Теоретическая и практическая сложность задачи о выполнимости булевых формул

Е. А. Поцелуевская

Введение

Проблема выполнимости булевых формул (проблема пропозициональной выполнимости) — это одна из наиболее известных NP-полных задач. Несмотря на то, что в общем случае проблема выполнимости не разрешима за полиномиальное время, нахождение случаев, когда ответ может быть получен быстро, очень важно для различных прикладных задач. В частности, тесты, основанные на проблеме выполнимости сегодня широко применяются для автоматизации проектирования, а также для проверки разрабатываемых программ. С другой стороны, выявление сложных случаев задачи о выполнимости, позволяет реализовывать более эффективные системы защиты информации. В данной работе приведен обзор различных формулировок задачи о выполнимости, а также алгоритмов её решения.

Обзор основан на результатах работы Алексеева В. Б. и Носова В. А [13] по проблеме NP-полноты, обзоре дискретных алгоритмов решения задачи Всемирнова М. А., Гирша Э. А., Данцина Е. Я. и Иванова С. В. [14], на статьях из сборника Дингжу, Гу и Пардалоса [4] и других работах.

1. Задача о выполнимости и её модификации

1.1. Выполнимость функции, заданной в конъюнктивной нормальной форме (КНФ)

Дано. Формула над булевыми переменными $x_1, \ldots, x_n$, имеющая вид: $(x_{i_1}^{\alpha_1} \vee \cdots \vee x_{i_k}^{\alpha_k})(x_{j_1}^{\beta_1} \vee \cdots \vee x_{j_l}^{\beta_l}) \ldots (x_{t_1}^{\gamma_1} \vee \cdots \vee x_{t_p}^{\gamma_p})$, где $(\alpha_i, \beta_i, \ldots, \gamma_i$ — булевы константы).

Вопрос. Существует ли набор значений переменных $x_1 = \sigma_1, \ldots, x_n = \sigma_n$, обращающий формулу в единицу?

Известно, что эта задача NP-полна. Рассмотрим некоторые ее ограничения.

1.2. k-выполнимость КНФ

Дано. Формула КНФ над булевыми переменными $x_1, \ldots, x_n$, имеющая в каждой скобке фиксированное число k переменных.

Вопрос. Существует ли набор значений переменных $x_1 = \sigma_1, \ldots, x_n = \sigma_n$, обращающий формулу в единицу?

Задача k-выполнимости является NP-полной для всех фиксированных $k \geqslant 3$. Она остается NP-полной, даже если $k = 3$ и каждая скобка содержит либо все переменные без отрицания, либо все переменные с отрицанием. При $k = 2$ задача 2-выполнимости полиномиально разрешима. В литературе имеется ряд алгоритмов полиномиальной сложности для задачи 2-выполнимости, в том числе имеющих и линейную сложность. Индивидуальные задачи для 2-выполнимости называются биюнктивными формулами. Имеется критерий биюнктивности произвольной булевой функции $f(x_1, \ldots, x_n)$. Булева функция $f(x_1, \ldots, x_n)$ биюнктивна в том и только в том случае, когда для любых трех наборов переменных $(x_1, \ldots, x_n), (y_1, \ldots, y_n), (z_1, \ldots, z_n)$ выполнено условие (операции покоординатные): $\bar{f}(xy + yz + xz)f(x)f(y)f(z) \equiv 0$.

Это дает полиномиальный алгоритм проверки биюнктивности, если f задана таблицей, совершенной дизъюнктивной нормальной формой (СДНФ) или совершенной конъюнктивной нормальной формой (СКНФ).

1.3. k-выполнимость при различных литералах

Дано. Формула КНФ над булевыми переменными $x_1, \ldots, x_n$, имеющая в каждой скобке фиксированное число k переменных.

Литералом называется булева переменная или её отрицание.

Вопрос. Существует ли набор значений переменных $x_1 = \sigma_1, \ldots, x_n = \sigma_n$, обращающий формулу в единицу, такой что в каждой скобке для этого набора есть хотя бы один истинный литерал и хотя бы один ложный литерал?

Задача k-выполнимости при различных литералах является NP-полной для всех фиксированных $k \geqslant 3$. При $k = 2$ задача 2-выполнимости при различных литералах полиномиально разрешима.

1.4. k-выполнимость при одном истинном литерале

Дано. Формула КНФ над булевыми переменными $x_1, \ldots, x_n$, имеющая в каждой скобке фиксированное число k переменных.

Вопрос. Существует ли набор значений переменных $x_1 = \sigma_1, \ldots, x_n = \sigma_n$, обращающий формулу в единицу, такой что в каждой скобке для этого набора есть в точности один истинный литерал?

Задача k-выполнимости при одном истинном литерале является NP-полной для всех фиксированных $k \geqslant 3$. При $k = 2$ задача 2-выполнимости при одном истинном литерале полиномиально разрешима.

1.5. k-выполнимость при ложном литерале

Дано. Формула КНФ над булевыми переменными $x_1, \ldots, x_n$, имеющая в каждой скобке фиксированное число k переменных.

Вопрос. Существует ли набор значений переменных $x_1 = \sigma_1, \ldots, x_n = \sigma_n$, обращающий формулу в единицу, такой что в каждой скобке для этого набора есть, по крайней мере, один ложный литерал?

Задача k-выполнимости при одном истинном литерале является NP-полной для всех фиксированных $k \geqslant 3$. При $k = 2$ задача 2-выполнимости при ложном литерале полиномиально разрешима.

Приведенные выше задачи полиномиально разрешимы при $k = 2$. В то же время, в задаче 2-выполнимости может появляться трудноразрешаемость. Следующая задача NP-полна.

1.6. Максимальная 2-выполнимость

Дано. Формула КНФ над булевыми переменными $x_1, \ldots, x_n$, имеющая в каждой скобке 2 переменных, и натуральное число k.

Вопрос. Существует ли набор значений переменных $x_1 = \sigma_1, \ldots, x_n = \sigma_n$, такой что для этого набора не менее k скобок истинны?

Задача становится полиномиально разрешимой при k, равном числу скобок в формуле.

1.7. (r, s)-выполнимость КНФ

Дано. Формула КНФ над булевыми переменными $x_1, \ldots, x_n$, в которой каждая скобка содержит r переменных и каждая переменная входит самое большее в s скобок.

Вопрос. Существует ли набор значений переменных $x_1 = \sigma_1, \ldots, x_n = \sigma_n$, обращающий формулу в единицу?

Доказано, что $(3, 4)$ — выполнимость есть наиболее сильное ограничение для задачи выполнимости, при которой она остается NP-полной. Доказано также, что каждая формула класса (r, r)-выполнимость является выполнимой. Представляет интерес следующий результат, полученный в [5]. Пусть r_0 и s_0 таковы, что каждая формула класса (r_0, s_0)-выполнимости является выполнимой. Тогда выполнима каждая формула класса $(r_0 + 1, s_0 + [s_0/r_0])$-выполнимости, где $[x]$ — целая часть числа x. Существовала гипотеза, что любая формула класса (r, s)-выполнимости при $s = 2^{r-1}$ является выполнимой. В работе [5] был построен соответствующий контрпример с $r = 5, s = 11$. В то же время, данная гипотеза справедлива при $r < 4$.

1.8. Выполнимость слабоотрицательных формул

Дано. Формула КНФ над булевыми переменными $x_1, \ldots, x_n$, в которой каждая скобка содержит только переменные с отрицаниями кроме, быть может, одной, то есть формула вида: $(x_{i_1}^{\alpha} \vee \bar{x}_{i_2} \vee \cdots \vee \bar{x}_{i_k})$ $(x_{j_1}^{\beta} \vee \bar{x}_{j_2} \vee \cdots \vee \bar{x}_{j_l}) \ldots (x_{t_1}^{\gamma} \vee \bar{x}_{t_2} \vee \cdots \vee \bar{x}_{t_k})$, где $(\alpha, \beta, \ldots, \gamma$ — булевы константы).

Вопрос. Существует ли набор значений переменных $x_1 = \sigma_1, \ldots, x_n = \sigma_n$, обращающий формулу в единицу?

Полиномиальная разрешимость задачи выполнимости для данного класса формул установлена Шефером. Имеется следующий критерий слабоотрицательности произвольной булевой функции $f(x_1, \ldots, x_n)$. Булева функция $f(x_1, \ldots, x_n)$ слабоотрицательна в том и только в том случае, когда для любых двух наборов переменных $(x_1, \ldots, x_n), (y_1, \ldots, y_n)$ выполнено условие (операции покоординатные): $f(\bar{x} \wedge y)f(x)f(y) \equiv 0$.

Это дает полиномиальный алгоритм проверки слабоотрицательности, если f задана таблицей, СДНФ или СКНФ.

1.9. Выполнимость слабоположительных формул

Дано. Формула КНФ над булевыми переменными $x_1, \ldots, x_n$, в которой каждая скобка содержит только переменные без отрицаний, кроме, быть может, одной, то есть формула вида: $(x_{i_1}^{\alpha} \vee x_{i_2} \vee \cdots \vee x_{i_k})$ $(x_{j_1}^{\beta} \vee x_{j_2} \vee \cdots \vee x_{j_l}) \ldots (x_{t_1}^{\gamma} \vee x_{t_2} \vee \cdots \vee x_{t_k})$, где $(\alpha, \beta, \ldots, \gamma$ — булевы константы).

Вопрос. Существует ли набор значений переменных $x_1 = \sigma_1, \ldots,$ $x_n = \sigma_n$, обращающий формулу в единицу?

Полиномиальная разрешимость задачи выполнимости для данного класса формул установлена Шефером. Имеется следующий критерий слабоположительности произвольной булевой функции $f(x_1, \ldots, x_n)$. Булева функция $f(x_1, \ldots, x_n)$ слабоположительна в том и только в том случае, когда для любых двух наборов переменных $(x_1, \ldots, x_n), (y_1, \ldots, y_n)$ выполнено условие (операции покоординатные):$f(\bar{x} \vee y)f(x)f(y) \equiv 0$.

Это дает полиномиальный алгоритм проверки слабоположительности, если f задана таблицей, СДНФ или СКНФ.

1.10. Выполнимость мультиаффинных формул

Дано. Формула над булевыми переменными $x_1, \ldots, x_n$,представляющая собой конъюнкцию линейных форм, то есть формула вида: $(a_1x_1 + \ldots a_nx_n + a_0)(b_1x_1 + \ldots b_nx_n + b_0) \ldots (c_1x_1 + \ldots c_nx_n + c_0)$, где $(a_i, b_i, \ldots, c_i$ — булевы константы).

Вопрос. Существует ли набор значений переменных $x_1 = \sigma_1, \ldots,$ $x_n = \sigma_n$, обращающий формулу в единицу?

Ясно, что данная задача выполнимости полиномиально разрешима и имеет ту же сложность, что и задача проверки совместности системы линейных уравнений над $GF(2)$.

1.11. Задача об F-выполнимости

Дано $F = F_1, \ldots, F_m$ — любое конечное множество формул (функциональных символов). Определим F-формулу как конъюнкцию $F_{i_1}(\cdot)F_{i_2}(\cdot)\ldots F_{i_k}(\cdot)$ с переменными x_1, ,x_n, расставленными некоторым образом.

Вопрос Существует ли набор значений переменных $x_1 = \sigma_1, \ldots,$ $x_n = \sigma_n$, обращающий F-формулу в единицу?

То есть, проблема F-выполнимости — это проблема выполнимости F-формул. Важный результат Шефера [11] состоит в следующем.

Теорема 1. *Проблема F-выполнимости полиномиально разрешима, если все функции F_i из множества F одновременно удовлетворяют, по крайней мере, одному из условий:*

- $F_i(0, \ldots, 0) = 1$*;*
- $F_i(1, \ldots, 1) = 1$*;*
- F_i — *мультиаффинна;*
- F_i — *биюнктивна;*
- F_i — *слабоположительна;*
- F_i — *слабоотрицательна.*

В противном случае проблема F-выполнимости является NP-полной.

Таким образом, классы мультиаффинных, слабоположительных, слабоотрицательных, биюнктивных формул играют роль предполных классов.

1.12. Задача поиска выполняющего набора

Все описанные выше варианты задачи о выполнимости были сформулированы как проблемы распознавания, то есть для данной формулы (F-формулы) необходимо выдать ответ только о существовании выполняющего набора. Однако вопрос может быть поставлен шире: если существует выполняющий набор $x_1 = \sigma_1, \ldots, x_n = \sigma_n$, необходимо выдать его, если нет — выдать ответ, что задача невыполнима.

Задачи распознавания и поиска выполняющего набора полиномиально сводятся друг к другу, то есть если $|F|$ — длина входной формулы F, а $T(F)$ — время решения одной из задач, то вторая задача разрешима за время $poly(|F|)T(F)$. В связи с этим, далее рассматриваются алгоритмы для поиска выполняющего набора.

2. Основные классы алгоритмов решения задачи и их теоретическая сложность

Все известные на сегодняшний день алгоритмы решения задачи о выполнимости (в том числе при различных ограничениях, приведенных в предыдущем разделе), могут отнесены к одному из следующих основных классов:

- по области поиска решения: дискретные и непрерывные;
- по наличию условий, налагаемых при решении задачи: условные и безусловные;
- по последовательности вычислений: последовательные и параллельные.

Большая часть существующих алгоритмов изначально являются последовательными. Параллельные же алгоритмы, как правило, появляются в результате незначительного преобразования последовательных алгоритмов. Далее описываются только последовательные алгоритмы.

2.1. Дискретные условные алгоритмы

В алгоритмах, относящихся к данной категории, переменные $x_1, \ldots, x_n$ принимают только значения из множества $\{0, 1\}$. Задача о выполнимости здесь рассматривается как задача условной оптимизации.

Формулировка задачи: дана булева формула в КНФ: $F = D_1 \wedge \cdots \wedge D_m$, где m — число дизъюнктов в формуле, $x = (x_1, \ldots, x_n)$ — вектор из n булевых переменных. Целевая функция задается следующим образом: $N(x) = \sum\limits_{i=1}^{m} D_i(x)$, где

$$D_i(x) = \prod_{j=1}^{n} Q_{i,j}(x_j);$$

$$Q_{i,j}(x_j) = \begin{cases} 1 - x_j, & \text{если } x_j \text{ входит в дизъюнкт } D_i; \\ x_j, & \text{если } \bar{x_j} \text{ входит в дизъюнкт } D_i; \\ 1, & \text{в противном случае.} \end{cases}$$

Задача: найти $\min_{x \in \{0,1\}^n} N(x)$ при условии, что $D_i(x) = 0 \quad \forall i \in \{1, 2, \ldots, m\}$.

Алгоритмы данного класса как правило заключаются в сведении задачи для входной формулы F к задаче для полиномиального числа формул $F_1, \ldots, F_p$. Этот процесс может быть как детерминированным (рекурсивно вызывать алгоритм для каждой из формул F_i), так и вероятностным (случайно выбирать одну из формул F_i). В работе [14] подобные алгоритмы были названы *расщепляющими*. Современные расщепляющие алгоритмы можно разделить на два семейства: DPLL-алгоритмы и PPSZ-алгоритмы.

2.1.1. DPLL-алгоритмы

DPLL-алгоритмы основаны на процедурах, описанных в работах Дэвиса и Патнема [3] и Дэвиса, Лоджмана и Лавлэнда [2]. Алгоритмы, относящиеся к данному типу, принято считать первыми алгоритмами для решения задачи выполнимости. Большинство существующих алгоритмов решения задачи выполнимости основаны именно на этой технике.

Общая последовательность шагов DPLL-алгоритмов приведена ниже.

Процедура 1.

1) Упростить входную формулу F, то есть преобразовать F в некоторую другую формулу G, используя некоторые правила преобразования.
2) Если задача G тривиальна, выдать ответ.
3) Выбрать переменную x_i, входящую в G, используя некоторую эвристику. Построить формулы $G[x_i]$ и $G[\bar{x}_i]$, где $G[x_i]$ — это формула, полученная из формулы G присваиванием переменной x_i значения 1. Далее алгоритм рекурсивно применяется для каждой из формул $G[x_i]$ и $G[\bar{x}_i]$. Если хотя бы один из рекурсивных вызовов вернул выполняющий набор, обновить его, добавив x_i или $\bar{x}_i$ соответственно, и вернуть результат (обновление может также включать изменения, вызванные использованием правил преобразования). В противном случае выдать ответ, что формула невыполнима.

Таким образом, описанная процедура имеет следующие параметры:

- Правила преобразования, служащие для упрощения формул (предполагается, что упрощение занимает полиномиальное время);
- Эвристика для выбора переменной для расщепления (также за полиномиальное время).

Векторы расщепления Работа процедуры 1 может быть представлена с помощью *дерева расщепления*. Его корень помечен формулой, полученной в результате упрощения исходной формулы F. Если вершина дерева помечена формулой G, то два ребра, выходящие из неё, заканчиваются вершинами, которые отмечаются формулами, полученными в результате упрощения формул $G[x_i]$ и $G[\bar{x}_i]$. Листья помечены тривиальными (не содержащими переменных) формулами.

Если имеется дерево расщепления, то для каждой его вершины можно записать рекуррентное неравенство для верхней оценки $T(n)$ времени работы процедуры 1:

$$T(n) \leqslant 2T(n-1) + poly(|F|).$$

Кульманом и Люкхардтом [6, 7] была предложена техника для работы с подобными рекуррентными неравенствами. Вместо рекуррентного неравенства каждой вершине расщепления приписывается *вектор расщепления*. Например, если требуется найти сложность относительно числа переменных в формуле, вектор расщепления образуется следующим образом. Рассмотрим расщепление формулы G, содержащей n переменных на формулы $G_1, \ldots, G_d$, содержащие $n_1, \ldots, n_d$ переменных соответственно: $G \to G_1, \ldots, G_d$. Тогда вектор расщепления $(t_1, \ldots, t_d)$ состоит из произвольных чисел $t_i \leqslant n - n_1$. Соответствующее *число расщепления* есть единственное решение уравнения

$$\sum_{i=1}^{d} x^{-t_i} = 1$$

на интервале $(0, +\infty)$. Тогда время работы процедуры 1 ограничено сверху величиной $poly(|F|) \cdot \tau^n$, где τ — наибольшее из чисел расщепления для всех вершин дерева. Оценка $1.505^n poly(|F|)$ для 3-выполнимости из работы [6], полученная таким образом, являлась рекордной среди детерминированных алгоритмов в течение шести лет.

Правила преобразования Ниже приведен список некоторых правил преобразования, используемых в DPLL-алгоритмах.

1) Удаление единичных дизъюнктов. Если F содержит дизъюнкт, состоящий из единственного литерала x_i^α, где $\alpha \in \{0, 1\}$, положить значение x_i равным α.

2) Чистый литерал. Если F содержит *чистый* литерал, то есть литерал x_i^α такой что его отрицание не входит в F, положить $x_i = \alpha$.

3) Резолюция. Пусть x_i — переменная из F. Добавить к F все резольвенты по x_i и удалить из F все дизъюнкты, содержащие x_i или её отрицание. Резольвентой двух дизъюнктов $(x_i \vee x_{i_2}^{\alpha_2} \vee \cdots \vee x_{i_k}^{\alpha_k})$ и $(\bar{x_i} \vee x_{j_2}^{\beta_2} \vee \cdots \vee x_{j_l}^{\beta_l})$ называется дизъюнкт $(x_{i_2}^{\alpha_2} \vee \cdots \vee x_{i_k}^{\alpha_k} \vee x_{j_2}^{\beta_2} \vee \cdots \vee x_{j_l}^{\beta_l})$, если $x_{i_t} \neq x_{j_r}$ для всех i_t, j_l и 1 в противном случае.

4) Поглощение. Если F содержит два дизъюнкта $D_i \in D_j$, удалить D_j.

5) Автаркность. Если существует (частичный) набор A, такой что $F[A]$ не содержит дизъюнктов, не входящих в формулу F, заменить F на $F[A]$.

6) Черные и белые литералы. Пусть P — некоторое свойство формул и литералов, вычислимое за полиномиальное время. Предположим также, что свойства $P(F, x)$ («x — белый литерал, $\bar{x}$ — черный литерал») и $P(F, \bar{x})$ («$\bar{x}$ — белый литерал, x — черный литерал») не могут выполняться одновременно. Если каждый дизъюнкт F, который содержит литерал x_i^α, удовлетворяющий условию $P(F, x_i^\alpha)$, содержит также литерал x_j^β, удовлетворяющий условию $P(F, x_j^{\beta+1})$, заменить F на $F[\{x_j^\beta | P(F, x_j^{\beta+1})\}]$.

7) Заблокированный дизъюнкт. Дизъюнкт D называется *заблокированным*, если он содержит литерал x_i^α, такой что каждый дизъюнкт, содержащий $x_i^{\alpha+1}$, содержит также отрицание некоторого другого литерала из D. Любой заблокированный дизъюнкт может быть удален из F.

Выбор переменной Основная эвристика для выбора переменной заключается в следующем: необходимо выбрать переменную, соответствующую наименьшему числу расщепления. Однако в большинстве случаев используются эвристики: «выбрать переменную, входящую в самый короткий дизъюнкт» и «выбрать переменную, входящую в наибольшее количество дизъюнктов».

Алгоритм решения задачи об F-выполнимости, приведенный в [16] относится к DPLL-алгоритмам. Основная его особенность заключается в том, что выбор переменной для ветвления осуществляется из минимального по мощности множества S переменных, входящих во все дизъюнкты. В этом случае сложность алгоритма составляет $\left(1 + \sum\limits_{i=1}^{k} 2^{|S_i|}\right) poly(|x|)$, где $|x|$ — длина входа, множества S_i — это множества переменных, вычисляемые в ходе работы алгоритма, для которых выполнено: $S = \bigsqcup\limits_{i=1}^{k} S_i$.

2.1.2. PPSZ-алгоритмы

Данный класс алгоритмов основан на результатах Патури, Пудлака, Сакса и Зейна [8]. Разработанный ими алгоритм выбирает случайную перестановку переменных исходной формулы и производит расщепления в соответствующем порядке. Анализ работы такого алгоритма основан на оценке количества переменных, не требующих расщепления. Если эта оценка равна s, то можно ограничиться рассмотрением деревьев расщепления глубиной не более $n - s$. Если выполняющий набор не найден в результате построения дерева глубины $n - s$, то формула невыполнима.

Описание одного из лучших алгоритмов решения задачи о k-выполнимости, при $k = 3$, из данного класса, приведено ниже.

1) Выбрать случайным образом перестановку π из S_n, где S_n означает множество всех перестановок $\{1, \ldots, n\}$.
2) Используя Процедуру 1 с единственным правилом преобразования (1. Удаление единичных дизъюкнтов), построить дерево расщепления глубины $2n/3$. На каждом шаге расщепления выбирать переменную x_i таким образом, чтобы x_i входила в формулу и $\pi(i)$ было наименьшим. Если выполняющий набор найден при помощи процедуры 1, выдать его и остановиться; в противном случае выдать ответ, что формула невыполнима.

Время работы данного алгоритма равно $2^{2n/3} poly(|F|)$. Описанный алгоритм является вероятностным и имеет одностороннюю допустимую вероятность ошибки, то есть выдаваемый выполняющий набор всегда является правильным, а ответ «формула невыполнима» является правильным с вероятностью, по крайней мере, $\frac{1}{poly|F|}$. Повторяя такой алгоритм полиномиальное количество раз, можно понизить вероятность ошибки, так чтобы она стала меньше любой наперед заданной константы.

2.2. Дискретные безусловные алгоритмы

В алгоритмах из данного класса переменные $x_1, \ldots, x_n$ принимают значения из множества $\{0, 1\}$. В данном случае задача о выпол-

нимости здесь рассматривается как задача дискретной безусловной минимизации.

Формулировка задачи: дана булева формула в КНФ: $F = D_1 \wedge \cdots \wedge D_m$, где m — число дизъюнктов в формуле, $x = (x_1, \ldots, x_n)$ — вектор из n булевых переменных. Целевая функция задается следующим образом: $N(x) = \sum\limits_{i=1}^{m} D_i(x)$, где

$$D_i(x) = \prod_{j=1}^{n} Q_{i,j}(x_j);$$

$$Q_{i,j}(x_j) = \begin{cases} 1 - x_j, & \text{если } x_j \text{ входит в дизъюнкт } D_i; \\ x_j, & \text{если } \bar{x_j} \text{ входит в дизъюнкт } D_i; \\ 1, & \text{в противном случае.} \end{cases}$$

Задача: найти $\min_{x \in \{0,1\}^n} N(x)$.

Алгоритмы данного класса основаны на локальном поиске и включают в себя жадный поиск, «осторожный» поиск, случайное блуждание, другие стратегии и комбинации (см. обзор в [4]). Многие из этих алгоритмов хорошо изучены экспериментально, однако хорошие оценки доказаны только для алгоритмов, основанных на случайных блужданиях.

2.2.1. Алгоритмы, основанные на случайных блужданиях

Алгоритмы для задачи о выполнимости, основанные на случайных блужданиях, — это простые вероятностные алгоритмы, которые выбирают начальный набор случайным образом и шаг за шагом двигаются к выполняющему набору: на каждом шаге алгоритм изменяет значение одной переменной, выбранной случайным образом из некоторого дизъюнкта, на противоположное. Такие алгоритмы решают задачу о 2-выполнимости за полиномиальное время, а задачу о k-выполнимости за время $(2 - \frac{2}{k})^n$ с точностью до полиномиального сомножителя, где n — число переменных во входной формуле. В частности, для $k = 3$ эта оценка равна $(4/3)^n$, что является наилучшей из известных на данный момент верхних оценок для алгоритмов, решающих задачу о 3-выполнимости.

Следующий алгоритм (параметризованный функциями α и τ) описывает семейство алгоритмов для задачи о выполнимости, основанных на случайных блужданиях. Для данной входной формулы F, имеющей n переменных, алгоритм производит не более $\alpha(n)$ блужданий, выходящих из случайного начального набора; при этом каждое блуждание состоит из не более чем $\tau(n)$ шагов.

Процедура 2

1) Повторить $\alpha(n)$ раз:
   а) Выбрать случайным образом набор γ (в соответствии с равномерным распределением).
   б) Если γ выполняет F, выдать ответ γ и остановиться. В противном случае повторить следующие инструкции $\tau(n)$ раз:
      - Взять произвольный невыполненный дизъюнкт D_i в F.
      - Выбрать случайным образом (в соответствии с равномерным распределением) переменную x_j среди переменных, входящих в D_i.
      - Модифицировать набор γ, изменив значение переменной x_j в γ на противоположное ($\gamma_j := \gamma_j + 1$).
      - Если модифицированный набор γ выполняет F, выдать ответ γ и остановиться.
2) Выдать ответ «Формула невыполнима» и остановиться.

Процедура 2 при $\alpha(n) = 1$ и $\tau(n) = 2n^2$ представляет собой полиномиальный алгоритм Пападимитриу для задачи 2-выполнимости [9]. Процедура 2 при $\alpha(n) = (2 - \frac{2}{k})^n$ и $\tau(n) = 3n$ представляет собой алгоритм Шонинга сложности $(2 - \frac{2}{k})^n$ для k-выполнимости [12].

Рассмотрим работу описанного алгоритма на формуле в k-КНФ. Предположим, что входная формула имеет выполняющий набор σ, который отличается от начального набора γ значениями ровно i переменных. Заметим, что на каждом шаге алгоритм приближается к σ с вероятностью по крайней мере $1/k$, поскольку невыполненный дизъюнкт всегда содержит хотя бы одну переменную, значения которой в σ и текущем наборе различны. Таким образом, процесс мо-

дификации, начинающийся с γ, связан со следующим равномерным случайным блужданием.

Рассмотрим частицу, блуждающую на интервале $[0, \ldots, n]$. Частица стартует с позиции i в момент времени $t = 0$ и совершает $\tau(n)$ шагов. На каждом шаге, если частица находится в позиции j при $0 < j < n$, она переходит в $j - 1$ с вероятностью $1/k$ и в $j + 1$ с вероятностью $1 - 1/k$. Если $j = 0$, то частица остается в той же позиции с вероятностью 1. Обозначим через $p_{i,t}$ вероятность того, что частица, вышедшая из i достигнет 0 за t шагов. Следующая лемма устанавливает связь между $p_{i,t}$ и вероятностью ошибки алгоритма.

Лемма 1. *Вероятность ошибки Процедуры 2 не превосходит*

$$\exp\left(-\frac{\alpha(n)}{2^n}\sum_{i=0}^{n} C_n^i\right) p_{i,\tau(n)},$$

где C_n^i — число сочетаний из n элементов по k.

Доказательство. Достаточно рассмотреть случай, когда входная формула выполнима. Пусть σ — произвольный выполняющий набор. Рассмотрим одно из $\alpha(n)$ блужданий, совершенных алгоритмом. Для каждого i начальный набор γ отличается от σ значениями ровно i переменных с вероятностью $\frac{C_n^i}{2^n}$. Легко видеть, что такое блуждание находит σ (или другой выполняющий набор, если некоторый предшествующий в ходе блуждания набор оказался выполняющим) с вероятностью по крайней мере $p_{i,\tau(n)}$. Суммируя по всем возможным значениям i, получаем нижнюю оценку:

$$p = \sum_{i=0}^{n} \frac{C_n^i}{2^n} p_{i,\tau(n)}$$

на вероятность того, что блуждание находит выполняющий набор за $\tau(n)$ шагов. Следовательно, вероятность ошибки алгоритма не превосходит

$$(1-p)^{\alpha(n)} = \exp(\alpha(n)\ln(1-p)) \leqslant \exp(-\alpha(n)\cdot p) = \\ = \exp\left(-\alpha(n)\right)\sum_{i=0}^{n}\frac{C_n^i}{2^n}p_{i,\tau(n)}.$$

Лемма доказана.

2.2.2. Алгоритмы, основанные на случайных блужданиях с «кнопкой возврата»

Преобразуем алгоритм, основанный на случайных блужданиях (обозначим его RW — Random Walks) в новый алгоритм, который обозначим RWB — Random Walks with Back button. Новый алгоритм RWB хранит стек истории H, элементами которого являются наборы. Верхний элемент стека соответствует текущему набору. Как и RW, алгоритм RWB совершает не более $\alpha(n)$ блужданий; каждое блуждание состоит из не более чем $\tau(n)$ шагов, но каждый шаг является либо шагом вперед, либо шагом назад. При шаге вперед RWB модифицирует текущий набор так же, как и RW; если модифицированный набор γ выполняет F, то RWB выдает ответ γ и останавливается; в противном случае RWB помещает новый набор γ в стек H. Шаг назад состоит в том, что RWB извлекает верхний элемент из стека H (таким образом, новым верхним элементом становится текущий набор). Начиная с пустого стека, RWB работает следующим образом:

1) Повторить $\alpha(n)$ раз:
   а) Выбрать случайным образом равномерно распределенный набор γ.
   б) Если γ выполняет F выдать ответ γ и остановиться. В противном случае поместить γ в стек H и повторить следующую инструкцию $\tau(n)$ раз:
      - Если H содержит только один элемент, сделать шаг вперед. В противном случае сделать шаг назад с вероятностью b и шаг вперед с вероятностью $1 - b$.
2) Выдать ответ «Формула невыполнима» и остановиться.

Вероятность b называется *вероятностью возврата*.

Чтобы представить работу RWB с помощью одномерных случайных блужданий, модифицируем модель одномерного случайного блуждания для RW (описанную после процедуры 2) следующим образом. Блуждание для RWB хранит стек истории H, элементами которого являются позиции частицы. Каждый шаг блуждания является либо шагом вперед, либо шагом назад. При шаге вперед частица двигается в соответствии с правилами для RW, и ее новая позиция

помещается в стек H. При шаге назад из H извлекается верхний элемент, и частица переходит в позицию, которая после этой операции оказывается верхним элементом стека. В момент времени $t = 0$ стек истории H содержит только начальную позицию i. Затем частица двигается аналогично RWB: если H содержит только один элемент, частица совершает шаг вперед; в противном случае она совершает шаг назад с вероятностью b и шаг вперед с вероятностью $1 - b$. Обозначим через $p_{i,t}^{(b)}$ вероятность того, что частица достигнет 0 не более, чем за t шагов.

Лемма 2. *Вероятность ошибки алгоритма RWB не превосходит*

$$\exp\left(-\frac{\alpha(n)}{2^n}\sum_{i=0}^{n} C_n^i\right) p_{i,\tau(n)}^{(b)}.$$

Доказательство аналогично доказательству Леммы 1.

В работе [14] доказывается, что наличие «кнопки возврата» не может увеличить вероятность достижения 0 более чем на полиномиальный сомножитель, а также что наличие «кнопки возврата» не приводит к существенному проигрышу, если вероятность шага назад не превосходит $\frac{1}{2}$.

2.3. Непрерывные условные алгоритмы

Алгоритмы, относящиеся к классу непрерывных, характеризуются тем, что исходная формула рассматривается над непрерывным множеством таким образом, что решения задачи в непрерывном случае служат булевыми решениями для исходной задачи. Для того, чтобы найти решение, необходимо решить задачу непрерывной условной оптимизации.

Формулировка задачи:дана булева формула в КНФ: $F = D_1 \wedge \cdots \wedge D_m$, где m — число дизъюнктов в формуле, $x = (x_1, \ldots, x_n)$ — вектор из n булевых переменных. Целевая функция задается следующим образом: $f(y) = \sum\limits_{i=1}^{m} c_i(y)$, где

$$c_i(y) = \prod_{j=1}^{n} q_{i,j}(y_j);$$

$$q_{i,j}(y_j) = \begin{cases} |y_j - T|, & \text{если } x_j \text{ входит в дизъюнкт } D_i; \\ |y_j + P|, & \text{если } \bar{x_j} \text{ входит в дизъюнкт } D_i; \\ 1, & \text{в противном случае,} \end{cases}$$

где T и P — положительные константы. Задача: найти $\min_{y \in \mathbb{R}^n} f(y)$, при условии что $c_i(y) = 0 \quad \forall i \in \{1, 2, \ldots, m\}$.

Алгоритмы данного класса основаны на том, что задача о выполнимости булевой формулы, заданной в КНФ, может рассматриваться как частный случай задачи целочисленного программирования. Поэтому проблема выполнимости может быть решена методом релаксации. В частности, к данному классу относятся метод ветвей и границ, метод деления плоскости и метод внутренней точки. Значительное количество алгоритмов основано на поиске лагранжиана целевой функции. Перечисленные методы относятся к классической теории оптимального управления и в данной статье не описываются. Применение алгоритмов непосредственно к задаче выполнимости может быть найдено в работе [4].

2.4. Непрерывные безусловные алгоритмы

Для данной категории алгоритмов существуют различные исходные формулировки задачи. Приведем одну из них.

Формулировка задачи (UniSAT): дана булева формула в КНФ: $F = D_1 \wedge \cdots \wedge D_m$, где m — число дизъюнктов в формуле, $x = (x_1, \ldots, x_n)$ — вектор из n булевых переменных. Целевая функция задается следующим образом: $f(y) = \sum\limits_{i=1}^{m} c_i(y)$, где

$$c_i(y) = \prod_{j=1}^{n} q_{i,j}(y_j);$$

$$q_{i,j}(y_j) = \begin{cases} |y_j - T|, & \text{если } x_j \text{ входит в дизъюнкт } D_i; \\ |y_j + P|, & \text{если } \bar{x_j} \text{ входит в дизъюнкт } D_i; \\ 1, & \text{в противном случае,} \end{cases}$$

где T и P — положительные константы. Задача: найти $\min_{y\in\mathbb{R}^n} f(y)$.

Алгоритмы данного класса преобразуют дискретную формулу в пространстве $\{0,1\}^n$ в функцию над действительными числами. При этом полученная задача решается известными методами глобальной оптимизации. Применение методов глобальной оптимизации для решения задачи выполнимости приведено в работе [4].

В основе непрерывных алгоритмов, описанных выше, лежит переход от булевых переменных к действительным. Этот переход имеет смысл, когда целевая функция, заданная на множестве действительных чисел, «сглаживает» некоторые недопустимые решения, что приводит рассмотрению меньшего числа локальных минимумов. Однако в случае, когда этого не происходит, непрерывные методы имеют вычислительную сложность намного большую, чем дискретные, и их применение невыгодно.

3. Результаты последних лет и практическая сложность решения задачи

В течение последних десяти лет исследование проблемы выполнимости в основном заключалось в модернизации известных классических алгоритмов, в переходе к использованию параллельных вычислений, а также в комбинировании приведенных выше методов. Современные алгоритмы решения задачи о выполнимости, в большинстве своем, относятся к классу DPLL-алгоритмов. При этом авторы стремятся достичь наилучшего быстродействия за счет выбора оптимальных правил преобразования и эвристики выбора переменной.

Кроме того, начиная с 2003 года широкое распространение получили алгоритмы основанные на технике распространения обобщений (Survey Propagation — SP) [1], которые относятся к классу дискретных безусловных алгоритмов. SP-алгоритмы преобразуют исходную формулу в граф, вершинами которого делятся на два класса: дизъюнкты и переменные, входящие в формулу. Ребрам графа, соединяющим вершины дизъюнктов и вершины переменных, в ходе работы алгоритма приписываются значения, позволяющие сделать выводы

относительно выполнимости соответствующих дизъюнктов, и зафиксировать значения переменных выполняющего набора. SP-алгоритмы основаны на вероятностном методе обмена сообщениями между вершинами графа. Данный метод обобщает известные процедуры распространения предупреждений (Warning Propagation — WP) и распространения доверия (Belief Propagation — BP). Алгоритмы из данного семейства до начала работы требуют задания максимального числа итераций и требуемой точности, так как при наличии петель в графе существует вероятность зацикливания.

Что касается практической реализации алгоритмов, за последние годы создавались всё более быстродействующие программы (см. рис. 1 из [10]).

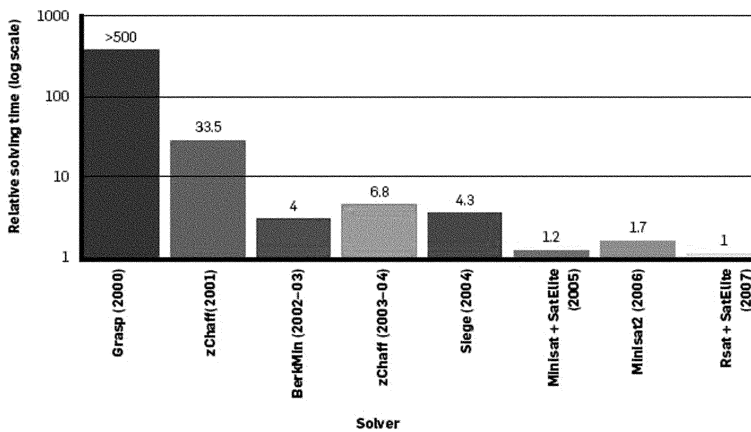

Рис. 1. Быстродействие решателей задачи выполнимости за последние годы (по логарифмической шкале).

Было создано сообщество исследователей проблемы выполнимости, ежегодно проводящее конкурсы с целью определить наиболее быстрый решатель задачи. Результаты конкурса, а также актуальная информация о проблеме выполнимости регулярно публикуются на портале `http://www.satlive.org`.

4. Заключение

Несмотря на то, что основные методы решения задачи пропозициональной выполнимости были заложены ещё в 60–70-х годах прошлого века, эта тема имеет активное развитие и сегодня. Статистические данные о практической работе алгоритмов показывают, что для значительной части формул задача всё же может быть решена за полиномиальное время.

Автор работы выражает признательность В. А. Носову за научное руководство.

Список литературы

[1] Braunstein A., Mezard M., Zecchina R. Survey propagation: An algorithm for Satisfiability // Random Structures and Algorithms. 27. 2005. P. 201–226.

[2] Davis M., Logeman G., Loveland D. A machine program for theorem-proving // Communications of the ACM. 5 (7). 1962. P. 394–397.

[3] Davis M., Putman H. A computing procedure for quantification theory // Journal of the ACM. 7 (3). 1960. P. 201–215.

[4] Du D., Gu J., Pardalos P. M. Satisfiability problem: theory and applications // Proceedings of a DIMACS workshop. March 11–13, 1996.

[5] Dubois O. On the r, s-satisfiability problems and a conjecture of Tovey // Discrete Appl. Math. 1990. V. 26. P. 51–60.

[6] Kullmann O. New methods for 3-SAT decision and worst-case analysis // Theoretical Computer Science. 223 (1–2). 1999. P. 1–72.

[7] Kullmann O., Luckhardt. Deciding propositional tautologies: Algorithms and their complexity / Preprint. `http://www.cs.toronto.edu/~kullmann`

[8] Paturi R., Pudlák P., Saks M. E., Zane F. An improved exponential-time algorithm for k-SAT // Proceedings of FOCS'98. 1998. P. 628–637.

[9] Papadimitriou C. H. On selecting a satisfying truth assignment // Proceedings of FOCS'91. 1991. P. 163–169.

[10] Malik S., Zhang L. Boolean satisfiability. From theoretical hardness to practical success // Communications of the ACM. No. 8. Vol. 52. 2009.

[11] Schaefer T. J. The complexity of satisfiability problems // Proceedings of the 10th ACM Symposium on Theory of Computing. 1978. P. 216–226.

[12] Shöning U. A probabilistic algorithm for k-SAT and constraint satisfaction problems // Proceedings of FOCS'99. 1999. P. 410–414.

[13] Алексеев В. Б., Носов В. А. NP-полные задачи и их полиномиальные варианты. Обзор // Обозрение прикладной и промышленной математики. Т. 4, вып. 2. 1997. С. 165–193.

[14] Всемирнов М. А., Гирш Э. А., Данцин Е. Я., Иванов С. В. Алгоритмы для пропозициональной выполнимости и верхние оценки их сложности // Теория сложности вычислений. VI. Зап. научн. сем. ПОМИ. 277. СПб.: ПОМИ, 2001.

[15] Гизунов С. А., Носов В. А. Сложность распознавания классов Шефера // Вестник МГУ. Сер. 1. 1995.

[16] Поцелуевская Е. А. Полиномиальные случаи решения задачи об F-выполнимости булевых формул // Интеллектуальные системы. Т. 12. 2008. С. 351–362.