Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Кафедра компьютерных технологий

М. Э. Дворкин

Методы автоматизации доказательства NP-полноты двумерных локально зависимых задач

Дипломная работа

Научный руководитель: Г. А. Корнеев

Санкт-Петербург
2008

Содержание

Введение

В настоящее время в рамках теории вычислительной сложности — активно развивающейся области информатики — появляются новые результаты, касающиеся NP-полноты задач, относящихся к различным разделам информатики и математики. Доказательство NP-полноты некоторой задачи является значительным результатом, поскольку оно полностью определяет место данной задачи в иерархии классов сложности.

В общем случае доказательство NP-трудности некоторой задачи является в определенном смысле отрицательным результатом — он показывает, что данная задача «достаточно сложна». В частности, если класс сложности P не равен классу сложности NP (один из важнейших открытых вопросов математики), то для NP-трудной задачи не существует полиномиального решения.

Однако бывают ситуации, в которых доказательство NP-трудности задачи весьма желательно. Примером является исследование игр для одного игрока (головоломок). Если некоторая головоломка оказывается NP-трудной, то существуют «сколь угодно сложные» экземпляры этой головоломки. Соответственно, человек, разгадывающий эту головоломку, обязан использовать при решении достаточно сложные цепочки рассуждений, в чем и заключается эстетическое удовольствие от решения головоломок. Д. Эппштейн, профессор Кафедры информатики Калифорнийского университета (Computer Science Department, University of California) пишет: «Есть любопытная связь между вычислительной сложностью и качеством головоломки. На мой взгляд, лучшие головоломки — NP-полные» [10].

Многие головоломки являются задачами на прямоугольном клетчатом поле. Для доказательства NP-трудности задач на клетчатом поле распространен следующий подход: «на языке задачи» выражается булева формула, и тем самым задача выполнимости булевой формулы сводится к

задаче на клетчатом поле. При этом для выражения логических элементов используются «устройства» — части поля, несущие некоторую логическую функциональность. Доказательство NP-трудности задачи в таком случае заключается в построении достаточного набора устройств. Зачастую это крайне сложная математическая задача. Так, Р. Хёрн из Лаборатории информатики и искусственного интеллекта Массачусетского технологического института (MIT Computer Science and Artificial Intelligence Laboratory) пишет: «Иногда устройства могут быть чрезвычайно сложны, и для того, чтобы их придумать, нужна удивительная изобретательность» [13].

В настоящей работе предложен *новый подход к доказательству NP-полноты задач на клетчатом поле — автоматическое построение достаточного набора устройств.* При таком подходе на порядок уменьшается необходимость применения естественного интеллекта и математической интуиции. Конструкции, специфические для каждой задачи, которые можно построить вручную, лишь хорошо понимая суть, структуру и «выразительные возможности» конкретной задачи, предлагается находить автоматически с помощью предложенного в настоящей работе алгоритма.

Излагаемый подход может позволить сэкономить время, затрачиваемое на доказательство NP-полноты каждой конкретной задачи, а также повысить надежность (вероятность успеха), поскольку работа предложенного алгоритма на компьютере — детерминированный процесс; этого нельзя с уверенностью сказать про мыслительный процесс человека, доказывающего то же утверждение вручную.

Глава 1. NP-полнота и двумерные локально зависимые задачи

В данной главе приведен обзор предметной области.

В разд. 1.1 описан класс задач на клетчатом поле с локальными условиями.

В разд. 1.2 определено понятие NP-полноты и преведены примеры NP-полных задач, в том числе задач на клетчатом поле.

В разд. 1.3 описан метод доказательства NP-полноты двумерных задач, заключающийся в применении устройств и проводов.

В разд. 1.4 описаны алгоритмы динамичесного программирования по профилю, на которых базируется предложенный в настоящей работе подход.

В разд. 1.5 приведена постановка задачи автоматизации доказательства NP-полноты двумерных локально зависимых задач.

1.1. Двумерные локально зависимые задачи

В данном разделе формализован класс задач, к которым применим описываемый в настоящей работе метод, и приведены примеры таких задач.

1.1.1. Определение

Рассмотрим шестерку (A, B, a_0, b_0, s, V), задающую следующие объекты:

- A — конечное множество типов клеток;
- B — конечное множество значений клеток;
- $a_0 \in A$ — тип клетки «пустая клетка»;
- $b_0 \in B$ — значение клетки «пустая клетка»;

- $s \in \mathbb{Z}_+$ — линейный размер квадрата, которого хватает для описания формулировки задачи;
- $V \subset (A \times B)^{s^2}$ — множество допустимых квадратов размера $s \times s$, причем квадрат, состоящий из s^2 элементов (a_0, b_0), принадлежит V.

Определение 1.1. *Полем* называется прямоугольник, каждая клетка которого — элемент множества A.

Определение 1.2. *Заполнением* G поля F называется прямоугольник того же размера, что и F, каждая клетка которого — элемент множества B, такой что для F и G выполнено следующее утверждение:
«Условие корректного заполнения». Рассмотрим любой квадрат $s \times s$, имеющий общие клетки с полем F, а именно, либо лежащий целиком внутри F, либо частично лежащий внутри и частично выходящий за границу. В прямоугольнике G рассмотрим квадрат $s \times s$, расположенный в такой же позиции. Скомбинируем их, получив квадрат R, в котором каждая клетка — это пара из типа соответствующей клетки в поле F и значения соответствующей клетки в прямоугольнике G. Для клеток же, расположенных снаружи F и G, положим в R значение (a_0, b_0). Таким образом, получим квадрат из $s \cdot s$ элементов множества $A \times B$. Этот квадрат должен принадлежать V.

Определение 1.3. *Двумерная локально зависимая задача* (2-Dimensional Locally Dependent, 2-DLD), заданная шестеркой (A, B, a_0, b_0, s, V), формулируется следующим образом: дано поле, требуется определить, существует ли его заполнение.

В разд. 2.1 будет доказано, что класс задач 2-DLD вложен в класс задач NP.

1.1.2. Примеры

Определение 1.4. *Задача замощения* фиксированным набором полимино формулируется следующим образом:

Параметр. Конечное множество полимино с заданной ориентацией (полимино нельзя поворачивать и переворачивать).

Вход. Прямоугольник, клетки которого покрашены в два цвета — белый и черный.

Вопрос. Можно ли замостить все белые клетки данного прямоугольника (без наложений и выходов за границу прямоугольника) данными полимино, если каждое полимино можно использовать произвольное число раз?

Заметим, что набор полимино является параметром, а не входом задачи, таким образом, задач, заданных определением 1.4, бесконечно много — столько же, сколько конечных множеств полимино.

Также заметим, что запрет поворота полимино лишь расширяет рассматриваемое семейство задач. Действительно, если в набор полимино добавить все возможные повернутые копии всех присутствующих там полимино, тем самым становятся разрешенными повороты. Аналогично, при добавлении к имеющимся полимино их перевернутых копий, становятся разрешенными отражения.

Утверждение 1.5. *Задача замощения принадлежит 2-DLD (является двумерной локально зависимой задачей).*

Доказательство. Приведем шестерку (A, B, a_0, b_0, s, V), задающую данную задачу замощения.

Множество типов клеток $A = \{a_0, a_1\}$ состоит из типа a_0 — «черная клетка» и типа a_1 — «белая клетка».

Множество значений клеток B содержит значение b_0 — «черная клетка», а также для каждого полимино из имеющегося набора столько элементов, сколько клеток в этом полимино, — по одному элементу на клетку. Например, для T-тетрамино () создаются четыре элемента множества B: «левая клетка T-тетрамино», «центральная клетка T-тетрамино», «нижняя клетка T-тетрамино» и «правая клетка T-тетрамино».

Наконец, положим $s = 2$, и опишем множество допустимых квадратов 2×2. Для этого приведем список правил, при невыполнении хотя бы

одного из которых квадрат 2×2 становится запрещенным (а при выполнении всех — разрешенным):

- Клетке типа a_0 («черная клетка») может соответствовать только значение b_0.
- Клетке типа a_1 («белая клетка») не может соответствовать значение b_0.
- Пусть в некотором полимино из имеющегося набора есть две соседние по горизонтали клетки, которым соответствуют в множестве B значения b_i (для левой клетки) и b_j (для правой). Тогда в квадрате 2×2 в левой половине может стоять значение b_i тогда и только тогда, когда в соответствующей клетке в правой половине стоит значение b_j.
- Аналогичное правило для пар соседних по вертикали клеток в имеющихся полимино.

Благодаря первым двум правилам, для произвольного региона устанавливается, что полимино расположены на всех белых клетках и только на них, а благодаря последним двум — что рассматриваемое заполнение является корректным замощением региона данным набором полимино. □

Также в настоящей работе исследуется задача N-CROSS SUM, являющая обобщением популярной головоломки «какуро» [17]. Опишем ее правила.

Определение 1.6. Задача N-CROSS SUM формулируется следующим образом:

Параметр. Целое положительное число N — максимальное допустимое значение в клетке.

Вход. Прямоугольное поле, состоящее из белых и черных клеток со следующими условиями и ограничениями:

- Ряд из белых клеток идущих подряд по горизонтали или по вертикали, ограниченный по краям черными клетками и/или краями поля, называется *группой*.

- Каждая группа состоит не менее чем из двух клеток.
- Для каждой группы задано положительное целое число, называемое *подсказкой*.

Вопрос. Можно ли записать в белые клетки положительные целые числа так, чтобы выполнялись следующие условия:

- В каждую белую клетку записано число от 1 до N.
- В каждой группе сумма чисел, записанных в клетки этой группы, равняется подсказке, данной для этой группы.
- В каждой группе все числа различны.

Некоторые источники [16] добавляют условие, что множество белых клеток должно быть связно, однако там же отмечено, что это условие не имеет большого значения, так как задачу с несвязным набором белых клеток можно рассматривать как несколько независимых задач со связными наборами. В настоящей работе условие связности не накладывается.

Головоломка какуро есть не что иное как экземпляр задачи 9-CROSS SUM.

Пример 1.7. На рис. 1.1 приведен экземпляр задачи 9-CROSS SUM. Здесь и далее используется стандартная нотация [17]: число в правой части черной клетки является подсказкой для горизонтальной группы клеток, расположенной справа от нее, а число в нижней части — для вертикальной группы клеток снизу от нее.

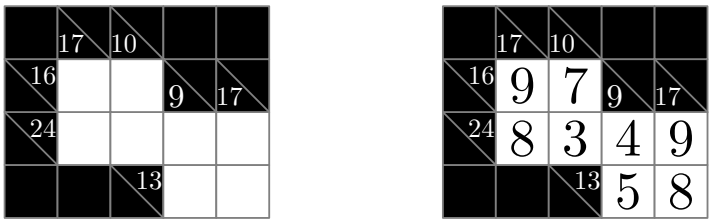


Рис. 1.1. Экземпляр задачи 9-CROSS SUM и его решение

1.2. NP-полные задачи

Понятие NP-полноты было введено С. Куком. Класс NP-полных задач весьма обширен и включает в себя важные задачи из различных разделов математики, информатики и даже других наук [9]. Важной особенностью задач этого класса является то, что все они в некотором смысле «одинаково сложные». Так, если хотя бы одна NP-полная задача имеет полиномиальное решение, то полиномиальное решение имеют и все остальные NP-полные задачи (как и вообще все задачи из класса NP). Впрочем, большинство специалистов полагают, что ни одна NP-полная задача не имеет полиномиального решения, то есть $\mathrm{P} \neq \mathrm{NP}$ [4].

1.2.1. Определение и примеры

Приведем определение класса задач NP и NP-полных задач. Здесь и далее под словом «задача» будем подразумевать «задачу решения» (decision problem) — задачу, подразумевающую ответ «да» или «нет».

Определение 1.8. Задача принадлежит классу NP, если существует недетерминированная машина Тьюринга, решающую эту задачу за полиномиальное время.

Определение 1.9. Задача L называется NP-полной, если

- L принадлежит NP.
- Для всякой задачи L' из NP существует полиномиальное сведение L' к L.

Подробное изложение основ теории вычислительной сложности, в том числе определение машины Тьюринга и основных классов задач, имеется в книге [6].

Определение 1.10. Задача 1-in-3 SAT формулируется следующим образом:

Вход. Конъюнкция предикатов вида $R(L_1, L_2, L_3)$, где L_1, L_1 и L_3 — литералы, а R — предикат, означающий «Из этих трех значений ровно одно —

истина». R можно выразить формально:

$$R(A, B, C) = (A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge C)$$

Вопрос. Можно ли присвоить переменным булевы значения так, чтобы данная конъюнкция была истина?

Например, нижеприведенный экземпляр задачи 1-in-3 SAT имеет ответ «да» (поскольку существует, например, следующее решение: $A = B = true$, $C = D = false$):

$$R(A, \neg B, C) \wedge R(B, C, D) \wedge R(\neg A, \neg B, \neg C)$$

В работе [15] показано, что 1-in-3 SAT является NP-полной задачей. В настоящей работе этот факт активно используется.

Определение 1.11. Задача о раскраске графа в k цветов (k-COLOR [4]) формулируется следующим образом:
Параметр. Целое положительное число k.
Вход. Граф G.
Вопрос. Можно ли раскрасить вершины графа G в k цветов так, чтобы каждое ребро соединяло две вершины разного цвета?

Данная задача является NP-полной при $k \geq 3$ [4].

1.2.2. Примеры NP-полных двумерных локально зависимых задач

В работе [16] показано, что N-CROSS SUM является NP-полной при $N \geq 7$. Вопрос о принадлежности классу P и о NP-полноте задачи N-CROSS SUM для $N \in [3, 6]$ являлся открытым [16, 18]. В настоящей работе часть этой задачи решена — в разделе 4.2 будет доказана NP-полнота задачи N-CROSS SUM для $N \geq 4$.

Определение 1.12. Задачей (p, q)-замощения будем называть задачу замощения (определение 1.4) для следующего набора полимино: горизонтально расположенная полоса $1 \times p$ и вертикально расположенная полоса $q \times 1$.

Не умаляя общности, будем в дальнейшем полагать $p \leq q$. В работе [8] показано, что при $p \geq 2$ и $q \geq 3$ задача (p, q)-замощения является NP-полной.

1.2.3. Пример не NP-полной двумерной локально зависимой задачи

Пример 1.13. *Замощение домино* — то же самое, что $(2, 2)$-замощение, то есть задача замощения с параметром, состоящим из двух домино:  и  .

Задача замощения домино имеет полиномиальное решение. Для замощения данного региона следует рассмотреть граф, вершинами которого являются белые клетки данного региона, а ребро соединяет две клетки, соседние по стороне. Этот граф является двудольным: если раскрасить плоскость в шахматном порядке, то клетки одного цвета являются одной долей, а клетки другого — другой. Замощению региона домино в рассмотренном графе соответствует такой набор ребер, что каждая вершина принадлежит ровно одному ребру, то есть полное паросочетание. Поскольку поиск максимального паросочетания в двудольном графе осуществим за полиномиальное время [4], то и задача замощения домино принадлежит P.

1.3. Применение устройств для доказательства NP-полноты

Как следует из определения, для доказательства NP-полноты задачи достаточно доказать ее принадлежность классу задач NP и ее NP-трудность. Наиболее естественный и часто применяемый способ доказательства NP-трудности задачи — полиномиальное сведение к этой задаче известной NP-полной задачи.

Весьма распространено в качестве сводимой NP-полной задачи использовать задачу выполнимости булевой формулы (SAT) или одну из ее модификаций. В этом случае по исходной формуле строится экземпляр рассматриваемой задачи, эквивалентный формуле, то есть дающий ответ

«да» тогда и только тогда, когда формула выполнима. Для построения этого экземпляра требуется «на языке рассматриваемой задачи» выразить переменные, встречающиеся в формуле, и логические операторы, связывающие их друг с другом. Это делается с помощью *устройств* (gadgets), которые позволяют «перевести» исходную булеву формулу на язык рассматриваемой задачи. Устройства соединяются между собой *проводами*, позволяющими передавать информацию от одного устройства к другому.

В данном разделе описана общая схема сведения задачи выполнимости к произвольной задаче, и приведены примеры наборов устройств, позволяющие это сделать.

1.3.1. Общая схема сведения

Если каждая переменная входит в формулу только один раз, то формула, как легко понять, выполнима (такие формулы называются бесповторные). Поэтому, имеет смысл рассматривать повторные формулы. Соответственно, после создания переменной необходимо создать несколько ее экземпляров, то есть несколько проводов, передающих одинаковое значение — значение этой переменной. Наиболее простое устройство, позволяющее этого добиться — устройство «раздвоение провода» (splitter, fork), которое из одного провода делает два, передающих такое же значение. Применив это устройство несколько раз, можно добиться любого необходимого числа экземпляров каждой переменной.

Теперь необходимый набор экземпляров переменных создан, но экземпляры одной переменной расположены на плоскости рядом друг с другом. Для описания же искомой формулы может потребоваться заметно изменить порядок расположения экземпляров на плоскости. Этого легко добиться, если два провода, передающих произвольную информацию, можно перекрестить. Устройство, делающее это, называется «перекрещивание проводов» (crossover). С его помощью можно, например, произвольное число идущих параллельно проводов расположить в требуемом порядке.

Затем к проводам, расположенным на плоскости в соответствии с позициями переменных в формуле, (а точнее не к проводам, а к значениям, передаваемым проводами) следует применить логические операции, из которых складывается формула. Например, для этого достаточно реализовать на языке рассматриваемой задачи полный набор булевых функций [2].

Наконец, требуется проверить, что значение формулы истинно. Для этого можно, например, использовать устройство «валидатор истинности», которое, будучи применено к проводу, передающему итоговое значение формулы, гарантирует, что все поле в целом имеет заполнение тогда и только тогда, когда по этому проводу передается значение `истина`, то есть когда набор булевых значений, использованный при создании переменных, делает формулу истинной, а это и означает, что формула выполнима.

Здесь, впрочем, есть место для упрощения конструкции. Так, если формула является конъюнкцией нескольких предикатов, то для проверки ее истиности достаточно проверить истиность каждого из предикатов (например, с помощью устройства «валидатор истинности»). В этом случае не требуется собирать воедино провода, передающие значения, соответствующие этим предикатам.

1.3.2. Примеры наборов устройств

Для доказательств NP-полноты задач Minesweeper (Сапер, [14]), TipOver [13] и Cubic [12] был применен следующий набор устройств:

- «начало/конец провода»;
- «раздвоение провода»;
- «перекрещивание проводов»;
- «логическое И»;
- «логическое ИЛИ»;
- «логическое НЕ».

Следует отметить, что к задачам на клетчатом поле сводят не только задачу выполнимости и ее модификации. Так в работе [11] к задаче

Corral Puzzles сводится задача раскраски планарного графа в три цвета (определение 1.11), при этом используются следующие провода и устройства:

- провод, передающий одно из трех значений («красный», «зеленый», «синий»);
- «создание провода»;
- «раздвоение провода»;
- «стыковка проводов, возможная лишь при различных передаваемых значениях».

1.4. Динамическое программирование по профилю

Метод динамического программирования был предложен Р. Беллманом в 1955 году. Суть этого подхода состоим в том, чтобы вместе с задачей, которую требуется решить, также решить некоторое множество ее подзадач [4]. При этом для решения одних подзадач требуется знать решение других, поэтому важен порядок решения подзадач.

1.4.1. Динамическое программирование по прямому профилю

Динамическим программированием по профилю называют динамическое программирование на графе, в котором каждая вершина представляет собой некоторое состояние (профиль) [5]. Для того, чтобы отличать понятие профиля от понятия изломанного профиля (разд. 1.4.2), будем называть рассматриваемое понятие *прямым профилем*.

В двумерных задачах на клетчатом поле чаще всего прямой профиль представляет собой срез поля. Нестрого говоря, прямой профиль в таком случае — это вся необходимая информация для дальнейшей обработки поля, если обработано несколько первых столбцов. (Здесь и далее будем, не умаляя общности, рассматривать вертикальные профили).

Пример 1.14. Рассмотрим задачу о *расстановке королей*. Она формулируется так: сколькими способами можно расставить на поле $h \times w$ несколько шахматных королей так, чтобы они не били друг друга?

Прямым профилем в данной задаче — то есть информацией, необходимой для дальнейшей обработки клеток правее некоторого вертикального среза — являются данные, какие клетки следующего столбца побиты королями, стоящими в предыдущем столбце (рис. 1.2, побитые клетки окрашены в серый цвет). Таким образом, профиль состоит из h бит, где h — высота поля. Соответственно, число профилей не превышает 2^h. Это — оценка сверху, так как некоторые профили противоречивы (например, если клетка побита, то должна быть побита также одна из двух соседних с ней клеток).

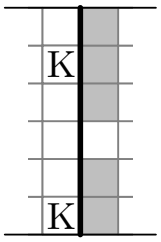


Рис. 1.2. Прямой профиль в задаче о расстановке королей

Для перехода от одного слоя к другому (то есть обработки одного столбца) требуется знать, сколькими способами можно перейти от каждого профиля предыдущего слоя к каждому профилю следующего. Эта информация хранится в *таблице переходов*.

Обычно таблица переходов требует много памяти для хранения и много времени для построения. В следующем разделе будет указан подход, позволяющих улучшить время работы и объем хранимой информации.

1.4.2. Динамическое программирование по изломанному профилю

Изломанный профиль — обобщение прямого профиля на случай, когда обработанным является не целое число столбцов, а некоторое число столбцов и несколько первых клеток следующего столбца [1]. Не умаляя

общности далее в тексте и в иллюстрациях будем считать, что внутри одного столбца клетки перебираются в порядке снизу вверх.

Рассмотрим, что является изломанным профилем в задаче о расстановке королей. После обработки нескольких столбцов и нескольких клеток следующего столбца для дальнейшей работы нужна следующая информация: какие из клеток, имеющих общие точки с границей среза, побиты уже поставленными королями (рис. 1.3, побитые клетки окрашены в серый цвет).

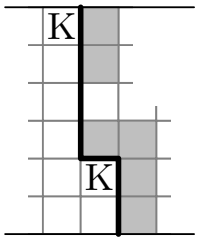


Рис. 1.3. Изломанный профиль в задаче о расстановке королей

Для хранения изломанного профиля требуется $h+1$ бит (или меньше, если позиция излома — $y = 0$, то есть профиль «не изломан»). Кроме того, в изломанный профиль, строго говоря, также входит позиция излома, которая лежит в промежутке $[0, h)$. Итого имеется не более $h \cdot 2^{h+1}$ изломанных профилей.

Основным достоинством этого подхода является вычислительная простота перехода от одного изломанного профиля к следующему. Так, в задаче о расстановке королей переход состоит в следующем: рассматривается очередная клетка (нижняя из еще необработанных) текущего столбца, и для нее рассматриваются два варианта — либо поставить туда короля (если это возможно), либо не поставить. В обоих случаях изломанный профиль следующего слоя формируется достаточно несложно.

1.5. Постановка задачи

Целью настоящей работы является создание метода автоматизации доказательства NP-полноты двумерных локально зависимых задач. Для

этого требуется решить следующие задачи:

- формализация класса рассматриваемых задач;
- доказательство принадлежности рассматриваемых задач классу NP;
- построение набора устройств, достаточного для доказательства NP-полноты задачи, и доказательство его достаточности;
- формализация правил композиции устройств;
- *разработка метода автоматического построения устройства* заданного размера с заданной функциональностью;
- анализ асимптотики времени работы предложенного алгоритма.

Для апробации разработанного подхода следует решить следующие задачи:

- программная реализация разработанного алгоритма;
- применение программы для доказательства NP-полноты конкретных задач;
- оптимизация программы с целью уменьшения времени работы и требуемой памяти.

Глава 2. Наборы устройств и правила композиции

В этой главе описаны наборы устройств и накладываемые на них условия (в плане формы, размера и возможности стыковки друг с другом), достаточные для доказательства NP-полноты задачи из класса 2-DLD, и доказано, что любая задача из 2-DLD, для которой построены все устройства из такого набора, является NP-полной.

2.1. Общая схема доказательства NP-полноты

Доказательство NP-полноты двумерной локально зависимой задачи состоит из двух частей — доказательства ее принадлежности классу NP и доказательства ее NP-трудности. Первая часть следует из теоремы 2.1. Вторая часть составляет содержание всего данного раздела.

Теорема 2.1. *2-DLD* $\subset NP$.

Доказательство. Покажем, что для любой задачи L из 2-DLD существует сертификат полиномиального размера и алгоритм, позволяющий за полиномиальное время проверить верность сертификата.

Зафиксируем задачу L из 2-DLD, задаваемую шестеркой (A, B, a_0, b_0, s, V). Рассмотрим некоторый вход задачи L, а именно прямоугольное поле F. Сертификатом является заполнение G поля F. Поскольку множества A и B конечны и фиксированы, размер сертификата G составляет $O(|F|)$.

Теперь покажем, что проверку корректности сертификата можно осуществить за полиномиальное (относительно $|F|$) время. Проверка сертификата заключается в последовательной проверке всех квадратов $s \times s$, имеющих общие клетки с полем F. Если прямоугольник F имеет размеры $m \times n$, то число таких квадратов составляет $(m + s - 1)(n + s - 1) = O(mn) = O(|F|)$, поскольку s — константа.

Итак, имеется $O(|F|)$ квадратов, каждый из которых следует проверить на принадлежность множеству V. Поскольку множество V фиксировано и конечно, и размеры элементов множества V не превосходят некой константы, каждая такая проверка занимает $O(1)$ времени.

Таким образом и размер сертификата, и время работы алгоритма его проверки, полиномиальны относительно размера входа. Следовательно, $L \in NP$. □

Теперь опишем сведение известной NP-полной задачи, в данном случае 1-in-3 SAT, к рассматриваемой двумерной локально зависимой задаче.

Для представления формулы на плоскости используются различные *устройства*, имеющие ту или иную функциональность: создание булевой переменной, применение логических операторов и т. п. Важнейшей частью сведения является то, каким образом булево значение передается от одного устройства к другому. Для этого используются *провода* — куски поля, позволяющие «передавать» булевы значения от одного устройства к другому.

В разделе 2.2 описаны необходимые характеристики проводов и сложности, возникающие при проектировании проводов. В разделе 2.4 описан алгоритм полиномиального сведения 1-in-3 SAT к двумерной локально зависимой задаче, для которой известен достаточный набор устройств.

2.2. Одинарные и двойные провода

Провод призван передавать булево значение от одного устройства к другому. Следует отметить, что слово «передавать» хоть и уместно, но не очень строго, поскольку подразумевает некоторую динамику, в то время как рассматриваемая задача совершенно статична. В рассматриваемом случае то, что провод «передает» значение от первого устройства ко второму, следует понимать так:

- Если первое устройство заполнено так, что проводу передается булево значение `ложь`, то провод можно заполнить так и только так, что второму устройству передается значение `ложь`.

- Аналогично, если провод от первого устройства получает `истина`, то второму устройству передается значение `истина`.

В настоящей работе, за исключением отдельно оговоренных случаев, будем считать, что провод направлен слева направо. К сожалению, определение двумерных локально зависимых задач никоим образом не гарантирует изотропность плоскости, так, например, могут существовать задачи, в которых не существует проводов, идущих слева направо, но существуют провода, идущие снизу вверх. Однако во многих случаях можно ограничиваться именно проводами, направленными вправо.

2.2.1. Одинарные провода

Наиболее простым случаем является горизонтальный провод, шириной в одну клетку поля. В некоторых задачах провод направлен по диагонали и представляет собой «лесенку» из клеток в каком-то смысле также шириной в одну клетку. В дальнейшем такие провода будем называть *одинарными*. Сразу отметим, что отличие одинарного провода от более сложных проводов совершенно неформально и интуитивно. Однако во многих задачах существуют провода, которые естественно назвать одинарными.

Рис. 2.1. Примеры одинарных проводов

Казалось бы, в случае существования простого одинарного провода его достаточно для передачи информации между устройствами. Оказывается, что интересной особенностью некоторых двумерных локально зависимых задач является несуществование важнейшего устройства — «создания провода», даже при том, что сам провод существует.

Устройство «создание провода» — это устройство, из которого выходит один провод, и которое имеет такое заполнение, что провод передает значение `ложь`, и такое заполнение, что провод передает значение `истина`.

Приведем примеры задач, в которых при существующем одинарном проводе не существует такого устройства.

Пример 2.2. Задача (p, p)-замощения с горизонтальным проводом.

Один из наиболее естественных проводов — горизонтальная полоса белых клеток. Ясно, что единственный способ заполнить ее — выложить ее горизонтально ориентированными прямоугольниками $1 \times p$. Соответственно, в зависимости от «фазы», то есть от положения этих горизонтальных полосок (их горизонтальной координаты, взятой по модулю p), можно говорить о проводе, передающим значение `ложь` или `истина`.

Такой провод, сам по себе, конечно, существует, однако предположим, что существует устройство конечного размера, из которого выходит один такой провод:

Рис. 2.2. Гипотетическое устройство «создание одинарного провода»

Посчитаем число белых клеток в этом устройстве по модулю p. Ясно, что эта величина однозначно определяет положение горизонтальных полос внутри провода. Следовательно, такое устройство не позволяет варьировать между передаваемыми значениями `ложь` и `истина`.

Пример 2.3. Задача $(2, 3)$-замощения с горизонтальным проводом.

Снова предположим, что существует устройство «создание одинарного провода», из которого выходит один горизонтальный провод, значение передаваемое по которому можно варьировать. Следовательно, при некотором значении (`ложь` или `истина`) это устройство (отдельно взятое) имеет заполнение, а при другом значении имеет заполнение это устройство плюс одна клетка (первая клетка провода). Однако заметим следующий факт.

Лемма 2.4. *В любом поле задачи* $(2,3)$*-замощения, имеющем заполнение, число белых клеток с четной координатой* x *и число белых клеток с нечетной координатой* x *сравнимы по модулю 3.*

Доказательство. Докажем методом математической индукции по числу белых клеток в поле (обозначим эту величину w).

База. $w = 0$. Число белых клеток как с четной, так и с нечетной координатой x равно нулю, $0 \equiv 0 \pmod 3$.

Переход. Рассмотрим поле с $w > 0$. Выберем любую белую клетку и рассмотрим полимино P, покрывающее ее. Временно покрасим клетки, покрытые P, в черный цвет — получим поле, с меньшим числом белых клеток, чем исходное. По предположению индукции для него выполнено утверждение леммы. Перекрасим обратно в белый цвет клетки, покрытые P. Возможны два варианта:

- P — горизонтальное домино. Тогда к полю добавляются две клетки: одна с четной координатой x, и одна — с нечетной. Число тех и других клеток увеличивается на единицу, и эти два числа все также сравнимы по модулю 3.
- P — вертикальное тримино. Тогда к полю добавляются три клетки с равной координатой x. Из двух рассматриваемых чисел одно не изменяется, а другое увеличивается на три. После этого числа продолжают быть сравнимыми по модулю 3. □

Вернемся к гипотетическому устройству «создание одинарного провода». Как было отмечено, если оно существует, то существуют два заполняемых поля, отличающихся цветом одной клетки. Однако из рассматриваемых в лемме 2.4 двух чисел у этих полей одно число совпадает, а другое отличается на единицу. Обе пары чисел не могут быть сравнимы по модулю 3, следовательно, предположение о существовании устройства неверно.

Пример 2.5. Задача (p,q)-замощения с горизонтальным проводом.

Приведенное в предыдущем примере рассуждение расширяется на случай задачи (p,q)-замощения с произвольными $p, q \geq 2$. В этом случае

расширенное утверждение формулируется так:

Утверждение 2.6. *В любом поле задачи (p, q)-замощения, имеющем заполнение, для всех целых i в интервале $[0, p-1)$ число белых клеток, чья координата x сравнима с i по модулю p, дает один и тот же остаток при делении на q.*

Доказательство этого факта аналогично доказательству леммы 2.4 — ни добавление горизонтальной полосы $1 \times p$, ни добавление вертикальной полосы $q \times 1$ не влияют на истинность приведенного утверждения.

Еще раз предположим, что существует устройство «создание одинарного провода», из которого выходит один горизонтальный провод, причем положение горизонтальных полос, заполняющих его, можно варьировать. Тогда существует заполнение для отдельно взятого этого устройства, и существует заполнение для поля «устройство плюс несколько (от 1 до $p-1$) клеток провода».

Посчитаем для этих полей число белых клеток, с координатой x, сравнимой с i по модулю p. Для некоторых i эти числа у обоих полей совпадают, а для некоторых — отличаются на единицу. Соответственно, все эти числа не могут быть сравнимы друг с другом по модулю q ($q \geq 2$), и устройства «создание одинарного провода» не существует.

Пример 2.7. Задача N-CROSS SUM.

В задаче N-CROSS SUM один из естественных одинарных проводов направлен по диагонали и выглядит следующим образом:

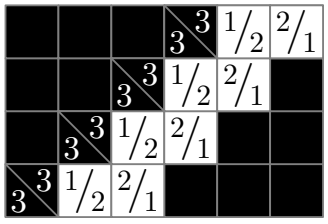


Рис. 2.3. Одинарный провод для задачи N-CROSS SUM

Видно, что присвоив значение одной из клеток этого провода, значения остальных клеток определяются однозначно. Таким образом, у этого провода имеется два заполнения — значения, стоящие выше косой черты в

белых клетках, и значения, стоящие ниже. Одно из этих заполнений соответствует передаваемому значению `ложь`, другое — значению `истина`.

Снова предположим, что существует устройство, создающее такой одинарный провод (рис. 2.4).

Рис. 2.4. Гипотетическое устройство «создание одинарного провода»

При решении головоломок какуро (экземпляров задачи 9-CROSS SUM) используется следующая техника: выбрать некоторый участок и попытаться вычислить сумму значений, которые должны стоять в белых клетках этого участка. Во-первых, это можно сделать, посчитав сумму подсказок по всем столбцам этого участка, а во-вторых, посчитав сумму по строкам. Иногда разница полученных величин является ценной информацией, необходимой для решения головоломки.

Так и в случае с созданием провода, найдем сумму всех подсказок для столбцов и для строк, содержащих клетки рассматриваемого устройства. Разность этих двух величин равна значению клетки A. Таким образом, заполнение провода детерминировано, и варьировать между значениями `ложь` и `истина` невозможно.

2.2.2. Двойные провода

К счастью, существует способ, позволяющий во многих случаях (в частности, в приведенных в предыдущем разделе примерах) бороться с описанной выше проблемой.

Для передачи информации он одного устройства к другому будем использовать *двойные* провода — пары одинарных проводов, расположенных параллельно друг другу.

При таком подходе устройство «создание провода» на выходе имеет два провода, каждый из которых несет информацию. В задачах, рассматриваемых в настоящей работе, более естественным оказывается следующий инвариант: двойной провод состоит из двух одинарных, передающих разные значения (провода «в противофазе»). Таким образом, возможны два варианта:

- Двойной провод передает значение `ложь`: первый одинарный провод передает значение `ложь`, второй — `истина`.
- Двойной провод передает значение `истина`: первый одинарный провод передает значение `истина`, второй — `ложь`.

Возможны и задачи, в которых более уместен двойной провод, состоящий из двух одинарных, передающих одинаковое значение (логично предположить, что это и есть значение, передаваемое двойным проводом). Однако среди примеров, рассматриваемых в настоящей работе, таких задач нет.

В дальнейшем будут рассмотрены и устройства, работающие с двойными проводами, и устройства, работающие с одинарными. Во всех случаях будет оговорено, какие именно провода входят и выходят из устройства.

2.3. Нотация описания устройств

В следующих разделах и главах описываются различные устройства, позволяющие представлять формулы на языке той или иной задачи. Для того, чтобы упростить описание устройств, введем нотацию, позволяющую задавать функциональность устройства.

Данная нотация предназначена для описания логики устройства — то есть возможных наборов значений, передаваемых входящими и исходящими проводами. Следует отметить, что кроме логики устройства важны также его размеры и расположение входящих и исходящих проводов относительно устройства. Данная нотация не отражает эти характеристики.

Нотация устройства состоит из двух частей:

- возможные наборы значений, передаваемых входящими проводами;
- соответствующие им возможные наборы значений, передаваемых исходящими проводами.

Таким образом, обе части состоят из равного числа *ситуаций*. Ситуации разделяются запятыми. Каждая ситуация — это перечисленные через пробел наборы значений проводов. Значения проводов перечисляются от проводов, расположенных выше, к проводам, расположенным ниже. Значение `ложь` обозначается буквой «`f`», значение `истина` — буквой «`t`».

Если ситуация состоит из пустого множества значений проводов (такое имеет место, если устройство «запрещает» некоторую входную ситуацию), она описывается одной буквой «`x`».

Пример 2.8. `[ff,ft,tf,tt][ff,tf,ft,tt]` — устройство, рассматривающее все четыре возможные ситуации и во всех случаях передающее по исходящим проводам те же значения, что переданы по входящим, но в обратном порядке. Таким образом, это устройство — «перекрещивание одинарных проводов».

Пример 2.9. `[ft,tf,ff tt][ft,tf,x]` — устройство, рассматривающее три ситуации: два случая передачи по входящим проводам противоположных значений и обобщенный случай передачи двух одинаковых значений. В первых двух случаях устройство успешно передает информацию далее по исходящим проводам, а в третьем — устройство просто не имеет заполнения. Таким образом, это устройство гарантирует, что два провода передают противоположные значения, — «валидатор противоположности значений». Как будет показано в разд. 2.4.2, это устройство очень полезно для разветвления проводов.

2.4. Сведение к задаче 1-in-3 SAT

В данном разделе приведен алгоритм сведения задачи 1-in-3 SAT к произвольной двумерной локально зависимой задаче L, для которой суще-

ствует соответствующий набор устройств.

Для каждой формулы E — экземпляра задачи 1-in-3 SAT — данный алгоритм за полиномиальное время (относительно размера формулы) строит поле для задачи L, которое имеет заполнение тогда и только тогда, когда исходная формула выполнима.

Поскольку алгоритм работает за полиномиальное время относительно размера формулы, то получаемое поле имеет также полиномиальный размер.

Получаемое поле является прямоугольником, состоящим из *ячеек* одинакового размера. Дополнительный термин вводится сознательно, чтобы не возникало путаницы с клетками: клетка имеет размер 1×1, а ячейка — $h_0 \times w_0$ для некоторых целых h_0 и w_0. Размеры ячейки являются характеристикой сведения и не зависят от E (формулы, для которой строится поле).

Каждое устройство занимает одну или несколько ячеек. Между соседними по горизонтали ячейками может передаваться одно булево значение — из левой ячейки в правую (разд. 2.2). При этом должна быть зафиксирована единая конвенция передачи информации — каким образом передается значение `ложь`, и каким образом передается значение `истина`.

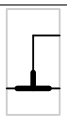
2.4.1. Набор устройств для одинарных проводов

Определение 2.10. Набор устройств $\mathcal{A}_1$ представлен в табл. 2.1.

Следует отметить, что здесь и далее подразумевается существование «пустого устройства» — устройства, занимающего одну ячейку и имеющего пустую функциональность, обозначаемую в нотации из разд. 2.3 `[] []`. Согласно определению двумерной локально зависимой задачи такое устройство есть всегда: достаточно рассмотреть ячейку, состоящую из клеток типа a_0. По этой причине далее оно упоминаться не будет.

Теорема 2.11. *Двумерная локально зависимая задача L, для которой существует набор устройств $\mathcal{A}_1$, является NP-полной.*

Таблица 2.1. Набор устройств $\mathcal{A}_1$

Обозначение	Размер (в ячейках)	Название	Функциональность
	1×1	Провод	`[f,t][f,t]`
	1×1	Создание провода	`[][f t]`
	2×1	Раздвоение провода	`[f,t][ff,tt]`
	2×1	Перекрещивание проводов	`[ff,ft,tf,tt][ff,tf,ft,tt]`
	1×1	Логическое `НЕ`	`[f,t][t,f]`
	3×1	1 из 3	`[fft,ftf,tff,fff ftt tft ttf ttt][,,,x]`

Доказательство. Поле состоит из пяти *слоев*, каждый слой — это один или несколько столбцов ячеек, заключающий в себе ту или иную функциональность. Перечислим все пять слоев слева направо (по направлению «передачи информации»):

1. Для каждой переменной A, встречающейся в формуле E, создается провод, передающий значение `ложь` или `истина`, соответствующее значению переменной A.
2. Для каждой переменной A, встречающейся в формуле E, соответствующий ей провод раздваивается необходимое число раз, для того, чтобы число проводов, соответствующих A, равнялось числу вхождений переменной A в формулу E.
3. Имеющиеся провода перекрещиваются необходимое число раз так, чтобы их порядок соответствовал порядку вхождений переменных в формулу E.
4. К проводам, соответствующим отрицательным литералам в формуле E, применяется логическое `НЕ`.
5. К каждой тройке проводов, соответствующей предикату в форму-

ле E, применяется устройство, гарантирующее, что ровно один из проводов несет значение `истина`.

Пример 2.12. На рис. 2.5 изображено поле, соответствующее формуле $R(A, \neg B, C) \wedge R(B, C, D) \wedge R(\neg A, \neg B, \neg C)$. Третий слой упрощен вручную (для компактности).

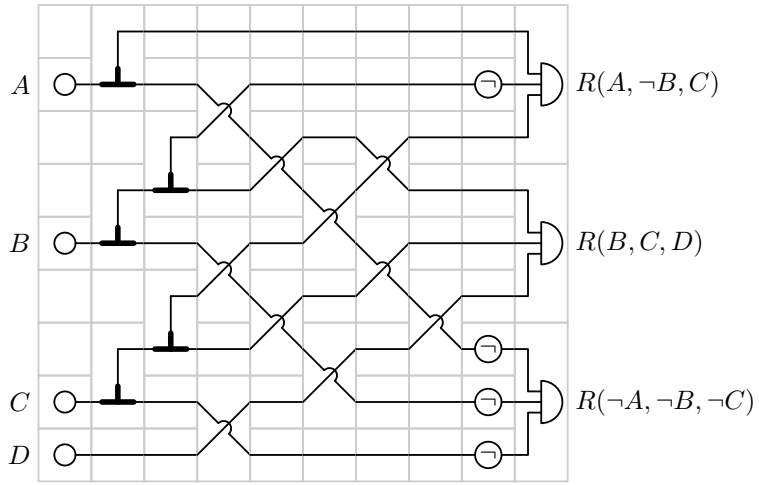


Рис. 2.5. Поле, соответствующее формуле $R(A, \neg B, C) \wedge R(B, C, D) \wedge R(\neg A, \neg B, \neg C)$ (набор устройств $\mathcal{A}_1$)

Высота получаемого поля (в ячейках) равна утроенному числу предикатов в формуле E, то есть составляет $O(|E|)$ — как в ячейках, так и в клетках, поскольку высота ячейки — константа (не зависит от E).

Оценим ширину слоев. Первый, четвертый и пятый слои — шириной в одну ячейку. Во втором слое каждый столбец увеличивает число созданных проводов хотя бы на единицу, всего же создается $O(|E|)$ проводов, следовательно, ширина второго слоя составляет $O(|E|)$ ячеек.

Наконец, в третьем слое требуется переупорядочить созданные провода в соответствии с порядком вхождения переменных в формулу E. Избрав самый наивный способ действия — в каждом столбце изменяя лишь одну пару неправильно расположенных (друг относительно друга) проводов — можно добиться ширины в $O(|E|^2)$ ячеек. Действительно, число инверсий, составляющее вначале $O(|E|^2)$, после каждого столбца уменьшается на единицу. Таким образом, после $O(|E|^2)$ столбцов оно станет равным нулю.

Действуя же более интеллектуальным способом, можно упорядочить провода, используя $O(|E|)$ столбцов. Для этого потребуется аналог *четно-нечетной сортировки* [3]. Для упорядочения n проводов требуется n столбцов (пронумеруем их для удобства от 1 до n):

- Если столбец имеет нечетный номер, рассматриваются следующие пары проводов: (1, 2), (3, 4) и т. д. Для каждой пары, если она упорядочена неправильно, применяется устройство «перекрещивание проводов».
- Если столбец имеет четный номер, аналогично обрабатываются пары проводов (2, 3), (4, 5) и т. д.

В книге [3] показано, что соответствующая сеть компараторов (изображенная на рис. 2.6) является сортирующей сетью, следовательно, данные n столбцов правильно упорядочат созданные в первых двух слоях n проводов.

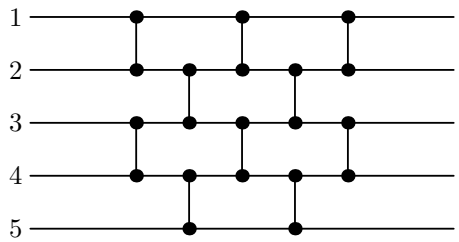


Рис. 2.6. Сеть компараторов, соответствующая четно-нечетной сортировке (для $n = 5$)

Итак, в сумме пять слоев имеют ширину $O(|E|)$ ячеек. Таким образом, все поле состоит из $O(|E|^2)$ ячеек, причем площадь одной ячейки — константа, не зависящая от E. Следовательно, площадь поля составляет $O(|E|^2)$ клеток.

Полученное поле имеет заполнение тогда и только тогда, когда формула E выполнима. Действительно, пусть E выполнима — существует набор значений переменных, при котором E истинна. Тогда заполним устройства «создание провода» в соответствии со значениями из этого набора, и все четыре последующих слоя заполнятся соответствующим образом — передавая эти значения устройствам «1 из 3», которые также будут иметь

заполнения (так как все предикаты формулы E выполнены).

Наоборот, если поле имеет заполнение, то набор значений, создаваемых в первом слое, будучи подставлен в формулу E, сделает ее истинной. Действительно, пройдя через слои со второго по четвертый, эти значения будут переданы устройствам «1 из 3», каждое из которых имеет заполнение, следовательно, в каждое из которых подано ровно одно значение `истина` и ровно два значения `ложь`. Следовательно, соответствующие этим устройствам предикаты формулы E выполнены, и формула E верна.

Таким образом, описан полиномиальный (относительно размера входа $|E|$) алгоритм, который строит по входу задачи 1-in-3 SAT экземпляр задачи L, имеющий такой же ответ («да» или «нет»). Следовательно, задача L NP-трудна, а с учетом результата теоремы 2.1 и NP-полна. □

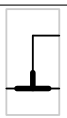
2.4.2. Набор устройств для двойных проводов

Как было отмечено в разд. 2.2.1, устройство «создание одинарного провода» может не существовать в силу тех или иных причин. В данном разделе описан набор устройств, достаточный для доказательства NP-полноты двумерной локально зависимой задачи, позволяющий обойти это препятствие.

Для начала рассмотрим естественный перенос всех устройств набора $\mathcal{A}_1$, описанного в предыдущем разделе, на язык двойных проводов. Для этого, по сути, заменим то, что раньше было одинарным проводом (`[f,t]`) на двойной провод (`[ft,tf]` — напомним, что в разд. 2.2.2 было принято решение работать с двойными проводами, одинарные провода внутри которых «в противофазе» — несут противоположные значения). Результат приведен в табл. 2.2.

Легко понять, что новый набор устройств достаточен для NP-полноты задачи — в доказательстве теоремы 2.11 надо лишь заменить все упоминания одинарных проводов двойными проводами. Однако новый набор слишком громоздкий и содержит в некотором смысле дублирующие

Таблица 2.2. Естественный перенос набора $\mathcal{A}_1$ на двойные провода

Устройство в $\mathcal{A}_1$	Старая функциональность	Новая функциональность
	`[f,t][f,t]`	`[ft,tf][ft,tf]`
	`[][f t]`	`[][ft tf]`
	`[f,t][ff,tt]`	`[ft,tf][ftft,tftf]`
	`[ff,ft,tf,tt][ff,tf,ft,tt]`	`[ftft,fttf,tfft,tftf]` `[ftft,tfft,fttf,tftf]`
	`[f,t][t,f]`	`[ft,tf][tf,ft]`

	`[fft,ftf,tff,`
	`fff ftt tft ttf ttt][,,,x]`
		`[ftfttf,fttfft,tfftft,`
		`ftftft fttftf tffttf tftfft tftftf]`
		`[,,,x]`

друг друга устройства. Рассмотрим другой, более подходящий для применения, набор устройств.

Определение 2.13. Набор устройств $\mathcal{A}_2$ представлен в табл. 2.3.

Теорема 2.14. *Двумерная локально зависимая задача L, для которой существует набор устройств $\mathcal{A}_2$, является NP-полной.*

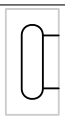
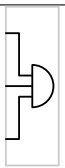
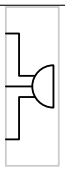
Доказательство. Для доказательства достаточности набора $\mathcal{A}_2$ воспользуемся доказанной ранее теоремой 2.11 и выразим все устройства набора $\mathcal{A}_1$ с помощью устройств набора $\mathcal{A}_2$. Шесть новых устройств изображены на рис. 2.7.

При этом при построении поля, описанном в доказательстве теоремы 2.11, произойдут следующие изменения:

- высота всех слоев (в ячейках) увеличится в два раза;
- ширина третьего и пятого слоя (в ячейках) увеличится в три раза;
- первые два слоя будут изменены.

Для упрощения построения (а также для уменьшения размеров получаемого поля) изменим суть первых двух слоев. В новой конструкции слои несут следующую функциональность:

Таблица 2.3. Набор устройств $\mathcal{A}_2$

Обозначение	Размер (в ячейках)	Название	Функциональность
	1×1	Одинарный провод	`[f,t][f,t]`
	2×1	Создание двойного провода	`[][ft tf]`
	2×1	Валидатор противоположности значений	`[ft,tf,ff tt][ft,tf,x]`
	2×1	Перекрещивание одинарных проводов	`[ff,ft,tf,tt][ff,tf,ft,tt]`
	3×1	1 из 3	`[fft,ftf,tff,fff ftt tft ttf ttt][,,,x]`
	3×1	2 из 3	`[ftt,tft,ttf,fff fft ftf tff ttt][,,,x]`

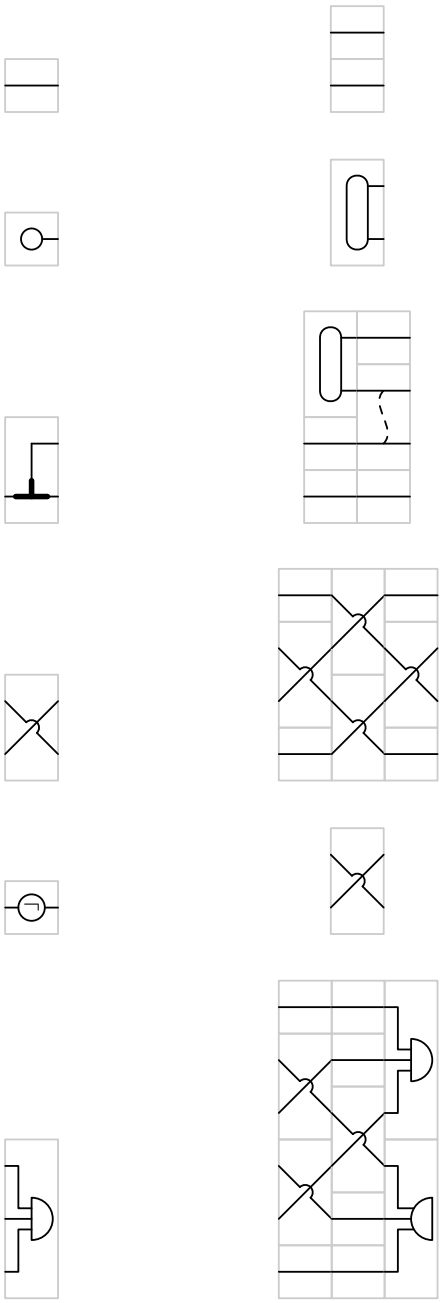


Рис. 2.7. Устройства из набора $\mathcal{A}_1$ (слева) и их реализации при помощи набора $\mathcal{A}_2$ (справа)

1. Для каждой переменной A, входящей в формулу E, создается несколько двойных проводов — ровно столько, сколько раз A входит в формулу E.
2. Для каждой переменной A, входящей в формулу E более одного раза, все пары соседних двойных проводов, соответствующих переменной A, проходят через «валидатор противоположности значений», чем и обеспечивается согласованность всех двойных проводов, соответствующих одной переменной.

В данной конструкции и первый, и второй слои имеют ширину в одну ячейку.

Ясно, что увеличение размера некоторых слоев (в ячейках) в константное число раз (а тем более уменьшение ширины второго слоя) не влияет на полиномиальность сведения. Таким образом, приведено полиномиальное сведение 1-in-3 SAT к L, следовательно, L является NP-трудной. Применяя же теорему 2.1, получаем, что L является NP-полной. □

Пример 2.15. На рис. 2.8 изображено поле, соответствующее формуле $R(A, \neg B, C) \wedge R(B, C, D) \wedge R(\neg A, \neg B, \neg C)$. Третий слой заметно упрощен вручную (для компактности).

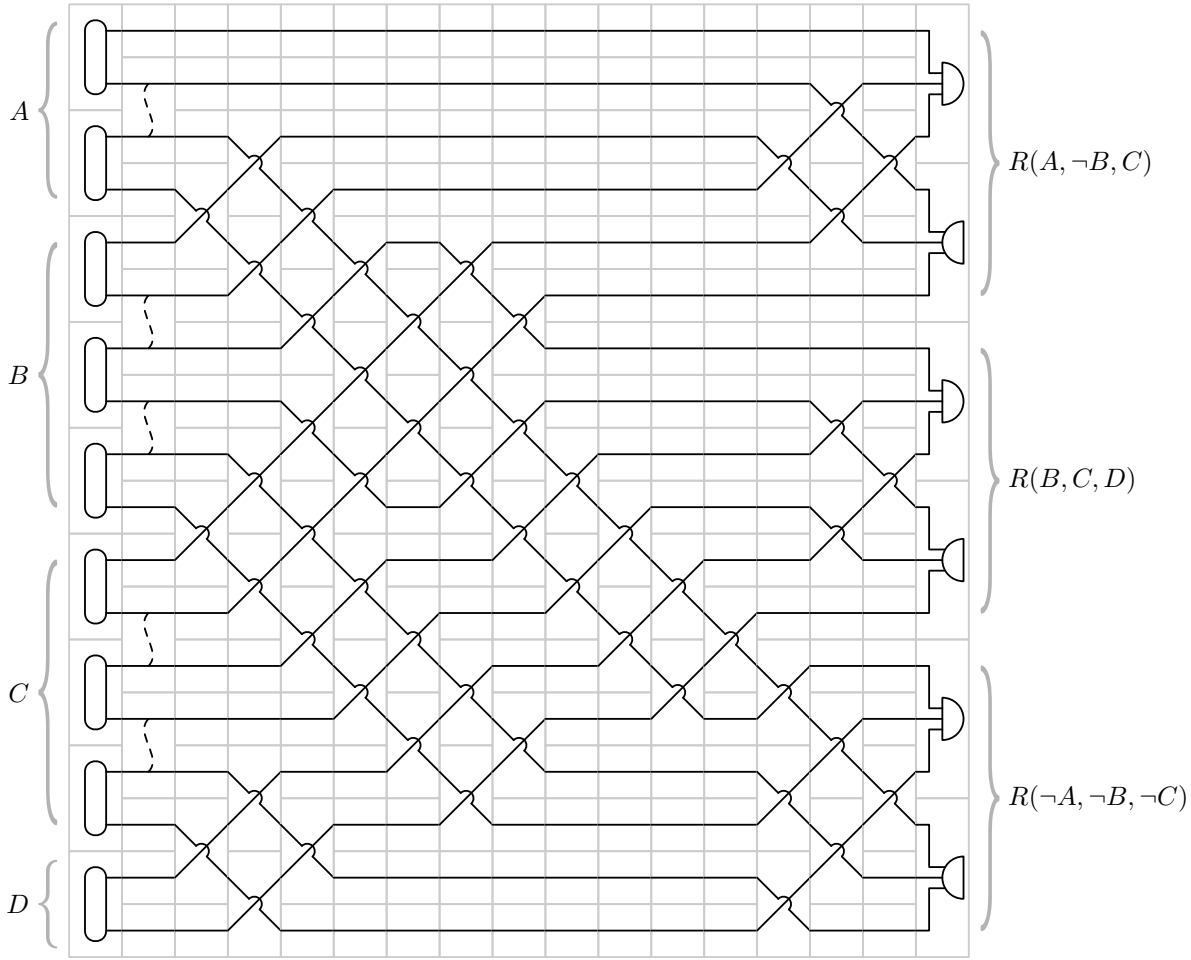


Рис. 2.8. Поле, соответствующее формуле $R(A, \neg B, C) \wedge R(B, C, D) \wedge R(\neg A, \neg B, \neg C)$ (набор устройств $\mathcal{A}_2$)

Глава 3. Автоматизация построения устройств

В настоящей работе предложен подход, позволяющий получать устройства, описанные в главе 2, не вручную, как это происходит повсеместно, а автоматически. В данной главе изложена суть предложенного подхода, а также произведен анализ времени его работы, и предложены несколько приемов оптимизации, позволяющие сократить время работы.

3.1. Применение динамического программирования для построения устройств

Рассмотрим задачу построения устройства с определенной функциональностью. А именно, пусть зафиксирована двумерная локально зависимая задача L, для нее выбран используемый провод (разд. 2.2), в том числе выбрано, как передается бит между двумя соседними по горизонтали ячейками.

Для данной задачи с данными конвенциями поставим перед собой цель: найти устройство, имеющее заданную функциональность, размером $h \times w$ клеток.

Пусть функциональность задана с помощью нотации, описанной в разд. 2.3, тогда введем обозначения:

n — число рассматриваемых ситуаций;

w_{in} — число входящих проводов;

w_{out} — число исходящих проводов.

Будем также считать, что положение входящих и исходящих проводов (их координата y) зафиксировано.

3.1.1. Общая идея

Для решения поставленной задачи рассмотрим следующий подход:

- Перебрать все поля размером $h \times w$ клеток.
- Для каждого поля проверить, является ли оно искомым устройством.

Существует $|A|^{h \cdot w}$ полей размера $h \times w$. Для каждого поля надлежит проверить, является ли оно искомым устройством.

Поле является устройством с искомой функциональностью в том случае, если в каждой из n ситуаций, описанных в нотации, при условии, что по входящим w_{in} проводам передаются значения, описанные в нотации, данное поле имеет такое множество заполнений, которое соотвествует всем наборам значений w_{out} исходящих проводов, описанным в нотации, и только им.

Зафиксируем некоторый набор значений входящих проводов. Поскольку имеется единая конвенция передачи бита между соседними ячейками, этот набор задает некоторую информацию о том, как заполнены ячейки слева от рассматриваемого нами поля. Следующее определение описывает наиболее удобный для работы случай.

Определение 3.1. Конвенцию передачи бита будем называть *однозначной*, если каждый набор булевых значений, передаваемых по входящим проводам, соответствует ровно одному профилю данной задачи.

В случае однозначной конвенции каждому из наборов значений входящих проводов, описанных в нотации, соответствует ровно один профиль, а каждой ситуации — множество профилей. Для каждой ситуации следует убедиться, что после заполнения поля при таком множестве начальных профилей, на выходе возможно ровно такое множиство профилей, которое соотвествует передаче по исходящим проводам наборов значений, описанным в нотации.

Для осуществления такой проверки имеет смысл применять метод динамического программирования по изломанному профилю (разд. 1.4.2).

3.1.2. Динамическое программирование по изломанному профилю в двумерных локально зависимых задачах

Убедимся в применимости динамического программирования по изломанному профилю к произвольной двумерной локально зависимой задаче L. Для этого исследуем, какое множество клеток в принципе влияют на возможность или невозможность заполнения поля справа от линии профиля. Пример такого множества для $s = 3$ приведен на рис. 3.1.

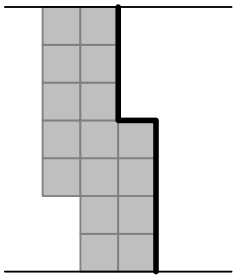


Рис. 3.1. Клетки, влияющие на дальнейшее заполнение, при $s = 3$

Несложно убедиться, что в общем случае число важных для дальнейшего заполнения клеток не превосходит $(h + 1)(s - 1)$. Действительно, в каждой из h строк важными точно являются правые $s - 1$ клетка. Кроме того, s-я справа клетка может также быть важна, если она влияет на клетку над изломом (ближайшую подлежащую обработке клетку). Таких клеток не более $s - 1$.

Отсюда следует верхняя оценка возможного числа профилей. Для каждой из h позиций излома на профили влияют не более $(h + 1)(s - 1)$ клеток, каждая из которых может иметь один из $|A|$ типов и быть заполнена одним из $|B|$ значений. Таким образом, число профилей не превышает $h \cdot (|A| \cdot |B|)^{(h+1)(s-1)}$.

Однако во многих ситуациях лишь очень малая доля всех теоретически возможных наборов типов и значений этих клеток задает непротиворечивую ситуацию. В реальных задачах число непротиворечивых профилей на порядки меньше приведенной оценки. С другой стороны, не зная специфику задачи, сложно (либо невозможно) дать разумную оценку сверху на число изломанных профилей.

В связи с этим введем дополнительное обозначение. Обозначим p максимальное число непротиворечивых профилей среди всех h позиций излома. Из вышесказанного следует, что $p \leq (|A| \cdot |B|)^{(h+1)(s-1)}$, однако стоит иметь в виду, что в реальных ситуациях p много меньше этой величины.

Оценим теперь время, необходимое для построения таблицы переходов. Для каждого изломанного профиля, число которых не превышает $h \cdot p$, следует перебрать все варианты следующей клетки — $|A|$ возможных типов и $|B|$ возможных значений. Зная новую клетку, следует зафиксировать переход в новый изломанный профиль, либо зафиксировать, что такого перехода нет.

Один из подходов, впрочем, иногда не самый эффективный, состоит в том, чтобы хранить каждый профиль как набор типов и значений «важных» клеток (отмеченных на рис. 3.1). (При этом можно отождествлять эквивалентные профили и хранить для отождествленного профиля какой-нибудь пример набора важных клеток, соответствующего этому профилю.) При таком подходе обработка одной клетки проходит следующим образом. Проверяется квадрат $s \times s$, содержащий новую клетку. Если он некорректен, то перехода нет, иначе имеет место переход к новому профилю. Новый профиль — это набор новых важных клеток, построение которого занимает $O(h \cdot s)$ времени.

Соответственно, даже при неэффективной (никак не учитывающей специфику задачи) обработке профилей таблица переходов строится за время $O(h^2 \cdot p \cdot s \cdot |A| \cdot |B|)$. В дальнейшем при оценке времени работы алгоритмов таблица переходов будет считаться уже построенной, и время построения таблицы переходов учитываться не будет.

3.1.3. Тривиальная реализация

Напомним, что рассматривается задача проверки конкретного поля на то, является ли оно определенным устройством. Соответственно, типы всех клеток поля на данный момент фиксированны и известны. Опишем

один алгоритм проверки.

Зафиксируем одну из n ситуаций. Сформируем начальное множество профилей, соответствующее наборам значений, передаваемых по входящим проводам в этой ситуации, а также конечное множество профилей, соответствующее наборам значений, передаваемых по исходящим проводам.

Обработаем по очереди все клетки от самого левого столбца к самому правому. Перед каждой итерацией имеется текущее множество изломанных профилей M, достижимых из начального множества (до первой итерации это просто оно само). Переберем все значения (множество B), которые можно поставить в обрабатываемую клетку. Для каждого профиля из M и для каждого значения из B согласно таблице переходов известен новый изломанный профиль, включающий себя обработанную клетку (рис. 3.2). Возможно также, что новый изломанный профиль не существует, что обозначает невозможность подстановки в данную клетку данного значения при данном профиле.

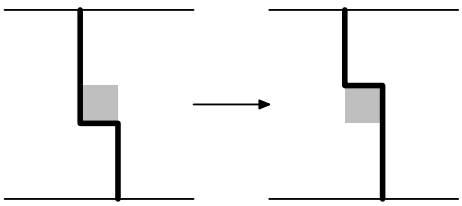


Рис. 3.2. Стандартный переход в динамике по изломанному профилю

Таким образом, перебрав все значения для этой клетки, формируется новое множество изломанных профилей N, в которых данная клетка значится как обработанная. Это множество и становится текущим при обработке следующей клетки и т. д.

Наконец, после обработки последней клетки, осталось лишь сравнить имеющееся текущее множество профилей с искомым конечным множеством, соответствующим данной ситуации. Если они совпадают, то искомое устройство построено.

Оценим время работы представленного алгоритма (для фиксиро-

ванного поля). Перед каждой итерацией множество достижимых профилей содержит не более p элементов, а для его хранения достаточно p битов памяти. Итерация состоит в рассмотрении не более p профилей, для каждого из которых рассматривается $|B|$ значений клетки. После этого в новом множестве профилей добавляется (или не добавляется) новый изломанный профиль. Если считать, что в таблице переходов хранятся номера профилей, а новое множество хранится как массив булевых значений, обозначающих принадлежит ли профиль с соответствующим номером новому множеству, то эта операция требует $O(1)$ времени. Соответственно, одна итерация требует $O(p \cdot |B|)$ времени.

Таким образом, обработка одной ситуации, заключающаяся в рассмотрении всех клеток поля, занимает время $O(h \cdot w \cdot p \cdot |B|)$, а обработка всех ситуаций требует $O(n \cdot h \cdot w \cdot p \cdot |B|)$ времени.

И, наконец, наиболее значимый фактор (причем в буквальном смысле, от лат. *factor* — множитель): большое число полей, определяемое соотношением $O(|A|^{h \cdot w})$, которое требуется рассмотреть для построения устройства. Поэтому время работы алгоритма в целом оценивается как $O(|A|^{h \cdot w} \cdot n \cdot h \cdot w \cdot p \cdot |B|)$.

3.1.4. Улучшенная реализация

Рассмотрим два поля, у которых совпадают левые половины, и отличаются только правые. При обработке этих двух полей будут производиться одни и те же действия вплоть до рассмотрения первой несовпадающей клетки. Таким образом, при работе со вторым полем половина действий будут напрасными. Существует несложный прием, позволяющий избежать лишней работы. Для этого требуется объединить фазу перебора всех полей и фазу обработки каждого поля в отдельности. При этом процесс перебора полей должен выглядеть следующим образом:

- для очередной клетки перебрать все возможные ее типы (множество A);

- для каждого типа клетки провести итерацию описанного выше алгоритма (для всех ситуаций) и рекурсивно запустить эту же процедуру для следующей клетки.

Тем самым для всех полей, имеющих «общий префикс», — для всех полей с совпадающими несколькими первыми клетками, эти общие клетки будут обрабатываться не каждый раз для каждого поля, как ранее, а один раз для всех полей.

Оценим время работы улучшенного алгоритма.

После каждого ветвления в памяти хранится уже выбранная и зафиксированная часть поля, а также — для каждой ситуации — множество изломанных профилей, достижимых из начального множества профилей, после обработки имеющейся части поля (их не более p).

Время обработки очередной клетки с фиксированным типом оценивается аналогично времени работы итерации предыдущего алгоритма, с той лишь разницей, что на этот раз рассматриваются все n ситуаций. В данном алгоритме очередная клетка обрабатывается за время $O(n \cdot p|B|)$.

Теперь необходимо посчитать, сколько раз обрабатывается каждая клетка. Пронумеруем клетки в порядке их обработки от 1 до $h \cdot w$. Исследуем, сколько раз обрабатывается клетка, имеющая номер i. Рекурсивная процедура для этой клетки будет вызвана столько раз, сколько существует «префиксов» поля до нее (не включительно), а именно $|A|^{i-1}$ раз. Кроме того, при каждом вызове будут перебраны все $|A|$ типов для этой клетки. Таким образом, клетка номер i будет обработана $|A|^i$ раз. (Например, последняя клетка, имеющая номер $h \cdot w$ обрабатывается один раз для каждого поля, то есть $|A|^{h \cdot w}$ раз).

Оценим суммарное число обрабатываемых клеток:

$$\sum_{i=1}^{h \cdot w} |A|^i = |A|^{h \cdot w} \cdot \left(1 + |A|^{-1} + |A|^{-2} + \ldots + |A|^{h \cdot w - 1}\right) <$$

$$|A|^{h \cdot w} \cdot \left(1 + |A|^{-1} + |A|^{-2} + \ldots\right) = |A|^{h \cdot w} \cdot \frac{1}{1 - \frac{1}{|A|}} \leq 2 \cdot |A|^{h \cdot w}$$

В последнем неравенстве используется предположение $|A| > 1$, которое представляется вполне логичным, так как иначе задача L тривиальна (поле обязано состоять только из клеток типа a_0, следовательно, заполнив его значениями b_0, получаем на задачу ответ «да»).

Итак, суммарно клетки обрабатываются $O(|A|^{h \cdot w})$ раз, и каждая обработка занимает $O(n \cdot p \cdot |B|)$ времени. Таким образом, общее время работы алгоритма составляет $O(|A|^{h \cdot w} \cdot |B| \cdot n \cdot p)$.

3.2. Метадинамическое программирование по изломанному профилю

В алгоритмах, описанных в разд. 3.1 имеется простор для оптимизации. Рассмотрим два поля, возможно заметно отличающиеся друг от друга по типу клеток, в момент когда у обоих обработаны первые i клеток. Для каждого из полей в данный момент хранится n множеств: для каждой из n ситуаций хранится множество изломанных профилей, достижимых из начального множества профилей. Отметим важное наблюдение: если эти n множеств у полей совпадают, то дальнейшая обработка оставшейся части поля будет для обоих полей проходить одинаково. Действительно, вся необходимая информация об уже обработанной части поля хранится в этих множествах профилей, и знания о конкретных типах клеток уже обработанной части поля если и понадобятся, то лишь для восстановления и вывода ответа.

Таким образом, из полей, входящих в один класс эквивалентности в описанном выше смысле, достаточно далее обрабатывать только одно поле. Эта идея оптимизации лежит в основе описываемого в данном разделе подхода — *метадинамического программирования*.

3.2.1. Понятие метапрофиля

Определение 3.2. *Метапрофилем* называется упорядоченный набор из n множеств изломанных профилей.

Метапрофиль содержит всю необходимую информацию для дальнейшей обработки поля после того, как обработаны первые i клеток. Рассмотрим теперь, как пересчитываются метапрофили по мере обработки клеток поля.

В начальный момент (до обработки первой клетки) метапрофиль содержит для каждой ситуации множество профилей, соответствующее наборам булевых значений, передаваемых по входящим проводам в этой ситуации.

Теперь рассмотрим обработку произвольной клетки поля. После обработки всех клеток до нее имеется текущий метапрофиль, задающий все дальнейшее развитие ситуации. Выберем для рассматриваемой клетки некоторый тип.

Теперь по очереди рассмотрим каждую из n ситуаций. Для каждой из них имеется множество изломанных профилей, достижимых из начального множества.

Для каждого профиля проведем стандартный шаг динамического программирования по изломанному профилю, а именно, переберем все значения рассматриваемой клетки (множество B) и для каждого значения согласно таблице переходов получим новый изломанный профиль.

Легко понять, что множество изломанных профилей, достижимых из начального после обработки этой клетки (при конкретном типе этой клетки), есть не что иное, как множество всех новых изломанных профилей, полученных в рамках этого перебора (всех профилей предыдущего шага и всех значений клетки).

Так для каждой из n ситуаций можно получить новое множество изломанных профилей. Соотвественно, новый метапрофиль — это упорядоченный набор из этих n множеств.

Таким образом, при обработке одной клетки из одного старого метапрофиля получается $|A|$ новых — по одному для каждого типа рассматриваемой клетки.

3.2.2. Общая схема реализации метадинамического программирования

Основной выигрыш при использовании метадинамического программирования происходит засчет того, что на каждом слое метапрофили можно отождествлять. Естественно, два метапрофиля считаются равными, если для всех n ситуаций множества изломанных профилей, соответствующие этой ситуации, в обоих метапрофилях совпадают.

Алгоритм метадинамического программирования состоит в следующем:

- Сформировать начальный метапрофиль согласно конвенции передачи данных.
- Для каждой клетки провести следующую процедуру: рассмотреть все текущие метапрофили, для каждого из них перебрать все возможные типы рассматриваемой клетки (множество A) и построить для каждого типа новый метапрофиль. Затем из полученного набора метапрофилей удалить все повторяющиеся равные элементы, и получить новое текущее множество метапрофилей.
- После обработки последней клетки рассмотреть все текущие метапрофили и для каждого из них проверить, является ли он искомым согласно конвенции передачи данных.

Метапрофиль является искомым, если для каждой из n ситуаций его множество профилей, соответствующее этой ситуации, в точности соответствует наборам значений исходящих проводов, описанных в нотации устройства для этой ситуации.

Таким образом можно за конечное время проверить существование устройства с заданной функциональностью заданного размера. Однако кроме наличия устройства желательно также знать, как оно выглядит — получить поле, которое имеет множество заполнений в точности подходящее под нотацию искомой функциональности.

Для этого предлагается вместе с каждым метапрофилем хранить

информацию, позволяющую восстанавливать поле (точнее, уже обработанный на данный момент префикс поля), которому соответствует такой метапрофиль. Следует отметить, что таких полей (префиксов полей) может быть более одного, если одинаковые профили были отождествлены. Достаточно информации несут следующие записи:

- непосредственно префикс поля — последовательность элементов множества A;
- тип последней обработанной клетки и ссылка на метапрофиль, имевший место до ее обработки.

Если все метапрофили (в том числе метапрофили предыдущих слоев) в каждый момент хранятся в памяти, то более эффективен (по требуемой памяти для хранения) второй подход, если же обработанный слой в памяти не хранится, то второй подход не применим.

Интересен также вопрос, какой из метапрофилей следует оставлять в рассматриваемом множестве в случае отождествления равных метапрофилей. (Если никакая информация о поле не хранится, то метапрофили абсолютно идентичны, в противном же случае один метапрофиль хранит информацию об одном поле, другой — о другом).

С формальной точки зрения оба поля, соответствующие одному метапрофилю, равноправны, однако для упрощения понимания человеком полученных результатов иногда желательно, чтобы искомое устройство выглядело «максимально просто».

Определение 3.3. *Критерием красоты* в задаче L называется численная величина, определенная для каждого префикса поля задачи L, оптимизация которой (неформально) соответствует упрощению поля.

В случае, когда вместе с метапрофилем хранится весь префикс поля, соответствующий ему, можно использовать любой критерий красоты, с тем только требованием, чтобы его вычисление не занимало слишком много времени. В случае, когда метапрофиль хранит только последнюю клетку префикса, в качестве критерия красоты разумно использовать индуктив-

ную функцию [7], то есть позволяющую вычислить критерий красоты нового метапрофиля, зная лишь критерий красоты старого и тип последней клетки.

Соответственно, при отождествлении равных метапрофилей в рассмотрении отстается тот из них, в котором хранимый префикс поля лучше согласно критерию красоты рассматриваемой задачи.

3.2.3. Оценка времени работы для метадинамического программирования

Оценим время работы алгоритма метадинамического программирования. Для этого сначала оценим время обработки одной клетки в условиях одного метапрофиля (это процедура была описана в разд. 3.2.1).

Перебираются все $|A|$ типов этой клетки, и для каждого типа строится свой метапрофиль. Для этого в каждой из n ситуаций рассматриваются все достижимые в условиях этой ситуации изломанные профили, их не более p штук. Для каждого изломанного профиля перебираются все $|B|$ возможных значений клетки, и рассматривается новый изломанный профиль согласно таблице переходов (если такой переход есть). Получается не более $p \cdot |B|$ новых профилей, из которых формируется множество — одно из n множеств нового метапрофиля.

Таким образом, обработка одной клетки требует $O(|A| \cdot |B| \cdot n \cdot p)$ времени, и после этой операции из одного метапрофиля получается $|A|$ новых метапрофилей.

Введем обозначение m_i — число метапрофилей в текущем слое до обработки i-й клетки поля. Известно, что $m_1 = 1$, то есть до обработки первой клетки имеется один метапрофиль — начальный, построенный в соответствии с конвенцией передачи данных. Кроме того, процедура построения нового слоя метапрофилей по старому слою гарантирует неравенство $m_{i+1} \leq |A| \cdot m_i$.

С помощью введенного обозначения оценим время работы метадинамического программирования. Построение новых метапрофилей проис-

ходит для всех слоев метапрофилей, кроме последнего. Таким образом, описанная выше операция, занимающая $O(|A| \cdot |B| \cdot n \cdot p)$ времени, выполняется $\sum_{i=1}^{h \cdot w} m_i$ раз.

Кроме построения новых метапрофилей имеется также фаза отождествления равных, однако, если хранить метапрофили в хеш-таблице [4] или в боре (trie [3]), то добавление нового метапрофиля в имеющуюся коллекцию метапрофилей занимает (асимптотически) не больше времени, чем его построение. Аналогичное утверждение верно и для проверки каждого метапрофиля из последнего слоя на то, является ли он искомым конечным метапрофилем. Поэтому достаточно оценивать время, требуемое для построения всех новых метапрофилей.

Осталось оценить величины m_i. Во-первых, согласно приведенным выше ограничениям, $m_i \leq |A|^{i-1}$, что является наиболее разумной оценкой, не учитывающий возможность отождествления метапрофилей.

В действительности же, в каждом слое все метапрофили различны, поэтому размер слоя не может превосходить числа теоретически возможных метапрофилей. Найдем это число. Метапрофиль состоит из n элементов, каждый из которых является подмножеством множества профилей. Всего изломанных профилей (в каждом слое) не более p, следовательно, множеств профилей не более 2^p. А наборов из n таких объектов, соответственно, не более $(2^p)^n = 2^{n \cdot p}$. (Проще говоря, в каждой из n ситуаций каждый из p изломанных профилей может быть либо достижимым, либо недостижимым).

Отсюда следует оценка сверху, верная для произвольного слоя: $m_i \leq 2^{n \cdot p}$.

Из имеющихся неравенств получим оценку на сумму $\sum_{i=1}^{h \cdot w} m_i$:

$$\begin{cases} \sum_{i=1}^{h \cdot w} m_i \leq 1 + |A| + \ldots + |A|^{h \cdot w - 1} \leq 2 \cdot |A|^{h \cdot w - 1} \\ \sum_{i=1}^{h \cdot w} m_i \leq h \cdot w \cdot 2^{n \cdot p} \end{cases}$$

(в первом неравенстве используется предположение $|A| > 1$, аналогично вычислениям из разд. 3.1.4).

Комбинируя эти оценки, получаем, что время работы алгоритма метадинамического программирования составляет

$$O(|B| \cdot n \cdot p \cdot \min(|A|^{h \cdot w}, |A| \cdot h \cdot w \cdot 2^{n \cdot p})).$$

В этой оценке присутствует величина p, зависящая от специфики задачи. Если абстрагироваться от нее, оценка времени работы принимает следующий вид:

$$O(|B| \cdot n \cdot (|A| \cdot |B|)^{(h+1)(s-1)} \cdot \min(|A|^{h \cdot w}, |A| \cdot h \cdot w \cdot 2^{n \cdot (|A| \cdot |B|)^{(h+1)(s-1)}})).$$

Интересно, что второй аргумент функции min является линейным от w, хотя, безусловно, константа при w во всех реальных ситуациях чрезвычайно велика.

Следует также отметить, что первый аргумент функции min — не что иное как время работы улучшенной реализации динамического программирования (разд. 3.1.4), таким образом метадинамическое программирование работает (асимптотически) не хуже.

3.3. Применение метадинамического программирования для доказательства NP-полноты

Алгоритм метадинамического программирования, описанный в разд. 3.2, применим для доказательства NP-полноты двумерных локаль-

но зависимых задач. А именно, поскольку этот алгоритм позволяет искать устройства с заданной фунцкциональностью заданного размера, его можно использовать для автоматического построения устройств, составляющих наборы, описанные в главе 2.

3.3.1. Общая схема применения

Основное «узкое место» при применении метадинамического программирования для доказательства NP-полноты конкретной задачи заключается в том, что необходимо знать изломанные профили, которые соответствуют передаче по некоторому набору проводов того или иного набора булевых значений — для формирования начального и конечного (искомого) метапрофиля при построении каждого устройства. В общем случае для этого необходимо и достаточно знать, как устроен провод, передающий значение `ложь`, и провод, передающий значение `истина`. Профиль, соответствующий нескольким проводам, обычно тривиально складывается из отдельных частей, соответствующих проводам по отдельности.

Таким образом, прежде чем начинать автоматически искать устройства, необходимо выбрать конвенцию передачи данных. Рассматриваемый в настоящей работе подход не включает методов для автоматизации этого выбора. В каждой конкретной задаче считается целесообразным использовать естественный интеллект человека для придумывания способа передачи информации между ячейками.

Также применение естественного интеллекта требуется для программной реализации динамического программирования по профилю. А именно, необходимо реализовать процедуру построения следующего изломанного профиля по предыщему при известном типе и значении обрабатываемой клетки. Именно в этой процедуре выражена суть и структура рассматриваемой двумерной локально зависимой задачи.

Наконец, необязательной, но возможной задачей для человека является выбор критерия красоты (определение 3.3) — формализация понятия

о том, какое устройство более желательно получить в качестве ответа в случае существования нескольких искомых устройств.

После описанных действий применение метадинамического программирования к конкретной задаче не требует интеллектуальных изысков. Человеческая роль заключается в выборе (зачастую переборе) размеров искомых устройств.

3.3.2. Ограничение высоты рассматриваемой области

Приведенная в разд. 3.2.3 оценка времени работы алгоритма метадинамического программирования показывает экспоненциальную зависимость времени работы от высоты рассматриваемой области. И хотя указанная асимптотика является оценкой сверху, результаты работы программы на примерах подтверждают существенную зависимость времени работы от h.

В связи с вышесказанным, встает вопрос о возможной минимизации высоты устройств, возможно, впрочем, засчет увеличения ширины.

Во-первых, уменьшить рассматриваемые высоты можно, если на этапе придумывания конвенции передачи данных сформулировать такие правила, которые позволят разместить «позиции входа» и «позиции выхода» соседних проводов максимально близко друг к другу.

Во-вторых, если можно выбирать или варьировать искомый набор устройств, следует предпочитать устройства, у которых «высота в смысле числа проводов», то есть величина $\max(w_{in}, w_{out})$, как можно меньше. В разд. 3.3.3 будет показан один прием, применимый на практике.

3.3.3. Взаимозаменяемость входящих и исходящих проводов

С точки зрения минимизации высоты рассматриваемой области, наборы устройств, описанные в разд. 2.4.2, не оптимальны. Так в наборе $\mathcal{A}_2$ имеются устройства «1 из 3» и «2 из 3», принимающие по три входных провода. В задачах, в которых высота рассматриваемой области критич-

на, такие устройства являются наиболее сложными для автоматического построения. К счастью, существует набор устройств, достаточный для доказательства NP-полноты задачи, в котором каждое устройство имеет не более двух входящих и не более двух исходящих проводов. Покажем, какой прием позволяет получить такой набор.

Рассмотрим устройство «1 из 3». Оно имеет три входящих провода и ноль исходящих. Возникает естественное желание перенести один входящий провод с левой стороны на правую. Будем временно считать его не «исходящим», а «входящим с правой стороны». Функциональность устройства при этом остается той же: оно имеет заполнение тогда и только тогда, когда из трех входящих (теперь с разных сторон) проводов ровно один несет значение `истина`.

Попробуем теперь развернуть одинарный провод на 180°. Для этого нужно устройство «разворот одинарного провода» с двумя входящими проводами и без исходящих проводов. При этом есть два варианта функциональности этого устройства:

- `[ff,tt,ft tf][,,x]` — провод поворачивает, не меняя значение.
- `[ft,tf,ff tt][,,x]` — провод поворачивает, меняя значение.

Это устройство в некотором смысле симметрично устройству «создание двойного провода», причем два варианта функциональности соответствуют двум вариантам структуры двойных проводов: они могут состоять из одинарных, либо идущих «в фазе», либо идущих «в противофазе». Поскольку, как было отмечено в разд. 2.2.2, в настоящей работе рассматривается второй тип двойных проводов, по аналогии будем рассматривать поворот провода, при котором передаваемое значение меняется (второй вариант).

Следует отметить, что случай двойных проводов, состоящих из одинарных, передающих равные значения, не сложнее и рассматривается аналогично.

Вернемся к устройству «1 из 3». Один из входящих проводов теперь

перенесен в другую сторону и имеет противоположное значение (это и есть общее правило переноса входящих проводов из левой части в правую). В таком случае, устройство должно иметь заполнение в следующих и только в следующих ситуациях:

- По входящим проводам переданы два значения `ложь`, по исходящему передано значение `ложь`.
- По входящим проводам переданы разные значения, по исходящему передано значение `истина`.

В нотации, описанной в разд. 2.3, функциональность такого устройства записывается как `[ff,ft,tf,tt][f,t,t,x]`. Таким образом, в случае двух входящих значений `истина` устройство не имеет заполнения, а иначе — передает далее логическое ИЛИ значений входящих проводов. Иными словами, это устройство выясняет, верно ли, что из входящих проводов ровно один передает значение `истина`. Поэтому будем называть это устройство «1 из 2».

Теперь рассмотрим устройство «2 из 3». В нем также перенесем один из входящих проводов в правую часть, обратив его значение. Новое устройство должно иметь заполнения ровно в следующем множестве ситуаций:

- По входящим проводам переданы два значения `истина`, по исходящему передано значение `истина`.
- По входящим проводам переданы разные значения, по исходящему передано значение `ложь`.

В используемой нотации данная функциональность записывается как `[tt,ft,tf,ff][t,f,f,x]`. Нестрого говоря, это устройство выясняет, верно ли, что из входящих двух проводов оба передают значение `истина`. Будем называть это устройство «2 из 2».

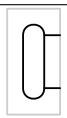
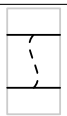
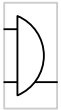
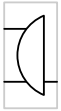
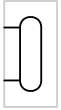
Наконец, вернемся к устройству «разворот одинарного провода». Оно запрещает входящим двум проводам передавать равные значения. Однако устройство со сходной функциональностью уже есть — это «валидатор противоположности значений» (разд. 2.4.2). Если поставить «валидатор»

перед «разворотом», то «разворот» не должен будет разбирать случай равных значений; это новое, более простое устройство, имеет функциональность `[ft,tf][,]`. Логично назвать это устройство «окончание двойного провода».

Приведенные рассуждения позволяют представить набор устройств, достаточный для доказательства NP-полноты, в котором все устройства имеют не более двух входящих и не более двух исходящих проводов.

Определение 3.4. Набор устройств $\mathcal{A}'_2$ представлен в табл. 3.1.

Таблица 3.1. Набор устройств $\mathcal{A}'_2$

Обозначение	Размер (в ячейках)	Название	Функциональность
	1×1	Одинарный провод	`[f,t][f,t]`
	2×1	Создание двойного провода	`[][ft tf]`
	2×1	Валидатор противоположности значений	`[ft,tf,ff tt][ft,tf,x]`
	2×1	Перекрещивание одинарных проводов	`[ff,ft,tf,tt][ff,tf,ft,tt]`
	2×1	1 из 2	`[ff,ft,tf,tt][f,t,t,x]`
	2×1	2 из 2	`[tt,ft,tf,ff][t,f,f,x]`
	2×1	Окончание двойного провода	`[ft,tf][,]`

Теорема 3.5. *Двумерная локально зависимая задача L, для которой существует набор устройств $\mathcal{A}'_2$, является NP-полной.*

Доказательство. Воспользуемся доказанной ранее теоремой 2.14 и выразим с помощью набора $\mathcal{A}'_2$ все устройства набора $\mathcal{A}_2$. А именно, достаточно привести реализации устройств «1 из 3» и «2 из 3», так как остальные устройства набора $\mathcal{A}_2$ присутствуют в $\mathcal{A}'_2$. Искомые реализации приведены

на рис. 3.3 и 3.4.

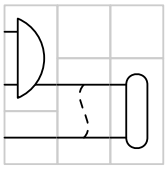


Рис. 3.3. Реализация устройства «1 из 3» при помощи набора $\mathcal{A}_2'$

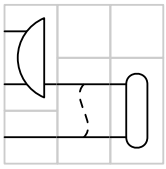


Рис. 3.4. Реализация устройства «2 из 3» при помощи набора $\mathcal{A}_2'$

Соответственно, в конструкции, соответствующей набору $\mathcal{A}_2$, произойдет одно изменение: ширина пятого слоя увеличится на две ячейки. Ясно, что это изменение не влияет на полиномиальность сведения. Таким образом, набора $\mathcal{A}_2'$ достаточно для доказательства NP-полноты двумерной локально зависимой задачи L. □

Доказанная теорема свидетельствует, что для доказательства NP-полноты двумерных локально зависимых задач методом, излагаемым в настоящей работе, достаточно рассматривать устройства, имеющие не более двух входящих и не более двух исходящих проводов, и, соответственно, высотой не более чем в две ячейки. В разделе 4.2 будет приведен пример задачи, доказательство NP-полноты которой было получено именно благодаря описанным выше приемам.

Глава 4. Практическое внедрение

Описанный в главе 3 алгоритм метадинамического программирования был реализован программно и применен для доказательства NP-полноты некоторых двумерных локально зависимых задач, в том числе тех, для которых ранее вопрос об их NP-полноте оставался открытым.

Раздел 4.1 содержит доказательство NP-полноты задачи $(2,3)$-замощения, повторяющее результат, полученный в 1995-м году [8]. Раздел 4.2 содержит доказательство NP-полноты задачи 4-CROSS SUM. Более того, полученный набор устройств для задачи 4-CROSS SUM оказывается также достаточным для доказательства NP-полноты задачи N-CROSS SUM при любом $N \geq 4$. (утверждение 4.2). Таким образом, оказывается разрешенным ранее открытый вопрос об NP-полноте задачи N-CROSS SUM для $N \in \{4,5,6\}$.

4.1. Доказательство NP-полноты задачи $(2,3)$-замощения

В данном разделе описан набор устройств $\mathcal{A}_2$, полученный автоматически для задачи $(2,3)$-замощения, и тем самым, согласно теореме 2.14 доказана ее NP-полнота.

Для начала сформулируем, что является изломанным профилем для задачи $(2,3)$-замощения. Пусть уже заполнены несколько столбцов целиком и несколько первых клеток очередного столбца. На рис. 4.1 сплошная линия отделяет уже заполненные клетки от (возможно) еще не заполненных.

Исследуем, какие клетки справа от сплошной линии могут быть покрыты уже поставленными полимино. Утверждается, что это клетки, отмеченные на рисунке знаком вопроса. Действительно, каждая из них может быть покрыта горизонтальным домино, поставленным ранее при обработке клетки слева. Наоборот, никакие другие клетки не могут быть покрыты ни

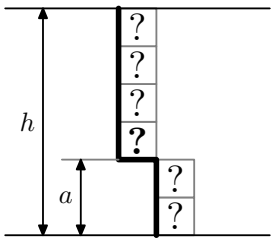


Рис. 4.1. Профиль в задаче $(2, 3)$-замощения

горизонтальным домино (ближайшая обработанная клетка на две левее), ни вертикальным тримино (в стобце просто нет обработанных клеток).

Таким образом, изломанный профиль хранит следующую информацию: положение излома a $(0 \leq a < h)$ и h битов, обозначающих покрыта ли полимино самая левая необработанная клетка в каждой строке.

Число различных профилей в данной задаче составляет $h \cdot 2^h$.

Теперь опишем провод. Как оказалось, в данной задаче можно рассмотреть наиболее естественную конструкцию провода — горизонтальную полосу высоты 1. Ясно, что такую полосу можно замостить только горизонтальными домино, причем существуют две возможных «фазы» замощения — линии границы между соседними домино могут иметь либо четные координаты x, либо нечетные. Эти два варианта соответствуют передаваемым значениям `ложь` и `истина`(рис. 4.2).

Рис. 4.2. Одинарный провод для задачи $(2, 3)$-замощения

Для данного одинарного провода были найдены (как автоматически, так и вручную) вспомогательные устройства, позволяющие сдвигать одинарный провод на три клетки вниз или вверх (рис. 4.3, 4.4). Эти вспомогательные устройства пригодятся в дальнейшем: их можно будет вставить перед сложным, найденным автоматически, устройством, чтобы сдвинуть точку входа и точку выхода проводов.

Интересно, что создать вспомогательное устройство, сдвигающее провод по вертикали на число клеток, не кратное 3, невозможно. Причиной

Рис. 4.3. Сдвиг одинарного провода на три клетки вверх

Рис. 4.4. Сдвиг одинарного провода на три клетки вниз

тому является утверждение, аналогичное утверждению 2.6, но рассматривающее координаты y белых клеток:

Утверждение 4.1. *В любом поле задачи (p, q)-замощения, имеющем заполнение, для всех целых j в интервале $[0, q-1)$ число белых клеток, чья координата y сравнима с j по модулю q, дает один и тот же остаток при делении на p.*

Имея это утверждение, предположим, что существует устройство, сдвигающее горизонтальный провод со строчки y_1 на строчку y_2, причем $y_1 \not\equiv y_2 \pmod 3$. Рассмотрим два случая передаваемого значения, `ложь` и `истина`. В одном из них имеет заполнение поле «гипотетическое устройство плюс клетка с координатой y_1», а в другом — «гипотетическое устройство плюс клетка с координатой y_2».

Согласно утверждению 4.1, в заполненном поле число белых клеток с координатой y, сравнимой по модулю 3 с нулем, с единицей и с двойкой — три числа одинаковой четности. Однако при переходе от одного рассмотренного нами поля к другому из этих трех чисел два числа меняют четность, а третье — не меняет. Получили противоречие, следовательно сдвиг провода возможен только на число клеток, кратное 3.

Перейдем, наконец, к построению набора $\mathcal{A}_2$ на языке задачи $(2, 3)$-замощения. Автоматически были получены все шесть устройств искомого набора, после чего они были простейшими приемами (продлением проводов и применением вспомогательных устройств сдвига по вертикали) приведе-

ны к единому размеру и единой конвенции стыковки.

Ячейки имеют размер 6×10, передача информации (стыковка) происходит на третьей сверху строке ячейки. Правило стыковки формулируется следующим образом: если домино заканчивается ровно на границе между ячейками, это соответствует передаче значения `ложь`, если домино переходит границу и занимает одну клетку в следующей ячейке, это соответствует значению `истина`.

Шесть устройств набора $\mathcal{A}_2$ приведены на рис. 4.5—4.9.

Наиболее сложным для нахождения оказалось устройство «перекрещивание одинарных проводов». Если положить расстояние между входящими и исходящими проводами равным трем, то наименьшее по площади занимаемого поля найденное устройство имеет размер 6×14, оно изображено на рис. 4.10. В процессе построения этого устройства были обработаны 1 335 539 метапрофилей.

К сожалению, два устройства такого вида нельзя располагать непосредственно одно над другим, без разделения черными ячейчками. Если бы это было возможно, то существовал бы набор $\mathcal{A}_2$ с высотой ячейки, составляющей всего три клетки.

Если же перейти к расстоянию между проводами, равному шести, то наименьшее по площади найденное устройство «перекрещивание одинарных проводов» — использованное в приведенном наборе (рис. 4.8). Для его нахождения потребовалось перебрать 12 027 186 метапрофилей.

Следует отметить, что при построении устройств для задачи $(2, 3)$-замощения критерием красоты поля (определение 3.3) считалось наименьшее число белых клеток.

4.2. Доказательство NP-полноты задачи 4-CROSS SUM

В данном разделе описан набор устройств $\mathcal{A}'_2$ (определение 3.4), полученный автоматически для задачи 4-CROSS SUM, и тем самым, согласно

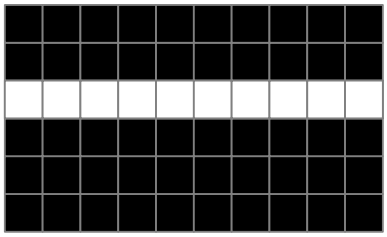


Рис. 4.5. «Одинарный провод»

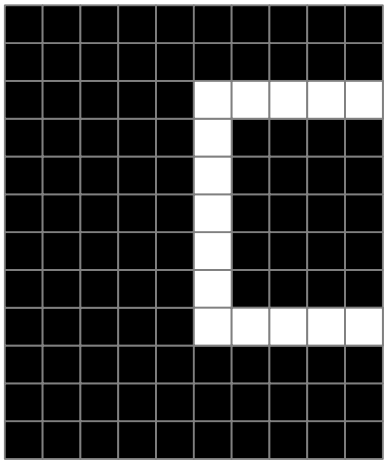


Рис. 4.6. «Создание двойного провода»

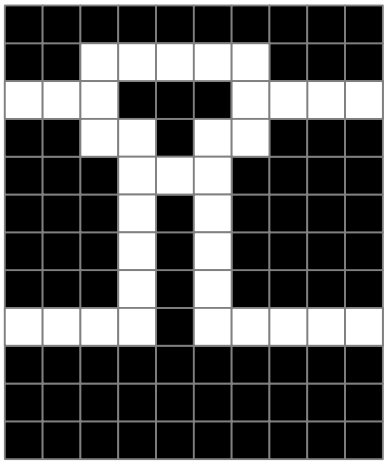


Рис. 4.7. «Валидатор противоположности значений»

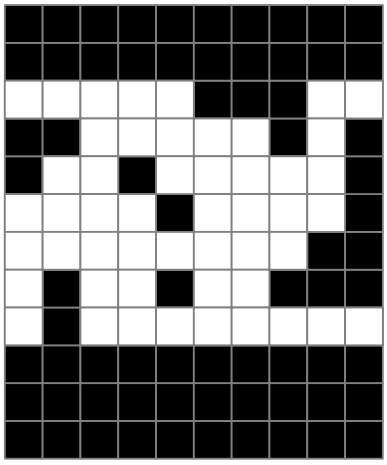


Рис. 4.8. «Перекрещивание одинарных проводов»

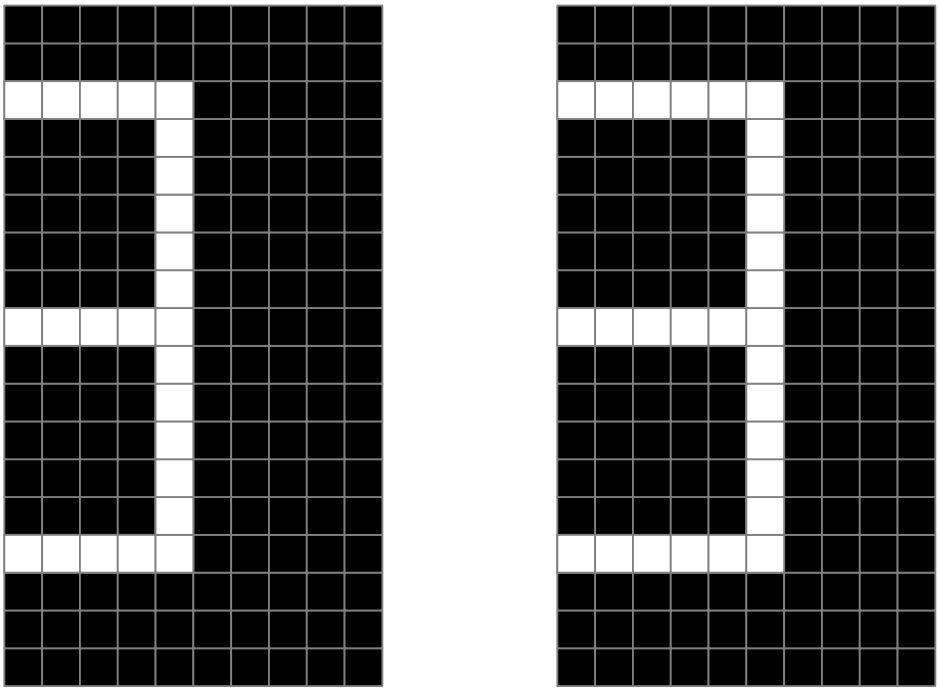


Рис. 4.9. Устройства «1 из 3» и «2 из 3»

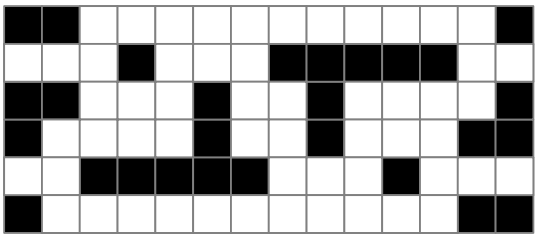


Рис. 4.10. Альтернативный вариант устройства «Перекрещивание одинарных проводов»

теореме 3.5 доказана ее NP-полнота.

Для начала опишем профиль задачи 4-CROSS SUM. Пусть уже заполнены несколько столбцов целиком и несколько первых клеток очередного столбца. На рис. 4.11 сплошная линия отделяет заполненные клетки от незаполненных.

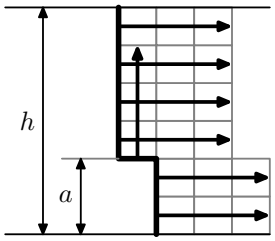


Рис. 4.11. Профиль в задаче 4-CROSS SUM

Рассмотрим, какая информация нам необходима для дальнейшего заполнения поля. Зафиксируем некоторую строку, пусть в ней непосредственно перед сплошной линией расположена группа заполненых белых клеток. Тогда для дальнейшего заполнения этой строки необходимо знать множество цифр, стоящих в этой группе — иначе есть возможность поставить в продолжении этой группы некоторую цифру повторно. Следует отметить, что порядок расположения цифр до сплошной черты не важен, важно только множество значений.

Кроме того, если в текущем рассматриваемом столбце непосредственно перед сплошной линией расположена группа заполненных белых клеток, профиль также обязан хранить множество цифр, стоящих в ней.

Итого профиль хранит положение излома a $(0 \leq a < h)$ и $h + 1$ запись о том, какое множество цифр поставлено в текущую «открытую» группу белых клеток во всех строках и в текущем столбце. Каждая такая запись содержит 4 бита значимой информации. При этом запись, говорящая, что в текущей группе поставлено пустое множество цифр, соответствует тому, что текущей группы нет, то есть последняя заполненная клетка — черная.

Из вышесказанного следует, что число возможных профилей в задаче 4-CROSS SUM составляет $h \cdot 2^{4(h+1)}$, а в случае обобщенной задачи

N-CROSS SUM — $h \cdot 2^{N(h+1)}$. Однако это — оценка сверху, так как многие профили противоречивы, например, профиль, в котором значится, что в текущем столбце нет открытой группы белых клеток (последняя клетка — черная), а в последней обработанной строке открытая группа есть (последняя клетка — белая).

Для задачи 4-CROSS SUM был автоматически построен набор $\mathcal{A}'_2$, описанный в главе 3. Этот выбор обусловлен критичностью высоты рассматриваемой области. Для построения всех устройств этого набора потребовалось рассмотреть устройства высотой в 5 клеток. В этих условиях существуют 797 257 непротиворечивых профилей. При построении устройств высотой в 6 клеток потребовалось бы работать с 9 251 304 профилями, а число профилей в поле высотой в 7 клеток слишком велико, чтобы их можно было хранить в оперативной памяти современного компьютера.

Наиболее сложным для нахождения оказалось устройство «валидатор противоположности значений». При его построении было рассмотрено 2 856 129 метапрофилей.

Опишем полученный набор устройств. Ячейки имеют размер 6×12. Кроме того, в целях возможности стыковки, на устройства накладываются следующие ограничения:

- Верхняя строка устройства состоит из черных клеток.
- В правом столбце каждой ячейки верхние три ячейки — черные.
- В левом столбце каждой ячейки нижние две ячейки — черные.

Передача бита информации происходит на третьей снизу строке. При этом конвенции передачи состоят в следующем:

- Третья снизу строка левой ячейки оканчивается открытой группой белых клеток.
- Все возможные заполнения левой ячейки удовлетворяют следующему условию: дополнить открытую группу белых клеток в третьей снизу строке можно только единственным способом, а именно одной цифрой — либо единицей, либо двойкой.

- Заполнение, требующее в этом месте единицу, обозначает передачу значения `ложь`, а заполнение, требующее двойку, — значения `истина`.

Описанные выше конвенции проиллюстрированы на рис. 4.12.

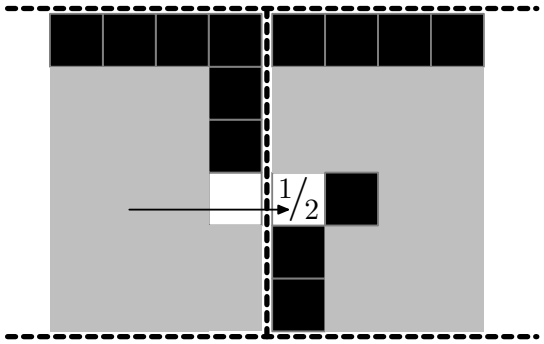


Рис. 4.12. Конвенция стыковки устройств

Полученный набор устройств $\mathcal{A}'_2$, удовлетворяющий указанным условиям, приведен на рис. 4.13 — 4.19. Как и следовало ожидать, устройство «окончание двойного провода» есть устройство «создание двойного провода», повернутое на 180 градусов, и сдвинутое для соблюдений конвенций стыковки.

При построении устройств для задачи 4-CROSS SUM критерием красоты считалось в первую очередь минимальное число белых клеток, при равенстве этого параметра — минимальная сумма квадратов всех подсказок.

Важным представляется следующий факт.

Утверждение 4.2. *Приведенный набор устройств достаточен для доказательства NP-полноты задачи N-CROSS SUM для всех* $N \geq 4$.

Доказательство. Во-первых, отметим, что данное утверждение не является тривиальным: те или иные свойства устройства, выполненные в задаче 4-CROSS SUM, могут быть не выполнены для того же устройства в задаче 5-CROSS SUM. Например, группа из двух белых клеток с суммой 6, которая ранее имела только заполнения $2 + 4$ и $4 + 2$, теперь может быть заполнена как $1 + 5$ или $5 + 1$ и т. п.

Однако полученные устройства, к счастью, обладают следующим свойством: переход от задачи 4-CROSS SUM к задаче N-CROSS SUM для

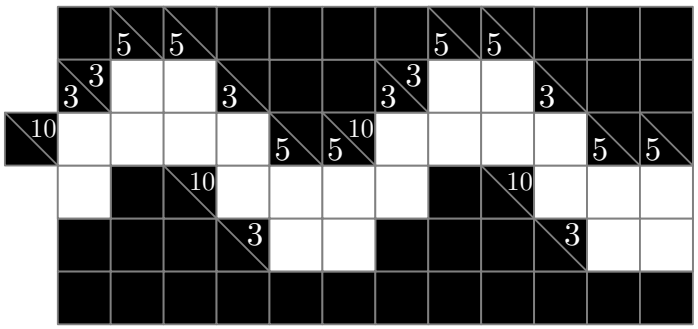


Рис. 4.13. «Одинарный провод»

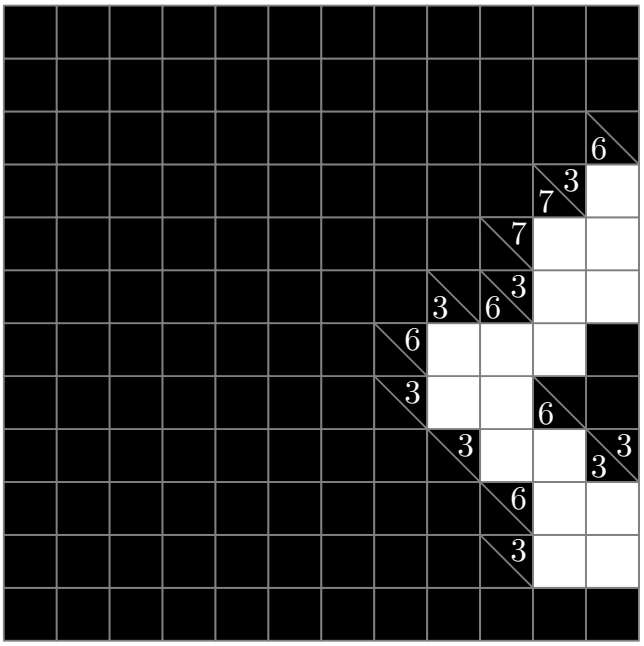


Рис. 4.14. «Создание двойного провода»

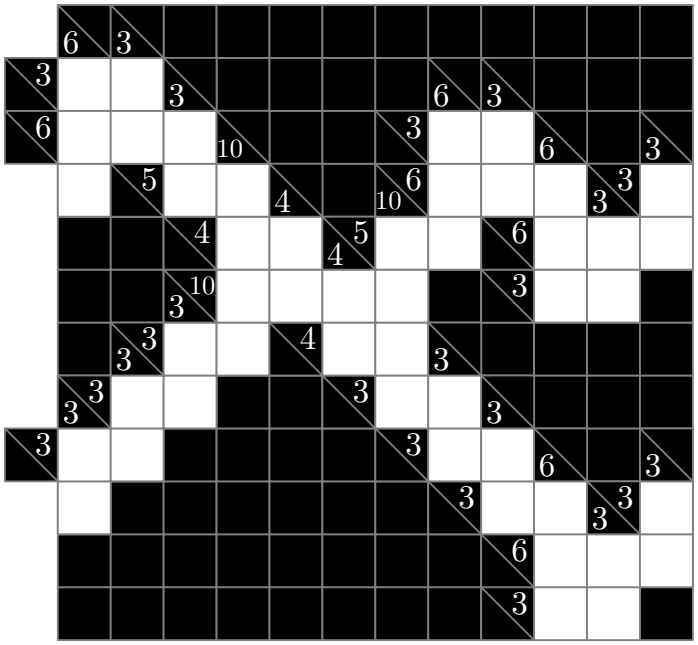


Рис. 4.15. «Валидатор противоположности значений»

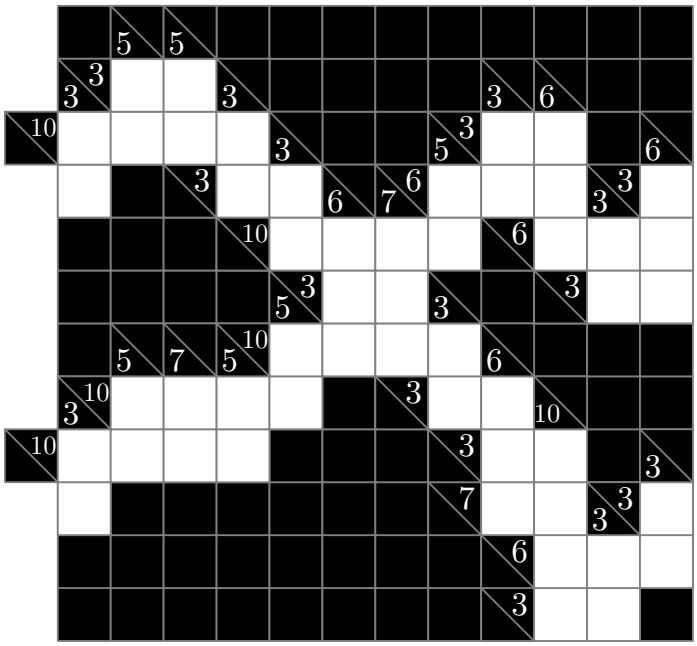


Рис. 4.16. «Перекрещивание одинарных проводов»

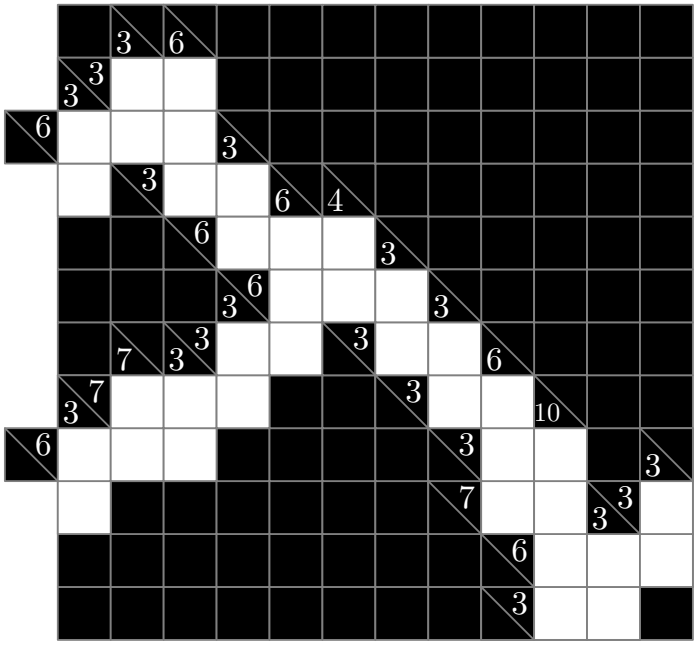


Рис. 4.17. Устройство «1 из 2»

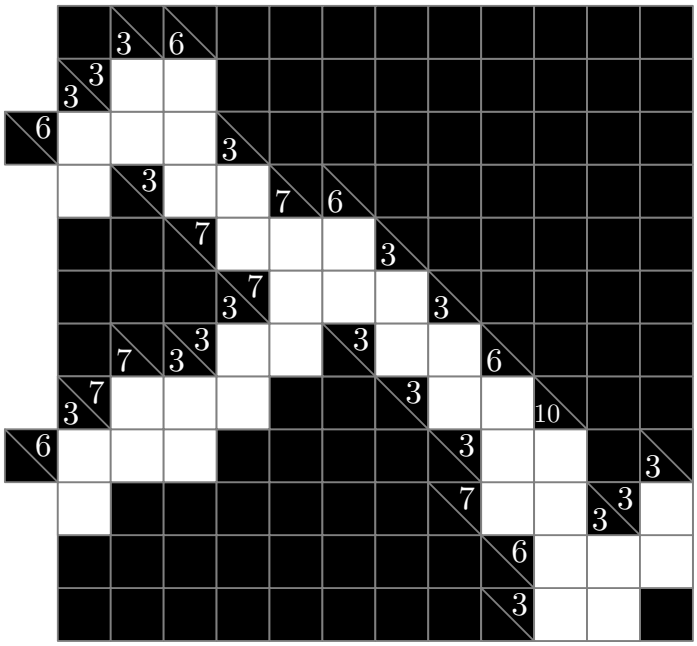


Рис. 4.18. Устройство «2 из 2»

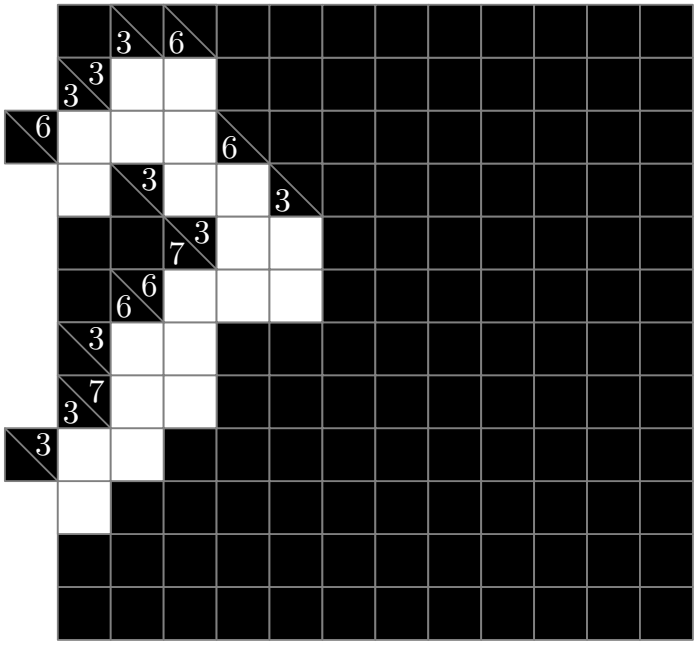


Рис. 4.19. Устройство «Окончание двойного провода»

любого $N > 4$ (иными словами, разрешение использовать цифры, большие 4) не увеличивает множество возможных заполнений всех приведенных выше устройств.

Докажем этот факт. Для этого заметим, что каждая белая клетка каждого устройства принадлежит одной из нижеприведенных групп (либо горизонтальной, либо вертикальной, либо обеим сразу):

- 2 клетки с суммой 3;
- 2 клетки с суммой 4;
- 2 клетки с суммой 5;
- 3 клетки с суммой 6;
- 3 клетки с суммой 7;
- 4 клетки с суммой 10.

Как видно, подстановка в любую из этих групп цифры, большей 4, невозможна. Следовательно, для каждой белой клетки есть группа, гарантирующая, что эта клетка может быть заполнена только цифрой от 1 до 4. Поэтому разрешение цифр, больших 4, не создает никаких новых заполнений. □

Тем самым, для задачи N-CROSS SUM при любом $N \geq 4$ существует набор устройств $\mathcal{A}'_2$, и согласно теореме 3.5 задача N-CROSS SUM является NP-полной при $N \geq 4$.

Соответственно, остававшийся открытым [16, 18] вопрос об NP-полноте (или принадлежности классу P) задачи N-CROSS SUM для $N \in [3, 6]$ теперь остается открытым лишь для $N = 3$.

Заключение

В настоящей работе был формализован класс задач, описывающий задачи на клетчатом поле, для проверки корректности решения которых достаточно проверять квадраты фиксированного размера. Была доказана вложенность описанного класса задач в класс задач NP.

Для двумерных задач была описана схема доказательства NP-полноты, использующая провода и устройства, и приведены наборы устройств, достаточные для сведения к рассматриваемой задаче задачи выполнимости булевой формулы. Также были формализованы и описаны ограничения, накладываемые на устройства, и доказана NP-полнота двумерной локально зависимой задачи, для которой существует один из указанных наборов устройств с соотвествующими ограничениями.

Для автоматического построения устройств с заданной функциональностью заданного размера был предложен алгоритм метадинамического программирования, базирующийся на методе динамического программирования по изломанному профилю. Был проведен анализ времени работы предложенного алгоритма.

С помощью программной реализации алгоритма метадинамического программирования были построены достаточные для доказательства NP-полноты наборы устройств для двух двумерных локально зависимых задач: $(2, 3)$-замощения и 4-CROSS SUM. Набор устройств, построенный автоматически для задачи 4-CROSS SUM, оказался также достаточным для доказательства NP-полноты задачи N-CROSS SUM при любом $N \geq 4$. Таким образом, был разрешен открытый вопрос о NP-полноте задачи N-CROSS SUM для $N \in \{4, 5, 6\}$.

В целом в настоящей работе был предложен новый подход к доказательству NP-полноты задач на клетчатом поле, и показана, как теоретически, так и на конкретных примерах, его применимость на практике.

Источники

[1] *Василевский Б. О.* Динамическое программирование по изломанному профилю. http://informatics.mccme.ru/moodle/mod/book/view.php?id=290&chapterid=78.

[2] *Верещагин Н. К., Шень А.* Лекции по математической логике и теории алгоритмов. Языки и исчисления. М.: МЦНМО, 2002.

[3] *Кнут Д. Э.* Искусство программирования, том 3. Сортировка и поиск. М.: Вильямс, 2004.

[4] *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО: БИНОМ, 2004.

[5] *Павлов Д. С.* Лекции по информатике. http://g6prog.narod.ru/dpl.ps.

[6] *Хопкрофт Д. Э., Мотвани Р., Ульман Д. Д.* Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2002.

[7] *Шень А.* Программирование: теоремы и задачи. М.: МЦНМО, 2004.

[8] *Beauquier D., Nivat M., Rémila E., Robson M.* Tiling figures of the plane with two bars // *Computational Geometry*. 1995. Vol. 5. Pp. 1–25.

[9] *Cocking C.* Finding an optimal tooth color match // *ACM Crossroads*. 2008. Vol. 14, no. 3. Pp. 22–25.

[10] *Eppstein D.* Computational Complexity of Games and Puzzles. http://www.ics.uci.edu/~eppstein/cgt/hard.html.

[11] *Friedman E.* Corral Puzzles are NP-complete. http://www.stetson.edu/~efriedma/papers/corral/corral.html.

[12] *Friedman E.* The Game of Cubic NP-complete. http://www.stetson.edu/~efriedma/papers/cubic.pdf.

[13] *Hearn R. A.* TipOver is NP-complete. http://www.dartmouth.edu/~rah/tipover.pdf.

[14] *Kaye R.* How Complicated is Minesweeper. http://web.mat.bham.ac.uk/R.W.Kaye/minesw/ASE2003.pdf.

[15] *Schaefer T. J.* The complexity of satisfiability problems // *ACM Symposium on Theory of Computing*. 1978. Pp. 216–226.

[16] *Seta T.* The complexities of puzzles, CROSS SUM and their another solution problems (ASP) // *Senior Thesis, Department of Information Science, the Faculty of Science, the University of Tokyo*. 2002.

[17] Web Nikoli, Kakuro rules outline. http://nikoli.co.jp/en/puzzles/kakuro/index_text.htm.

[18] *Yato T., Seta T.* Complexity and completeness of finding another solution and its application to puzzles // *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*. 2003. Vol. 86, no. 5. Pp. 1052–1060.