерпретации моделей, и она поддерживается новыми языковыми и графическими средствами ИМ [2].

В обеих версиях дискретного ИМ симулятор-движитель продвигает вперед текущий «временной срез» системы. Этот срез, а точнее – временной слой, имеет минимальную «толщину», необходимую для правильного определения ближайших предстоящих изменений и для правильного рекуррентного пересчета показателей, требующего помнить моменты

последних произошедших изменений. В основе построения «движителя» лежат две основные схемы построения алгоритмов моделирования: схема событий и схема процессов [4]. Схема событий используется при дискретно-событийном моделировании, а схема процессов – при мультиагентном моделировании.

Таким образом, при моделировании стохастических систем парадигма *имитационного* (статистического, вероятностного [4]) *моделирования* включает две составляющие: симулятор – «движитель», реализующий продвижение системного времени, и метод Монте-Карло, обеспечивающий разыгрывание «случайностей». В совокупности эти две составляющие и строят траектории – реализации функционирования моделируемой системы, в которой присутствуют случайные факторы и объекты. Обе составляющие наличествуют в обоих методах дискретного ИМ.

Для реализации ИМ на ЭВМ разработаны как специализированные языки моделирования, так и имитационные системы. Наиболее известными языками моделирования являются SIMULA, SIMSCRIPT, GPSS, SOL, CSL. Эффективность языков моделирования существенно зависит от наличия диалоговых и графических средств. Поэтому в систему имитации помимо совокупности имитационных моделей с их программным и информационным обеспечением входят средства программного сервисного обеспечения. Популярны системы AnyLogic, MATLAB и Simulink, Arena, СИМПАС и др.

Удобство языка моделирования во многом определяется ориентацией на определенную предметную область. Но это и определенное ограничение языков моделирования по сравнению с универсальными языками программирования высокого уровня. В вузах, например, широко используют Паскаль и его расширения, C++ и др. Поэтому в данном пособии авторы сделали акцент на модельную и алгоритмическую составляющие при изложении вопросов, связанных с имитационным моделированием систем и сетей телекоммуникаций.

Вопросы и задания

1. Приведите примеры прямой и обратной задач.
2. В чем отличие аналитического моделирования от имитационного?
3. Какова основная цель математического моделирования?
4. Приведите пример задачи настройки параметров.
5. В чем суть дискретного моделирования?
6. Что означает динамическое моделирование?
7. В чем суть мультиагентного моделирования?
8. Каким образом реализуется дискретно-событийное моделирование?
9. Объясните назначение метода Монте-Карло в моделировании.

10. Какие виды объектов можно представлять с помощью графов и почему?
11. Что называют задачами управления?
12. Какова основная цель моделирования вообще? Приведите примеры.
13. Объясните, как реализуется непрерывное моделирование.

2. МАТЕМАТИЧЕСКИЕ СХЕМЫ МОДЕЛИРОВАНИЯ СИСТЕМ И СЕТЕЙ ТЕЛЕКОММУНИКАЦИЙ

2.1. Графовые модели сетей

Моделирование на основе аппарата теории графов является важным направлением при анализе и синтезе сетей с очень большим количеством узлов, структура которых нерегулярна, сложна и динамически эволюционирует во времени. Учение о графах, представляющих собой системы линий, соединяющих какие-то заданные точки, объединяет геометрическую наглядность с математической содержательностью. Первым исследователем графов, являющихся математическим образом сетей, был Леонард Эйлер (1707–1783 гг.). Изначально теория графов описывала регулярные графы, однако начиная с 1950-х гг., сложные сети стали описывать с помощью теории случайных графов, предложенных в качестве наиболее подходящей модели сложных сетей.

2.1.1. Некоторые понятия теории графов

Граф - это совокупность узлов (*вершин*) со связями (ребрами) между ними**.** В строгом определении графом называется пара множеств $G=(V, E)$, где V есть подмножество любого счётного множества, E - подмножество VЧV. Примеры графов изображены на рис. 2.1.

Отметим некоторые частные представления графов.

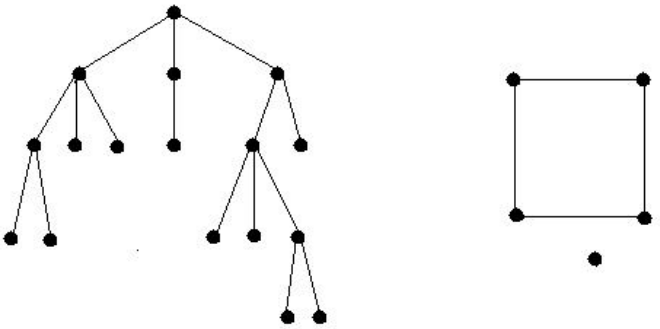


Рис. 2.1. Примеры графов

Простые неориентированные графы.

Схема, состоящая из «изолированных» вершин, называется *нулевым* графом**.** Графы, в которых не построены все возможные ребра, называются *неполными* графами (рис.2.2).

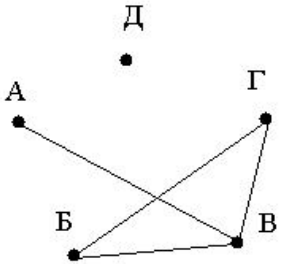


Рис. 2.2. Неполный граф с пятью вершинами

При изображении графов на рисунках или схемах отрезки могут быть прямолинейными или криволинейными; длины отрезков и расположение точек произвольны. Например, все три фигуры на рис. 2.3 изображают один и тот же граф.

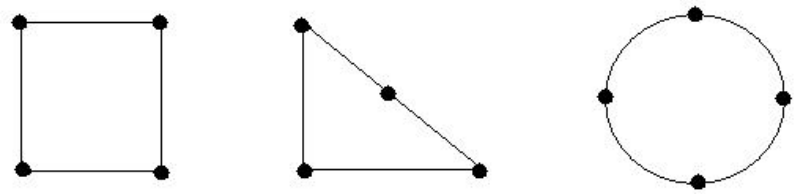


Рис. 2.3. Разное изображение одного и того же графа

На рис. 2.4. изображен граф, соответствующий всем связям между вершинами (каждый узел соединен с каждым). Такой граф называется *полным*.

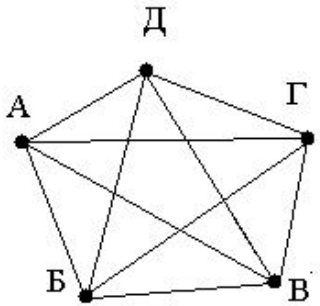


Рис. 2.4. Полный граф с пятью вершинами

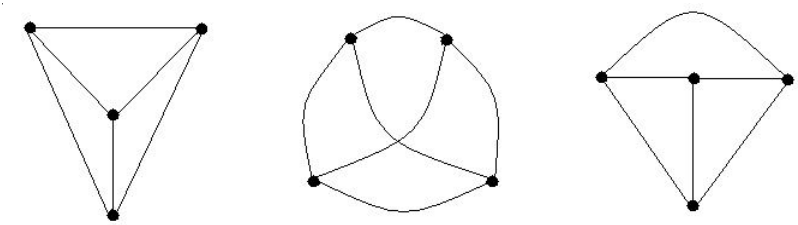
Если полный граф имеет n вершин, то количество ребер будет равно $n(n\text{-}1)/2$. Совокупность вершин графа с ребрами, превращающими непол-

ный граф в полный, называется *дополнением* графа.

На рис. 2.5 представлено несколько вариантов изображения полного графа с четырьмя вершинами.

Графы, изображенные на рис. 2.5, дают одну и ту же информацию о совершенных связях. Такие графы называют изоморфными (одинаковыми).

Графы изоморфны, если у них одинаковое количество вершин, и если вершины одного графа соединены ребром, то и соответствующие им вершины другого графа тоже соединены ребром.



а) б) в)

Рис. 2.5. Варианты изображения полного графа с четырьмя вершинами

Граф, который можно начертить так, чтобы его ребра пересекались только в вершинах, называется плоским графом. На рис. 2.5,а и 2.5,в изображены плоские графы. Простые циклы и деревья являются наглядными примерами плоских графов.

Граф называется связным, если любая пара его вершин - связная. Граф называется *несвязным*, если в нём есть хотя бы одна несвязная пара вершин. Совокупность вершин, связанных между собой ребрами, но отдаленных от остальной части графа, называется *компонентом* графа.

На рис. 2.6 изображен несвязный граф, состоящий из двух компонентов. Если, например, между вершинами Д и Е либо какими другими разных компонентов графа (рис. 2.6) провести ребро, то граф станет связным. Такое ребро в теории графов (после удаления которого граф из связного превращается в несвязный) называется *мостом***.**

В современной теории сетей число связей узла, количество ребер, присоединенных к *i*-й вершине, называется *степенью* (degree), или *порядком* k_i этой вершины, а две вершины, соединенные ребром, называются *соседними.* Как видно из рис. 2.7, средний узел (В) имеет степень пять, остальные узлы - степень три.

Вершина называется *нечётной*, если степень этой вершины нечётная, *чётной* - если степень этой вершины четная.

Для рассмотренных выше определений в теории графов сформулированы следующие закономерности:

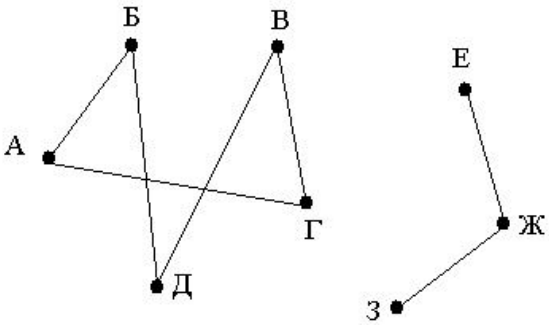


Рис. 2.6. Пример несвязного графа

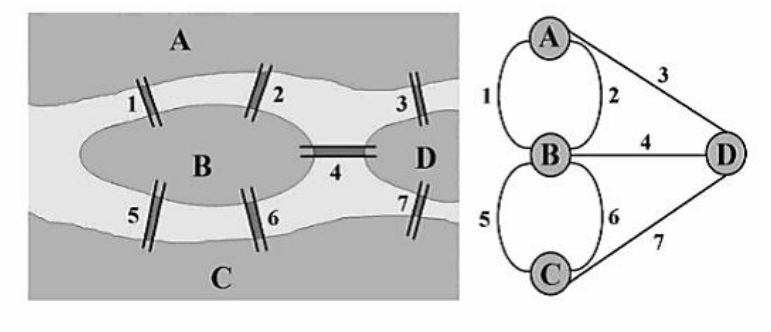


Рис. 2.7. Граф для проблемы Эйлера.
(узлы этого графа – четыре района старого Кенигсберга, соединенные между собой семью мостами, которым соответствуют ребра графа. В 1736 г. Эйлер доказал, что начав с некоторой точки, невозможно пройти все мосты и вернуться в исходную точку, не посетив один из мостов дважды)

- если степени всех вершин графа равны, то граф называется *регулярным*. Любой полный граф – однородный, и степени вершин полного графа на единицу меньше числа вершин этого графа;

- сумма степеней вершин графа число чётное, равное удвоенному числу ребер графа. Эта закономерность справедлива не только для полного, но и для любого графа;

- число нечётных вершин любого графа чётно.

Понятие степень (*порядок*) является *локальной* характеристикой графа. Нелокальную, *целостную* структуру сети определяют двумя понятиями: *путь* (path) и *петля* (loop), или *цикл* (cycle). Путь – это чередующаяся последовательность смежных узлов и связей между этими узлами, когда узлы не повторяются. Циклом или петлей называется путь, когда начальный и конечный узел совпадают. *Сети без циклов называются деревьями*. Число *узлов N* (называемое *размером* сети) и число *связей L* в де-

ревьях связаны простым соотношением $N = L - 1$.

Приведённые на рисунках конструкции относятся к классу *простых неориентированных графов* (их называются также *невырожденными графами*). У невырожденных графов две вершины либо не соединены, либо связаны неориентированными ребрами, но ни одна из вершин не соединена сама с собой. В *вырожденных* или *псевдографах* возможны многократные связи между вершинами.

Ориентированные графы.

Существуют значительные классы практических задач, которые решить с помощью ранее рассмотренных типов графов невозможно. Так, например, схема дорог и площадей города изображается с помощью плоского графа. Но если нужно этой схемой воспользоваться с целью проезда по городу на автомашине, а движение на отдельных (или на всех) улицах одностороннее, тогда могут помочь сориентироваться в этой ситуации стрелки, расположенные, например, прямо на ребрах-улицах рассматриваемой схемы (графа) города.

Ребро графа называется *ориентированным ребром*, если одну из его вершин считать началом, а другую - концом этого ребра. Граф, у которого все ребра ориентированные, называется *ориентированным* графом.

На практике часто ориентированные графы используют для наглядного представления процесса и результата спортивных соревнований. Так, на рис. 2.8 с помощью ориентированного графа показаны результаты игр между командами А, Б, В, Г, Д, Е. Если игра может быть сыграна

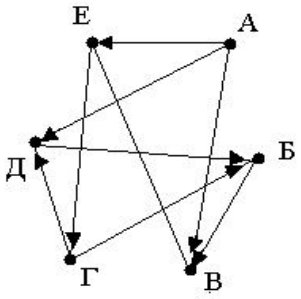


Рис. 2.8. Ориентированный граф АБВГДЕ

вничью, то обычно ребро графа оставляют неориентированным (ребро ВЕ) и такой граф называют *смешанным*.

По аналогии с ранее рассмотренными (неориентированными) графами ориентированные графы имеют такие характеристики, как *степень вершины*, понятия *пути* и *цикла*.

Степенью выхода вершины ориентированного графа называется число ребер, для которых эта вершина является началом (число ребер, «выходящих» из вершины).

Степенью входа вершины ориентированного графа называется число ребер, для которых эта вершина является концом (число ребер, «входящих» в вершину).

Так, на рис. 2.9 изображен ориентированный граф АБВГД. Степени входа и выхода некоторых его вершин такие:

Ст.вх.А=2, Ст.вых.А=1, Ст.вх.В=2, Ст.вых.В=0, Ст.вх.Д=1, Ст.вых.Д=3.

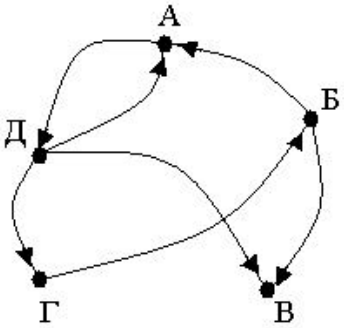


Рис. 2.9. Ориентированный граф АБВГД

Путём в ориентированном графе от вершины А1 к вершине А*n* называется последовательность ориентированных ребер А1А2, А2А3, ..., А*n*-1, А*n*, в которой конец каждого предыдущего ребра совпадает с началом следующего, и каждое ребро встречается в этой последовательности только один раз.

На рис. 2.10 показаны примеры путей в ориентированном графе. Причём, первые два пути – простые, ни одна из вершин не содержится в нём

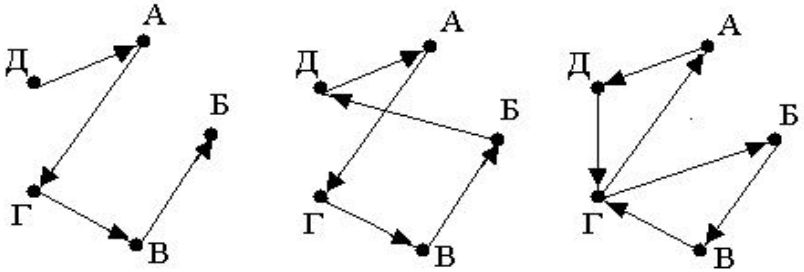


Рис. 2.10. Примеры путей в ориентированном графе

более одного раза. Третий путь не является простым, так как через вершину Г путь «проходил» дважды.

Ориентированным циклом называется замкнутый путь в ориентированном графе. На рис. 2.10 б, в приведены примеры ориентированных циклов в графах. Цикл, как и любой другой путь в графе, имеет длину, которая определяется числом ребер в этом пути. Так, на рис. 2.11 пути от А к Д могут быть различны и иметь различную длину. Первый путь имеет длину 2, второй - 3, а третий - 4.

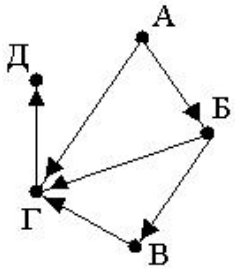


Рис. 2.11. Примеры путей разной длины в ориентированном графе

Длина «кратчайшего пути» между двумя вершинами называется расстоянием между ними. Так, расстояние между вершинами А и Д на графе рис. 2.11 равно 2; записывают так: S(АД)=2.

Если в ориентированном графе нельзя «пройти» от одной вершины до другой, то расстояние между ними называют бесконечным (обозначают значком бесконечности). Так, расстояние между вершинами Б и Д графа, представленного на рис. 2.12 бесконечно: $S(БД) = \infty$.

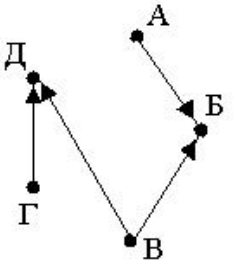


Рис. 2.12. Ориентированный граф, в котором $S(БД)= \infty$

Нагрузка узла (betweenness centrality). Для многих сетей (транспортных, телекоммуникационных, социальных) необходимо определить относительную важность входящих в неё узлов. Загруженность узла в сети определяется как суммарное число кратчайших путей между всеми остальными узлами, которые проходят через данный узел i:

$$B(i) = \frac{\sum_{\text{АГ}} \sigma_{\text{АГ}}(i)}{\sigma_{\text{АГ}}},$$

где $B(i)$ – загруженность i-го узла сети; $\sigma_{\text{АГ}}(\text{i})$ – число кратчайших путей из узла А в узел Г через узел i; $\sigma_{\text{АГ}}$ – число кратчайших путей между всеми парами А и Г.

Эта величина важна в изучении транспортных потоков и обычно называется *нагрузкой* (загруженностью) узла (или связи), поскольку характеризует долю проходящих через узел кратчайших путей. Эту величину можно также считать индикатором наиболее влиятельных и важных персон (VIP) в социальной сети. Узлы с высоким значением B являются наиболее загруженными. В отличие от степени узла, понятие важности узла отражает топологию всей сети.

Помимо транспортных задач ориентированные графы активно используются в сетевом планировании, в математике: теория игр, теория множеств, при решении многих задач, в частности, комбинаторных.

2.1.2. Случайные графы и сети

Графы, в которых *распределение связей* между узлами имеет случайный характер, называются *случайными графами*. Впервые случайные графы были изучены венгерскими математиками Полом Эрдосом (Paul Erdxs) и Альфредом Реньи (Alfred Rényi).

Произвольный конкретный граф (сеть) является конкретной реализацией случайного графа (случайной сети). В свою очередь, всякая числовая характеристика случайного графа может рассматриваться как случайная величина. Для того чтобы получить среднее значение для некоторой величины в случайной сети, нужно усреднять эту величину по всем реализациям, принимая во внимания их статистический вес.

Числовые характеристики случайных сетей.

Кратчайшая длина пути (*геодезическая линия*). Длины всех связей между смежными узлами считаются равными единице. *Кратчайшее расстояние* l_{ij} между узлами i и j есть *длина самого короткого пути* (*геодезическая линия*) между ними в сети. Среднее межузловое расстояние l есть среднее l_{ij} по всем тем парам узлов (i,j), между которыми существует хотя бы один соединяющих их путь (сеть может содержать несоединённые между собой узлы).

Диаметр сети. Диаметр сети l_D есть максимальное расстояние между узлами сети. Зависимость l или от размера N сети зависит и от

архитектуры сети. Кратчайшая длина пути сети и её диаметр являются *нелокальными* свойствами сети.

Матрицы смежности. Сетевые структуры можно описывать в матричной форме. Сеть из N узлов описывается квадратной матрицей смежности a размерности $N \times N$. Ненулевые элементы такой матрицы обозначают наличие связей между соответствующими узлами. Для неориентированных сетей недиагональный элемент a_{ij} матрицы смежности равен числу связей q_{ij} между узлами i и j и, следовательно, матрица для такой сети симметрична. Предполагается, что петли единичной длины и кратные связи запрещены, следовательно, значения диагональных элементов равны нулю. Любые структурные свойства сети могут быть выражены через свойства матрицы смежности. Например, степень узла i равна:

$$q_i = \sum_j q_{ij}.$$

Реальные сети и их модели являются разреженными. Число соединений в таких сетях значительно меньше, чем у полносвязанной сети. Таким образом, огромное большинство элементов матриц смежности реальных сетей равно нулю.

Помеченные графы – такие графы, у которых различаются как вершины, так и рёбра. Каждый такой граф можно представить матрицей смежности, элемент a_{ij} которой определяется количеством рёбер между вершинами i и j, если $i \neq j$, или удвоенное число рёбер, если $i = j$.

Распределение узлов по числу связей (*degree distribution*). Перечень порядков (степеней q) графа определяет распределение вероятностей p_q по порядкам k (степеням q). Часто распределением узлов по числу связей (degree distribution) для отдельного графа g называется отношение вида $p_g(q) = N_g / N p_g(g) = N_g / N$, где N_g есть число узлов степени q в графе g. Бинарная функция $p(q, q`)$ задаёт совместную вероятность одной вершине иметь порядок $q$, а другой – $q`$.

Распределение узлов по числу связей является простейшей статистической характеристикой случайной сети. Во многих случаях знание этой характеристики достаточно для понимания свойств такой сети и процессов, которые в ней происходят.

Кластеризация – это локальная характеристика сети. Она характеризует степень взаимодействия между собой ближайших соседей данного узла. В большинстве сетей, если узел А соединен с узлом Б, а узел Б – с узлом Д, то существует большая вероятность, что узел А соединен с узлом Д (друзья наших друзей, обычно, также являются и нашими друзьями).

Коэффициент кластеризации данного узла есть вероятность того, что два ближайших соседа этого узла сами есть ближайшие соседи. Други-

ми словами, если узел j имеет q_j ближайших соседей с числом t_j связей между ними, то локальный коэффициент кластеризации равен:

$$C_j(q_j) = \frac{t_j}{q_j(q_j - 1)/2}.$$

Число t_j есть суммарное число треугольников (циклов длины 3), прикрепленных к узлу j, а $q_j(q_j\text{-}1)/2$ – максимально возможное число возможных треугольников (см. рис. 2.13). Если все ближайшие соседи узла j взаимосвязаны, то $C_j = 1$. Когда между ними нет связей (как у деревьев), то $C_j = 0$.

Изображенная на рис. 2.13 сеть содержит один треугольник (цикл длины 3) и восемь соединенных триад. Следовательно, у этой сети коэффи-

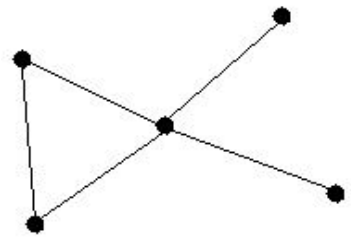


Рис. 2.13. Фрагмент сети

циент кластеризации равен $C = 3 \cdot 1/8 = 3/8$. Отдельные узлы имеют коэффициенты кластеризации 1, 1, 1/6, 0 и 0, а среднее значение равно $C = 13/30$.

Кластеризация всей сети определяется как $C = 3M_\Delta/M_V$, где M_Δ - число треугольников в сети, а M_V - число связанных триад. «Связанная триада» означает узел и два его ближайших соседа (рис. 2.13). Можно показать, что число триад из узлов равно

$$\sum_j q_j(q_j - 1)/2.$$

По сути, коэффициент кластеризации C есть доля тех триад, у которых есть три ребра, образующих треугольник, т.е. циклов длины 3.

Среднее значение кластеризации по всем узлам:

$$\overline{C} = \frac{1}{N}\sum_i C_i \quad \langle C \rangle.$$

Таким образом, *кластеризация характеризует статистику циклов* (треугольников) в сети. Большинство реальных сетей в мире обладают высокой кластеризацией,

В пределе с фиксированным q при $N \to \infty$ получаем *разряженные сети*, в которых число связей в узле намного меньше, чем в полносвязан-

ном графе. В таких неориентированных сетях вводится понятие гигантского связанного кластера узлов сети. Если относительный размер наибольшего связанного кластера узлов сети приближается к ненулевому значению при росте числа узлов к бесконечности, этот кластер называется *гигантским связанным кластером* или наибольшей *связанной компонентой* сети. Без гигантского связанного кластера сеть представляет собой лишь множество маленьких разделенных кластеров. Возникновение такого кластера можно рассматривать как *структурный фазовый переход*.

Подграфы и мотивы. Для характеристики локальных структурных свойств сетей используется подход на основе анализа паттернов* связей узлов. Например, связанный подграф представляет собой подмножество узлов, соединенных между собой специфической структурой соединений.

Так, узлы A, B, C образуют треугольный подграф, а узлы A, B, F, и G образуют четырехугольный подграф (рис.2.14). Число различных возможных подграфов растёт экспоненциально с ростом числа узлов в сети.

Не все подграфы возникают в сетях с одинаковой частотой. Так, в квадратных решетках имеются только квадратные подграфы. В сложных

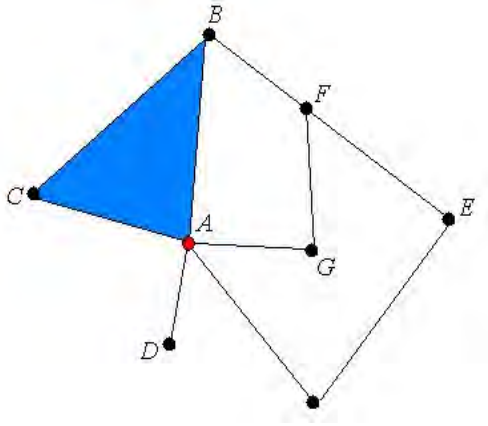


Рис. 2.14. Различные типы подграфов в сети

сетях со случайным соединением связей между узлами можно обнаружить и треугольные, и квадратные, и пятиугольные подграфы.

Конфигурация (форма) и взаимосвязь как два важнейших аспекта организации графа объединены в понятие *паттерна.* При этом некоторые подграфы, называемые *мотивами*, имеют большую частоту появления в сетях определенной структуры, чем в рандомизированных версиях тех же самых сетей. Например, ориентированный треугольный подграф на рис. 2.15 можно обнаружить во многих биологических сетевых структурах (нейронных сетях, регуляторных клеточных структурах).

В то же время мотив из четырех узлов на рис. 2.16 можно найти в электрических сетях, но его не встретишь в биологических структурах.

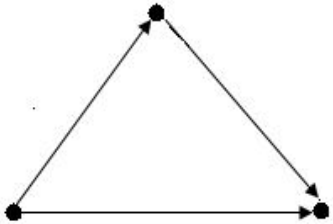


Рис. 2.15. Ориентированный треугольный подграф

Решётки. В регулярных решетках конечной размерности зависимость $\bar{l}(N)$ имеет степенной закон:

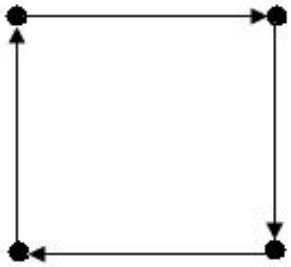


Рис. 2.16. Ориентированный четырехугольный подграф

$$\bar{l} \infty (N)^{1/d},$$

где d есть размерность решетки – целое число. Например, для двумерной решетки из 10^{12} узлов $\bar{l} \infty 10^{6}$.

К сетевым структурам, в которых растет более медленно, чем по степенному закону с положительным индексом, применяется термин *феномен тесного мира* (small world phenomenon).

Клики, сообщества, общины, группы, коммуны. Кликами (cliques) называются полносвязные подграфы некоторого графа.

Под *сообществами* понимается подграфы, для которых связи между узлами внутри подграфов сильнее, многочисленнее и насыщеннее, чем между узлами различных подграфов (рис. 2.17).

Допустим, связи с максимальной важностью (betweenness centrality) удаляются одна за другой. Каждое такое удаление изменяет структуру

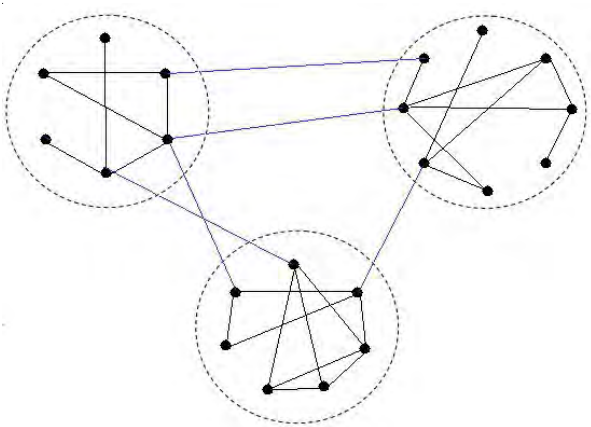


Рис. 2.17. Схематическое изображение сети из трёх сообществ

кратчайших путей в сети, а следовательно, и важность каждой связи, и поэтому эти параметры пересчитываются после каждого удаления. На некотором шаге сеть оказывается разделенной на два кластера – два самых больших сообщества, и далее процедура продолжается. В результате, получается дерево, в котором сообщества малых размеров включены в более большие сообщества. Распределение по размерам сообществ, выявляемых в результате этой процедуры, в большинстве реальных сетей подчинено степенному закону.

Модели случайных графов.

Случайную сеть (граф) можно определить не как единичную сеть, а как *статистический ансамбль*, в котором каждая конкретная сеть имеет определенную вероятность реализации, т.е. каждая сеть ансамбля имеет свой собственный статистический вес. Для моделирования сложных стохастических сетей используют модель «классический случайный граф», модель графа «Тесный мир» и модель «бесмасштабного» графа.

Модель «Классический случайный граф». Простейшими случайными сетями являются так называемые *классические случайные графы* (модель Эрдеша-Реньи (Erdos-Renyi)), в статистическом ансамбле которых все возможные графы с числом узлов N и числом связей L имеют *одинаковый статистический вес реализации*, т.е. для таких сетей вероятность существования связи между любыми двумя узлами одинакова.

Для случайных сетей Эрдеша-Реньи $\bar{l} \infty \log N$.

Ансамбли, в которых их статистические веса не изменяются во времени, не эволюционируют, называются *равновесными*. В *неравновесных* (способных к эволюции) ансамблях статистические веса изменяются во

времени и множество конфигураций также изменяется. *Растущие* сети являются неравновесными. Даже среди сетей с фиксированным числом связей возможны неравновесные сети.

Наиболее известны две модели классического случайного графа:

1) L ребер распределены произвольно и независимо между N вершинами графа;

2) фиксируется вероятность p, с которой может объединяться каждая пара вершин.

В пределе при $N \to \infty$ для обоих вариантов распределение порядков вершин определяется формулой Пуассона:

$$p_k = \frac{\bar{k}^k}{k!} e^{-\bar{k}},$$

где среднее значение порядка $\bar{k} = 2L/N$ для первой модели и $\bar{k} = pN$ – для второй модели.

Процесс построения случайного графа заключается в следующем. Вначале генерируется N изолированных вершин. К вершинам последовательно добавляются ребра, случайным образом соединяющие произвольные пары вершин. Образуется множество малых компонент графа[1]. Разрастание малых компонент приводит к образованию *гигантского кластера* связанных вершин, число которых составляет конечную долю полного числа N.

Образование гигантского кластера наблюдается только при условии, что вероятность p связывания вершин, постепенно возрастающая в процессе генерации, приобретает значение, превышающее критический порог. В результате произойдет спонтанное образование гигантского кластера, напоминающее конденсацию капли воды в пересыщенном паре. Такой процесс имеет ярко выраженный характер *фазового перехода*, в результате которого доля связанных вершин, принадлежащих гигантскому кластеру, определяется выражением:

$$G = 1 - \frac{1}{\bar{k}} = \sum_{n=1}^{\infty} \frac{n^{n-1}}{n!} \left(\bar{k} e^{-\bar{k}} \right)^n .$$

В классическом случайном графе гигантский компонент появляется при критической плотности ребер $d = 1$. До этого момента компонент имеет число вершин $\ln(N)$, а после начинает линейно увеличиваться с ростом N. В классическом случайном графе с фиксированным распределением

---

[1] Компонентом графа называется совокупность вершин, связанных между собой, но отдаленная от основной части графа.

порядка p_d условие возникновения гигантского кластера определяется неравенством:

$$\sum_{d=3}^{N} d(d-1)p_d > p_1.$$

В точке превращения распределения размеров случайного кластера, обладающего определенным распределением порядков вершины, значение спадает по степенному закону с показателем (-3/2). В окрестностях этой точки распределение размеров компонентов также следует степенному закону, который, однако, обрезается экспонентным хвостом. Такое поведение подобно явлению *перколяции*, где в критическом состоянии размеры компонентов также распределены по степенному закону.

Модель графа «Тесный мир». Существует обычно круг ближайших знакомых персоны, затем люди, знающие ближайших знакомых персоны (но не знающие непосредственно персону) и т.д. и т.п. В результате возникают цепочки, из которых ясно, что любой член общества опосредовано знаком с любым другим членом общества. В этом смысле мир является тесным и его модель носит название «Тесный мир».

Компьютерная модель «Тесного мира» была разработана Уотсом и Строгацем (Watts D. J., Strogatz S. H.). Её построение начинается с одномерной периодической цепочки, состоящей из N вершин (рис.2.18). Сначала каждая вершина соединяется q ребрами с другими вершинами,

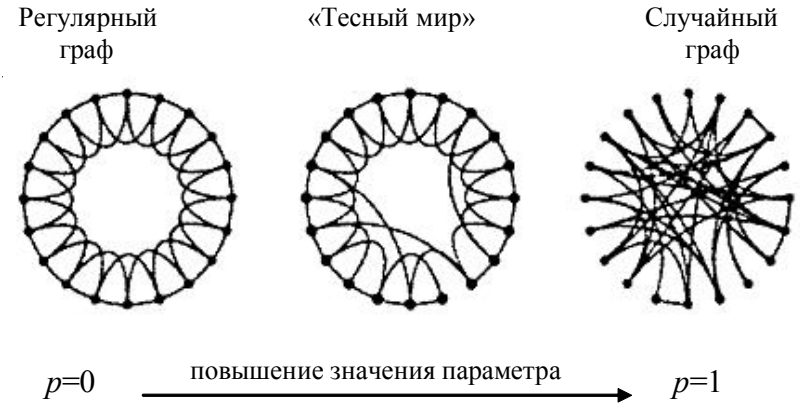


Рис. 2.18. Схема трансформации регулярной цепочки в граф «Тесного мира», а затем в случайный граф. Параметр p определяется вероятностью переброса ребер в случайные положения

где q – чётное положительное число. Затем с некоторой вероятностью p каждое ребро перебрасывается в произвольную позицию.

Компьютерное моделирование показало, что свойства “тесного

мира" обладают сети с высокой степенью кластеризации и малой средней длиной пути между узлами. Регулярные кристаллические решётки (например, треугольные) имеют высокое значение коэффициента кластеризации и большую среднюю длину пути. Классические случайные сети имеют низкое значение коэффициента кластеризации и небольшую среднюю длину пути.

Сети тесного мира можно рассматривать как суперпозицию регулярных решёток и классических случайных сетей, поскольку они обладают высокой кластеризацией и низким значением средней длины пути.

Сети тесного мира – весьма специфический вид сетевых структур. Свойством тесного мира обладают практически все реальные сети. Формально, сети со свойством «тесного мира» имеют бесконечную размерность. Реальные сетевые структуры, как правило, имеют и высокий коэффициент кластеризации.

Модель «безмасштабного» графа. Распределение числа связей по закону Пуассона имеет строгий максимум около среднего значения. Однако для многих реальных сетей, например, Интернета и его виртуального двойника World Wide Web такого среднего значения не существует. В таких сетях небольшое число узлов содержит очень большое число связей, а огромное число узлов содержит лишь несколько связей. Соответствующее вероятностное распределение подчиняется степенному закону

$$P(q) \infty q^{-\lambda}.$$

Степенная функция является признаком самоподобия системы, которая не обладает каким-либо масштабом изменения характерных величин. Поэтому такие графы называются *безмасштабными*, а сети получили название *безмасштабных сетей* (scale free networks) (отсутствие узла с типичным числом связей в такой сети). Отличие таких графов в том, что классический случайный граф имеет намного более быстро спадающий хвост $1/d! \cong d^{-d} 1/d! \approx d^{-d}$, чем безмасштабный[2].

Для безмаштабных сетей $l \infty \log\log N$, где N – число узлов в сети. Для случайных классических сетей – $l \infty \log N$. Получены свидетельства того, что функциональные связи в мозге человека образуют безмасштабные сети. Полная сеть содержит до 31503 узла.

Модель Барабаши - Альберт. Барабаши и Альберт показали, что для возникновения и эволюции безмасштабных сетей необходимы два условия:

---

[2] Вероятность наличия в графе узлов с возрастающим порядком быстрее снижается в классическом случайном графе, чем в безмасштабном (степенном).

1. Рост. Начиная с небольшого числа узлов, на каждом временном шаге добавляется один новый узел с m (m) связями, которые соединяют этот новый узел с различными уже существующими узлами.

2. Предпочтительное присоединение (Preferencial attachment). Когда выбираются узлы, к которым присоединяется новый узел, предполагается, что вероятность P, с которой новой узел будет соединяться с уже существующим узлом i, зависит от числа связей, которыми этот узел уже связан с другими узлами, так что

$$P(q_i) = \frac{q_i}{\sum_j q_j} .$$

Безмасштабные сети - это одно из проявлений феноменологии критических явлений в сложных системах, поскольку их структура подчиняется степенному закону.

2.1.3. Перколяция. Основные понятия

Существующие методы моделирования сетей ориентированы на анализ систем с регулярной структурой и не очень приспособлены для моделирования сетей, имеющих случайную структуру. Одним из подходов, который может помочь преодолеть существующие трудности, является использование теории перколяции.

Перколяционные процессы в природе и технологиях.

Перколяция является удобной моделью для описания широкого класса явлений, которые принято называть критическими. Техника, развитая для теории перколяции, имеет многочисленные приложения в задачах о случайных процессах. Для описания перколяции рассмотрим пример. Представим себе большой противень, на котором случайным образом размещены кружочки жидкого теста разного диаметра. Затем противень с печеньем помещается в духовку. Допускается, что в процессе выпечки каждая капля теста может расплыться. Если два печенья соприкасаются, то они сливаются и образуется одно печенье. И если не следить за выпечкой, то можно получить одно гигантское печенье, которое будет занимать значительную часть противня (рис. 2.19).

Если такое образование простирается от одного края до другого, то говорится, что оно «просачивается» (перколирует) сквозь структуру.

Типы перколяций.

Наиболее распространенными задачами теории перколяции являются *решёточные* задачи: *задача узлов и задача связей*. Представим шах-

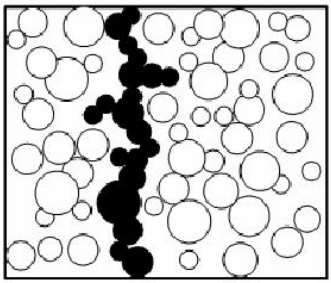


Рис. 2.19. Путь из перекрывающихся кружков (закрашены) соединяет верхнюю и нижнюю стороны «противня»

матную доску как квадратную решетку. Предположим, что каждый квадрат, или «ячейка» этой решетки может находиться в двух состояниях: «занято» или «пусто». Квадратная решетка может быть бесконечной либо с заданным размером, например 3Ч3 (рис. 2.20). Закрасим («займем») часть квадратов черным цветом. В нашем случае их 2,3. Доля закрашенных квадратов составляет $p = N$чёрн$/N$=1/2,3. Каким образом выбираются квадратики для закрашивания? Во-первых, можно выбирать квадратики случайно и независимо; во-вторых, можно ввести какие-либо правила. В первом случае говорят о *случайной перколяции* (математики называют её *перколяцией Бернулли*), во втором – *коррелированной*.

Занятые ячейки либо изолированы друг от друга, либо образуют группы занятых ячеек решетки, связанных с ближайшим соседом по стороне ячейки (рис.2.20 а,б,в,г). Такие группы называют кластерами. *Кластер* (cluster – гроздь) – *это группа связанных объектов*, состоящая из ближайших соседей.

Простой способ занятия квадратиков (ячеек решетки) основан на использовании генератора случайных чисел. Вся процедура сводится к тому, чтобы сгенерировать случайное число, а затем занять ячейку решётки, если случайное число меньше p. Эта процедура выполняется для каж-

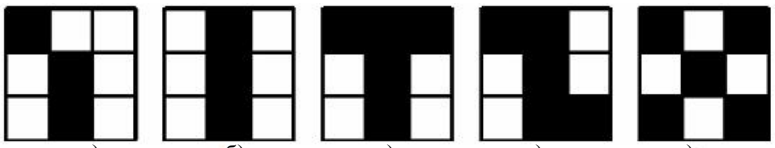


Рис. 2.20. Пример ячеечного перколяционного кластера на квадратной решётке со стороной L=3: а) две ближайшие занятые ячейки (закрашенные квадраты в этом случае являются одним кластером; б) три занятые ячейки образуют один кластер; в) и г) кластер состоит из 5-ти ячеек; д) кластеров нет

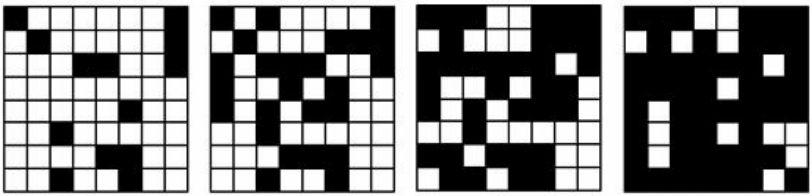
дой ячейки решетки. Если вероятность занятия ячейки мала, то можно ожидать, что будут присутствовать только небольшие изолированные кластеры (рис.2.21,а). Если $p \approx 1$, то ожидается, что большинство занятых ячеек образуют один большой кластер, который протянется от одной стороны решетки до другой (рис.2.21,г). Такой кластер называют *соединяющим кластером*. Посмотрим, что произойдет для промежуточных значений p, например, для p от 0,4 до 0,6 (рис.2.21,б,в).

В пределе бесконечной решётки существует вполне определённая "пороговая" вероятность p, такая, что для $p \geq p_c$ сущетсвует один соединяющий кластер, или *путь*; для $p < p_c$ нет ни одного соединяющего кластера и все кластеры *конечны*. Такая модель просачивания называется *ячеечной перколяцией*.

Характерной особенностью перколяции является связность. Поскольку связность обнаруживает качественное изменение при конкретном значении некоторого параметра, который можно менять непрерывно, то переход из состояния, не содержащего соединяющий кластер, в состояние с одним соединяющим кластером представляет собой *фазовый переход*.

Ещё одна модель просачивания – *цепная перколяция*. Заменим шахматную доску квадратной сеткой (решеткой, бесконечным регулярным графом). Точки пересечения линий называют узлами (вершинами). Сами линии – связями (ребрами). Простейшей задачей цепной перколяции является перколяция сквозь Бернуллиевы решётки.

Представим себе большую квадратную решётку, составленную из двух видов стержней: одни сделаны из изолирующего винила, другие – из электропроводной меди. Такая решётка может считаться *решёткой Бернулли*, если каждый стержень выбран совершенно случайно, неза-



p=0,2 а) *p*=0,4 б) *p*=0,6 в) *p*=0,8 г)

Рис. 2.21. Примеры ячеечных перколяционных кластеров на квадратной решетке со стороной $L = 8$ для значений $p = 0{,}2$, 0,4, 0,6 и 0,8: в среднем доля занятых ячеек (закрашенные квадраты) равна p; для значения $p = 0{,}6$ существует кластер, который соединяет стороны решетки в горизонтальном направлении, но не в вертикальном; при значении $p = 0{,}8$ кластер соединяет стороны решетки и по вертикали, и по горизонтали

висимо от других стержней, причём вероятность выбора проводящего стержня равна p. Наибольшие скопления связанных между собой медных или виниловых стержней образуют, соответственно, медные или виниловые кластеры. Если решётка содержит хотя бы одну непрерывную цепочку медных стержней, электрический ток сможет пройти всю решётку насквозь, от одного края до другого. В таких случаях говорят, что решётка перколирует. Все стержни, находящиеся в неразрывном электрическом контакте одновременно с верхним и нижним краями решётки образуют соединяющий кластер, а стержни, непосредственно участвующие в передаче, составляют так называемую «магистраль» кластера. Наиболее распространенными задачами теории перколяции являются решёточные задачи: *задача узлов* и *задача связей*.

В задаче связей ищут ответ на вопрос: какую долю связей нужно удалить (перерезать), чтобы сетка распалась на две части? В задаче узлов блокируют узлы (удаляют узел, перерезают все входящие в узел связи) и ищут, при какой доле блокированных узлов сетка распадется.

Квадратная сетка является только одной из возможных моделей. Можно рассматривать перколяцию на треугольной, шестиугольной сетках, деревьях, трёхмерных сетках, например, кубической или в пространстве с размерностью больше 2,3. Сетка не обязательно должна быть регулярной. Рассматриваются процессы и на случайных сетках.

Порог перколяции.

Теория перколяции позволяет описать процессы самой разной природы, когда при плавном изменении одного из параметров системы (концентрации чего-то) свойства системы меняются скачком. Одним из основных вопросов, на которые пытается ответить теория перколяции: при какой доле p_c закрашенных квадратов возникает цепочка черных квадратов, соединяющая стороны решетки?

Для сетки конечного размера такие цепочки могут возникать при разных концентрациях (рис. 2.20 и 2.21). Однако если размер решётки устремить к бесконечности, то критическая концентрация станет вполне определённой. Это строго доказано. Такую *критическую концентрацию* называют *порог перколяции*.

Рассмотрим квадратную решетку со стороной L и присвоим каждой ячейке этой решётки случайные числа от нуля до единицы. Ячейка занимается, если присвоенное ей случайное число меньше p. Так порождается ячеечная перколяционная конфигурация. В вероятностном подходе порог перколяции p_c определяется как такая вероятность p, при которой появляется первый *бесконечный* (*соединяющий*) *кластер* на бесконечной решётке. Для конечной решётки со стороной L, которую

можно промоделировать на компьютере, всегда существует *ненулевая* вероятность того, что будет появляться соединяющий кластер, связывающий одну сторону решетки с другой. Для малых значений p эта вероятность порядка pL. По мере увеличения L вероятность появления соединяющего кластера стремится к нулю и для достаточно малых значений p будут существовать только конечные кластеры. Поскольку правило «протекания» необходимо применить для конечной решетки, определим $p_c(L)$ как среднее значение p, при котором впервые появляется соединяющий кластер. Для конечной решетки определение протекания произвольно и, следовательно, вычисленное значение p_c зависит от критерия протекания. Например, можно определить соединяющий путь одним из способов: он связывает решетку в выбранном направлении (в вертикальном либо в горизонтальном); соединяет решетку в обоих направлениях. Все эти правила протекания должны приводить к одному и тому же экстраполированному значению p_c при $L \to \infty$.

Кроме квадратной решетки наиболее известной двумерной решёткой является *треугольная*. Существенным различием между квадратной и треугольной решётками является число ближайших соседей.

Характеристики перколяционных сетей.

Определим, какие основные характеристики вычисляют в перколяционных моделях. Итак, при перколяции существует порог перколяции p_c, и появляется соединяющий путь, или кластер, при $p \geq p_c$ (p – вероятность занятия ячейки). Более полную информацию можно получить из распределения кластеров по размерам, определяемого формулой

$$n_S = N_S / N, \qquad (2.1.1)$$

где N – полное число ячеек решетки; N_S – число кластеров размером s.

При $p \geq p_c$ соединяющий кластер исключается из n_S. По историческим причинам под размером кластера подразумевается число ячеек в кластере, а не его пространственная протяженность. Анализ рис. 2.21,а показывает, что для проведенного одного испытания $n_S(p=1,2) = 5/64, 1/64$ и $2/64$ для $s = 1, 2$ и 3 соответственно и равно нулю в других случаях, поскольку представляет собой концентрацию занятых ячеек, а – концентрацию занятых ячеек в кластерах размером s, величина

$$\omega_s = \frac{s n_s(p)}{\sum_s s n_s(p)} \qquad (2.1.2)$$

является вероятностью того, что занятый узел, выбранный случайным образом, принадлежит кластеру размером s. Следовательно, средний размер кластера s определяется как

$$s = \sum_s s\omega_s = \frac{\sum_s s^2 n_s(p)}{\sum_s s n_s(p)}. \qquad (2.1.3)$$

Так, средний размер кластера на примере рис.2.21,а равен $s = 27/12{,}3$. Обозначим через N_∞ число ячеек в соединяющем кластере, тогда вероятность того, что случайным образом выбранный занятый узел принадлежит соединяющему кластеру (или иначе эту величину называют *параметром порядка*) выражается отношением

$$P_\infty = \frac{N_\infty}{N_{занятых}}. \qquad (2.1.4)$$

В случае бесконечной решетки $(p) = 0$ при $p<p_c$ и $(p) = 1$ при $p = 1$. Анализ рис. 2.21, в показывает, что $(p = 0{,}6) = 25/36$ для приведённой конфигурации.

Итак, кластер – это цепочка связанных объектов. Кластер, соединяющий две противоположные стороны системы, называется *соединяющим* (перколяционным, бесконечным или стягивающим). Ниже порога перколяции имеются только кластеры конечного размера. Остов кластера – «токопроводящая» часть кластера. Мёртвые концы – части кластера, соединенные с остовом посредством одного узла (связи, стороны ячейки). Мертвые концы составляют большую часть кластера, однако не участвуют в проводимости.

Красные связи – одиночные связи, при разрушении которых перколяционный кластер перестает «проводить ток». *Скелет кластера* – объединение всех кратчайших путей от данного узла до узлов на заданном расстоянии. *Эластичный остов* – объединение всех кратчайших путей между двумя данными узлами. *Оболочка,* или *внешний периметр* состоит из тех узлов кластера, которые соприкасаются с пустыми узлами и соединены с бесконечностью посредством пустых узлов. *Полный периметр* включает также пустоты внутри кластера. Все эти подструктуры описываются различными *фрактальными* размерностями, для ряда из них на сегодняшний день значения получены только путём компьютерного моделирования.

Критические показатели и масштабная инвариантность.

В окрестности порога перколяции поведение системы тесно связано с наличием больших, но конечных кластеров. Более прямой способ наблюдения влияния длины связан с введением характерного линейного размера или длины корреляции (средней длины связности) $\xi(p)$. Для больших значений L $\xi(p)$ – возрастающая функция в диапазоне $p<p_c$ и убывающая для $p > p_c$ (рис. 2.22).

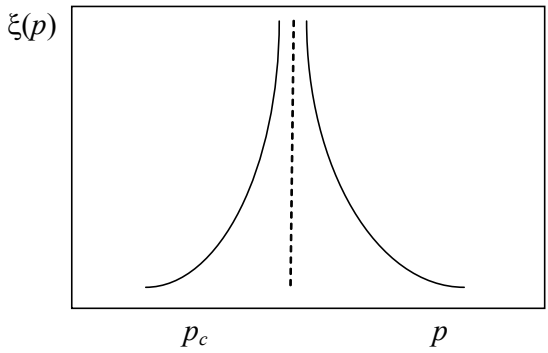


Рис. 2.22. Качественная зависимость длины корреляции $\xi(p)$ от p: стремление $\xi(p)$ к бесконечности в критической области происходит по степенному закону

Качественное поведение функции ξ соответствует физическому представлению о кластерах: по мере приближения p к p_c возрастает вероятность того, что два занятых узла находятся в одном кластере. Такое качественное рассмотрение наталкивает на мысль, что в пределе $L \to \infty$ $\xi(p)$ сингулярна в критической области $|p - p_c| << 1$.

Другой рассматриваемой величиной является средний размер кластера $s(p)$.

На конечной решетке не могут происходить истинные фазовые переходы, описываемые расходящимися физическими величинами. Вместо этого ξ и s достигают конечного максимума при значении $p = p_c(L)$.

Фазовый переход определяется только для бесконечных систем. Используя терминологию теории фазовых переходов, можно сказать, что ячеечная и цепная перколяции принадлежат одному классу универсальности и их критические показатели равны.

Другой важной идеей в области критических явлений считается существование зависимостей между критическими показателями. Задача определения, казалось бы, относительно простая. Однако за редкими исключениями не удается вычислить аналитически – необходимо довольно сложное численное моделирование.

К задачам, решаемым в рамках теории перколяции и анализа сложных сетей относятся такие, как определение предельного уровня проводимости (пропускной способности), изменения длины пути и его траектории (извилистости, параллельности) при приближении к предельному уровню проводимости, количества узлов, которые необходимо удалить, чтобы нарушить связанность сети.

Теория перколяции применима в различных областях науки. Термин перколяция, означающий *протекание*, отражает, соответственно, и воз-

можность моделирования протекания жидкости, например, воды или нефти в почвенных средах. В физике теория перколяции применяется при расчёте распространения токов в случайной сетке проводников. В разделе материаловедения «теория разрушений» она используется для изучения распространения трещин при деформации материалов. В медицине теорию перколяции используют для моделирования распространения инфекций при заболеваниях организма.

Хорошие перспективы применения имеет эта теория в информатике и вычислительной технике. Если, например, необходимо оградить глобальную сеть от компьютерного вируса, то достаточно, чтобы каждый отдельный узел этой сети был недоступен для вируса с вероятностью, хотя бы немного превосходящей критическую вероятность. В этом случае сеть в целом будет гарантированно оставаться связной и работоспособной. Частичные потери функциональности сети (например, потери её суммарной производительности) определяются при этом распределением размеров кластеров, поражаемых вирусом.

Перколяцию рассматривают и на случайных сетях Эрдоша-Реньи, Уаттса-Строгатса и т.п. Так, перколяционный подход использовали исследователи из Калифорнийского университета для разработки быстрого алгоритма маршрутизация в пиринговых сетях по принципу пчелиного роя. Алгоритм использует принцип порога перколяции связей между тесно связанными узлами в случайным образом сформированных масштабируемых сетях, таких как интернет.

Для определения критической вероятности в подобных нерегулярных сетях и для установления законов распределения контактных кластеров требуется проводить статистическое моделирование решёток с *нерегулярной структурой*.

Вопросы и задания

1. Изобразите с помощью графа договорные отношения между предприятиями А, Б, В, Г, Д, Е, если к рассматриваемому моменту:

- предприятие А установило договорные отношения со всеми другими предприятиями;
- Б установило с Г и Д;
- В установило со всеми предприятиями, кроме предприятия Е. Сколько вершин и сколько ребер имеет полученный граф?

2. Сколько ребер нужно провести чтобы достроить граф, изображённый на рис. 2.23 до полного?

3. Сколько существует путей из вершины А в вершину Д на графе рис. 2.23.

4. Найдите простые пути на графе рис. 2.23.

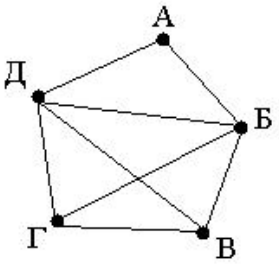


Рис. 2.23. Неполный граф

5. Укажите степени вершин графа на рис. 2.23.

6. Дайте определение изоморфного графа и укажите, какой из графов, изображенных на рис. 2.24, не является изоморфным с другими.

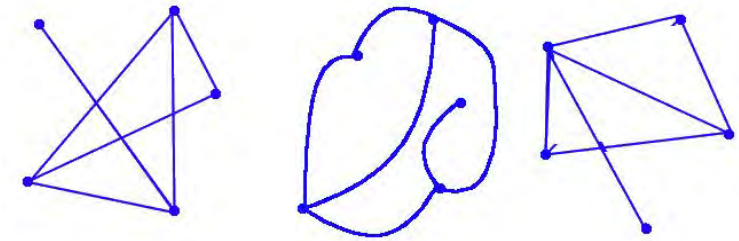


Рис. 2.24. Примеры графов

7. Среди графов, изображенных на рис. 2.24 найдите плоский граф и дайте ему определение.

8. Приведите пример нулевого графа и дайте ему определение.

9. Приведите собственный пример несвязного графа и моста на нём.

10. Между четырьмя государствами были подписаны двухсторонние договорные обязательства. Каждый договор был подписан президентами обоих договаривающихся государств. Сколько всего подписей фигу-

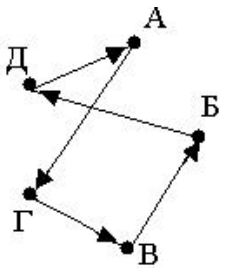


Рис. 2.25. Ориентированный граф

рировало в договорах, если каждый договор подписывался в двух экземплярах? Укажите закономерность, которая позволяет найти число подписей.

11. Определите степени входа и выхода всех вершин ориентированного графа на рис. 2.25.

12. Дайте определение длины пути и найдите её значение от вершины Д до всех остальных вершин графа, изображенного на рис. 2.25.

13. Дайте определения пути и цикла на графе и найдите их, если они есть на графе рис. 2.25.

2.2. Системы и сети массового обслуживания

Система массового обслуживания – одна из основных моделей, используемых инженерами-системотехниками. В терминах *систем массового обслуживания* (СМО) описываются многие реальные системы: вычислительные системы, узлы сетей связи, системы посадки самолетов, магазины, производственные участки – любые системы, где возможны очереди и (или) отказы в обслуживании.

В вычислительной системе роль обслуживающего прибора играет ЭВМ, роль заявок – решаемые задачи. Источником заявок служат терминалы пользователей. Моментом выдачи заявки является момент нажатия клавиши для подачи директивы о запуске задачи на решение. Операционная системы ЭВМ исполняет роль диспетчера: определяет очередность решения задач. В роли ячеек буфера выступают ячейки памяти ЭВМ, хранящие сведения о задачах, требующих решения.

В системе разгрузки судна другой пример реальной системы, источниками заявок являются направления, откуда прибывают суда. Момент выдачи заявки – это момент прибытия судна в зону морского порта для разгрузки/погрузки. Обслуживающим прибором является причал вместе с персоналом и техническими средствами, организующими разгрузку/погрузку. Роль буфера играет акватория порта.

В телекоммуникационной сети в качестве отдельной СМО оказывается коммутатор, в котором источниками заявок являются входящие каналы, момент выдачи заявки – поступление на вход порта коммутатора очередного пакета (кадра), обслуживающим прибором оказывается коммутационная схема, в роли ячеек буфера выступают накопители портов.

Целью использования СМО как модели является анализ качества функционирования указанных и других систем-оригиналов, связанных с распределением или/и расчетом того или иного ресурса.

Как *модель* СМО рассматривается в теории массового обслуживания (другое название – теория очередей). Теория массового обслуживания связана с разработкой и анализом математических, т.е. абстрактных

моделей, которые описывают процесс обслуживания некоторых объектов, поступающих на вход обслуживающего прибора в виде некоторого потока и образующего в общем случае очередь на входе обслуживающего прибора.

Поскольку рассматриваются абстрактные модели, совершенно не важна природа обслуживаемых объектов и их физические свойства (будь то вызовы, управляющие или информационные кадры в сети связи или посетители магазина, или детали на автоматической линии и т.п.). Существенным являются моменты появления этих объектов, правила и законы (математические) их обслуживания, так как от этих моментов и законов зависит адекватное отображение эволюции моделируемого объекта во времени. Поэтому, когда говорят о методах анализа очередей, имеют в виду математические (абстрактные) модели, а из контекста всегда должно быть ясно, для исследования какой реальной системы применяются эти модели.

Термин «массовое обслуживание» предполагает многократную повторяемость ситуаций (много прибывших в систему и обслуженных заявок, большое число находящихся в эксплуатации аналогичных систем) и статистическую устойчивость картины [5]. Усложнение структур и режимов реальных систем затрудняет применение классических методов теории массового обслуживания ввиду возрастающей размерности решаемых задач, что особенно характерно для систем с сетевой структурой. Одним из возможных путей преодоления размерности является использование моделей в форме сетей массового обслуживания (СеМО).

СеМО представляет собой совокупность конечного числа обслуживающих узлов, в которой циркулируют заявки, переходящие в соответствии с маршрутной матрицей из одного узла в другой. При этом отдельные СМО отображают функционально самостоятельные части реальной системы, связи между СМО связи между этими частям, а требования, циркулирующие по СеМО, – составляющие материальных потоков (сообщения (пакеты) в коммуникационной сети, задания в мультипроцессорных системах, контейнеры грузопотоков и т.п.).

СеМО используют для определения важнейших системных характеристик информационных систем: производительности; времени доставки пакетов; вероятности потери сообщений и блокировки в узлах; области допустимых значений нагрузки, при которых обеспечивается требуемое качество обслуживания и др. Наиболее разработана теория экспоненциальных СеМО. Сеть называется экспоненциальной, если входящие потоки требований в каждую СМО пуассоновские, а времена каждого этапа обслуживания, реализуемого на любой СМО сети, имеют экспоненциальное распределение. Это позволяет считать, что этапы обслуживания независимы между собой и не зависят ни от параметров

входящего потока, ни от состояния сети, ни от маршрутов следования требований.

Теорию экспоненциальных СеМО широко применяют как для исследования сетей передачи данных, так и для исследования мультипроцессорных вычислительных систем (ВС). Разработаны практические формы расчета вероятностно-временных характеристик (ВВХ) таких сетей и систем. СеМО – это, прежде всего, совокупность взаимосвязанных систем массового обслуживания. Поэтому необходимо вспомнить основные особенности этих систем.

2.2.1. Система массового обслуживания как модель

Основными элементами системы массового обслуживания являются входной поток заявок, очередь, прибор (канал) обслуживания.

Заявки (требования) на обслуживание поступают через постоянные или случайные интервалы времени. *Приборы* (каналы) служат для обслуживания этих заявок. Обслуживание длится некоторое время, постоянное или случайное. Если в момент поступления заявки все приборы заняты, заявка помещается в *очередь* (буфер) и ждёт там начала обслуживания. Если все места в очереди заняты, заявка получает *отказ* в обслуживании и теряется. *Вероятность потери заявки* (вероятность отказа) – одна из основных характеристик СМО. Другие характеристики: *среднее время ожидания* начала обслуживания, *средняя длина очереди*, *коэффициент загрузки прибора* (доля времени, в течение которого прибор занят обслуживанием) и т.д.

В зависимости от объема очереди различают *СМО с отказами*, где нет буфера, *СМО с ожиданием*, где буфер не ограничен (например, очередь в магазин на улице) и *СМО смешанного типа*, где буфер имеет конечное число заявок. В СМО с отказами нет очереди, в СМО с ожиданием нет потерь заявок, в СМО смешанного типа то и другое возможно.

Иногда различают заявки по их *приоритету*, т.е. по важности. Заявки высокого приоритета обслуживаются в первую очередь. *Абсолютный приоритет* дает право прервать обслуживание менее важной заявки и занять её место в приборе (или в буфере, если все приборы заняты столь же важными заявками). Вытесненная заявка либо теряется, либо поступает в буфер, где ждет дообслуживания. Иногда приходится возобновлять обслуживание вытесненной заявки с начала, а не продолжать с точки прерывания. Если заявка вытеснена из буфера, она, естественно, теряется. Примером заявки с абсолютным приоритетом является судно, получившее пробоину и нуждающееся в срочной разгрузке. В вычислительных системах абсолютным приоритетом обладают команды оператора. *Относительный приоритет* дает право первоочередного занятия осво-

бодившегося прибора. Он не дает право на вытеснение заявки из прибора или буфера. Лица, имеющие льготы при обслуживании в кассе, у врача и т.п., как правило, имеют относительный приоритет. Абсолютный и относительный приоритеты различаются и моментом действия: абсолютный реализуется в момент поступления, а относительный - в момент освобождения прибора.

Различают фиксированные и динамические приоритеты. Фиксированные приоритеты чаще называют дисциплиной обслуживания.

Дисциплина обслуживания задает порядок выбора из очереди в освободившийся прибор заявок одинакового приоритета. Выделим следующие дисциплины: FIFO (First Input - First Output): первым пришёл – первым обслужен, LIFO (Last Input - First Output): последним пришёл – первым обслужен, RAND (Random): случайный выбор из очереди. В быту обычно действует дисциплина FIFO. Дисциплина LIFO реализуется в буфере, организованном по принципу стека. Такая дисциплина может оказаться целесообразной, например, при передаче информации, если её ценность быстро падает со временем.

В теории массового обслуживания важным является понятие *случайного потока* как некоторой последовательности событий, наступающих в случайные моменты времени.

Случайный поток может быть задан функцией распределения величины промежутка (интервала) времени между моментами наступления событий:

$$\tau_j = t_j - t_{j-1}, P(\tau_j \le t).$$

Если величины τ_j независимы в совокупности, то поток обладает *ограниченным последействием.* В случае $P(\tau_j \le t) = P(\tau \le t)$ для всех $j \ge 2$ поток является *рекуррентным*.

Рекуррентный поток, для которого $P(\tau \le t) = 1 - \exp(-\lambda t)$, называется *пуассоновским*. Для такого потока вероятность наступления за промежуток времени $[0,t]$ n событий есть $P_n(t) = [(\lambda t)^n/n!]\exp(-\lambda t)$, а математическое ожидание числа событий, наступивших за время t, λt, где λ – среднее число событий, наступающих в единицу времени.

Пуассоновский поток характеризуется отсутствием последействия. Если кроме этого выполняются условия *стационарности* и *ординарности,* то пуассоновский поток будет *простейшим.*

Для стационарного потока распределение не зависит от положения интервала τ на оси времени и зависит только от длительности τ. Отсутствие последействия означает независимость числа событий в неперекрывающихся интервалах τ_j.

Свойство ординарности заключается в том, что вероятность появления более одного события на бесконечно малом интервале имеет поряд-

ок малости выше, чем вероятность появления одного события на этом интервале.

Величину λ в случае пуассоновского потока называют *интенсивностью* потока событий. Если $\tau_j = \tau = \text{const}$, то поток является регулярным или детерминированным.

Для обозначения типа СМО используется система обозначений, имеющих вид $\Delta|\Theta|\Xi|\Omega$. Здесь Δ – обозначение закона распределения вероятностей для интервалов поступления заявок, Θ – обозначение закона распределения вероятностей для времени обслуживания, Ξ – число каналов обслуживания, Ω – число мест в очереди.

Обозначение законов распределения в позициях Δ и Θ выполняется обычно буквами из следующего списка:

M – экспоненциальное;

E^k – эрланговское порядка k;

R – равномерное;

D – детерминированное (постоянная величина);

G – произвольное (любого вида) и т.д.

Если число мест в очереди не ограничено, то позиция Ξ не указывается. Например, $M|M|1$ означает простейшую СМО (оба распределения экспоненциальные, канал обслуживания один, очередь не ограничена). Обозначение $R|D|2|100$ соответствует СМО с равномерным распределением интервалов поступления требований, фиксированным временем их обслуживания, двумя каналами и 100 местами в очереди. В этой СМО заявки, приходящие в моменты, когда все места в очереди заняты, покидают систему (т.е. теряются).

Если в СМО поступает n потоков заявок (у каждого потока свой приоритет), то Δ и Θ приписывают число n в виде индекса. Например, $M2|M2|1$ обозначает СМО с двумя потоками заявок, на входе имеющими экспоненциальное распределение, с экспоненциальным временем обслуживания, своим для каждого потока. В системе $M2|M|1$ время обслуживания всех заявок имеет одно и то же распределение. В случае нескольких входных потоков, имеющих разные приоритеты, необходимо дополнительно указывать типы приоритетов - абсолютные, относительные.

2.2.2. Экспоненциальная система массового обслуживания

Одноканальная однородная экспоненциальная СМО.

Одноканальная экспоненциальная СМО определяется следующими свойствами. СМО имеет канал. В СМО приходят *заявки*. Если СМО пустая (нет заявок), то приходящая заявка занимает канал. Приходящая в непустую СМО заявка становится в очередь последней. Любая занявшая канал

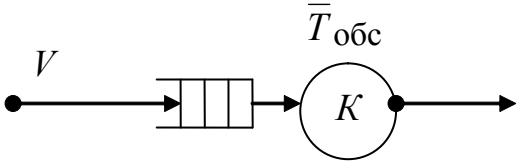


Рис. 2.26. Одноканальная СМО

заявка обслуживается, освобождает канал и уходит из СМО. Если в момент ухода очередь непустая, первая в ней заявка выходит из очереди и занимает канал. Кружком обозначен канал *K*, тремя прямоугольниками - очередь (рис. 2.26).

Стрелки указывают направление движения заявок, точки у стрелок – вход и выход СМО. Приходы заявок образуют пуассоновский поток событий. Это означает, что время между приходами любых двух последовательных заявок есть независимая случайная величина с экспоненциальной функцией распределения вероятностей:

$$F(X) = 1 - e^{-VX}. \qquad (2.2.1)$$

Параметр *V* есть интенсивность потока заявок, т.е. среднее число заявок, приходящих в единицу времени, равно *V*. В дальнейшем интенсивность прихода заявок в СМО будем обозначать через λ. Время обслуживания заявки - тоже независимая случайная величина с экспоненциальной функцией распределения вероятностей вида (2.2.1). Но параметр *V* в этом случае имеет другое значение. Будем обозначать его через μ. Величину $1/\mu$, равную среднему времени обслуживания заявки, обозначим через $\overline{T}_{обс}$.

В виде одноканальной экспоненциальной СМО можно промоделировать, например, периферийное устройство мультипрограммной вычислительной системы. Тогда приходы заявок будут соответствовать обращениям программ к устройству для выполнения операции ввода или вывода информации; λ будет интенсивностью таких обращений, $\overline{T}_{обс}$ – средним временем выполнения требуемой операции. Одноканальная экспоненциальная СМО задается параметрами λ, $\overline{T}_{обс}$. Цель её анализа заключается в расчёте характеристик, важнейшие из которых следующие:

– коэффициент загрузки ρ;
– средняя длина *L* очереди;
– среднее число *M* заявок в СМО;
– среднее время ожидания обслуживания $\overline{T}_{ож}$;
– среднее время $\overline{T}_{пр}$ пребывания заявки в СМО .

Коэффициент загрузки рассчитывается по формуле

$$\rho = \lambda \cdot \overline{T}_{обс}. \qquad (2.2.2)$$

Если выполняется условие:

$$\rho \le 1, \qquad (2.2.3)$$

то существует стационарный режим функционирования СМО. В стационарном режиме все вероятностные характеристики системы являются постоянными во времени величинами. Сами происходящие в СМО события остаются при этом случайными. Если (2.2.3) не выполняется, то стационарного режима у СМО не существует.

В стационарном режиме среднее число M заявок в СМО постоянно, поэтому среднее число заявок, приходящих в СМО в единицу времени, равно среднему числу заявок, в единицу времени, уходящих из СМО. Следовательно, в стационарном режиме интенсивность потока уходящих заявок равна λ. Коэффициент загрузки ρ в стационарном режиме есть:

а) среднее значение той части единицы времени, в течение которой канал занят;

б) вероятность того, что канал занят;

в) среднее число заявок в канале.

В последующем речь будет идти только о стационарных значениях характеристик.

Средняя длина очереди (среднее число заявок в очереди) в одноканальной экспоненциальной СМО рассчитывается по формуле

$$L = \frac{\rho^2}{1 - \rho}. \qquad (2.2.4)$$

Среднее число M заявок в СМО равно сумме среднего числа L заявок в очереди и среднего числа ρ заявок в канале:

$$M = \frac{\rho}{1 - \rho}. \qquad (2.2.5)$$

Заявка перемещается в очереди в среднем с постоянной скоростью. Среднее число переходов заявки в очереди на одно место вперед за единицу времени равно λ. При такой скорости перемещения L переходов произойдет за время, равное в среднем

$$\overline{T}_{ож} = \frac{\overline{T}_{обс} \cdot \rho}{1 - \rho}. \qquad (2.2.6)$$

Формула (2.2.6) дает среднее время прохождения заявки через очередь. Это есть среднее время ожидания.

Среднее время пребывания заявки в СМО есть сумма среднего времени ожидания и среднего времени обслуживания заявки:

$$\overline{T}_{\text{пр}} = \frac{\overline{T}_{\text{обс}}}{1-\rho}. \qquad (2.2.7)$$

Вероятность наличия в системе k требований определяется с помощью геометрического закона распределения в виде

$$(1-\rho)\cdot\rho^{k}, k = 0,1,2,...$$

Характеристики (2.2.2) – (2.2.7) могут давать ценную информацию о моделируемой в виде СМО системе. Пусть, например, СМО изображает периферийное устройство вычислительной системы. Тогда ρ равен коэффициенту использования устройства, (1- ρ) – коэффициенту простоя. Необходимо, чтобы коэффициент использования был достаточно велик. Величина характеризует среднее время, в течение которого программы ожидают освобождения устройства. В это время программы фактически "простаивают". Желательно, чтобы оно было достаточно мало.

Многоканальная экспоненциальная СМО отличается от одноканальной числом каналов, которых в ней более одного. Приходящая заявка становится в очередь, если все каналы заняты. В противном случае заявка занимает свободный канал.

Многоканальная экспоненциальная СМО.

Многоканальная экспоненциальная СМО задается тремя параметрами: интенсивностью V прихода заявок, средним временем обслуживания $\overline{T}_{\text{ож}}$ и числом K каналов (рис. 2.27).

Формулы для расчёта характеристик многоканальной экспоненциальной СМО немногим сложнее (2.2.2) – (2.2.7).

Коэффициент загрузки определяется в виде:

$$\rho = \frac{\lambda\overline{T}_{\text{обс}}}{K}. \qquad (2.2.8)$$

Его значение должно отвечать условно стационарности (2.2.3).

Средняя длина очереди в блоке ожидания:

$$L = \beta_0 \frac{(\lambda\overline{T}_{\text{обс}})^{K+1}}{K!\cdot K\left(1-\frac{\lambda\overline{T}_{\text{обс}}}{K}\right)^{2}}, \qquad (2.2.9)$$

где β_0 – стационарная вероятность того, что в СМО нет заявок. Эта вероятность определяется в виде:

$$\beta_0 = \frac{1}{\frac{(\lambda \overline{T}_{обс})^K}{K!(1-\frac{\lambda \overline{T}_{обс}}{K})} + \sum_{m=0}^{K-1} \frac{(\lambda \overline{T}_{обс})^m}{m!}}. \qquad (2.2.10)$$

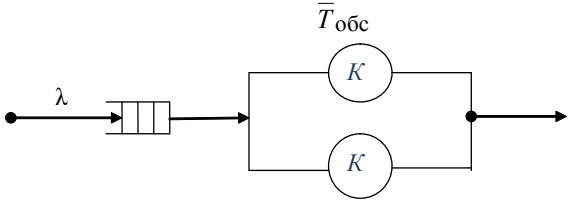


Рисунок 2.27. Двухканальная СМО

Остальные характеристики вычисляются через параметры СМО следующим образом:

$$M = L + K \cdot \rho, \qquad (2.2.11)$$

$$\overline{T}_{ож} = \frac{L}{\lambda}, \qquad (2.2.12)$$

$$\overline{T}_{пр} = \overline{T}_{ож} + \overline{T}_{обс}. \qquad (2.2.13)$$

Многоканальную СМО можно поставить в соответствие, например, многопроцессорному блоку вычислительной системы, имеющему общую память для всех процессоров и, следовательно, общую очередь задач.

Формула Полячека–Хинчина.

Формула Полячека-Хинчина [4] для однолинейной СМО $M|G|1$ при прямой процедуре обслуживания (первым пришёл – первым обслужен) с пуассоновским потоком на входе и произвольным характером времени обслуживания в системе определяет среднее время ожидания обслуживания в виде:

$$\overline{T}_{ож} = \frac{\lambda b_2}{2(1-\lambda b_1)}, \qquad (2.2.14)$$

где λ – интенсивность входного простейшего потока заявок; b_1 – среднее время обслуживания; $b_2 = b_1^2 + D$ – второй момент распределения длительности обслуживания (D – дисперсия).

Заявка перемещается в очереди в среднем с постоянной скоростью. Среднее число переходов заявки в очереди на одно место вперед за единицу времени равно λ. При такой скорости переходов за время $\overline{T}_{ож}$ заявка

совершит L_C переходов. Это есть средняя длина очереди, т.е.

$$L_c = \overline{T}_{\text{ож}} \cdot \lambda. \qquad (2.2.15)$$

Подставляя в (2.2.15) вместо его определение (2.2.14), получаем выражение для средней очереди СМО $M|G|1$ в виде:

$$L_c = \frac{\rho^2}{2(1-\rho)}\left[1+\frac{D}{b_1^{\,2}}\right]. \qquad (2.2.16)$$

Здесь $\rho = \lambda b_1$ – коэффициент загрузки СМО.

Из (2.2.16), в частности, следует, что для модели $M|M|1$ (экспоненциальное время обслуживания), когда $D = b_1^{\,2}$, для средней длины очереди справедливо соотношение:

$$L_{\text{сэ}} = \frac{\rho^2}{(1-\rho)}.$$

Формула Полячика–Хинчина широко используется для расчета характеристик отдельных СМО, но применение этой формулы для анализа сетевых систем затруднено.

Системы M|M|1 обладают тем свойством, что выходящий поток обслуженных требований в стационарном режиме является пуассоновским. Сохранение пуассоновости в выходящем из отдельных СМО потоке облегчает построение и расчёт характеристик аналитических моделей сетей, представленных в виде стохастических очередей (СеМО), поскольку при этом условии сохраняется независимость длительностей пребывания требований в различных узлах моделей сетевых систем.

Системы M|G|1 не гарантируют сохранение пуассоновости в выходящем потоке. Отсутствие независимости длительностей пребывания требований в различных узлах моделей сетевых систем с нестандартными дисциплинами приводит к значительным трудностям при анализе моделей сетевых систем. Так, например, при достаточно реалистическом предположении о том, что длина требования остается постоянной в процессе его передачи через узлы сети (что характерно для сетей с пакетной коммутацией), необходимо прослеживать путь каждого требования, что делает невозможным аналитический расчет характеристики для сети с числом узлов $N > 2$. Аналитический анализ допускают экспоненциальные СеМО.

2.2.3. Экспоненциальные сети массового обслуживания

Сеть массового обслуживания представляет собой совокупность конечного числа N обслуживающих узлов, в которой циркулируют заяв-

ки, переходящие в соответствии с маршрутной матрицей из одного узла в другой. Для наглядного представления СеМО используется граф, вершины которого (узлы) соответствуют отдельным СМО, а дуги отображают связи между СМО (узлами).

Переход заявок между узлами происходит мгновенно в соответствии с переходными вероятностями:

$$p_{ij},\ i,j=\overline{1,N},$$

где p_{ij} – вероятность того, что заявка после обслуживания в узле i перейдет в узел j. Естественно, если узлы непосредственно не связаны между собой, то $p_{ij}= 0$.

Если из i-го узла переход только в один какой-либо узел j, то $p_{ij} = 3$.

Таким образом, экспоненциальной будем называть СеМО, отвечающую требованиям:

– входные потоки в СеМО пуассоновские;

– во всех N СМО время обслуживания заявок имеет экспоненциальную функцию распределения вероятностей и заявки обслуживаются в порядке прихода;

– переход заявки с выхода i-й СМО на вход j-й является независимым случайным событием, имеющим вероятность

$$p_{ij},\ i,j=\overline{1,N},$$

где p_{io} – вероятность ухода заявки из СеМО.

Если заявки приходят в сеть и уходят из неё, то сеть называется *разомкнутой*. Если заявки не приходят в сеть и из неё не уходят, сеть называется *замкнутой*. Число заявок в замкнутой сети постоянное.

Свойства разомкнутой экспоненциальной СеМО.

В разомкнутой СеМО входным потоком заявок СМО будем называть поток заявок, приходящих на вход отдельной СМО из внешней среды сети, т.е. не с выхода какой-либо СМО. В общем случае число входных потоков СеМО равно числу СМО.

В экспоненциальной СеМО поток заявок на входе отдельной СМО складывается из входного потока (возможно, имеющего нулевую интенсивность) и из потоков, поступающих с выходов некоторых других СМО. Характеристики СМО отвечают формулам (2.2.2) - (2.2.13). Поэтому для их расчёта в заданной СеМО достаточно найти интенсивности $\lambda_1,..,\lambda_N$ входных потоков СМО. Нахождение интенсивностей $\lambda_1,\ldots,\lambda_N$ осуществляется на основе уравнений баланса сети с учётом простых свойств слияния и разветвления потоков. При слиянии N потоков заявок с интенсив-

ностями $\lambda_1,\ldots,\lambda_N$ образуется поток, имеющий интенсивность $\lambda = \lambda_1+,\ldots,+\lambda_N$. При ветвлении потока с интенсивностью λ на N направлений, вероятности перехода заявки в которые равны $p_1,\ldots, p_N$, образуется N потоков с интенсивностями $\lambda p_1,\ldots,\lambda p_N$ соответственно.

В стационарной СеМО среднее число заявок в любой её фиксированной части постоянное. Отсюда следует, что суммарная интенсивность входящих в эту часть потоков равна суммарной интенсивности выходящих. Запись данного закона в математической форме называется *уравнением баланса*. Выделяя различные части в СеМО и составляя для них уравнения баланса, можно получить систему уравнений, связывающую неизвестные интенсивности $\lambda_1,\ldots,\lambda_N$ с известными $I_1,..,I_N$. Обычно при этом в качестве отдельных частей СеМО выделяют все СМО. В этом случае для N неизвестных имеется N уравнений. Можно добавить к ним уравнение баланса для входных и выходных потоков всей СеМО. Тогда получится $N+1$ уравнение, и одно из них можно использовать в качестве проверочного.

Разомкнутая экспоненциальная СеМО задается следующими параметрами:

1) числом N СМО;

2) числом $K_1,\ldots, K_N$ каналов в СМО 1, … , N;

3) матрицей $P = \| p_{ij} \|$ вероятностей передач, $i = 1,\ldots,N$; $j = 0,\ldots, N$;

4) интенсивностями $I_1,\ldots, I_N$ входных потоков заявок;

5) средними временами обслуживания $\overline{T}_{\text{обс1}}, \ldots, \overline{T}_{\text{обс1}}$ заявок в СМО.

Например, СеМО (рис. 2.28) будет задана численно в следующем виде:

a) N=3;

b) K_1=1; K_2=1; K_3=2;

c) P=
$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0{,}1 & 0 & 0{,}5 & 0{,}4 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

d) $I_1 = 1$; $I_2 = 0$; $I_3 = 0$;

e) $\overline{T}_{\text{обс1}} = 0{,}07$; $\overline{T}_{\text{обс2}} = 0{,}06$; $\overline{T}_{\text{обс3}} = 0{,}33$.

Баланс интенсивностей в сети можно учесть, обозначая интенсивности на входах и выходах СМО и СеМО так, как показано на рис. 2.28. Применяя свойства слияния и ветвления потоков, запишем, что

$$\begin{aligned} &\lambda_3 = I_1 + \lambda_2 + \lambda_3 \\ &\mathrm{I}_1 = p_{10}\cdot\lambda_1 \\ &\lambda_2 = p_{12}\cdot\lambda_1 \\ &\lambda_3 = p_{13}\cdot\lambda_1 \end{aligned} \qquad (2.2.17)$$

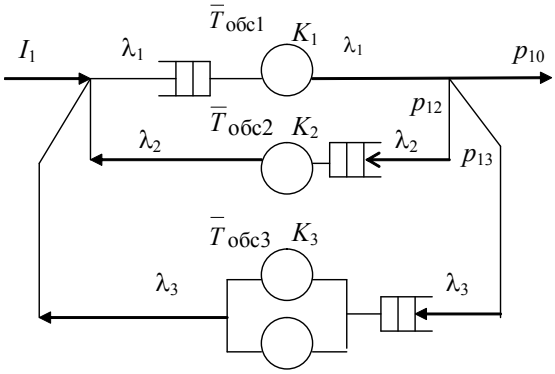


Рис. 2.28. Разомкнутая экспоненциальная СеМО

При известных $I_1 = 1$, $p_{10} = 0{,}1$; $p_{12} = 0{,}5$; $p_{13} = 0{,}4$ из последних трёх уравнений находим $\lambda_1 = 10$, $\lambda_2 = 5$, $\lambda_3 = 4$. Используя первое уравнение в (2.2.17) для проверки, подставляем в него найденные значения интенсивностей и получаем тождество $10 = I + 5 + 4$, подтверждающее правильность произведённых вычислений.

Далее выполняем проверку стационарности СеМО. СеМО стационарна, если стационарны все СМО, т.е. если

$$\rho_j \le 1,\ j = \overline{1, N}. \qquad (2.2.18)$$

Проверить эти условия после того, как определены, λ_j, не представляет труда. Условие (2.2.18) выполняется, поскольку

$$\rho_1 = \lambda_1 \cdot \overline{T}_{обс1} = 10 \cdot 0{,}07 = 0{,}7.$$
$$\rho_2 = \lambda_2 \cdot \overline{T}_{обс2} = 5 \cdot 0{,}06 = 0{,}3.$$
$$\rho_3 = \lambda_3 \cdot \overline{T}_{обс3} = 4 \cdot 0{,}35/2 = 0{,}7.$$

Для стационарной экспоненциальной СеМО с известными интенсивностями λ_j расчёт локальных характеристик сводится к применению формул (2.2.2) - (2.2.13).

Находим, что $c_1 = 0{,}7$; $L_1 = 1{,}63$; $M_1 = 2{,}33$; $\overline{T}_{ож1} = 0{,}163$; $\overline{T}_{пр1} = 0{,}233$; $c_2 = 0{,}3$; $L_2 = 0{,}13$; $M_2 = 0{,}43$; $\overline{T}_{ож2} = 0{,}026$; $\overline{T}_{пр2} = 0{,}086$; $c_3 = 0{,}7$; $\beta_0 = 0{,}176$; $L_3 = 0{,}402$; $M_3 = 1{,}802$; $\overline{T}_{ож3} = 0{,}1$; $\overline{T}_{пр3} = 0{,}43$.

С помощью такой СеМО можно промоделировать, например, вычислительную систему. Тогда входные потоки заявок СеМО будут изображать запросы, поступающие на вход вычислительной системы, отдельные СМО будут соответствовать этапам их обработки на устройствах (процессорах, периферийных устройствах и др.), выходные заявки СеМО - результатам обработки запросов.

Расчёт системных характеристик разомкнутых СеМО.

Характеристики СеМО определяются обычно на уровне средних значений и делятся на локальные и системные. К *локальным* характеристикам СеМО относятся характеристики всех входящих в нее СМО. *Системные* характеристики отражают свойства сети в целом, рассматриваемой как единая, неделимая на части система.

Наиболее важными системными характеристиками СеМО являются:

1. *Среднее время* $\overline{T}_{\text{пр}}$ *пребывания в сети.* Временем пребывания в сети называется время между приходом заявки в сеть и её уходом из сети.

2. *Передаточные коэффициенты* α_{ij}, $i, j = \overline{1, N}$. Пусть заявка входит в сеть из *i*-го входного потока. Её маршрут в сети случаен, поэтому случайно и число приходов в *j*-ю СМО за время пребывания в сети. Среднее значение α_{ij} этого числа приходов называют передаточным коэффициентом. Он однозначно определяется для любых *i*,*j*, матрицей *P* вероятностей передач.

3. *Входовые средние времена* $F_1,\ldots,F_N$ пребывания в сети. Величина F_j определяется как среднее время пребывания в сети заявки, поступающей из *j* - го входного потока ($j = \overline{1, N}$).

4. *Условные пропускные способности* $B_1,\ldots,B_N$. Предположим, что в заданной СеМО значение интенсивности I_j заменено на максимальное значение, при котором сеть ещё стационарна. Это значение B_j будем называть условной пропускной способностью по входу *j*.

При заданных I_k ($k \neq j$) сеть стационарна для любых значений $I_j \leq B_j$.

5. *Абсолютные пропускные способности* A_j. Предположим, что в заданной СеМО интенсивности всех входных потоков, кроме *j*-го, заменены на нулевые, а I_j заменена на предельное значение, при котором сеть ещё стационарна. Это значение A_j будем называть абсолютной пропускной способностью по *j*-му входу.

Если $I_j > A_j$, то сеть нестационарна, каковы бы ни были интенсивности остальных входных потоков.

6. *Запасы* $D_1,\ldots,D_N$ по пропускным способностям. Запас $D_j = B_j - J_j$, $j = \overline{1, N}$. Запас D_j показывает, насколько может быть увеличена интенсивность прихода заявок на *j*-м входе (при заданных остальных) без нарушения условия стационарности.

Если в виде СеМО моделируется некоторая реальная система, то названные характеристики могут дать ценную информацию о свойствах этой реальной системы. Например, если СеМО изображает вычислительную систему реального времени, то среднее время пребывания характеризует среднее время ответа системы, а запасы D_i выражают готовность системы продолжать устойчивое функционирование при увеличении нагрузки (интенсивности запросов) по тому или иному входу.

Среднее время пребывания заявки в СеМО рассчитывается по следующей формуле

$$\overline{T}_{\text{пр}} = \frac{1}{I}\sum_{j=1}^{N}\lambda_j \overline{T}_{\text{пр}j}, \qquad (2.2.19)$$

где $I = I_1 + ,...., + I_N$ - передаточные коэффициенты.

Важное и полезное свойство передаточных коэффициентов состоит в следующем. В стационарном режиме при любых $I_1 + ,...., + I_N$ для $\lambda_1,\ldots,\lambda_N$ справедливо

$$\begin{cases} \lambda_1 = \alpha_{11} I_1 + \alpha_{21} I_2 + ... + \alpha_{N1} I_N \\ \lambda_2 = \alpha_{12} I_1 + \alpha_{22} I_2 + ... + \alpha_{N2} I_N \\ \vdots \\ \lambda_N = \alpha_{1N} I_1 + \alpha_{2N} I_2 + ... + \alpha_{NN} I_N \end{cases} \qquad (2.2.20)$$

Обратим внимание на то, что строка передаточных коэффициентов в (2.2.20) представляет собой столбец матрицы $\| \alpha_{ij} \|$. Система (2.2.20) выражает интенсивности λj прихода заявок в СМО через интенсивности входных потоков сети.

Приведём алгоритм вычисления матрицы $\| \alpha_{ij} \|$.

1. Составить уравнения баланса сети, включающие интенсивности $I_1,\ldots I_N$ в буквенном виде,

2. Положить $k = 3$.

3. Решить уравнения баланса для случая, когда $I_k = 1$, остальные $I_i = 0$. Полученные значения $\lambda_1,\ldots,\lambda_N$ записать в k-ю строку матрицы передаточных коэффициентов.

4. Положить $k = k + 3$.

5. Если $k < N$, перейти к 3), иначе к 6).

6. Конец.

Входовое среднее время пребывания. Рассмотрим СеМО на рис. 2.28 и проследим, как формируется входовое время пребывания в сети заявки первого потока. Это время состоит из двух слагаемых. Первое слагаемое есть среднее время пребывания в СМО – $\overline{T}_{\text{пр1}}$. Второе слагаемое включает составляющие: с вероятностью p_{10} равной нулю (заявка уходит из сети), с вероятностью p_{12} равной вхо-довому времени пребывания для входа 2 (заявка входит в сеть через СМО 2) и с вероятностью p_{13} – входовому времени пребывания для входа 3. Итак, в среднем второе слагаемое составляет $p_{10} \cdot 0 + p_{12}F_2 + p_{13}F_3 = p_{12}F_2 + p_{13}F_3$ величину. В целом среднее входовое время пребывания F_1 равно сумме средних значений первого и второго слагаемых:

$$F_1 = T_{\text{пр}1} + p_{12} F_2 + p_{13} F_3 . \qquad (2.2.21)$$

Рассуждая аналогично о входовых средних временах пребывания F_2 и F_3 можно записать для них сходные с (2.2.21) уравнения, которые вместе с (2.2.21) составят следующую систему уравнений:

$$\begin{aligned} F_1 &= \overline{T}_{\text{пр}1} + p_{12} F_2 + p_{13} F_3 \\ F_2 &= \overline{T}_{\text{пр}2} + F_1 \\ F_3 &= \overline{T}_{\text{пр}3} + F_1 . \end{aligned} \qquad (2.2.22)$$

Развернутая форма условия стационарности. Условие стационарности СеМО запишем в виде:

$$\frac{\lambda_j T_j}{K_j} \le 1, \quad j = \overline{1, N}.$$

Эта запись эквивалентна следующей: $\lambda_j \le K_j / T_j, \quad j = \overline{1, N}.$

Выражая λ_j через I_j по формуле (2.2.20), получим развернутую форму условия стационарности:

$$\begin{cases} \alpha_{11} I_1 + \alpha_{21} I_2 + \ldots + \alpha_{N1} I_N \le K_1 / T_1 \\ \alpha_{12} I_1 + \alpha_{22} I_2 + \ldots + \alpha_{N2} I_N \le K_2 / T_2 \\ \qquad\qquad\qquad\qquad \vdots \\ \alpha_{1N} I_1 + \alpha_{2N} I_2 + \ldots + \alpha_{NN} I_N \le K_N / T_N \end{cases} \qquad (2.2.23)$$

Эта система неравенств эквивалентна (2.2.20).

Абсолютная пропускная способность. Используя развернутую форму условий стационарности, абсолютную пропускную способность A_i по i-му входу можно найти непосредственно по её определению. Действительно, если все входные интенсивности СеМО, кроме I_i, положить равными нулю, то из (2.2.23) получим, что для стационарности необходимо условие:

$$\begin{cases} \alpha_{i1} I_i \le K_1 / \overline{T}_{\text{обс}1} \\ \alpha_{i2} I_i \le K_2 / \overline{T}_{\text{обс}2} \\ \ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots \\ \alpha_{iN} I_i \le K_N / \overline{T}_{\text{обс}N} \end{cases}$$

Это условие удобно переписать так:

$$\begin{cases} I_i \le K_1/(\overline{T}_{\text{обс1}}\alpha_{i1}) \\ I_i \le K_2/(\overline{T}_{\text{обс2}}\alpha_{i2}) \\ \ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots \\ I_i \le K_N/(\overline{T}_{\text{обс}N}\alpha_{iN}) \end{cases}. \qquad (2.2.24)$$

Из определения A_i вытекает, что эта величина равна максимальному из значений I_i, отвечающих (2.2.24). Следовательно, A_i равно наименьшей из правых частей в (2.2.24).

Условная пропускная способность. Условная пропускная способность, как и абсолютная, может быть найдена из (2.2.23). Для нахождения B_i в (2.2.23) следует подставить за-данные значения всех входных интенсивностей СеМО, кроме I_j. Затем полученная система разрешается относительно I_i в виде:

$$\begin{cases} I_1 \le B_1 \\ I_2 \le B_2 \\ \quad\vdots \\ I_N \le B_N \end{cases}. \qquad (2.2.25)$$

и B_i находится как наименьшая из правых частой в (2.2.25). Если условие стационарности СеМО содержит в себе лишь одно неравенство, как на рис. 2.28, то нахождение B_i упрощается.

Запасы по пропускным способностям. Формула для вычисления запасов D_i дана непосредственно в их определении. Рассмотрим пример анализа СеМО. Определим системные характеристики для СеМО на рис. 2.28.

Среднее время пребывания заявки в СеМО:

$$\overline{T}_{\text{пр}} = \frac{\sum_{j=1}^{3}\lambda_j \overline{T}_{\text{пр}j}}{I_1 + I_2 + I_3} = \frac{10\cdot 0{,}233 + 5\cdot 0{,}086 + 4\cdot 0{,}45}{1+0+0} = 4{,}56 \text{ с.}$$

Передаточные коэффициенты. Значения коэффициентов α_{ij} однозначно определяются матрицей P вероятностей передач. Из (2.2.20) вытекает, что при $I_2 = \ldots I_N = 0$, $I_1 = 1$ имеет место

$$\begin{cases} \lambda_1 = \alpha_{11} \\ \lambda_2 = \alpha_{12} \\ \vdots \\ \lambda_N = \alpha_{1N} \end{cases}$$

Это позволяет найти строку коэффициентов α_{1j}- матрицы $\| \alpha_{ij} \|$ путём решения уравнений баланса сети для случая найденные значения $\lambda_1,\ldots,$ λ_N будут численно равны коэффициентам $\alpha_{11},\ldots, \alpha_{1N}$. Аналогично для случая, когда $I_k = 1$, остальные $I_i = 0$, решение уравнений баланса даст значения $\alpha_{k1},\ldots, \alpha_{kN}$.

Найдем матрицу $\| \alpha_{ij} \|$ для СеМО на рис. 2.28, составим уравнения баланса:

$$\begin{cases} \lambda_1 = I_1 + \lambda_2 + \lambda_3 \\ I_1 + I_2 + I_3 = p_{10} \\ \lambda_2 = p_{12}\lambda_1 + I_2 \\ \lambda_3 = p_{13}\lambda_1 + I_3 \end{cases}$$

Решим эти уравнения для $I_1=1$, $I_2=I_3=0$. Получим $\lambda_1=10$, $\lambda л_2=5$, $\lambda_3=4$. Для $I_2=1$, $I_1=I_3=0$ решением будет $\lambda_1=10$, $\lambda_2=6$, $\lambda_3=4$ и для $I_3=1$, $I_1=I_2=0$ решением будет $\lambda_1=10$, $\lambda_2=5$, $\lambda_3=3$.

Следовательно, матрица $\| \alpha_{ij} \|$ этой СеМО имеет вид:

10 5 4
10 6 4
10 5 5

Входовое среднее время пребывания. Из системы (2.2.22) при известных $\bar{T}_{\text{пр}j}$ (найденных при расчёте схемы на рис. 2.28 нетрудно найти $F_1 = 4{,}56$, $F_2 = 4{,}64$, $F_3 = 5{,}01$.

Развернутая форма условия стационарности. Для конкретных СеМО некоторые из неравенств системы (2.2.23) оказываются излишними: такие неравенства можно исключать из сис-темы, не изменяя решения. Для рассматриваемой СеМО условие (2.2.23) примет вид:

$$\begin{aligned} &10\,I_1 + 10\,I_2 + 10\,I_3 \le 1/0{,}07 \\ &5\,I_1 + 6\,I_2 + 5\,I_3 \le 1/0{,}06 \\ &4\,I_1 + 4\,I_2 + 5\,I_3 \le 2/0{,}35 \end{aligned}$$

или, после сокращения на положительные коэффициенты,

$$\begin{aligned} &I_1 + I_2 + I_3 \le 10/7 \\ &I_1 + 1{,}2\,I_2 + I_3 \le 10/3\,. \\ &I_1 + I_2 + 1{,}25\,I_3 \le 10/7 \end{aligned} \qquad (2.2.26)$$

В этой системе второе неравенство вытекает из первого (сравните их, предварительно умножив первое на 1,2). Поэтому второе неравенство может быть отброшено. Кроме того, первое неравенство вытекает из третьего, поэтому его тоже можно отбросить. Следовательно, условие стационарности (2.2.26) эквивалентно следующему:

$$I_1 + I_2 + 1{,}25\, I_3 \le 10/7. \qquad (2.2.27)$$

Абсолютная пропускная способность. Для СеМО на рис. 2.28 нахождение A_i несколько упрощается благо-даря тому, что условие стационарности сети (2.2.27) содержит лишь одно неравенство. Так, полагая $I_2 = I_3 = 0$ для I_1 из (2.2.27) получим $I_1 \le 10 / 7$, от-куда $A_1 = 10/ 7$. Аналогично вычисляются $A_2 = 10/7$ и $A_3 = 8/7$. Вполне ес-тественно, что найденные значения совпадают с максимальными значе-ниями для I_i, показанными в правых частях уравнений (2.2.26).

Условная пропускная способность. Условную вероятность для рассматриваемой СеМО найдём из (2.2.27):

$B_1 = 10/7, B_2 = 3/7, B_3 = 12/33$.

Запасы по пропускным способностям. Запасы составляют D_1=10/ 7-1= 3/7, D_2= 3/7- 0 =3/7, D_3= 12/35- 0 =12/33.

Схема расчёта замкнутой СеМО.

Итерационный метод анализа [1] средних значений характеристик замкнутых СеМО (ЗСеМО) позволяет определять такие практически важные показатели функционирования, как средние длины очередей и времена ожиданий (пребываний в СеМО), производительности сети и загрузки её узлов и т.д.

В ЗСеМО циркулирует конечное число заданий. Узел *i* такой ЗСеМО представляет собой обслуживающий прибор (возможно, многоканальный) с экспоненциальным распределением времени обслуживания и очередь заявок, ожидающих обслуживания. Дисциплина обслуживания "первым пришел - первым обслужен" такова, что обслуживающий прибор не простаивает при наличии хотя бы одной заявки в очереди. После обработки заявки в узле *i* она с вероятностью P_{ij}

$$(\sum_{n=1}^{N} P_{ij} = 1, \text{ где } N \text{ - общее число узлов в ЗСеМО})$$

переходит на обслуживание в другой узел *j*. Общее число заявок по всей сети постоянно и равно *J*.

Решающим в анализе средних значений является вычисление среднего времени задержки в узле *i*, i=1,…, N, ЗСеМО. Рассмотрим момент, когда заявка поступает в эту систему Средняя задержка, которую испы-

тывает заявка, состоит из времени t_k её обслуживания и времени обслуживания заявок, ожидавших перед ним в очереди, включая заявку, находившуюся на обслуживании.

Среднее время задержки (пребывания) заявки в *k*-м узле при наличии в сети *j* заявок связано со средним числом ожидающих при наличии в сети (*j* - 1) заявок соотношением [1]:

$$T_{\text{пр}} = \tau_k \times \left[1 + \left\{ \frac{n_k(j-1)}{\alpha_k} \right\} \right],$$

где $n_k(j)$, по определению, есть среднее число заявок в узле *k* при наличии в сети *j* заявок; α_k - число обслуживающих приборов в *k*-м узле.

Данное равенство имеет в точности желаемую рекуррентную форму. Применительно ко всей ЗСеМО эта форма выглядит следующим образом. Обозначим:

- сеть содержит *N* узлов;

- $T_{\text{пр}}(v) = \left[T_{\text{пр}_k}(v) \right],\ k = \overline{1,N}$ – вектор средних времен пребывания заявок в СМО сети при наличии в сети *v* заявок;

- $\overline{T}_{\text{обс}_k}$ – среднее время обслуживания заявок в *k*-й СМО, ;
- $n_k(\nu)$ – среднее число заявок в *k*-й СМО при наличии в сети *v* заявок;
- α_k – число обслуживающих приборов в *k*-й СМО;
- $\overline{T}_{\text{пр}}(v)$ – среднее время пребывания заявок в СеМО при наличии в сети *v* заявок.

Для $v = \overline{1,\gamma}$ и $k = \overline{1,N}$ рассчитывается среднее время пребывания в *k*-й СМО при наличии в сети *v* заявок.

$$\overline{T}_{\text{пр}\ k}(v) = \overline{T}_{\text{обс}\ k} \left(1 + \frac{n_k(v-1)}{\alpha_k} \right). \qquad (2.2.28)$$

Среднее время пребывания заявок в замкнутой СеМО при наличии в сети *v* заявок

$$\overline{T}_{\text{пр}}(v) = e T_{\text{пр}}(v); \qquad (2.2.29)$$

$$\lambda(v) = \frac{v}{\overline{T}_{\text{пр}}(v)}; \qquad (2.2.30)$$

$$n_k(v) = \lambda(v) e_k T_{\text{пр}\ k}(v). \qquad (2.2.31)$$

Величина $\lambda(\nu)$ – пропускная способность ЗСМО при наличии в ней ν заявок.

Вектор $e = [e_k], k = \overline{1, N}$ является решением системы линейных уравнений (уравнений баланса для замкнутой сети):

$$e = eP, \qquad (2.2.32)$$

которая определяет стационарное распределение цепи Маркова, управляющей переходами заявок в ЗСеМО, с матрицей вероятностей переходов:

$$P = \lfloor P_{ij} \rfloor, \; i = \overline{1, N}, \; j = \overline{1, N},$$

где $\sum_{j=1}^{N} P_{ij} = 3$. Система (2.2.32) решается при дополнительном ограничении

$$\sum_{i=1}^{N} e_i = 1.$$

Выполнение процедуры (2.2.28) - (2.2.31) начинается с $\nu = 1$, $n_k = 0$ для $\forall k = \overline{1, N}$.

Вычисления (увеличение ν на 1) ведутся до тех пор, пока ЗСеМО не войдет в состояние насыщения.

Критерий (признак) насыщения:

$$[\lambda(\nu - 1) / \lambda(\nu)] \geq \varepsilon, \qquad (2.2.33)$$

где $0{,}9 < \varepsilon < 1$ – численное значение критерия насыщения.

Значение , удовлетворяющее (2.2.33), принимается за пропускную способность (производительность) замкнутой СеМО.

Характер зависимости $\lambda(\nu - 1) / \lambda(\nu)$ приведён на рис.2.29:

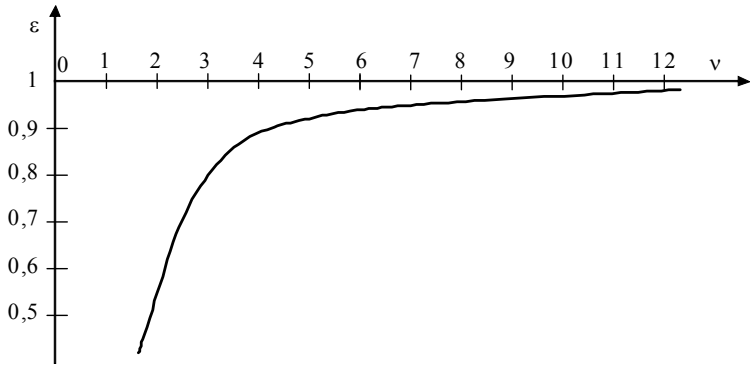


Рис. 2.29. Характеристика процесса насыщения

Рекуррентная численная процедура (2.2.28) - (2.2.31) расчёта замкнутых сетей массового обслуживания широко используется для расчёта производительности многомодульных систем, например, коммутаторов в корпоративных сетях.

Вопросы и задания

1. Какие объекты называют СМО? Перечислите характеристики СМО.

2. Какой характер имеет зависимость характеристик M, λ, $\overline{T}_{\text{пр}}$ от ρ в одноканальной экспоненциальной СМО?

3. Подстановка $K = 1$ в (2.2.8) – (2.2.13) должна дать формулы для расчёта характеристик одноканальной СМО. Проверьте, так ли это.

4. Что такое экспоненциальная СеМО?

5. Что такое уравнения баланса и для чего они применяются?

6. В чём состоят особенности моделирования многоканальных и многофазовых СМО?

7. По каким признакам различают СМО? Какова структура описания СМО?

8. Возможно ли улучшение характеристик СМО и какое?

9. Что составляет предмет исследования СМО?

10. Перечислите системные характеристики СеМО.

11. Объясните, в чем физический смысл характеристики «запасы по пропускным способностям».

12. Объясните, в чём физический смысл характеристики “условная пропускная способность”.

13. Предположим, что все входные потоки некоторой СеМО, кроме 2-го и 3-го, имеют нулевые интенсивности $I_i = 0$ и требуется найти характеристики F_i, B_i, A_i, D_i, для $i = 2, 3$.Необходимо ли для этого вычислить всю матрицу $||\alpha_{ij}||$ передаточных коэффициентов или достаточно иметь её 2-ю и 3-ю строки? Если достаточно, то как по ним найти требуемые характеристики?

3. АЛГОРИТМЫ МОДЕЛИРОВАНИЯ СИСТЕМ И СЕТЕЙ ТЕЛЕКОММУНИКАЦИЙ

3.1. Имитационное моделирование систем

Имитационное моделирование (ИМ) – это метод исследования, который основан на том, что анализируемая динамическая система заменяется имитатором и с ним производятся эксперименты для получения об изучаемой системе новых сведений. Роль имитатора зачастую выполняет программа ЭВМ. Имитационные модели связаны не с аналитическим представлением, а с принципом динамической имитации с помощью информационных и программных средств сложных процессов и систем.

Модель системы при ИМ представляет собой совокупность моделей элементов и их функциональные взаимосвязи. Модель элемента (агрегата, обслуживающего прибора) - это, в первую очередь, набор правил (алгоритмов) поведения устройства по отношению к входным воздействиям и правил изменений состояний элемента. Элемент отображает функциональное устройство на том или ином уровне детализации. В простейшем случае устройство может находиться в работоспособном состоянии или в состоянии отказа. В работоспособном состоянии устройство может быть занято, например, выполнением определенной операции или быть свободным.

При имитационном моделировании стохастических систем определяющими являются два аспекта: построение моделирующего алгоритма и разыгрывание случайных факторов, обуславливающих возможные состояния системы.

Моделирующий алгоритм обеспечивает воспроизведение во времени процесса смены состояний системы, т.е. строит траектории процесса функционирования системы. Функционирование системы представляется как хронологическая последовательность событий. Продвижение системного времени реализуется посредством программирования симулятора – «движителя», который воспроизводит во времени движение (смену состояний) объекта моделирования. Для разыгрывания случайных факторов применяется метод статистических испытаний (метод Монте-Карло), особенности которого рассматриваются в следующем разделе.

3.1.1. Основные понятия имитационного моделирования

Устройство (средство) – элемент имитационной модели, который позволяет провести имитацию процесса обслуживания. Простые (одноканальные) имитационные модели обслуживают одновременно одну заявку; сложные (многоканальные) позволяют одновременно обслуживать несколько заявок.

Заявка инициирует начало какого-либо процесса в системе. Заявка характеризуется внутренней структурой: одиночная или групповая (группа однотипных заявок). Генератор заявок реализует законы их поступления в систему. В соответствии с законом поступления заявки бывают: детерминированные (время поступления заявки в систему четко определено) и вероятностные (используется нормальное, равномерное, экспоненциальное и др. виды распределений).

Очередь – элемент модели, который включает заявки, которые по каким либо причинам не могут быть обслужены. Очереди ставятся перед каждым устройством, на входе системы, на выходе, либо в точках, которые являются потенциальными «узкими» местами в системе, либо в этой точке необходимо провести дополнительное накопление результата.

Процесс – последовательность событий и работ, описывающая поведение во времени какого-либо объекта в моделируемой системе.

Простые процессы характеризуются последовательным характером выполнения, минимальным количеством типов заявок и условий инициации процесса и обслуживания заявок, наличием простых устройств в обслуживании; сложные процессы описываются большим количеством типов заявок, имеют сложные условия развития и инициации, используют сложные, многофазные устройства.

Для описания процесса необходимо знать:

1. Заявки, которые с ним связаны;

2. Характер их поступления в систему (условия инициации самого процесса);

3. Устройства, которые связаны с обслуживанием в рамках данного процесса;

4. План-график выполнения работ или задач в рамках данного процесса;

5. Условия связи с другими процессами;

6. Критерий оценки эффективности.

События – связаны с изменением состояния системы и её объектов. События обеспечивают прерывистость процесса. Процесс представляется из набора активностей и пассивностей. Начало каждой активности связано с возникновением события в системе

Работа – действие, происходящее в моделируемой системе в течение определенного промежутка времени. Работа начинается и заканчивается событиями.

Системное время. Механизмы учёта системного времени:

1.Время изменяется равномерно с определенной дискретностью;

2.Скачкообразное изменение времени в соответствии с возникновением событий. Список будущих событий – каждое событие имеет характеристику времени возникновения.

Управляющая программа (монитор) просматривает список будущих событий и извлекает событие, которое находится в вершине, производит:

– запуск на выполнения данного события;

– изменение значения счетчика времени.

Случайные факторы в моделировании. Источники появления случайных факторов могут быть внешними и внутренними. Для моделирования случайных факторов необходимо знать закон, по которому изменяются случайные факторы. Данный закон обычно задается при помощи соответствующих теоретических либо эмпирических функций распределения. При этом необходимо использовать генераторы псевдослучайных чисел для имитации случайности тех или иных событий.

Задачи – представляют собой любую активность – элемент процесса.

3.1.2. Построение моделирующего алгоритма

При разработке имитационной модели и планирования проведения модельных экспериментов различают модельное (системное) время и машинное время. Машинное время отражает затраты времени ЭВМ на проведение имитации. Системное время соответствует реальному времени. С помощью механизма системного времени отображается переход моделируемой системы из одного состояния в другое, производится синхронизация работы компонентов модели и квазипараллельная реализация событий, которые в реальной системе возникают (протекают) одновременно. Опишем, как моделируется система в общем случае.

В памяти ЭВМ отводится несколько ячеек для переменных, характеризующих состояние системы в целом и отдельных её элементов. Содержимое этих ячеек изменяется в соответствии с алгоритмом моделирования так, как это происходит в реальной системе при её функционировании. Отдельная ячейка содержит текущее *системное время*, указывающее, к какому моменту времени относится состояние системы, записанное в памяти ЭВМ. Содержимое указанных ячеек памяти меняется путем циклического повторения *основной* части алгоритма моделирования, называемой *шагом* (*циклом*) *имитации*. За один шаг осуществляется переход к следующему значению системного времени, т.е**.** *продвижение по времени*. Попутно изменяется значение переменных, характеризующих состояние системы. Таким образом, шаг за шагом имитируется *смена состояний системы*, т.е. моделируется процесс функционирования системы.

Рассмотрим *принципы продвижения по времени*.

Принцип Δt. Первая мысль – увеличивать системное время за один шаг на постоянную величину Δt. Это – так называемый *принцип* Δt. При

использовании этого принципа возникает проблема выбора длины интервала Δt. Как правило, алгоритм шага рассчитан на имитацию одного события: поступления заявки, завершения фазы обслуживания и т.п. *Событие* – это любое изменение в системе. Все изменения, связанные друг с другом причинно-следственными связями и происходящие в один и тот же момент времени, обычно рассматриваются как одно событие. Допустим, поступила заявка. При этом увеличилось число заявок в системе. Если эта заявка сразу поступила на обслуживание, то изменилось состояние прибора и количество занятых приборов. Все это – одно событие, поступление заявки.

Возможно, что за время Δt в системе (например, СМО) произойдет несколько событий (в разные моменты). В таком случае алгоритм, рассчитанный на имитацию одного события за один шаг, неправильно отразит изменения, произошедшие в системе. Чтобы избежать это есть два пути. Первый путь – разработка алгоритма шага, рассчитанного на имитацию нескольких событий. Этот путь приводит к значительному усложнению алгоритма. Другой путь – использование столь малого интервала Δt, что за это время практически не происходит более одного события. Этот путь приводит к резкому увеличению затрат машинного времени, так как с уменьшением Δt соответственно возрастает число шагов, которое надо выполнить, чтобы имитировать процесс на заданном отрезке времени. При малом Δt большинство шагов окажутся пустыми, так как события будут происходить лишь на некоторых интервалах Δt, а на прочих интервалах состояния СМО будет сохраняться таким же, как на соседних интервалах.

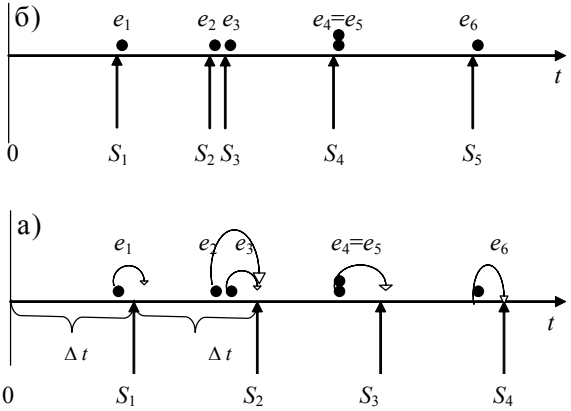
Принцип особых моментов. Последнее замечание наводит на мысль, что целесообразно проскакивать за один шаг весь промежуток времени между соседними событиями и тем самым избегать пустых шагов. Это – *принцип особых моментов*. Особым моментом принято называть такой момент времени, когда в системе происходит какое-либо изменение, иначе говоря, происходит событие. За один шаг осуществляется продвижение по времени на случайный отрезок: от одного особого момента до другого. Рис. 3.1 демонстрирует способы представления и управления временем в обоих случаях.

Примем следующие соглашения:

– пусть время – частично упорядоченное множество $T=\{t_1, t_2,\ldots, t_n\}$;

– пусть существует множество событий $e_i \in E$, $i = 1, 2 \ldots$;

– все происходящие в модели события фиксируются в календаре событий, (обозначим его $S_{\text{событий}}$). Календарь – это список элементов S_i, где каждый элемент S_i представляет собой пару (e_i, t_i). По оси времени отложена одна и та же последовательность событий e_i. Как видно, два



$S_{событий}$		
$S1$	e_1	t_1
$S2$	e_2	t_2
$S3$	e_3	t_3
$S4$	e_4, e_5	t_4
$S5$	e_6	t_5

$S_{событий}$		
$S1$	Δt	e_1
$S2$	$2{\cdot}\Delta t$	e_2, e_3
$S3$	$3{\cdot}\Delta t$	e_4, e_5
$S4$	$4{\cdot}\Delta t$	e_6

Рис. 3.1. Течение модельного времени с постоянным шагом (а) и по особымсостояниям (б)

события e_4 и e_5 появляются одновременно. Стрелки указывают на точки, в которых происходит приращение времени на один такт, и моменты наступления очередных событий в обеих моделях. В модели, использующей принцип особого состояния, имитируемое время при изменении сдвигается вперед точно на момент наступления самого раннего из последующих событий, а в модели, использующей принцип постоянного шага, имитируемое время сдвинется ровно на фиксирован-ное значение Δt. Управляющая программа (симулятор) в модели на рис. 3.1,а продвигает время с постоянным шагом Δt и фиксирует в календаре событий состояние, в котором находится исследуемый объект. Симулятор на рис. 3.1,б продвигает время до ближайшего следующего события и в календаре фиксирует само событие и время, когда оно произошло – это и будет новое состояние, в котором находится исследуемый объект до тех пор, пока не произойдет новое событие.

Чтобы ЭВМ могла вычислить очередной особый момент, используется так называемый *календарь*. Календарь, или расписание предстоящих событий – это группа ячеек памяти, где для каждого типа события указан ближайший момент, когда такое событие произойдет. Имея календарь, нетрудно определить очередной особый момент. Это наименьший из моментов, записанных в календаре.

Чтобы заполнить календарь и в дальнейшем корректировать его содержимое, осуществляется *планирование событий*. Допустим, в текущий момент поступила заявка и сразу была взята на обслуживание первым прибором, так как он оказался свободным. Закон распределения времени обслуживания задан наряду с другими данными. Обратившись к специальной подпрограмме - *датчику случайных чисел* ЭВМ генери-

рует случайное время обслуживания в соответствии с указанным законом распределения. Прибавив это время к текущему моменту, ЭВМ вычисляет момент, когда освободится первый прибор, и заносит это число в ячейку календаря, отведенную для первого прибора. Аналогично заполняются другие ячейки календаря.

При моделировании СМО целью является определение характеристик качества функционирования СМО, например, вероятности потери заявки, или других величин: коэффициента загрузки прибора, средней длины очереди и т.п. Эти характеристики и вычисляются по окончании имитации на основе *статистик,* накопленных в процессе имитации. Примеры статистик: $K_З$ – количество поступивших заявок, $K_П$ – количество потерянных заявок, $T_{СЗ}$ – суммарное время занятости прибора. На основании этих статистик можно вычислить оценки искомых характеристик: вероятности потери заявки $P_{ПОТ}= K_П/K_З$ и коэффициента загрузки одного прибора $P_{ЗП}= T_{СЗ}/T$, где T – длина реализации, т.е. длительность имитированного процесса. Операторы, осуществляющие *пополнение статистик*, входят в состав алгоритма шага.

Основная часть алгоритма имитации представляет собой циклическое повторение шагов имитации до тех пор, пока не будет выполнено *условие остановки*. Остановка производится после выполнения заданного числа шагов или достижения заданного значения системного времени (длины реализации). Чем длиннее реализация, тем точнее оценки искомых характеристик.

Рассмотрим организацию цикла на примере моделирования СМО с отказами (рис. 3.2). Предполагается получить оценку вероятности потери заявки (вероятности отказа) путем воспроизведения на ЭВМ достаточно длинного отрезка реализации.

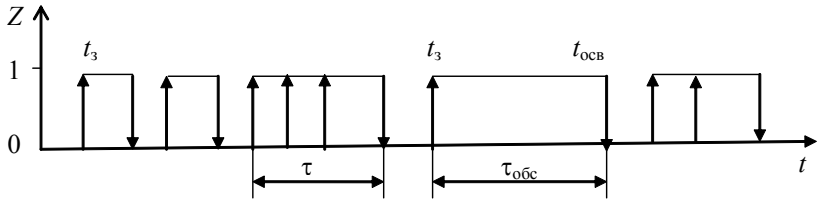


Рис. 3.2. Поток заявок, состояние СМО

Два способа оценивания вероятности потери заявки: по заявкам $P^З_{ОТК}= K_{ОТ}/K_С$; по времени $P^Т_{ОТК}= T_1/T_М$, где $K_{ОТ}$ – количество заявок, получивших отказ; $K_С$ – общее количество заявок, поступивших в систему; T_1 – суммарное время, когда прибор находится в занятом состоянии; $T_М$ – общее время моделирования.

Z – состояние системы: $Z = 0$ –свободен, $Z = 1$– занят; τ – интервал между заявками; $\tau_{ОБС}$ – время обслуживания заявки; $t_з$ – момент поступления заявки; $t_{ОСВ}$ – момент освобождения прибора (канала) обслуживания; $t_Т$ – текущий момент.

Цикл, организованный по принципу Δt (рис.3.3). Перед началом цикла в памяти хранится: $t_з$, если $Z = 0$; $t_{осв}$, если Z=1. Время измеряется числом тактов.

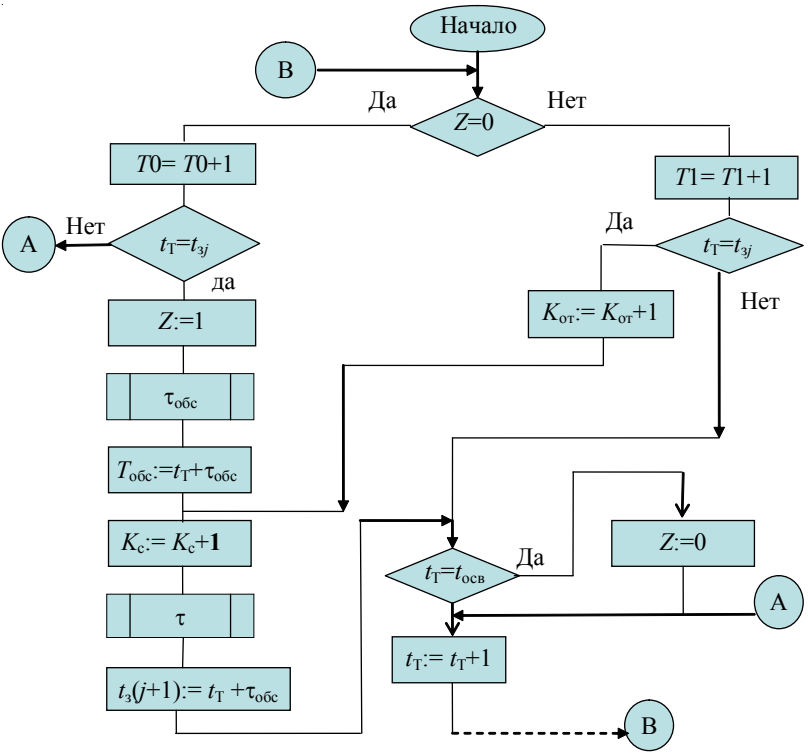


Рис. 3.3. Цикл, организованный по принципу Δt

Цикл, организованный по принципу особых моментов (рис. 3.4). Особый момент: поступает заявка, закончено обслуживание заявки (освобождение прибора). К началу цикла в памяти хранится: $t_П$ – последний особый момент; $t_з$ – момент поступления очередной заявки; $t_{ОСВ}$ – момент освобождения прибора, следующий за $t_П$, если в момент $t_П$ система занята. Начинается цикл с того, что последний особый момент рассматривается как предыдущий и определяется очередной особый момент. В каждом конкретном случае принцип продвижения времени выбирается

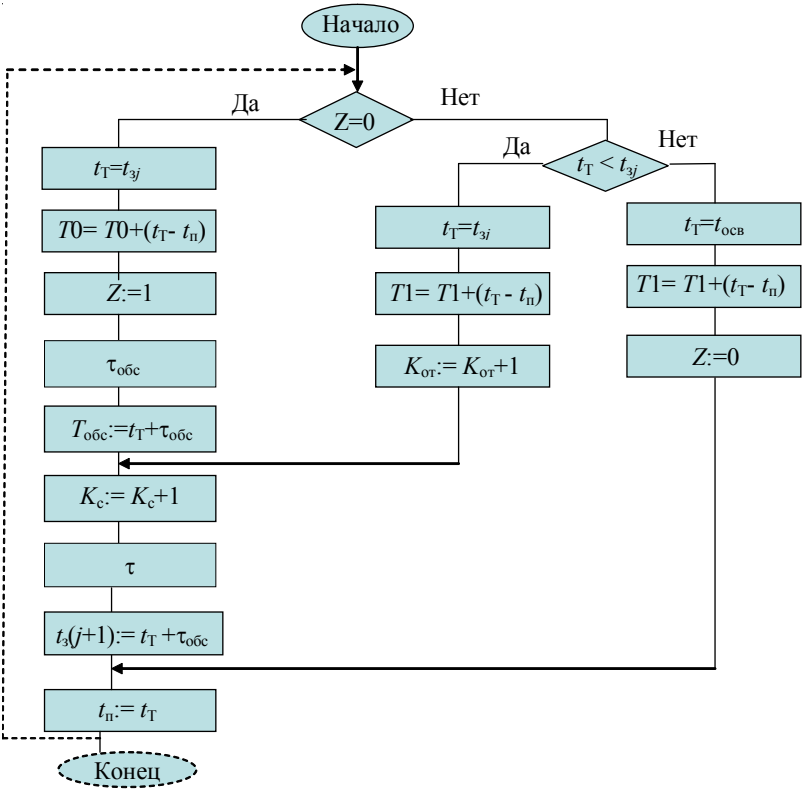


Рис. 3.4. Цикл, организованный по принципу особых моментов

разработчиком в зависимости от характера системы, которую необходимо моделировать. Принцип Δt предпочтительнее, если события появляются более менее регулярно и распределены во времени относительно равномерно. Принцип особых моментов позволяет экономить машинное время, когда существенные события могут длительное время не наступать; не требует определения величины приращения времени; эффективен при неравномерном распределении событий во времени.

Парадоксы времени.

Алгоритм управления временем должен следить за тем, чтобы события выполнялись в хронологическом порядке. Эта задача не является тривиальной. Действительно, *логический* процесс заранее не может знать о том, на какое время будет запланировано событие, которое он получает от другого логического процесса.

Рассмотрим пример. Пусть модель представляет собой совокупность трёх процессов. Один процесс отображает поведение покупателя, второй – магазина, в котором покупатель совершает покупки, а третий процесс – деятельность банка, со счёта которого покупатель снимает деньги. Предположим, что покупатель приобрёл товары в магазине на определённую сумму N (в кредит) (событие e_1, произошло в момент времени $t_1 = 9$). Магазин уведомил об этом банк (e_2, $t_2 = 10$) Сумма на счёте уменьшается: $S = S - N$. Покупатель посещает банк с целью снять деньги со счёта (e_3, t_3=11). Если денег на счёте достаточно, то банк выдаёт клиенту (которым является покупатель) запрошенную им сумму. Если счёт меньше запрошенной суммы, то покупателю будет отказано. Хронологический порядок событий: e_1, e_2, e_3 представлен на рис. 3.5.

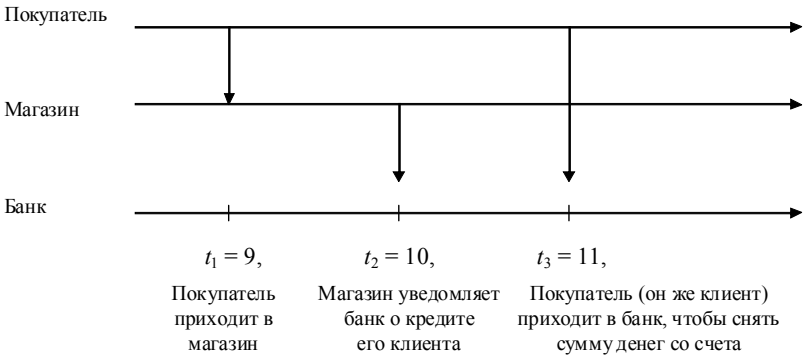


Рис. 3.5. Банк получает сообщения в хронологическом порядке

Рассмотрим ситуацию, когда уведомление в банк из магазина поступит позже того, как покупатель снимет сумму с вклада в банке (которой уже нет на счёте), и банк понесёт убытки. Ситуация, которая обрисована выше, возникла вследствие того, что хронология событий была нарушена (рис. 3.6).

Нарушение хронологического порядка могло произойти по той причине, что в распределённом моделировании время для разных логических процессов движется с разной скоростью. Например, если процесс, реализующий работу магазина, выполняется на загруженном процессоре, то уведомление банку поступает позже, поскольку процесс банка «убежал» вперёд (он выполняется на менее загруженном процессоре).

Имитационный алгоритм должен уметь бороться с такими парадоксами времени. В этом заключается проблема синхронизации компонентов имитационного моделирования. Было предпринято множество попыток решить эту проблему. В настоящее время все алгоритмы синхрониза-

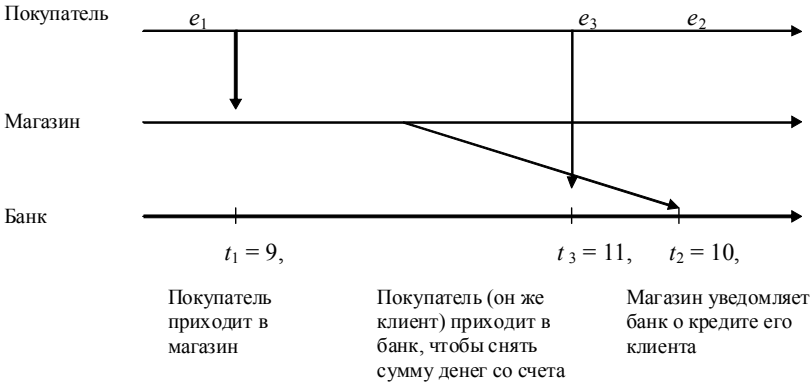


Рис. 3.6. Хронологический порядок событий нарушен

ции делятся на консервативные и оптимистические. Консервативный алгоритм не позволит логическому процессу обрабатывать событие с определённой временной отметкой, пока не убедится, что другой логический процесс не запланировал событие с меньшей временной меткой. Оптимистический алгоритм позволяет выбирать из списка необработанных событий очередное событие и обрабатывать его, исключив проверку событий, планируемых другими логическими процессами. Однако отдельный программный механизм реализует обнаружение ошибок и восстановление от ошибок выполнения событий, которые происходят не в хронологическом порядке.

3.1.3. Схемы построения моделирующего алгоритма

Наряду со схемой событий большое распространение получила и схема процессов. Сформулируем необходимые понятия: событие, работа, процесс и др.

Событие – это изменение в системе, относящееся к какому-то моменту времени. Применительно к модели – это скачкообразное изменение одной или нескольких дискретных величин либо достижение непрерывно изменяющейся величиной заданной границы. Примеры событий: поступление заявки, освобождение прибора, наступление срока хранения скоропортящегося продукта.

Работа – действие, происходящее в моделируемой системе в течение определенного промежутка времени. Примеры: подготовка заявки, обслуживание заявки в приборе. Время, требующееся для работы, может быть случайной величиной, но к моменту имитации работы случайная величина должна получить конкретное значение (с помощью датчика случайных чисел). Работа начинается и заканчивается событиями.

В моделируемой системе событие происходит мгновенно, а работа требует времени (системного). В имитационной модели картина обратная: на имитацию события затрачивается некоторое машинное время, а имитация работы практически не требует машинного времени, так как имитируется не сама работа, а завершающее её событие.

Как правило, для начала работы требуется наличие определенных условий. Например, чтобы начать кирпичную кладку, требуются кирпичи, раствор и рабочий-каменщик. При моделировании дискретных процессов в качестве условий чаще всего рассматривается наличие ресурсов. Например, чтобы начать обслуживание заявки, требуется наличие свободного прибора. Ресурс можно понимать как оборудование, необходимое для выполнения работы (станок, прибор), но это может быть и работник. Для ресурса характерно, что он занимается в момент начала работы (или заранее) и освобождается в момент ее окончания.

Процесс – последовательность событий и работ, описывающая поведение во времени какого-либо объекта в моделируемой системе. Приведём три примера.

Процесс "генерирование заявок источником":
– подготовка заявки (работа);
– выдача заявки (событие).

Процесс "прохождение заявки":
– занятие прибора (событие);
– обслуживание заявки в приборе (работа);
– освобождение прибора (событие).

Процесс "обработка детали на станке":
– занятие рабочего (событие);
– занятие станка (событие);
– обработка детали (работа);
– освобождение станка (событие);
– контрольные измерения (работа);
– освобождение рабочего (событие).

Каждая работа или событие – это *фаза процесса*. Имитация процесса на ЭВМ производится по фазам. Фаза имитируется без прерывания, но по окончании имитации фазы возможно прерывание и переход к имитации другого процесса.

Опираясь на введенные понятия, рассмотрим основные черты схемы событий и схемы процессов, их положительные и отрицательные стороны.

При моделировании по схеме событий понятия работы и процесса не используются. Выделяется несколько событий так, чтобы они исключали друг друга и в совокупности охватывали все возможные изменения в системе. Это напоминает разбиение множества элементарных событий

на непересекающиеся подмножества (события) в теории вероятностей с той разницей, что событие - это не только подмножество элементарных событий, но и причинно-следственные связи между элементарными событиями. Например, при моделировании СМО событие "поступление заявки" охватывает несколько элементарных событий: выдача заявки источником, занятие прибора, начало обслуживания, потеря заявки (в случае отсутствия свободного прибора). Видно, что элементарные события взаимосвязаны; если не удастся занять прибор, то обслуживание не начнется, а произойдет потеря заявки. В алгоритме имитации отражаются не все элементарные события, а только те, без которых невозможно построение реализации, обеспечивающее нахождение искомых характеристик.

Основная трудность при разработке модели по схеме событий - выделение типов событий и составление алгоритмов для каждого типа так, чтобы не упустить каких-то нужных элементарных событий и правильно учесть взаимосвязи. *Каждому типу события соответствует блок в алгоритме имитации, а каждому событию - определенное место в календаре*. Алгоритм шага состоит в выяснении типа очередного события (с помощью календаря) и выполнении тех блоков алгоритма, которые соответствуют этому типу события.

При моделировании по схеме процессов за основу берётся понятие процесса. Процесс отражает поведение одного объекта (источника, заявки, детали, *рыси, зайца* и т.п.), а объектов в системе много. Поэтому *параллельно* развиваются несколько процессов. Их имитация производится поочередно (на одном процессоре). Если бы процессы были независимы, можно было бы имитировать до конца один процесс, а потом переходить к имитации другого процесса. Однако процессы пользуются общими ресурсами. Если очередная фаза процесса состоит в занятии ресурса, а свободного ресурса нет, процесс приостанавливается и ждёт освобождения ресурса. При этом ЭВМ переходит к имитации другого процесса. Таким образом, ЭВМ поочередно продвигает процессы на одну или несколько фаз, учитывая их взаимодействие во времени и в использовании общих ресурсов.

Продвижение процессов по системному времени можно осуществлять по принципу Δt или по принципу особых моментов. Будем придерживаться принципа особых моментов: какому процессу отводится место в календаре, где указывается, с какой фазы надо начать или продолжить имитацию процесса и к какому моменту системного времени это событие относится.

Процессы могут быть постоянными (т.е. развиваться бесконечно долго) и временными. *Временный процесс* создается в некоторый момент времени, существует какое-то время, после чего ликвидируется. Одна из фаз процесса может представлять собой создание нового процесса. В свя-

зи с существованием временных процессов количество заполненных мест в календаре оказывается переменным. Признаком свободного места в календаре может быть нуль в ячейке, отведенной для обозначения (метки) очередной фазы процесса. Если число одновременно существующих временных процессов не имеет ограничения сверху, число мест в календаре отводится с большим запасом, а в случае переполнения календаря должно выдаваться соответствующее сообщение. При создании процесса отыскивается свободное место в календаре и номер этого места становится номером процесса. После ликвидации процесса его место в календаре освобождается.

За один шаг (основной цикл в алгоритме) производится поочерёдный просмотр всех процессов и имитация тех из них, для которых созрели условия, в частности, наступило время исполнения очередной фазы. В ходе имитации очередной фазы процесса может освободиться ресурс, которого не хватало для имитации ранее просмотренных процессов. Кроме того, может быть создан процесс и помещен на свободное место в календаре под меньшим номером, чем текущий процесс. В обоих случаях требуется повторный просмотр процессов в рамках того же шага. Для организации повторного просмотра удобно использовать *флаг* – двоичную переменную. Этой переменной перед началом просмотра присваивается значение единицы (“подъём” флага). Освобождение ресурса или создание нового процесса влечет за собой “сброс” флага в нуль. Завершение шага и переход к следующему шагу производится только если флаг остался поднятым.

Отметим теперь, что *термин “процесс” имеет два разных смысла: процесс как алгоритм и процесс как структура данных*. Процесс как алгоритм относится ко всему множеству процессов одинакового типа, а процесс как структура данных описывает конкретного представителя этого множества. Например, процесс “генерирование заявок J-м источником” может отражаться следующими величинами (их часто называют *атрибутами* процесса):

– номер источника (J);
– емкость источника (максимальное число заявок в нем);
– приоритет, присвоенный заявками этого источника;
– среднее время подготовки заявки источником;
– среднее время обслуживания заявки этого источника в приборе;
– число заявок в источнике в данный момент;
– предстоящий момент выдачи очередной заявки;
– метка очередной фазы процесса.

Первые пять величин постоянны (но у каждого источника имеются свои значения), остальные три меняются в ходе имитации. Две последние - компоненты календаря, эта совокупность данных занимает определён-

ную группу ячеек в памяти ЭВМ. Номер этой группы - это номер процесса. О требуемом количестве групп можно сказать то же, что сказано выше о требуемом числе мест в календаре и о возможности переполнения.

Процесс “генерирование заявок источником”, понимаемый как алгоритм, существует в виде части программы моделирования. Когда эта часть программы выполняется, она использует значения атрибутов для конкретного источника. Создание процесса - это заполнение группы ячеек атрибутами этого процесса. Имитация процесса сопровождается изменением переменных атрибутов. Ликвидация процесса означает освобождение группы ячеек, занятых атрибутами.

Чтобы различать указанные два смысла, можно поступить двояко. Первый путь: термин «процесс» оставить за алгоритмом, а структуру данных называть экземпляром процесса. Второй путь: *термин «процесс» применять к структуре данных, а алгоритм называть классом процессов.* В языках моделирования применяется и другие термины, но они не использует слово процесс, которое желательно сохранить.

Часть программы, управляющая динамикой модели, т.е. последовательностью выполнения отдельных блоков программы в процессе имитации, носит название *монитор*. Рассматриваемые схемы построения моделирующего алгоритма (схема событий и схема процессов) имеют существенные различия, но принцип построения монитора у них одинаковый - повелительный. *Повелительный* (императивный) принцип состоит в том, что для предстоящих событий заранее определяется время их наступления (оно записываются в календарь) и события имитируются, когда приходит их время. Календарь как бы предписывает, в какие моменты должны происходить события, в каком порядке их имитировать - отсюда и название принципа. События, которые запланированы на один и тот же момент времени, могут имитироваться в любом порядке. То обстоятельство, что при использовании принципа процессов некоторые события “забегают вперед”, не меняет сути монитора: в пределах одного процесса последовательность событий не нарушается, а её нарушение между событиями носит временный характер (сначала один процесс забегает вперед, потом другой). Главное - для всех событий указано время, когда они должны произойти.

С целью дополнительного пояснения заметим, что повелительный принцип ассоциируется с синхронными процессами. Синхронные процессы управляются внешними часами: события в них происходят в заранее заданные моменты времени.

Сопоставим *схему событий и схему процессов*, так как их можно считать конкурирующими. Схема событий более стройна: события не пересекаются, за один шаг имитируется одно событие, события имитируются в хронологическом порядке, алгоритм шага делится на этапы с чёт-

ким функциональным назначением (имитация события, пополнение статистик, планирование новых событий). Однако в сложных ситуациях довольно трудно сформировать перечень типов событий и правильно разработать соответствующие им части алгоритма. С этой точки зрения схема процессов удобнее, так как не требует при разработке алгоритма учитывать сразу все, что может происходить в системе, а допускает раздельную разработку отдельных процессов. Особенно упрощается разработка имитационных моделей при использовании универсальной системы моделирования, когда пользователю требуется только описать последовательность событий и работ в процессах, а учёт взаимодействия процессов, сбор статистики, управление порядком имитации процессов берёт на себя система моделирования. Схема процессов не позволяет выделить функционально различные части алгоритма: пополнение статистик и планирование событий исследуют с операциями смены состояний в рамках одной фазы процесса. Это чревато упущениями при разработке алгоритма. Подводя итоги, можно оказать, что на этапе начального обучения моделирование и при моделировании простых систем целесообразно применять схему событий, а при моделировании сложных систем с использованием универсальных средств предпочтительнее схема процессов. Для повышения эффективности моделирующих алгоритмов применяются семафоры и списковые структуры данных.

3.1.4. Семафоры и связные списки

Семафоры являются удобным механизмом разделения общих ресурсов. *Семафор* – это целочисленная переменная, характеризующая текущее состояние группового ресурса, например, группы одинаковых приборов. Если значение этой переменной неотрицательное, оно показывает количество свободных приборов, а если отрицательное, то без учёта минуса это будет количество процессов, стоящих в очереди к данному групповому ресурсу.

С семафорами производятся две операции, обычно обозначаемые как *P* и *V*. Операция *P* производится, когда какой-то процесс пытается занять ресурс. Состоит эта операция в вычитании единицы из семафора, что как раз отражает результат попытки. Действительно, если имелся свободный прибор, то он был занят, и потому число оставшихся свободных приборов должно быть уменьшено на единицу. Если же свободных приборов нет, процесс становится в очередь и вычитание единицы из семафора как раз отражает удлинение очереди. Операция *V*, напротив, состоит в прибавлении единицы к семафору. Она производится при освобождении прибора (элемента группового ресурса) и отражает либо уменьшение очереди, либо увеличение числа свободных приборов. Ана-

лизируя содержимое семафоров, можно определять, по какой ветви алгоритма следует направить выполнение программы. Отсюда и термин семафор.

Для организации календаря и всякого рода очередей полезно применять *связные списки* (рис. 3.7).

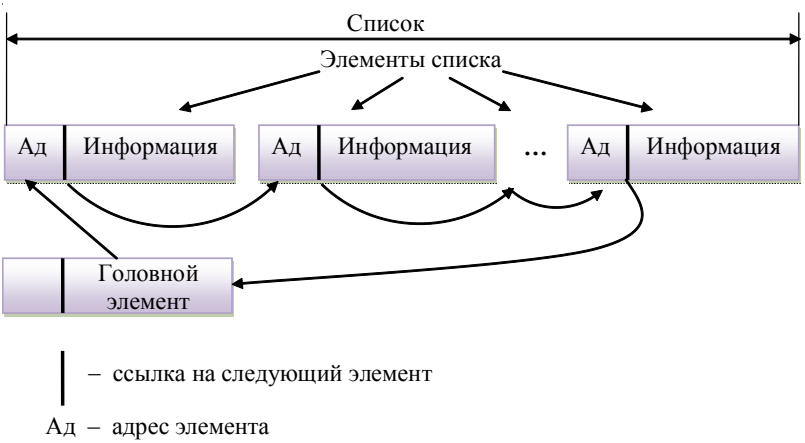


Рис. 3.7. Кольцевая очередь

Список состоит из элементов, содержащих какую-то информацию, например, данные о процессе. Элемент имеет номер, или адрес, по которому его можно разыскать в памяти ЭВМ. Связный список отличается от простого списка тем, что каждый его элемент помимо основной информации, составляющей содержание элемента, имеет *ссылку на следующий элемент*, т.е. адрес следующего элемента. Элементы как бы связаны в цепочку, причём соседние элементы этой цепочки могут быть расположены далеко друг от друга, т.е. иметь существенно различные адреса.

Связный список снабжается *головным элементом*, который состоит только из ссылки на первый элемент списка. Последний элемент на месте ссылки содержит признак конца списка, например нуль. Удобство применения связных списков в том, что для удаления элемента из списка, включения нового элемента или перестановки элементов достаточно изменить некоторые ссылки, основная же информация остается на месте без изменения.

Рассмотрим, как организуется календарь процессов с помощью связного списка. В рассмотренном выше календаре каждый процесс имел номер, обозначенный J, и содержал две величины: $T(J)$ и M. Следовательно, календарь был представлен двумя массивами: T и M – каждый длиной K. Это был простой список. Чтобы получить связный список, до-

бавим для ссылок массив A той же длины и ячейку $A\varnothing$ для головного элемента. Свяжем процессы по принципу возрастания запланированного момента активизации. Тогда $A\varnothing$ будет содержать номер процесса с наименьшим значением $T(J)$. Допустим, $T(5)\le T(7)\le T(2)$. Тогда $A\varnothing=5$, $A(A\varnothing)=A(5)=7$, $A(A(A\varnothing))=A(7)=2$ и т.д.

Выясним, что даёт связный список по сравнению с ранее рассмотренной организацией календаря в виде простого списка. Во-первых, теперь не надо просматривать подряд элементы календаря, чтобы найти те процессы, момент активизации которых совпадает с текущим моментом $t_т$, так как, если такие есть, то они занимают первые места в цепочке. Во-вторых, чтобы найти $t_т$, не надо искать минимальный элемент – это $T(A\varnothing)$.

Но за преимущества всегда надо чем-то платить. Это не только дополнительные затраты памяти на массив A, но и дополнительные операции. Если раньше для создания нового процесса достаточно было отыскать номер свободной группы Jv и написать параметры нового процесса в $T(Jv)$ и $M(Jv)$, то теперь надо еще вставить новый процесс на определенное место в цепочку. Для этого надо отыскать такой номер J, что $T(A(J))\le T(Jv)\le T(A(A(J)))$, после чего проделать следующие операции со ссылками: $A(Jv):=A(A(J))$; $A(J):=Jv$.

При ликвидации процесса с номером J помимо обычной операции $M(J):=0$ потребуется операция со ссылкой $A(J_L):=A(A(J))$, где J_L – номер процесса, стоящего в цепочке перед процессом J. Для нахождения J_L потребуется просмотреть цепочку от начала до элемента J. Чтобы избежать этого просмотра, можно применить *двусвязный список*. В нём имеется еще один массив, который содержит ссылки на предыдущий элемент цепочки.

3.1.5. Алгоритмы обслуживания очередей

Чаще всего применяют следующие алгоритмы управления (обслуживания) очередями:

– FIFO;

– приоритетное обслуживание, которое называют также «подавляющим»;

– взвешенное обслуживание.

Традиционный алгоритм FIFO. Достоинство – простота реализации и отсутствие необходимости конфигурирования. Недостаток – невозможность дифференцированной обработки заявок различных потоков.

Приоритетное обслуживание (Priority Queuing). Сначала необходимо решить отдельную задачу – разбить общий входной поток устройства на классы (приоритеты). Признак, по которому производят разбивку

на классы, помещают в поле приоритета (рис. 3.8). Затем требование помещается в очередь, соответствующую заданному приоритетному классу.

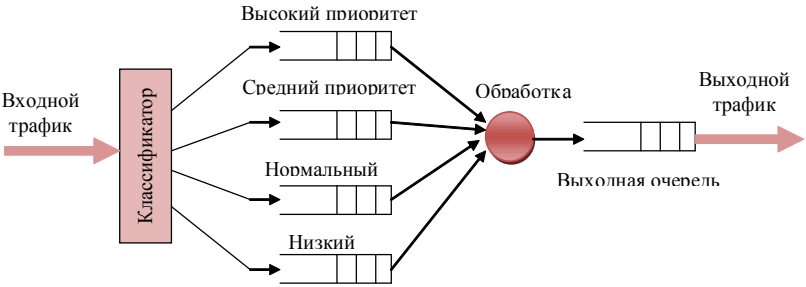


Рис. 3.8. Приоритетное обслуживание

Рассмотрим пример с четырьмя приоритетными очередями: высокий, средний, нормальный и низкий приоритет. Здесь очереди имеют абсолютный приоритет — пока не обработаны пакеты из очереди более высокого приоритета не производится переход к более низкоприоритетной очереди.

При моделировании можно:

– выделить одинаковое количество буферов для всех очередей;

– на основе анализа трафика поступлений установить нужный размер для каждой из очередей.

Недостаток: если высока интенсивность высокоприоритетного трафика, обслуживание низкоприоритетного трафика может совсем не производиться.

Этот метод можно, например, использовать при моделировании сети, если в качестве высокоприоритетного будет выбран голосовой трафик (IP - телефония). Это связано с тем, что его интенсивность невелика (обычно 4-16 Кбит/с).

Если же в сети в качестве высокоприоритетного выбран видеотрафик, остальным потокам может и не достаться пропускной способности.

Взвешенные настраиваемые очереди (Weighted Queuing). Здесь делается попытка предоставить всем классам определенный минимум пропускной способности, т.е. обслуживания, и гарантировать некоторые требования к задержкам. Под весом для каждого класса понимается процент предоставляемой классу пропускной способности (от полной выходной пропускной способности). Алгоритм, используемый администратором для назначения весов, называется настраиваемой очередью (рис.3.9).

Очереди обслуживаются циклически и в каждом цикле обслуживания из каждой очереди выбирается такой объем нагрузки, который соответствует весу этой очереди. Например, цикл = 1 с., скорость выходного

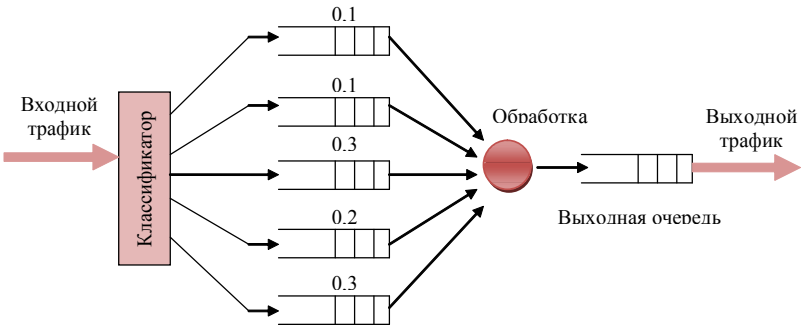


Рис. 3.9. Взвешенные настраиваемые очереди

интерфейса = 100 Мбит/с. В каждом цикле из очередей выбираются следующие объёмы данных: 1 – 10 Мбит, 2–10 Мбит, 3 – 30 Мбит, 4 – 20 Мбит, 5 – 30 Мбит. Здесь, как и в приоритетном обслуживании, можно назначать различные длины очередям. Тем самым появляется возможность сглаживания нагрузки.

Вопросы и задания

1. В чем особенность имитационного моделирования?
2. Какие виды времен учитываются при разработке имитационной модели?
3. Объясните суть метода постоянного шага отсчета системного времени.
4. В чем особенность принципа особых моментов?
5. Почему принцип особых моментов предпочтительнее принципа Δt ?
6. Что такое календарь и зачем он нужен?
7. Как осуществляется первоначальное заполнение и последующая корректировка календаря?
8. В состав алгоритма шага (цикла) входят следующие части: определение момента очередного события, изменение состояния системы в целом (имитация события), планирование событий (корректировка календаря). Что ещё добавить в указанный перечень?
9. Когда возникают парадоксы времени в имитационном моделировании и какие пути существуют для того, чтобы они не возникали?
10. Перечислите этапы процесса имитации.
11. Перечислим отличия схемы событий (СС) от схемы процессов (СП).

СС: в календаре отведены места для событий. СП: в календаре отведены места для процессов.

СС: в календаре для каждого события одна величина (момент события.).

СП: в календаре для каждого процесса две величины (момент активизации и метка фазы).

СС: за один шаг имитируется одно событие. СП: за один шаг имитируется одна или несколько фаз одного или нескольких процессов.

СС: все объекты существуют постоянно. СП: возможны временные объекты.

Продолжите этот перечень.

12. В чем состоят преимущества и недостатки схемы процессов по сравнению со схемой событий?

13. Составьте программу моделирования СМО по схеме процессов, реализующую приведенный алгоритм. Убедитесь, что она дает те же средние значения искомых характеристик, что и программа, построенная по схеме событий.

14. Измените алгоритм моделирования СМО так, чтобы он учитывал возможность существования неограниченной очереди к приборам (потерянных заявок не будет). При этом используйте семафор.

15. Сравните по быстродействию три способа организации календаря: простой список, связный список, двусвязный список.

3.2. Статистическое моделирование

Статистическое моделирование – это метод решения вероятностных и детерминированных задач, основанный на эффективном использовании случайных чисел и законов теории вероятностей. Статистическое моделирование эксплуатирует способность современных компьютеров порождать и обрабатывать за короткие промежутки времени огромное количество случайных чисел.

Подавая последовательность случайных чисел на вход исследуемой функции или модели, на её выходе получают преобразованную последовательность случайных величин – *выборку*. При правильной организации подобного статистического эксперимента выборка содержит ценную информацию об исследуемой функции или модели, которую трудно или практически невозможно получить другими способами. Информация извлекается из выборки методами математической статистики – раздела теории вероятностей.

Метод статистического моделирования (синоним этого названия – метод Монте-Карло) позволяет, таким образом, опираясь на строгие законы теории вероятностей, свести широкий класс сложных задач к относительно простым арифметико-логическим преобразованиям выборок. Поэтому такой метод получил весьма широкое распространение. В частности, он почти всегда используется при имитационном моделировании реальных сложных систем.

3.2.1. Концепция статистического моделирования

В основе статистического моделирования лежит процедура, применяемая для моделирования случайных величин и функций и носящая название метода статистических испытаний (метод Монте-Карло).

Общая схема метода Монте-Карло может быть записана в виде

$$\theta = \int y(x)\,p(x)dx \approx \tilde{\theta} = \frac{1}{N}\sum_{i=1}^{N} y(x_i),\ \ x_i \approx p(x). \qquad (3.2.1)$$

Результат ищется как математическое ожидание некоторой случайной величины Y, которая чаще всего является неслучайной функцией случайной величины X, имеющей распределение $p(x)$. Случайная величина X имеет распределение $p(x)$ и запись $X \sim p(x)$ означает, что для непрерывной случайной величины плотность вероятности равна $p(x)$; для дискретной случайной величины функцию $p(x)$ надо понимать как функцию вероятности. Для случайной дискретной величины интеграл (3.2.1) заменяется суммой $\sum y(x)p(x)$, в которой суммирование осуществляется по всем возможным значениям X. Функция $y(x)$ может иметь несколько аргументов, т.е. зависеть от нескольких случайных величин. В таком случае запись (3.2.1) остается в силе, только интеграл надо считать многомерным, X рассматривать как вектор, а $p(x)$ – как многомерную плотность (или функцию) вероятности. Приближенная оценка неизвестного математического ожидания, совпадающая с искомым результатом, находится как среднее арифметическое результатов независимых опытов. Это отражено в правой части (3.2.1). По закону больших чисел среднее арифметическое сходится к математическому ожиданию.

В каждом опыте разыгрывается реализация x случайной величины X (в i-м опыте реализация x_i) в соответствии с распределением $p(x)$ и вычисляется значение функции в виде $y(x_i)$. Индекс i подчеркивает, что для каждой (i-й) реализации процесса аргументы, составляющие вектор X, имеют свои случайные значения. Вычисленное очередное значение $y(x_i)$добавляется к накапливаемой сумме $\sum y(x_i)$. На этом заканчивается очередной опыт. После того как проведено М опытов, вычисляется итоговая оценка в виде правой части выражения (3.2.1).

Опыты повторяются до тех пор, пока дисперсия оценки $\tilde{\theta}$ не снизится до требуемой величины, зависящей от допустимой погрешности и коэффициента доверия.

Проиллюстрируем суть метода Монте-Карло следующим примером.

Пример: оценка надежности системы.

Пусть требуется оценить надежность системы, структура которой представлена на рис. 3.10.

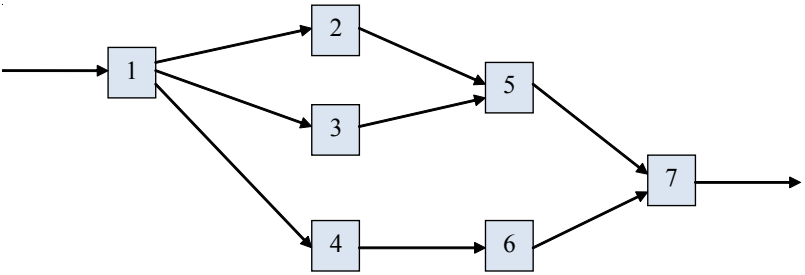


Рис. 3.10. Блочная структура системы

Система выполняет свою функцию, если работают цепочки блоков: 1,2,5,7; 1,3,5,7; 1,4,6,7. Каждый блок характеризуется временем безотказной работы τ_i, $i=\overline{1,7}$. Пусть заданы плотности распределения $p_i(\tau_i)$, $i=\overline{1,7}$. Какие-то блоки могут отказать. Какова надежность системы в целом? В качестве критерия оценим среднее время безотказной работы данной системы. Рассмотрим случайную величину

$$\gamma = \min\{\tau_1, \max[\min(\tau_4, \tau_6), \min[\max(\tau_2, \tau_3),]], \tau_7\}, \qquad (3.2.2)$$

где γ – время безотказной работы системы.

Применим метод статистических испытаний. В отдельном опыте разыгрываются значения всех τ_i, $i=\overline{1,7}$ в соответствии с $p_i(\tau_i)$, $i=\overline{1,7}$. Используя полученные реализации τ_i, $i=\overline{1,7}$, по формуле (3.2.2) вычисляем отдельную реализацию γ. Один опыт дает одну реализацию (одно выборочное значение) γ. Проводим N опытов (испытаний), получаем «статистический» материал (выборку). Берём среднее арифметическое времени безотказной работы системы в качестве оценки надежности системы. При необходимости можно построить закон распределения вероятностей случайной величины γ в виде соответствующей гистограммы.

Таким образом, испытания реальной системы заменены испытаниями математической модели. Каждое испытание сопровождается расчётом. Поэтому имитационное моделирование и называют численным экспериментом на ЭВМ с математической моделью (модель выступает как объект исследования). При реализации испытания возможны и логические операции. И расчетные, и логические операции реализуются на ЭВМ с помощью соответствующих алгоритмов, которые в совокупности и составляют моделирующий алгоритм.

Моделирующий алгоритм обеспечивает построение траекторий смены состояний системы во времени, а воспроизведение случайных факторов, определяющих эти состояния, конструируется с использованием заданных законов случайных событий и величин. Значения произвольных случайных величин реализуются преобразованиями значений базовой случайной величины (БСВ).

3.2.2. Моделирование случайных факторов

Случайными факторами называют случайное событие, случайную величину, случайный процесс и вообще любые объекты, случайный выбор которых определяется соответствующими вероятностями. Под реализацией случайного фактора понимают сам акт случайного выбора одного такого объекта из заданного множества, наделённого вероятностной мерой.

Построение модели любого случайного фактора сводится к подходящему функциональному преобразованию базовых случайных величин z_i. При моделировании случайных факторов исходят из того, что имеющийся в распоряжении датчик БСВ идеален, т. е. значения z_i на его выходе равномерно распределённые ($z_i \sim R[0,1]$) и независимые. Это позволяет конструировать модели разнообразных случайных факторов и устанавливать их свойства, опираясь только на положения теории вероятностей и не прибегая к громоздким процедурам статистического тестирования.

В целях наглядности представления результатов будем модели случайных факторов иллюстрировать некоторыми примерами простой статистической обработки выборок, получаемых с помощью этих моделей.

Построение и тестирование датчиков БСВ.

Базовой случайной величиной (БСВ) в статистическом моделировании называют непрерывную случайную величину z, равномерно распределенную на интервале ($0 \le t \le 1$). Её плотность распределения вероятностей (п. р. в.) задаётся формулой:

$$f(t) = 1, \qquad (0 \le t \le 1). \qquad (3.2.3)$$

Математическое ожидание (м. о.) $M(z)$ и дисперсия $D(z)$ базовой случайной величины z имеют следующие значения: $M(z) = 1/2$ и $D(z) = 1/15$. Нетрудно определить и начальный момент r-го порядка: $M(z^r) = 1/(r+1)$, $(r = 1, 2, \ldots)$.

БСВ моделируются на ЭВМ с помощью программных датчиков. Датчик БСВ – это программа, выдающая по запросу одно случайное значение БСВ . Путём многократного обращения к датчику БСВ получают выборку независимых случайных значений $z_1, z_2, z_3, \ldots, z_n$.

Программный датчик БСВ обычно вычисляет значения $z_1, z_2, z_3, \ldots$ по какой-либо рекуррентной формуле вида:

$$z_{i+1} = f(z_i) \qquad (3.2.4)$$

при заданном стартовом значении z_0.

Заданное значение z_0 полностью определяет посредством формулы (3.2.4) всю последовательность $z_1, z_2, z_3, \ldots$, поэтому величину z на выходе датчика БСВ называют *псевдослучайной* величиной. В практическом применении датчиков БСВ статистические свойства псевдослучайной последовательности чисел в широких пределах идентичны свойствам «чисто случайной» последовательности.

Программные датчики БСВ обладают, по сравнению с аппаратными датчиками, следующими достоинствами:

– простотой создания датчиков;

– простотой применения;

– простотой тиражирования датчиков;

– надёжностью;

– быстродействием;

– компактностью;

– высокой точностью достижения необходимых статистических свойств, сравнимой с точностью представления вещественных чисел;

– возможностью повторного воспроизведения, когда это нужно, любой последовательности случайных значений без их предварительного запоминания.

Путём преобразования БСВ можно получать модельные реализации многих других случайных объектов, включая любые непрерывные или дискретные случайные величины (как простые, так и многомерные), случайные события, случайные процессы, графы, схемы и т. д. Поэтому БСВ z называют *базовой* случайной величиной.

Для построения программно реализуемых датчиков широко используется конгруэнтный метод. Так называемый мультипликативный конгруэнтный датчик БСВ однозначно задаётся двумя параметрами: *модулем* m и *множителем* k. Обычно эти параметры представляют собой достаточно большие целые числа. При заданных m, k псевдослучайные числа z_i вычисляются мультипликативным конгруэнтным датчиком по рекуррентной формуле:

$$A_i = (k \times A_{i-1}) \bmod m,\ i = 1, 2, \ldots \qquad (3.2.5)$$

$$z_i = A_i / m,$$

где m – модуль, k – множитель, A_0 – начальное значение, mod – операция вычисления остатка от деления произведения (kЧА$_{i-1}$) на m. Заметим, что вычисление остатка от деления эквивалентно выбору младшей цифры частного в системе счисления с основанием m.

Датчик (3.2.5) даёт *периодическую* псевдослучайную последовательность $z_1, z_2, \ldots$ с длиной периода $T \le m-1$. Чтобы длина периода T была максимальной, модуль m берут близким к максимальному представимому в компьютере целому числу.

Для упрощения операций деления и вычисления остатков в двоичных ЭВМ часто берут $m = 2^n$. Рекомендуется также брать достаточно большой множитель k, причем взаимно простой с m.

Заметим, что не существует рекомендаций, гарантирующих высокое качество датчиков до того, как будет проведено их специальное тестирование. Обозначим равномерное распределение вероятностей на интервале (0, 1) через $R[0,1]$ и утверждение, что БСВ z имеет распределение $R[0,1]$, кратко запишем в виде $z \sim R[0,1]$.

С помощью статистических тестов проверяют два свойства датчика БСВ, делающих его точной моделью идеальной математической БСВ: во-первых, проверяют *равномерность распределения* чисел, выдаваемых датчиком на интервале (0, 1), и, во-вторых, их *статистическую независимость*. При этом последовательность псевдослучайных чисел z_i на выходе датчика рассматривают как статистическую выборку.

Проверка равномерности распределения БСВ сводится к построению эмпирических вероятностных характеристик (моментов и распределений) случайной величины (с.в.) z по выборке $z_1, z_2, z_3, \ldots, z_n$ и их сравнению с теоретическими характеристиками равномерного распределения $R[0,1]$. В силу закона больших чисел с ростом длины выборки n эмпирические характеристики должны приближаться к теоретическим. При этом, поскольку компьютер позволяет легко получать выборки весьма большой длины, такое сближение эмпирических и теоретических характеристик можно наблюдать непосредственно, без использования специальных статистических тестов.

Простейшую проверку статистической независимости БСВ можно осуществить, оценивая линейную корреляцию между числами z_i и z_{i+s}, отстоящими друг от друга в псевдослучайной последовательности на фиксированный шаг $s \ge 1$. Тогда во всей выборке $z_1, z_2, \ldots, z_n$ имеем следующие $(n-s)$ реализаций пары

$$(z_1, z_{1+s}), (z_2, z_{2+s}), \ldots, (z_{n-s}, z_n).$$

По этим реализациям можно рассчитать оценку R коэффициента корреляции для значений БСВ по формуле

$$R = 12 \frac{1}{n-s} \left(\sum_{i=1}^{n-s} z_i z_{i+s} \right) - 3.$$

Коэффициент корреляции двух случайных величин (с.в.) характеризует степень линейной зависимости между ними. Поэтому с ростом длины выборки *n* оценка должна приближаться к нулю. Если это выполняется, то тест на *линейную* зависимость можно считать успешно пройденным.

Как явная дискретность БСВ z_i, так и явная их функциональная зависимость (каждое следующее значение БСВ однозначно определяется предыдущим $z_i=[(km \cdot z_{i-1})\text{mod } m]/m)$ логически несовместимы с требованиями непрерывности и независимости, но на практике это имеет ограниченное значение. В то же время, это замечание о противоречивом характере требований к датчикам БСВ указывает на необходимость применения только тех датчиков БСВ, которые разработаны и тщательно выверены компьютерными математиками-профессионалами.

В настоящее время, как правило, любые языки программирования и пакеты моделирования содержат встроенные датчики БСВ и необходимость в самостоятельной разработке или тестировании датчиков возникает редко.

Моделирование случайных событий.

Случайные события бывают различного типа: простыми и сложными, совместными и несовместными, зависимыми и независимыми. Случайные величины – дискретными и непрерывными. Поэтому моделирование случайных событий и величин подразделяется на ряд отдельных процедур в зависимости от условий. Рассмотрим их по мере усложнения.

Моделирование случайного события. Основной характеристикой любого случайного события *A* является вероятность его появления – $P(A)$. Процедура моделирования простого случайного события *A* состоит из:

– формирования значения БСВ z_iR;

– сравнения z_i с заданной вероятностью $P(A)$ появления события *A*. Если условие $z_i < P(A)$ выполняется, то исходом моделирования является событие *A*.

В противном случае – противоположное событие $\overline{A}$. $P(\overline{A}) = (1 - P(A))$.

Алгоритм моделирования отдельных случайных событий имеет вид, приведенный на рис. 311.

Моделирование полной группы несовместных случайных событий. Пусть задано некоторое множество элементарных исходов $\{A_1, \ldots, A_n\}$ с их вероятностями $p_1, \ldots, p_n$ соответственно $(p_1 + \ldots + p_n = 1)$. Чтобы построить программную модель, «оживляющую» такую совокупность исходов, разобьём мысленно интервал значений БСВ $(0 \le t \le 1)$ на *n* отрезков

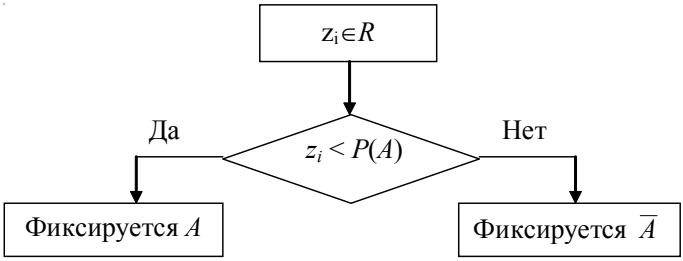


Рис. 3.11. Алгоритм моделирования отдельных случайных событий

длиной $p_1, p_2, ..., p_n$. Это всегда возможно, так как $p_1 + ... + p_n = 1$. Например, можно определить отрезки так:

$$\alpha_1 = (0, p_1),$$
$$\alpha_2 = (p_1,\ p_1 + p_2),$$
$$\alpha_3 = (p_1 + p_2,\ p_1 + p_2 + p_3),$$
$$\ldots$$
$$\alpha_n = (p_1 + p_2 + \ldots + p_{n-1},\ 1).$$

Алгоритм моделирования случайных исходов A_j состоит в том, чтобы, обратившись к датчику БСВ, определить, в какой из интервалов $\alpha_1, \alpha_2, ..., \alpha_n$ попало значение БСВ. Факт его попадания в конкретный интервал α_j предопределяет переход алгоритма к процедуре имитации соответствующего, имеющего тот же номер, исхода A_j. Поскольку вероятность попадания БСВ в интервал α_j равна его длине p_j, то и вероятность исхода A_j будет равна p_j. Такой метод моделирования простых независимых событий называют «исход испытания по жребию».

В качестве примера построим модель операции, состоящей в вытаскивании шара из урны, содержащей пять белых шаров (Б), три красных (К) и два черных (Ч). Так как исходы Б, К, Ч имеют вероятности $p_1 = 0{,}5$, $p_2 = 0{,}3$ и $p_3 = 0{,}2$ соответственно, то интервал (0,1) разбиваем на отрезки (0;0,5), (0,5;0,8) и (0,8;1).

Алгоритм моделирования имеет примерно следующий вид:

1. Получить значение z из датчика БСВ.

2. Если $z \le 1/2$, вывести “Б”, иначе, если $z \le 8/10$, вывести “К”, иначе вывести “Ч”.

Приведём пример 60-кратного выполнения этого алгоритма на компьютере; мы видим, что частота появления каждого исхода примерно соответствует его вероятности:

{БКБКБКБКББКЧБББКККБКЧЧББЧЧКББЧБЧБЧБКЧЧБББКККБЧКБЧКБЧКБЧББ}

Так, исход "Б" здесь появился 31 раз (52% случаев), "К" – 15 раз (25%) и "Ч" – 14 раз (23%).

Моделирование сложных случайных событий. Сложные события являются исходом, зависящим от двух или более простых событий. Модели этого типа рассматриваются для двух случаев: для независимых простых событий и для зависимых простых событий.

Генерирование сложного события, являющегося результатом наблюдения простых независимых случайных событий рассмотрим на примере, когда даны два простых события A и B, для которых заданы вероятности их появления $P(A)$ и $P(B)$ соответственно. События A и $\overline{A}$ как и B и $\overline{B}$, образуют полные группы несовместных событий, т.е.

$$P(A)+P(\overline{A})=1;\ P(B)+P(\overline{B})=1.$$

Возможные исходы совместных испытаний: AB, $A\overline{B}$, $\overline{A}B$, $\overline{A}\ \overline{B}$ и соответствующие этим исходам вероятности $P(A)P(B)$, $P(A)P(\overline{B})$, $P(\overline{A})P(B)$, $P(\overline{A})P(\overline{B})$. Эти сложные исходы образуют полную группу независимых событий, т.е. $P(A)P(B)+P(A)P(\overline{B})+P(\overline{A})P(B)+P(\overline{A})P(\overline{B})=1$. Используя эти вероятности и способ «исход испытания по жребию» разыгрываем все возможные исходы.

Второй способ моделирования заключается в поочередном моделирования события A, а затем события B, или, наоборот, сначала B, затем A. Получаем один из четырех приведенных выше исходов.

Генерирование зависимых случайных событий.

В тех случаях, когда A и B являются зависимыми событиями, процедура моделирования сложного события выполняется несколько иначе. Исходными данными для модели являются вероятности появления событий A, B и B/A, т.е. соответственно $P(A)$, $P(B)$ и $P(B/A)$ – условная вероятность появления события B при условии, что событие A наступило.

Возможным исходам AB, $A\overline{B}$, $\overline{A}B$, $\overline{A}\ \overline{B}$ соответствуют вероятности:

$$P(A)\,P(B/A),\ P(A)\,(1-P(B/A)),\ (1-P(A))P(B/\overline{A}),\ P(\overline{A})P(1-P(B/\overline{A})). \quad (3.2.6)$$

Условная вероятность $P(B/\overline{A})$ находится предварительно из формулы полной вероятности для события B:

$$P(B)=P(A)\,P(B/A)+P(\overline{A})P(B/\overline{A}).$$

Используя вероятности (3.2.6), разыгрываем возможные реализации сложного события способом «исход испытания по жребию» либо,

используя вероятности A, B и B/A, поочередно моделируем события A, а затем события B.

Моделирование дискретных случайных величин.

Принцип моделирования дискретных с.в. не отличается от принципа моделирования случайных событий. Дискретная с.в. x задаётся конечным или счётным множеством возможных значений $x_1, x_2, \ldots$ и их вероятностями $p_1, p_2, \ldots$. Она реализуется по тому же принципу, по которому моделируются случайные события. Интервал (0, 1) предварительно разбивается на отрезки, длины которых равны вероятностям $p_1, p_2, \ldots$ элементарных исходов A_j. При этом каждый конкретный исход A_j рассматривается здесь как выбор случайной величиной соответствующего значения $x = x_j$.

Пример. Пусть требуется реализовать дискретную с. в. x, принимающую значения 2; 5; 15; -30 и 3,14 с вероятностями 0,1; 0,15; 0,45; 0,2 и 0,1 соответственно. Если отрезки с длинами, равными перечисленным вероятностям, откладывать на числовой оси вправо от нуля, то их правыми границами будут точки 0,1; 0,25; 0,7; 0,9 и 1. Составим таблицу чисел (табл. 3.1), в которой определенной правой границе сопоставлена соответствующее значение дискретной с.в. x.

Для реализации датчика данной дискретной с.в. остаётся определить, между какими границами, указанными в верхней строке таблички, попадает значение БСВ, и выбрать соответствующее значение с.в. из нижней строки таблички. В двух частных случаях этот общий алгоритм реализации дискретной с. в. целесообразно упростить.

Первый случай – это случай целочисленной с.в. x, принимающей значения 0, 1, ..., n–1 с одинаковыми вероятностями, равными $1/n$. Такую дискретную с. в. можно получить с помощью БСВ z одним оператором присваивания, реализующим формулу вычисления целой части:

$$x = \lfloor n \cdot z \rfloor. \qquad (3.2.7)$$

Формулу (3.2.7) можно легко приспособить и к другим близким ситуациям. Например, для моделирования игральной кости с числом очков x от 1 до 6 можно результат её выбрасывания проимитировать по формуле:

Таблица 3.1

Правые границы дискретной с.в. x

0,1	0,25	0,7	0,9	1
2	5	15	-30	3,14

$$x = \lfloor 6 \cdot z \rfloor + 1.$$

Второй случай, когда алгоритм реализации дискретной с.в. следует упрощать, это случай дискретной с.в. с бесконечным (счётным) множеством возможных значений. В такой ситуации часто можно построить компактную программу, применяя рекурсивный вариант общего метода. Суть проблемы состоит в том, что перед программированием алгоритма разбить интервал (0, 1) на бесконечное число вероятностных отрезков с длинами $p_1, p_2, \ldots$ невозможно. Поэтому в программе сначала реализуется значение БСВ z, а затем выполняется построение и одновременно – проверка вероятностных отрезков интервала (0, 1) по одному (последовательно, одного за другим), до тех пор, пока не будет построен и проверен отрезок, в котором при его проверке обнаружится реализованное значение БСВ z. После этого случайной величине x присваивается соответствующее найденному отрезку значение x_j. Благодаря такому последовательному построению и просмотру вероятностных отрезков, для реализации любого значения x_j приходится строить лишь конечное их число. Конкретный пример использования этого метода для реализации пуассоновской с.в. приводится в [1].

Моделирование непрерывных случайных величин.

В современной учебной литературе по моделированию методы построения датчиков непрерывных с.в. наиболее полно разбираются в книге [5]. Вместе с тем, во многих языках и пакетах моделирования имеется большое число удобных для использования встроенных датчиков непрерывных с.в.. Так, например, в MS Excel в меню *Сервис/Анализ данных/Генерация случайных чисел* предоставляется возможность генерации выборок из следующего списка распределений: равномерное, нормальное, Бернулли, биномиальное, Пуассона. С учётом сказанного ниже приводятся лишь алгоритмы и формулы для реализации наиболее часто встречающихся непрерывных с.в.

Моделирование непрерывных с.в. методом обращения. Пусть требуется реализовать непрерывную с.в. x, которая имела бы заданную функцию распределения вероятностей (ф.р.в.) $F(t)=P\{x \le t\}$, т.е. требуется, чтобы было $x \sim F(t)$. Согласно методу обращения, это можно сделать с помощью БСВ z по следующей формуле:

$$x = F^{-1}(z).$$

Примечание. Выражение $x=F^{-1}(z)$ для расчета x через z можно получить следующим известным способом:

1. Записать формальное уравнение $F(x) = z$.
2. Разрешить его относительно x.

Моделирование экспоненциальной с.в. Экспоненциальная с.в. x имеет ф.р.в.:

$$F(t) = 1 - e^{-\lambda t},$$

где $t \geq 0$ и параметр $\lambda > 0$. Для экспоненциальной с.в. x

$$M(x) = 1/\lambda, \quad D(x) = 1/\lambda^2.$$

Для построения генератора такой с.в. используем метод обращения.

1. Записываем формальное уравнение $F(x) = z$:

$$F(t) = 1 - e^{-\lambda t} = z.$$

2. Решаем его относительно x:

$$x = -(1/\lambda) \ln(1 - z). \qquad (3.2.8)$$

Формулу (3.2.8) можно упростить, заменив $(1-z)$ на z, так как обе эти величины совпадают по распределению. Тогда из (3.2.8) получаем:

$$x = -(1/\lambda) \ln(z). \qquad (3.2.9)$$

Частотная гистограмма экспоненциальной с.в. при $M = 1$ и $n = 10000$.

При независимых z генератор (3.2.9) дает независимые значения экспоненциальной с.в. x. На рис. 3.12 приводится частотная гистограмма, полученная при испытании этого генератора для $\lambda=1$. Оценки моментов составили $M = 0{,}990$, $D = 0{,}978$.

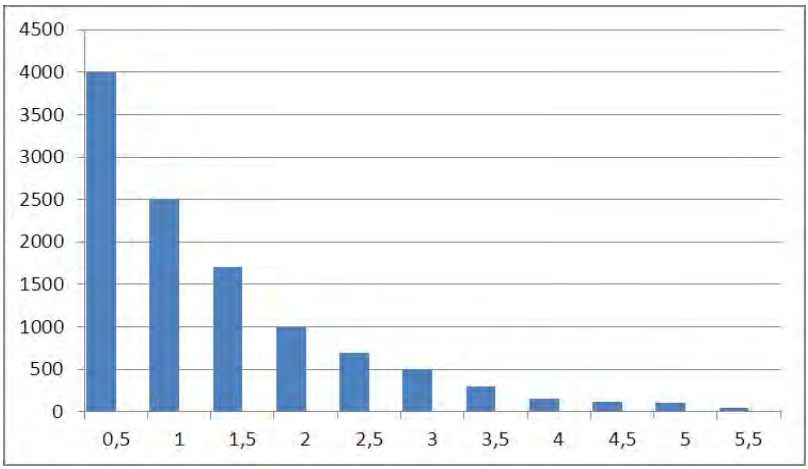


Рис. 3.12. Частотная гистограмма экспоненциальной с.в. ($M = 1$) при $n = 10000$

Моделирование равномерной случайной величины.

Построим датчик равномерно распределённой на интервале $A \le t \le B$ с.в. x, используя метод обращения.

Функция распределения с.в. $x \sim \mathrm{R}[A, B]$ имеет вид

$$.F(x) = (x - A) / (B - A)$$

Отсюда получаем модель для разыгрывания значений с.в., равномерно распределённой на интервале $A \le t \le B$, в виде

$$x = A + (B - A) \cdot z .$$

Для равномерной с.в. математическое ожидние и дисперсия равны

$$M(x) = A + (B–A)/2,\ D(x) = (B–A)^2/12.$$

На рис. 3.13 приводится гистограмма относительных частот, полученная при испытании генератора равномерной с.в. Оценки моментов составили $M = 0{,}5$, $D = 1/12$.

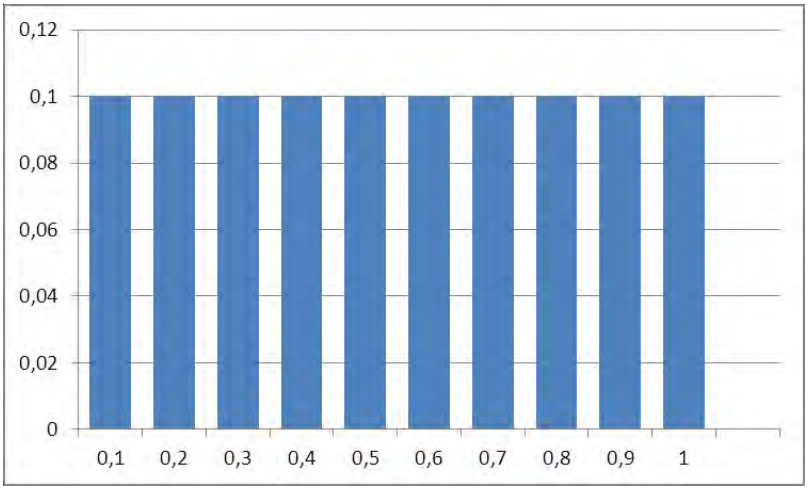


Рис. 3.13. Гистограмма относительных частот
равномерной с.в. (M = 0,5) при n = 10000

Моделирование эрланговской случайной величины.

Эрланговская с.в. x порядка $k \ge 1$ имеет функцию распределения вероятностей

$$F(t) = 1 - \left(\sum_{i=0}^{k-1} \frac{(\lambda t)^i}{i!} \right) \cdot e^{-\lambda t}, t \ge 0,$$

и плотность распределения вероятностей

$$f(t) = \frac{\lambda(\lambda t)^{k-1}}{(k-1)!} \cdot e^{-\lambda t}, \qquad t \geq 0,$$

где параметр $\lambda > 0$. Её математическое ожидание и дисперсия таковы:

$$M(x) = k/\lambda, \qquad D(x) = k/\lambda^2.$$

Поскольку распределением Эрланга обладает сумма k независимых экспоненциальных с.в., имеющих одно и то же значение параметра λ, то, с учётом (3.2.9), сгенерировать эрланговскую с.в. x можно просто как сумму:

$$x = \sum_{i=1}^{k} -\frac{1}{\lambda} \ln z_i = -\frac{1}{\lambda} \ln(z_1 \ldots z_k), \qquad (3.2.10)$$

где z_i ($i = 1, \ldots, k$) – независимые реализации базовой случайной величины.

На рис. 3.14 приводятся результаты моделирования 10000 значений x по (3.2.10) при λ=1, k=3. Так как $x = x_1 + x_2 + x_3$, где $M(x_i) = 1/\lambda = 1$, $D(x_i) = 1/\lambda^2 = 1$, i=1,2,3, то $M(x)$=3, $D(x)$=3. Статистические оценки, полученные при испытаниях, составили $M(x) = 2{,}968$, $D(x) = 2{,}958$.

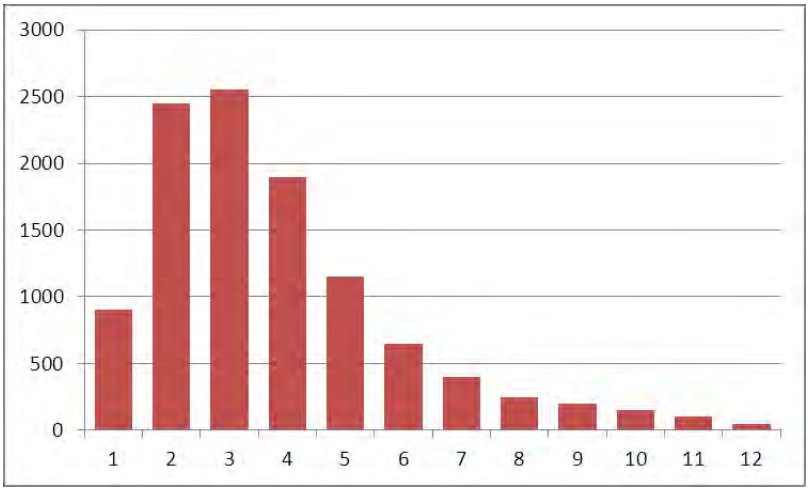


Рис. 3.14. Гистограмма эрланговской с.в. 3-го порядка при n = 10000

Моделирование нормальной случайной величины.

Утверждение, что некоторая с.в. x имеет нормальное распределение с м.о. $M(x) = m$ и дисперсией $D(x) = \sigma^2$, будем записывать в виде $x \sim N[m, \sigma]$. Функция распределения вероятностей такой с.в. имеет вид

$$F(x) = \frac{1}{2}\left[1 + \Phi\left(\frac{(x-m)}{\sigma}\right)\right]. \qquad (3.2.11)$$

Составив уравнение $F(x) = z$ и разрешив его относительно x, получим модель для разыгрывания значений нормальной с.в.:

$$x = m + \sigma\Phi^{-1}(2z - 1). \qquad (3.2.12)$$

Практическое применение формулы (3.2.12) при использовании БСВ затруднительно, так как оно связано с необходимостью вычисления обратной функции Лапласа $\Phi^{-1}(\cdot)$. Аналитических формул для этого нет. Существуют и применяются в расчётах таблицы значений обратной функции. Алгоритмизировать же расчёт обратной функции Лапласа в процедурах ИМ при необходимости получения нормально распределенных чисел, представляется возможным только с применением приближенных формул. Для того чтобы это избежать, применяется другой метод.

Для реализации любой нормальной с.в. достаточно иметь датчик стандартной (т. е. нормированной и центрированной) нормальной с.в.

$$\dot{x} \sim N(0,1).$$

Чтобы реализовать с.в. x с распределением (3.2.11) используют следующее линейное преобразование стандартной нормальной с.в. $\dot{x}$:

$$x = \sigma\dot{x} + m.$$

При этом стандартную нормальную с.в. часто реализуют приближённо как сумму других с.в., основываясь на центральной предельной теореме теории вероятностей. Например, её можно реализовать в виде суммы двенадцати независимых значений БСВ:

$$\dot{x} = \sum_{i=1}^{12} z_i - 6. \qquad (3.2.13)$$

Однако такой подход даёт плохое приближение для больших уклонений от среднего, превышающих 2σ.

Метод Бокса и Мюллера позволяет получить два независимых значения $\dot{x}_1$ и $\dot{x}_2$ стандартной нормальной с. в. из двух независимых значений z_1 и z_2 базовой случайной величины по формулам:

$$\begin{cases} \dot{x}_1 = \sqrt{-2\ln z_1}\,\sin(2\pi z_2), \\ \dot{x}_2 = \sqrt{-2\ln z_1}\,\cos(2\pi z_2). \end{cases} \qquad (3.2.14)$$

На рис. 3.15 приведена гистограмма нормальной с.в., полученной методом Бокса и Мюллера при n=10000.

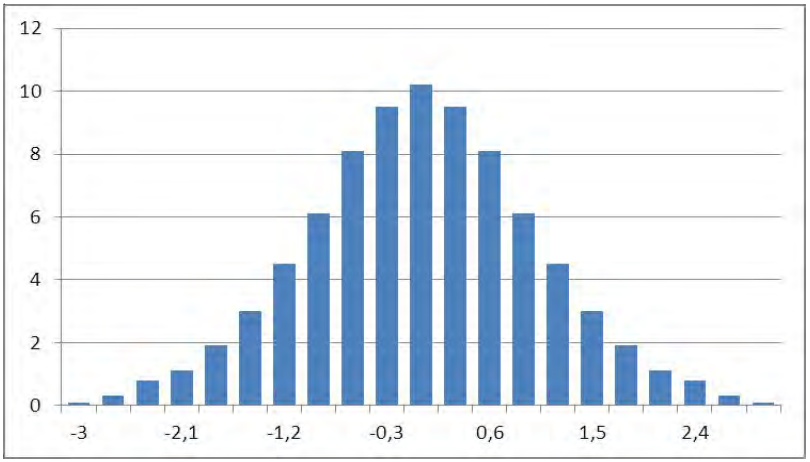


Рис. 3.15. Гистограмма нормальной с.в. при n = 10000

Этот метод точный, но считается трудоёмким [4]. Однако, как показывают эксперименты, это мнение устарело ввиду того, что современные персональные компьютеры оснащены арифметическими сопроцессорами. Точный метод (3.2.14) в действительности оказывается также и более быстрым, чем приближённый метод (3.2.13) [7].

Метод отбора. Метод предложен фон Нейманом. Называется также методом Неймана, методом исключения [3].

Рассмотрим вариант метода, когда требуемое распределение задано плотностью вероятности, отличной от нуля на конечном интервале $[a,b]$. Алгоритм состоит в следующем. Из очередной пары базовых чисел z^1 и z^2 находятся два числа:

$$u_1 = a + (b - a) \cdot z^1,$$
$$u_2 = f_m \cdot z^2, \qquad (3.2.15)$$

где f_m максимальное значение плотности $f(x)$.

Если $u_2 < f(u_1)$, т.е. если точка (u_1,u_2) лежит под графиком $f(x)$ (рис. 3.16), то u_1 принимается в качестве очередного числа x_n. В противном случае берётся следующая пара базовых чисел, и процедура повторяется.

Убедимся, что отобранные числа имеют требуемый закон распределения. Согласно (3.2.15) точка (u_1,u_2) равномерно распределена в прямоугольнике с основанием $[a,b]$ и высотой f_m, площадь которого равна

$$S = (b - a) \cdot f_m.$$

Следовательно, вероятность попадания точки под график $f(x)$ равна отношению площади под графиком к площади прямоугольника, т.е. равна

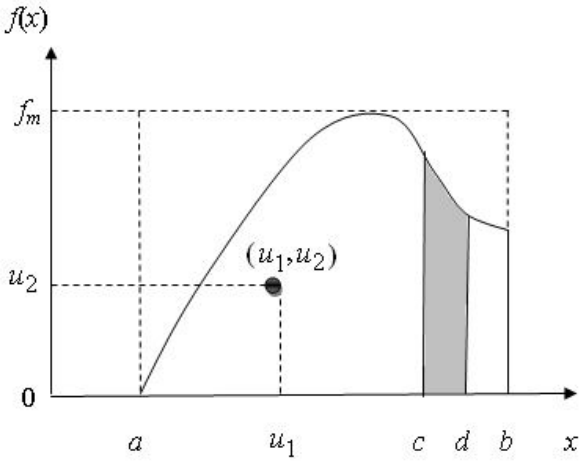


Рис. 3.16. Иллюстрация метода отбора

$$\frac{1}{S}\int_a^b f(x)dx = \frac{1}{S},$$

а вероятность попадания точки под график в пределах участка [*c,d*] (заштрихованная область на рис. 3.16) равна

$$\frac{1}{S}\int_c^d f(x)dx .$$

Таким образом, доля точек, отобранных в качестве x_n на участке [*c,d*] по отношению ко всем точкам составляет

$$\frac{1}{S}\int_c^d f(x)dx \;:\; \frac{1}{S} = \int_c^d f(x)dx ,$$

что и означает соблюдение требуемого закона распределения.

3.2.3. Точность результатов моделирования

При моделировании методом Монте-Карло многократно повторяются опыты со случайным исходом и оценка $\overline{\theta}$ искомого результата θ определяется путём осреднения результатов отдельных опытов:

$$\overline{\theta} = \frac{x_1 + \ldots + x_n}{n}, \qquad (3.2.16)$$

где x_i результат *i*-го опыта.

Алгоритм моделирования строится так, что математическое ожидание результата одного опыта совпадает с искомым результатом, т.е.

$$M[x_i] = \theta. \qquad (3.2.17)$$

При этом оценка (3.2.17) оказывается несмещённой, т.е.

$$M[\ \bar{\theta}\ ] = \theta. \qquad (3.2.18)$$

Хотя в среднем оценка $\bar{\theta}$ совпадает с θ, в каждом конкретном случае она отличается от θ на случайную величину, средний квадрат которой представляет собой дисперсию оценки. Чем больше дисперсия, тем больше случайные отклонения оценки от искомого результата, т.е. меньше её точность. С увеличением числа опытов дисперсия оценки уменьшается, а точность, соответственно, растет. Ниже выводится правило, указывающее, до какой величины надо снижать дисперсию путем увеличения числа опытов, чтобы можно было считать оценку достаточно точной.

Поскольку оценка является случайной величиной, точность ее рассматривается применительно ко всему множеству возможных значений оценки с учетом вероятностей этих значений. Требования к точности обычно формулируются в виде вероятностного утверждения:

$$P\{\ |\ \bar{\theta}\ -\ \theta\ |\leq \Delta_{\text{доп}}\} = \gamma, \qquad (3.2.19)$$

где $\Delta_{\text{доп}}$ – допустимая погрешность; γ – заданный коэффициент доверия.

В конкретном случае значение оценки может отличаться от искомого результата больше чем на $\Delta_{\text{доп}}$. Однако с вероятностью γ это не произойдет, если условие (3.2.19) соблюдено.

Связь между $\Delta_{\text{доп}}$ и γ зависит от распределения оценки . Согласно центральной предельной теореме, теория вероятностей, оценка (3.2.16) имеет распределение, близкое к нормальному, если *n* не слишком мало. С учётом (3.2.18) можно записать

$$\bar{\theta} \sim \mathrm{N}(\theta, \sigma^2) \text{ или } \bar{\theta} \sim N(0, 1)^3.$$

Следовательно,

$$P\left\{\left|\bar{\theta} - \theta\right| \leq k\sigma\right\} = P\left\{-k \leq \frac{\bar{\theta} - \theta}{\sigma} \leq k\right\} = \frac{1}{\sqrt{2\pi}} \int_{-k}^{k} e^{-0{,}5t^2}\, dt.$$

Это соотношение позволяет, задав $\Delta_{\text{доп}} = \sigma$, вычислить

$$\gamma = \frac{1}{\sqrt{2\pi}} \int_{-k}^{k} e^{-0{,}5t^2} dt = 2\Phi(k) - 1,$$

---

[3] Запись $X \sim N$ (m, σ^2) обозначает, что с. в. X имеет нормальное распределение с математическим ожиданием m и дисперсией σ^2.

где $\Phi(\cdot)$ – интегральная функция стандартного нормального распределения $N(0, 1)$. В частности, для $k = 1,2,3$ значение γ равно соответственно 0,6826; 0,9545; 0,9973. Утверждая, что погрешность не превышает 3σ, мы ошибаемся менее чем в трёх случаях из 1000. Следовательно, обеспечив $\sigma = \bar{\theta}/3$, можно быть достаточно уверенным, что погрешность не превысит допустимого значения.

Из сказанного вытекает практическая рекомендация: *число опытов при моделировании должно быть настолько большим, чтобы дисперсия оценки снизилась до величины*

$$\sigma^2 = \frac{\Delta^2_{\text{доп}}}{k^2}, \qquad (3.2.20)$$

где выбор k определяется требуемым коэффициентом доверия. Чтобы применять это правило остановки, необходимо уметь вычислять дисперсию оценки при заданном числе опытов γ.

Оценка по множеству независимых реализаций. Для многих случаев результаты опытов представляют собой повторную выборку $(x_1+..+x_n)$, т.е. совокупность независимых реализаций с.в. X. Математическое ожидание её должно совпадать с θ. Дисперсию этой с.в. обозначим σ_1^2. Поскольку реализации независимы, дисперсия оценки (3.2.16) равна $\sigma^2 = \sigma_1^2 / n$.

В соответствии о правилом (3.2.20) требуемое число опытов равно

$$n = \frac{k^2 \sigma_1^2}{\Delta^2_{\text{доп}}}. \qquad (3.2.21)$$

Таким образом, *требуемое число опытов обратно пропорционально квадрату допустимой погрешности*. Этим обусловлена трудность получения высокой точности результатов при вероятностном моделировании.

Особые трудности возникают, когда искомым результатом является вероятность некоторого события, причем вероятность малая. В этом случае

$$x_i = \begin{cases} 1, \text{ с вероятностью } p \\ 0, \text{ с вероятностью } (1\text{-}p) \end{cases}$$

$$M[x_i] = p, \quad \sigma_i^2 = D[x_i] = p(1-p), \quad i = 1,...n.$$

Требуемое число опытов удобно выразить через допустимую относительную погрешность

$$\delta_{\text{доп}} = \frac{\Delta_{\text{доп}}}{p}.$$

Формула (3.2.21) принимает вид

$$n = \frac{k^2(1-p)}{\delta^2_{\text{доп}} \cdot p}. \qquad (3.2.22)$$

Чем меньше вероятность p, *тем больше требуется опытов для обеспечения допустимой относительной погрешности*. При малых вероятностях даже грубая оценка требует слишком много опытов. Например, при p порядка 10^{-5}, $k = 3$, $\delta_{\text{доп}} = 0{,}3$ формула (3.2.22) дает $n = 10^7$. Обычно такое число опытов требует много машинного времени.

Заметим, что в формулу (3.2.21) входит величина , которая заранее не известна. Её приближенно находят в процессе моделирования по формуле

$$\sigma_1^2 = \frac{1}{(n-1)}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right).$$

Проделав некоторое число опытов, вычисляют , а затем по формуле (3.2.21) находят примерно необходимое число опытов. Опыты продолжают до получения нужного их числа, после чего можно заново вычислить уже более точно. Аналогично поступают с величиной p. Вначале пробным моделированием определяют

$$p = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

а затем подставляют найденное значение в формулу (3.2.22). По мере увеличения количества опытов значения можно эпизодически уточнять.

3.2.4. Планирование статистического эксперимента

Задача планирования. Пусть $X = (x_1, \ldots, x_k)$ – вектор случайных величин, имеющий распределение P_X. Во многих случаях цель статистического эксперимента может быть сведена к определению математического ожидания с.в. $y = g(X)$, где $g(X)$ – заданная функция распределения с.в. X. Типовая схема статистического эксперимента имеет следующий вид (рис. 3.17).

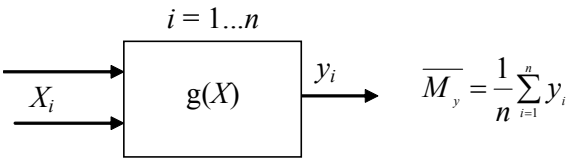


Рис. 3.17. Типовая схема статистического эксперимента

Здесь оценка $\overline{M}_y$ приближается к точному значению $M(y)$ с ростом числа опытов n. Задача *планирования* статистического эксперимента состоит в нахождении такого n, при котором достигается заданная точность оценки $\overline{M}_y$.

$$X_i \sim P_Y;\ X_i = (x_{1i},...,x_{ki});$$

y_i – значение с.в. Y в i-м опыте; n - число опытов.

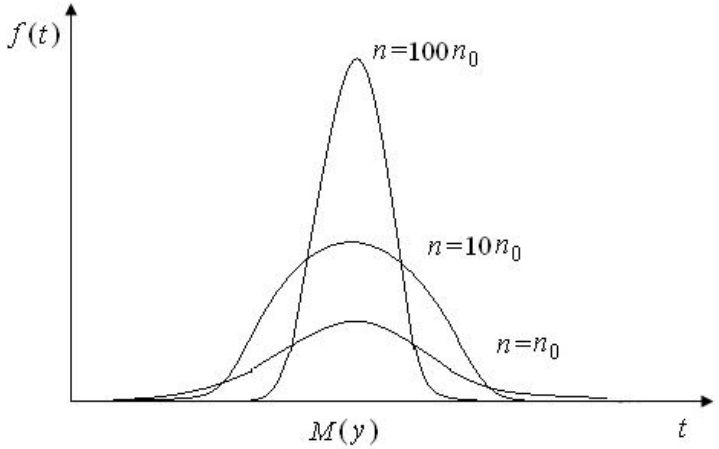


Рис. 3.18. Распределение с.в. $\overline{M}_y$ в зависимости от n

Теоретическое решение задачи планирования. Оценка $\overline{M}_y$ является случайной величиной, так как представляет собой функцию от случайных величин:

$$\overline{M}_y = \frac{1}{n}\sum_{i=1}^{n} y_i . \qquad (3.2.23)$$

В типовой схеме статистического эксперимента (рис. 3.17) используются независимые реализации x_i, поэтому с.в. y_i в формуле (3.2.23) также независимы. Из центральной предельной теоремы вытекает, что с.в. $\overline{M}_y$ в (3.2.23) при больших n имеет нормальное распределение, т.е. $\overline{M}_y \sim N[m,\sigma]$. Определим параметры этого распределения:

$$M\left(\frac{1}{n}\sum_{i=1}^{n} y_i\right) = \frac{1}{n} nM(y) = M(y), \qquad (3.2.24)$$

$$\sigma^2 = D(\overline{M}_y) = D\left(\frac{1}{n}\sum_{i=1}^{n} y_i\right) = \frac{1}{n^2} nD(y) = \frac{D(y)}{n},$$

$$\sigma = \sqrt{D(y)/n} = \sigma_y / \sqrt{n}\,. \qquad (3.2.25)$$

Таким образом, при увеличении числа опытов на порядок σ уменьшается в $\sqrt{10} \approx 3$ раза. На рис. 3.18. схематически показан вид распределения оценки $\bar{M}_y$ в зависимости от n.

Диапазон вероятных отклонений оценки от точного значения $M(y)$ сужается пропорционально $\sqrt{n}$. Параметр σ используют как показатель точности оценки. Поскольку $\bar{M}_y$ имеет нормальное распределение, то практически достоверно, что $\bar{M}_y$ отклоняется от искомого $M(y)$ не более чем на 3σ. Можно сказать, что σ является аналогом абсолютной погрешности, а 3σ – самой абсолютной погрешностью. В качестве аналога относительной погрешности для с.в. $\bar{M}_y$ можно рассматривать её коэффициент вариации:

$$v = \sigma / m,$$

а в качестве самой относительной погрешности – величину $3v$. Из (3.2.24) и (3.2.25) находим, что

$$v = \frac{\sigma}{m} = \frac{\sigma_y}{\sqrt{nM(y)}} = \frac{v_y}{\sqrt{n}}, \qquad (3.2.26)$$

где v_y – коэффициент вариации с.в. y.

Соотношения (3.2.25) и (3.2.26) показывают, что "абсолютная погрешность" – 3σ и "относительная погрешность" – $3v$ оценки $\bar{M}$ убывают пропорционально $\sqrt{n}$.

Из формулы (3.2.25) находим решение задачи планирования: для достижения заданной точности σ требуется n опытов

$$n = \left(\frac{\sigma_y}{\sigma} \right)^2. \qquad (3.2.27)$$

Если требование к точности задано в форме коэффициента вариации v, то требуемое число опытов определяется из формулы (3.2.26) в виде:

$$n = \left(\frac{v_y}{v} \right)^2. \qquad (3.2.28)$$

В решении (3.2.27) кроме заданного σ для определения n необходимо знать σ_y, в варианте (3.2.28) – коэффициент вариации v_y. Ни то, ни другое до эксперимента, как правило, не известно. Поэтому планирование числа опытов на практике осуществляется в ходе самого статистического эксперимента. Достаточно удобными методами такого планирования являются метод *автоостанова*.

Метод автоостанова. Схема статистического эксперимента с автоостановом изображена на рис. 3.19. Покажем на примере достижения точности $v = 0{,}01$.

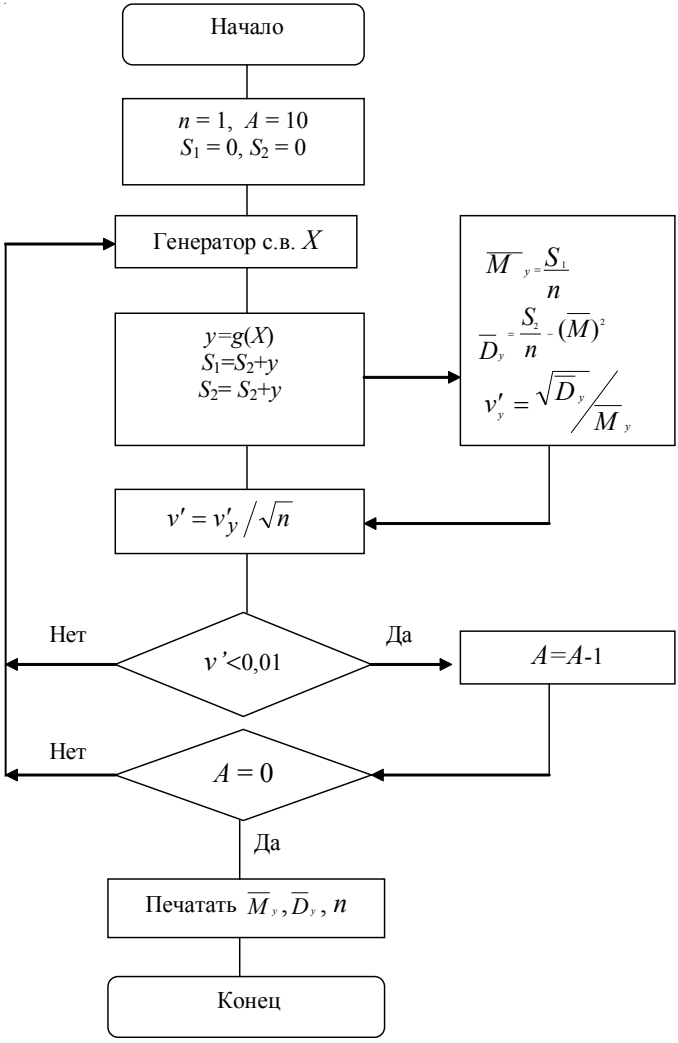


Рис. 3.19. Схема эксперимента с автоостановом

Как видно из рисунка, в ходе эксперимента ведется непрерывный контроль за оценкой v’ коэффициента вариации v. Когда оценка v’ устойчиво входит в зону v’ $< 0{,}01$, эксперимент завершается.

Устойчивое достижение неравенства $v' < 0{,}01$ здесь обеспечивается требованием, чтобы это неравенство было подтверждено $A = 10$ раз.

Заметим, что минимальное начальное значение счётчика подтверждений на рис. 3.19, задаваемое в начале алгоритма, должно составлять A=2, так как необходимо исключить одно ложное подтверждение, которое имеет место при $n = 1$. Действительно, при $n = 1$ оценка дисперсии равна нулю, поэтому получается $v' = 0$. При $A = 10$ получится несколько "лишних" повторений цикла. Однако так как n обычно составляет десятки тысяч, то увеличение его на несколько лишних единиц не имеет существенного практического значения.

Вопросы и задания

1. От каких величин зависит требуемое число опытов? Откуда берутся их численные значения?
2. Во сколько раз надо увеличить число опытов, чтобы снизить погрешность в 10 раз?
3. Приведите алгоритм моделирования отдельных случайных событий.
4. Приведите алгоритм генерирования зависимых случайных событий.
5. Какие методы тестирования датчиков БСВ Вы знаете? Приведите примеры.
6. Как можно оценить коэффициент корреляции с.в.? Что он показывает?
7. Перечислите характеристики, которыми задается генератор заявок.
8. Раскройте суть метода Монте-Карло.
9. Каким образом реализуется приоритетное обслуживание заявок?
10. Объясните, каким образом решается задача планирования статистического эксперимента.
11. Какими методами может быть оценена точность результатов моделирования?
12. Объясните каким образом осуществляется моделирование сложных случайных событий.
13. Объясните схему статистического эксперимента с автоостановом, изображенную на рис. 3.19.

3.3. Генетические алгоритмы в моделировании

Генетические алгоритмы (ГА) – это специальная технология для поиска оптимальных решений, опирающаяся на метод проб и ошибок, которая успешно применяется в различных областях науки и бизнеса.

Генетические алгоритмы часто применяются совместно с нейронными сетями, позволяя создавать предельно гибкие, быстрые и эффективные инструменты анализа данных.

3.3.1. История возникновения генетических алгоритмов

Генетические алгоритмы возникли в результате наблюдения и попыток копирования естественных процессов, происходящих в мире живых организмов, в частности, эволюции и связанной с ней селекции (естественного отбора) популяций живых существ, поэтому они получили название генетических алгоритмов.

Эволюционная теория утверждает, что каждый биологический вид целенаправленно развивается и изменяется для того, чтобы наилучшим образом приспособиться к окружающей среде. В процессе эволюции многие виды насекомых и рыб приобрели защитную окраску, ёж стал неуязвимым благодаря иглам, человек стал обладателем сложнейшей нервной системы. Можно сказать, что эволюция – это процесс оптимизации всех живых организмов. Рассмотрим, какими же средствами природа решает эту задачу оптимизации.

Основной механизм эволюции - это естественный отбор. Его суть состоит в том, что более приспособленные особи имеют больше возможностей для выживания и размножения и, следовательно, приносят больше потомства, чем плохо приспособленные особи. При этом благодаря передаче генетической информации (генетическому наследованию) потомки наследуют от родителей основные их качества. Таким образом, потомки сильных индивидуумов также будут относительно хорошо приспособленными, а их доля в общей массе особей будет возрастать. После смены нескольких десятков или сотен поколений средняя приспособленность особей данного вида заметно возрастает.

Чтобы сделать понятными принципы работы генетических алгоритмов, рассмотрим, как устроены механизмы генетического наследования в природе. В каждой клетке любого животного содержится вся генетическая информация этой особи. Эта информация записана в виде набора очень длинных молекул ДНК. Каждая молекула ДНК - это цепочка, состоящая из молекул нуклеотидов четырех типов, обозначаемых A, T, C и G. Собственно, информацию несет порядок следования нуклеотидов в ДНК. Таким образом, генетический код индивидуума - это просто очень длинная строка символов, где используются всего 4 буквы. В животной клетке каждая молекула ДНК окружена оболочкой - такое образование называется хромосомой.

Каждое врожденное качество особи (цвет глаз, наследственные болезни, тип волос и т.д.) кодируется определенной частью хромосомы,

которая называется *геном* этого свойства. Например, ген цвета глаз содержит информацию, кодирующую определенный цвет глаз. Различные значения гена называются его *аллелями*.

При размножении животных происходит взаимодействие двух родительских ДНК, образуя ДНК потомка. Основной способ взаимодействия - *кроссовер (cross-over, скрещивание).* При кроссовере ДНК предков делятся на две части, а затем обмениваются своими половинками.

При наследовании возможны мутации из-за радиоактивности или других воздействий, в результате которых могут измениться некоторые гены в клетках одного из родителей. Измененные гены передаются потомку и придают ему новые свойства. Если эти новые свойства полезны, они, скорее всего, сохранятся в данном виде - при этом произойдет скачкообразное повышение приспособленности вида.

В начале 70-х гг. XX в. была озвучена идея использования данного природного процесса в задачах оптимизации, причем вне зависимости от сферы применения полученных результатов. Высказал эту идею американский учёный Джон Холланд и предложил реализовать в виде компьютерной программы алгоритм, который будет решать сложные задачи так, как это делает природа - путем эволюции. Алгоритм оперировал последовательностями двоичных цифр (единиц и нулей), которые и получили название хромосом. Алгоритм имитировал эволюционные процессы в поколениях таких хромосом: в них были реализованы механизмы селекции и репродукции, аналогичные применяемым при естественной эволюции. Так же, как и в природе, генетические алгоритмы осуществляли поиск «хороших» хромосом без использования какой-либо информации о характере решаемой задачи. Требовалась только некая оценка каждой хромосомы, отражающая её приспособленность. Механизм селекции заключается в выборе хромосом с наивысшей оценкой (т.е. наиболее приспособленных), которые репродуцируют чаще, чем особи с более низкой оценкой (хуже приспособленные). Репродукция означает создание новых хромосом в результате рекомбинации генов родительских хромосом. Рекомбинация - это процесс, в результате которого возникают новые комбинации генов. Для этого используются две операции: скрещивание, позволяющее создать две совершенно новые хромосомы потомков путем комбинирования генетического материала пары родителей, а также мутация, которая может вызывать изменения в отдельных хромосомах. Таким образом, генетический алгоритм представляет собой метод, отражающий естественную эволюцию методов решения проблем и, в первую очередь, задач оптимизации. Генетические алгоритмы - это процедуры поиска, основанные на механизмах естественного отбора и наследования. В них используется эволюционный принцип выживания наиболее приспособленных особей. Они отличаются от традиционных

методов оптимизации несколькими базовыми элементами. В частности, генетические алгоритмы:

- обрабатывают не значения параметров самой задачи, а их закодированную форму;
- осуществляют поиск решения исходя не из единственной точки, а из их некоторой популяции;
- используют только целевую функцию, а не её производные либо иную дополнительную информацию;
- применяют вероятностные, а не детерминированные правила выбора.

Эти свойства приводят в результате к устойчивости генетических алгоритмов и к их превосходству над другими широко применяемыми технологиями.

Основным недостатком ГА является то, что неизвестно, сколько понадобится времени для решения задачи.

Только в конце 90-х г. стали появляться научные работы, показавшие применимость и значимость использования генетических алгоритмов в задачах оптимизации технических систем.

3.3.2. Интерпретация основных понятий генетических алгоритмов

При описании генетических алгоритмов используются определения, заимствованные из генетики. Например, речь идёт о популяции особей, в качестве базовых понятий применяются ген, хромосома, генотип и другие. Этим генетическим терминам соответствуют определения из технического лексикона, в частности, цепь, двоичная последовательность, итерация, структура.

Генетический алгоритм осуществляет поиск лучшего решения в пространстве поиска решений. Это пространство под воздействием механизмов эволюции изменяется в направлении «улучшения» содержащихся в нём решений. Результатом такой эволюции должно быть пространство поиска, содержащее наилучшие (или приемлемые) решения, которые и обнаруживаются алгоритмом.

В терминологии ГА пространство поиска решений интерпретируется как *популяция* $P = \{A_1,..., A_k,..., A_N\}$, а каждое решение из этого пространства – как *хромосома* (или *особь*) A_k, представляемая в виде строки символов, называемых *генами*:

$$A_k = \{a_1,..., a_j,..., a_L\}.$$

Для этого выполняется кодирование независимых хромосом либо в двоичном формате, либо в формате с плавающей запятой. Тогда геном

в этой хромосоме будет один бит. С каждым геном a_j сопоставлено множество $W_j = \{w^h_j\}$ его возможных значений w - *аллелей* («0» или «1»). На множестве решений (хромосом) задается целевая функция (ЦФ) – *функция полезности* $F = f(A_k)$ - эффективность решения A_k.

Эта функция играет важнейшую роль, поскольку позволяет оценить степень приспособленности конкретных особей в популяции и выбрать из них наиболее приспособленные (т.е. имеющие наибольшие значения функции приспособленности) в соответствии с эволюционным принципом выживания «сильнейших» (лучше всего приспособившихся). В задачах оптимизации функция приспособленности, как правило, оптимизируется (точнее, максимизируется) и называется целевой функцией. Таким образом, генетический алгоритм осуществляет поиск хромосомы A^*, для которой

$$F(A^*) = \max_p F(A_k).$$

На каждой итерации генетического алгоритма приспособленность каждой особи данной популяции оценивается при помощи функции приспособленности, и на этой основе создается следующая популяция особей, составляющих множество потенциальных решений проблемы.

Размерность (численность) N *популяции* может изменяться в процессе работы алгоритма, но наиболее распространены ГА, для которых N = const. От размера популяции зависит время поиска решения и его «качество». Важно, что размер популяции значительно меньше числа всех допустимых решений. Именно это позволяет находить решение за приемлемое время.

Очередная популяция в генетическом алгоритме называется *поколением*, а к вновь создаваемой популяции особей применяется термин *«новое поколение»* или *«поколение потомков»*.

3.3.3. Операторы генетического алгоритма

Последовательность генетических операторов обычно включает операторы отбора (репродукции), кроссинговера (скрещивание), мутации и инверсии. Все эти операторы являются вероятностными, т.е. их параметры – случайные величины.

Отбор (селекция) есть формирование следующей популяции из имеющейся в данный момент. Он имитирует принцип «выживания сильнейших» и осуществляется путем стохастической, но целенаправленной селекции, т.е. отбора лучших хромосом на основе их значений функции «полезности». Основные виды селекции: пропорциональная, турнирная и усечённием.

При *пропорциональной селекции* вероятность выбора *r*-й хромосомы определяется как

$$P(A_k) = \frac{F(A_k)}{\sum_{k=1}^{N} F(A_k)} \qquad (3.3.1)$$

при условии, что $F(A_k) > 0$ для всех $k = 1 \ldots N$.

Этот принцип программно реализуется на основе метафоры «колеса рулетки». Данный метод отбирает особей с помощью N «запусков» рулетки. Колесо рулетки содержит по одному сектору для каждого члена популяции. Всё колесо рулетки соответствует сумме значений функции приспособленности всех хромосом рассматриваемой популяции. Каждой хромосоме A_k соответствует сектор колеса $v(A_k)$, выраженной в процентах согласно формуле:

$$v(A_k) = P(A_k)100\%.$$

Поэтому чем больше значение функции приспособленности, тем больше сектор на колесе рулетки. Следовательно, члены популяции с более высокой приспособленностью с большей вероятностью будут чаще выбираться, чем особи с низкой приспособленностью (рис.3.20).

1. 000110110 15%
2. 111001101 25%
3. 000110110 40%
4. 111101111 20%

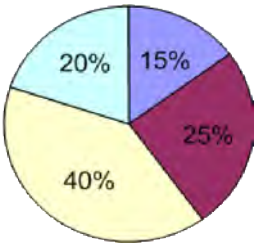


Рис. 3.20. Метод рулетки с пропорциональными секторами функции приспособленности

Если всю окружность колеса рулетки представить в виде цифрового интервала [0, 100], то выбор хромосомы можно отождествить с выбором числа из интервала [*a*, *b*], где *a* и *b* обозначают соответственно начало и окончание фрагмента окружности, соответствующего этому сектору колеса; очевидно, что $0 \le a < b \le 100$. В этом случае выбор с помощью колеса рулетки сводится к выбору числа из интервала [0, 100], которое соответствует конкретной точке на окружности колеса.

При *турнирной селекции* из популяции P_t случайным образом выбирается подмножество $P^*_t \subset P_t$ и хромосома из этого подмножества,

имеющая максимальное значение функции приспособленности, выбирается в следующую популяцию P_{t+1}. Эта операция повторяется N раз. Строки в полученном промежуточном массиве затем используются для скрещивания (также случайным образом). Размер группы строк, отбираемых для турнира, часто равен 2. В этом случае говорят о двоичном/парном турнире, а t называют численностью турнира. Преимуществом данной стратегии является то, что она не требует дополнительных вычислений и упорядочивания строк в популяции по возрастанию приспособленности.

Отбор усечением. Данная стратегия использует отсортированную по возрастанию популяцию. Число особей для скрещивания выбирается в соответствии с порогом [0; 1]. Порог определяет, какая доля особей, начиная с самой первой (самой приспособленной), будет принимать участие в отборе. В принципе, порог можно задать и числом больше 1, тогда он будет просто равен числу особей из текущей популяции, допущенных к отбору. Среди особей, попавших "под порог" случайным образом N раз выбирается самая везучая и записывается в промежуточный массив, из которого затем выбираются особи непосредственно для скрещивания. Из-за того, что в этой стратегии используется отсортированная популяция, время её работы может быть большим для популяций большого размера и зависеть также от алгоритма сортировки.

Кроссинговер (скрещивание). В простейшем случае кроссинговер в генетическом алгоритме реализуется так же, как и в биологии, его условная схема приведена на рис. 3.21. При этом хромосомы разрезаются в случайной точке и обмениваются частями между собой. Например, если хромосомы (1, 2, 3, 4, 5) и (0, 0, 0, 0, 0) разрезать между третьим и четвертым генами и обменять их части, то получатся потомки (1, 2, 3, 0, 0) и (0, 0, 0, 4, 5).

На первом этапе скрещивания выбираются пары хромосом из родительской популяции (родительского пула). Это временная популяция состоящая из хромосом, отобранных в результате селекции и предназначенных для дальнейших преобразований операторами скрещивания и мутации с целью формирования новой популяции потомков. На данном этапе

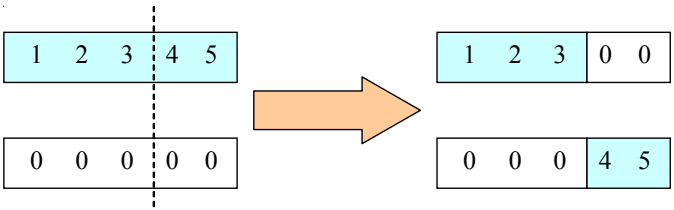


Рис. 3.21. Условная схема кроссинговера

хромосомы из родительской популяции объединяются в пары. Это объединение производится случайным способом в соответствии с вероятностью скрещивания p_c. Далее для каждой пары отобранных таким образом родителей разыгрывается позиция гена в хромосоме, определяющая так называемую точку скрещивания. Если хромосома каждого из родителей состоит из L генов, то очевидно, что точка скрещивания l_K представляет собой натуральное число, меньше L. Поэтому фиксация точки скрещивания сводится к случайному выбору числа из интервала (1, L-1]. В результате скрещивания пары родительских хромосом получается следующая пара потомков:

1) потомок, хромосома которого на позициях от 1 до l_K состоит из генов первого родителя, а на позициях от $l_K + 1$ до L – из генов второго родителя;

2) потомок, хромосома которого на позициях от 1 до l_K, состоит из генов второго родителя, а на позициях от $l_K + 1$ до L - из генов первого родителя.

Приведённые примеры относятся одноточечному кроссинговеру. Используются также различные варианты многоточечного кроссинговера, когда хромосомы-родители разрываются не в одном месте, а в нескольких.

Мутация применяется с некоторой достаточно низкой вероятностью P_m (обычно $P_m \approx 0{,}001$) к хромосомам, полученным после кроссинговера. Оператор одноточечной мутации выбирает случайным образом ген в хромосоме и меняет его значение на другое из области допустимых значений. Например, если в хромосоме [100110101010] мутации подвергается ген на позиции 7, то его значение, равное 1, изменяется на 0, что приводит к образованию хромосомы [100110001010]. Вероятность p_m мутации может эмулироваться, например, случайным выбором числа из интервала [0,1] для каждого гена и отбором для выполнения этой операции тех генов, для которых разыгранное число оказывается меньшим или равным значению p_m. Применяется и многоточечная мутация, которая является композицией нескольких одноточечных мутаций.

Инверсия может применяться с ненулевой и очень низкой вероятностью наряду с мутациями. Она изменяет не значения генов, а только порядок их следования в случайно выбранной хромосоме. В простейшем случаем в такой хромосоме случайным образом выбирается точка разрыва, и меняются местами начало и конец хромосомы. Допустимо применение многоточечной инверсии.

Кроссинговер, мутация и инверсия могут порождать новые хромосомы (решения), которые никогда не встречались в предыдущих популяциях. Их следует оценить значениями функции приспособленности. Пос-

ледующая популяция P_{t+1} может быть образована как целиком только из потомков хромосом предыдущей популяции P_t, полученных в результате применения к P_t генетических операторов, так и частичным сохранением наилучших особей из P_t. В первом случае новые хромосомы полностью заменяют старые, а во втором – только менее ценные из них.

Работа ГА прекращается при достижении текущей популяцией состояния адаптации, которое идентифицируется по стягиванию ядра популяции сначала в круг, а затем в точку. Алгоритм не гарантирует получение глобального экстремума (хотя это и возможно), а даёт некоторое «хорошее» решение за приемлемое время.

3.3.4. Классический генетический алгоритм

Работа ГА – итеративный процесс, продолжающийся вплоть до выполнения заданных условий остановки: близость полученных решений к заданному значению, прекращение роста средней «полезности» популяции, минимальная численность популяции, число итераций и др. Основной (классический) генетический алгоритм состоит из следующих шагов:

1) генерация случайным образом начальной популяции хромосом $P_0 = \{A^0_{r1},..., A^0_{rk},..., A^0_{rNo}\}$;

2) оценка приспособленности хромосом в популяции;

3) проверка условия остановки алгоритма;

4) селекция хромосом;

5) применение к начальной популяции последовательности генетических операторов, порождающей новую популяцию P_t;

б) формирование новой популяции P_{t+1};

7) выбор «наилучшей» хромосомы.

Блок-схема основного генетического алгоритма изображена на рис. 3.22. Рассмотрим конкретные этапы этого алгоритма с использованием дополнительных подробностей, представленных на рис. 3.23.

Генерация или формирование исходной популяции заключается в случайном выборе заданного количества хромосом (особей), закодированных двоичными последовательностями фиксированной длины.

Оценивание приспособленности хромосом в популяции состоит в расчёте функции приспособленности для каждой хромосомы этой популяции. Чем больше значение этой функции, тем выше «качество» хромосомы. Предполагается, что функция приспособленности всегда принимает неотрицательные значения и, кроме того, что для решения оптимизационной задачи требуется максимизировать эту функцию.

Проверка условия остановки алгоритма. Определение условия остановки генетического алгоритма зависит от его конкретного приме-

Рисунок 3.22. Схема классического генетического алгоритма

нения. В оптимизационных задачах, если известно максимальное (или минимальное) значение функции приспособленности, то остановка алгоритма может произойти после достижения ожидаемого оптимального значения, возможно, с заданной точностью. Остановка алгоритма также может произойти в случае, когда его выполнение не приводит к улучшению уже достигнутого значения. Алгоритм может быть остановлен по истечении определенного времени выполнения или после выполнения заданного количества итераций. Если условие остановки выполнено, то производится переход к завершающему этапу выбора «наилучшей» хромосомы. В противном случае на следующем шаге выполняется селекция.

Селекция хромосом заключается в выборе (по рассчитанным на втором этапе значениям функции приспособленности) тех хромосом, которые будут участвовать в создании потомков для следующей популяции, т.е. для очередного поколения. Такой выбор производится, согласно принципу естественного отбора, по которому наибольшие шансы на

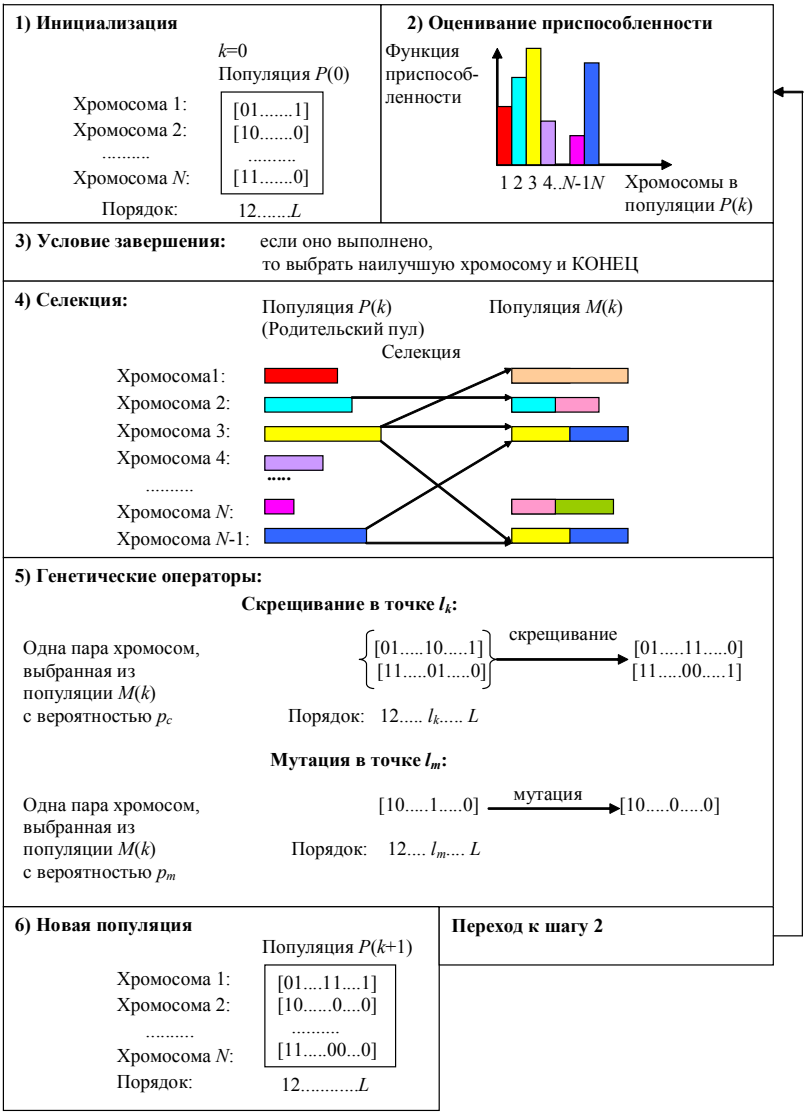


Рис. 3.23. Этапы генетического алгоритма

участие в создании новых особей имеют хромосомы с наибольшими значениями функции приспособленности.

Существуют различные методы селекции, наиболее популярные описаны в разд. 3.3.3. В результате процесса селекции создается родительская популяция, также называемая родительским пулом с числен-

ностью N, равной численности текущей популяции. *Применение генетических операторов* к хромосомам, отобранным с помощью селекции, приводит к формированию новой популяции потомков от созданной на предыдущем шаге родительской популяции.

В классическом генетическом алгоритме применяются два основных генетических оператора: оператор скрещивания и операторы мутации и инверсии. Однако следует отметить, что операторы мутации и инверсии играют явно второстепенную роль по сравнению с оператором скрещивания. Это означает, что скрещивание в классическом генетическом алгоритме производится практически всегда, а мутация и инверсия – достаточно редко. Вероятность скрещивания, как правило, достаточно велика (обычно $0{,}5 \le p_C \le 1$), тогда как вероятности мутации и инверсии устанавливаются весьма малыми (чаще всего $0 \le p_m \le 0{,}1$; $0 \le p_C \le 0{,}1$). Это следует из аналогии с миром живых организмов, где мутации происходят чрезвычайно редко.

В генетическом алгоритме мутация и инверсия хромосом могут выполняться на популяции родителей перед скрещиванием или на популяции потомков, образованных в результате скрещивания.

Формирование новой популяции. Хромосомы, полученные в результате применения генетических операторов к хромосомам временной родительской популяции, включаются в состав новой популяции. Она становится так называемой текущей популяцией для данной итерации генетического алгоритма. На каждой очередной итерации рассчитываются значения функции приспособленности для всех хромосом этой популяции. Затем проверяется условие остановки алгоритма и либо фиксируется результат в виде хромосомы с наибольшим значением функции приспособленности, либо осуществляется переход к следующему шагу генетического алгоритма, т.е. к селекции. В классическом генетическом алгоритме вся предшествующая популяция хромосом замещается новой популяцией потомков, имеющей ту же численность.

Выбор наилучшей хромосомы. Если условие остановки алгоритма выполнено, то следует вывести результат работы, т.е. представить искомое решение задачи. Лучшим решением считается хромосома A^*с наибольшим значением функции приспособленности

$$F(A^*) = \max_p F(A_k).$$

3.3.5. Применение генетического алгоритма для решения задачи обеспечения отказоустойчивости вычислительного кластера

Рассмотрим задачу обеспечения отказоустойчивости вычислительного кластера. Этот подход реализуется посредством отказоустойчивого

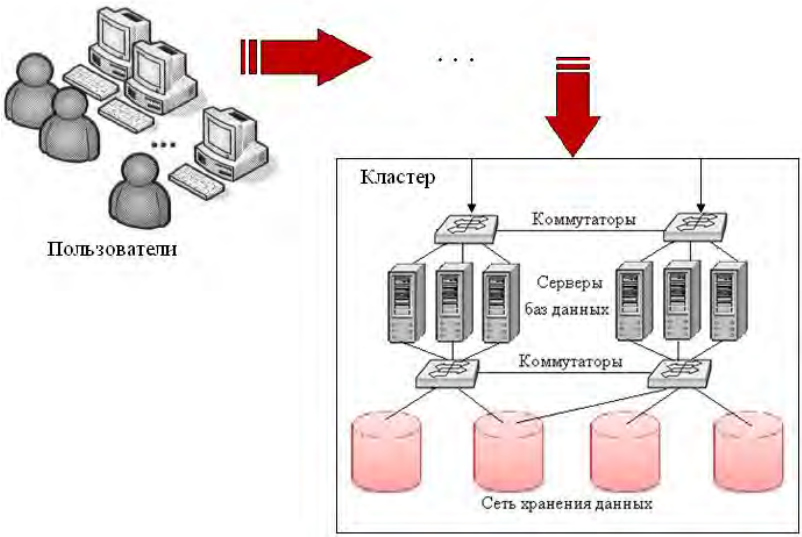


Рис. 3.24. Общая схема взаимодействия пользователей с кластером

размещения задач, т.е. такого статического распределения задач по серверам, при котором определенное число копий каждой задачи размещается на различных серверах.

Кластер рассматривается как совокупность *n* однотипных серверов, объединенных коммутационной системой (рис. 3.24). Каждый сервер имеет один процессор и память ограниченного объема. Кластер выполняет фиксированное задание Г, которое представляет собой известное множество задач $\Gamma=\{U_1, U_2, \ldots, U_L\}$ с заданными требованиями к порядку их выполнения и взаимосвязям.

Каждая задача U_j характеризуется следующими показателями:

b_j – вес задачи, т.е. величина, определяющая важность задачи U_j для системы (более важной задаче соответствует больший вес);

v_j – объем оперативной памяти, требуемый для хранения и выполнения задачи;

t_j – время выполнения задачи.

В процессе работы кластера возможны отказы серверов. Состояние кластера определяется как $s_v=\sigma_1,\ldots, \sigma_n$, где $\sigma_i=0$, если M_i – работоспособный сервер (*p*-сервер) и $\sigma_i=1$, если M_i – отказавший сервер (*o*-сервер); $s^0= 00\ldots0$ – начальное состояние кластера; все состояния $s_v \neq s^0$ называются искаженными. Известно начальное распределение задач по серверам при отсутствии их отказов (для состояния s^0), описываемое

матрицей $D^0 = |d^0_{ji}|$, где $d^0_{ji} = 1$, если задача U_j назначена для выполнения на сервере M_i и размещена в нем, $d^0_{ji} = 0$ в противном случае; каждая задача назначена на один и только один сервер. Задачи, назначенные на сервер M_i в соответствии с начальным размещением задач, назовем собственными задачами этого сервера.

Качество работы кластера оцениваем его функциональной мощностью E_v в состоянии s_v, которое определяется как сумма весов всех задач, назначаемых для выполнения в работоспособные серверы в состоянии s_v.

Требуемый уровень отказоустойчивости кластера задается множеством $S=\{s_v\}$ работоспособных состояний, которое определяется как множество всех таких состояний, число k отказавших серверов для которых не превосходит заданного значения d, очевидно $S = s^0 \cup S^\omega$, где $S^\omega=\{s_\omega\}$ – множество искаженных работоспособных состояний.

Требуется путем организации при отказах сервера надлежащего перераспределения задач между различными p-серверами обеспечить выполнение заданных требований к отказоустойчивости кластера и к другим его показателям. Для этого при проектировании кластера или при подготовке к реализации в нём определенного задания, необходимо найти оптимальные планы $D_\omega = |d^\omega_{ji}|$ распределения задач для каждого состояния $s_\omega \in S^\omega$ и результирующее отказоустойчивое размещение задач $Z = |z_{ji}|$ для всех серверов, где $z_{ji} = 1$, если задача U_j размещена на сервере M_i (т.е. её программный код загружен в память сервера M_i), $z_{ji} = 0$ в противном случае; z_{ji} определяется как дизъюнкция значений d^v_{ji} для всех состояний $s^v \in S$ (включая начальное).

После того как оптимальные планы D_ω найдены, каждый сервер заранее обеспечивается аппаратными и программными ресурсами, необходимыми для выполнения задач, которые могут назначаться ему в любом состоянии $s^v \in S$. При переходе кластера вследствие отказов некоторых серверов в любое искаженное состояние $s_\omega \in S^\omega$ и обнаружении этого факта во всех p-серверах начинается выполнение задач, назначенных им в соответствии с планом D_ω.

Применим ГА для решения задачи рационального статистического перераспределения задач.

Постановка задачи: Для каждого искаженного работоспособного состояния $s_\omega \in S^\omega$ найти такой план $D_\omega = |d^\omega_{ji}|$ распределения задач при котором достигается максимальное значение функциональной мощности системы в состоянии

$$E^f_\omega = \sum_{U_j \in \Omega^f_\omega} \sum_{M_i \in H_\omega} d^\omega_{ji} b_j \to \max, \qquad (3.3.2)$$

где Ω^f_ω – множество собственных задача отказавших серверов; H_ω – множество p-серверов для состояния s_ω, при ограничениях:

1) на суммарное время выполнения сервером M_i всех задач, назначенных ему в состоянии

$$\Delta T^{\omega}_{\Sigma i} = \sum_{U_j \in \Omega^f_\omega} \tau_j d^{\omega}_{ji} \le \Delta T^{*}_{\Sigma i},\ i = 1,...,g_\omega, \qquad (3.3.3)$$

где g_ω – число p-серверов в состоянии s_ω, $\Delta T^{*}_{\Sigma i} = T^{*}_{\Sigma i} - T^{0}_{\Sigma i}$ – ресурс времени для размещения на сервер M_i копий задач отказавших серверов,

$T^{*}_{\Sigma i}$ – заданное граничное значение суммарного времени выполнения всех задач, назначенных на p-сервер M_i; $T^{0}_{\Sigma i}$ – суммарное время выполнения всех собственных задач сервера M_i;

2) на объём памяти каждого p-сервера M_i, доступный для размещения копий задач, дополнительно назначаемых этому серверу для выполнения в искаженном состоянии s_ω:

$$\Delta V_{\omega i} = \sum_{U_j \in \Omega^f_\omega} d^{\omega}_{ji} \nu_j \le \Delta V^{*}_{\omega i}, \qquad (3.3.4)$$

$$\forall\ M_i \in H_\omega,\ \mathrm{i} = 1,..., g_\omega,$$

где $\Delta V_{\omega i} = \dfrac{(V^m_i - V^0_i}{N(S^{\omega r}_i)}$ – максимальный объём памяти сервера M_i; – объём памяти сервера M_i, занятый его собственными задачами; $N(S^{\omega r}_i)$ – число таких состояний $s_\omega \in S^\omega$, в которых данный сервер M_i не отказал и при учёте, что в каждом состоянии s_ω задача $U_j \in \Omega^f_\omega$ может либо назначена только в один p-сервер, либо отброшена, т.е.

$$\sum_{i=1}^{n} d^{\omega}_{ji} \le 1, \quad \forall U_j \in \Omega^f_\omega. \qquad (3.3.5)$$

Рассматриваем случай, когда в каждом состоянии s_ω подвергаются перераспределению, т.е. передаются для выполнения в какие-либо p-серверы, либо отбрасываются только собственные задачи отказавших серверов, а собственные задачи всех p-серверов продолжают выполняться на «своих» серверах.

Решение задачи. В качестве функции полезности $F(A_k)$ принята функциональная мощность системы в состоянии s_ω – сумма весов тех задач U_j, для которых $a_j \neq 0$ для данной хромосомы A_k. Решение задачи состоит в вычислении с помощью ГА плана размещения задач D_ω для каждого состояния $s_\omega \in S^\omega$. Затем на основе всех полученных планов D_ω и плана

начального размещения задач формируется план $Z = |z_{ji}|$ результирующего отказоустойчивого размещения задач для всех серверов, как указано выше. Последовательность выполнения ГА, формирующего план D_ω состоит из следующих шагов.

1. *Создание начальной популяции* P^0 состоит в выполнении операции инициализации для каждой особи из её N_0 хромосом, т.е. случайного распределения задач по *p*-ПМ некоторого данного состояния с одновременной проверкой заданных ограничений на объём памяти ПМ и на максимальное суммарное время выполнения задач на каждом сервере. N_0 задается пользователем. Алгоритм инициализации одной хромосомы:

– формирование случайного списка задач;

– выбор очередной задачи из списка, если он не пуст, в противном случае окончание работы алгоритма;

– попытка размещения выбранной задачи в *i*-й *p*-сервер данного состояния ($i = 1,\ldots,g$, g – число *p*-серверов), начиная с $i = 1$ при проверке ограничений (3.3.3) и (3.3.4): данная задача назначается для решения в *i*-й сервер, если ограничения (3.3.3) и (3.3.4) для него не нарушены, иначе – попытка разместить её в (i+1)-й сервер; процедура продолжается до наибольшего номера *p*-сервера;

– если данную задачу не удается разместить ни на один сервер – отбрасывание этой задачи и переход к шагу 2.

2. *Селекция (отбор) хромосом для скрещивания*. В данной реализации алгоритма выбран метод пропорциональной селекции хромосом непосредственно из предшествующей популяции (на первой итерации – из начальной). Из них случайным образом формируются пары различных хромосом, и над каждой из этих пар с заданной вероятностью выполняется операция кроссинговера.

3. *Выполнение операторов генетического алгоритма*. Над данной парой хромосом вначале с заданной вероятностью $P_{\text{Кр}}$ выполняется операция кроссинговера. Реализуется операция следующим образом. Если случайно сгенерированное число $r < P_{\text{Кр}}$, то над данной парой выполняется кроссинговер. Потомки проверяются на ограничения и сравниваются с родительскими хромосомами. Если потомок не удовлетворяет ограничениям или если полезность потомка меньше полезности родительской хромосомы, то потомок отбрасывается и ищется новая родительская пара для кроссинговера. Если $r > P_{\text{Кр}}$, то над данной парой кроссинговер не выполняется, а «несостоявшиеся родители» переходят в следующий родительский пул R для выполнения следующего генетического оператора – мутации. Если потомок удовлетворяет ограничениям и полезность потомка больше полезности родителя, то потомок переходит в промежуточный родительский пул R, а родители отбрасываются. В

данном варианте реализации ГА используется одноточечный кроссинговер.

Операция одноточечной мутации выполняется для каждого элемента (бита) каждой хромосомы из популяции R с заданной вероятностью $P_{\text{Мут}}$: попытка обмена задачи U_m, соответствующей выбранному элементу A_m, на какую-либо из отброшенных задач U_s, то есть назначенных согласно данной хромосоме в «фиктивный» сервер с номером «0». Данная хромосома переходит на следующий этап алгоритма, если после некоторой попытки такого обмена она удовлетворяет ограничениям (3.3.3) и (3.3.4), иначе ищется другая задача из множества отброшенных задач. В случае неудачи таких попыток для всех отброшенных задач хромосома переходит на следующий этап без изменения.

Пусть P^C_t - дочерняя популяция, полученная из популяции P_t в результате кроссинговера её особей и последующей мутации. В следующую (новую) популяцию P_{t+1} могут попасть как особи только дочерней популяции P^C_t, так и лучшие особи из дочерней P^C_t и родительской P_t популяций.

4. *Создание новой популяции* P_{t+1}: хромосомы, полученной в результате операций отбора, кроссинговера и мутации над хромосомами текущей популяции P_t, переходят в P_{t+1}.

5. *Вычисление качества популяции*. Данный этап необходим для ведения статистики и для определения значения критерия останова: после каждого шага ГА вычисляется полезность каждой хромосомы и полезность всей популяции, как сумма всех при k =1…N.

6. *Замена предыдущей популяции на новую и проверка критерия останова*. В реализации ГА для данной задачи использованы следующие критерии останова:

а) заданное максимальное количество популяций;

б) заданное максимальное значение полезности хромосомы – заданный процент от суммы весов всех задач начального множества.

Самая полезная хромосома в популяции, полученная перед остановом, является решением задачи. Данный алгоритм реализован на языке программирования C++.

Интерфейс программы содержит две формы ввода:

а) ввод данных, относящихся к основным параметрам кластера (рис. 3.25):

– число заданий (количество транзакций) – (1);

– суммарное число задач N, которое необходимо выполнить, чтобы обслужить заявки (задания) пользователей (количество ПМ[4]) – (2);

[4] ПМ – Программный модуль, так названы задачи в том смысле, что их решение в кластере подразумевает выполнение задач на программном уровне.

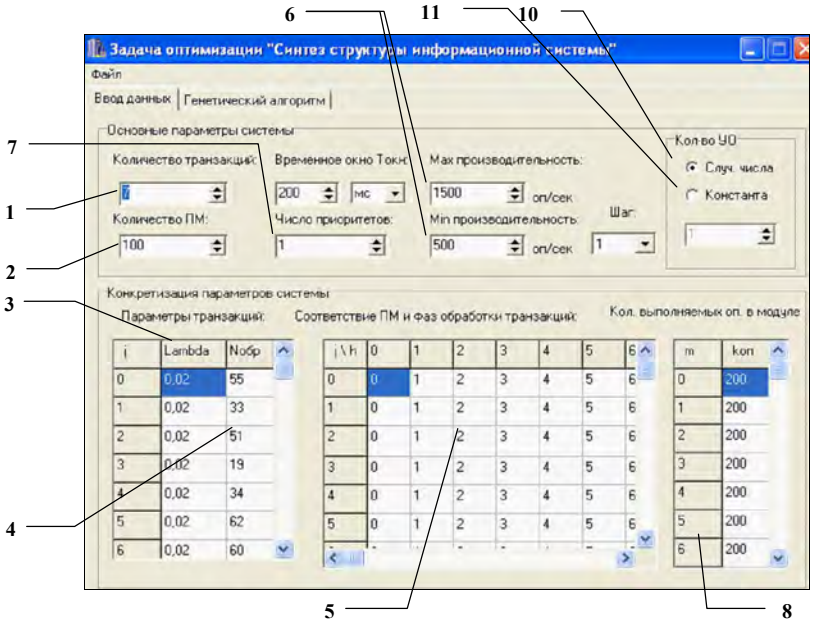


Рис. 3.25. Окно ввода исходных данных - основных параметров кластера

– интенсивность поступления заданий (lambda) – (3);

– распределение задач по заданиям ($N_{обр}$) – (4), разыгрывается случайно на интервале [1, *N*] – (5);

– производительность серверов (max производительность и min производительность) – (6), разыгрывается случайно;

– приоритеты выполняемых задач – (7);

– количество операций в каждой задаче – (8);

– количество серверов (устройств обработки – УО), которое можно задать (кнопка «Константа» – (9), тогда задачи будут оптимально распределены по существующим серверам или не задавать (кнопка «Случ. число» – (10), тогда алгоритм определит оптимальное число серверов в кластере для обеспечения должного выполнения заданий пользователей.

б) ввод данных, относящихся к параметрам генетического алгоритма (рис. 3.26):

– размер популяции – (1), задается пользователем;

– вероятность мутации – (2), задается пользователем;

– критерий останова, который реализован в двух вариантах:

а) порог – полезность популяции из поколения в поколение меняется незначительно – (3);

б) число шагов выполнения алгоритма – (4).

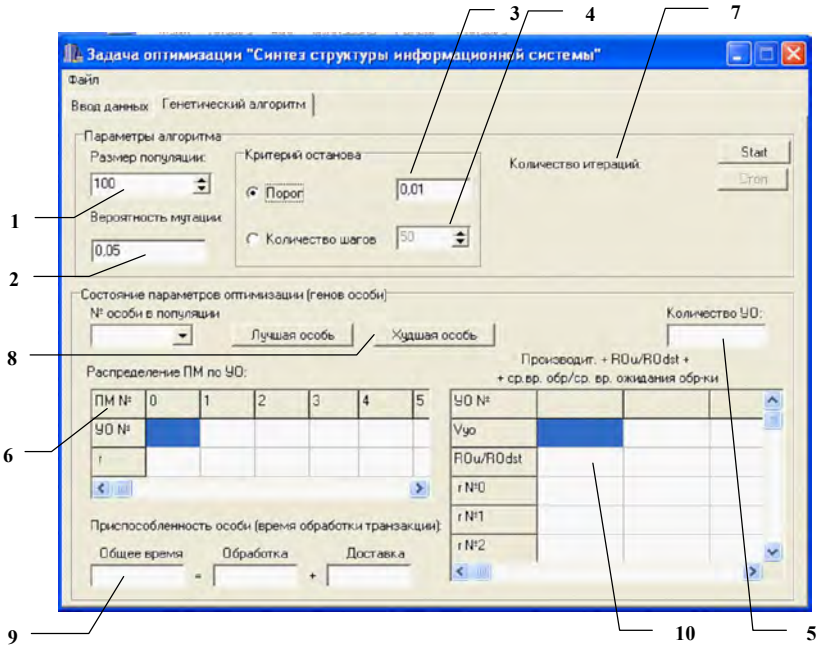


Рис. 3.26. Окно ввода исходных данных – параметров генетического алгоритма и вывода результатов работы генетического алгоритма

Результат работы генетического алгоритма выводится в это же интерфейсное окно на рис. 3.26:

– количество серверов в кластере, необходимое для выполнения заданий (транзакций) пользователя с обеспечением оптимального времени обслуживания заданий – (5).

– распределение задач по серверам; если задан приоритет выполнения задач *r*, то и распределение приоритетов – (6);

– количество итераций, которое понадобилось для получения лучшей особи – (7);

– можно посмотреть лучшую особь и худшую особь за всё время работы генетического алгоритма – (8);

– приспособленность особи: время обработки задания, которое складывается соответственно из обработки и доставки результата задания (9);

– распределение времен обработки и ожидания задач по серверам кластера – (10).

Оценим работу генетического алгоритма для двух вариантов: без приоритета выполнения задач на серверах кластера и с приоритетом.

Вариант 1: *без приоритетная обработка задач в кластере, количество серверов – случайное число*.

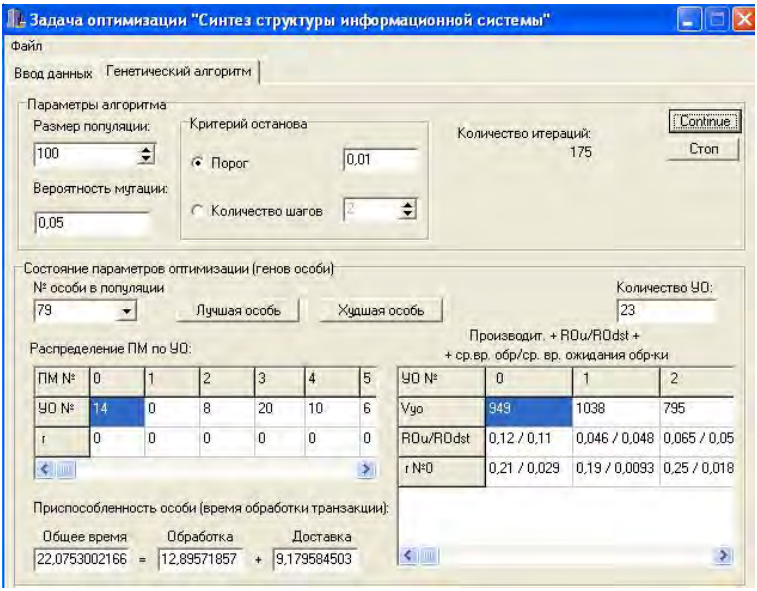


Рис. 3.27. Результаты работы генетического алгоритма для варианта 1

Исходные данные:
– количество заданий – 7;
– количество задач – 100;
– обработка задач – без приоритетная;
– количество операций в каждой задаче одинаково и равно 200;

Таблица 3.2

Распределение задач по серверам кластера для варианта 1

ПМ №	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
УО №	14	0	8	20	10	6	8	3	0	8	1	7	6	2	15	12
ПМ №	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
УО №	5	8	15	20	15	10	17	13	12	0	17	14	8	7	17	2
ПМ №	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
УО №	15	0	12	4	17	14	9	12	0	18	12	19	18	5	15	11
ПМ №	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
УО №	1	14	8	6	14	22	6	11	4	8	20	16	1	10	19	12
ПМ №	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
УО №	0	13	22	2	5	13	21	6	1	12	11	3	1	14	9	1
ПМ №	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
УО №	19	7	14	33	3	3	6	24	7	20	0	17	22	12	2	14
ПМ №	96	97	98	99												
УО №	10	6	3	14												

– количество серверов кластера – случайное число;

– производительность серверов варьируется от 500 оп/с до 1500 оп/с;

– матрица распределения интенсивностей и задач по заданиям приведена на рис. 3.25, обозначения (3) и (4).

Результат работы генетического алгоритма для варианта 1 приведен на рис. 3.27 и табл. 3.2.

Лучшая хромосома A^* за все время работы генетического алгоритма №79, максимальное значение $F(A_k)$ полезности хромосомы – это время обработки задания, которое составило 22,075 единицы времени, из них 12,896 – это обработка и 9,1796 – доставка. Решение, близкое к реальному, было найдено за 175 итераций. Для обслуживания заданий пользователей необходимо 23 сервера.

Полное распределение задач по серверам кластера, которое получается в результате работы генетического алгоритма, приведено в табл. 3.2.

Распределение задач по серверам позволяет определить производительность обработки заданий V_{yo}, среднее время обработки ROu и среднее время ожидания ROdst заданий (рис. 3.27).

Вариант 2: *без приоритетная обработка задач в кластере, количество серверов – задано*.

Исходные данные: те же, что и для варианта 1, но количество серверов кластера задано и равно 10 (рис. 3.28). Соответственно, задачи будут

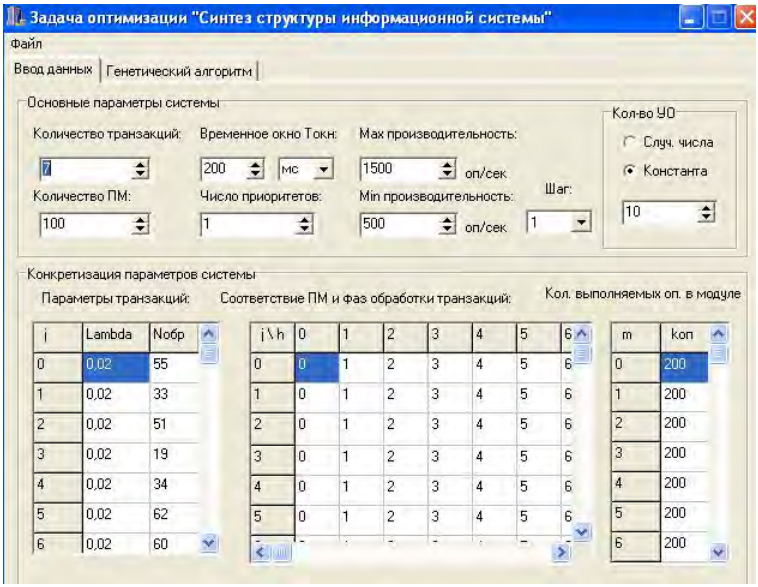


Рис. 3.28. Исходные данные для варианта 2

перераспределены с 23 серверов на 10 (будем считать, что 13 серверов в состоянии *о*-сервер). Оценим, как изменится при данных обстоятельствах время обработки заданий пользователей.

Результат работы генетического алгоритма для варианта 2 приведен на рис. 3.29 и табл.3.3. Как видно из рис. 3.29, общее время обработки

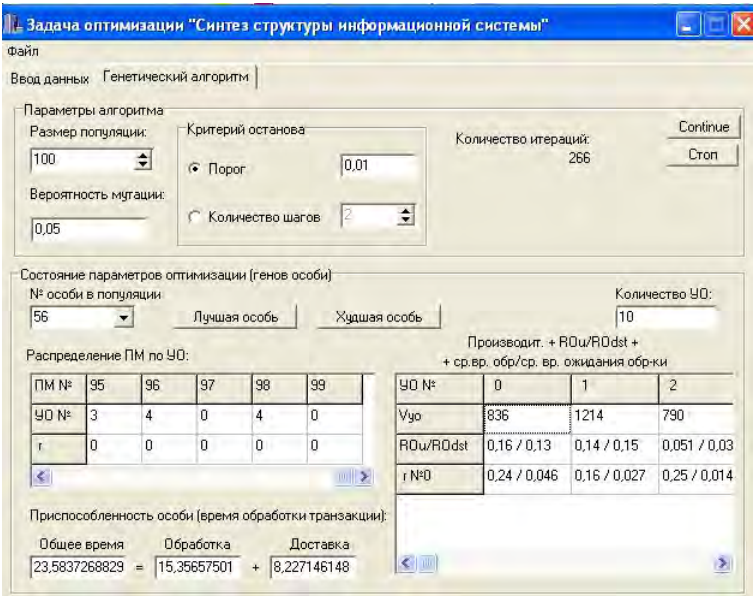


Рис. 3.29. Результаты работы генетического алгоритма для варианта 2

Таблица 3.3

Новое распределение задач по серверам кластера для варианта 2

ПМ №	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
УО №	10	6	11	13	11	9	12	14	9	8	14	7	6	12	11	15
ПМ №	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
УО №	21	8	13	23	29	17	17	26	31	9	27	28	16	7	30	12
ПМ №	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
УО №	15	16	32	28	34	21	27	24	29	28	24	19	35	11	31	23
ПМ №	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
УО №	23	29	17	13	31	40	15	25	14	18	38	33	5	19	23	27
ПМ №	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
УО №	12	19	22	28	15	26	21	16	23	25	19	7	18	28	9	22
ПМ №	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
УО №	19	7	14	33	7	9	6	24	7	20	5	17	22	12	2	3
ПМ №	96	97	98	99												
УО №	4	0	4	0												

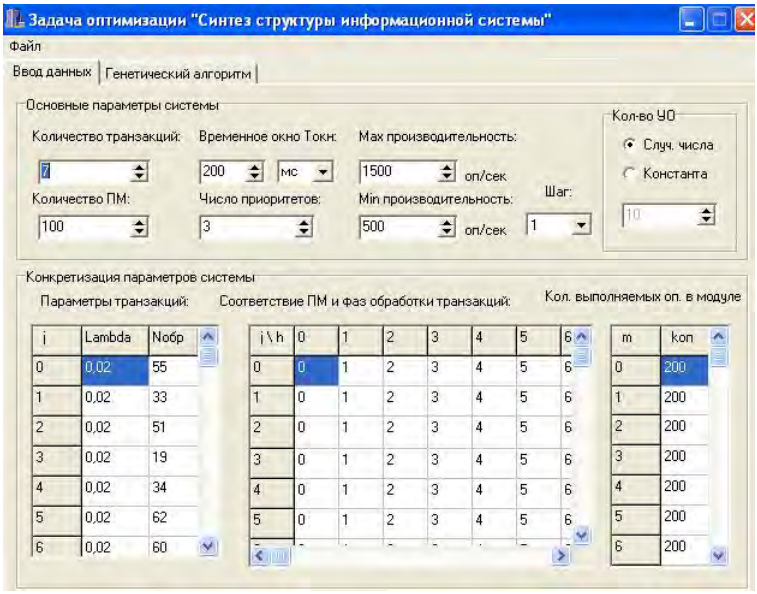


Рис. 3.30. Исходные данные для варианта 3

заданий увеличилось, поскольку увеличилась загруженность серверов кластера, находящихся в состоянии *р*-сервер.

Вариант 3: *приоритетная обработка задач в кластере, количество серверов – случайное число*.

Исходные данные:

- количество заданий, задач, матрицы распределений задач по транзакциям и количество операций для каждой задачи не изменилось; установлено 3 уровня приоритета обслуживания задач (рис. 3.30).

Результат работы генетического алгоритма для варианта 3 приведен на рис. 3.31 и табл. 3.4.

С введением приоритета время обработки изменилось незначительно.

Вариант 4: *приоритетная обработка задач в кластере, количество серверов – задано*.

Исходные данные: те же, что и для варианта 3. Количество серверов кластера задано и равно 7 (рис. 3.32). Соответственно, задачи будут перераспределены с 15 серверов на 7 (будем считать, что 8 серверов в состоянии *о*-сервер). Оценим, как изменится при данных обстоятельствах время обработки заданий пользователей.

Результат работы генетического алгоритма для варианта 4 приведен на рис. 3.33 и табл. 3.5.

В результате перераспределения задач с 15 серверов на меньшее –

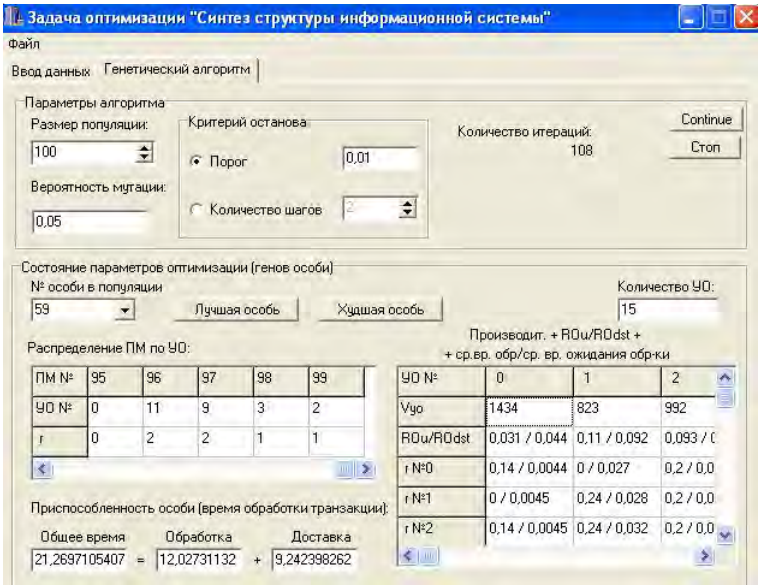


Рисунок 3.31. Результаты работы генетического алгоритма для варианта 3

Таблица 3.4

Распределение задач с приоритетами по серверам кластера для варианта 3

ПМ №	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
УО №	6	10	2	10	1	9	11	8	3	8	11	9	4	8	9	5
r	0	1	1	0	1	1	2	2	1	0	2	0	0	2	1	0
ПМ №	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
УО №	0	1	3	14	11	7	5	2	5	9	1	12	9	10	13	10
r	0	2	1	1	0	1	1	0	0	1	2	2	0	0	2	0
ПМ №	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
УО №	13	11	7	0	11	8	6	8	12	4	12	11	4	7	12	5
r	1	0	2	2	1	1	0	1	1	1	1	2	1	0	0	0
ПМ №	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
УО №	2	14	2	1	6	7	4	8	7	9	2	14	7	14	0	1
r	0	1	1	2	0	0	1	0	0	1	2	0	1	2	0	2
ПМ №	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
УО №	6	2	10	12	5	5	12	3	14	3	9	12	6	1	12	7
r	2	0	1	0	1	1	0	0	2	1	1	0	1	1	2	2
ПМ №	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
УО №	0	0	4	10	13	7	2	4	8	1	9	8	5	7	14	0
r	0	2	1	1	1	1	1	1	1	1	2	2	1	0	0	0
ПМ №	96	97	98	99												
УО №	11	9	3	2												
r	2	2	1	1												

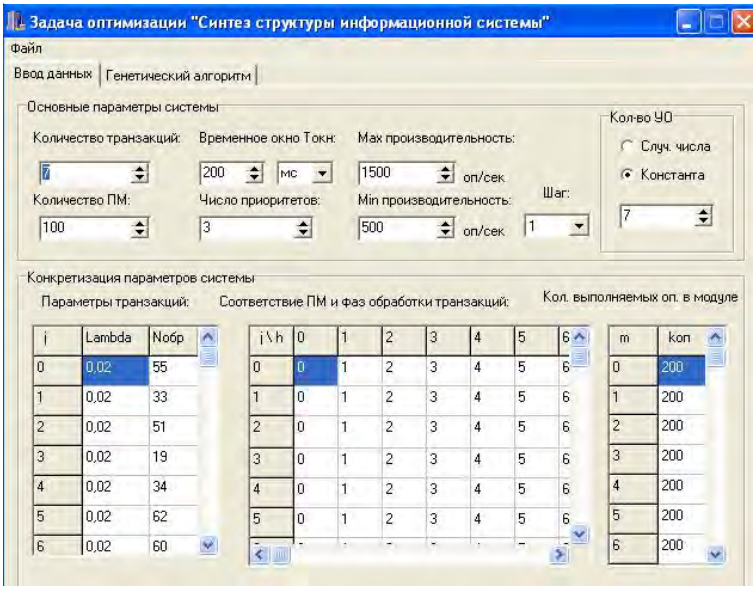


Рисунок 3.32. Исходные данные для варианта 4

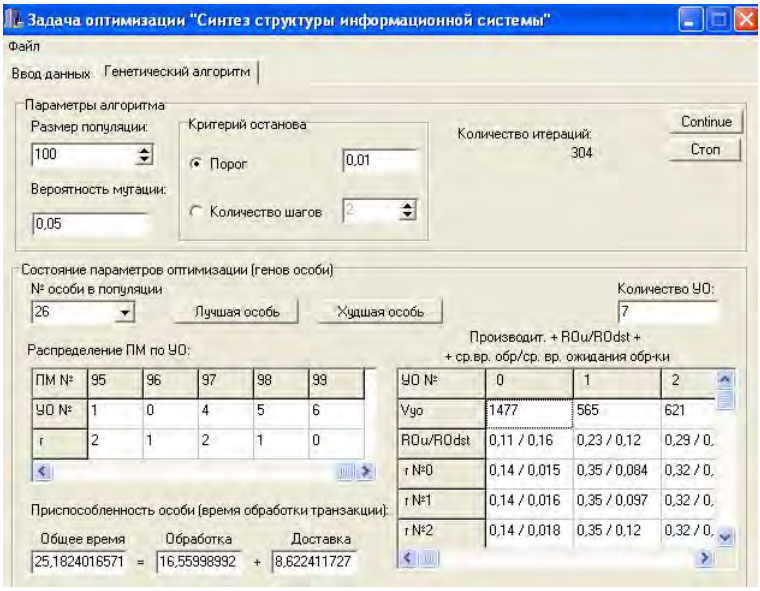


Рисунок 3.33. Результаты работы генетического алгоритма для варианта 4

Таблица 3.5

Перераспределение задач с приоритетами по серверам кластера для варианта 4

ПМ №	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
УО №	7	12	9	12	7	11	11	7	8	12	13	9	9	13	19	16
r	0	1	1	0	1	1	2	2	1	0	2	0	0	2	1	0
ПМ №	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
УО №	13	9	6	14	15	14	9	8	9	14	11	12	12	18	17	21
r	0	2	1	1	0	1	1	0	0	1	2	2	0	0	2	0
ПМ №	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
УО №	23	22	15	7	19	13	12	16	20	9	18	22	9	14	16	12
r	1	0	2	2	1	1	0	1	1	1	1	2	1	0	0	0
ПМ №	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
УО №	5	28	7	8	14	14	9	17	15	12	6	19	14	14	3	5
r	0	1	1	2	0	0	1	0	0	1	2	0	1	2	0	2
ПМ №	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
УО №	12	6	18	20	11	11	12	13	20	7	19	21	22	5	23	18
r	2	0	1	0	1	1	0	0	2	1	1	0	1	1	2	2
ПМ №	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
УО №	4	5	8	17	21	14	6	9	12	11	15	16	15	14	14	1
r	0	2	1	1	1	1	1	1	1	1	2	2	1	0	0	0
ПМ №	96	97	98	99												
УО №	0	4	5	6												
r	2	2	1	1												

7 серверов общее время обработки заданий пользователей увеличилось с 21,27 единиц времени на 25,18.

Вопросы и задания

1. Составьте схему поиска оптимального пути на графе в терминах генетического алгоритма.

2. Приведите последовательность действий при реализации метода отбора усечением.

3. Разработайте последовательность действий (алгоритм) реализации метода пропорциональной селекции.

4. Разработайте последовательность действий (алгоритм) реализации метода турнирной селекции.

5. От чего зависит число итераций в генетическом алгоритме? Обоснуйте ответ.

6. В чём состоят недостатки генетического алгоритма? Как эти недостатки сказываются на результате работы генетического алгоритма?

7. Каково назначение операции «мутация» в генетическом алгоритме, какое значение вероятности она принимает и почему?

8. Каково назначение операции «кроссинговер» в генетическом алгоритме, какое значение вероятности она принимает и почему?

9. Каково назначение операции «инверсия» в генетическом алгоритме, какое значение вероятности она принимает и почему?

10. Каким образом оценивается функция приспособленности и какую роль она играет в генетическом алгоритме?

11. Составьте схему поиска оптимального количества серверов вычислительного кластера в терминах генетического алгоритма. Количество серверов в кластере должно быть достаточным, чтобы заявки пользователей обрабатывались за допустимое время.

12. Приведите схему классического генетического алгоритма. Объясните, как он работает.

13. Объясните суть задачи обеспечения отказоустойчивости вычислительного кластера, решаемой с применением генетического алгоритма.

ЗАКЛЮЧЕНИЕ

Моделирование является основным методом исследований во всех областях знаний и научно обоснованным методом оценок характеристик сложных систем, используемым для принятия решений в различных сферах инженерной деятельности. Существующие и проектируемые системы можно эффективно исследовать с помощью математических моделей (аналитических и имитационных), реализуемых на современных компьютерах, которые в этом случае выступают в качестве инструмента экспериментатора с моделью системы.

Модели сложных систем, какими являются сети телекоммуникаций, обычно имеют комплексный вид, используют в своём составе сразу несколько представлений. Сложная система - это такая, которую нельзя понять, рассматривая её только с одной точки зрения и отобразив эту точку зрения в виде наглядной модели (рисунка на бумаге) или в виде математической модели, взаимосвязанной с наглядной моделью. Чтобы осмыслить многие аспекты, необходимые для решения задачи проектирования сложной системы, её разработчики должны рассматривать эту систему с нескольких точек зрения. Практика показала, что разработчикам иногда приходится отображать отдельную точку зрения на сложную систему не одной, а несколькими моделями. Различные точки зрения на систему и все отображающие эти точки зрения модели объединяются в проектной документации и в умах разработчиков в единое (интегральное) представление о системе.

Модели могут принимать различную форму в зависимости от способа мышления исследователя, его взгляда на мир, используемой алгебры. Использование различных математических аппаратов впоследствии приводит к различным возможностям в решении задач.

Каждый подход к моделированию имеет свои достоинства и недостатки. Разные модели имеют разные возможности (мощность) для решения задач, разные потребности в вычислительных ресурсах. Один и тот же объект может быть описан различными способами. Инженер должен грамотно применять то или иное представление, исходя из текущих условий и стоящей перед ним проблемы.

ЛИТЕРАТУРА

1. Задорожный В.Н. Имитационное и статистическое моделирование: Учебное пособие. – Омск: Изд-во ОмГТУ, 2007. –120 с.
2. Карпов Ю. Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5. – СПб. : БХВ-Петербург, 2005. – 400 с.
3. Кутузов О.И., Задорожный В. Н., Олзоева С. И. Имитационное моделирование сетей массового обслуживания: Учебное пособие. – Улан-Удэ: Изд-во ВСГТУ, 2001. – 228 с.
4. Поляк Ю. Г. Вероятностное моделирование на ЭВМ. – М: Советское радио, 1971. – 400 с.
5. Рыжиков Ю. И. Имитационное моделирование. Теория и технологии. – СПб. : КОРОНА принт, 2004. – 384 с.
6. Советов Б. Я., Яковлев С.А. Моделирование систем: Учебник для вузов – М.: Высшая школа, 2001. – 343 с.
7. Соболь И. М. Численные методы Монте-Карло. – М., 1973. – 212 с.

СОДЕРЖАНИЕ

CONTENTS

Учебное издание

Олег Иванович Кутузов,
Татьяна Михайловна Татарникова

МОДЕЛИРОВАНИЕ СИСТЕМ И СЕТЕЙ ТЕЛЕКОММУНИКАЦИЙ

Учебное пособие

Редактор О.С. Крайнова
Компьютерная вёрстка К.П. Ерёмин

ЛР № 020309 от 30.12.96

Подписано в печать 14.09.2012. Формат 60х90 1/16. Гарнитура “Таймс”
Печать цифровая. Усл. печ. л. 8,5. Тираж 300 экз. Заказ № 119
РГГМУ, 195196, Санкт-Петербург, Малоохтинский пр. 98.
Отпечатано в ЦОП РГГМУ