ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

УДК 519.712

ОБ АСИМПТОТИКЕ РЕШЕНИЙ РЕКУРРЕНТНЫХ СООТНОШЕНИЙ СПЕЦИАЛЬНОГО ВИДА И ТЕХНИКЕ КУЛЬМАНА — ЛЮКХАРДТА

В. В. Быкова

Институт математики и фундаментальной информатики Сибирского федерального университета, г. Красноярск, Россия

E-mail: bykvalen@mail.ru

Обсуждаются проблемы решения рекуррентных соотношений, возникающих в анализе вычислительной сложности рекурсивных алгоритмов. Класс рассматриваемых алгоритмов ограничен алгоритмами расщепления, а именно DPLL-алгоритмами, предназначенными для решения задачи пропозициональной выполнимости. Исследована техника Кульмана — Люкхардта, традиционно применяемая при исследовании вычислительной сложности алгоритмов расщепления. Предложена теорема, устанавливающая верхние оценки времени выполнения DPLL-алгоритма в случае сбалансированного расщепления. Теорема расширяет возможности техники Кульмана — Люкхардта, так как учитывает неоднородность рекуррентного соотношения, описывающего время работы DPLL-алгоритма.

Ключевые слова: *вычислительная сложность рекурсивных алгоритмов, алгоритмы расщепления, решение рекуррентных соотношений.*

Введение

Задача пропозициональной выполнимости (англ. вариант — SATISFIABILITY, или коротко SAT) является одной из известных задач дискретной математики и теории сложности вычислений. Это задача о выполнимости формулы логики высказываний. Для формул, записанных в конъюнктивной нормальной форме, задача SAT является NP-полной (С. А. Кук, 1971) [1]. Многие проблемы из класса NP естественным образом сводятся к данной задаче [2].

Задача SAT формулируется следующим образом. Пусть X — конечное множество переменных, принимающих значения `false` и `true`. Формула F в конъюнктивной нормальной форме (КНФ) представляет собой конъюнкцию конечного числа элементарных дизъюнкций. Требуется установить, существует ли для входной формулы F, заданной в КНФ над множеством переменных X, набор значений переменных, при котором каждая дизъюнкция формулы F принимает значение `true` (такой набор принято называть выполняющим). Формула считается выполнимой, если для неё существует хотя бы один выполняющий набор, и невыполнимой, если выполняющего набора не существует.

Входной формуле F обычно приписывают числовую величину, которая принимает неотрицательные целые значения (принадлежит $\mathbb{Z}_+$) и характеризует размерность этой формулы. Такие величины называют мерами сложности формулы логики высказываний. В качестве основных мер сложности рассматривают: $N = |X|$ — число

переменных; K — число дизъюнкций; $L = |F|$ — длину формулы F, как количество входящих в неё литералов (переменных или их отрицаний). Относительно этих мер сложности традиционно определяют вычислительную сложность (время работы) исследуемого алгоритма решения задачи SAT.

Задача пропозициональной выполнимости NP-полная, поэтому маловероятно, что она может быть решена за полиномиальное время. Так, алгоритм полного перебора, анализирующий все 2^N различных наборов значений N переменных, устанавливает тривиальную верхнюю оценку вычислительной сложности задачи SAT в худшем случае. Поиск алгоритмов, работающих быстрее полного перебора, активно ведётся с начала 60-х годов прошлого века и до настоящего времени. Среди них наиболее известными являются алгоритмы расщепления [3]. Алгоритмы расщепления — это декомпозиционные алгоритмы. Они сводят задачу SAT для входной формулы F к конечному числу подзадач, каждая из которых — это SAT для формул меньшей сложности, чем сложность F. Причём это сведение может быть детерминированным (алгоритм производит рекурсивные вызовы самого себя для формул меньшей сложности, полученных при расщеплении F) или вероятностным (случайный выбор одной из образованных после расщепления формул). Детерминированные алгоритмы расщепления принято назвать DPLL-алгоритмами (по первым буквам фамилий их авторов: M. Davis, H. Putman, G. Logemann, D. Loveland) [4, 5]. Значительная часть алгоритмов, разработанных за последние пятьдесят лет для задачи SAT, базируется на идеях, которые заложены в DPLL-алгоритмах. Далее везде под алгоритмами расщепления будем иметь в виду DPLL-алгоритмы.

В данной работе алгоритмы расщепления рассматриваются с точки зрения вычислительной сложности. Будем использовать традиционные для анализа алгоритмов асимптотические обозначения [1]. Например, для вещественнозначных функций $q(n)$ и $h(n)$ от $n \in \mathbb{Z}_+$ запись $q(n) = \Theta[h(n)]$ означает, что существуют такие константы $c_1, c_2 > 0$ и такое $n_0 \in \mathbb{Z}_+$, что $c_1 h(n) \leqslant q(n) \leqslant c_2 h(n)$ при $n \geqslant n_0$. Аналогично, пишут $q(n) = \mathrm{O}[h(n)]$, если найдётся константа $c > 0$ и такое $n_0 \in \mathbb{Z}_+$, что $q(n) \leqslant c \cdot h(n)$ при всех $n \geqslant n_0$. Точно так же $q(n) = \mathrm{o}[h(n)]$ тогда и только тогда, когда для любой положительной константы c существует такое $n_0 \in \mathbb{Z}_+$, что всегда $q(n) < c \cdot h(n)$ при $n \geqslant n_0$. Все эти обозначения предполагают, что функции $q(n)$ и $h(n)$ неотрицательные асимптотически, т. е., по крайней мере, начиная с некоторого значения n.

В п. 1 работы приводится общая схема функционирования алгоритма расщепления; в п. 2 определяется вид рекуррентного соотношения, описывающего время работы DPLL-алгоритма, исследуются вычислительные возможности получения для него решения в явном виде или в виде асимптотических оценок, а также анализируется техника Кульмана — Люккардта, традиционно применяемая при рассмотрении вычислительной сложности алгоритмов расщепления; в п. 3 представляется основной результат — теорема, которая устанавливает верхние оценки времени выполнения DPLL-алгоритма в случае сбалансированного расщепления. В отличие от техники Кульмана — Люккардта, данная теорема определяет асимптотические оценки в явном виде и позволяет учитывать временные затраты алгоритма, которые не связаны с его рекурсивными вызовами.

1. Схема работы алгоритма расщепления

В алгоритмах расщепления снижение меры сложности входной формулы осуществляется с помощью редуцирования и расщепления этой формулы. В общем виде схему работы DPLL-алгоритма можно представить следующим образом.

Вход: формула F в КНФ над X.
Ответ: «Выполнима» / «Невыполнима».

1. Редуцировать F, т. е. преобразовать формулу F в формулу F_0 меньшей сложности с помощью конечного набора правил редукции. Эти правила применять к F до тех пор, пока хотя бы одно из них применимо.
2. Если задача SAT тривиально решается для F_0, то выдать соответствующий ответ.
3. Выбрать переменную $x \in X$ по определённому правилу. Таких переменных может быть несколько. Их число определяется правилом расщепления, употребляемом на следующем шаге.
4. Выполнить расщепление формулы F_0 на конечное число формул меньшей сложности по заданному закону — правилу расщепления, используя различные значения переменных, выбранных на шаге 3.
5. Осуществить рекурсивные вызовы данного алгоритма применительно к полученным формулам. Выдать ответ, основываясь на результатах, возвращённых рекурсивными вызовами: если хотя бы один из рекурсивных вызовов вернул выполняющий набор, то ответ — «Выполнима», в противном случае — «Невыполнима».

Таким образом, DPLL-алгоритм — рекурсивный алгоритм, вызывающий сам себя на формулах меньшей сложности. Рекурсивные вызовы прекращаются, как только задача SAT решается тривиально, т. е. если в результате преобразований получается формула, для которой решение задачи может быть найдено за полиномиальное время. В частности, к полиномиально разрешимым случаям приводят следующие классы формул: пустые формулы (они не содержат дизъюнкций вообще, всякая такая формула интерпретируется как `true`); формулы в 2-КНФ (каждая их дизъюнкция содержит не более двух литералов); хорновские формулы (в них каждая дизъюнкция содержит переменные с отрицанием, кроме одной переменной) и ряд других классов формул [6, 7]. К трудоёмкости шагов DPLL-алгоритма предъявляют ряд естественных требований. Суть их в том, что все действия, не связанные с рекурсивными вызовами (это шаги 1–4 алгоритма), должны выполняться за полиномиальное время относительно выбранной меры сложности формул.

Различные расщепляющие алгоритмы отличаются друг от друга в основном правилами редукции, стратегией выбора переменных на шаге 3, правилами расщепления и используемой мерой сложности формул. Уже предложено и доказано значительное число правил редукции, наиболее известные из них можно найти в работах [3 – 5]. Среди них: «единичная дизъюнкция», «чистый литерал», «резолюция», «поглощение» и др. Типичные стратегии по выбору переменных для расщепления — это, чаще всего, эвристики вида: «взять любую переменную из самой короткой дизъюнкции», «предпочтение отдать переменной, входящей в наибольшее число дизъюнкций». Правила расщепления также могут быть различными. Например, простейшее правило расщепления сводится к замене формулы F_0 на две формулы

$$F_1 = F_0[x] \text{ и } F_2 = F_0[\neg x]. \tag{1}$$

При этом формула F_1 получается из формулы F_0 присваиванием значения `true` переменной x. Это влечёт удаление всех дизъюнкций, содержащих x без отрицания, и удаление литерала $\neg x$ в оставшихся дизъюнкциях. Формула F_2 получается аналогичным образом — присваиванием переменной x значения `false` и дальнейшим соответствующим упрощением полученной формулы. Таким образом, при данном правиле расщепления DPLL-алгоритм выполняет два рекурсивных вызова для формул F_1 и F_2

соответственно. Своеобразные правила расщепления, например такие, как

$$F_1 = F_0[x, y], \quad F_2 = F_0[x, \neg y], \quad F_3 = F_0[\neg x]; \tag{2}$$

$$F_1 = F_0[x, y], \quad F_2 = F_0[x, \neg y], \quad F_3 = F_0[\neg x, y], \quad F_4 = F_0[\neg x, \neg y], \tag{3}$$

порождают три и более рекурсивных вызова алгоритма. Во всех случаях уменьшение меры сложности образованных формул в сравнении с F достигается за счёт редуцирования F и расщепления F_0 по выбранным переменным.

Важно отметить следующие факты. Если расщепление выполняется по правилу $F_1 = F_0[x]$ и $F_2 = F_0[\neg x]$, то DPLL-алгоритм в худшем случае (например, для невыполнимой формулы, к которой неприменимы рассматриваемые правила редукции) перебирает все 2^N различных наборов значений N переменных. Как будет показано ниже, подобное наблюдается и при других правилах расщепления, когда отсутствует этап редуцирования и в качестве меры сложности входной формулы выступает число переменных. Это вполне согласуется с тем, что задача SAT является NP-полной. Для данной задачи (когда отсутствуют какие-либо ограничения на длину и структуру дизъюнкций во входной формуле) до сих пор не найдено методов решения, в том числе и DPLL-алгоритмов, со временем работы $\mathrm{O}(c^N)$, где $c < 2$. Между тем известны алгоритмы расщепления с оценками вычислительной сложности $\mathrm{O}(1{,}238823^K)$, $\mathrm{O}(1{,}073997^L)$ относительно числа K дизъюнкций и длины L входной формулы соответственно [8]. Область применения DPLL-алгоритмов не ограничивается задачей SAT. Алгоритмы расщепления успешно применяются при решении многих NP-полных задач, например MAX-SAT, MAX-2-SAT [9]. Существует большое количество научных публикаций, посвященных этому классу алгоритмов. Более того, имеется много работ, описывающих DPLL-алгоритмы для SAT, где каждая следующая улучшает результат предыдущей за счёт введения иной меры сложности формул, оригинальных правил редукции и расщепления, более тщательного анализа алгоритмов. При этом предлагаемые алгоритмы сравниваются между собой преимущественно с точки зрения O-оценки (верхней асимптотической оценки) времени их работы в худшем случае.

2. Рекуррентные соотношения в анализе алгоритмов расщепления: асимптотика решений и техника Кульмана — Люкхардта

В общем случае анализ вычислительной сложности DPLL-алгоритмов представляет собой весьма непростую задачу ввиду их рекурсивности. В настоящее время основным математическим инструментом исследования вычислительной сложности рекурсивных алгоритмов является метод рекуррентных соотношений [1, 10]. Идея данного метода состоит в построении и решении рекуррентного соотношения, которому удовлетворяет функция, описывающая время работы рассматриваемого алгоритма. Однако данный метод не универсален. Существует несколько серьёзных препятствий для его практического воплощения. Во-первых, для построения рекуррентного соотношения необходимо, чтобы преобразование, уменьшающее значение параметра рекурсии, было линейным относительно этого параметра [11]. Во-вторых, если рекуррентное соотношение всё же найдено, то нет никакой гарантии, что удастся установить явную форму или асимптотическую оценку его решения. Причина в том, что общих методов решения рекуррентных соотношений до сих пор не найдено. Известны лишь методы решения некоторых классов рекуррентных соотношений [12, 13]. Интересно выяснить, какой вид имеет рекуррентное соотношение, описывающее время работы DPLL-алгоритма, и какие вычислительные возможности имеются для нахождения его решения? Дадим ответы на поставленные вопросы.

Пусть неотрицательная целая величина n — некоторая мера сложности входной формулы F (например, количество переменных или число дизъюнкций). Предположим, что DPLL-алгоритм вначале редуцирует, а затем расщепляет формулу F на $m > 1$ формул $F_1, F_2, \ldots, F_m$ сложности $n_1, n_2, \ldots, n_m$ соответственно, $n_m \leqslant \ldots \leqslant n_2 \leqslant n_1 < n$. Пусть $T(n)$ — монотонно неубывающая функция от n, представляющая общее время работы алгоритма расщепления для формулы F сложности n; её областью значений является множество неотрицательных вещественных чисел, а областью определения — множество неотрицательных целых чисел. Из схемы работы DPLL-алгоритма следует, что значение $T(n)$ складывается из времени выполнения рекурсивных вызовов для формул $F_1, F_2, \ldots, F_m$, а также из временных затрат, не связанных с рекурсивными вызовами:

$$T(n) = T(n_1) + T(n_2) + \cdots + T(n_m) + f(n).$$

Здесь $f(n)$ — функция полиномиального порядка роста [14], которая как раз и учитывает нерекурсивные затраты DPLL-алгоритма, т. е. время, затрачиваемое им на редуцирование F, выбор переменных для расщепления, построение решения для F из решений для $F_0, F_1, F_2, \ldots, F_m$ и на прочие подобные действия. Выражение для $T(n)$ можно представить так:

$$T(n) = T(n - t_1) + T(n - t_2) + \cdots + T(n - t_m) + f(n), \tag{4}$$

где $t_1 = n - n_1$, $t_2 = n - n_2, \ldots, t_m = n - n_m$, $1 \leqslant t_1 \leqslant t_2 \leqslant \ldots \leqslant t_m < n$. В публикациях по DPLL-алгоритмам вектор $(t_1, t_2, \ldots, t_m)$ обычно называют *вектором расщепления*. Далее будем придерживаться этой терминологии. Вектор расщепления назовём *сбалансированным*, если все его элементы равны, и *максимально несбалансированным*, если в нём нет ни одной пары равных элементов.

Поскольку в векторе расщепления возможны равные элементы, то равенство (4) можно преобразовать к следующему неоднородному линейному рекуррентному соотношению порядка $k \geqslant 1$ с постоянными коэффициентами:

$$T(n) = a_1 T(n-1) + a_2 T(n-2) + \cdots + a_k T(n-k) + f(n), \quad n \geqslant k; \tag{5}$$

$$T(n) = \Theta(1), \quad 0 \leqslant n \leqslant k-1. \tag{6}$$

Начальные условия (6) свидетельствуют о том, что время работы алгоритма расщепления ограничено сверху и снизу константами для формулы сложности не более $k-1$. В соотношении (5) коэффициенты $a_1, a_2, \ldots, a_k$ — положительные целые константы, не равные нулю одновременно. Для рекуррентного соотношения порядка k естественными являются условия $a_1, a_2, \ldots, a_{k-1} \geqslant 0$, $a_k \geqslant 1$.

Общих методов нахождения явного вида решения рекуррентного соотношения (5) неизвестно. Между тем при $f(n) \equiv 0$ к нему возможно применение классической теоремы Пуанкаре — Перрона [15, 16]. Согласно теореме Пуанкаре — Перрона, общий вид решения однородного линейного рекуррентного соотношения с постоянными коэффициентами устанавливается через корни соответствующего характеристического многочлена. Так, если $\alpha_1, \alpha_2, \ldots, \alpha_k$ — простые (некратные) корни уравнения

$$x^k - a_1 x^{k-1} - a_2 x^{k-2} - \cdots - a_k = 0, \tag{7}$$

то решение рекуррентного соотношения

$$T(n) = a_1 T(n-1) + a_2 T(n-2) + \cdots + a_k T(n-k)$$

определяется формулой

$$T(n) = c_1\alpha_1^n + c_2\alpha_2^n + \cdots + c_k\alpha_k^n, \tag{8}$$

где c_1, c_2, …, c_k — постоянные величины, значения которых вычисляются, исходя из начальных условий (6). Таким образом, если α — наибольший положительный вещественный корень уравнения (7), то, согласно (8), верна асимптотическая оценка

$$T(n) = O(\alpha^n). \tag{9}$$

Следует заметить, что (7) по правилу перемены знаков в последовательности коэффициентов алгебраического уравнения [17] имеет ровно один положительный вещественный корень. В силу единственности он простой. Именно этот корень и выступает в роли α. В работах, посвящённых DPLL-алгоритмам, величину α называют числом расщепления и пишут $\alpha = \alpha(t_1, t_2, \ldots, t_m)$, подчёркивая тем самым зависимость значения α от вектора расщепления $(t_1, t_2, \ldots, t_m)$. Применение к (7) метода Лагранжа для определения границ положительных вещественных корней алгебраического уравнения [17] приводит к оценке $\alpha \leqslant 2$, в которой равенство достижимо в ряде случаев. В частности,

$$\alpha(1,1) = \alpha(1,2,2) = \alpha(2,2,2,2) = 2,$$

в чём нетрудно убедиться, решив надлежащие характеристические уравнения. Примечательно, что векторы расщепления $(1,1)$, $(1,2,2)$, $(2,2,2,2)$ отвечают DPLL-алгоритмам, в которых отсутствует этап редуцирования, в роли n выступает число N переменных, а расщепление выполняется по правилам (1)–(3) соответственно. В данном случае оценка (9) принимает вид $\mathrm{O}(2^N)$. Это подтверждает приведённый в п. 1 факт, что DPLL-алгоритмы не способны улучшить время работы алгоритма полного перебора, если сложность формул измерять числом входящих в них пропозициональных переменных.

Для нахождения чисел расщепления нет общих вычислительных формул, поэтому они находятся преимущественно с помощью численных методов [17], исходя из вектора расщепления. В ряде частных случаев можно воспользоваться свойствами чисел расщепления, которые легко доказываются. Некоторые из них приведены в работе [9]. Например, для сбалансированного вектора расщепления с $m > 1$ равными элементами справедливо равенство

$$\alpha(\underbrace{k, k, \ldots, k}_{m \text{ раз}}) = m^{1/k} = 2^{(\log_2 m)/k},$$

а для максимально несбалансированного вектора расщепления $(1, 2, \ldots, m)$ верно строгое неравенство $\alpha(1, 2, \ldots, m) < 2$. Кроме того, всегда

$$\alpha(t_1, t_2, \ldots, t_m) \leqslant \alpha(p_1, p_2, \ldots, p_m),$$

если $t_i \geqslant p_i$ для всех i. Отсюда при $m > 1$

$$m^{1/m} = \alpha(\underbrace{m, m, \ldots, m}_{m \text{ раз}}) < \alpha(1, 2, \ldots, m) < 2,$$

т. е. сбалансированный вектор расщепления приводит к меньшему значению α, чем максимально несбалансированный с тем же числом элементов.

Разработчики DPLL-алгоритмов для анализа вычислительной сложности этих алгоритмов традиционно применяют технику Кульмана — Люкхардта [18, 19]. Многие оценки для задачи SAT и родственных с ней NP-полных задач получены с помощью данной техники. Её подробное описание дано в работе [3]. Исследование техники Кульмана — Люкхардта показало, что она полностью базируется на теореме Пуанкаре — Перрона и приведённых выше особенностях алгебраического уравнения (7), хотя её описание не содержит явных ссылок на эти известные математические факты [18, 19]. Неоднородность рекуррентного соотношения (5) в технике Кульмана — Люкхардта игнорируется. Чтобы как-то обозначить существование этой неоднородности, разработчиками DPLL-алгоритмов всегда делается оговорка, что оценка времени работы алгоритма получена с точностью до полиномиального сомножителя, для чего используется запись

$$T(n) = \mathrm{poly}(n) \cdot \mathrm{O}(\alpha^n). \tag{10}$$

Степень полинома в (10) не уточняется (в общем случае это сделать невозможно, ведь число расщепления α, как правило, приходится находить численно).

Техника Кульмана — Люкхардта имеет свои достоинства, которые способствуют её широкому практическому применению. Во-первых, данная техника служит единым инструментом, с помощью которого оценивается время работы алгоритмов расщепления. Это даёт возможность сопоставлять такие алгоритмы между собой. Во-вторых, численный характер техники позволяет её автоматизировать. Известны программы для автоматического доказательства верхних оценок DPLL-алгоритмов [20]. В-третьих, техника Кульмана — Люкхардта максимально аккумулирует математические возможности вычисления оценок для решений рекуррентных соотношений вида (5). Одним из возможных направлений развития данной техники является рассмотрение случаев, когда $f(n) \not\equiv 0$. Игнорирование неоднородности в (5) — это основной недостаток техники Кульмана — Люкхардта. Между тем учёт этой неоднородности чрезвычайно важен с точки зрения развития идей, заложенных в DPLL-алгоритмах. В самом деле, постоянное расширение списка правил редукции и расщепления направлено на уменьшение числа расщепления α, а значит, и второго сомножителя правой части (10). Однако это неизбежно приводит к увеличению нерекурсивных расходов DPLL-алгоритма и первого сомножителя в правой части (10). Таким образом, принимая во внимание допустимые значения n, целесообразно сопоставлять вклад обоих сомножителей в значение функции $T(n)$ (ведь при $1 \leqslant n \leqslant 100$ значение $10n^3$ гораздо больше, чем $1{,}1^n$). Предлагаемая ниже теорема даёт возможность учитывать неоднородность в (5) для одного частного случая, когда расщепление является сбалансированным и $f(n) = \Theta(n^\tau)$, где τ — неотрицательная вещественная константа.

3. Теорема для сбалансированного расщепления

Расщепление формулы F является сбалансированным относительно меры n, если ему отвечает сбалансированный вектор расщепления

$$(\underbrace{k, k, \ldots, k}_{a \text{ раз}}),$$

где a — число формул, получаемых при расщеплении, и каждая из этих формул характеризуется мерой сложности $(n - k)$, где $a > 1$. Заметим, что здесь число формул, получаемых в процессе расщепления, обозначено через a (вместо m, используемого ранее).

Рекуррентное соотношение (5) при сбалансированном расщеплении значительно упрощается, поскольку $a_1 = a_2 = \cdots = a_{k-1} = 0$, $a_k = a$. В данном случае (5), (6) можно записать так:

$$T(n) = aT(n-k) + f(n), \text{если } n \geqslant k; \tag{11}$$

$$T(n) = \Theta(1), \text{если } 0 \leqslant n \leqslant k-1. \tag{12}$$

Для такого неоднородного линейного рекуррентного соотношения возможно получение асимптотических оценок в явном виде, что подтверждает следующая теорема.

Теорема 1. Пусть задано рекуррентное соотношение (11) с начальными условиями (12), где $a > 1$, $k \geqslant 1$ — целые константы. Пусть $\tau \geqslant 0$ — вещественная константа. Тогда при любом достаточно большом значении n верны оценки

$$T(n) = \mathrm{O}(n^\tau a^{n/k}), \text{если } f(n) = \Theta(n^\tau); \tag{13}$$

$$T(n) = \mathrm{O}(a^{n/k}), \text{если } f(n) \equiv 0. \tag{14}$$

Доказательство. Покажем справедливость данной теоремы путём рассмотрения ряда возможных случаев.

С л у ч а й 1. Пусть $f(n) \equiv 0$. Оценка (14) прямо следует из (9). В данном случае $\alpha = a^{1/k}$.

С л у ч а й 2. Предположим, что n кратно k, т. е. представимо в виде $n = kp$, $p \in \mathbb{Z}_+$. Пусть также $f(n) = bn^\tau$, где $b > 0$, $\tau \geqslant 0$ — вещественные константы.

Введём функцию $Q(p) = T(kp)$. Тогда $T(kp-k) = T(k(p-1)) = Q(p-1)$ и (11),(12) можно представить следующим образом:

$$T(kp) = aT(kp-k) + bk^\tau p^\tau, \quad \text{если } p > 0,$$
$$T(kp) = \Theta(1), \quad \text{если } p = 0,$$

или

$$Q(p) = aQ(p-1) + bk^\tau p^\tau, \quad \text{если } p > 0, \tag{15}$$
$$Q(p) = \Theta(1), \quad \text{если } p = 0.$$

Подстановка соотношения (15) самого в себя при условии, что $Q(0) = \Theta(1)$, даёт

$$Q(p) = a^p\Theta(1) + bk^\tau S(a,p,\tau), \tag{16}$$

где $S(a,0,\tau) = 0$, а при $p > 0$

$$S(a,p,\tau) = a^{p-1} \cdot 1^\tau + a^{p-2} \cdot 2^\tau + \cdots + a^0 \cdot p^\tau = \sum_{i=1}^{p} a^{p-i} i^\tau.$$

Найдём значение конечной суммы $S(a,p,\tau)$. При $\tau = 0$ и $a > 1$ справедливо равенство

$$S(a,p,0) = \sum_{i=1}^{p} a^{p-i} i^0 = a^{p-1} + a^{p-2} + \cdots + a^0 = \frac{a^p - 1}{a-1}.$$

Таким образом, при $\tau = 0$ для рекуррентного соотношения (15) верна оценка

$$Q(p) = a^p\Theta(1) + bk^\tau S(a,p,\tau) = a^p\Theta(1) + b\frac{a^p-1}{a-1} = \Theta(a^p).$$

Поскольку $p = n/k$, то

$$Q(p) = T(kp) = T(n) = \Theta(a^{n/k}),$$

что отвечает оценке (13), ведь Θ-оценка функции всегда влечет справедливость соответствующей O-оценки той же функции.

Для произвольного вещественного $\tau \geqslant 0$ значение суммы $S(a, p, \tau)$ установить крайне сложно. Между тем значение $S(a, p, \tau)$ можно оценить сверху. В работе автора [10] доказано неравенство

$$S(a, p, \tau) \leqslant p^\tau \frac{a^p - 1}{a - 1},$$

из которого следует, что

$$Q(p) = a^p\Theta(1) + bk^\tau S(a, p, \tau) \leqslant a^p\Theta(1) + bk^\tau p^\tau \frac{a^p - 1}{a - 1},$$

$$T(kp) \leqslant a^p\Theta(1) + bk^\tau p^\tau \frac{a^p - 1}{a - 1}. \tag{17}$$

Принимая во внимание, что $k^\tau p^\tau = n^\tau$ и $p = n/k$, из (17) вытекает оценка

$$T(n) = \mathrm{O}(n^\tau a^{n/k}), \tag{18}$$

которая совпадает с (13).

С л у ч а й 3. Пусть по-прежнему $n = kp$, $p \in \mathbb{Z}_+$. Предположим теперь, что $f(n) = \Theta(n^\tau)$, $\tau \geqslant 0$. Функцию $f(n) = \Theta(n^\tau)$ можно записать в виде $f(n) = bn^\tau + \mathrm{o}(n^\tau)$, где $b > 0$ — некоторая вещественная константа. Тогда в правой части равенства (16) появляется дополнительное слагаемое

$$\mathrm{o}(k^\tau p^\tau \sum_{i=1}^{p} a^{p-i}) = \mathrm{o}\left(k^\tau p^\tau \frac{a^p - 1}{a - 1}\right) = \mathrm{o}\left(k^\tau p^\tau a^p\right). \tag{19}$$

С учётом этого неравенство (17) преобразуется к виду

$$T(kp) \leqslant a^p\Theta(1) + bk^\tau p^\tau \frac{a^p - 1}{a - 1} + \mathrm{o}\left(k^\tau p^\tau a^p\right),$$

из которого следует, что дополнительное слагаемое в (16) не изменяет оценку (18).

С л у ч а й 4. В предыдущих двух случаях предполагалось, что n кратно k. Однако это предположение можно снять, исходя из монотонности функции $T(n)$, поскольку если n не кратно k, то всегда найдется ближайшее сверху n^*, такое, что $n^* = kp$, $p \in \mathbb{Z}_+$, и $T(n) \leqslant T(n^*)$. Очевидно, что в этой ситуации порядок роста функции $T(n)$, выраженный верхней асимптотической оценкой (13), остаётся прежним для любого целого неотрицательного n. ■

Доказанная теорема расширяет возможности техники Кульмана — Люккардта, так как учитывает неоднородность рекуррентного соотношения, описывающего время работы DPLL-алгоритма в случае сбалансированного расщепления. Теорема может быть применена и тогда, когда расщепление не является сбалансированным. Для этого необходимо сбалансировать вектор расщепления и воспользоваться монотонностью функции $T(n)$. Конечно, полученная оценка может оказаться хуже реальной, но зато она

определяется в явном виде и без привлечения процедуры нахождения числа расщепления. Покажем это на примере. Пусть задано рекуррентное соотношение

$$T(n) = 2T(n-6) + 2T(n-7) + n^3. \tag{20}$$

В данном случае вектор расщепления равен (6, 7). Техника Кульмана — Люкхардта приводит к оценке

$$T(n) = \mathrm{poly}(n) \cdot \mathrm{O}(1{,}24^n).$$

Поскольку $T(n-7) \leqslant T(n-6)$, то возможен переход к сбалансированному вектору расщепления (6, 6) и соответствующему рекуррентному соотношению

$$T(n) \leqslant 4T(n-6) + n^3.$$

Применение теоремы 1 (здесь $a = 4$, $k = 6$) даёт оценку

$$T(n) = \mathrm{O}(n^3 1,26^n),$$

которая учитывает неоднородность соотношения (20). Какой из полученных оценок отдать предпочтение, зависит от специфики исследуемого алгоритма расщепления и входных данных решаемой задачи.

ЛИТЕРАТУРА

1. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: Вильямс, 2012.
2. *Отпущенников И. В., Семёнов А. А.* Технология трансляции комбинаторных проблем в булевы уравнения // Прикладная дискретная математика. 2011. № 1(11). С. 96–115.
3. *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. СПб., 2001. Т. 277. С. 14–46.
4. *Davis M., Logemann G., and Loveland D.* A machine program for theorem-proving // Comm. ACM. 1962. No. 5(7). P. 394–397.
5. *Davis M. and Putman H.* A computing procedure for quantification theory // J. ACM. 1960. No. 7(3). P. 201–215.
6. *Franco J. and Gelder A. V.* A perspective on certain polynomial-time solvable classes of Satisfiability // Discr. Appl. Math. 2003. V. 125. Iss. 2–3. P. 177–214.
7. *Алексеев В. Б., Носов В. А.* NP-полные задачи и их полиномиальные варианты. Обзор // Обозрение прикладной и промышленной математики. 1997. Т. 4. Вып. 2. С. 165–193.
8. *Hirsch E. A.* New worst-case upper bounds for SAT // J. Automated Reasoning. 2000. No. 24(4). P. 397–420.
9. *Куликов А. С., Куцков К.* Новые верхние оценки для задачи максимальной выполнимости // Дискретная математика. 2009. № 21(1). С. 139–157.
10. *Быкова В. В.* Математические методы анализа рекурсивных алгоритмов // Журнал Сибирского федерального университета. Сер. Математика и физика. 2008. № 1(3). С. 236–246.
11. *Головешкин В. А., Ульянов М. В.* Теория рекурсии для программистов. М.: Физматлит, 2006.
12. *Гельфонд А. О.* Исчисление конечных разностей. М.: Физматлит, 1959.
13. *Грэхем Р., Кнут Д., Паташник О.* Конкретная математика. Основание информатики. М.: Бином. Лаб. знаний, 2006.
14. *Быкова В. В.* Эластичность алгоритмов // Прикладная дискретная математика. 2010. № 2(8). С. 87–95.

15. *Poincare H.* Sur les equations lineaires aux differentielles ordinaires et aux differences finies // Amer. J. Math. 1885. V. 7. P. 203–258.
16. *Perron O.* Uber einen Satz des Herrn Poincare // J. Reine Angewand. Math. 1909. B. 136. S. 17–34.
17. *Бахвалов Н. С.* Численные методы. М.: Бином. Лаб. знаний, 2006.
18. *Kullmann O.* New methods for 3-SAT decision and worst-case analysis // Theor. Computer Sci. 1999. No. 223. P. 1–72.
19. *Kullmann O. and Luckhardt H.* Deciding propositional tautologies: Algorithms and their complexity // Preprint. 1997. `http://www.cs.swan.ac.uk/~csoliver`.
20. *Куликов А. С., Федин С. С.* Автоматические доказательства верхних оценок на время работы алгоритмов расщепления // Теория сложности вычислений. IX. Зап. научн. сем. ПОМИ. СПб., 2004. Т. 316. С. 111–128.