УДК 519.7 DOI 10.17223/2226308X/15/23

О СКРЫТЫХ УПРОЩАЮЩИХ СТРУКТУРАХ В КОМБИНАТОРНЫХ ЗАДАЧАХ И ИХ ВЕРОЯТНОСТНЫХ ОБОБЩЕНИЯХ[1]

А. А. Семёнов

Дан обзор некоторых недавних результатов, связанных со структурами, за которыми в англоязычной литературе закрепился термин “Backdoor”. Наиболее близким аналогом в русском, по-видимому, является термин «лазейка». Лазейка — это такое множество переменных в произвольной задаче удовлетворения ограничений, знание которого существенно упрощает рассматриваемую задачу либо даёт верхнюю оценку трудности её решения, которая лучше трудности тривиальной переборной стратегии. Лазейки в последние годы являются популярным объектом исследований как в прикладных областях, так и в теоретических (главным образом, в параметризованной сложности). Обсуждается применение лазеек для повышения эффективности решения конкретных комбинаторных задач из семейств SAT (проблема булевой выполнимости) и 0-1-ILP (0-1-целочисленное линейное программирование).

Ключевые слова: *лазейки в комбинаторных задачах, проблема булевой выполнимости (SAT), 0-1-целочисленное линейное программирование.*

Понятие «лазейка» в применении к произвольной задаче удовлетворения ограничений (Constraint Satisfaction Problem, CSP) впервые дано в работе [1]. Нас в дальнейшем будут, в первую очередь, интересовать «сильные лазейки» (Strong Backdoor Sets), также введённые в [1].

Рассмотрим проблему булевой выполнимости (SAT) как один из простейших и наиболее наглядных примеров CSP. В произвольном экземпляре SAT фигурирует булева формула C (обычно в конъюнктивной нормальной форме, КНФ). Множество встречающихся в C переменных со значениями в $\{0,1\}$ обозначим через X. Элементарными ограничениями являются дизъюнкты, входящие в C. Для произвольного $B \subseteq X$ обозначим через $\{0,1\}^{|B|}$ множество всех наборов значений переменных из B. Подстановка значения $\alpha \in \{0,1\}$ произвольной переменной $x \in X$ в формулу C определяется стандартным образом [2]. Обозначим через $C\,[\beta/B]$ КНФ, которая получается в результате подстановки в C набора β значений переменных из B. Пусть O — некоторый полиномиальный алгоритм.

Определение 1 [1]. Сильной лазейкой (Strong Backdoor Set, SBS) для КНФ C относительно алгоритма O называется такое множество $B \subseteq X$, что для любого $\beta \in \{0,1\}^{|B|}$ задача выполнимости КНФ $C\,[\beta/B]$ разрешима алгоритмом O.

Таким образом, если B — сильная лазейка для КНФ C, то задача выполнимости C решается за время $\mathrm{poly}(|C|)\,2^{|B|}$, где $\mathrm{poly}(\cdot)$ — некоторый полином; $|C|$ — длина двоичного описания C. Можно привести массу примеров, когда C содержит сильную лазейку, число переменных в которой может не превосходить нескольких процентов (или даже долей процента) от общего числа переменных в X. Таковы, например, КНФ, кодирующие задачи криптоанализа. В этих случаях множество переменных, кодирующих вход рассматриваемой криптографической функции, образует SBS относитель-

[1]Исследование выполнено в рамках госзадания Минобрнауки России по проекту «Теоретические основы, методы и высокопроизводительные алгоритмы непрерывной и дискретной оптимизации для поддержки междисциплинарных научных исследований», № гос. регистрации 121041300065-9.

но простейшего правила распространения булевых ограничений — правила единичного дизъюнкта (Unit Propagation rule, UP). Например, КНФ, кодирующая задачу поиска прообраза криптографической функции SHA-256, то есть задачу обращения функции $f_{\text{SHA-256}} : \{0,1\}^{512} \to \{0,1\}^{256}$, имеет SBS относительно правила UP, которое состоит из 512 переменных, при том что общее число переменных в данной КНФ равно 49 098. Легко понять, что сильная лазейка размерности существенно меньше 512, если бы она существовала, давала бы нетривиальную атаку на данную функцию. Очевидный интерес представляют сильные лазейки наименьшей мощности, поскольку они дают лучшие верхние границы в рассматриваемом классе таких границ.

Определение 2. Для произвольной КНФ C и полиномиального алгоритма O сильную лазейку относительно O наименьшей мощности будем называть минимальной.

Теорема 1 [1]. Если КНФ C содержит сильную лазейку размера $|B| \leqslant |X|/2$, то существует алгоритм, который решает SAT для C за время

$$\text{poly}(|C|) \left(\frac{2\,|X|}{\sqrt{|B|}} \right)^{|B|}. \tag{1}$$

Следствие 1 [1]. Для формул с сильной лазейкой размера не более $n/4{,}404$ проблема SAT может быть решена детерминированным образом за время $\mathcal{O}\left(c^n\right)$, где $c < 2$.

Алгоритм из [1] (далее — «алгоритм A»), который подразумевается в формулировке теоремы 1, очень прост. Сначала проверяются на свойство быть сильной лазейкой все множества, состоящие из одной переменной — их $n = |X|$, затем — все множества из двух переменных — их $\binom{n}{2}$, и так далее. Очевидно, что первая сильная лазейка, найденная алгоритмом A, минимальная.

С другой стороны, в худшем случае алгоритм A пройдёт все подмножества множества X и переберёт все 2^n наборов значений переменных из X. Таким образом, в худшем случае сложность данного алгоритма есть следующая величина с точностью до некоторого полиномиального от $|C|$ сомножителя:

$$\sum_{i=1}^{n} \binom{n}{i} 2^i = \sum_{i=0}^{n} \binom{n}{i} 2^i - 1 = 3^n - 1. \tag{2}$$

Следствие 1 означает, что если существует сильная лазейка мощности $\leqslant |X|/2$, то даже алгоритм A может быть асимптотически (на соответствующем семействе формул) лучше полного перебора. Оценку (1) можно переписать в более наглядной форме. Это делается за счёт анализа биномиальных коэффициентов и использования формулы Стирлинга. Заметим, что в случае существования лазейки мощности K имеет место $T_A \leqslant p(|C|) \sum\limits_{i=1}^{K} \binom{n}{i} 2^i$, где $p(\cdot)$ — некоторый полином, что для $K \leqslant n/q$ и $q \geqslant 2$ с учётом свойств биномиальных коэффициентов даёт следующую оценку:

$$T_A \leqslant p(|C|)\, 2^K \binom{n}{K}. \tag{3}$$

Преобразуя (3) с использованием формулы Стирлинга, имеем

Следствие 2. Если КНФ C содержит сильную лазейку мощности $K = \lfloor n/q \rfloor$, $q \geqslant 2$, то существует алгоритм, решающий SAT для C за время T_A, где

$$T_A \leqslant p(|C|)\, 2^{n(1/q + \log q - (q-1)\log(q-1)/q)}. \tag{4}$$

К сожалению, использование (4) не даёт существенного улучшения оценки из [1]. Конкретно, (4) даёт следующий результат (аналог следствия 4.2 из [1]), который получен за счёт решения пакетом WolframAlpha уравнения $\frac{1}{q} + \log q - \frac{q-1}{q}\log(q-1) = 1$:

Следствие 3. Для формул с сильной лазейкой размера не более $n/4{,}40349$ проблема SAT может быть решена детерминированным образом за время $\mathcal{O}(c^n)$, где $c < 2$.

Соотношение (2) означает, что алгоритм A крайне неэффективен в применении к поиску минимальной сильной лазейки и вряд ли может использоваться для решения практических задач. При детальном рассмотрении становится понятно, что основной фактор неэффективности A — это необходимость проверять свойство произвольного $B \subseteq X$ быть сильной лазейкой: для этого требуется проверить разрешимость $C[\beta/B]$ алгоритмом O для каждого $\beta \in \{0,1\}^{|B|}$ (делая, таким образом, каждый раз $2^{|B|}$ проверок).

В работе [3] предложено вероятностное обобщение понятия сильной лазейки. Соответствующее определение выглядит следующим образом.

Определение 3. Пусть C — произвольная КНФ, X — множество переменных в C и O — некоторый полиномиальный алгоритм. Произвольное множество $B \subseteq X$ назовём ρ-лазейкой, $\rho \in [0,1]$, относительно алгоритма O, если среди всех задач вида $C[\beta/B]$, $\beta \in \{0,1\}^{|B|}$, доля задач, решаемых алгоритмом O, составляет не менее ρ.

Очевидно, что ρ-лазейка с $\rho = 1$ — это сильная лазейка. Если $\rho < 1$, то с практической точки зрения важно, чтобы ρ было как можно ближе к 1. Действительно, если B — такая ρ-лазейка, то мы можем алгоритмом O решить $\rho \cdot 2^{|B|}$ задач вида $C[\beta/B]$, а к оставшимся $(1-\rho)\,2^{|B|}$ задачам применить полный алгоритм решения SAT. Если считать ρ, проверяя, решается ли алгоритмом O каждая задача вида $C[\beta/B]$, то соответствующая процедура, как и в случае сильных лазеек, неэффективна. Однако, как показано в [3], можно эффективно оценить ρ, используя простой вероятностный тест.

Для произвольных C над X, $B \subseteq X$ и полиномиального алгоритма O зададим на $\{0,1\}^{|B|}$ равномерное распределение и определим случайную величину $\xi_B : \{0,1\}^{|B|} \to \{0,1\}$, которая принимает на $\beta \in \{0,1\}^{|B|}$ значение 1, если $C[\beta/B]$ решается алгоритмом O и значение 0 в противном случае. Очевидно, что если B — ρ-лазейка, то вероятность успеха в наблюдении величины ξ_B составляет не менее ρ и, таким образом, $E[\xi_B] \geqslant \rho$. Последнее означает, что можно оценивать параметр ρ произвольного $B \subseteq X$, оценивая величину $E[\xi_B]$. Поскольку ξ_B — случайная величина Бернулли, то для оценки $E[\xi_B]$ можно использовать следующее неравенство (являющееся вариантом известной границы Чернова):

$$\mathsf{P}\left[\left|\frac{1}{N}\sum_{j=1}^{N}\xi_j - E[\xi_B]\right| < \varepsilon\right] \geqslant 1 - 2e^{-N\varepsilon^2/4}. \tag{5}$$

Таким образом, мы можем оценить $E[\xi_B]$ с точностью ε и уровнем значимости δ за счёт использования N независимых наблюдений величины ξ_B, обозначенных в (5) через $\xi_1, \ldots, \xi_N$. Соответствующей оценкой является среднее арифметическое наблюдаемых значений. Алгоритмы оценивания такого типа относятся к методу Монте-Карло [4].

Для произвольного $B \subseteq X$ определим следующий тест Монте-Карло: зафиксируем $\varepsilon, \delta \in (0,1)$ и $N = \lceil 16\ln(2/\delta)/\varepsilon^2 \rceil$, пронаблюдаем ξ_B независимо N раз, пусть $\xi_1, \ldots, \xi_N$ — результаты этих наблюдений, вычислим $\widetilde{\rho} = \frac{1}{N}\sum_{j=1}^{N}\xi_j$. Скажем, что B про-

ходит тест, если $\widetilde{\rho} \in [1-\varepsilon/2, 1]$. Используя (5), нетрудно показать справедливость следующего факта.

Следствие 4. Если B проходит описанный тест Монте-Карло, то заключение, что $E[\xi_B] \in [1-\varepsilon, 1]$, верно с вероятностью $\geqslant 1-\delta$.

Теперь мы можем определить вероятностный аналог сильной лазейки.

Определение 4 [3]. Назовём B сильной (ε, δ)-лазейкой, если B проходит описанный тест Монте-Карло.

Например: если B проходит тест при $\varepsilon = \delta = 0{,}01$, то с вероятностью 99 % можно заключить, что B — это ρ-лазейка с $\rho \geqslant 0{,}98$ (то есть ρ очень близко к 1). Нетрудно заметить, что сложность описанного алгоритма оценивания $E[\xi_B]$ при фиксированных $\varepsilon, \delta \in (0,1)$ ограничена сверху величиной $p(|C|)16\ln(2/\delta)/\varepsilon^2$ и, следовательно, данный алгоритм эффективный.

Таким образом, в противовес алгоритму A, перебирающему множества B последовательно возрастающей мощности, мы можем искать сильные (ε, δ)-лазейки, минимизируя некоторую целевую функцию при помощи метаэвристических алгоритмов на множестве всех подмножеств X. Более конкретно, с произвольным $B \subseteq X$ в [3] связывается значение следующей функции:

$$\Phi_{C,A,N}(B) = \widetilde{\rho}_B \cdot 2^{|B|} + G_{C,A,N}(B). \tag{6}$$

В (6) часть $G_{C,A,N}(B)$ — это штрафная функция: её значение должно резко возрастать при уменьшении величины $\widetilde{\rho}_B$. В [3] использована штрафная функция вида

$$G_{C,A,N} = \begin{cases} (1-\widetilde{\rho}_B)2^{\omega|X|}, & \text{если } \widetilde{\rho}_B > 0; \\ \infty, & \text{если } \widetilde{\rho}_B = 0, \end{cases}$$

где $\omega \in (0,1)$ — параметр, подбираемый эмпирически для каждой конкретной КНФ C.

Как сказано ранее, для решения задач вида $C[\beta/B]$, не разрешимых полиномиальным алгоритмом O, можно использовать полный SAT-решатель. Для решения SAT в отношении трудной C можно использовать также несколько найденных сильных (ε, δ)-лазеек. Пусть $B_1, \ldots, B_s$ — такие лазейки. Для каждого B_r, $r \in \{1, \ldots, s\}$, построим множество Δ_r, состоящее из всех таких $\beta \in \{0,1\}^{|B_r|}$, что $C[\beta/B_r]$ не разрешима алгоритмом O. Заметим, что мощность каждого Δ_r, $r \in \{1, \ldots, s\}$, может составлять доли процента от $2^{|B_r|}$. Рассмотрим декартово произведение $\Delta = \Delta_1 \times \ldots \times \Delta_r$. Заметим, что каждый вектор из Δ состоит из $\sum\limits_{r=1}^{s} |B_r|$ бит и, следовательно, его подстановка в C может существенно упростить задачу. При этом мощность Δ равна

$$\left(\prod_{r=1}^{s}(1-\rho_r^*)\right) 2^{\sum\limits_{r=1}^{s}|B_r|}, \tag{7}$$

где ρ_r^* — доля таких $\beta \in \{0,1\}^{|B_r|}$, что $C[\beta/B]$ решается алгоритмом O. При близких к 1 величинах ρ_r^*, $r \in \{1, \ldots, s\}$, величина (7) может быть разумной, и соответствующие SAT-задачи оказываются простыми для современных SAT-решателей.

В [3] проведены вычислительные эксперименты по поиску сильных (ε, δ)-лазеек, которые подтвердили, что во многих комбинаторных задачах часто встречаются такие лазейки, имеющие малую мощность. Иногда их использование позволяет ускорить

решение исходной SAT-проблемы в десятки и сотни раз. Поиск сильных (ε, δ)-лазеек может рассматриваться как задача минимизации функции вида (6), для чего можно применять любой алгоритм псевдобулевой оптимизации. В [3] для минимизации функций вида (6) использован генетический алгоритм. Эксперименты проводились на вычислительном кластере с помощью программной системы EvoGuess [5]. В роли алгоритма O выступало простейшее правило единичного дизъюнкта [6].

Следует отметить, что концепция сильных (ε, δ)-лазеек может быть естественным образом перенесена на любую задачу удовлетворения ограничений. Например, пусть I — произвольная задача целочисленного линейного программирования (ILP, Integer Linear Programming) и X — множество фигурирующих в постановке I переменных со значениями в $\{0, 1\}$ (таким образом, речь идет о 0-1-ILP). Напомним, что в стандартной подстановке для решения I требуется минимизировать (или максимизировать) некоторую функцию при ограничениях, заданных системой целочисленных неравенств следующего вида:

$$Ax \geqslant b, \tag{8}$$

где A — целочисленная матрица размера $m \times n$; b — целочисленный вектор длины m; $x = (x_1, \ldots, x_n)$ — вектор переменных, принимающих значения в $\{0, 1\}$.

По аналогии мы можем ввести множество булевых переменных $X = \{x_1, \ldots, x_n\}$, рассмотреть произвольное B, $B \subseteq X$, а в качестве полиномиального алгоритма взять алгоритм преобразования системы неравенств (8) после подстановки в неё произвольного $\beta \in \{0, 1\}^{|B|}$. Далее можем искать ρ-лазейки и, в частности, сильные (ε, δ)-лазейки, используя шаги, аналогичные описанным выше. Первые вычислительные эксперименты в этом направлении (с использованием ILP-решателей Gurobi и SCIP) показали работоспособность концепции вероятностных лазеек: для некоторых трудных 0-1-ILP-задач найденные лазейки дают ускорение в десятки раз.

ЛИТЕРАТУРА

1. *Williams R., Gomes C. P., and Selman B.* Backdoors to typical case complexity // Proc. IJCAI'03. August 2003. P. 1173–1178.
2. *Чень Ч., Ли Р.* Математическая логика и автоматическое доказательство теорем. М.: Наука, 1983.
3. *Semenov A., Pavlenko A., Chivilikhin D., and Kochemazov S.* On probabilistic generalization of backdoors in Boolean satisfiability // Proc. AAAI-2022. `https://www.aaai.org/AAAI22Papers/AAAI-8477.SemenovA.pdf`.
4. *Metropolis N. and Ulam S.* The Monte Carlo method // J. Amer. Statistical Assoc. 1949. No. 44(247). P. 335–341.
5. `https://github.com/ctlab/EvoGuess/releases/tag/v2.0.0`.
6. *Dowling W. F. and Gallier J. H.* Linear-time algorithms for testing the satisfiability of propositional horn formulae // J. Logic Programming. 1984. V. 1. Iss. 3. P. 267–284.