УЧРЕЖДЕНИЕ РОССИЙСКОЙ АКАДЕМИИ НАУК
САНКТ-ПЕТЕРБУРГСКИЙ АКАДЕМИЧЕСКИЙ УНИВЕРСИТЕТ –
НАУЧНО-ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР НАНОТЕХНОЛОГИЙ РАН

На правах рукописи

Диссертация допущена к защите
Зав. кафедрой

_____

“ ”_____2011 г.

ДИССЕРТАЦИЯ

НА СОИСКАНИЕ УЧЕНОЙ СТЕПЕНИ

МАГИСТРА

Тема: Трудные примеры для эвристических DPLL алгоритмов для SAT

Направление: 010600.68 – Прикладные математика и физика

Магистерская программа: “Математические и информационные технологии”

Выполнил студент: Д. О. Соколов

(подпись)

Руководитель:
к.ф.-м.н. Д. М. Ицыксон

(подпись)

Рецензент:
к.ф.-м.н. С. И. Николенко

(подпись)

Санкт-Петербург
2011 г.

Реферат

Работа выполнена на 29 страницах в 3 главах, использовано 7 источников.

Известны экспоненциальные нижние оценки на время работы DPLL алгоритмов. На невыполнимой формуле нижняя оценка на время работы следует из нижней оценки на длину резолюционных доказательств, в работах [1, 2] доказываются нижние оценки на время работы близоруких DPLL алгоритмов на выполнимых формулах. В настоящей работе рассматриваются близорукие DPLL алгоритмы с возможностью отсечения ветвей дерева расщепления. Для каждого близорукого алгоритма с отсечением мы строим семейство пар (невыполнимая формула Φ, моделируемое за полиномиальное время распределение на выполнимых формулах $\mathcal{D}$) таких, что либо среднее время работы алгоритма на невыполнимой формуле Φ не менее $(1-\epsilon)2^n$, где n число переменных, либо вероятность правильной работы на распределении $\mathcal{D}$ не более $(1-\epsilon)$.

Содержание

Введение

DPLL алгоритмы. Одним из основных подходов к решению задачи выполнимости пропозициональных формул являются DPLL алгоритмы (алгоритмы расщепления). Данные алгоритмы на вход получают булеву формулу, а на выходе выдается выполняющий набор или сообщение о том, что формула является невыполнимой. DPLL алгоритм — рекурсивный алгоритм. Сначала он упрощает входную формулу Φ, затем выбирает одну переменную x и пробует присвоить ей значение c. Затем алгоритм рекурсивно вызывает себя на формуле $\Phi[x := c]$. Если был найден выполняющий набор, то он выдается, иначе возвращается результат запуска алгоритма на формуле $\Phi[x := 1-c]$. Рекурсивные вызовы прекращаются, когда формула становится тривиальной.

DPLL алгоритм параметризуется двумя эвристиками: A — выбирает переменную, по которой в данный момент нужно выполнить расщепление, B — выбирает, какое первое значение следует подставить переменной.

По работе DPLL алгоритма можно построить дерево расщеплений, где вершины соответствуют рекурсивным вызовам. По дереву расщеплений на невыполнимой формуле нетрудно построить резолюционное доказательство этой формулы, длина которого равна размеру дерева. Пользуясь нижними оценками на размер резолюционных доказательств ([3]), можно получить экспоненциальную нижнюю оценку на время работы DPLL алгоритмов на невыполнимых формулах.

Для выполнимых формул доказать суперполиномиальную нижнюю оценку на время работы не ограничивая эвристики, скорее всего не удастся, поскольку, если $\mathbf{P} = \mathbf{NP}$, то эвристика B может всегда подставлять значение из выполняющего набора. Известны экспоненциальные нижние оценки времени работы на выполнимых формулах для двух разновидностей DPLL алгоритмов. В работе [1] доказывается экспоненциальная нижняя оценка на время работы двух классов DPLL алгоритмов на выполнимых формулах: близоруких и “пьяных”. Эти классы покрывают многие известные DPLL алгоритмы. В близоруких алгоритмах эвристики выбора переменной для расщепления и выбора первого значения имеют следующие ограничения: они видят формулу, в которой стерты все знаки отрицания, им доступна информация о числе положительных и отрицательных вхож-

дений для каждой переменной. Кроме того, они могут запросить $K = n^{1-\epsilon}$ дизъюнктов точно. Для "пьяных" алгоритмов эвристика выбора переменной для расщепления может быть любой, а первое значение выбирается равновероятно случайным образом. Нижние оценки для близоруких алгоритмов были доказаны на формулах, кодирующих систему линейных уравнений для матриц-экспандеров, а нижние оценки для "пьяных" алгоритмов достигались на формулах, которые строятся по любому семейству трудных для резолюций формул.

В работе [1] существенным недостатком было то, что формулы, на которых доказывалась нижняя оценка, получались путем кодирования линейной системы уравнений (т.е. заведомо существует полиномиальный алгоритм, решающий задачу выполнимости на подобной формуле). В работе [2] этот недостаток был устранен для ослабленного. В отличие от [1], эти алгоритмы не могли использовать правила упрощения "чистые литералы", и на каждом шаге алгоритмам разрешалось читать только константное число дизъюнктов. варианта близоруких алгоритмов. В работе [4] была доказана экспоненциальная нижняя оценка для нелинейного случая на время работы "пьяных" DPLL алгоритмов. В работе [5] предлагается явная конструкция формул, обеспечивающих экспоненциальную нижнюю оценку. Также результат [2] был упрощен и усилен на "неурезанную" версию близоруких алгоритмов.

В данной работе расширяются возможности DPLL алгоритмов добавлением эвристики C, которая может обрезать ветви у дерева расщепления, т.е. если эвристика C выдает 0, то алгоритм вместо рекурсивных вызовов сразу выдает, что формула невыполнима. Заметим, что теперь DPLL алгоритм может ошибаться на выполнимых формулах. Однако, если бы доля ошибок была небольшой, но при этом время работы на невыполнимых формулах стало бы полиномиальным, то подобным алгоритмом можно было бы пользоваться на практике. Все результаты о нижних оценках на время работы DPLL алгоритмов на выполнимых формулах получаются путем доказательства того факта, что через несколько шагов расщепления с большой вероятностью получится невыполнимая формула, после чего применяется оценка на размер дерева расщепления на невыполнимой формуле. В случае DPLL алгоритмов с отсечением трудности возникают уже с оценками на невыполнимых формулах.

Если разрешить эвристике A видеть всю формулу, то, если $\mathbf{P} = \mathbf{NP}$, то существует DPLL алгоритм, который не ошибается на выполнимых фор-

мулах и работает полиномиальное время на невыполнимых. Приведем пример подобного алгоритма: если формула выполнима, то A просит подставить переменную x_1, иначе x_n. Если была подставлена переменная x_n, эвристика C должна обрезать все ветви, иначе эвристика B (решив задачу выполнимости) в первой же ветви найдет выполняющий набор. Таким образом, важным условием на ограничения эвристик является то, чтобы эвристике C нельзя было бы передать информацию о выполнимости формулы.

Результаты. В данной работе для любой детерминированной близорукой эвристики A построено семейство пар $(\Phi_n, \mathcal{D}_n)$, где Φ_n — невыполнимая формула, $\{\mathcal{D}_n\}_{i=1}^{\infty}$ — полиномиально моделируемое распределение, такое, что для любой эвристики B и любой близорукой эвристики C верно, что алгоритм, параметризованный эвристиками A, B, C, либо работает время, пропорциональное $(1-\epsilon)2^{\min(n^\delta, \Omega(\frac{n}{K}))}$, либо работает правильно на распределении с вероятностью менее $(1-\epsilon)$.

План доказательства. Основная идея заключается в том, чтобы рассмотреть невыполнимую формулу Φ, которую легко построить. Модифицируем эту формулу так, чтобы работа близорукого алгоритма на модифицированной формуле не отличалась от работы на исходной формуле Φ. Для этих целей подходят формулы, полученные из функции Голдрейха, основанной на экспандерах. Однако от экспандера потребуется ограниченность степеней всех вершин и полный ранг.

Таким образом, план доказательства разбивается на несколько частей.

1. Модификация экспандера так, чтобы его степень стала ограниченной, а ранг матрицы смежности стал полным.
2. Надстройка над DPLL алгоритмами (умный близорукий алгоритм), позволяющая проводить только корректные модификации невыполнимой формулы.
3. При помощи моделирования надстройки выбор $2^{\min(n^\delta, \Omega(\frac{n}{K}))}$ вершин в дереве расщепления, создаваемом алгоритмом без отсечений на невыполнимой формуле.
4. При помощи выбранных вершин конструирование выполнимых формул, на которых алгоритм ведет себя “похожим” образом.
5. Доказательство того, что если алгоритм с отсечениями посетит не более чем долю $(1-\epsilon)$ среди выбранных вершин, вероятность кор-

ректной работы на распределении, построенном из выполнимых формул в предыдущем пункте, не превосходит $(1 - \epsilon)$.

Глава 1. Основные понятия

Пусть $X = \{x_1, x_2, \ldots, x_n\}$ — множество пропозициональных переменных.

Частичной подстановкой назовем функцию $\rho : X \to \{0, 1, *\}$, которая по переменной выдает значение или информацию о том, что переменная еще не подставлена. Множество $\mathrm{Vars}(\rho) = \rho^{-1}(\{0, 1\})$ называется носителем подстановки; обозначим $|\rho| = |\mathrm{Vars}(\rho)|$.

Пусть ρ_1 и ρ_2 — частичные подстановки с непересекающимися носителями. Тогда естественным образом можно определить подстановку $\rho_1 \cup \rho_2(x_i) = \begin{cases} \rho_1(x_i), \ x_i \in \rho_1^{-1}\{0,1\}. \\ \rho_2(x_i), \ x_i \notin \rho_1^{-1}\{0,1\}. \end{cases}$

Будем говорить, что строка $y \in \{0, 1\}^n$ согласуется с частичной подстановкой ρ (обозначаем $y \sim \rho$), если для всех x_j из носителя ρ выполняется $y_j = \rho(x_j)$.

1.1. DPLL алгоритмы с отсечениями

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

- Процедура **A**, которая по формуле в КНФ выдает переменную из этой формулы. 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.
- Процедура **C**, которая по формуле выдает значение из $\{0, 1\}$. Значение 0 соответствует тому, что дальше расщепляться не нужно.

Будем говорить, что DPLL-алгоритм без отсечений, если вместо результата эвристики C всегда подставляется единица.

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 и работает следующим образом.

Алгоритм 1.1. Вход: формула φ и подстановка ρ

1. Упростить φ с помощью правил упрощения (считаем, что правила упрощения меняют φ и ρ, причем все переменные, подставленные подстановкой ρ, удаляются из формулы φ).
2. Запустить $\mathbf{C}(\varphi)$. Если получился 0, то выдать «формула невыполнима» и закончить работу.
3. 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».
4. $x_j := \mathbf{A}(\varphi)$.
5. $c := \mathbf{B}(\varphi, x_j)$.
6. Запустить алгоритм рекурсивно на $(\varphi[x_j := c], \rho \cup \{x_j := c\})$; если алгоритм выдал подстановку, то выдать ее, в противном случае рекурсивно запустить на $(\varphi[x_j := 1 - c], \rho \cup \{x_j := 1 - c\})$; если рекурсивный вызов выдал подстановку, то выдать ее, иначе выдать «формула невыполнима».

Определение 1.1. Близорукие алгоритмы [1] — это алгоритмы расщепления, в которых эвристики $\mathbf{A}, \mathbf{C}$ имеют следующие ограничения:

- они видят формулу, в которой все знаки отрицаний стерты;

- для каждой переменной им известно количество положительных и отрицательных ее вхождений в формулу;
- эвристика $\mathbf{A}$ может запросить $K = o(n)$ дизъюнктов, которые можно прочитать полностью (со знаками отрицаний); эти дизъюнкты становятся видны всем эвристикам.

Правила упрощения:

- *правило удаления единичного дизъюнкта*: если формула содержит дизъюнкт, в котором ровно один литерал, то сделать такое присваивание переменной этого литерала, чтобы выполнить этот дизъюнкт;
- *правило чистых литералов*: если формула содержит переменную, которая входит только с одним знаком, то присвоить ей такое значение, которое выполнит все дизъюнкты, в которые эта переменная входит.

Будем рассматривать только такие близорукие алгоритмы, у которых эвристика $\mathbf{A}$ детерминирована.

Временем работы алгоритма расщепления будем считать количество рекурсивных вызовов.

Лемма 1.1. Для любого близорукого алгоритма $\mathcal{A}$ найдется такой близорукий алгоритм $\mathcal{B}$, не использующий правила чистых литералов и единичных дизъюнктов, что время работы алгоритма $\mathcal{B}$ ограничено полиномом от времени работы алгоритма $\mathcal{A}$.

Доказательство. Если текущий предикат (с учетом сделанной подстановки ρ) в вершине $y \in Y$ является чистым, то все уравнения, в которые он попадает, имеют вид $x = c_i$, следовательно эвристика B сможет сделать правильную подстановку, т.к. она видит все биты. Аналогично, если формула содержит единичный дизъюнкт. □

Далее считаем, что близорукий алгоритм не использует правил упрощения.

Через T_Φ обозначим дерево расщеплений, создаваемое близоруким алгоритмом без отсечений на формуле Φ.

Определение 1.2. Назовем вершину $v \in T_\Phi$ точкой расщепления, если у нее ровно два потомка.

Определение 1.3. Пусть $v \in T_\Phi$ — точка расщепления. Уровнем точки расщепления $l(v)$ назовем число точек расщепления от нее до корня.

С каждой вершиной дерева расщеплений v свяжем частичную подстановку ρ_v, которая приводит алгоритм в данную вершину.

1.2. Экспандеры

Мы рассматриваем двудольные графы, в каждой из долей которых находится по n вершин. Первую долю мы обозначаем $X = \{x_1, x_2, \ldots, x_n\}$, вторую — $Y = \{y_1, y_2, \ldots, y_n\}$. Для каждой вершины доли Y есть упорядоченный список ее соседей из доли X, причем повторения в этом списке (кратные ребра) разрешаются. Все рассматриваемые нами графы будут d-регулярными, т.е. степень любой вершины множества Y равна некоторой константе d.

Каждому графу мы ставим в соответствие матрицу $n \times n$ над полем $\mathbb{F}_2$. Строки этой матрицы соответствуют вершинам множества Y, а столбцы — вершинам множества X. Число, стоящее, в клетке с координатами (y, x) соответствует четности числа ребер между вершинами y и x. Такую матрицу будем называть матрицей смежности графа.

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 1.4. Граф G называется (r, d, c)-экспандером, если:

- степени всех вершин из множества Y не превосходит более d,

- для любого множества $A \subseteq Y, |A| \leq r$ выполняется $\Gamma(A) \geq c|A|$.

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

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

Лемма 1.2 ([1, Лемма 1])**.** Каждый (r, d, c)-экспандер является граничным $(r, d, 2c - d)$-экспандером.

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 $\delta(A) \geq (2c - d)|A|$. □

Нам понадобятся граничные экспандеры; для этого достаточно иметь экспандеры с константой $c > d/2$.

Существуют также и явные конструкции таких экспандеров.

Лемма 1.3 ([6])**.** Для каждой константы $\epsilon > 0$ существует такая константа d, что за полиномиальное от n время можно построить (r, d, c)-экспандер, в котором $c = (1 - \epsilon)d$, $r = \Omega(n/d)$.

Следствие 1.1. Такой граф будет граничным $(\Omega(n/d), d, (1 - 2\epsilon)d)$-экспандером.

1.3. Функция Голдрейха

Голдрейх в [7] построил функцию $f : \{0,1\}^n \to \{0,1\}^n$, заданную с помощью графа G и предиката $P : \{0,1\}^d \to \{0,1\}$. Каждая строка из $\{0,1\}^n$ задает некоторое значение на множестве $X = \{x_1, x_2, \ldots, x_n\}$, значение $(f(x))_j$ (j-й символ строки $f(x)$) рассчитывается следующим образом: в вершину y_j входят ребра из вершин $x_{j_1}, x_{j_2}, \ldots, x_{j_d}$, тогда $(f(x))_j = P(x_{j_1}, x_{j_2}, \ldots, x_{j_d})$. Мы рассматриваем линейный предикат, т.е. $P(x_1, \ldots, x_d) = x_1 \oplus x_2 \ldots \oplus x_d$.

Опишем, как мы кодируем задачу обращения функции Голдрейха в виде задачи выполнимости булевой формулы.

Пусть $g : \{0,1\}^\ell \to \{0,1\}$. Канонической записью g в КНФ мы называем такую: для каждого набора $c \in \{0,1\}^\ell$, для которого $g(c) = 0$, пишется дизъюнкт $x_1^{c_1} \vee x_2^{c_2} \vee \cdots \vee x_\ell^{c_\ell}$, где $x_i^0 = x_i$, а $x_i^1 = \neg x_i$.

Пусть f — это функция Голдрейха, построенная по линейному предикату P и графу G. Уравнение $f(x) = b$ мы представляем следующим образом: для каждой вершины $y_j \in Y$, которая соединена с вершинами $x_{j1}, x_{j2}, \ldots, x_{jd}$, мы записываем каноническую формулу в КНФ от переменных $x_{j1}, x_{j2}, \ldots, x_{jd}$, которая кодирует равенство $b_j = P(x_{j1}, x_{j2}, \ldots, x_{jd})$. Полученную формулу будем обозначать $\Phi_{f(x)=b}$. Часть формулы, соответствующую вершинам из множества $A \subseteq Y$, будем обозначать $\Phi^A_{f(x)=b}$.

Заметим, что близорукий алгоритм, не видя знаков отрицаний, не отличит равенство $P(x) = 0$ от $P(x) = 1$ (без знаков отрицаний формулы выглядят одинаково, а количество вхождений с отрицанием равно количеству вхождений без отрицания). Также нетрудно видеть, что после подстановки значения любой переменной каноническая форма КНФ переходит в каноническую форму.

Глава 2. Граф зависимостей

Для доказательства основного результата нам потребуется граф со следующими свойствами:

- свойства экспандера;
- полный ранг матрицы смежности;
- ограниченная степень левой доли.

2.1. Явный экспандер

Лемма 2.1 ([6])**.** Для любого $\epsilon > 0$ существует такое k, что существует явная конструкция $(r, d, (1-\epsilon)d)$-экспандера с размерами долей m, n, где $r = \frac{m}{kd}$, $d = polylog(\frac{n}{m})$.

Следствие 2.1. Существует явная конструкция $(\frac{n}{kd}, d, 0.95d)$-экспандера с размерами долей $n, 2n$, где k, d — константы.

Доказательство. Следует из леммы 2.1 при $\epsilon = 0.05$, $\frac{n}{m} = 2$. □

Опишем алгоритм, который модифицирует граф так, чтобы он обладал искомыми свойствами.

Алгоритм 2.1. Вход: n, d.

1. Построить двудольный экспандер (X, Y, E) (E — множество ребер) из следствия 2.1.
2. Выкинуть из графа множество вершин $I \subseteq X$, степень которых больше $20kd$.
3. Выкинуть из графа множество вершин $Z \subseteq Y$, которое состоит из всех вершин, у которых число ведущих в I ребер, больше $\frac{d}{5}$.
4. Если $|Y| > |X|$, то выкинуть произвольные элементы из Y так, чтобы $|Y| = |X|$.

5. Выдать граф.

Лемма 2.2. Алгоритм 2.1 выдает $(\frac{n}{kd}, d, \frac{1}{2}d)$ граничный экспандер со степенью вершин левой доли, ограниченной $20kd$. При этом размер каждой доли равен n.

Доказательство. На первом шаге алгоритма текущий граф является $(\frac{n}{kd}, d, 0.95d)$-экспандером.

Оценим размер множества I. Всего ребер $2nd$, следовательно: $|I| \leq \frac{2nd}{20kd} = \frac{n}{10k}$.

Теперь докажем, что размер множества Z менее $\frac{n}{kd}$. От противного, пусть $|Z| \geq \frac{n}{kn}$. Возьмем $Z' \in Z$, для которого выполнено равенство $|Z'| = \frac{n}{kd}$. Из свойства экспандера следует, что $|\Gamma(Z')| \geq 0.95\frac{n}{k}$. Не менее чем $\frac{1}{5}|Z'|d$ ведут в множество I, следовательно, $|\Gamma(Z')| \leq |I| + \frac{4}{5}d|Z'| = \frac{n}{10k} + \frac{4}{5}d\frac{n}{kd} = 0.9\frac{n}{k} < 0.95\frac{n}{k}$. Получаем противоречие.

Таким образом, после третьего шага в Y останутся вершины, которые были связаны с X не более чем $\frac{d}{5}$ ребрами, т.е. степень расширения экспандера уменьшится не более чем на $\frac{d}{5}$. Получится $(\frac{n}{kd}, d, 0.75d)$-экспандер. Из леммы 1.2 получаем требуемое утверждение. □

2.2. Биективная функция Голдрейха

По лемме 2.2 мы научились строить граничный экспандер с ограниченной степенью левой доли. Однако, ранг матрицы смежности такого графа не обязательно полный. Добавим ребра в граф так, чтобы степень любой из долей увеличилась не более чем на 1.

Пусть G_1 — двудольный граф, степени, которого ограничены d_X и d_Y соответственно, а G_2 — граф, степени которого ограничены d'_X и d'_Y. Определим граф $G_1 + G_2$, как граф, степени которого ограничены $d_X + d'_X$ и $d_Y + d'_Y$, списки соседей для каждой вершины множества Y получаются в результате добавления списка из графа G_2 в конец списка из графа G_1.

Матрица смежности графа $G_1 + G_2$ равняется сумме матриц смежностей для графов G_1 и G_2.

Утверждение 2.1. Если граф G является (r, d, c)-экспандером, а степени G' ограничены d', то $G + G'$ является $(r, d + d', c)$-экспандером.

Теорема 2.1. По графу G можно за полиномиальное от n время построить граф T со степенями, ограниченными 2, такой, что ранг матрицы смежности $G + T$ равен n.

Доказательство. Сначала докажем вспомогательную лемму.

Лемма 2.3. Пусть $a = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_2^n$ и $\sum\limits_i \alpha_i = 0$. Тогда все векторы $b_i = (\alpha_1, \ldots, \alpha_{i-1}, \alpha_i + 1, \alpha_{i+1}, \ldots, \alpha_n)$ линейно независимы. Если же $\sum\limits_i \alpha_i = 1$, то ровно $n - 1$ вектор среди b_i линейно независим.

Доказательство. Докажем от противного. Пусть линейная зависимость есть. Тогда существует такая последовательность $i_1, \ldots, i_k$, что $b_{i_1} + \ldots + b_{i_k} = 0$. Пусть $c_i = b_i - a$. Заметим, что любая нетривиальная сумма c_i не равна нулю.

Пусть $k = 2\ell$. Тогда $b_{i_1} + \ldots + b_{i_k} = 2\ell a + c_{i_1} + \ldots + c_{i_k} = c_{i_1} + \ldots + c_{i_k} \neq 0$.

Пусть $k = 2\ell + 1$. Если $\forall i \sum\limits_j (b_i)_j \neq 0$, то $b_{i_1} + \ldots + b_{i_k} \neq 0$, иначе существует ровно одна последовательность $i_1, \ldots, i_k$, для которой $a = c_{i_1} + c_{i_2} + \ldots + c_{i_k}$. □

Пусть $T = T_1 + T_2$, и степени вершин графов T_1 и T_2 были ограничены 1.

Опишем построение графа T_1. Начнем с пустого множества ребер. Если вершина v из правой доли графа G имеет нечетную степень, то добавим в T ребро исходящее из соответствующей вершины (в графе $G + T$ степень у вершины v станет четная), в вершину из левой доли с нулевой степенью (такая вершина всегда найдется, т.к добавляется не более n ребер).

Теперь перейдем к графу T_2. Начнем с пустого графа и будем добавлять ребра по одному. Выделим линейно зависимые строки в матрице графа $G + T_1$. Будем добавлять к ним по очереди единицу так, чтобы ранг $G + T_1 + T_2$ строго увеличивался на каждом шаге. Это всегда можно сделать по лемме 2.3 (роль вектора a играет i-я строка матрицы смежности графа $G + T_1$).

Осталось показать, что степень левой доли графа T_2 ограничена единицей. Но действительно, пусть ранг матрица графа G равен $n - k$. Тогда в матрице графа T_2 ровно k единиц (не более одной в каждой строке), следовательно, если две единицы попали в один столбец, то ранг матрицы смежности графа T не более $k - 1$. Но $rank(G + T) \leq rank(G) + rank(T) \leq (n - k) + (k - 1) = n - 1$, и мы пришли к противоречию с тем, что ранг полный. □

Следствие 2.2. Если граф G является (r, d, c)-экспандером со степенью левой доли, ограниченной kd, то построенный в теореме граф $G + T$ будет $(r, d + 2, c)$-экспандером со степенью левой доли, ограниченной $kd + 2$, а функция Голдрейха f, построенная по $G + T$ и линейному предикату, будет биективной.

Глава 3. Умный близорукий алгоритм

3.1. Замыкание

Пусть граф G — граничный (r, d, c)-экспандер. Предикат $P(x_1, \ldots, x_d) = x_1 + \cdots + x_d$; функция Голдрейха f задана графом G и предикатом P. Будем считать, что $c > 4$.

Определение 3.1. Пусть $J \subseteq X$. Замыканием множества J будем называть множество вершин $I \subseteq Y$, удовлетворяющее следующим свойствам:

- $|I| \leq r$
- $\delta(I) \setminus J = \emptyset$
- $\forall I' \subseteq Y,\ |I'| \leq r,\ |I'| > |I| \Rightarrow \delta(I') \setminus J \neq \emptyset$

Лексикографически первое замыкание множества J будем обозначать $Cl(J)$.

Лемма 3.1 ([1])**.** Пусть I — замыкание J, если $|J| < \frac{r}{2}$, то $|I| \leq \frac{|J|}{c}$.

Доказательство. От противного: пусть $|I| > \frac{|J|}{c}$, но так как $|I| \leq r$, следовательно $\delta(I) \geq c|I| > |J|$. Получаем, что $\delta(I) \setminus J \neq \emptyset$, противоречие. □

Следствие 3.1. Пусть $J \subseteq X, |J| < \frac{r}{2}$. Тогда замыкание J определяется единственным образом.

Доказательство. Пусть I, I' — замыкания J, и $I \neq I'$. По лемме 3.1 $|I \cup I'| \leq 2\frac{|J|}{c} \leq |J| \leq \frac{r}{2}$. Следовательно, $I \cup I'$ также является замыканием J, но $|I \cup I'| > |I|$ — противоречие. □

Следствие 3.2. Пусть $J \subseteq J^*, |J^*| < \frac{r}{2}$. Тогда $Cl(J) \in Cl(J^*)$.

Доказательство. По лемме 3.1, $|Cl(J) \cup Cl(J^*)| \leq 2\frac{|J|}{c} \leq \frac{r}{2}$. Следовательно, $|Cl(J) \cup Cl(J^*)| = |Cl(J^*)|$, и $Cl(J) \cup Cl(J^*) = Cl(J^*)$. □

Определение 3.2. Пусть $f : \{0,1\}^n \to \{0,1\}^n$ — функция Голдрейха, построенная по графу G и предикату P, $b \in \{0,1\}$. Частичная подстановка ρ называется локально корректной для уравнения $f(x) = b$, если существует такая строка $z \in \{0,1\}^n$, согласованная с ρ и $I = Cl(\mathrm{Vars}(\rho))$, что выполняется равенство $f(z)|_I = b|_I$.

Лемма 3.2 ([1])**.** Если частичная подстановка $\rho, |\rho| < \frac{r}{2}$ локально корректна для уравнения $f(x) = b$, то для любого $Z \subseteq X$, $|Z| \leq \frac{r}{2}$, существует такая строка $z \in \{0,1\}^n$, согласованная с ρ, что выполняется равенство $f(z)|_Z = b|_Z$.

Доказательство. От противного. Возьмем минимальное по размеру $Z \subseteq Y$ такое, что $|Z| \leq \frac{r}{2}$ и при всех z, согласованных с ρ, выполняется $f(z)|_Z \neq b|_Z$. Пусть $I = Cl(\mathrm{Vars}(\rho))$ из определения 3.2. Поскольку ρ локально корректна, то $Z \setminus I \neq \emptyset$.

Поскольку $|\rho| < \frac{r}{2}$, следовательно, по лемме 3.1 $|I| \leq \frac{r}{2}$. Т.к. $|Z| \leq \frac{r}{2}$, то из определения замыкания следует, что существует такой $y \in Z \setminus I$, что существует граничная переменная множества Z (не соединенная с замыканием и не являющаяся подставленной), которая соединена с y (иначе $Z \cup I$ было бы замыканием).

Из минимальности Z следует, что существует такой $z \in \{0,1\}^n$ согласованный с ρ, что $f(z)|_{Z \setminus y} = b|_{Z \setminus y}$. Но путем инвертирования бита в z, соответствующего связанной с y граничной переменной, можно выполнить уравнение соответствующее y. Таким образом, существует такой z, согласованный с ρ, что $f(z')|_Z = b|_Z$. Противоречие. □

Лемма 3.3. Если G — (r, d, c)-экспандер с n вершинами в каждой доле, тогда, если $J \subseteq X, |J| \leq \frac{r}{2}$, то существует такая константа $m = m(\frac{c}{d})$, что выполняется включение $Cl(J) \subseteq \Gamma^{m \log(|J|)}(J)$, где Γ^i обозначает множество вершин, которые удалены от множества J не более, чем на i.

Доказательство. Разобьем замыкание на уровни. Каждый уровень отве-

чает соответствующему расстоянию до J. Занумеруем уровни от самого удаленного. Таким образом: $Cl(J) = A_1 \cup A_2 \cdots \cup A_k$, $a_i = |A_i|$, $S_i = \bigcup\limits_{j=1}^{i} A_j$, $s_i = |S_i|$, где A_i — множество вершин отстоящих от множества J на расстоянии $2(k-i)+1$

Заметим, что из определения замыкания следует, что $\forall i\ s_i \leq r$. $\delta(Cl(J)) \in J$, следовательно граница множества $S_i,\ (1 \leq i < k)$ соединена с другими элементами из $Cl(J)$, но по построению множеств S_i выполнено $\delta(S_{i-1}) \subseteq \Gamma(A_i)$. Запишем неравенство на число ребер исходящих из множества A_i: $da_{i+1} \geq cs_i$. Отсюда получаем, что $s_{i+1} \geq (1 + \frac{c}{d})s_i$. Из леммы 3.1 следует искомое утверждение. □

Следствие 3.3. Если G — (r, d, c)-экспандер, степень левой доли которого ограничена $kd,\ k > 1$, то найдется такое $\delta > 0$, что если $J \subseteq X, |J| \leq n^\delta$, то найдется такое i, что $Cl(J) \in \Gamma^i(J)$ и $|\Gamma^i(J)| \leq \frac{r}{2}$.

Доказательство. Оценим размер $\Gamma^i(J)$. Из каждой вершины путей длины i не более $(kd)^i$ (из ограничения на степень вершин), следовательно, $|\Gamma^i(J)| \leq |J|(kd)^i$. Пусть $J \leq n^\delta$. Тогда $|\Gamma^{m\log(|J|)}| \leq n^\delta (kd)^{m\delta\log(n)} = n^{\delta(1+m\log(kd))}$ Применяя лемму 3.3, получаем, что $Cl(J) \in \Gamma^{m\log(|J|)}$, значит, нужное δ найдется. □

3.2. Умный близорукий алгоритм

Опишем умный близорукий алгоритм, который является надстройкой над близоруким DPLL алгоритмом. Он получает на вход формулы, кодирующие задачу обращения функции Голдрейха в КНФ. Умный близорукий алгоритм не будет делать некоторые подстановки, которые заведомо приводят к невыполнимой формуле, он снабжен дополнительной процедурой $\mathbf{D}$, которая будет отсекать подобные ветви. Мы рассматриваем только линейный предикат.

Опишем умный близорукий алгоритм формально.

Алгоритм 3.1. Вход: формула ϕ в КНФ, частичная подстановка ρ

1. Запустить $\mathbf{C}(\varphi)$, если получился 0, то выдать «формула невыполнима» и закончить работу.
2. 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».
3. $x_j := \mathbf{A}(\varphi)$.
4. $c := \mathbf{B}(\varphi, x_j)$.
5. Случай $|\rho| \leq \frac{r}{2}$. Запустить алгоритм рекурсивно на $(\varphi[x_j := c], \rho \cup \{x_j := c\})$; если алгоритм выдал подстановку, то выдать ее, в противном случае рекурсивно запустить на $(\varphi[x_j := 1 - c], \rho \cup \{x_j := 1 - c\})$; если рекурсивный вызов выдал подстановку, то выдать ее, иначе выдать «формула невыполнима».
6. Случай $|\rho| \leq \frac{r}{2}$. Если $\mathbf{D}(\varphi, \rho \cup \{x_j := c\}) = 1$, то запустить алгоритм рекурсивно на $(\varphi[x_j := c], \rho \cup \{x_j := c\})$; если алгоритм выдал подстановку, то выдать ее, в противном случае если $\mathbf{D}(\varphi, \rho \cup \{x_j := 1 - c\}) = 1$, то рекурсивно запустить на $(\varphi[x_j := 1 - c], \rho \cup \{x_j := 1 - c\})$; если рекурсивный вызов выдал подстановку, то выдать ее, иначе выдать «формула невыполнима».

Эвристики $\mathbf{A}, \mathbf{C}$ — близоруки. $\mathbf{D}(\varphi, \rho)$ видит биты $Cl(\rho)$ и возвращает 1 тогда и только тогда, когда переданная ей подстановка локально корректной. Пусть Z — множество битов, открытых эвристиками $\mathbf{A}$ и $\mathbf{C}$. Множеством открытых битов умного близорукого алгоритма будем называть $Z \cup Cl(\rho)$.

Под деревом расщепления, которое соответствует умному близорукому алгоритму, будем понимать дерево расщеплений, созданное алгоритмом, в котором ответ эвристики C заменен на 1.

Замечание 3.1. Пусть v_1 — потомок v и $\rho_{v_1} \leq \frac{r}{2}$. Тогда множество от-

крытых битов в вершине v содержится во множестве открытых битов в вершине v_1, т.к. замыкания на всех шагах вложены друг в друга.

Лемма 3.4 ([1])**.** Если $|\rho| \leq \lfloor \frac{(c-1)r}{4dK} \rfloor$, то умный близорукий алгоритм откроет не более чем $\frac{r}{2}$ выходных битов.

Доказательство. Заметим, что $|\rho| < \frac{r}{2}$. Следовательно, по лемме 3.2 $Z_D = Cl(\rho)$. Количество открытых битов не превышает $K\frac{(c-1)r}{4dK} + |Cl(\rho)|$. По лемме 3.1 $|Cl(\rho)| \leq \frac{|\rho|}{c-1}$. При этом $|\rho| \leq \frac{(c-k-1)r}{4}$, и $K\frac{(c-1)r}{4dK} + |I| \leq \frac{(c-1)r}{4d} + \frac{r}{4} \leq \frac{r}{2}$ □

Следствие 3.4. Пусть умный близорукий алгоритм создает дерево расщепления T_Φ, тогда если $|\rho_v| \leq \lfloor \frac{(c-1)r}{4dK} \rfloor$, то у вершины v есть хотя бы один потомок.

Доказательство. Если $|\rho_v| \leq \lfloor \frac{(c-1)r}{4dK} \rfloor$, то количество открытых битов не превосходит $\frac{r}{2}$. По индукции можно доказать, что ρ_v является локально корректной. База индукции очевидна. Из леммы 3.2 следует, что хотя бы одно из двух значений для подставляемой переменной удовлетворяет всем открытым битам. □

3.3. Распределение

Определение 3.3. Будем говорить, что ансамбль распределений $\mathfrak{A} = \{\mathcal{D}_i\}_{i=1}^{\infty}$ полиномиально моделируемый, если существует такой полиномиальный по времени вероятностный алгоритм $\mathcal{A}$, что для любого x верно равенство $\Pr[\mathcal{A}(1^n) = x] = \mathcal{D}_n(x)$. Множество всех полиномиально моделируемых распределений обозначим $PSamp$.

Докажем следующее утверждение.

Теорема 3.1. Пусть **A** — детерминированная близорукая эвристика, работающая полиномиальное от размера формулы время. Тогда существует семейство $(\Phi_n, \mathcal{D}_n)$, где Φ_n — невыполнимая формула от n переменных, кото-

рую можно построить за полиномиальное время, $\mathfrak{A} = \{\mathcal{D}_i\}_{i=1}^{\infty} \in PSamp$ — ансамбль распределений на выполнимых формулах, и $\delta > 0$ такие, что для любой эвристики $\mathbf{B}$ и любой близорукой эвристики $\mathbf{C}$, если DPLL алгоритм параметризованный эвристиками $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, работает корректно с вероятностью $p > 1 - \epsilon$ на распределении $\mathcal{D}_n$, то математическое ожидание времени работы алгоритма на формуле Φ_n не менее $(1 - \epsilon)2^{\Omega(min(n^{\delta}, \frac{n}{K}))}$.

Для построения данного распределения будем рассматривать только формулы, полученные при обращении функции Голдрейха (разд. 1.3), параметризованной линейным предикатом и графом G. От G потребуем, чтобы он был (r, c, d)-граничным экспандером с полным рангом и при этом $c > 2$. Также необходимо, чтобы степень левой доли была ограничена kd; в частности, подходит конструкция, описанная в разделе 2. Для упрощения подсчетов будем считать, что $c = \frac{d}{5}$, $d > 15$.

Определение 3.4. Для $f(x) = b$ назовем x корректным продолжением частичной подстановки ρ, если $x \sim \rho$ и $f(x)|_Z = b|_Z$, где Z — множество открытых битов алгоритма.

Основная идея построения распределения — рассмотреть некоторую невыполнимую формулу Φ и построить выполнимые формулы на основе корректных различных частичных подстановок. Для выбора таких подстановок возьмем такой уровень вершин $N = min(n^{\delta}, \Omega(\frac{n}{K}))$, что дойдя до этого уровня алгоритм еще не сможет определить, что формула невыполнима; δ здесь берется из следствия 3.3. Здесь и далее будем полагать $N = min(n^{\delta}, \frac{(c-1)r}{8Kd})$.

Покажем, что вершины уровня N лежат не слишком глубоко в дереве, т. е., дойдя до любой из них, алгоритм еще не сможет понять, что формула не выполнима. Для этого докажем вспомогательные леммы.

Замечание 3.2. Рассмотрим систему линейных уравнений вида: $x_{i1} + x_{i2} + \ldots + x_{ik} = b_i$. Тогда, если у данной системы есть решения, то

размерность пространства решений не зависит от значений b_i.

Замечание 3.3. Будем говорить, что уравнение $ax = b$ линейно зависимо с системой уравнений $Ax = c$, если ранг A равен рангу A с приписанной строкой a.

Лемма 3.5. Пусть $b \in \{0,1\}^n$, ρ — частичная подстановка, а $Z \subset Y$ — множество открытых битов, $|Z| < \frac{r}{2}$. Если уравнение $x_i = c$ линейно зависимо с системой $f(x)|_z = b|_z \cup x \sim \rho$, то оно линейно зависимо с системой $f(x)_I = b|_I \cup x \sim \rho$, где $I = Cl(\mathrm{Vars}(\rho))$.

Доказательство. Пусть уравнение линейно зависимо. Тогда существует такое c, что система $f(x)|_z = b|_z \cup x \sim \rho$ не имеет решений. Но из леммы 3.2 следует, что подстановка не является локально корректной, следовательно, уравнение $x_i = c$ так же линейно зависимо с уравнением $f(x)_I = b|_I \cup x \sim \rho$. □

Лемма 3.6. Для любого $b \in \{0,1\}^n$, пусть $v_0 \in T_{\Phi_{f(x)=b}}$ — точка расщепления, $l(v_0) = N$, тогда $|\rho_{v_0}| \leq \lceil \frac{c-1}{c-2} N \rceil$.

Доказательство. Рассмотрим вершину v_0 находящуюся на расстоянии $\lceil \frac{c-1}{c-2} N \rceil + 1$. Пусть $I_o = Cl(\mathrm{Vars}(\rho_{v_0}))$.

У вершины v один потомок тогда и только тогда, когда подстановка текущей переменной является линейно зависимой, т.е. уравнение $x_i = c$ линейно зависимо с $f(x)|_{Z_v} = b|_{z_v} \cup x \sim \rho$. По лемме 3.5 $x_i = c$ линейно зависимо с $f(x)|_I = b|_I \cup x \sim \rho$, где $I = Cl(\mathrm{Vars}(\rho))$. Но ранг системы $f(x)|_I = b|_I \cup x \sim \rho$ не менее, $|\rho|$ (т.к. все уравнения $x \sim \rho$ линейно независимы), следовательно не более $|I|$ уравнений в системе $f(x)|_I = b|_I \cup x \sim \rho$ могут быть линейно зависимыми. Следовательно, число вершин с одним потомком на пути к v_0 не более, чем I_{v_0}, что по лемме 3.1 не превосходит $\frac{|\rho_{v_0}|}{c-1} \leq \frac{N}{c-2} + \frac{2}{c-1}$, откуда получаем, что уровень вершины v_0 не менее $\frac{c-1}{c-2} N + 1 - \frac{N}{c-2} - \frac{2}{c-1} > N$. □

Следствие 3.5. Если $z \leq N$, то на уровне z ровно 2^z вершин.

Доказательство. Из леммы 3.6 следует, что расстояние от корня до любой вершины уровня N будет не более $\lfloor \frac{(c-1)r}{4dK} \rfloor$. Следовательно, по следствию 3.4 ни одна ветвь дерева не будет обрезана раньше уровня N. Из того, что у каждой точки расщепления ровно 2 потомка, следует указанное утверждение. □

Теперь перейдем к построению пары (формула, распределение).

Параметризуем близорукий алгоритм $\mathcal{A}$ заданной эвристикой $\mathbf{A}$ и эвристиками $\mathbf{B}$, $\mathbf{C}$, всегда выдающими единицу. Зафиксируем граф G с рангом n. Зафиксируем точку $b \in \{0,1\}^n$ и частичную подстановку ρ_0, для которой $\{x \mid x \sim \rho_0, f(x) = b\} = \emptyset, \mathrm{Vars}(\rho_0) = \{x_1\}$.

Рассмотрим формулу $\Phi = \Phi_{f(x)=b}$ и дерево T_{Φ,ρ_0}, построенное алгоритмом расщепления с входом Φ, ρ_0. Зададим алгоритм для получения распределения $\mathfrak{A}$ на выполнимых формулах.

Алгоритм 3.2.

1. Равновероятно выбрать вершину v уровня N в дереве расщеплений T_{Φ,ρ_0}. Для этого будем двигаться от корня к потомкам, если текущая вершина — точка расщепления (проверить является ли вершина точкой расщепления можно за полиномиальное время по лемме 3.7), то выбирать случайный бит и идти в соответствующего потомка. Остановиться на вершине уровня N.
2. Равновероятно выбрать корректное продолжение a частичной подстановки ρ_v. В вершине v уравнениями $x \sim \rho_v$ и $f(x)|_{Z_v} = b|_{Z_v}$ задается линейное многообразие. Найти базис линейного подпространства, которое порождает данное многообразие, с помощью процедуры Грамма-Шмидта получим ортогональный базис. Каждую координату выбирается случайно, по координатам нетрудно получить точку из многообразия.
3. Выдать формулу $\Phi_{f(x)=f(a)}$

Замечание 3.4. Если эвристика $\mathbf{A}$ алгоритма $\mathcal{A}$ полиномиальна по време-

ни относительно размера формулы, то алгоритм 3.2 также полиномиален.

Лемма 3.7. Пусть $J \subseteq X$. Тогда, если $|J| \leq n^{\delta}$, то существует полиномиальный алгоритм, который определяет, является ли текущая вершина точкой расщепления.

Доказательство. На вход алгоритм получает формулу и частичную подстановку. Вершина не является точкой расщепления тогда и только тогда, когда текущая подстановка линейно зависима с $Cl(\mathrm{Vars}(\rho)) \cup x \sim \rho$. Но, если текущая подстановка линейно зависима с $F \cup x \sim \rho$, где $F \subseteq Y$, $|F| \leq \frac{r}{2}, Cl(\mathrm{Vars}(\rho)) \subseteq F$, то по лемме 3.5 текущая подстановка линейно зависима и с $Cl(\mathrm{Vars}(\rho)) \cup x \sim \rho$. В качестве множества $|F|$ возьмем $\Gamma^{O(log(n))}(\rho_v)$ (см. лемму 3.3), а для построения данного множества существует полиномиальный алгоритм. □

Теперь докажем теорему 3.1.

Доказательство теоремы 3.1. Зафиксируем эвристики $\mathbf{B}$, $\mathbf{C}$. Выберем любое $b \in \{0,1\}^n$ и рассмотрим формулу $\Phi = \Phi_{f(x)=b, x\sim\rho_0}$, где ρ_0 определялось выше.

Рассмотрим работу алгоритма $\mathcal{A}$ на невыполнимой формуле, а также случайные величины $X_{ij} \in \{0,1\}$, X_{ij} равно 1, если алгоритм $\mathcal{A}$ посетит вершину с номером j на уровне i. Тогда $t_n = \sum\limits_{i,j} X_{ij}$, где t_n — время работы алгоритма $\mathcal{A}$. Оценим математическое ожидание t_n. $E[t_n] = \sum\limits_{i,j} E[X_{ij}] \geq \sum\limits_{j} E[X_{Nj}]$. Заметим, что $E[X_{Nj}] = p_j$, где p_j — вероятность того, что алгоритм $\mathcal{A}$ посетит вершину с номером j на уровне N. Осталось оценить сумму $\sum\limits_{j} p_j$.

Пусть $p = \Pr\limits_{\phi\leftarrow\mathcal{D}_n}[\mathcal{A}(\phi) = 1]$ — вероятность корректной работы алгоритма на распределении $\mathcal{D}_n$, построенному по алгоритму 3.2.

$p \leq \sum\limits_{i} \Pr[B_i = 1] \Pr\limits_{\phi\leftarrow\mathcal{D}_n, r}[\mathcal{A}(\phi) = 1 \mid B_i = 1]$, где $B_i = 1$ тогда и только тогда, когда выполнимая формула ϕ была получена из корректного продолжения локально корректной подстановки вершины уровня N с номером

i (в дереве расщепления полученном в алгоритме 3.2), а r — случайные биты, использованные эвристиками.

Из пункта 2 алгоритма 3.2, а также из следствия 3.5 следует, что $\Pr[B_i = 1] = \frac{1}{2^N}$.

Пусть $B_i = 1$. Рассмотрим пусть по дереву к выполняющему набору формулы ϕ. Остальные пути нам не интересны, т. к. у каждой формулы ровно один выполняющий набор, а значит на вероятность корректной работы алгоритма на формуле ϕ они никак не влияют. Заметим, что начальный отрезок этого пути повторяет начальный отрезок одного из путей в дереве расщепления на невыполнимой формуле, а сам путь повторяет один из путей в дереве расщепления построенном в алгоритме 3.2 (т.к. эвристика **A** детерминирована). Это следует из построения формулы ϕ, т.к. данная формула является корректным продолжением частичной подстановки и согласуются с открытыми входными битами. Следовательно $\Pr\limits_{\phi \leftarrow \mathcal{D}_n, r}[\mathcal{A}(\phi) = 1 \mid B_i = 1] \leq p_i$. Таким образом $p \leq \sum\limits_{i=1}^{2^N} \frac{1}{2^N} p_i$. Если $p > 1 - \epsilon$, то $\sum\limits_{i=1}^{2^N} p_i \geq (1-\epsilon)2^N$, а значит $E(t_n) \geq (1-\epsilon)2^{\Omega(min(n^\delta, \frac{n}{K}))}$. □

Теперь перейдем к построению единого распределения для всех близоруких DPLL алгоритмов. Если выбирать с некоторой вероятностью одно их распределений из теоремы 3.1, то получившееся распределение может перестать быть полиномиально моделируемым.

Теорема 3.2. Существуют семейство Φ_n, ансамбль распределений на выполнимых формулах $\mathfrak{A} = \{\mathcal{D}_i\}_{i=1}^{\infty} \in PSamp$, где Φ_n — невыполнимая формула от n переменных, и $\delta > 0$, для которых для любого близорукого DPLL алгоритма с полиномиальной детерминированной эвристикой **A** существуют такие ϵ и n_0, что для любого $n > n_0$, если алгоритм верно работает на D_n с вероятностью не менее $1 - \epsilon$, то алгоритм работает как минимум $2^{min(n^\delta, \Omega(\frac{n}{K}))-1}$ шагов на любой формуле из семейства формул Φ_n.

Доказательство. Пронумеруем все пары (M, t), где M — близорукий DPLL алгоритм с детерминированной эвристикой $\mathbf{A}$, n^t — оценка на время работы эвристики $\mathbf{A}$. За $(\Phi_n, \mathcal{D}_{in})$ обозначим семейство, построенное для алгоритма M из пары (M_i, t_i) алгоритмом 3.2. Отметим, что формула Φ_n для всех алгоритмов одинакова.

Построим распределение $\mathcal{D}_n$ следующим образом:

1. Выбрать $k \in \{1 \dots n\}$: $k = i$ с вероятностью $\frac{1}{2^{i+1}}$, где $i < n$; $k = n$ с вероятностью $\frac{1}{2^n}$.
2. Запустить алгоритм 3.2 для алгоритма M_k и формулы $\Phi_{\sqrt[t_k]{n}}$ на n^{t_k+1} шаг.
3. Если алгоритм остановился, то выдать формулу ϕ согласно распределению $\mathcal{D}_{k\,\sqrt[t_k]{n}}$, иначе выдать произвольную выполнимую формулу, кодирующую задачу обращения функции Голдрейха.

Для любого близорукого алгоритма M с детерминированной эвристикой $\mathbf{A}$, работающей полиномиальное время, найдется такая пара (M, t), что n^t — верхняя оценка на время работы эвристики $\mathbf{A}$. Номером алгоритма будем называть номер пары с таким минимальным t.

Рассмотрим алгоритм M с номером i, зафиксируем $\epsilon = \frac{1}{2^{i+1}}$. Для любого $n > i$ формула ϕ выбирается согласно распределению $\mathcal{D}_{i\,\sqrt[t]{n}}$ с вероятностью $\frac{1}{2^{i+1}}$. Таким образом, если алгоритм M работает правильно на распределении $\mathcal{D}_n$ с вероятностью не менее $1 - \frac{1}{2^{i+1}}$, то алгоритм M работает правильно на распределении $\mathcal{D}_{\sqrt[t]{n}}$ не менее $\frac{1}{2}$. Следовательно, по теореме 3.1 алгоритм M_i работает не менее $\frac{1}{2}2^{min(\sqrt[t]{n^\delta}, \Omega(\frac{\sqrt[t]{n}}{K}))}$ на формуле $\Phi_{\sqrt[t]{n}}$, а значит на формуле Φ_n алгоритм работает не менее чем $\frac{1}{2}2^{min(n^\delta, \Omega(\frac{n}{K}))}$ шагов. $\square$

Список литературы

[1] 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.

[2] 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.

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

[4] D. Itsykson. Lower bound on average-case complexity of inversion of Goldreich function by drunken backtracking algorithms. In *Proceedings of III International Computer Science Symposium in Russia*, volume 6072 of *Lecture Notes in Computer Science*, pages 204–215. Springer, 2010.

[5] D. Itsykson and D. Sokolov. The complexity of inversion of explicit Goldreich's function by DPLL algorithms. In *Proceedings of CSR 2011*, pages 134–147. Springer, 2011.

[6] M. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson. Randomness conductors and constant-degree expansion beyond the degree/2 barrier. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 659–668, 2002.

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