В решателе ngsat реализована версия DPLL с ограниченной формой нехронологического бэктрекинга. Хорошо известно, что нехронологический бэктрекинг существенно эффективнее обычного (хронологического). Однако реализации полноценного нехронологического DPLL мешает небольшой (в пересчёте на одну задачу) объём памяти GPU. Исходя из этого, в ngsat реализован ограниченный вариант нехронологического DPLL, общая идея которого описана в различных источниках (см., например, [4]). Эта идея состоит в том, чтобы для каждой переменной, значение которой выведено по BCP, хранить информацию о породивших этот вывод причинах. Для этой цели можно использовать стандартные приёмы, применяемые во всех современных SAT-решателях [5]. Хранения конфликтных дизъюнктов можно избежать, если в процессе вывода не использовать рестарты. Данный факт ограничивает «мощность вывода» [6], однако для целого ряда задач такой подход вполне оправдан.

Решатель ngsat протестирован на SAT-задачах, кодирующих поиск систем ортогональных латинских квадратов. Эти задачи являются очень трудными [7]. На данном классе тестов ngsat оказался более эффективным, чем широко известный SAT-решатель minisat [8].

ЛИТЕРАТУРА

1. *Беспалов Д. В., Булавинцев В. Г., Семенов А. А.* Использование графических ускорителей в решении задач криптоанализа // Прикладная дискретная математика. Приложение. 2010. № 3. С. 86–87.
2. *Biere A., Heule M., van Maaren H., and Walsh T.* (*eds.*). Handbook of satisfiability. IOS Press, 2009.
3. *Een N. and Sorensson N.* An extensible SAT-solver // LNCS. 2003. V. 2919. P. 502–518.
4. *Marques-Silva J. P. and Sakallah K. A.* GRASP: A search algorithm for propositional satisfiability // IEEE Trans. Comp. 1999. V. 48. No. 5. P. 506–521.
5. *Moskewicz M. W. et al.* Chaff: Engineering an efficient SAT solver // Proc. 38th Design Automation Conference. Las Vegas, NV, USA: ACM, 2001. P. 530–535.
6. *Beame P., Kautz H. A., and Sabharwal A.* Towards understanding and harnessing the potential of clause learning // J. Artif. Intell. Res. (JAIR). 2004. V. 22. P. 319–351.
7. *Zhang H.* Combinatorial designs by SAT solvers // Handbook of satisfiability. IOS Press, 2009. P. 533–568.
8. `http://minisat.se/`

УДК 519.712

ОБ АСИМПТОТИКЕ РЕШЕНИЙ РЕКУРРЕНТНЫХ СООТНОШЕНИЙ В АНАЛИЗЕ АЛГОРИТМОВ РАСЩЕПЛЕНИЯ ДЛЯ ПРОПОЗИЦИОНАЛЬНОЙ ВЫПОЛНИМОСТИ

В. В. Быкова

Исследована традиционная техника анализа алгоритмов расщепления для решения задачи пропозициональной выполнимости. Предложена теорема, устанавливающая асимптотические верхние оценки времени работы алгоритмов в случае сбалансированного расщепления.

Ключевые слова: *алгоритмы расщепления, сложность вычислений.*

Задача пропозициональной выполнимости (Satisfiability, SAT) является одной из известных NP-полных задач дискретной математики. Пусть X — множество булевых

переменных, т. е. переменных, принимающих значения `true` и `false`. Булева формула в конъюнктивной нормальной форме (КНФ) представляет собой конъюнкцию конечного числа клозов, где клоз — дизъюнкция конечного числа литералов, не содержащая ни одной переменной одновременно с её отрицанием, а всякий литерал — некоторая переменная $x \in X$ или её отрицание $\neg x$. Длина клоза определяется как количество его литералов, а длина формулы — как сумма длин её клозов. Задача SAT традиционно формулируется следующим образом: для булевой формулы F в КНФ выдать ответ «Выполнима», если существует выполняющий набор — набор значений переменных, при котором каждый клоз формулы F принимает значение `true`, и в противном случае выдать ответ «Невыполнима».

Переборный алгоритм решения задачи выполнимости с $N = |X|$ переменными сводится к анализу 2^N различных наборов этих переменных и определяет тривиальную верхнюю оценку сложности для SAT в худшем случае. Алгоритмы, направленные на улучшение тривиальной верхней оценки, активно разрабатываются и исследуются с начала 60-х годов прошлого века и до настоящего времени. Многие из них основаны на принципе «разделяй и властвуй» и названы расщепляющими алгоритмами [1]. Алгоритмы, приведённые в работах М. Дэвиса и Х. Патмена [2] и М. Дэвиса, Г. Лоджмана и Д. Лавленда [3], считаются первыми расщепляющими алгоритмами для задачи SAT и называются DPLL-алгоритмами (по первым буквам фамилий всех четырех авторов). Большинство расщепляющих алгоритмов, разработанных за последние пятьдесят лет для SAT, основаны на идеях, заложенных в DPLL-алгоритмах. Такой алгоритм сначала упрощает входную формулу, после чего расщепляет полученную формулу и производит рекурсивные вызовы самого себя для формул меньшей сложности. Мерами сложности формул обычно выступают число переменных $N = |X|$, число клозов K или длина формулы $L = |F|$. Процесс расщепления в общем виде можно подставить следующим образом.

Вход: формула F в КНФ.

1) Редуцировать F, т. е. преобразовать формулу F в формулу F_0 меньшей сложности, применяя некоторый конечный набор правил редукции.
2) Если задача SAT тривиально решается для F_0, то выдать соответствующий ответ.
3) Выбрать переменную x, входящую в F_0, используя определенную эвристику.
4) Выполнить расщепление формулы F_0 на конечное число формул меньшей сложности по некоторому правилу, использующему различные значения переменной x. Осуществить рекурсивные вызовы данного алгоритма для этих формул. Если хотя бы один из рекурсивных вызовов возвратил выполняющий набор, скорректировать его надлежащим образом и вернуть результат. В противном случае выдать ответ «Невыполнима».

Естественные требования к алгоритму расщепления: редуцирование F, эвристический выбор переменной x и построение решения для F из решения для F_0 должны выполняться за полиномиальное время относительно выбранной меры сложности формул. Расщепляющие алгоритмы, как правило, различаются набором правил редукции, эвристикой выбора переменной x, мерой сложности формул и правилом расщепления.

В настоящее время известно большое количество правил редукции и эвристик. Наиболее популярные правила редукции можно найти в [1 – 5]. Типичными эвристиками по выбору переменной для расщепления являются «взять любую переменную из самого короткого клоза» или «предпочтение отдать переменной, входящей в наибольшее чис-

ло клозов». Простейшее правило расщепления сводится к замене на шаге 4 формулы F_0 на две формулы $F_1 = F_0[x]$ и $F_2 = F_0[\neg x]$. При этом формула F_1 получается из формулы F_0 путём присваивания значения `true` переменной x, что влечёт удаление всех клозов, содержащих x без отрицания, и удаление литерала $\neg x$ в оставшихся клозах. Аналогичным образом определяется формула F_2, т. е. путём присваивания в F_0 переменной x значения `false`. Мера сложности формул F_1 и F_2 понижается по отношению к F за счет применения на шаге 1 правил редукции и присваивания значений переменной x на шаге 4. Таким образом, при использовании простейшего правила расщепления выполняются два рекурсивных вызова для формул F_1 и F_2 соответственно. Очевидно, что число рекурсивных вызовов увеличивается для расщеплений более сложного вида, к примеру, таких: $F_1 = F_0[x, y]$, $F_2 = F_0[x, \neg y]$, $F_3 = F_0[\neg x]$ или $F_1 = F_0[x, y]$, $F_2 = F_0[x, \neg y]$, $F_3 = F_0[\neg x, y]$, $F_4 = F_0[\neg x, \neg y]$.

Область применения алгоритмов расщепления не ограничивается задачей SAT. Данный класс алгоритмов плодотворно применяется при решении многих NP-полных задач (например, MAX-SAT, MAX-2-SAT, максимальное сечение и др.). Для задачи SAT имеется множество работ, описывающих алгоритмы расщепления, где каждая следующая работа улучшает результат предыдущей за счет введения новых правил редукции и расщепления, новой меры сложности формул или более тщательного анализа алгоритма с получением верхней оценки времени его работы. На алгоритмах расщепления базируются многие современные SAT-солверы.

Пусть n — некоторая мера сложности входной формулы F (например, число переменных или число клозов). Предположим, что на шаге 4 алгоритм расщепления разбивает формулу F_0 на m формул F_1, F_2, …, F_m сложности n_1, n_2, …, n_m соответственно и $1 \leqslant n_1 \leqslant n_2 \leqslant \ldots \leqslant n_m < n$. Пусть временные затраты на редуцирование F, эвристический выбор переменной x и построение решения для F из решений F_1, F_2, …, F_m составляют $f(n)$, где $f(n)$ является неубывающей функцией субполиномиального или полиномиального порядка роста [6]. Если $T(n)$ обозначает время работы алгоритма расщепления для входной формулы F, то $T(n)$ является решением следующего рекуррентного соотношения:

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

или

$$T(n) = T(n - t_1) + T(n - t_2) + \ldots + T(n - t_m) + f(n), \tag{1}$$

где $t_1 = n - n_1$, $t_2 = n - n_2$, …, $t_m = n - n_m$. Вектор $(t_1, t_2, \ldots, t_m)$ принято называть вектором расщепления. Поскольку в векторе расщепления возможны равные элементы, то рекуррентное соотношение (1) можно преобразовать к неоднородному линейному рекуррентному соотношению с постоянными коэффициентами

$$T(n) = a_1 T(n-1) + a_2 T(n-2) + \ldots + a_k T(n-k) + f(n) \tag{2}$$

и k начальными условиями $T(0) = \Theta(1)$, $T(1) = \Theta(1)$, …, $T(k-1) = \Theta(1)$. В соотношении (2) величины $a_1, a_2, \ldots, a_k$ — целые константы, a_1, a_2, …, $a_{k-1} \geqslant 0$, $1 \leqslant a_k \leqslant m$, $1 \leqslant k \leqslant n$. Начальные условия свидетельствуют о том, что на формулах сложности k и менее время работы алгоритма расщепления ограничено сверху и снизу константой. Общих методов, дающих решение соотношения (2) в замкнутом виде, не известно. Когда $f(n) \equiv 0$, соотношение (2) становится однородным. Такие соотношения решаются с помощью нахождения корней соответствующего характеристического многочлена [7]. Известны некоторые приёмы учета неоднородности в (2).

Часто для нахождения асимптотической оценки решения рекуррентного соотношения (2) применяют дерево рекурсии [8]. Вычисление сводится к взвешиванию определённым образом вершин этого дерева и подсчёту числа вершин. Именно на таком подходе основана техника анализа алгоритмов расщепления, предложенная О. Кульманом и Х. Люккардтом [4, 5]. Многие оценки для задачи SAT и родственных с ней NP-полных задач получены с помощью этой техники; она подробно описана в [1]. Известны программные реализации этой техники [9].

В работе детально исследована техника Кульмана и Люккардта. Отмечены её достоинства и недостатки. Предложен и доказан новый вариант теоремы из работы [10]. Данная теорема позволяет находить непосредственно асимптотические верхние оценки для частного случая рекуррентного соотношения (2), когда выполняется сбалансированное расщепление (при $a_1 = a_2 = \ldots = a_{k-1} = 0$, $a_k > 1$). Показана область применения этой теоремы для анализа алгоритмов расщепления. Теорема формулируется следующим образом.

Теорема 1. Пусть дано рекуррентное соотношение

$$T(n) = \begin{cases} \Theta(1), \text{ если } 0 \leqslant n \leqslant k-1, \\ aT(n-k) + f(n), \text{ если } n \geqslant k, \end{cases} \tag{3}$$

где $a > 1$, $k \geqslant 1$ — целые константы. Пусть $\tau \geqslant 0$ — вещественная константа. Тогда при $n \to \infty$

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

В рекуррентном соотношении (3) константа a трактуется как число формул, получаемых на шаге расщепления, при этом все формулы имеют одну и ту же сложность $(n-k)$. Вторая оценка данной теоремы отвечает случаю «бесплатного» рекурсивного перехода в алгоритмах расщепления.

ЛИТЕРАТУРА

1. *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.
2. *Davis M. and Putman H.* A computing procedure for quantification theory // J. ACM. 1960. No. 7(3). P. 201–215.
3. *Davis M., Logemann G., and Loveland D.* A machine program for theorem-proving // Comm. ACM. 1962. No. 5(7). P. 394–397.
4. *Kullmann O.* New methods for 3-SAT decision and worst-case analysis // Theor. Comp. Sci. 1999. No. 223. P. 1–72.
5. *Kullmann O. and Luckhardt H.* Deciding propositional tautologies: Algorithms and their complexity / Preprint. 1997. `http://cs-svr1.swan.ac.uk/~csoliver`
6. *Быкова В. В.* Эластичность алгоритмов // Прикладная дискретная математика. 2010. № 2(8). С. 87–95.
7. *Грэхем Р., Кнут Д., Паташник О.* Конкретная математика. Основание информатики. М.: Бином. Лаборатория знаний, 2006.
8. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 1999.

9. *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.
10. *Быкова В. В.* Математические методы анализа рекурсивных алгоритмов // Журнал Сибирского федерального университета. Сер. Математика и физика. 2008. №1(3). С. 236–246.

УДК 519.6

К РЕШЕНИЮ БОЛЬШИХ СИСТЕМ СРАВНЕНИЙ

К. Д. Жуков, А. С. Рыбаков

Пусть дано конечное множество S натуральных чисел, такое, что почти все его элементы попарно взаимно просты. Рассматривается алгоритм нахождения всех элементов $s \in S$, таких, что $(s, s') > 1$ для некоторого $s' \in S$, $s' \neq s$, который позволяет сводить произвольную систему полиномиальных сравнений к нескольким системам с взаимно простыми модулями.

Ключевые слова: *взаимно простая база, наибольший общий делитель, дерево наибольших общих делителей, НОД слиянием.*

Определение 1. Пусть S — конечное подмножество натуральных чисел. Взаимно простой базой для S называется конечное подмножество натуральных чисел B, такое, что любое число из S представляется в виде произведения элементов из B и любые два элемента множества B взаимно просты.

Взаимно простые базы используются в ряде приложений, например при решении систем сравнений в целых числах [1].

Пусть задана система полиномиальных сравнений вида $f_i(x) = 0 \pmod{a_i}$, $i = 0, 1, \ldots, m-1$, где x — набор переменных. Построим для множества чисел $\{a_0, \ldots, a_{m-1}\}$ взаимно простую базу $\{b_0, \ldots, b_{r-1}\}$. Для каждого элемента b_j нужно последовательно составить системы сравнений по модулям b_j^l, где $l = 1, 2, \ldots, t_j$, а t_j — максимальное, для которого найдётся a_i, делящееся на $b_j^{t_j}$. В эти системы войдут те уравнения $f_i(x) = 0$, для которых число a_i делится на b_j^l. Поднимая по лемме Гензеля решения от системы к системе, получим множество решений по модулю $b_j^{t_j}$. Найдя такие решения для каждого b_j, применим китайскую теорему об остатках, чтобы получить решения вида $x \bmod \prod_j b_j^{t_j}$. Каждое такое решение будет решением исходной системы.

Построение взаимно простой базы применяется и в других приложениях, например в задачах нахождения алгебраических зависимостей среди радикалов [2], вычисления нормальных базисов в расширениях полей [3] и др. [4].

В работе [4] Д. Бернштейн предложил алгоритм построения взаимно простой базы с некоторыми дополнительными свойствами, называемой естественной взаимно простой базой.

Ниже предлагается подход к построению взаимно простой базы для специального случая. Подход эффективен, когда заранее известно, что нетривиальная часть взаимно простой базы (элементы, отличные от 1 и не содержащиеся в S) достаточно мала. Метод состоит в том, чтобы быстро отбросить элементы $s \in S$, такие, что $(s, s') = 1$ для любого $s' \in S$, $s' \neq s$. Для остальных чисел можно применить известные алгоритмы построения взаимно простой базы.