1985
Комбинаторика, сложность и случайность

Ричард М. Карп

Ричард М. Карп из Калифорнийского университета, Беркли, получил в 1985 г. премию АСМ им. А. М. Тьюринга за свои фундаментальные достижения в теории сложности. Эта премия, являющаяся высшей наградой АСМ за исследования по информатике, была вручена на ежегодной конференции Ассоциации в Денвере, в октябре.

К 1972 г. Карп приобрел репутацию одного из ведущих теоретиков информатики в основном из-за своей новаторской статьи «Сводимость комбинаторных задач» (в книге Complexity of Computer Computations (Symposium Proceedings), Plenum, New Jork, 1972). Продолжая более раннюю работу Стивена Кука, он применил понятие сводимости в полиномиальное время и показал, что если только P не совпадает с NP, большинство классических задач комбинаторной оптимизации NP-полны и, следовательно, практически неразрешимы. Это изменило подход специалистов по информатике к практическим задачам, таким, как выбор маршрута (включая знаменитую задачу о коммивояжере), упаковка, покрытие, паросочетания, разбиение и составление расписаний, и усилило внимание к приближенным методам решения этих трудных задач. Потом в своей работе Карп впервые применил вероятностный анализ, чтобы обосновать эффективность таких приближенных методов.

Р. Карп — профессор трех факультетов в Бэркли: электротехники и информатики, математики, а также инженерного дела и исследования операций. В этом году он сопредседатель одногодичной исследовательской программы по вычислительной сложности в Математическом институте (Mathematical Sciences Research Institute), финансируемой Национальным научным фондом.

Родился в Бостоне, степень доктора философии по прикладной математике получил в Гарвардском университете в 1959 г. Он занимался девять лет информатикой в исследовательской лаборатории IBM в Йорктаун Хайтс, Нью-Йорк, и преподавал в Мичиганском, Колумбийском и Нью-Йоркском университетах и Политехническом институте в Бруклине. С 1968 г. он преподает в Беркли и получил профессорство Миллера в Беркли:

в 1980—1981 гг. В 1980 г. был избран в Национальную академию наук.

Лауреат Тьюринговской премии за 1985 г. представил свой обзор развития исследований в области, которая сейчас стала называться теоретической информатикой.

* * *

Эта лекция посвящена памяти моего отца Абрахама Луиса Карпа.

Для меня честь и удовольствие быть лауреатом Тьюринговской премии этого года. Какое бы удовлетворение ни доставило мне такое признание, я нахожу, что мое величайшее удовлетворение как исследователя связано с выполнением самого исследования и с дружескими связями, которые я при этом приобрел. Я хотел бы описать вам все 25 лет исследований в области комбинаторных алгоритмов и вычислительной сложности и рассказать о некоторых понятиях, которые мне кажутся важными, и о некоторых людях, которые меня вдохновляли и влияли на меня.

НАЧАЛО

Я пришел в информатику довольно случайно. По окончании в 1955 г. Гарвардского колледжа с дипломом по математике я стал думать, что же делать дальше. Зарабатывать на жизнь было малопривлекательным, и очевидный выбор пал на аспирантуру. Одна из возможностей была продолжать занятия математикой, но тогда все увлекались абстрактностью и общностью, а конкретная и прикладная математика, вдохновлявшая меня, казалось, была не в моде.

Итак, почти не имея другого выбора, я приступил к программе подготовки диссертации доктора философии в Гарвардской вычислительной лаборатории. Большинства предметов, которые теперь стали непременными составляющими курсов информатики, тогда и в мыслях ни у кого не было, и я прослушал эклектичный набор дисциплин: теория переключательных схем, численный анализ, прикладная математика, вероятность и статистика, исследование операций, электроника и математическая лингвистика. Хотя программа оставляла желать лучшего в смысле глубины и согласованности, там была особая атмосфера: мы знали, что присутствием при рождении новой науки, изучающей компьютеры и их применения. Я обнаружил, что нахожу красоту и элегантность в структуре алгоритмов и люблю дискретную математику, составлявшую основу для изучения компьютеров и вычислений. Короче, я попал более или менее случайно в научную область, которая мне очень понравилась.

ЛЕГКИЕ И ТРУДНЫЕ КОМБИНАТОРНЫЕ ЗАДАЧИ

Уже с той ранней поры я приобрел особый интерес к комбинаторным задачам поиска, — задачам, которые можно уподобить головоломкам, где нужно собрать определенным образом отдельные части в одну общую структуру. Такие задачи включают поиск в конечном, но очень большом структурированном множестве возможных решений, образов или конфигураций с целью найти одно решение, удовлетворяющее заданному множеству условий. Вот несколько примеров таких задач: размещение и соединение компонент интегральных схем, составление расписания игр национальной футбольной лиги и управление движением школьных автобусов.

Во всякой из этих комбинаторных головоломок таится возможность комбинаторного взрыва. По причине огромного, неудержимо растущего числа возможностей, возникающих при поиске, можно столкнуться с большим объемом вычислений, если не применить какую-нибудь тонкость при поиске в пространстве возможных решений. Я хотел бы начать техническую часть своей беседы с рассказа о первых моих встречах с комбинаторными взрывами.

Первое поражение при столкновении с этим явлением я потерпел вскоре после того, как поступил в Исследовательский центр IBM на Йорктаун Хайтс в 1959 г. Я был назначен в группу, возглавляемую Дж. Ротом, известным алгебраическим топологом, который внес заметный вклад в теорию переключательных схем. Задача нашей группы состояла в создании компьютерной программы для автоматического синтеза переключательных схем. На входе программы было множество булевых формул, описывающих, как выходы схемы зависят от ее входов; программа должна была строить схему, состоящую из минимального числа логических элементов. На рис. 1 показана схема для функции голосования от трех переменных, выход принимает значение единица, когда по крайней мере 2 из 3 переменных x, y и z принимают значение единица.

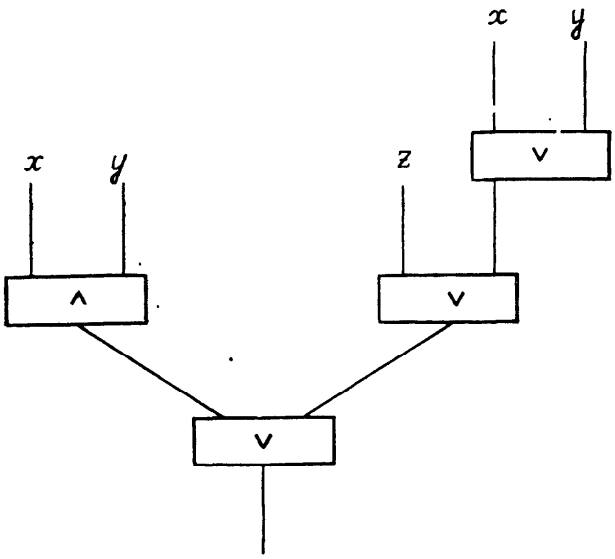


Рис. 1. Схема для функции голосования.

Программа, которую мы разработали, содержала много элегантных упрощений и ухищрений, но ее основной механизм состоял в

перечислении возможных схем в порядке возрастания их стоимости (сложности). Число схем, которые эта программа должна была просматривать, страшно росло по мере роста числа входных переменных, и, как следствие, мы не смогли продвинуться далее решения игрушечных задач. Сегодня наш оптимизм в самой попытке применить перечислительный подход может показаться крайне наивным, но мы были не единственными, кто попал в эту ловушку, многие работы по автоматическому доказательству теорем за последние два десятилетия начинались с прилива энтузиазма от успешного решения игрушечных задач, за которым следовало разочарование, как только становилась очевидной вся серьезность явления комбинаторного взрыва.

Примерно в то же самое время я начал работать над задачей о коммивояжере с Майклом Хелдом из IBM. Эта задача получила свое название от ситуации коммивояжера, который хочет посетить все города на своей территории, начиная и кончая путь в родном городе и минимизируя при этом общую стоимость путешествия. В специальном случае, когда города — точки на плоскости и стоимость путешествия равна евклидову расстоянию, задача состоит просто в нахождении многоугольника с минимальным периметром, проходящим через все города (рис. 2). Несколькими годами ранее Джордж Данциг, Раймонд Фалкерсон и Селмер Джонсон из Рэнд корпорейшн, используя комбинацию ручных и автоматических вычислений, преуспели в решении задачи для 49 городов, и мы надеялись побить этот рекорд.

Несмотря на невинный вид, задача о коммивояжере потен-

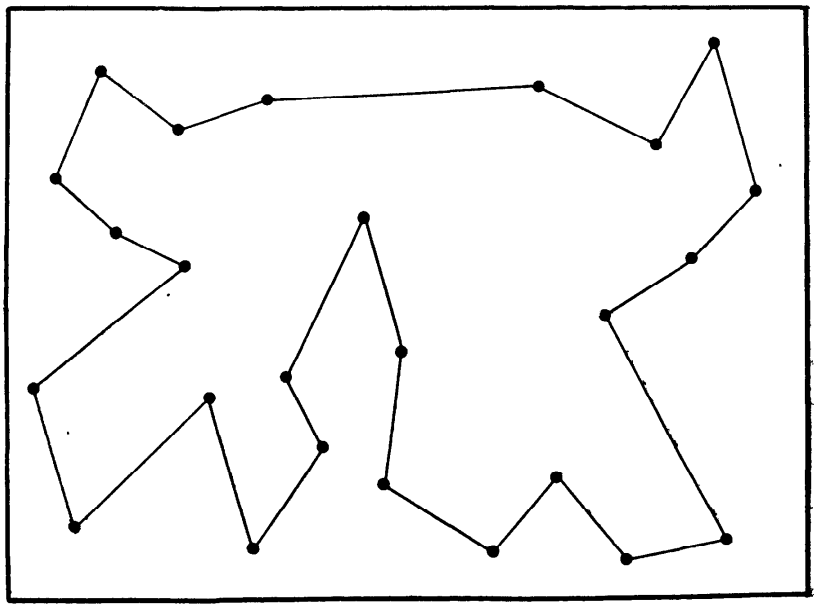


Рис. 2. Маршрут коммивояжера.

циально способна привести к комбинаторному взрыву, так как число возможных маршрутов на плоскости через n городов равно $(n-1)!/2$, очень быстро растущей функции n. Например, когда число городов равно всего лишь 20, время, требуемое для бесхитростного перечисления всех возможных маршрутов со скоростью миллион маршрутов в секунду, составит больше тысячи лет.

Хелд и я испробовали ряд подходов к задаче о коммивояжере. Мы начали с переоткрытия сокращения, основанного на динамическом программировании, на которое указал Ричард Беллман. Метод динамического программирования уменьшает время поиска до $n^2 2^n$, но эта функция также быстро растет, и метод ограничен практически задачами не более чем с 16 городами. На некоторое время мы оставили идею решить задачу точно и экспериментировали с методами локального поиска, которые приводят к достаточно хорошим, но не оптимальным маршрутам. При этих методах начинают с произвольного маршрута, снова и снова отыскивая локальные изменения, улучшающие его. Процесс продолжается, покуда не найден маршрут, который локально не может быть улучшен. Наши методы локальных изменений были довольно неуклюжими, Чень Линь и Брайан Керниган из лаборатории Бэлл позднее нашли гораздо лучшие. Такие быстрые и грубые методы часто весьма полезны на практике, если не требуется найти строго оптимальное решение, но никогда нельзя гарантировать, насколько хорошо они будут работать.

Затем мы начали исследовать методы ветвей и границ. Такие методы по сути перечислительные, но они выигрывают в эффективности в результате отсечения больших областей пространства возможных решений. Это делается путем вычисления нижней оценки стоимости каждого маршрута, включающего некоторые связи и исключающего некоторые другие; если нижняя оценка достаточно велика, то отсюда следует, что такой маршрут не может быть оптимальным. После длинной серии безуспешных экспериментов мы с Хелдом случайно обнаружили сильный метод получения нижних оценок. Эта техника ограничений позволила нам сильно сокращать пространство поиска, так что мы сумели решать задачи даже с 65 городами. Я не думаю, что какой-нибудь из моих теоретических результатов потряс меня столь же сильно, как вид чисел, появляющихся из компьютера ночью, когда Хелд и я впервые испытывали наш метод ветвей и границ. Позднее мы обнаружили, что наш метод — не что иное, как вариант старой техники, именуемой релаксациями Лагранжа, которая теперь рутинно используется для построения нижних оценок в методе ветвей и границ.

Короткое время наша программа была мировым чемпионом

по решению задачи о коммивояжере, но в наши дни существуют гораздо более впечатляющие программы. Они базируются на технике, называемой полиэдральной комбинаторикой, которая пытается свести частные случаи задачи о коммивояжере к очень большим задачам линейного программирования. Такие методы могут решать задачи более чем с 300 городами, но такой подход не полностью исключает комбинаторные взрывы, так как время, требуемое для решения задачи, продолжает расти экспоненциально как функция числа городов.

Задача о коммивояжере остается очаровательной загадкой. Недавно опубликована книга объемом более чем 400 страниц, покрывающая большинство из того, что известно об этой непростой задаче. Позднее мы обсудим теорию NP-полноты, которая дает необходимое обоснование того, что задача о коммивояжере по самой своей природе практически нерешаема, так что независимо от конструкции алгоритмов нельзя полностью исключить возможности комбинаторных взрывов, вспыхивающих внутри этой задачи.

В начале 1960-х годов лаборатория IBM на Йорктаун Хайтс имела превосходную группу специалистов по комбинаторике, и под их руководством я научился важным методам решения некоторых комбинаторных задач такого рода, позволяющим избежать комбинаторных взрывов. Например, я познакомился со знаменитым простым алгоритмом Данцига для линейного программирования. Задача линейного программирования состоит в нахождении точки на многограннике в многомерном пространстве, ближайшей к данной внешней гиперплоскости (многогранник — это обобщение многоугольника в двумерном пространстве или обычного многогранного тела в трехмерном пространстве, а гиперплоскость — это обобщение прямой на плоскости или плоскости в трехном пространстве). Ближайшая к гиперплоскости точка — всегда угловая точка, или вершина многогранника (рис. 3). На практике симплекс-метод очень быстро позволяет найти нужную вершину.

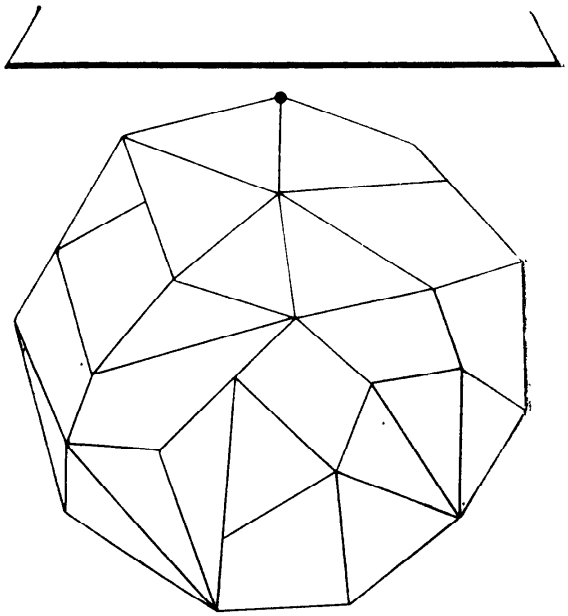


Рис. 3. Задача линейного программирования.

Я также изучал прекрасную теорию потоков в сетях Лестера Форда и Фалкерсона. Эта теория описывает, с какой скоростью некую субстанцию, например нефть, газ, электри-

чество или биты информации, можно прогонять через сеть, в которой каждое ребро имеет ограниченную пропускную способность. Многие комбинаторные задачи, которые на первый взгляд кажутся не имеющими никакого отношения к потокам, протекающим через сети, могут быть переформулированы как задачи в потоках в сетях, и эта теория способствует элегантному и эффективному решению таких задач без использования других арифметических операций, кроме сложения и вычитания.

Позвольте мне проиллюстрировать эту красивую теорию наброском так называемого венгерского алгоритма для решения задачи комбинаторной оптимизации, известной как задача марьяжа. Эта задача относится к обществу из n мужчин и n женщин. Задача состоит в разбиении на пары мужчин и женщин способом с наименьшими затратами, где стоимость каждого сочетания задана. Эти стоимости заданы матрицей $n \times n$, в которой каждая строка соответствует одному из мужчин и каждый столбец — одной из женщин. Вообще каждое паросочетание n мужчин с n женщинами соответствует выбору n элементов из матрицы, никакие два из которых не принадлежат одному столбцу и одной строке, стоимость паросочетания есть сумма n выбранных элементов. Число возможных паросочетаний есть $n!$; эта функция растет настолько быстро, что простой перебор будет малоэффективен. На рис. 4 (а) приведен пример 3×3-матрицы, в котором мы видим, что стоимость паросочетания мужчины номер 3 с женщиной номер 2 равна 9, элементу третьей строки и второго столбца данной матрицы. Ключевое наблюдение, лежащее в основе венгерского алгоритма, состоит в том, что задача не изменяется, если из всех элементов одной строки матрицы вычесть одну и ту же константу. Используя эту свободу модификации матрицы, алгоритм пытается построить матрицу, в которой все элементы неотрицательны, так что каждое полное паросочетание имеет неотрицательную общую стоимость, и в которой существует полное паросочетание со всеми нулевыми элементами. Такое паросочетание, очевидно, оптимально для построенной матрицы стоимости, и оно оптимально также для исходной матрицы. В нашем примере 3×3-матрицы алгоритм начинает с вычитания наименьшего элемента каждой строки из всех остальных элементов этой строки, после чего образуется матрица, в которой каждая строка со-

$$\begin{bmatrix} 3 & 4 & 2 \\ 8 & 9 & 1 \\ 7 & 9 & 5 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 0 \\ 7 & 8 & 0 \\ 2 & 4 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 \\ 6 & 6 & 0 \\ 1 & 2 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 \\ 5 & 5 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

(a) (b) (c) (d)

Рис. 4. Пример задачи паросочетания.

держит по крайней мере один нуль (рис. 4 (b)). Затем, чтобы получить нуль в каждом столбце, алгоритм вычитает из всех элементов каждого столбца, который еще нуля не содержит, наименьший элемент этого столбца (рис. 4 (c)). В этом примере все нули полученной матрицы лежат в первой строке или третьем столбце, так как полное паросочетание содержит точно один элемент из каждого столбца и каждой строки; этого недостаточно для нахождения полного паросочетания, состоящего целиком из нулевых элементов. Чтобы построить такое паросочетание, необходимо получить нуль в нижней левой части матрицы. В данном случае алгоритм строит такой нуль вычитанием 1 из первого и второго столбцов и добавлением единицы к первой строке (рис. 4 (d)). В полученной неотрицательной матрице три обведенных кружками элемента дают полное паросочетание нулевой стоимости, и это паросочетание, следовательно, оптимально как в итоговой, так и в исходной матрицах.

Этот алгоритм гораздо тоньше и эффективнее, чем простой перебор. Время, затрачиваемое им на решение проблемы марьяжа, растет только как третья степень (куб) n, числа строк и столбцов матрицы, и как следствие, можно решать примеры с тысячами строк и столбцов.

Поколение исследователей, основавших теорию линейного программирования и теорию потоков в сетях, заняло прагматическую позицию по отношению к задачам сложности вычислений: алгоритм рассматривался как эффективный, если он быстро работал на практике, и не так уж важно было доказывать, что он быстр во всех возможных случаях. В 1967 г. я заметил, что стандартный алгоритм для решения некоторых задач о потоках в сетях имеет теоретический изъян, из-за чего он очень медленно работает на некоторых специально подобранных примерах. Я нашел, что нетрудно исправить этот изъян, и рассказал об этом результате на семинаре по комбинаторике в Принстоне. Принстонцы сообщили мне, что Джек Эдмондс из Национального бюро стандартов представлял очень похожие результаты на том же семинаре на предыдущей неделе.

В результате такого совпадения мы с Эдмондсом начали работать вместе над теоретической эффективностью алгоритмов для потоков в сетях и затем написали совместную статью. Но главный результат нашего сотрудничества состоял в укреплении моей приверженности некоторым идеям о вычислительной сложности, к которым я уже ощупью двигался и которым предстояло оказать сильное влияние на дальнейшее направление моих исследований. Эдмондс был искусным практиком и умело использовал идеи, относящиеся к линейному программированию, при разработке прекрасных алгоритмов для ряда комбинаторных задач. Но вдобавок к его искусности в построении алго-

ритмов он определил своих современников в другом важном отношении. Он разработал ясное и точное понимание того, что означает для произвольного алгоритма «быть эффективным». Его статьи сделали общепринятой ту точку зрения, что алгоритм нужно рассматривать как «хороший», если время его выполнения ограничено полиномиальной функцией от размера входного слова (числа входов), а не, скажем, экспоненциальной функцией. Например, согласно эдмондсовской концепции, венгерский алгоритм для задачи марьяжа — хороший алгоритм, потому что время его работы растет как куб размера входа. Но, насколько мы знаем, может не существовать хорошего алгоритма для задачи о коммивояжере, потому что все алгоритмы, которые мы испытывали, имели экспоненциальную зависимость времени выполнения от размера задачи. Определение Эдмондса дало ясный критерий разграничения легких и трудных комбинаторных задач и впервые открыло, по крайней мере в моем понимании, возможность того, что мы однажды можем прийти к теореме, которая докажет или опровергнет предположение, что задача о коммивояжере по своей природе практически неразрешима.

ПУТЬ К NP-ПОЛНОТЕ

Параллельно с развитием событий в области комбинаторных алгоритмов набирал силу в течение 60-х годов другой важный поток исследований — теория вычислительной сложности. Основы этого предмета были заложены в 30-е годы группой логиков, включающей Аллана Тьюринга, которых заинтересовало существование или несуществование автоматических процедур для установления того, истинны или ложны математические утверждения.

Тьюринг и другие первооткрыватели теории вычислимости были первыми, кто доказал, что некоторые корректно определенные математические задачи неразрешимы, т. е. что в принципе не существует алгоритма, способного к решению всех частных случаев таких задач. Первым примером такой задачи была задача остановки, по существу — вопрос об отладке компьютерных программ. Вход задачи остановки есть компьютерная программа вместе с ее входными данными; задача состоит в том, чтобы решить, будет ли исполнение программы в конце концов завершено. Как может не быть алгоритма для такой корректно определенной задачи? Сложности возникают из-за возможности неограниченного поиска. Очевидное решение состоит в выполнении программы до остановки. Но в какой момент становится логичным прервать исполнение программы, решив, что она никогда не остановится сама? Похоже, что

нет никакого способа установить предел объема необходимых поисков. Используя так называемый диагональный метод, Тьюринг построил доказательство того, что не существует алгоритма, который может успешно рассмотреть все частные случаи проблемы остановки.

Годы шли, и неразрешимые задачи были найдены почти во всех областях математики. Вот пример из теории чисел — задача решения диофантовых уравнений: для данного полиномиального уравнения, скажем

$$4xy^2+2xy^2z^3-11x^3y^2z^2=-1164,$$

определить, существует ли его решение в целых числах. Задача нахождения общей разрешающей процедуры для решения таких диофантовых уравнений была впервые поставлена Давидом Гильбертом в 1990 г. и стала известна как 10-я проблема Гильберта. Проблема оставалась открытой до 1971 г., когда было доказано, что такой разрешающей процедуры существовать не может.

Один из фундаментальных инструментов, используемых в проведении границы между разрешимыми и неразрешимыми задачами, — понятие сводимости, которое было впервые выдвинуто на первый план благодаря работе логика Эмиля Поста. Задача A сводима к задаче B, если, исходя из процедуры, способной решить задачу B, можно построить алгоритм для решения задачи A. Например, большое значение имел следующий результат: проблема остановки сводима к 10-й проблеме Гильберта. Отсюда следует, что 10-я проблема Гильберта должна быть неразрешимой, так как в противном случае мы могли бы использовать это сведение, чтобы построить алгоритм для проблемы остановки, которая, как известно, неразрешима. Понятие сводимости вновь возникнет, когда мы будем обсуждать P-полноту и проблему P=NP.

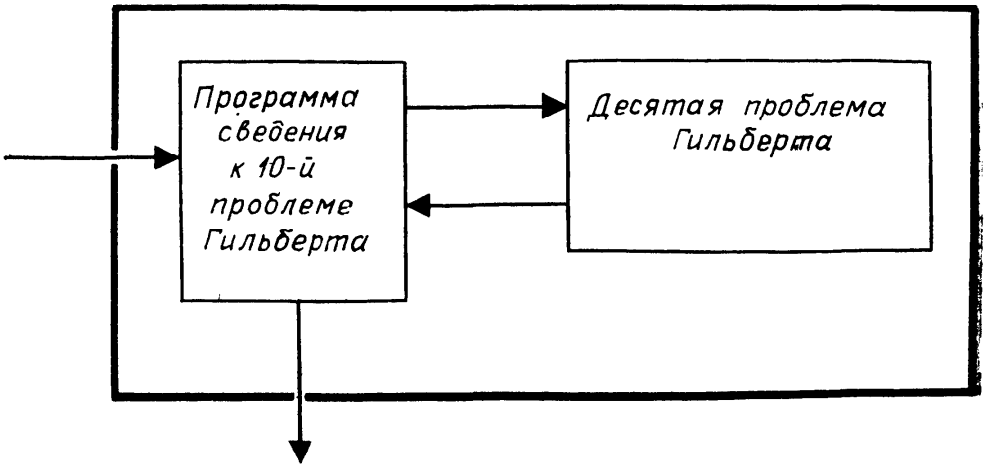


Рис. 5. Проблема остановки сводима к 10-й проблеме Гильберта.

Другой важной темой, которую теория сложности унаследовала от теории вычислимости, является различие между способностью решить задачу и способностью проверить решение. Даже несмотря на то, что не существует общего метода нахождения решения для диофантова уравнения, легко проверить предлагаемое решение. Например, чтобы проверить, является ли $x=3$, $y=2$, $z=-1$ решением приведенного выше диофантова уравнения, просто подставляют в него эти значения и выполняют арифметические действия. Как мы увидим ниже, в различии между решением и проверкой заключается проблема P=NP.

Некоторые из самых старых областей теоретической информатики возникли из абстрактных машин и других формализмов теории вычислений. Одна из наиболее важных среди этих областей — теория сложности вычислений. Вместо того чтобы просто спрашивать, разрешима ли вообще задача, теория сложности спрашивает, насколько трудно решить эту задачу. Другими словами, теория сложности интересуется возможностями универсальных вычислительных устройств, таких, как машина Тьюринга, при наличии ограничений на время выполнения или объем памяти, которая может быть использована. Первые проблески теории сложности появились в статьях, опубликованных в 1959 и 1960 гг. Микаэлем Рабином, а также Робертом Макнотоном и Хидео Ямадой, но начало современной эры теории сложности отметила статья 1965 г. Юриса Хартманиса и Ричарда Стирнса. Используя машину Тьюринга как модель абстрактного вычислителя, Хартманис и Стирнс предложили точное определение «класса сложности», состоящего из всех задач, решаемых за число шагов, ограниченных некоторой данной функцией длины входа n. Приспосабливая диагональный метод, который Тьюринг использовал для доказательства неразрешимости проблемы остановки, они доказали много интересных результатов о структуре классов сложности. Все, кто прочитал их статью, не могли не понять, что теперь имеется удовлетворительная формальная структура для работы над вопросами, которые Эдмондс поставил ранее интуитивно, — вопросами о том, например, разрешима ли задача о коммивояжере за полиномиальное время.

В том же самом году я изучал теорию вычислимости по великолепной книге Хартли Роджерса, который ранее был моим учителем в Гарварде. Я вспоминаю, как спрашивал себя тогда, может ли понятие сводимости, которое занимает столь важное место в теории вычислимости, играть роль в теории сложности, но я не видел, как осуществить эту связь. Примерно в то же время Микаэль Рабин, который впоследствии получил тьюринговскую премию в 1976 г., был гостем Исследова-

тельской лаборатории IBM в Йорктаун Хайтс, приехав из Еврейского университета в Иерусалиме. Нам обоим случилось жить в одном и том же доме на окраине Нью-Йорка, и мы завели привычку совершать вместе поездку к Йорктаун Хайтс и обратно. Рабин — глубоко оригинальный мыслитель и один из основателей как теории автоматов, так и теории сложности, и благодаря своим ежедневным дискуссиям с ним во время поездок вдоль Соумилл ривер парквэй я получил гораздо более широкий взгляд на логику, теорию вычислимости и теорию абстрактных вычислительных машин.

В 1968 г., — возможно, под влиянием общественных волнений, охвативших страну, — я решил перебраться в Калифорнийский университет в Беркли, который был центром студенческого движения. Годы в IBM были решающими в моем становлении как ученого. Возможность работать с такими выдающимися учеными, как Алан Хофман, Раймонд Миллер, Арнольд Розенберг и Шмуэль Виноград, была просто бесценной. Мой новый круг коллег включал Майкла Харрисона, известного специалиста по теории языков, который и соблазнил меня перебраться в Беркли; Юджина Лоулера, эксперта по комбинаторной оптимизации; Мануэля Блюма, основателя теории сложности, который потом выполнил выдающуюся работу на границе теории чисел и криптографии, и Стивена Кука, чья работа по теории сложности повлияет на меня так сильно через несколько лет. В Отделении математики работали Джулия Робинсон, чья работа над 10-й проблемой Гильберта вскоре принесет плоды; Роберт Соловей, знаменитый логик, открывший важный рандомизированный алгоритм распознавания простоты числа, и Стив Смейл, чья потрясающая работа по вероятностному анализу алгоритмов линейного программирования оказала на меня влияние через несколько лет. А на другом берегу Калифорнийской бухты, в Станфорде, были Данциг, отец линейного программирования, Дональд Кнут, основавший область структур данных и анализа алгоритмов, а также Роберт Тарьян, тогда студент старшего курса, и Джон Хопкрофт, приглашенный профессор из Корнеллского университета, который блестяще применял технику структур данных к анализу графовых алгоритмов.

В 1971 г. Кук, который к тому времени переехал в Университет Торонто, опубликовал свою историческую статью «О сложности процедур доказательства теорем». Кук рассмотрел классы задач, которые мы теперь называем P и NP, и ввел понятие, которое мы теперь называем NP-полнотой. Неформально класс P состоит из всех тех задач, что могут быть решены за полиномиальное время. Так, задача марьяжа лежит в P, потому что венгерский алгоритм решает частный случай раз-

мера n примерно за n^3 шагов, но задача о коммивояжере, похоже, не лежит в P, так как каждый из известных методов ее решения требует экспоненциального времени. Если исходить из того, что вычислительная задача практически неразрешима, если не существует алгоритма с полиномиальным временем для ее решения, то все практически решаемые задачи лежат в P. Класс NP состоит из всех тех задач, для которых любое предложенное решение может быть проверено за полиномиальное время. Например, рассмотрим версию задачи о коммивояжере, в которой входные данные состоят из расстояний между всеми парами городов и «целевого числа» T, и задание состоит в определении того, существует ли маршрут длины, меньшей или равной T. По-видимому, крайне трудно определить, существует ли такой маршрут, но если такой маршрут нам задан, мы можем легко проверить, верно ли, что его длина меньше или равна T; следовательно, эта версия задачи о коммивояжере лежит в классе NP. Подобным образом через введение целевого числа T все комбинаторные задачи оптимизации, обычно рассматриваемые в областях коммерции, науки и техники, имеют версии, лежащие в классе NP.

Таким образом, NP представляет собой область, в которую обычно попадают комбинаторные задачи; внутри NP лежит P, класс задач, имеющих эффективные решения. Фундаментальный вопрос состоит в следующем: как соотносятся класс P и класс NP? Ясно, что P является подклассом NP, и вопрос, к которому Кук привлек внимание, состоит в том, могут ли P и NP быть одним и тем же классом. Если бы P совпадал с NP, то возникли бы ошеломляющие следствия: это означало бы, что всякая задача, решения для которой легко проверить, решались бы легко; это означало бы, что если теорема имеет короткое доказательство, то некой универсальной процедурой можно было бы найти это доказательство быстро; это означало бы, что все обычные задачи комбинаторной оптимизации решались бы за полиномиальное время. Короче, это означало бы, что можно избавиться от проклятия комбинаторных взрывов. Но несмотря на все эти эвристические доводы о том, что было бы слишком хорошо, если бы P и NP совпадали, никакого доказательства, что $P \neq NP$, найдено не было, и некоторые специалисты даже верят в то, что никакого доказательства никогда и не будет найдено.

Наиболее важное достижение статьи Кука состоит в доказательстве того, что $P = NP$ тогда и только тогда, когда конкретная вычислительная задача, называемая проблемой выполнимости, принадлежит P. Проблема выполнимости происходит из математической логики и имеет приложения в теории переключательных схем, но может быть сформулирована как простая

комбинаторная головоломка: можно ли из нескольких заданных последовательностей прописных и строчных букв выбрать по букве из каждой последовательности, не выбирая обе (прописную и строчную) версии никакой буквы? Например, если последовательности суть *Abc*, *BC*, *aB* и *ac*, можно выбрать *A* из первой последовательности, *B* из второй и третьей и *c* из четвертой; отметим, что одна и та же буква может быть выбрана более чем однажды при условии, что мы не выбираем обе ее (прописную и строчную) версии. Пример, где требуемый выбор невозможен, задается следующими четырьмя последовательностями: *AB*, *Ab*, *aB* и *ab*.

Проблема выполнимости, очевидно, принадлежит NP, так как легко проверить, удовлетворяет ли условиям задачи предложенный выбор букв. Кук доказал, что если проблема выполнимости решается за полиномиальное время, то всякая задача из NP тоже решается за полиномиальное время, так что P=NP. Таким образом, мы видим, что эта кажущаяся странной и непоследовательной задача есть архетипическая комбинаторная задача, она служит ключом к эффективному решению всех задач из NP.

Доказательство Кука было основано на понятии сводимости, которое мы упоминали ранее, когда говорили о теории вычислимости. Он показал, что всякий частный случай задачи из NP может быть преобразован в соответствующий случай проблемы выполнимости таким образом, что оба они одновременно или имеют, или не имеют решения. Более того, это преобразование может быть выполнено за полиномиальное время. Другими словами, проблема выполнимости является достаточно общей для фиксации структуры любой задачи из NP. Отсюда следует, что если бы мы умели решить проблему выполнимости за полиномиальное время, то мы сумели бы построить алгоритм, работающий полиномиальное время, для любой задачи из NP. Этот алгоритм состоял бы из двух частей: процедуры, переводящей за полиномиальное время частные случаи данной задачи в частные случаи проблемы выполнимости, и процедуры для решения самой проблемы выполнимости за полиномиальное время (рис. 6).

По прочтении статьи Кука я тотчас же понял, что его концепция архетипической комбинаторной задачи была формализацией идеи, уже давно бывшей частью фольклора комбинаторной оптимизации. Работающие в этой области понимали, что задача целочисленного программирования, которая по существу есть проблема разрешимости для системы линейных неравенств в целых числах, является достаточно общей, чтобы выразить условия любой из обычно встречающихся задач комбинаторной оптимизации. Данциг опубликовал статью на эту тему в 1960 г.

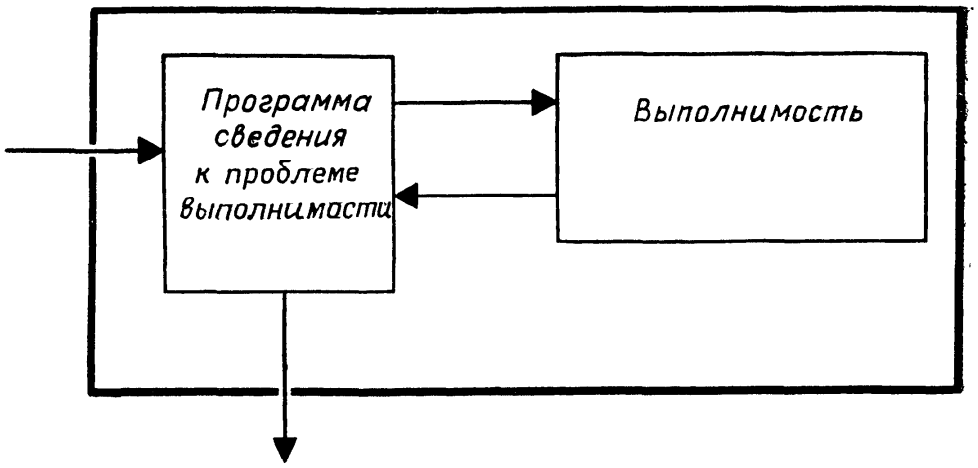


Рис. 6. Задача о коммивояжере за полиномиальное время сводима к проблеме выполнимости.

Из-за того что Кук интересовался доказательством теорем, а не комбинаторной оптимизацией, он выбрал другую архетипическую задачу, но основная идея была та же самая. Однако было решающее отличие: используя аппарат теории сложности, Кук построил формализм, в котором архетипическая природа данной задачи смогла стать теоремой, а не неформальным тезисом. Интересно, что Леонид Левин, который тогда работал в Ленинграде, а сейчас профессор Бостонского университета, независимо открыл по существу ту же самую совокупность идей. Его архетипическая задача была связана с покрытием конечных областей плоскости костями домино.

Я решил исследовать, являются ли некоторые классические комбинаторные задачи, которые издавна считались практически невыполнимыми, архетипическими в смысле Кука. Я назвал такие задачи «полиномиально полными», но этот термин был вытеснен более точным термином «NP-полные». Задача NP-полна, если она лежит в классе NP и всякая задача из NP полиномиально сводима к ней. Таким образом, по теореме Кука проблема выполнимости NP-полна. Для доказательства того, что задача из NP является NP-полной, достаточно показать, что некоторая задача, NP-полнота которой доказана, полиномиально сводима к данной задаче. Построением серии полиномиальных по времени сведений я показал, что большинство классических задач упаковки, покрытия, паросочетания, разбиения, выбора маршрутов и составления расписаний, которые возникают в комбинаторной оптимизации, NP-полны. Я представил эти результаты в 1972 г. в статье под названием «Взаимная сводимость комбинаторных задач». Мои ранние результаты были быстро уточнены и усилены другими авторами, и в следующие несколько лет для сотен различных задач, возникающих фактически

в каждой области, где делаются вычисления, было показано, что они NP-полны.

РАБОТА С NP-ПОЛНЫМИ ЗАДАЧАМИ

Я был вознагражден за мои достижения в исследовании NP-полных задач административным постом. С 1973 по 1975 г. я возглавлял вновь образованный отдел информатики в Беркли, и мои обязанности оставляли мне мало времени для исследований. В результате я оказался на обочине в самый активный период, в то время, когда были найдены многие примеры NP-полноты и были предприняты первые попытки обойти отрицательные следствия NP-полноты.

Результаты по NP-полноте, доказанные в начале 1970-х годов, показали, что огромное большинство задач комбинаторной оптимизации, возникающих в коммерции, науке и технике, практически невыполнимы, если не верно P=NP; никакие методы их решения не могут полностью избежать комбинаторных взрывов. Как же тогда нам справляться с такими задачами на практике? Один из возможных подходов исходит из того, что достаточно хороши решения, близкие к оптимальным: коммивояжер, вероятно, будет удовлетворен маршрутом, на несколько процентов длиннее оптимального. Следуя такому подходу, исследователи начали искать полиномиальные по времени алгоритмы, которые гарантируют почти оптимальные решения NP-полных комбинаторных задач. В большинстве случаев критерий эффективности приближенного алгоритма формулировался в виде верхней оценки отношения стоимости решения этим алгоритмом к стоимости оптимального решения.

Некоторые из наиболее интересных работ по приближенным алгоритмам с гарантиями эффективности содержали одномерную задачу упаковки в контейнер. В этой задаче совокупность объектов различных размеров должна быть упакована в контейнеры одинаковой емкости. Цель состоит в минимизации числа контейнеров, используемых для упаковки, при условии, что сумма размеров упаковываемых объектов в каждом контейнере не должна превосходить емкости контейнера. В середине 1970-х серия статей по приближенным алгоритмам упаковки в контейнеры достигла кульминации в анализе Дэвида Джонсона алгоритма «первый подходящий по убыванию». В этом простом алгоритме объекты рассматриваются в порядке убывания их размеров, и каждый объект — по очереди — помещается в первый из контейнеров, который может его принять. В примере, изображенном на рис. 7, например, есть четыре контейнера, емкость каждого равна 10, и девять предметов по размеру в пределах от 2 до 8. Джонсон показал, что этот простой ме-

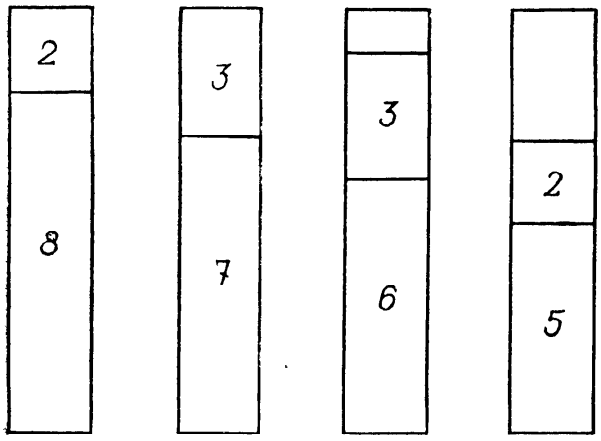


Рис. 7. Упаковка, порождаемая алгоритмом «первый подходящий по убыванию».

тод гарантирует относительную погрешность, не превышающую 2/9, другими словами, число требуемых контейнеров никогда не превосходит больше чем примерно на 22 процента числа контейнеров в оптимальном решении. Через несколько лет эти результаты были еще улучшены, и в конце концов было показано, что относительная погрешность может быть сделана сколь угодно малой, хотя в этом полиномиальном по времени алгоритме отсутствовала простота проанализированного Джонсоном алгоритма «первый подходящий по убыванию».

Исследования полиномиальных по времени приближенных алгоритмов выявили интересные различия среди NP-полных задач дискретной оптимизации. Для некоторых задач относительная погрешность может быть сделана сколь угодно малой, для других ее можно снизить до некоторого уровня, но, по-видимому, не более того, иные задачи сопротивляются всем попыткам найти алгоритмы с ограниченной относительной погрешностью, и, наконец, существуют задачи, для которых существование полиномиального по времени приближенного алгоритма с ограниченной относительной погрешностью означало бы, что P=NP.

В течение седьмого года, который последовал за моим пребыванием на административном посту, я начал размышлять о разрыве между теорией и практикой в области комбинаторной оптимизации. В области теоретической новости были неутешительны. Почти все задачи, которые хотелось решить, были NP-полны, и в большинстве случаев полиномиальные по времени алгоритмы не могли дать таких гарантий эффективности, которые были бы полезны на практике. Тем не менее было много алгоритмов, которые, судя по всему, работают вполне хорошо на практике, хотя они и нуждаются в теоретическом обосновании. Например, Линь и Керниган разработали весьма успешную стратегию локальных улучшений для задачи о коммивояжере. Их алгоритм просто стартовал со случайного маршрута и все улучшал и улучшал его добавлением и исключением нескольких ребер, покуда маршрут в конце концов не становился таким, что не допускал локальных улучшений. В специально построенных частных случаях их алгоритм работал ужасно, но на практике он приводил к почти оптимальным решениям.

Подобная ситуация наблюдается и для симплекс-алгоритма, одного из наиболее важных среди всех вычислительных методов: он надежно решает те крупные задачи линейного программирования, которые возникают в приложениях, несмотря на то, что в некоторых искусственно построенных примерах он выполнялся за экспоненциальное число шагов.

Представлялось, что успех таких неточных или теоретически необоснованных алгоритмов был эмпирическим фактом, требующим объяснения. Кроме того, представлялось, что объяснение этого факта будет неизбежно требовать отхода от традиционных парадигм теории сложности, которые оценивают алгоритм согласно его эффективности в худшем из возможных случаев. Традиционный анализ худшего случая — доминирующее направление теории сложности — соответствует сценарию, в котором частные случаи решаемой задачи построены бесконечно умным противником, который знает структуру алгоритма и подбирает входную информацию так, что будет предельно усложнять работу этого алгоритма. Такой сценарий приводит к заключению, что симплекс-алгоритм и алгоритм Линя — Кернигана безнадежно дефектны. Я начал изучать другой подход, в котором входная информация предполагается поступающей от пользователя, который просто выбирает свои частные случаи, исходя из некоторого разумного распределения вероятности, не пытаясь ни повредить, ни помочь алгоритму.

В 1975 г. я решил попытать счастья и взялся за исследование вероятностного анализа комбинаторных алгоритмов. Я должен сказать, что это решение потребовало некоторой храбрости, так как имело своих оппонентов, которые совершенно правильно указывали, что нет никакого способа узнать, какая именно входная информация поступит на входы алгоритма, и что лучший вид гарантий, если можно получить, их, — гарантии для худшего случая. Я чувствовал, однако, что в случае NP-полных задач мы не получим желаемых гарантий для худшего случая и что вероятностный подход — лучший путь — и, возможно, единственный — к пониманию того, почему эвристические комбинаторные алгоритмы работают так хорошо на практике.

Вероятностный анализ начинается с предположения, что частные случаи задачи представляют собой репрезентативную выборку из генеральной совокупности с заданным распределением вероятностей. В случае задачи о коммивояжере, например, одно из возможных допущений состоит в том, что расположение каждого из n городов выбирается независимо из равномерного распределения на единичном квадрате. В этом предположении мы можем изучать распределение вероятностей длины оптимального маршрута или длину маршрута, построенного

тем или иным алгоритмом. В идеале цель состоит в доказательстве того, что некоторый простой алгоритм дает оптимальные или почти оптимальные решения с высокой вероятностью. Конечно, такой результат осмыслен только в том случае, когда распределение вероятностей для частных случаев задачи некоторым образом подобно совокупности случаев, возникающих в реальной жизни, или тогда, когда вероятностный анализ достаточно нечувствителен к характеристикам распределения, чтобы быть пригодным для широкого диапазона возможных распределений вероятностей.

Один из наиболее удивительных феноменов теории вероятностей — закон больших чисел, согласно которому кумулятивный эффект большого числа случайных событий в высокой степени предсказуем, даже если исходы индивидуальных событий совершенно непредсказуемы. Например, мы можем уверенно предсказать, что в длинной серии бросаний монеты примерно половина исходов — орел. Вероятностный анализ обнаружил, что то же явление управляет поведением многих комбинаторных алгоритмов оптимизации, когда входная информация подчинена простому распределению вероятностей: с очень высокой вероятностью выполнение алгоритма развивается хорошо предсказуемым образом и полученное решение близко к оптимальному. Например, статья 1960 г. Бердвуда, Холтона и Хаммерсли показывает, что если n городов в задаче о коммивояжере выбираются независимо с равномерным распределением на единичном квадрате, то при достаточно большом n длина оптимального маршрута будет почти наверняка очень близкой к некоторой универсальной константе, умноженной на корень квадратный из числа городов. Воодушевленный этим результатом, я показал, что, когда число городов чрезвычайно велико, простой алгоритм «разделяй и властвуй» почти всегда будет приводить к маршруту с длиной, очень близкой к длине оптимального маршрута (рис. 8). Алгоритм стартует с разбиения области, где расположены города, на прямоугольники, каждый из которых содержит малое число городов. Затем он строит оптимальный маршрут для городов каждого прямоугольника. Объединение всех этих маленьких маршрутов сильно напоминает общий маршрут, но отличающийся от него лишними посещениями пограничных городов. Наконец, алгоритм выполняет своего рода «локальную хирургию» с целью исключения этих излишних посещений и образования маршрута.

Можно привести еще много примеров, где простые приближенные алгоритмы почти наверняка дают близкие к оптимальным решения для случайных «больших» частных случаев NP-полных оптимизационных задач. Например, моя студентка Салли Флойд, продолжая ранние работы об упаковке в контей-

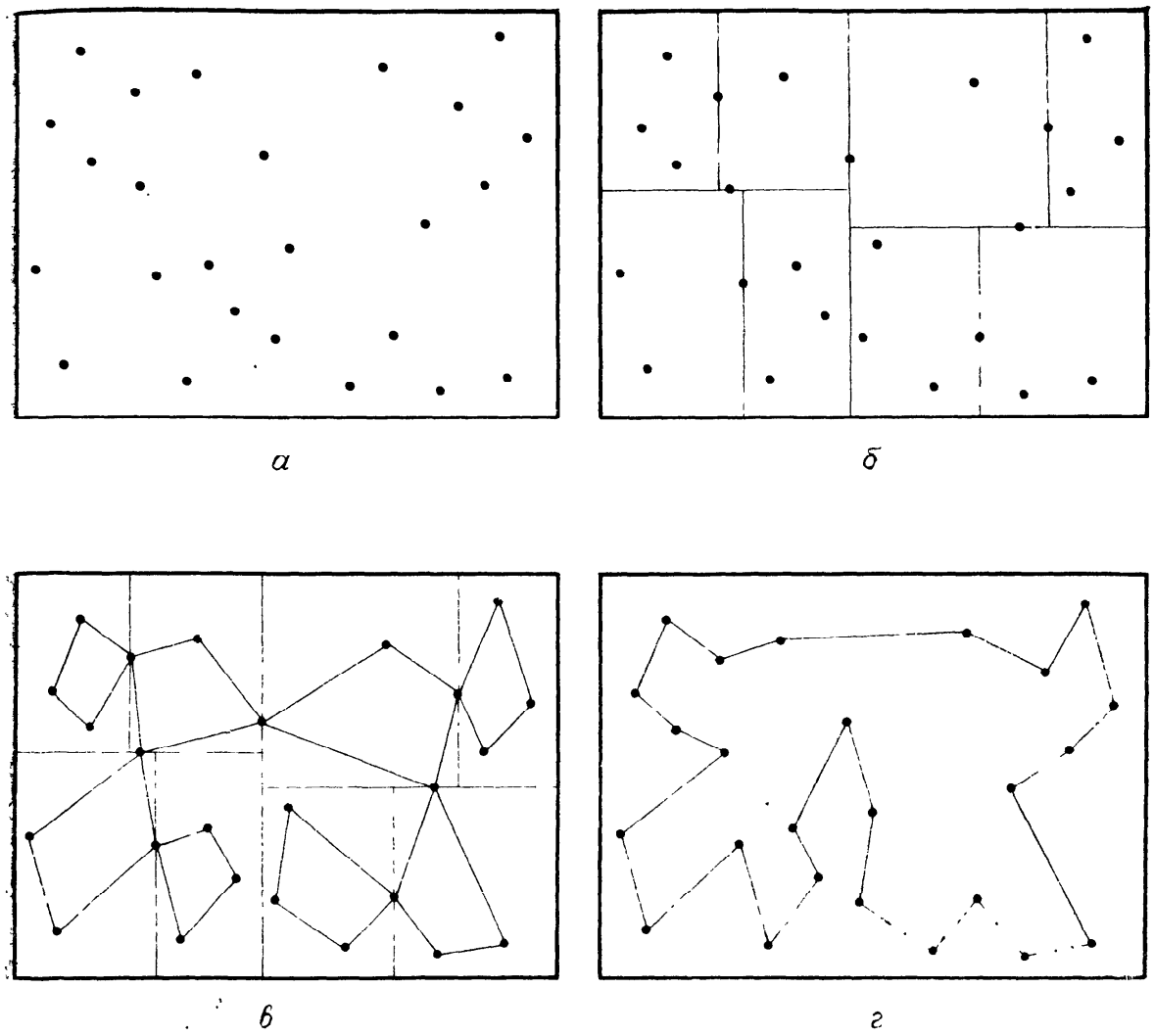


Рис. 8. Алгоритм «разделяй и властвуй» для задачи о коммивояжере на плоскости.

неры Бентли, Джонсона, Лейтона, Макгеоха и Макгеоха, показала, что если объекты, которые нужно упаковать, выбираются независимо с равномерным распределением на интервале [0, 1/2], то независимо от числа объектов алгоритм «первый подходящий по убыванию» почти наверняка даст упаковку, в которой неиспользованный объем не превышает объема 10 контейнеров.

Среди наиболее замечательных приложений вероятностного анализа — его применение к задаче линейного программирования. Геометрически эта задача сводится к нахождению вершины многогранника, ближайшей к некоторой внешней гиперплоскости. Алгебраически это эквивалентно минимизации линейной функции в области, ограниченной линейными неравенствами. Линейная функция измеряет расстояние до гиперплоскости, а линейные неравенства соответствуют гиперплоскостям, ограничивающим многогранник.

Симплекс-алгоритм для задач линейного программирования есть метод взбирания на гору. Он все время переходит от одной вершины к соседней вершине, всегда приближаясь к внешней гиперплоскости. Алгоритм заканчивает работу, когда он достигает вершины, ближайшей к этой гиперплоскости по сравнению с соседними вершинами; эта вершина заведомо есть оптимальное решение. В худшем случае симплекс-алгоритм требует растущего экспоненциально числа операций относительно числа линейных неравенств, необходимых для описания многогранника, но на практике число итераций редко превосходит число линейных неравенств более чем в три или четыре раза.

Карл-Хейнц Боргвардт из Западной Германии и Стив Смэйл из Беркли были первыми исследователями, использовавшими вероятностный анализ для объяснения непонятного успеха симплекс-алгоритма и его вариаций. Их анализ основан на вычислении некоторых многомерных интегралов. С моей ограниченной подготовкой в области математического анализа я нашел их метод трудновоспринимаемым. К счастью, один из моих коллег в Беркли, Айлан Адлер, предложил один подход, который сулил привести к вероятностному анализу вовсе без вычислений. При этом используются некоторые принципы симметрии, чтобы сделать требуемые усреднения и волшебным образом прийти к ответу.

Следуя этому направлению исследований, Адлер, Рон Шамир и я показали в 1983 г., что при весьма широких допущениях о вероятностных характеристиках входных данных ожидаемое число итераций, выполненных одной из версий симплекс-алгоритма, растет всего лишь как квадрат числа линейных неравенств. Тот же самый результат получил при помощи многомерных интегралов Майкл Тодд, а также Адлер и Нимрод Меджиддо. Я считаю, что эти результаты существенно помогают понять, почему симплекс-метод работает так хорошо.

Вероятностный анализ алгоритмов комбинаторной оптимизации был главной темой моих исследований за последнее десятилетие. В 1975 г., когда я впервые занялся такими исследованиями, было очень мало примеров анализа этого рода. К настоящему времени на эту тему написаны сотни статей, и все классические комбинаторные задачи были подвергнуты вероятностному анализу. Эти результаты привели к существенному пониманию того, насколько эти задачи могут быть «приручены» на практике. Тем не менее я думаю, что это предприятие удалось лишь отчасти. В силу ограничений существующих методов мы продолжаем работать с простейшими вероятностными моделями, и даже тогда многие из наиболее интересных и успешных алгоритмов не поддаются анализу. После всего, что было

сказано и сделано, построение практических алгоритмов комбинаторной оптимизации остается в той же мере искусством, что и наукой.

РАНДОМИЗИРОВАННЫЕ АЛГОРИТМЫ

Алгоритмы, в процессе выполнения которых бросается монета, время от времени предлагались с тех пор, как появились первые компьютеры, но систематическое изучение таких рандомизированных алгоритмов началось только примерно с 1976 г. Интерес к этой теме привлекли два удивительно эффективных рандомизированных алгоритма проверки того, является ли число n простым; один из этих алгоритмов предложили Соловей и Фолкер Штрассен, а другой — Рабин. Следующая статья Рабина дала дальнейшие примеры и обоснование систематического изучения рандомизированных алгоритмов, и докторская диссертация Джона Гилла под руководством моего коллеги Блюма заложила основы общей теории рандомизированных алгоритмов.

Чтобы понять преимущества бросания монеты, вернемся снова к сценарию, связанному с анализом худшего случая, в котором всезнающий противник выбирает те частные случаи, в которых алгоритм работает хуже всего. Рандомизация делает поведение алгоритма непредсказуемым, даже когда частный случай фиксирован, и так можно сделать трудным и даже невозможным для противника выбрать частный случай, который затруднит работу алгоритма. Существует полезная аналогия с футболом, в которой алгоритм подобен наступающей команде и противник защищается. Детерминированный алгоритм подобен команде, которая полностью предсказуема в тактике игры, что позволяет другой команде построить непробиваемую защиту. Как знает всякий полузащитник, внесение небольшого разнообразия в игру достаточно, чтобы заставить защищающую команду играть честно.

В качестве конкретной иллюстрации преимуществ бросания монеты я приведу простой рандомизированный алгоритм сопоставления с образцом, изобретенный Рабином и мною в 1980 г. Задача сопоставления с образцом — одна из основных в обработке текстов. Задана цепочка n бит, называемая образцом, и гораздо более длинная цепочка, называемая текстом; задача состоит в том, чтобы обнаружить, входит ли образец в текст как связный блок (рис. 9). Грубый метод решения этой задачи состоит в непосредственном сравнении образца с каждым n-разрядным блоком внутри текста. В худшем случае время выполнения этого метода пропорционально произведению длины образца на длину текста. Во многих приложениях обработки

Образец 11001

Текст 011011101 $\boxed{11001}$ 00

Рис. 9. Проблема сопоставления с образцом.

текстов этот метод работает неприемлемо медленно, кроме случая очень коротких образцов.

Наш метод обходит эту трудность при помощи приема простого вырубания. Мы определим «функцию отпечатков пальцев», которая связывает с каждой n-разрядной последовательностью гораздо более короткую, называемую «отпечатком пальца». Функция отпечатка выбрана так, что можно пройти по тексту, быстро вычисляя отпечаток каждого n-разрядного блока. Тогда вместо того, чтобы сравнивать образец с каждым таким блоком текста, мы сравниваем отпечаток образца с отпечатком каждого такого блока. Если отпечаток образца отличается от отпечатков всех блоков, то мы знаем, что образец не появляется в качестве блока в тексте.

Метод сравнения коротких отпечатков вместо длинных цепочек сильно уменьшает время работы, но приводит к возможности ложных сопоставлений, когда образец и некоторый блок текста имеют одинаковые отпечатки, хотя сами они не совпадают. Ложное сопоставление представляет серьезную проблему: фактически для всякого конкретного выбора функции отпечатков противник может построить такой пример образца и текста, что ложное сопоставление возникнет в каждой позиции текста. Поэтому необходимо некоторое дублирование для защиты от ложных сопоставлений, и преимущества метода отпечатков, казалось бы, должны потеряться.

К счастью, преимущества метода отпечатков можно сохранить при помощи рандомизации. Вместо работы с единственной функцией отпечатков рандомизированный метод имеет в своем распоряжении большое семейство различных легковычислимых функций отпечатков. Когда частный случай, состоящий из образца и текста, предъявлен, алгоритм наудачу выбирает функцию отпечатка из этого большого семейства и использует эту функцию для проверки совпадения образца и текста. Из-за того что функция отпечатков заранее неизвестна, для противника невозможно построить частный случай задачи, который приводит к ложному сопоставлению. Можно показать, что независимо от выбора образца и текста вероятность ложного сопоставления очень мала. Например, если длина образца составляет 250 бит и текста — 4000 бит, можно работать с легковычислимыми 32-битовыми отпечатками, и все еще будет гарантия, что вероятность ложного сопоставления меньше одной

тысячной для каждого возможного частного случая. Во многих приложениях обработки текстов эта вероятностная гарантия достаточно хороша, чтобы исключить необходимость повторной проверки, и таким образом преимущества метода отпечатков снова восстановлены.

Рандомизированные алгоритмы и вероятностный анализ алгоритмов — два разных способа избежать анализа эффективности детерминированных алгоритмов в худшем случае. В первом способе случайность вводится в поведение самого алгоритма, а во втором предполагается, что случайность присутствует в выборе частных случаев. Подход, основанный на рандомизированных алгоритмах, конечно, более привлекателен из этих двух, так как он обходится без предположений о среде, в которой алгоритм будет использоваться. Однако пока не показана эффективность рандомизированных алгоритмов в борьбе с комбинаторными взрывами, характерными для NP-полных задач, и, видимо, оба этих подхода будут иметь свои применения.

ЗАКЛЮЧЕНИЕ

Итак, мой рассказ подошел к концу, и мне бы хотелось закончить кратким замечанием, что значит сегодня работать в теоретической информатике. Когда я принимаю участие в ежегодном симпозиуме ACM по теории вычислений, или присутствую на ежемесячном теоретическом семинаре Залива, или поднимаюсь на холм позади университетского городка Беркли к Математическому научно-исследовательскому институту, где выполняется годовая программа по вычислительной сложности, — я поражен масштабами работы, проводимой в этой области. Я горд, что связан с областью исследования, в которой выполнено так много великолепных работ, и рад, что я в состоянии время от времени помогать очень талантливым молодым исследователям стать на ноги в этой области. Благодарю вас за предоставленную мне возможность выступить как представителю моей области на этом собрании.

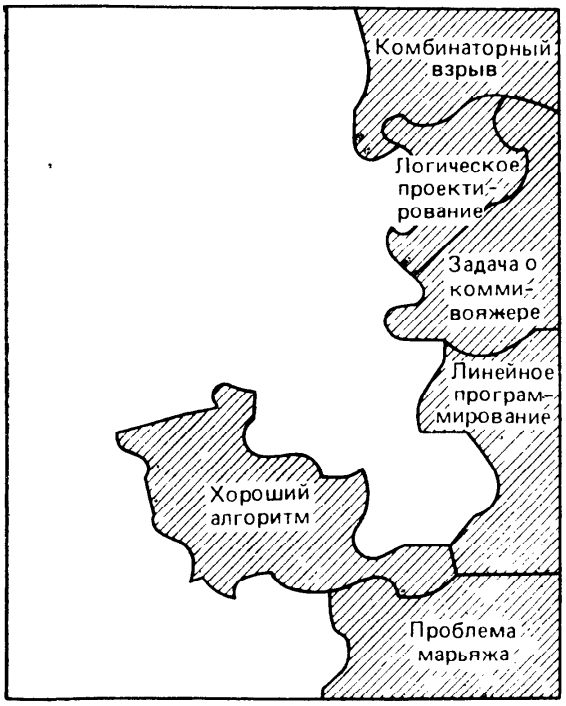
Посткриптум

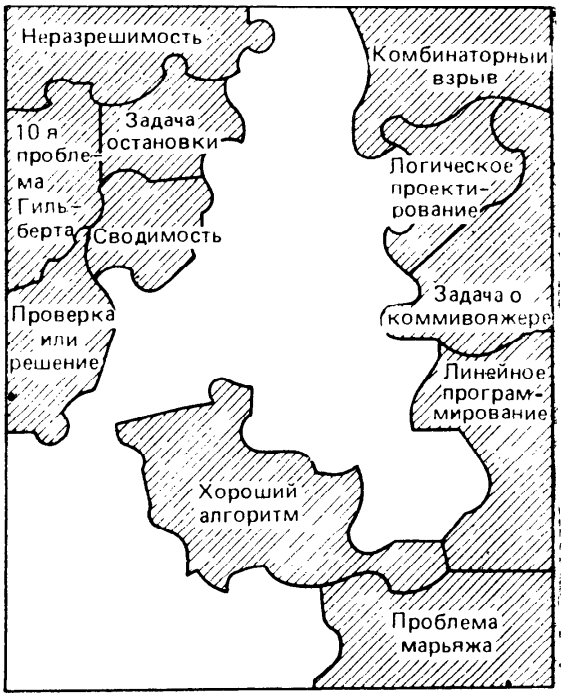
Сборка теории сложности из кусков

Карен А. Френкель

Обозреватель Communications of the ACM

Для иллюстрации того, «в какой удивительной степени в теории сложности употребляются аналоги теории вычислимости», Ричард Карп построил эту концептуальную карту или головоломку. Чтобы собрать эту головоломку на плоскости, он использует «алгоритм для плоских графов». Наиболее удаленные друг от друга части могут при первом рассмотрении казаться независимыми, «но в конце концов теория NP-полноты собирает их вместе», говорит Карп.





В верхнем правом углу головоломки располагаются понятия, относящиеся к комбинаторным взрывам, и понятия «хорошего» или «эффективного» алгоритма. В свою очередь «слож-

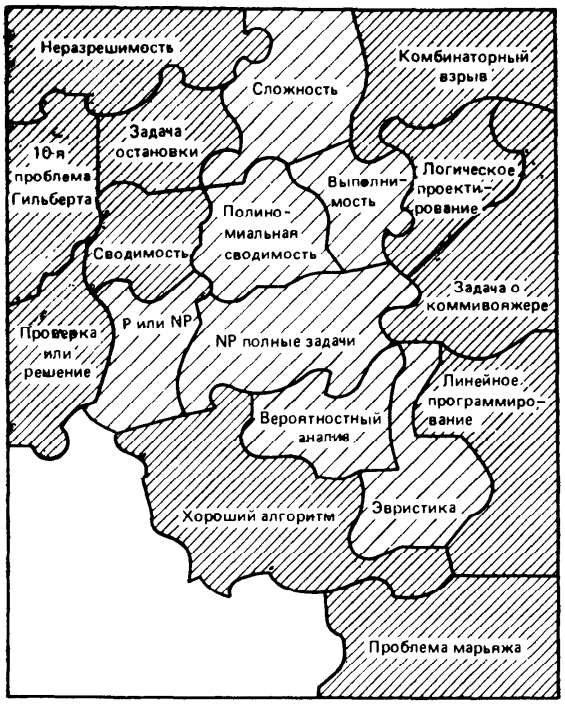
ность» связывает эти понятия с левым верхним углом, в котором представлено то, что интересовало прежних специалистов по теории вычислимости.

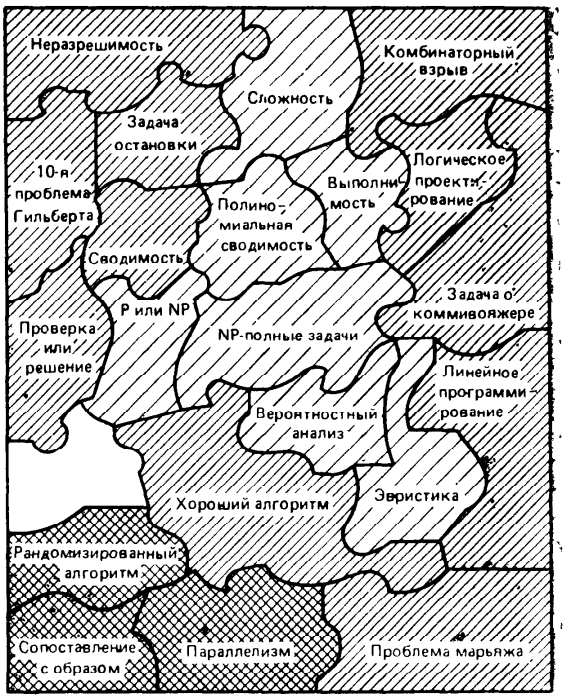
Задача о коммивояжере ближе к верхнему правому углу, потому что она, по всей вероятности, практически неразрешима. Она поэтому граничит с «*NP*-полнотой» и «Комбинаторным взрывом».

До некоторой степени, однако, некоторые границы нечетки. «Линейное программирование», например, имеет аномальный статус — наиболее широко используемые на практике алгоритмы решения таких задач нехороши в теоретическом смысле, а те, что хороши теоретически, нехороши для практики. Одним из примеров является метод эллипсоидов, бывший объектом пристального внимания шесть лет назад. Он выполнялся за полиномиальное время, но полином был такой высокой степени, что этот метод хорош лишь в смысле формального соответствия теоретическому критерию и неэффективен на практике. «Причина состоит в том, что понятие полиномиального по времени алгоритма неточно передает содержание интуитивного представления об эффективных алгоритмах», — объясняет Карп. — Когда степень полинома доходит до 5 или 6, то трудно назвать такой алгоритм действительно эффективным. Таким образом, предложенное Эдмондсом понятие хорошего алгоритма — недостаточно совершенная формализация хорошего в интуитивном смысле». Далее, симплекс-алгоритм хорош на практике во всех отношениях, замечает Карп, но недостаточно хорош в соответствии со стандартной парадигмой теории сложности. Последнее добавление к решениям линейного программирования — алгоритм, построенный Нарендой Кармаркаром и, по мнению некоторых, бросающий вызов симплекс-алгоритму, хорош в техническом смысле и также, видимо, вполне эффективен на практике, говорит Карп.

«Хороший алгоритм» примыкает к «Эвристике», потому что эвристический алгоритм может работать хорошо, но имеет недостаточное теоретическое обоснование. Некоторые эвристические алгоритмы всегда быстры, но иногда не могут дать хороших решений. Другие всегда дают оптимальные решения, но не гарантируют быстроты выполнения. Симплекс-алгоритм — последнего типа.

«Неразрешимость», «Комбинаторный взрыв» и «Сложность» располагаются на одном уровне, потому что они аналогичны друг другу; неразрешимость включает неограниченный поиск, тогда как комбинаторные взрывы по определению представляют собой очень длинные, но не неограниченные поиски. Теория сложности заполняет разрыв, потому что вместо того, чтобы спрашивать о том, может ли быть вообще решена задача,





она ставит вопросы о *ресурсах*, необходимых для решения задачи.

В нижнем левом углу содержатся сегменты, которые интересовали Карпа в самое последнее время и в которых имеются открытые вопросы. «Рандомизированный алгоритм», например, располагается напротив «Вероятностного анализа», потому что это две альтернативы анализа детерминированных алгоритмов в худшем случае. Рандомизированные алгоритмы могут быть способны решать задачи за полиномиальное время, тогда как детерминированные не могут, и это могло бы означать расширение понятия хороших алгоритмов. Возможно, через построение программных конструкций не фон-неймановских машин можно построить более эффективные на практике алгоритмы посредством параллелизма.

Наконец, некоторые части головоломки еще не определены. Как говорит Карп, «они соответствуют неизвестной теории, которую предстоит исследовать в будущем».

Посткриптум

Интервью тьюринговских лауреатов. Сложность и параллельные вычисления

ИНТЕРВЬЮ С РИЧАРДОМ КАРПОМ

Карен А. Френкель

Обозреватель Communications of the ACM

В приводимом ниже интервью, которое он дал на конференции ACM 1985 г. в Денвере, Карп обсуждает связь между его работами и новейшими вычислительными дисциплинами, такими, как параллельные вычисления и искусственный интеллект. Прослеживая свой опыт новатора в высокой степени теоретических исследованиях в области информатики, Карп описывает, как решение пойти против устоявшихся взглядов привело к работе, благодаря которой он наиболее известен, и как находки коллег позволили ему увидеть связи между этими двумя прежде независимыми областями. Подчеркивается важность обмена идеями с коллегами, который помогает найти новые ключевые принципы.

К. Ф. Вы решили перейти от математики к информатике в самом начале вашей карьеры. Ощущаете ли вы себя теоретическим математиком, работающим в области информатики, или специалистом по информатике, работающим над ее теоретическими аспектами?

Р.К. Я думаю, что я представляю собой нечто среднее между прикладным математиком и специалистом по информатике. По-моему, a priori мою работу можно вести и на факультете математики и на факультете информатики, но исторически сложилось так, что серьезные начинания в развитии теоретической информатики брали на себя факультеты информатики.

Большинство математических факультетов «упустили мяч». Есть исключения, но, как правило, они не поняли вовремя потенциал этой области и не начали ее развивать. Таким образом, она попадала в зону внимания факультетов информатики. Сейчас математические факультеты наконец-то все более признают теоретическую информатику.

К.Ф. Математики и специалисты по информатике по-разному подходят к вычислениям?

Р.К. Когда математики применяют ЭВМ, они склонны действовать совсем не как теоретики. Если специалисту по теории чисел нужно разложить на множители какое-то число, он не

пожалеет для этого усилий. Ему нужен ответ, а более общие вопросы сложности вычислений его не волнуют. Так же обстоит дело с работающими в теории групп или в алгебраической геометрии. Их интересует конкретная группа или конкретная поверхность, и для них им нужен ответ — они становятся совсем как инженеры. И я такой же, когда программирую. Первые пять минут не забываю о вопросах теории, а потом мне просто нужно, чтобы программа работала, и я забываю, что я теоретик.

К. Ф. Почему такое большое внимание уделяется задаче о коммивояжере?

Р.К. Задача о коммивояжере — это прототип и упрощенный вариант встречающихся в практике более трудных задач. Все знают, что задача о коммивояжере — это метафора или миф; ясно, что ни один коммивояжер не будет стремиться к абсолютной минимизации своего километража. Но это задача интересная и просто формулируемая. Вероятно, она привлекает внимание в большей степени, чем того заслуживает, из-за своего завлекательного названия. Есть и другие важные задачи-прототипы с не столь завлекательными названиями, такие, как раскраска, упаковка, паросочетание, составление расписания и т. д. Так и развивается теория — если учитывать все сложности реального мира, то чистой теорией заниматься нельзя. Поэтому упрощаешь постановки задач, изучаешь их возможно тщательнее, глубоко вникаешь в их структуру и надеешься, что результаты будут переноситься на реальные проблемы.

К. Ф. Похоже, вы исследуете метатеорию — классы задач, а не реальные задачи.

Р.К. Да, верно. Есть три уровня проблем. Есть уровень решения вполне конкретного случая задачи: скажем, вам нужен кратчайший маршрут по 48 столицам континентальных штатов плюс Вашингтон. К практику этот уровень ближе всего. Затем есть уровень изучения общей задачи с упором на методы ее решения: вы хотите знать, какова сложность задачи о коммивояжере или задачи о подборе пар для худшего случая. Этот уровень выше, поскольку вас интересует не просто конкретный случай. Затем есть уровень метатеории, на котором изучается вся структура целого класса задач. Такую точку зрения мы унаследовали у логики и теории вычислимости. Вопрос приобретает вид: «Какова в целом схема классификации? Давайте взглянем на иерархию сложностей задач по мере возрастания предельно допустимого времени вычисления».

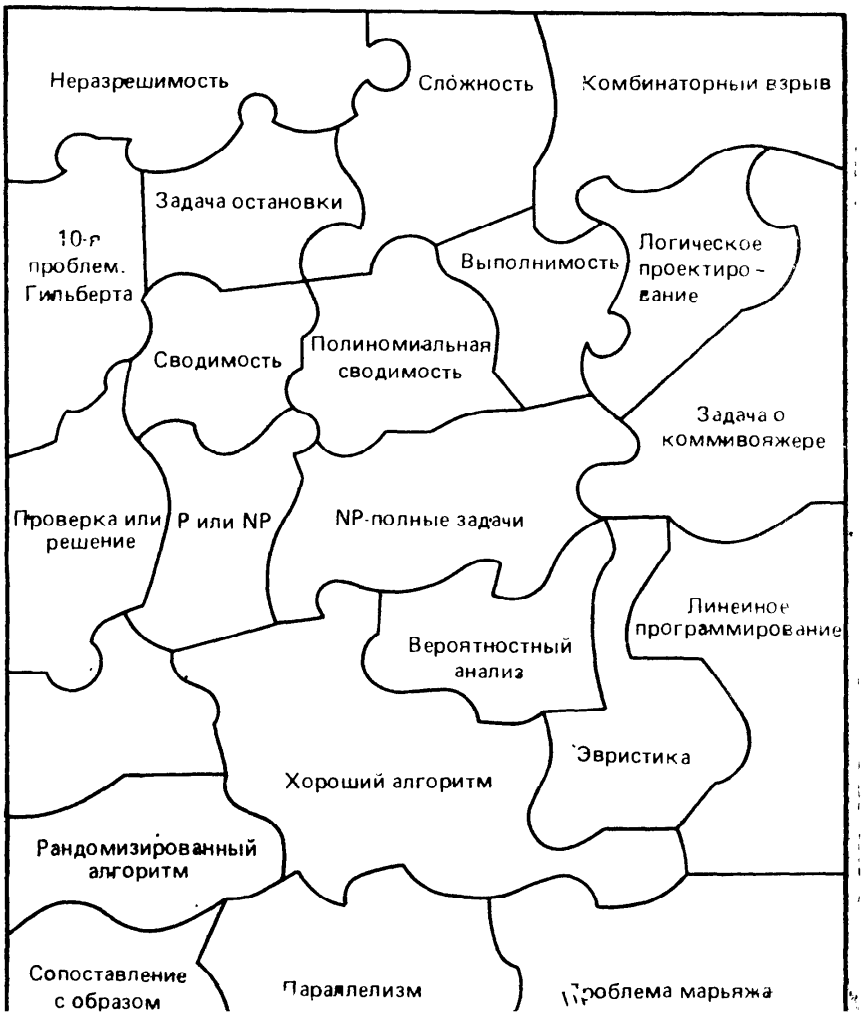
Временами два уровня соприкасаются. Такие соприкосновения обычно очень важны. В науке взаимодействие двух областей, близость которых до этого не замечали, приводит к появлению большого числа важных работ. Абстрактное изучение классов сложности связывается со свойствами конкретных

задач — вроде задачи о коммивояжере или проблемы выполнимости — посредством понятия NP-полноты.

К.Ф. Сделанный вами шаг в сторону вероятностного анализа явился отходом от парадигмы анализа для худшего случая. И вы продолжали разрабатывать вероятностный подход вопреки его хулителям. Чем было мотивировано ваше решение?

Р.К. Я не хочу создать впечатление, что до меня о вероятностном подходе никто ничего не слышал. Он, бесспорно, применялся, но в основном к задачам сортировки, поиска и структур данных, а не к задачам комбинаторной оптимизации.

Мое решение было особенно трудным потому, что до некоторой степени я был согласен с хулителями. Налицо действительно фундаментальная методологическая проблема: как выбирать распределения вероятностей? Откуда вы можете знать, какова будет статистика частных случаев задачи? Никто никогда не замерял тщательно характеристики задач из реального мира, и даже если бы измерили, то все равно лишь для одной среды вычислений. Но выхода я не видел, ведь не пойди мы по веро-



ятностной дороге, NP-полнота оказалась бы просто опустошительной.

Кроме того, было еще одно направление — исследование приближенных алгоритмов, не гарантирующих результата. Имея перед собой NP-полную задачу оптимизации, можно ослабить требование получения оптимального решения и попытаться построить быстрый алгоритм, гарантирующий для худшего случая результат, который хуже оптимального, скажем, не более чем на 20 процентов. Это еще одна очень интересная парадигма, она исследовалась, и результаты были неоднородны. Для некоторых задач трудности действительно были устранены — можно получить решение, сколь угодно близкое к оптимуму. Для некоторых других можно гарантировать попадание с точностью 22, 23 или 50 процентов. Результаты эти были очень изящны, но, по-моему, не характеризовали то, что происходит при применении практических эвристик. Практические эвристики очень хороши почти всегда, но не в худшем случае.

Так что не очень-то я рвался разрабатывать новое направление, основания которого можно было бы оспорить. Это был и личный риск, поскольку меня могли счесть слабаком. Ну, знаете: «Задачу решить не может, так вводит вероятностные предположения и облегчает себе жизнь». Но иного пути я опять-таки не видел. Некуда было деваться от явления NP-полноты.

К.Ф. Не имея оптимального решения задачи, откуда вы знаете, что ваша эвристика дает решение, близкое к оптимальному?

Р.К. Это действительно методологическая трудность. Когда выполняешь эвристический алгоритм и он вроде бы дает очень хорошие решения, уверенности все равно быть не может, поскольку не знаешь, где лежит оптимум. Можно запускать программу из различных начальных точек и получать все время одно и то же решение. Если никто никак не найдет лучшего решения — это косвенное доказательство, что ваше решение наилучшее. Можно также затратить очень много машинного времени на вычисление методом ветвей и границ и получить в конце концов решение, про которое сумеете доказать. что оно хуже оптимального не более чем на полпроцента. Затем вы запускаете на три минуты для той же задачи вашу быструю эвристику и смотрите, насколько она может приблизиться к первому решению. Иногда можно построить задачу искусственно, так что будешь знать оптимальное решение. Но вы указали на действительно серьезную методологическую проблему.

К.Ф. Недавно вы начали работу над параллельными вычислениями. Каким образом вопросы параллельной обработки влияют на наше понимание того, что такое хороший или эффективный алгоритм?

Р.К. Параллельные вычисления меня чрезвычайно интересуют, и, по-моему, эта область очаровательная. Есть несколько направлений исследований, между которыми еще не вполне установились связи; изучаются разные типы параллельной архитектуры, существует множество вопросов о том, какими должны быть процессоры и как они должны быть взаимосвязаны. Есть также вопросы численного анализа, вопросы сложности и синтеза алгоритмов.

За последние два года значительная часть моей работы выполнена в сотрудничестве с двумя израильскими коллегами, Ави Вигдерсоном и Эли Упфалом. Мы изучали сложность параллельных алгоритмов в плане довольно-таки теоретическом, работая с идеализированными моделями параллельных компьютеров. Этим мы абстрагируемся от определенных вопросов связи между компонентами, возникающими из-за того, что параллельная система — это на самом деле также и распределенная система. Например, мы можем предполагать, что возможно прямое общение между любыми двумя процессорами, что просто неверно. Но это полезные абстракции, позволяющие нам работать над некоторыми структурными вопросами, например: «Что именно в задаче поддается параллельной обработке? При каких обстоятельствах мы можем построить совершенно новый алгоритм, который в огромной степени уменьшит время, необходимое для решения задач?»

Это интересные и важные упражнения для ума, потому что они подводят к открытию совершенно новой методики структурирования алгоритмов. Очень часто предлагаемые нами параллельные алгоритмы совершенно непохожи на последовательные алгоритмы для тех же задач.

Работая над параллельными вычислениями, специалисты по информатике в основном заняты дальнейшим убыстрением алгоритмов, и так полиномиальных по времени. Мы же задаемся другими вопросами: «Какие задачи этого класса можно распараллелить в колоссальной степени? При каких условиях возможно огромное сжатие вычислений?» В будущей работе я буду больше внимания уделять применению распараллеливания к NP-полным задачам. Человек, настроенный строго теоретически, возможно, скажет: «Безнадежное дело, экспоненциальное время не уменьшить до полиномиального за счет привлечения многих процессоров, разве что у вас экспоненциально растущее число процессоров». С другой стороны, пусть даже никогда не удастся перейти от экспоненты к полиному, ясно и то, что для этих задач есть огромные возможности для распараллеливания, и оно может все-таки помочь нам справляться с комбинаторными взрывами.

Я намереваюсь проанализировать метод ветвей и границ,

деревья игр, структуры «цель — подцель», прологоподобные структуры, поиск с возвратами и все разнообразные виды комбинаторного поиска, потому что такие задачи, по-моему, очень хорошо подходят для параллельных вычислений. Пока не ясно, какую именно форму примет эта теория.

К.Ф. Какова связь между вашим теперешним интересом к параллелизму и прежней работой с Раймондом Э. Миллером?

Р.К. Было два основных периода, когда я занимался изучением параллельных вычислений. Первый — в первой половине шестидесятых годов, когда я работал с Миллером над рядом формализмов для описания параллельных вычислений. Второй — сравнительно недавний, когда я работал с Упфалом и Вигдерсоном над построением и анализом параллельных алгоритмов. Исходным вопросом для нас с Миллером был вопрос о том, нужно ли и можно ли сконструировать аппаратные средства специального назначения для параллельного выполнения часто применяющихся итеративных вычислений. Мы предложили несколько формализмов для описания параллельных вычислений. Один был вполне конкретным, а еще один очень теоретический по своему характеру. Модели и методы, предложенные нами, по духу очень походили на «систолические схемы», выдвинутые впоследствии Х. Т. Кунгом, Чарлзом Лейзерсоном и другими, хотя я не хочу сказать, что мы предвосхитили все их идеи. Многого мы, конечно, не увидели, но в некотором смысле мы делали это слишком рано — мир не вполне был к этому готов.

Нас интересовали и некоторые вопросы более качественного характера, например: «Что происходит, если работа идет асинхронно и нет общих часов, так что нельзя сказать, происходит ли А раньше В или В раньше А? Можно ли все же получить вполне определенный результат, даже если невозможно управлять порядком, в котором происходят различные события в параллельных процессах?»

Работа последнего времени идет в другом направлении. Нас интересует сложность: насколько быстро можно решить задачу при заданном числе процессоров? Эти два направления очень разные.

К.Ф. Как по-вашему, сможет ли ваша работа способствовать также конструированию параллельных процессоров и помочь определению наилучших способов связывания разных процессоров внутри машины?

Р.К. На схемном уровне то, что я делаю, имеет непосредственное отношение к таким работам. Разработка чипа для интегральной схемы немного похожа на проектирование города на 50 000 жителей. Налицо всевозможные комбинаторные проблемы, связанные с размещением разных модулей схемы и трас-

сировкой соединений между ними. На уровне архитектурном моя работа по параллельным вычислениям непосредственно неприменима, потому что я пользовался идеализированными моделями, в которых игнорировались проблемы связи между процессорами. Надеюсь, что моя работа станет ближе к вопросам архитектуры.

К.Ф. Можем ли мы из теоретических исследований узнать что-нибудь воодушевляющее о распределенной связи и распределенных протоколах?

Р.К. Да. Есть очень красивые теоретические разработки о том, сколько теряешь, когда вынужден полагаться на передачу сообщений в разреженной сети процессоров вместо прямой попарной связи между процессорами. В реалистичной распределенной системе процессоры должны не только считать, но и сотрудничать подобно почте, где сообщения летают между местами обработки. Есть очень хорошие теоретические работы о различных типах протоколов, где, как в так называемой задаче о византийских генералах, — еще одно из этих громких названий, — коллектив процессоров должен прийти к соглашению посредством передачи сообщений, даже когда часть процессоров неисправна и действует враждебно, пытаясь все запутать. Уже стало ясным, что здесь очень хорошо работает рандомизация. Протоколы, нужные для этих задач сотрудничества и связи в распределенной системе, можно упростить, если использовать бросание монеты. Что в такого рода задачах применимы рандомизированные алгоритмы — это важное открытие. Так что есть много связей между теорией и протоколами для реальных распределенных систем.

К.Ф. Применяются и такие алгоритмы, которые, хотя и недостаточно обоснованы теоретически, на практике работают очень хорошо. И практики могли бы сказать: «Эти результаты очаровательны, но раз мы можем посредством проб и ошибок наткнуться на прекрасно работающие алгоритмы, зачем же заботиться о теории?» Как ваша работа будет применяться в будущем в самом практическом смысле?

Р.К. Некоторые из важнейших комбинаторных алгоритмов никогда не удалось бы изобрести в процессе проб и ошибок, правильный теоретический подход был абсолютно необходим. Когда общая форма алгоритма определилась, часто бывает возможно настроить его эмпирически, но если действовать чисто эмпирически, то ваши знания ограничены теми весьма конкретными обстоятельствами, в которых вы экспериментируете. Результаты же анализа в большей степени поддаются обобщению. Оправданием теории, помимо ее очевидной эстетической привлекательности, служит то, что, когда вы получаете теоретический результат, он обычно приложим к целому классу

ситуаций. Это напоминает соотношение моделирования и анализа. Конечно, свое место есть и у того и у другого, но результаты моделирования по большей части говорят вам об одной очень ограниченной ситуации, в то время как анализ иногда может сказать вам о целом классе ситуаций. Но решение задач комбинаторной оптимизации, бесспорно, в такой же степени искусство, как и наука, и есть люди с чудесно отточенной интуицией для построения эвристических алгоритмов решения задач.

К.Ф. На чем будут сосредоточены исследования в Институте математических наук (ИМН)?

Р.К. Рад, что вы меня спросили об ИМН, потому что этот проект мне очень близок. Это исследовательский институт в горах за университетским городком Беркли, но официально он не связан с университетом, и там финансируются годичные исследовательские программы в математических науках. В прошлом эти программы в основном были по чистой математике. Средства выделяет в первую очередь Национальный научный фонд (ННФ).

Около двух лет тому назад мы со Стивеном Смейлом из математического факультета в Беркли предложили годичный проект по вычислительной сложности, и мы были очень довольны, что его приняли. По-моему, это указывает на то, что математики, которые не торопились заниматься вопросами вычислительной сложности, сейчас стали к ней очень восприимчивы. В этой исследовательской работе по теории сложности будут участвовать около 70 специалистов. Это число поровну поделено между математиками и специалистами по информатике.

Я очень горжусь группой, которую мы собрали. Люди занимаются широким спектром задач. Некоторые — метатеорией, сосредотачиваясь не на конкретных задачах, а на классах вроде P и NP. Некоторые — вычислительной теории чисел, где центральная проблема — разложение очень больших чисел на множители. Другие сосредоточились на комбинаторных задачах. Мы исследуем пограничные вопросы между численным анализом и теорией сложности. И важная тема — параллельные вычисления. Я просто в восторге от того, как идут дела — там прямо-таки Мекка для специалистов по теории сложности. Всего за пару месяцев, пока мы работаем, уже кое-что получено для параллельных алгоритмов.

К.Ф. Нельзя ли поконкретнее?

Р.К. Приводить конкретные результаты было бы преждевременно; скажу лишь, что некоторые из них заставляют воспринимать мои ранние работы как устаревшие.

К.Ф. Сколько денег выделено на этот проект ИМН?

Р.К. Бюджет проекта по сложности в ИМН составляет ок-

ругленно 500 тыс. долл. от Национального научного фонда и 140 тыс. долл. от военных ведомств.

Этот проект для теории сложности оказался своего рода неожиданным подарком судьбы, но я очень отчетливо чувствую: что в целом информатика давно не финансировалась так плохо. Национальный научный фонд поддерживает несколько очень достойных новых инициатив, но это делается без соответствующего расширения финансовой основы, так что эти инициативы субсидируются за счет уже существующих программ. Хотя я должен сказать, что программа ИМН является исключением, в целом люди, занимающиеся теоретической информатикой, потеснены уменьшением финансирования вследствие смещения приоритетов ННФ — в основном в сторону инженерных исследований.

К.Ф. В чем заключается практический интерес Министерства обороны и трех этих ведомств?

Р.К. Поддержка, которая исходит от них, касается в основном параллельных и распределенных вычислений, и мы планируем провести весной рабочее совещание, которое соберет вместе математиков, специалистов по численному анализу и компьютерной архитектуре. Конечно, в наши дни бывают всевозможные мероприятия по суперкомпьютерам и параллельным вычислениям, а это должно способствовать более тесному взаимодействию между теорией сложности и более реальными проблемами пользователей и конструкторов.

К.Ф. Было много дебатов о достоинствах Стратегической оборонной инициативы (СОИ). Не хотели бы вы их прокомментировать?

Р.К. Я не намерен ни выступать на тему о звездных войнах, ни претендовать на то, чтобы быть экспертом по построению программ. Но я изучил некоторые данные, свидетельствующие о том, что опасно строить распределенную систему беспрецедентных масштабов, которая не может работать или быть проверенной до критического момента. Я убежден этими аргументами, и лично я в этом участвовать не буду.

К.Ф. Изучением сложности заняты исследователи во многих областях. Не могли бы вы проиллюстрировать, если возможно, отношения между изучением сложности в информатике и в других дисциплинах?

Р.К. Сложность означает много разных вещей — существует дескриптивная сложность и вычислительная сложность. Алгоритм может быть чрезвычайно сложным в смысле способа его построения и при этом работать очень быстро, так что его вычислительная сложность низка. Таким образом, мы имеем различные понятия о сложности. Мне неясно, имеют ли в виду одно и тоже понятие инженеры-электронщики, экономисты,

математики, специалисты по информатике и физики, употребляя термин сложность. Однако я думаю, что имеется несколько стоящих и интересных аналогий между вопросами сложности в информатике и экономике. Например, в экономике традиционно предполагается, что субъекты экономики имеют универсальную вычислительную мощность и мгновенно узнают, что происходит в остальной экономике. Специалисты по информатике отрицают, что алгоритм может иметь бесконечную вычислительную мощность. Они фактически изучают ограничения, возникающие из-за вычислительной сложности. Таким образом. здесь явная связь с экономикой.

Кроме того, можно сказать, что традиционная экономика — здесь я в самом деле выхожу за рамки своей специальности — не учитывает задержку информации, а также то, что мы принимаем решения, не располагая полной информацией об экономических возможностях, открытых для нас. Это очень похоже на то, что имеет место в сетях связи: узел в распределенной компьютерной сети видит только свое непосредственное окружение и посылаемые ему сообщения. Таким образом, аналогия неоспорима, но надо быть осторожным, потому что мы не всегда имеем в виду одно и то же, когда рассуждаем о сложности.

К.Ф. Используете ли вы понятие «эвристика» иначе, чем делают это исследователи ИИ?

Р.К. Специалисты по ИИ различают алгоритмы и эвристики. Я думаю, что это все — алгоритмы. Для меня алгоритм — это просто любая процедура, которая может быть выражена на языке программирования. Эвристики — это всего лишь алгоритмы, которые мы понимаем не слишком хорошо. Я стремлюсь обитать в искусственно точном мире, где я точно знаю, что мой алгоритм должен делать. Ну а когда вы говорите о программе, которая должна хорошо играть в шахматы, переводить с русского на английский или решать, что заказать в ресторане, — я упомянул несколько задач с привкусом ИИ, — ясно, что требования к ней гораздо расплывчатее. Это характерно для тех программ, которые в ИИ считаются эвристическими.

К.Ф. Дэвид Парнас указывает в последней статье, что системы, разработанные в рамках эвристического программирования, т. е. программирования путем проб и ошибок в отсутствие точной спецификации, в принципе менее надежны, чем программы, созданные более формальными методами.

Р.К. Да, и это возвращает нас к СОИ. Это одна из причин относиться к ней с тревогой. Я думаю, что мы располагаем гораздо более сильным аппаратом отладки программ, когда мы по крайней мере определили, что программа должна делать.

К.Ф. Некоторые члены сообщества исследователей ИИ отве-

чают на эту критику утверждением, что они пытаются моделировать человека и что люди не имеют четких спецификаций. Самое лучшее, на что они могут надеяться, — это моделирования ненадежной системы.

Р.К. Я в самом деле верю в попытки, основанные на жестких гипотезах и жестких заключениях. Я понимаю, что в некоторых областях компьютерной науки должны преобладать эмпирические исследования, но это не освобождает нас от ответственности серьезно думать о том, что мы измеряем, чего мы пытаемся достичь и когда мы можем сказать, что наша конструкция оказалась успешной. И я думаю, что в некоторой мере необходим научный метод. Для меня неприемлема идея, что просто потому, что вы моделируете непознаваемый до конца процесс человеческого познания, вы освобождены от обязательств точно формулировать, что вы делаете.

К.Ф. Что, по вашему мнению, вообще представляет собой информатика как дисциплина?

Р.К. Информатика имеет огромные преимущества по причине колоссальной важности и привлекательности этой области сегодня. По определенным меркам мы достигли немалых успехов. Значительная часть талантливой молодежи страны тянется в нашу область, особенно в направления искусственного интеллекта и теоретической информатики.

Если же говорить о развитии нашей области как науки, то мне кажется, что мы являемся в некоторой мере жертвами собственных успехов. Существует столько способов заработать деньги, так много вещей, которые хочется попробовать, так много увлекательных направлений, что мы иногда забываем подумать об основаниях нашей дисциплины. Нам нужно постоянно поддерживать равновесие между состоянием, когда мы даем волю своим порывам к изготовлению всевозможных искусных поделок, и планированием наших экспериментов научными методами, гарантирующими нормальное развитие фундаментальных исследований. Инструменты наши столь мощны, перспективы столь велики, диапазон возможных приложений столь огромен, что есть большой соблазн пахать все дальше. И пахать надо. Но надо также помнить, что мы — научная дисциплина, а не просто ветвь высокой технологии.

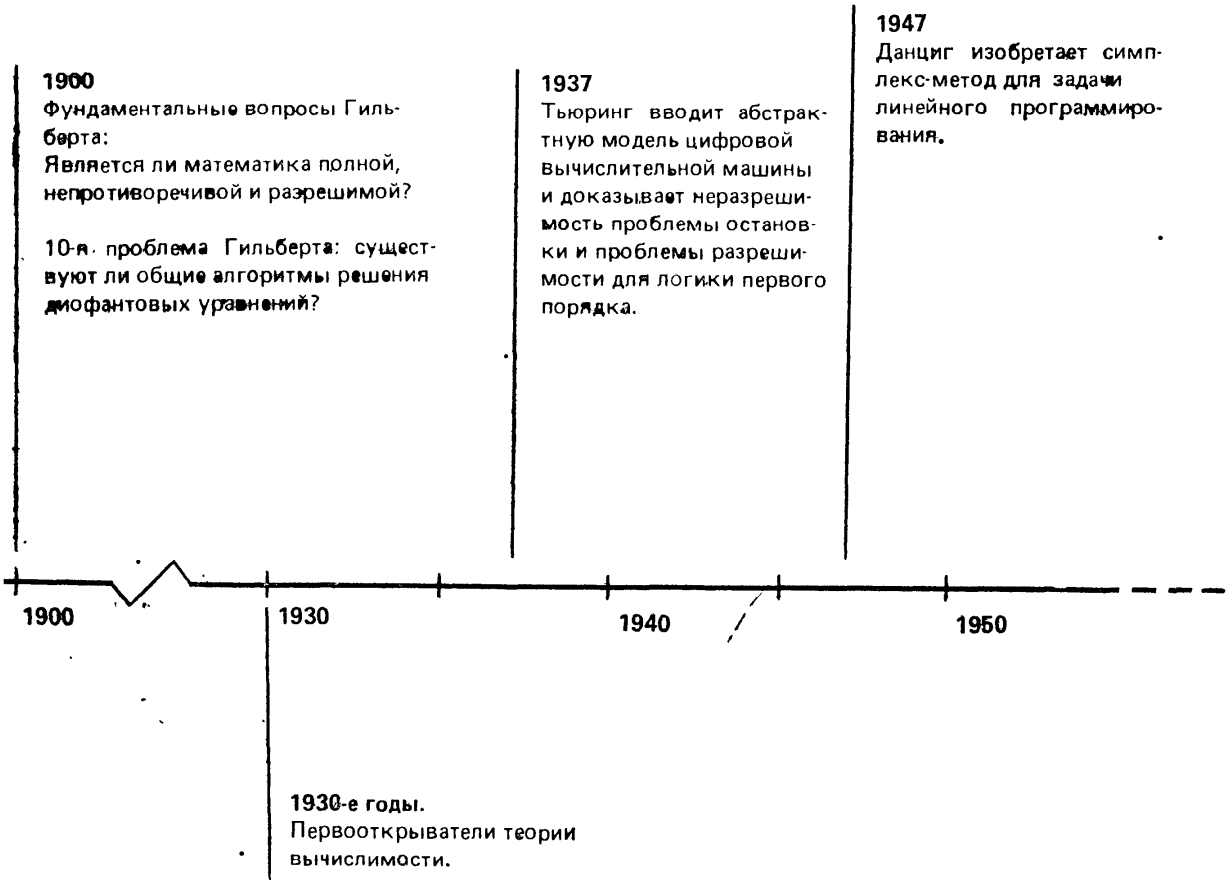
К.Ф. Вы обратили внимание на важность сочетания искусства и науки, проницательности и интуиции, равно как и более строгих методов исследования. Бывало ли, что что-то вам просто являлось и вы испытывали так называемый феномен «эврика!», о котором говорят изобретатели?

Р.К. Я думаю, что мы все это испытывали — утром просыпаешься с готовым решением задачи. Надо помнить, что этим «эврикам» обычно предшествует большой тяжелый труд, кото-

рый иногда кажется непродуктивным. Например, когда я прочел статью Кука 1971 г., я довольно скоро понял, что он нашел ключ к вещам феноменально важным, и начал работать дальше, пытаясь продемонстрировать размах и значение его результатов. В некотором смысле это было почти мгновенно, но было подготовлено более чем десятилетием работы. По-моему, характерно, что такие моменты, когда устанавливаются связи, наступают после долгого подготовительного периода.

К.Ф. Бывало ли у вас, что вы с кем-нибудь беседуете и от случайного замечания у вас в голове — «щелк!»?

Р.К. О да. Мне бывает очень полезно объяснить, чем я занимаюсь, потому что мои ошибки становятся мне ясны гораздо быстрее. А других я слушаю, потому что в самом деле верю в



пользу расширения своей базы знаний. Это сильно увеличивает вероятность обнаружить впоследствии неожиданные связи.

РАЗВИТИЕ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ И ТЕОРИИ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ

1965
Харманис и Стирнз определяют понятие "сложность", вводя подход к вычислительной сложности посредством абстрактных машин, и получают результаты о структуре классов сложности.

Эдмондс определяет "хороший" алгоритм как такой, время работы которого ограничено полиномиальной функцией от размера входа. Находит такой алгоритм для проблемы марьяжа.

1957
Форд и Фалкерсон с сотр. предложили эффективный алгоритм для проблемы потоков в сетях.

1970-е годы
Поиск почти оптимальных решений, основанный на верхней оценке отношения стоимости данного решения к стоимости оптимального решения.

1971
На основе работы Дэвиса, Робинсона и Путмана Матиясевич решает 10-ю проблему Гильберта: не существует общей процедуры решения диофантовых уравнений.

Теорема Кука: все NP-полные проблемы полиномиально сводимы к проблеме выполнимости

Левин также открывает этот принцип.

≈ 1980
Боргвардт, Смейл и др. выполняют вероятностный анализ симплекс-метода

1984
Кармаркар изобретает теоретически эффективный и практически удобный алгоритм линейного программирования.

1960 1970 1980

1959
Рабин, Макнотон и Ямада: первые проблески теории вычислительной сложности

1976
Рабин и др. приступают к изучению рандомизированных алгоритмов

1975
Карп отходит от парадигмы изучения худшего случая и разрабатывает вероятностный анализ комбинаторных алгоритмов.

1973
Мейер, Стокмейер и др. доказывают практическую неразрешимость некоторых проблем разрешения в логике и теории автоматов.

1972
Карп использует полиномиальную сводимость, чтобы показать, что 21 проблема (упаковки, паросочетания, замощения и др.) NP-полные.