ПРЕПРИНТЫ ПОМИ РАН

Учредитель: Санкт-Петербургское отделение Математического института им. В. А. Стеклова Российской академии наук

Свидетельство о регистрации средства массовой информации: ЭЛ №ФС 77-33560 от 16 октября 2008 г. Выдано Федеральной службой по надзору в сфере связи и массовых коммуникаций

Контактные данные: 191023, г. Санкт-Петербург, наб. реки Фонтанки, дом 27

телефоны:(812)312-40-58; (812) 571-57-54

e-mail: admin@pdmi.ras.ru

http://www. pdmi.ras.ru /preprint/

Заведующая информационно-издательским сектором Симонова В.Н

ПРЕПРИНТ ПОМИ — 3/2009

Нижняя оценка на среднее время обращения функции Голдрейха «пьяными» алгоритмами расщепления [1]

Д.М. Ицыксон

Санкт-Петербургское отделение
Математического института им. В. А. Стеклова
Российской академии наук

http://logic.pdmi.ras.ru/~dmitrits

Июль, 2009

Аннотация

Мы доказываем экспоненциальную нижнюю оценку на среднее время обращения функции Голдрейха [Gol00] *«пьяными»* [AHI05] алгоритмами, тем самым разрешаем открытый вопрос, поставленный в работе [CEMT09]. Функция Голдрейха, которую мы используем задана графом-расширителем и предикатом $P_d(x_1, x_2, \ldots, x_d) = x_1 \oplus x_2 \oplus \cdots \oplus x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$, где Q — произвольный k-местный предикат, $k + 1 < \frac{d}{4}$. «Пьяные» алгоритмы — это семейство алгоритмов, основанных на расщеплении, которые используют ничем не ограниченную эвристику выбора переменной для расщепления, а значение, которое подставляется первым, выбирается случайным образом. Техника доказательства существенно упрощает технику, используемую в [AHI05] и [CEMT09].

[1] Исследования частично поддержаны грантом РФФИ 08-01-00640, президентским грантом поддержки ведущих научных школ НШ-4392.2008.1, фондом содействия отечественной науки и государственным контрактом с Федеральным агенством по образованию.

ПРЕПРИНТЫ
Санкт-Петербургского отделения
Математического института им. В. А. Стеклова
Российской академии наук

PREPRINTS
of the St. Petersburg Department of Steklov Institute of Mathematics

---

1 Введение

В 2000-м году О. Голдрейх предложил кандидата в односторонние функции, основанного на графах-расширителях [Gol00]. Функция имеет n бинарных входов и n бинарных выходов. Каждый выход функции зависит только от каких-то d входов и вычисляется по ним с помощью заданного d-местного предиката. Голдрейх предлагал использовать графы со свойством расширения в качестве графа зависимостей и случайный d-местный предикат. Функция, предложенная Голдрейхом, имеет некоторое сходство с псевдо-случайным генератором Нисана-Вигдерсона [NW94].

Одним из практических подходов к обращению односторонних функций является использование современных программ, решающих задачу выполнимости булевой формулы [MZ06]. Практически все реально используемые алгоритмы для задачи выполнимости булевой формулы основаны на методе расщепления (так называемые DPLL (по инициалам авторов Дэвис, Путнам, Логеман, Ловеленд) алгоритмы [DP60, DLL62]). Алгоритм расщепления — это рекурсивный алгоритм. При каждом вызове он упрощает входную формулу F (не влияя на ее выполнимость), выбирает переменную v из нее и делает два рекурсивных вызова на формулах $F[v := 1]$ и $F[v := 0]$ в некотором порядке. Он отвечает "Выполнима", если по крайней мере один рекурсивный вызов отвечает то же самое (заметим, что нет необходимости делать второй вызов, если первый был удачный). Рекурсия продолжается, пока формула не станет тривиальной.

Алгоритм расщепления определяется правилами упрощения и двумя эвристиками: эвристика **A** выбирает переменную, а эвристика **B** выбирает, какое значение переменной будет проверено раньше.

Для невыполнимых формул нижние оценки на время работы алгоритмов расщепления следуют из нижних оценок на размер резолюционных доказательств [Цей68]. Невыполнимые формулы, основанные на псевдослучайных генераторах Нисана-Вигдерсона, используются для доказательства нижних оценок в различных пропозициональных системах доказательств [ABSRW00]. Но формулы, получающиеся из задач обращения односторонних функций обычно являются выполнимыми. Вполне возможно, что расщепляющие алгоритмы быстро работают на выполнимых формулах. Если никак не ограничивать эвристики, то доказательство экспоненциальной нижней оценки на время работы расщепляющих алгоритмов на выполнимых формулах означало бы, что $\mathbf{P} \neq \mathbf{NP}$ (иначе эвристика **B** могла бы за полиномиальное время вычислить правильное значение для переменной).

В работе [AHI05] были получены безусловные экспоненциальные нижние оценки на время работы двух классов расщепляющих алгоритмов на выполнимых формулах. Первый класс алгоритмов — это *близорукие* алгоритмы. Эвристики в таких алгоритмах ограничены тем, что они могут прочитать лишь маленькую часть формулы точно, а остальную формулу они видят лишь схематично (например, как это сформулировано в [AHI05] не видят знаков отрицания, но имеют доступ к числу вхождений переменной с положительным и отрицательным знаками). Для близоруких алгоритмов была получена экспоненциальная нижняя оценка для формул, кодирующих системы линейных уравнений, в которых матрица обладает свойством расширения. На самом деле, такие формулы кодируют функцию Голдрейха в случае линейного предиката. Второй класс алгоритмов расщепления, которые рассматривались в [AHI05] — это *«пьяные»* алгоритмы. В таких алгоритмах эвристика выбора переменной ничем не ограничена и может быть даже невычислимой, а эвристика выбора значения, которое будет подставлено первым ограничено достаточно сильно: значение выбирается равновероятно случайным образом. Для «пьяных» алгоритмов нижняя оценка была построена для искусственных формул, которые

строятся на основе трудных невыполнимых формул.

Функции Голдрейха, основанные на линейном предикате, не очень интересные, поскольку их эффективно можно обратить с помощью метода Гаусса. В работе [CEMT09] была обобщена техника, разработанная в [AHI05] для доказательства нижней оценки для близоруких алгоритмов, на нелинейные предикаты. В частности в [CEMT09] доказана экспоненциальная нижняя оценка на среднее (по входам функции) время обращения функции Голдрейха с предикатом $x_1 \oplus x_2 \oplus \cdots \oplus x_{d-2} \oplus x_{d-1}x_d$ близорукими алгоритмами. Так же в [CEMT09] был произведен практический анализ времени работы современных программ для задачи выполнимости булевой формулы, которые обращали функцию Голдрейха с этим предикатом, было показано, что эти формулы трудны для программ MiniSAT 2.0 [EB05, ES03].

В работе [CEMT09] вопрос о экспоненциальных нижних оценках на время обращения функций Голдрейха «пьяными» алгоритмами оставлен открытым. Настоящая работа отвечает на этот вопрос. В частности, мы доказываем, что для случайного графа с большой вероятностью среднее время работы «пьяного» алгоритма на формуле, соответствующей функции Голдрейха, основанной на предикате $x_1 \oplus \cdots \oplus x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$, экспоненциально, где Q — произвольный k-местный предикат, $k+1 < \frac{d}{4}$, d достаточно большая константа.

План доказательства следующий: сначала мы доказываем нижнюю оценку для невыполнимых формул, пользуясь техникой из [BSW01], затем показываем, что можно ввести некоторую надстройку над «пьяными» алгоритмами, которая лишь уменьшает время работы, но гарантирует, что в течении некоторого количества первых шагов алгоритм не обнаружит невыполнимого дизъюнкта. Далее мы покажем, что с большой вероятностью число выполняющих наборов у функции Голдрейха маленькое, и с большой вероятностью в результате работы надстройки окажется невыполнимая формула, и можно будет применить нижнюю оценку для невыполнимых формул.

Общий план доказательства совпадает с планом, который был использован в [AHI05] и [CEMT09], но получившееся доказательство существенно проще, чем в упомянутых выше статьях.

2 Основные определения и обозначения

Пропозициональной переменной мы называем переменную, которая принимает одно из двух значений 0 или 1. Литералом называется пропозициональная переменная или ее отрицание. Дизъюнктом называется дизъюнкция литералов. Формулой в конъюнктивной нормальной форме (КНФ) называется конъюнкция нескольких дизъюнктов. Формула называется формулой в k-КНФ, если в каждый ее дизъюнкт входит не более k литералов. Формула называется выполнимой, если существует такая подстановка ее переменным, при которой значение формулы становится равным 1 (такую подстановку переменных мы называем выполняющим набором).

Множество всех функций из $\{0,1\}^n$ в $\{0,1\}^n$ будем обозначать $\mathfrak{F}_n$. Для каждой функции $f \in \mathfrak{F}_n$ и каждой строки $b \in \{0,1\}$ равенство $f(x) = b$ можно записать в виде формулы в КНФ с пропозициональными переменными $x_1, \ldots, x_n$. Будем обозначать такую формулу $\Phi_{f(x)=b}$.

Во всей работе $G(V,E)$ — это двудольный граф, в котором могут быть кратные ребра. Вершины графа G разбиты на две доли: $X = \{x_1, x_2, \ldots, x_n\}$ и $Y = \{y_1, y_2, \ldots, y_n\}$, число вершин в каждой доле равняется n. Вершины X мы будем называть входами, а вершины Y — выходами. Каждая вершина из Y имеет степень d.

Предложение 2.1 (оценка Чернова-Хоефдинга). Для независимых и одинаково распределенных случайных величин $X_1, X_2, \ldots, X_N$, таких что $X_i \in [0,1]$ и $E[X_i] = \mu$, выполняется $\Pr\{|\frac{\sum_{i=1}^{N} X_i}{N} - \mu| \geq \epsilon\} \leq 2e^{-2\epsilon^2 n}$.

2.1 Функция Голдрейха

Голдрейх в [Gol00] построил функцию $g : \{0,1\}^n \to \{0,1\}^n$, заданную с помощью двудольного графа $G(V,E)$ и предиката $P : \{0,1\}^d \to \{0,1\}$. Каждая строка из $\{0,1\}^n$ задает некоторое значение на входах $\{x_1, x_2, \ldots, x_n\}$, значение $(g(x))_j$ (j-й символ строки $g(x)$) рассчитывается следующим образом: в вершину y_j входят ребра из вершин $x_{j_1}, x_{j_2}, \ldots, x_{j_d}$, тогда $(g(x))_j = P(x_{j_1}, x_{j_2}, \ldots, x_{j_d})$. Мы считаем, что в каждой вершине множества Y задан некоторый порядок на входящих ребрах.

Голдрейх предлагал использовать случайный предикат и графы со свойством расширения.

Задачу обращения функции g на строке b (т.е., уравнение $g(x) = b$) можно записать в виде формулы $\Phi_{g(x)=b}$ в d-КНФ: каждое равенство $P(x_{j_1}, x_{j_2}, \ldots, x_{j_d}) = b_j$ записываем формулой в d-КНФ, состоящей из не более, чем 2^d дизъюнктов. Для множества вершин $A \subseteq Y$ будем обозначать через $\Phi^A_{g(x)=b}$, подформулу $\Phi_{g(x)=b}$, которая состоит из дизъюнктов, которые соответствуют вершинам из множества A.

2.2 Графы-расширители

Пусть G — это двудольный граф, вершины которого разбиты на две доли $X = \{x_1, x_2, \ldots, x_n\}$ и $Y = \{y_1, y_2, \ldots, y_n\}$. Для $A \subseteq Y$ обозначим через $\Gamma(A)$ (множество соседей A) множество вершин из X, соединенных как минимум с одной вершиной из A, а через $\delta(A)$ (граница A) множество вершин из X, которые соединены ровно с одной вершиной из A одним ребром.

Определение 2.1. Граф G называется (r,d,c)-расширителем, если

- Степени всех вершин из множества Y равняются d
- Для любого множества $A \subseteq Y, |A| \leq r$ выполняется $\Gamma(A) \geq c|A|$

Граф G называется (r,d,c)-граничным расширителем, если второе условие формулируется так:

- Для любого множества $A \subseteq Y, |A| \leq r$ выполняется $\delta(A) \geq c|A|$

Лемма 2.1 ([AHI05], лемма 1). Каждый (r,d,c)-расширитель является граничным $(r,d,2c-d)$-расширителем.

Доказательство. Пусть $A \subseteq Y$, $|A| \leq r$, тогда $|\Gamma(A)| \geq c|A|$. Множество ребер между A и $\Gamma(A)$ можно оценить так: $d|A| \geq |\delta(A)| + 2|\Gamma(A) \setminus \delta(A)| = 2|\Gamma(A)| - |\delta(A)| \geq 2c|A| - \delta(A)$. Иначе $\delta(A) \geq (2c-d)|A|$. □

Лемма 2.2 ([HLW06], лемма 1.9). При $d \geq 32$, для достаточно больших n с вероятностью 0.9 случайный двудольный d-регулярный граф, в котором в долях X и Y по n вершин и для каждой вершины из Y выбираются случайным образом равновероятно независимо d ребер (повторения разрешаются), является $(\frac{n}{10d}, d, \frac{5}{8}d)$-расширителем.

Следствие 2.1. В условиях теоремы этот граф является граничным $(\frac{n}{10d}, d, \frac{1}{4}d)$-расширителем.

Доказательство. Следует из леммы 2.1. □

2.3 Алгоритмы расщепления

Пусть $f \in \mathfrak{F}_n$ — некоторая функция. Задача вычисления $f^{-1}(b)$ эквивалентна нахождению выполняющего набора формулы $\Phi_{f(x)=b}$.

Мы рассмотрим широкий класс алгоритмов поиска выполняющего набора — алгоритмы, основанные на расщеплении. Алгоритм расщепления параметризован двумя *эвристиками* (процедурами):

- Процедура **A**, которая по формуле в КНФ выдает ее переменную. Это переменная, по которой будет проводиться расщепление.
- Процедура **B**, которая по формуле в КНФ и ее переменной выдает значение из $\{0, 1\}$. Это значение, которое будет подставляться при расщеплении первым.

Алгоритм может также использовать некоторые синтаксические *правила упрощения*, такие правила могут заменять формулу на эквивалентную ей и делать подстановки значений ее переменным, если их значение семантически следует из выполнимости формулы.

Алгоритм расщепления — это рекурсивный алгоритм, который получает на вход формулу φ и частичную подстановку ρ

Алгоритм 2.1. Вход: формула φ и подстановка ρ

- Упростить φ с помощью правил упрощения (считаем, что правила упрощения меняют φ и ρ, причем все переменные, подставленные подстановкой ρ удаляются из формулы φ).
- Если формула стала пустой (т.е., все ее дизъюнкты выполнены подстановкой ρ), то выдать ρ. Если формула содержит пустой дизъюнкт (заведомо невыполненный), то выдать "формула невыполнима".
- $x_j := \mathbf{A}(\varphi)$
- $c := \mathbf{B}(\varphi, x_j)$
- Запустить алгоритм рекурсивно на $(\varphi[x_j := c], \rho \cup \{x_j := c\})$, если алгоритм выдал подстановку, то выдать ее, в противном случае запустить на $(\varphi[x_j := 1 - c], \rho \cup \{x_j := 1 - c\})$, если была напечатана подстановка, то выдать ее, иначе выдать "формула невыполнима".

Определение 2.2. «Пьяные» алгоритмы [AHI05] — это такие алгоритмы расщепления, в которых эвристика **A** может быть произвольной (даже невычислимой), а эвристика **B** выбирает значение переменной равновероятно случайным образом. Правила упрощения:

- *Правило удаления единичного дизъюнкта*: если формула содержит дизъюнкт, в котором ровно один литерал, то сделать такую подстановку этому литералу, чтобы выполнить этот дизъюнкт.
- *Правило чистых литералов*: если формула содержит литералы, которые входят только с положительным знаком, то подставить этим литералам значение 1.

В разделе 3 мы покажем, что можно считать, что «пьяный» алгоритм не использует перечисленные выше правила упрощения, их использование всегда можно заменить разумным выбором переменной, по которой расщепляться.

Временем работы алгоритма расщепления будем считать количество рекурсивных вызовов.

3 Что можно эффективно решать «пьяными» алгоритмами?

В этом разделе мы покажем, что «пьяные» алгоритмы могут не использовать правила удаления единичного дизъюнкта и правило чистых литералов, при этом время работы алгоритмов увеличится не сильно. Затем покажем, что существует «пьяный» алгоритм, решающий выполнимые цейтинские формулы за полиномиальное время.

Предложение 3.2. Для любого «пьяного» алгоритма $\mathcal{A}$ существует другой «пьяный» алгоритм $\mathcal{B}$, который не использует правило удаления единичных дизъюнктов и время работы которого на любом входе разве что в n раз больше, где n — это число переменных в формуле.

Доказательство. Как только в формуле образовался единичный дизъюнкт, алгоритму $\mathcal{B}$ следует произвести расщепление по переменной из этого дизъюнкта. Одна из ветвей расщепления сразу же приведет к невыполнимой формуле, а другая ветвь будет точно такой же, как и при применении правила единичного дизъюнкта. □

Предложение 3.3. Для любого «пьяного» алгоритма $\mathcal{A}$ существует другой «пьяный» алгоритм $\mathcal{C}$, который не использует ни правило удаления единичных дизъюнктов, ни правило чистых литералов, время работы которого на любом входе разве что в n^2 раз больше, где n — это число переменных в формуле.

Доказательство. Заметим, что применение правила чистых литералов соответствует вычеркиванию некоторых дизъюнктов из формулы. Поэтому на невыполнимых формулах правило чистых литералов можно просто не применять (дизъюнкты из формулы не вычеркивать, а просто считать, что их нет). На выполнимых формулах возникает дополнительная проблема: в итоге может оказаться выполнимая формула, которая полностью упрощается с помощью последовательного применения правил чистых литералов. Рассмотрим формулу, которая упрощается до пустой (т.е. выполнимой) формулы последовательностью применений правил чистых литералов. Пусть это последовательность литералов $l_1, l_2, \ldots, l_t$, где применение правила чистых литералов в указанном порядке к литералу l_i удаляет из формулы множество дизъюнктов C_i. Пусть m — это число переменных в текущей формуле. Индукцией по $t + m$ покажем, что можно избавиться от применения правил чистых литералов. База индукции $m + t = 0$. Индукционный переход. Рассмотрим непустое множество дизъюнктов C_t. Пусть X — это множество переменных, входящих в дизъюнкты C_t, отличные от переменных литерала l_t. Рассмотрим два случая: в первом случае $X = \emptyset$, тогда все дизъюнкты C_t — это единичные дизъюнкты с литералом l_t, тогда к формуле можно применить правило удаления единичного дизъюнкта и формула будет упрощаться с помощью $t - 1$ правила чистых литералов, значит можно воспользоваться индукционным предположением. Во втором случае $X \neq \emptyset$, тогда сделаем подстановку переменной из множества X (заметим, что переменные множества X не могут пересекаться с переменными, соответствующими литералам $l_1, l_2, \ldots, l_t$). После такой подстановки формула все равно упрощается применением t правил чистых литералов, но число входящих в нее переменных на 1 меньше, т.е. можно воспользоваться индукционным предположением.

Таким образом, мы избавились от применения правил чистых литералов, сделав максимум n подстановок в каждой формуле, в которой выполняющий набор был найден по средствам применения правил чистых литералов. Осталось избавиться от применений правил удаления единичных дизъюнктов по предложению 3.2. □

Далее мы будем считать, что «пьяные» алгоритмы не используют правила упрощения, а при необходимости моделируют их.

Покажем, что «пьяные» алгоритмы быстро решают выполнимые цейтинские формулы. Цейтинская формула строиться по простому связному неориентированному графу $H(V,E)$, степень которого ограничена константой d, каждому ребру $e \in E$ соответствует переменная p_e, есть функция $f : V \to \{0,1\}$, для каждой вершины $v \in E$ записывается формула в КНФ, кодирующая $\bigoplus_{u \in V:(u,v)\in E} p_{(u,v)} = f(v)$ ($\oplus$ используется для обозначения суммы по модулю 2). Конъюнкция этих формул называется цейтинской формулой. Если $\bigoplus_{v\in v} = 1$, то цейтинская формула невыполнима. Действительно, достаточно просуммировать (по модулю два) все равенства, которые записаны в вершинах и получится $0 = 1$, поскольку каждая переменная будет встречаться ровно два раза. Если $\bigoplus_{v\in v} = 0$, то цейтинская формула выполнима: присвом всем переменным значения 0, равенства не будут выполняться в четном числе вершин. Рассмотрим две вершины, в которых не выполняются равенства, и заменим значения переменных на противоположные вдоль пути между этими двумя вершинами. Заметим, что после такой операции число вершин, в которых не выполняется равенство, уменьшится на 2. Осталось повторить эту операцию, пока равенство не начнет выполняться во всех вершинах. Для графов со свойством расширения алгоритмы расщепления работают на невыполнимых цейтинских формулах экспоненциальное время [Цей68, Urq87] (формально это следует из нижней оценки на размер резолюционных доказательств).

Выполнимые цейтинские формулы могут быть быстро решены «пьяными» алгоритмами следующим образом: будем считать, что подстановка переменной в формулу удаляет из графа соответствующее ребро. Пока в графе есть циклы, «пьяный» алгоритм будет выбирать для расщепления ребро из цикла. Заметим, что после каждой такой подстановки текущая формула остается выполнимой цейтинской (при подстановки 0 функцию f менять не надо, а при подстановке 1 функцию f нужно поменять в двух точках). Если граф стал деревом, то он содержит висячую вершину (вершину степени 1), а формула, соответствующая висячей вершине, образует единичный дизъюнкт. Дальше формула решается последовательностью применений правила удаления единичного дизъюнкта (от которых можно избавиться по предложению 3.2).

4 Поведение «пьяных» алгоритмов на невыполнимых формулах

Поведение алгоритмов расщепления на невыполнимых формулах тесно связано с резолюционной системой доказательств. Резолюционная система доказательств предназначена для доказательства невыполнимости формулы в КНФ. Доказательством невыполнимости формулы φ в резолюционной системе является последовательность дизъюнктов, заканчивающаяся пустым дизъюнктом, каждый дизъюнкт является либо дизъюнктом формулы φ, либо получается из предыдущих по правилу резолюции. Правило резолюции позволяет из пары дизъюнктов $(l_1 \vee l_2 \vee \cdots \vee l_n)$ и $(l'_1 \vee l'_2 \vee \cdots \vee l'_m)$, где $l'_m = \neg l_n$ вывести дизъюнкт $(l_1 \vee \cdots \vee l_{n-1} \vee l'_1 \vee \cdots \vee l'_{m-1})$. Доказательство называется древовидным, если каждый выведенный дизъюнкт используется как посылка правила не более одного раза.

По работе любого «пьяного» алгоритма на невыполнимой формуле можно построить дерево расщепления. Вершины дерева помечены переменными, по которой производится расщепление в этой вершине, из каждой вершины (кроме листьев) исходит два ребра, одно из них помечено значением 0, другое значением 1. В каждом из листьев опровергается как минимум один из дизъюнктов исходной формулы. Размер дерева расщеплений — это время работы «пьяного» алгоритма. Индукцией по размеру дерева легко показать, что если у невыполнимой формулы φ есть дерево расщеплений размера k, то можно построить древовидное резолюционное

доказательство φ размера k. База индукции очевидна, так как, если дерево содержит одну вершину, то формула обязана содержать пустой дизъюнкт. Для индукционного перехода заметим, что дерево обязано содержать два листа u и v, у которых есть общий родитель w. Пусть расщепление в вершине w проводилось по переменной x_i, и лист u соответствовал подстановке $x_i := 1$, а v подстановке $x_i := 0$. Тогда дизъюнкты, которые опровергаются в вершинах v и u содержат переменную x_i с разными знаками, а их резольвента (результат применения правила резолюции) C обязана опровергаться в вершине w. Составим новое дерево расщеплений следующим образом: от старого отрежем листья u и v, а в вершину w запишем дизъюнкт C. Получилось корректное дерево расщеплений уже для другой формулы, которая получается из исходной добавлением резольвенты для двух дизъюнктов. К получившемуся дереву можно применить индукционное предположение (число узлов дерева уменьшилось). Таким образом, мы получаем следующее утверждение:

Предложение 4.4. Время работы «пьяного» алгоритма на невыполнимой формуле не меньше, чем размер (количество дизъюнктов) кратчайшего древовидного резолюционного доказательства.

Бен-Сасон и Вигдерсон в [BSW01] вводят понятие ширины. Ширина дизъюнкта — это количество литералов в нем. Ширина формулы в КНФ — это ширина самого широкого дизъюнкта. Ширина резолюционного доказательства формулы — это ширина самого широкого дизъюнкта, входящего в него.

Теорема 4.1 ([BSW01], следствие 3.4). Размер древовидного резолюционного доказательства формулы φ не меньше, чем 2^{w-w_φ}, где w — это минимальная ширина резолюционного доказательства φ, а w_φ — это ширина φ (ширина самого широкого дизъюнкта).

Пусть G — это граничный (r, d, c)-расширитель. С каждой вершиной множества X отождествим пропозициональную переменную. Пусть каждой вершине y_j множества Y сопоставлена формула в КНФ от переменных, смежных с y_j. Обозначим формулу, стоящую в вершине y_j через φ_j. Очевидно, что ширина каждого дизъюнкта формулы φ_j не превосходит d. Формулу, соответствующую всему множеству Y, обозначим за Φ. Для множества $A \subseteq Y$ конъюнкцию формул, соответствующих вершинам из множества A будем обозначать Φ^A.

Будем говорить, что переменная чувствительна для формулы, если заменой значения этой переменной можно поменять значение формулы на противоположное при любом фиксированном значении остальных переменных.

Теорема 4.2. Пусть каждая формула φ_j содержит не более k нечувствительных переменных. Пусть ρ такая частичная подстановка переменным из X, что

- Формула $\Phi|_\rho$ невыполнима
- Для любого множества вершин $A \subseteq Y$, такого, что $|A| < \frac{r}{2}$, формула $\Phi|_\rho^A$ выполнима

Тогда любое резолюционное доказательство формулы $\Phi|_\rho$ имеет ширину как минимум $\frac{(c-k)r}{4} - |\rho|$.

Доказательство. Рассмотрим меру Бен-Сасона-Вигдерсона, которая определена на дизъюнктах резолюционного доказательства формулы $\Phi|_\rho$. $\mu(D)$ — это размер минимального числа таких вершин A, что из формулы $\Phi^A|_\rho$ семантически следует дизъюнкт D (это означает, что любой выполняющий набор формулы $\Phi^A|_\rho$ является выполняющим набором для дизъюнкта

D). Мера μ обладает полуаддитивностью, т.е., если дизъюнкт D получается применением правила резолюции из дизъюнктов C_1 и C_2, то $\mu(D) \leq \mu(C_1)+\mu(C_2)$. Поскольку для любого $A \subseteq Y$, для которого $|A| < \frac{r}{2}$, формула $\Phi|_\rho^A$ выполнима, то мера пустого дизъюнкта не меньше $\frac{r}{2}$. Из полуаддитивности следует, что в доказательстве существует дизъюнкт C, для которого выполняется $\frac{r}{2} > \mu(C) \geq \frac{r}{4}$. Пусть A — это минимальное множество вершин, что из формулы $\Phi|_\rho^A$ семантически следует дизъюнкт C, т.е. $|A| = \mu(C) \geq \frac{r}{4}$. В графе G выполняется: $\delta(A) \geq c|A|$. $\delta(A)$ — это множество переменных, которые входят ровно в одну формулу из множества A. Из них не менее $(c-k)|A|$ переменных, которые являются чувствительными для хотя бы одной из вершин множества A. В формулу $\Phi|_\rho^A$ входит как минимум $(c-k)|A|-|\rho|$ чувствительных переменных. Покажем, что в дизъюнкт C входят все чувствительные переменные. Действительно, если это не так, то найдется переменная x_j чувствительная для вершины $v \in A$, которая не входит в дизъюнкт C. Рассмотрим множество $A \setminus \{v\}$ из него семантически не следует дизъюнкт C, значит существует такое значение переменных, при которых все формулы из $A \setminus \{v\}$ истинны, а C ложно. Но заменой значения переменной x_j легко сделать, чтобы все формулы из A были бы истинны, а C ложно, противоречие с тем, что C семантически следует из A. □

Следствие 4.1. Размер дерева расщеплений для формулы $\Phi|_\rho$ не меньше $2^{\frac{(c-k)r}{4}-|\rho|-d}$

Доказательство. Следует из теоремы 4.2, теоремы 4.1 и предложения 4.4. □

5 Поведение «пьяных» алгоритмов на выполнимых формулах

5.1 Замыкание

Пусть граф G — двудольный граничный (r, d, c)-расширитель. Пусть $c > k + 1$.

Определение 5.1. Пусть $J \subseteq X$, множество вершин $I \subseteq Y$ будем называть k-замыканием множества J, если существует такая конечная последовательность множеств $I_1, I_2, \ldots, I_m$ (обозначим $C_\ell = \bigcup_{1 \leq i \leq \ell} I_\ell$, $C_0 = \emptyset$), что выполняются следующие свойства:

- $I_\ell \subseteq Y$ и $0 < |I_\ell| \leq \frac{r}{2}$ для всех $1 \leq \ell \leq m$
- $I_i \cap I_j = \emptyset$ для всех $1 \leq i, j \leq m$
- $|\delta(I_\ell) \setminus (\Gamma(C_{\ell-1}) \cup J)| \leq (1+k)|I_k|$ для всех $1 \leq \ell \leq m$
- для всех $I' \subseteq Y \setminus C_m$, если $0 < |I'| \leq \frac{r}{2}$, то $|\delta(I') \setminus (\Gamma(C_m) \cup J)| > (1+k)|I'|$
- $I = C_m$

Множество всех k-замыканий множества J будем обозначать $Cl^k(J)$.

Лемма 5.1. (1) Для любого множества $J \subseteq X$ существует k-замыкание.
(2) Пусть $J_1 \subseteq J_2$, тогда для любого $I_1 \in Cl^k(J_1)$ существует $I_2 \in Cl^k(J_2)$, что $I_1 \subseteq I_2$

Доказательство. (1) k-замыкание может быть получено как результат работы следующего алгоритма $\mathcal{C}$ на входе $(J, \emptyset)$.

Алгоритм 5.1. Алгоритм $\mathcal{C}(J, I_0)$

1. $I := I_0$ (переменная I обозначает некоторое подмножество Y)
2. Пока существует $I' \subseteq Y \setminus I$, что $0 < |I'| \leq \frac{r}{2}$, $|\delta(I') \setminus (\Gamma(I) \cup J)| \leq (1+k)|I'|$

- $I := I \cup I'$

3. Выдать I.

(2) Пусть $I_1 \in Cl^k(J_1)$, тогда можно получить $I_2 \in Cl^k(J_2)$ как результат работы алгоритма $\mathcal{C}$ на входе (J_2, I_1). При этом $I_1 \subseteq I_2$. □

Лемма 5.2 ([AHI05]). Пусть $|J| < \frac{(c-k-1)r}{2}$, тогда для любого множества $I \in Cl^k(J)$ выполняется неравенство $|I| \leq (c-k-1)^{-1}|J|$

Доказательство. От противного. Пусть $I_1, I_2, \ldots, I_m$ — это последовательность, соответствующая k-замыканию I, $C_\ell = \bigcup_{1 \leq i \leq \ell} I_\ell$. Пусть t — это наименьшее число при котором выполняется неравенство $|C_t| > (c-k-1)^{-1}|J|$. тогда $|C_t| \leq (c-k-1)^{-1}|J| + \frac{r}{2} \leq r$. Тогда $|\delta(C_t)| \geq c|C_t| > |J| + (k+1)|C_t|$. Индукцией по ℓ можно показать, что $|\delta(C_\ell) \setminus J| \leq (k+1)(|C_\ell|)$, которое противоречит полученному ранее неравенству при $l = t$. При $l = 1$ утверждение следует из того, что $|\delta I_1 \setminus J| \leq (k+1)|I_1|$. $|\delta(C_\ell) \setminus J| \leq |\delta(I_1 \cup \cdots \cup I_{\ell-1}) \setminus J| + |\delta(I_\ell) \setminus (J \cup \Gamma(C_{\ell-1}))| \leq (k+1)((|C_{\ell-1}|)) + (k+1)|I_\ell|$.

□

5.2 Надстройка над «пьяными» алгоритмами

Будем считать, что во время своей работы «пьяный» алгоритм создает дерево расщепления. В самом начале создается корень дерева и объявляется текущей вершиной. Каждой вершине дерева соответствует текущая формула, каждому ребру соответствует подстановка одной переменной. Путь от корня до вершины задает частичную подстановку, которая соответствует объединению всех подстановок вдоль ребер этого пути. Текущая вершина дерева объявляется листом, если либо текущая формула уже выполнена (т.е. выполнены все ее дизъюнкты), либо формула содержит пустой дизъюнкт, т.е. текущая формула заведомо невыполнима. Если найден выполняющий набор, то алгоритм останавливается и выдает найденный выполняющий набор. Если текущая вершина — это лист с невыполнимой формулой, то алгоритм ищет ближайшую на пути от корня дерева до этого листа вершину, в которой стоит точка возврата и делает ее текущей (в этом случае будем говорить, что алгоритм *совершает возврат*). Если вершина с точкой возврата на пути от вершины до корня не найдена, то алгоритм останавливается и выдает ответ «формула невыполнима». Если же в текущей вершине формула не является тривиально выполненной или невыполненной и в вершине не стоит точки возврата, то алгоритм выбирает переменную для расщепления и значение для расщепления, ставит в текущую вершину точку возврата и создает потомка, соответствующего выбранной подстановки, текущей вершиной становится созданный потомок. Если же в текущей вершине стоит точка возврата, то алгоритм снимает эту точку возврата и создает потомка, соответствующую той подстановке, которая еще не делалась в текущей вершине.

Мы опишем надстройку над «пьяными» алгоритмами, которая будет немного модифицировать его поведение на формуле $\Phi_{g(x)=b}$ первые несколько шагов алгоритма, потом надстройка прекращает работать и алгоритм продолжит работать в обычном режиме. Мы гарантируем, что время работы алгоритма с надстройкой не увеличивается. (Последнее утверждение не очень понятно в случае, когда речь идет о вероятностных алгоритмах. В нашем случае происходит следующее: исходный алгоритм использует p случайных битов, а алгоритм с надстройкой q, где $q \leq p$, тогда для каждой строки r случайных битов длины q можно указать

2^{p-q} строк случайных битов длины p так, что время работы исходного алгоритма на этих строках не меньше времени работы алгоритма с надстройкой на строке r, кроме того указанное соответствие покрывает все строчки длины p.)

Надстройка хранит свою частичную подстановку π, если подстановка π подставляет некоторое значение переменной x, то переменную x мы называем *форсированной*. Если «пьяный» алгоритм будет пытаться подставить значение форсированной переменной, отличное от значения, которое присваивает ей π, то надстройка не дает ему это сделать. Другими словами из дерева расщепления отрезается поддерево, причем мы гарантируем, что отрезается поддерево, соответствующая невыполнимой формуле, т.е. другое значение переменной алгоритму все равно нужно было бы подставлять. Кроме того гарантируется, что пока работает надстройка, алгоритм не будет совершать возвратов (т.е. все возвраты попадут в отрезанные поддеревья).

Опишем формально нашу надстройку. Пусть «пьяный» алгоритм $\mathcal{A}$ получает на вход выполнимую формулу $\Phi_{g(x)=b}$, G — это граничный (r,d,c)-расширитель, k — это количество нечувствительных переменных предиката P, $k+1<c$.

1. $J:=\emptyset$, $I:=\emptyset$ (всегда выполняется $I\in Cl^k(J)$), $\rho:=\emptyset$ (текущая подстановка)
2. $\pi:=\emptyset$ (т.е. ни одна переменная не является форсированной).
3. Пока $|J|<\frac{r(c-k-1)}{16d}$ и $|\rho|<n$ выполнять
   (a) Если алгоритм $\mathcal{A}$ собирается закончить работу или совершить возврат, то выйти из цикла.
   (b) Пусть $\mathcal{A}$ выбирает переменную x_j для расщепления.
   (c) Если переменная x_j является форсированной и π содержит подстановку $x_j:=a$, то $\rho:=\rho\cup\{x_j:=a\}$. В дерево расщепления добавляется один потомок, точка возврата не ставится.
   (d) В противном случае, переменная x_j не является форсированной, тогда
   - Пусть алгоритм $\mathcal{A}$ выбирает значение a, тогда $J:=J\cup\{x_j\}$, $\rho:=\rho\cup\{x_j:=a\}$. В текущую вершину ставится точка возврата.
   - Расширить I до элемента $Cl^k(J)$ (это возможно по пункту (2) леммы 5.1).
   - Если для какой-то переменной x_j из $\Gamma(I)$ и для некоторого $a\in\{0,1\}$, значение $x_j=a$ семантически следует из формулы $\Phi^I_{g(x)=b}|_\rho$, то $\pi:=\pi\cup\{x_j:=a\}$. (Формально возможно, что из формулы $\Phi^I_{g(x)=b}|_\rho$ следует как $x_j=0$, так и $x_j=1$, в таком случае в π добавляется только одна подстановка. Позже мы докажем, что такого быть не может.)
   - Создать потомка в дереве, соответствующего сделанной подстановке, и сделать его текущей вершиной.
4. Моделировать $\mathcal{A}$ без изменений на формуле $\Phi_{g(x)=b}|_\rho$ в текущей вершине дерева.

Пусть цикл на шаге 3 описанной выше надстройки выполнялся (до конца) t раз. Для $0\le i\le t$ обозначим через J_i,I_i,ρ_i значения переменных J,I,ρ перед $(i+1)$-м выполнением цикла на шаге 3 (I_t,J_t,ρ_t — это значение после выполнения t-ой итерации цикла).

Из следующей леммы следует, что, пока работает надстройка, алгоритм не совершает возвратов.

Лемма 5.3. Для всех $0 \le i \le t$ и для всех $A \subseteq Y$, для которых $|A| \le \frac{r}{2}$, формула $\Phi^A_{g(x)=b}|_{\rho_i}$ выполнима.

Доказательство. Доказательство по индукции. Проверим, что условие выполняется при $i = 0$, $\rho_0 = \emptyset$. От противного, рассмотрим самое маленькое такое множество $A \subset Y$, $|A| \le \frac{r}{2}$, что формула $\Phi^A_{g(x)=b}$ невыполнима. Поскольку G — граничный расширитель, то $|\delta(A)| \ge c|A|$, следовательно есть хотя бы $(c-k)|A|$ чувствительных граничных переменных для формул, соотвествующих множеству A, но любой такой граничной переменной можно изменить значение формулы, соответствующей одной из вершин множества A, т.е., эту формулу (и вершину) можно было бы выкинуть с сохранением невыполнимости, получаем противоречие с минимальностью A.

Индукционный переход. От противного, пусть утверждение неверно, рассмотрим минимальное такое множество $A \subseteq Y$, $|A| \le \frac{r}{2}$, что формула $\Phi^A_{g(x)=b}|_{\rho_{i+1}}$ невыполнима. Пусть $A_1 = A \cap I_{i+1}$, $A_2 = A \setminus I_{i+1}$. Если A_2 не пусто, то из определения 5.1 следует, что $|\delta(A_2) \setminus (\Gamma(I_{i+1}) \cup J)| > (k+1)|A_2|$, поэтому хотя бы одну вершину из множества A_2 можно выкинуть, так как формулу в ней можно выполнить за счет граничной чувствительной переменной (такая есть, так как в каждой вершине максимум k нечувствительных переменных), что противоречит минимальности A. Значит, $A_2 = \emptyset$ и $A \subseteq I_{i+1}$.

Разобьем A на $A' = A \cap I_i$ и $A'' = A \setminus I_i$. Рассмотрим случай, когда $A'' = \emptyset$. По индукционному предположению формула $\Phi^{A'}_{g(x)=b}|_{\rho_i}$ выполнима, поскольку $|A'| \le |A| \le \frac{r}{2}$. Формула $\Phi^{A'}_{g(x)=b}|_{\rho_{i+1}}$ тоже выполнима, поскольку ρ_{i+1} отличается от ρ_i только одной подстановкой, а переменные форсированы подстановкой π, если их значение семантически следуют из формулы $\Phi^{I_i}_{g(x)=b}|_{\rho_i}$, это значит, что невозможно, чтобы алгоритм $\mathcal{A}$ сделал одной подстановкой (на $(i+1)$-й итерации цикла) формулу $\Phi^{I_i}_{g(x)=b}|_{\rho_i}$ невыполнимой.

Пусть $A'' \ne \emptyset$. Поскольку $|A''| \le \frac{r}{2}$ и $A'' \cap I_i = \emptyset$, то $|\delta(A'') \setminus (\Gamma(I_i) \cup J)| > (k+1)|A''|$, т.е. множество A'' содержит не менее двух чувствительных граничных переменных (которые не попали в $\Gamma(I_i) \cup J$), поэтому после подстановки значения одной переменной останется хотя бы одна чувствительная граничная переменная, т.е. как минимум одну вершину из A'' можно выкинуть с сохранением невыполнимости $\Phi^A_{g(x)=b}|_{\rho_{i+1}}$, что противоречит минимальности A. □

Покажем, что пока работает надстройка, алгоритм не успеет найти выполняющий набор. Пока работает надстройка для всех $0 \le i \le t$ выполняется $|J_i| \le \frac{r(c-k-1)}{16d}$, тогда по лемме 5.2 $|I_i| \le \frac{r}{16d}$, тогда $|\Gamma(I_i)| \le \frac{r}{16}$. Количество переменных, которые подставлены во время работы надстройки не превосходит $|\Gamma(I_t)| \cup |J_t| \le \frac{r}{8}$ ($|J_t| \le \frac{r}{8}$, поскольку $c \le d$ в любом граничном (r,d,c)-расширителе). Этого не хватит, чтобы выполнить формулу, поскольку любое множество $A \subseteq Y$ размера r содержит как минимум r чувствительных переменных. А чтобы выполнить формулу нужно сделать подстановку всем чувствительным переменным.

Лемма 5.4. Пусть g — это функция Голдрейха, построенная по граничному (r,d,c)-расширителю G и предикату P, в котором не более k нечувствительных переменных, $c > k+1$. Пусть при данном b уравнение $g(x) = b$ имеет не более, чем $2^{\frac{r(c-k-1)}{64d}}$ решений. Тогда с вероятностью $1 - 2^{-\Omega(r)}$ «пьяный» алгоритм будет работать на формуле $\Phi_{g(x)=b}$ время $2^{\Omega(r)}$. В асимптотических обозначениях c, k и d считаются константами.

Доказательство. Поскольку надстройка не увеличивает времени работы алгоритма (она только обрезает заведомо невыполнимые ветви), то достаточно оценить время работы алгоритма с надстройкой. Поскольку надстройка работает пока $|J| \le \frac{r(c-k-1)}{8d}$, $|I| \in Cl^k(J)$, тогда из леммы 5.2 следует, что $|I| \le \frac{r}{16d}$. Тогда $|\Gamma(I)| \le \frac{r}{16}$, значит, пока работала надстройка, максимум

было сделано $|\Gamma(I)|+|J| \le \frac{r}{8}$ подстановок. J соответствует подстановкам, при которых ставится точка возврата (иначе говоря, расщеплениям). Значение, которое подставляется первым, выбирается случайным образом. Для формулы φ будем называть подстановку $x := b$ *удачной*, если она согласуется более, чем с половиной выполняющих наборов φ и *неудачной* в противном случае.

Пока надстройка работает, «пьяный» алгоритм делает $\frac{r(c-k-1)}{16d}$ подстановок, каждый раз, выбирая значение переменной случайным образом. С вероятностью $\frac{1}{2}$ выбирается неудачное значение переменной, т.е. число выполняющих наборов уменьшается как минимум в 2 раза. Из оценок Чернова следует, что с вероятностью $1-2^{-\Omega(r)}$ было сделано не меньше, чем $\frac{r(c-k-1)}{64d}$ подстановок, которые подставляли неудачное значение переменной. Таким образом, с вероятностью $1-2^{-\Omega(r)}$ после работы надстройки получается невыполнимая формула, а размер подстановки ρ при этом не превосходит $\frac{r}{8}$. Утверждение леммы следует теперь из следствия 4.1, в котором доказана нижняя оценка для невыполнимых формул. □

5.3 Оценка числа решений

В этом разделе мы оцениваем количество прообразов функции Голдрейха, основанной на предикате $P_d(x_1, x_2, \ldots, x_d) = x_1 \oplus x_2 \oplus \cdots \oplus x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$, где Q — произвольный k-местный предикат, $k+1 < \frac{d}{4}$.

Теорема 5.1 (ср. [CEMT09], теорема 4.1). Пусть предикат $P_d(x_1, x_2, \ldots, x_d) = x_1 \oplus x_2 \oplus \cdots \oplus x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$, где Q — произвольный k-местный предикат, $k+1 < \frac{d}{4}$. Граф G строится случайным образом: для каждой вершины из доли Y случайным образом независимо выбираются d ребер, ведущие в долю X (повторения разрешаются). Тогда $\mathrm{E}[\#(x,y) \mid g(x) = g(y)] = 2^{(1+2^{-\Omega(d)})n}$, где g — это функция Голдрейха, построенная по графу G и предикату P_d.

Доказательство. $\mathrm{E}[\#(x,y) \mid g(x) = g(y)] = \sum_{x,y \in \{0,1\}^n} \Pr[g(x) = g(y)]$.

Пусть $M \subseteq \{1, 2, \ldots, m\}$ — множество номеров битов, в которых различаются строки x и y, обозначим $m = |M|$. Поскольку для каждой вершины из Y графа G ребра выбираются независимо, то $\Pr[g(x) = g(y)] = (\Pr[(g(x))_1 = (g(y))_1])^n$.

$\Pr[(g(x))_1 = (g(y))_1] = \Pr[P_d(x_{i_1}, x_{i_2}, \ldots, x_{i_d}) = P_d(y_{i_1}, y_{i_2}, \ldots, y_{i_d})]$, где $i_1, i_2, \ldots, i_d$ — независимые случайные величины, принимающие значения от 1 до n с равными вероятностями.

Поскольку предикат Q нам неизвестен, распишем искомую вероятность как сумму двух условных: $\Pr[P_d(x_{i_1}, x_{i_2}, \ldots, x_{i_d}) = P_d(y_{i_1}, y_{i_2}, \ldots, y_{i_d})] = \Pr[P_d(x_{i_1}, x_{i_2}, \ldots, x_{i_d}) = P_d(y_{i_1}, y_{i_2}, \ldots, y_{i_d}) \mid Q(x_{i_{d-k+1}}, \ldots, x_{i_d}) = Q(y_{i_{d-k+1}}, \ldots, y_{i_d})] \cdot q + \Pr[P_d(x_{i_1}, x_{i_2}, \ldots, x_{i_d}) = P_d(y_{i_1}, y_{i_2}, \ldots, y_{i_d}) \mid Q(x_{i_{d-k+1}}, \ldots, x_{i_d}) \ne Q(y_{i_{d-k+1}}, \ldots, y_{i_d})] \cdot (1-q)$, где $q = \Pr[Q(x_{i_{d-k+1}}, \ldots, x_{i_d}) = Q(y_{i_{d-k+1}}, \ldots, y_{i_d}]$. Теперь оценим каждую из условных вероятностей отдельно. В первом случае нужно посчитать вероятность того, что из $i_1, i_2, \ldots, i_{d-k}$ четное число принадлежит M, во втором случае, что это число нечетное. Рассмотрим последовательность p_j — вероятность того, что из $i_1, i_2, \ldots, i_j$ четное число принадлежит M. Из независимости событий легко вывести следующее рекуррентное соотношение: $p_{j+1} = \alpha(1-p_j) + (1-\alpha)p_j = \alpha + (1-2\alpha)p_j$, где $\alpha = \frac{m}{n}$, $p_0 = 1$. Решая это рекуррентное соотношение, получаем, что $p_j = \frac{1+(1-2\alpha)^j}{2}$, значит $1-p_j = \frac{1-(1-2\alpha)^j}{2}$. Таким образом, $\Pr[f(x) = f(y)] \le \left(\frac{1+|1-2\alpha|^{d-k}}{2}\right)^n$.

$$\mathrm{E}[\#(x,y) \mid f(x)=f(y)] = \sum_{x\in\{0,1\}^n} \sum_{m=1}^{n} \sum_{y:\ \delta(x,y)=m} \Pr[f(x)=f(y)]$$
$$\leq 2^n \sum_k C_n^m \left(\frac{1+|1-2\frac{k}{n}|^{d-k}}{2}\right)^n \leq n \cdot \max_{0\leq\alpha\leq\frac{1}{2}} 2^{H(\alpha)n}(1+(1-2\alpha)^{d-k})^n,$$

где $\delta(x,y)$ — это количество битов, в которых отличаются строчки x и y, $H(\alpha) = -\alpha\log\alpha - (1-\alpha)\log(1-\alpha)$ — бинарная энтропия, log обозначает логарифм по основанию 2.

Лемма 5.5 ([CEMT09], из доказательства теоремы 4.1). Существует $\epsilon > 0$, что для всех $\alpha \in [0, \frac{1}{2}]$ при достаточно больших d выполняется неравенство $H(\alpha) + \log_2(1+(1-2\alpha)^{d-k}) \leq 1 + 2^{-\epsilon d}$, (где $k+1 < \frac{d}{4}$).

Доказательство. Рассмотрим 4 случая, в зависимости от величины α. Значение констант $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4$ мы выберем по ходу дела.

Случай 1: $\alpha > \epsilon_1$. $H(\alpha) + \log_2(1+(1-2\alpha)^{d-k}) \overset{\ln(1+x)\leq x}{\leq} 1 + (1-2\epsilon_1)^{\frac{3d}{4}}/\ln 2 \leq 1 + 2^{-\epsilon d}$, при $\epsilon < -\frac{3}{4}\log(1-2\epsilon_1)$ и больших d.

В остальных случаях $H(\alpha)$ мало. Из формулы Тейлора для $\ln x$ в точке 2 следует следующее неравенство: $\ln(1+x) \leq \ln 2 + \frac{x-1}{2}$. Тогда можно написать следующую оценку:

$$\log(1+(1-2\alpha)^{d-k}) \leq 1 + \frac{(1-2\alpha)^{d-k}-1}{2\ln 2} \overset{1+x\leq e^x}{\leq} 1 + \frac{e^{-2\alpha(d-k)}-1}{2\ln 2} \leq 1 + \frac{e^{-3\alpha\cdot d/2}-1}{2\ln 2}$$

Случай 2: $\epsilon_1 \geq \alpha > \epsilon_2/d$.

$$H(\alpha) + \log(1+(1-2\alpha)^{d-k}) \leq H(\epsilon_1) + 1 + \frac{e^{-3\epsilon_2/2}-1}{2\ln 2} \leq 1,$$

если мы выберем такое маленькое ϵ_1, что $H(\epsilon_1) < \frac{1-e^{-3\epsilon_2/2}}{2\ln 2}$.

При $0 \leq \alpha \leq \frac{1}{2}$ выполняется неравенство $(\alpha-1)\log(1-\alpha) \leq 2\alpha$ (при $\alpha = 0$ неравенство обращается в равенство, взятием производной можно убедиться, что разница между левой и правой частью неравенства увеличивается с ростом α). Можно написать следующую оценку $H(\alpha) \leq \alpha\log\frac{1}{\alpha} + 2\alpha$.

Для оставшихся случаев (когда $\alpha \leq \epsilon_2/d$) выберем $\epsilon_2 = \frac{1}{3}$, тогда $\frac{3}{2}\alpha d \leq \frac{1}{2}$. При $-\frac{1}{2} \leq x \leq 0$ выполняется неравенство $e^x - 1 \leq \frac{x}{2}$ (проверяется дифференцированием). Тогда можно оценить следующим образом:

$H(\alpha) + 1 + \frac{e^{-\frac{3}{2}\alpha d}-1}{2\ln 2} \leq (\alpha\log\frac{1}{\alpha} + 2\alpha) + 1 - \frac{3\alpha d}{8\ln 2} = 1 + \alpha(\log\frac{1}{\alpha} - \frac{3}{8\ln 2}\cdot d + 2)$

Случай 3: $\epsilon_2/d \geq \alpha > 2^{-\epsilon_3 d}$. При $\epsilon_3 < \frac{3}{8\ln 2}$ и достаточно большом d: $(\log\frac{1}{\alpha} - \frac{3}{8\ln 2}\cdot d + 2) < 0$.

Случай 4: $2^{-\epsilon_3 d} \geq \alpha$. Для $\epsilon < \epsilon_3$ и достоточно больших d выполняется: $\alpha\log\frac{1}{\alpha} \leq \epsilon_3 d 2^{-\epsilon_3 d} \leq 2^{-\epsilon d}$.

□

Из леммы 5.5 следует утверждение теоремы. □

5.4 Нижняя оценка

Теперь мы можем доказать основную теорему:

Теорема 5.2. Пусть $P_d(x_1, x_2, \ldots, x_d) = x_1 \oplus \ldots x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$, где Q — произвольный предикат, $k+1 < \frac{d}{4}$. Для достаточно больших d и для достаточно больших n случайный граф G обладает с вероятностью хотя бы 0.85 следующим свойством. Для любого «пьяного» алгоритма $\mathcal{A}$, $\Pr_{y \leftarrow U_n}[\Pr[t^{\mathcal{A}}_{\Phi_{g(x)=g(y)}} > 2^{\Omega(n)}] > 1 - 2^{-\Omega(n)}] > 0.9$, где $t^{\mathcal{A}}_{\Phi}$ обозначает время работы алгоритма $\mathcal{A}$ на формуле Φ.

Доказательство. По следствию 2.1 случайный граф с вероятностью 0.9 является граничным $(\frac{n}{10d}, d, \frac{1}{4}d)$-расширителем. По теореме 5.1 среднее число пар x и y, при которых $g(x) = g(y)$ равняется $2^{(1+2^{-\Omega(d)})n}$, где усреднение берется по случайным графам. По неравенству Маркова с вероятностью не меньше 0.95 для случайного графа число пар x и y, при которых $g(x) = g(y)$ равняется $2^{(1+2^{-\Omega(d)})n}$ (константа ушла в $\Omega(d)$). Из этого следует, что с вероятностью не меньше, чем 0.85 случайный граф является граничным экспандером и верна верхняя оценка на число пар с одинаковым значением. Зафиксируем такой граф G. Из неравенства Маркова следует, что как минимум для доли 0.9 строчек $y \in \{0,1\}^n$ выполняется неравенство $|g^{-1}(g(y))| < 2^{2^{-\Omega(d)}n}$. Предикат P содержит не более k нечувствительных переменных (нечувствительными могут быть только переменные $x_{d-k+1}, \ldots, x_d$), тогда по лемме 5.4 время работы любого «пьяного» алгоритма на формуле $\Phi_{g(x)=g(y)}$ с вероятностью $1 - 2^{-\Omega(n)}$ не менее $2^{\Omega(n)}$. □

Благодарности. Автор выражает благодарность И. Миронову, который обратил внимание автора на [CEMT09], и Э.А. Гиршу за плодотворные обсуждения и ценные комментарии.

Список литературы

[Цей68] Г. С. Цейтин. О сложности вывода в исчислении высказываний. *Записки научных семинаров ЛОМИ*, 8:234–259, 1968.

[ABSRW00] M. Alekhnovich, E. Ben-Sasson, A. A. Razborov, and A. Wigderson. Pseudorandom generators in propositional proof complexity. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 43, Washington, DC, USA, 2000. IEEE Computer Society.

[AHI05] Michael Alekhnovich, Edward A. Hirsch, and Dmitry Itsykson. Exponential lower bounds for the running time of DPLL algorithms on satisfiable formulas. *J. Autom. Reason.*, 35(1-3):51–72, 2005.

[BSW01] E. Ben-Sasson and A. Wigderson. Short proofs are narrow — resolution made simple. *Journal of ACM*, 48(2):149–169, 2001.

[CEMT09] James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. Goldreich's one-way function candidate and myopic backtracking algorithms. In *proceedings of TCC*, pages 521–538. Springer-Verlag, 2009.

[DLL62] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5:394–397, 1962.

[DP60] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.

[EB05] Niklas Eén and Armin Biere. Effective preprocessing in SAT through variable and clause elimination. *Theory and Applications of Satisfiability Testing*, pages 61–75, 2005.

[ES03] Niklas Een and Niklas Sorensson. An extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *SAT*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. Technical Report 00-090, Electronic Colloquium on Computational Complexity, 2000.

[HLW06] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43:439–561, 2006.

[MZ06] Ilya Mironov and Lintao Zhang. Applications of SAT solvers to cryptanalysis of hash functions. In Armin Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing—SAT 2006*, volume 4121 of *Lecture Notes in Computer Science*, pages 102–115. Springer, August 2006.

[NW94] N. Nisan and Avi Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[Urq87] A. Urquhart. Hard examples for resolution. *JACM*, 34(1):209–219, 1987.