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

На правах рукописи

Диссертация допущена к защите
Зав. кафедрой

_____

“ ”_____2012 г.

ДИССЕРТАЦИЯ

НА СОИСКАНИЕ УЧЕНОЙ СТЕПЕНИ

МАГИСТРА

Тема: Новые верхние оценки для задачи максимальной выполнимости

Направление: 010600.68 – Прикладные математика и физика

Магистерская программа: “Математические и информационные технологии”

Выполнил студент: И. А. Близнец

(подпись)

Руководитель:
к.ф.-м.н. А. С. Куликов

(подпись)

Рецензент:
к.ф.-м.н. Д. М. Ицыксон

(подпись)

Санкт-Петербург
2012 г.

Реферат

Работа выполнена на 34 страницах в 5 главах, использовано 19 источников.

Время работы известных алгоритмов для NP-трудных задач попрежнему экспоненциально. В предположении P≠NP полиномиальных алгоритмов для таких задач не существует. Поэтому интересным является вопрос, насколько может быть уменьшена константа в основании экспоненты времени работы алгоритмов для NP-трудных задача. В работе приведены два алгоритма, улучшающие результаты для следующих задач: параметризованная задача максимальной выполнимости и частный случай задачи максимальной выполнимости, где каждая переменная встречается не более трех раз. В первом случае время работы оценено относительного заданного параметра k, а во втором — относительно количества переменных. Алгоритм для MAX-SAT работает 1.358^k, а алгоритм для $(n, 3)$-MAX-SAT имеет время работы $2^{\frac{n}{3}}$ времени.

Содержание

Введение

Интерес к доказательству экспоненциальных верхних оценок для NP-трудных задач в последние несколько десятилетий остается на стабильно высоком уровне. Задача пропозициональной выполнимости является одной из наиболее известных NP-полных задач. Данной задаче посвящена международная конференция (The International Conference on Theory and Applications of Satisfiability Testing), проводящаяся уже более десяти лет, а также научный журнал (Journal of Satisfiability, Boolean Modeling and Computation). Поскольку задача имеет огромное практическое значение, важное место в исследовании задачи выполнимости занимает разработка программ, решающих SAT (подобные программы называются SAT-солверами). Ежегодно проводятся соревнования таких программ. Современные SAT-солверы способны быстро решать многие задачи, считавшиеся нерешаемыми несколько лет назад.

Важным оптимизационным обобщением задачи выполнимости является задача максимальной выполнимости. Это NP-трудная задача со следующей формулировкой: для данной булевой формулы в конъюнктивной нормальной форме определить наибольшее число одновременно выполнимых дизъюнктов (клозов). В параметризованной версии задачи MAX-SAT для данного натурального числа k надо определить, можно ли одновременно выполнить как минимум k дизъюнктов формулы. Легко видеть, что обе задачи являются NP-трудными, поскольку обобщают задачу выполнимости булевой формулы. Согласно теории об NP-полноте, такие задачи не могут быть решены за полиномиальное время в предположении P$\neq$NP.

Выше упомянутый факт о сложности этих задач не уменьшает потребности в решении подобных задач, ввиду их практической необходимости. Из-за тесной связи параматризованной задачи выполнимости с задачей выполнимости, параметризованный MAX-SAT имеет широкое применение в искусственном интеллекте, комбинаторной оптимизации, экспертных системах, в системах баз данных [1, 2, 3, 4, 5, 6, 7, 8]. Было разработано много методов для борьбы с NP-трудными задачами, среди них приближенные алгоритмы, эвристические алгоритмы, точные параметризованные алгоритмы.

Для задачи MAX-SAT существуют приближенные алгоритмы, на-

ходящие за полиномиальное время решение с некоторой гарантированной точностью [9]. В то же время известно, что в предположении P≠NP не существует полиномиального по времени алгоритма, находящего приближенное решение, сколь угодно близкое к оптимальному.

В этой работе мы сфокусированы на поиске точного решения для параметризованной задачи о выполнимости. Маловероятно получение алгоритма, отличного от экспоненциального. Но стоит заметить, что экспонента может быть значительна улучшена в сравнении с полным перебором. Многие NP-трудные задачи изучены с этой точки зрения. Примеры экспоненциальных алгоритмов для NP-трудных задач включают поиск независимого множества, 3-SAT, 3-раскраска. Тесно связанной областью с точными экспоненциальными алгоритмами является область fixed-parameter tractable задач. Данная область нашла большое практическое применение в искусственном интеллекте, базах данных, логике, вычилительной биологии. Известная задача, принадлежащая данной категории, это задача о вершинном покрытии. Другие примеры параметризованных задач включают в себя задачи о планарном доминирующем множестве, k-размерном сопоставлении.

Задача максимальной выполнимости и вместе с тем ее параметризованный случай играют важную роль в обоих этих подходах. Значительные усилия были приложены, чтобы улучшить верхнюю оценку в худшем случае для этих задач. Лучшая верхняя оценка 1.370^k для параметризованной задачи максимальной выполнимости была получена в 2002 году Ченем и Канжем [10]. Список лучших верхних оценок для параметризованной версии задачи о максимальной выполнимости приведен ниже в таблице. Здесь и далее временные оценки представлены с точностью до полиномиальных множителей.

Время	Авторы	Год
1.618^k	Mahajan, Raman [11]	1999
1.3995^k	Niedermeier, Rossmanith [12]	1999
1.3803^k	Bansal, Raman [13]	1999
1.3695^k	Chen, Kanj [10]	2002

Одним из наиболее хорошо изученных подходов к доказательству таких оценок является метод расщепления. Впервые данный метод был предложен в 1960-м году Дэвисом и Патнемом. Основная идея метода заключается в построении нескольких более простых задач, таких что, построив решение для каждого из них, можно за полиномиальное время най-

ти решение для первоначальной задачи. Для многих NP-полных задач лучшие результы получены именно методом расщепления. Простейший алгоритм расщепления для задачи выполнимости на формуле с N переменными имеет время работы 2^N. Первым улучшением данной оценки были работы Мониена, Шпекенмейера[14] и Данцина[15] для формул, длины дизъюнктов которых ограничены некоторой константой. Позднее было получено много оценок, улучшающих тривиальную для некоторых NP-полных подклассов задач выполнимости и максимальной выполнимости.

Типичный алгоритм расщепления сначала некоторым образом расщепляет формулу, после чего производит рекурсивные вызовы для формул меньшей сложности. Анализ такого алгоритма содержит разбор большого количества случаев, в каждом из которых показывается, что алгоритм всегда производит рекурсивные вызовы для формул, сложность которых меньше сложности исходной формулы на некоторую константу.

В работе будет предложен алгоритм со временем работы 1.358^k. Также в работе будет рассмотрен частный случай задачи о максимальной выполнимости $(n, 3)$-MAX-SAT. В этом частном случае, в отличие от общего, мы рассматриваем только те формулы, в которых каждая переменная встречается не более трех раз. Для $(n, 3)$-MAX-SAT представлено два алгоритма, время одного из них зависит от n, а другого от k

Результаты

1. Получен новый класс формул для которых задача максимальной выполнимости может быть решена за полиномиальное время.

2. Придумано правило, помогающее получить алгоритм для расщепления по переменным степени три. Данный алгоритм дает лучшее расщепления по сравнению с другими аналогами. Подобный метод помогает доказать новую оценку для параметризованной задачи выполнимости, возможно с помощью такого алгоритма можно также улучшить оценку для задачи максимальной выполнимости.

3. Улучшен алгоритм для параметризованной задачи максимальной выполнимости относительно праметра k.

4. Разработан алгоритм для $(n, 3)$-MAX-SAT относительно количества переменных, улучшающий ранее известные.

Глава 1. Предварительные сведения

Литералы и формулы Везде в работе n обозначает число переменных в формуле, m — число дизъюнктов, а k — число дизъюнктов, которое надо выполнить. Через MAX-SAT (F) обозначим максимальное количество одновременно выполнимых дизъюнктов. MAX-SAT $(F, k) = true$ тогда и только тогда когда MAX-SAT$(F) \geq k$. Отождествим константы ИСТИНА (true) и ЛОЖЬ (false) с 1 и 0, соответсвенно. Через $\#_F(l)$ обозначим число вхождений литерала l в формуле F. $F[l]$ обозначает формулу, полученную из F удалением всех литералов $\bar{l}$ и всех дизъюнктов, содержащих l. $F[x = y]$ будет обозначать формулу, где литералы x и $\bar{x}$ заменены на y и $\bar{y}$, соответственно. Мы говорим, что переменная имеет степень p, если она встречается в формуле ровно p раз. Также мы говорим, что переменная x имеет тип (a, b) если литерал x встречается a раз, а литерал $\bar{x}$ — b раз. Говорим, что литерал y — это сосед литерала x, если существует дизъюнкт, содержащий x и y одновременно. Переменная является $(k, 1)$-одиночкой, если ее положительный литерал встречается k раз, а отрицательный образует отдельный дизъюнкт. Аналогично переменная $(k, 1)-$общительная, но если у отрицательного литерала есть сосед. Клоз состоящий из одной переменной назовем единичным. Литерал l являтся чистым если литерал $\bar{l}$ не присутствует в формуле. Литерал y доминирует литерал x, если все дизъюнкты, содержащие литерал x, также содержат и литерал y. Два дизъюнкта называются комплементарными, если один из них есть отрицание другого. Знак "$\ldots$"обозначает остаток дизъюнкта. Так, к примеру, $x \vee \bar{y} \vee \ldots$ обозначает дизъюнкт, содержащий литералы $x, \bar{y}$ и, возможно, что-то еще.

Расщепление В задаче параметризованной максимальной выполнимости экземпляр задачи есть пара (F, k) и нам необходимо определить, можно ли выполнить не менее k дизъюнктов в формуле F.

Для $q > 1$ мы говорим, что сущесвует $(a_1, \ldots, a_q)$-расщепление, если мы можем найти формулы $F_1, F_2, \ldots, F_q$ такие, что ответ для первоначальной задачи может быть получен из ответов для следующих задач: $(F_1, k-a_1), (F_2, k-a_2), \ldots, (F_q, k-a_q)$. Легко видеть, что всегда существует $(F[l], k - \#_F(l)), (F[\bar{l}], k - \#_F(\bar{l}))$-расщепление, где l — литерал формулы. Хорошо известно, что если алгоритм на каждом шаге использует только расщепления из множества

$$(a_{1,1}, \ldots a_{1,q_1}), (a_{2,1}, \ldots a_{2,q_2}), \ldots (a_{t,q_t}, \ldots a_{t,q_t}),$$

где $a_{i,1} \leq a_{i,2} \leq \cdots \leq a_{i,q_i}$, для любого i от 1 до t, то время работы алгоритма не превосходит $O^*(c^k)$, где c — это наибольший положительный корень многочлена

$$p(X) = \prod_{j=1}^{t}(X^{a_{j,q_j}} - \sum_{i=1}^{q_j} X^{a_{j,q_j}-a_{j,i}}).$$

Для расщепления $(a_1, a_2 \ldots, a_q)$ через $\tau(a_1, a_2, \ldots, a_q)$ обозначим наибольший корень многочлена $X^{a_q} - (X^{a_q-a_1} + X^{a_q-a_2} + \cdots + X^{a_q-a_q})$. $\tau(a_1, a_2, \ldots a_q)$ называется числом расщепления.

Мы говорим, что расщепление $(b_1, b_2, \ldots, b_q)$ доминируется расщеплением $(a_1, a_2, \ldots a_q)$, если для всех i верно $a_i \geq b_i$.

Идея доказательства Расщепление по часто встречающейся переменной дает хорошее число расщепления. Поэтому главным препятсвием для алгоритмов с малой экспонентой являются переменные с небольшой степенью. Рассмотрим переменную x степени 3: $(x \vee A)(x \vee B)(\bar{x} \vee C)$, где A, B, C — это дизъюнкции литералов. Если A (или B) состоит из одного литерала, тогда заменим $(x \vee A)(x \vee B)(\bar{x} \vee C)$ на $(\bar{A} \vee B \vee C)(A \vee C)$. Если A и B длинные, тогда мы можем расщепиться на два случая:

- заменить $(x \vee A)(x \vee B)(\bar{x} \vee C)$ на $(A \vee C)(B \vee C)$,
- заменить на 0 все литералы из A, B, C.

Корректность этих шагов доказана в Упрощающее правило 5 и Расщепляющее правило 2.

Глава 2. Параметризованная задача максимальной выполнимости

2.1. Правила упрощения и расщепления

Правило упрощения 1 (**SR1**). Чистому литералу l можно присвоить значение 1.

Доказательство. Если $\bar{l}$ не встречается в формуле, то нет никакого смысла присваивать значение 0 литералу l. □

Правило упрощения 2 (**SR2**). Формулу, содержащую переменную степени два, можно упростить.

Доказательство. Пусть переменная x встречается два раза. Если один x или $\bar{x}$ является чистым, то воспользуемся $SR1(x)$. Иначе $F = G \wedge (x \vee A) \wedge (\bar{x} \vee B)$. Легко видеть, что MAX-SAT $(F, k) =$ MAX-SAT $(F \wedge (A \vee B), k - 1)$ □

Правило упрощения 3 (**SR3**). Пару дизъюнктов (x) и $(\bar{x})$ можно удалить.

Доказательство. Нетрудно видеть, что MAX-SAT $(F \wedge (x) \wedge (\bar{x}), k) =$ MAX-SAT $(F, k-$

□

Правило упрощения 4 (**SR4**). Если две переменные x, y степени три встречаются вместе в трех дизъюнктах, тогда эти три дизъюнкта можно одновременно выполнить, присвоив переменным x, y значения.

Доказательство. Мы можем выполнить два дизъюнкта, выбрав значение переменной x, после этого нам останется выполнить один дизъюнкт. Последний дизъюнкт легко выполнить, выбрав соответствующее значение для y. □

Правило упрощения 5 (SR5). Пусть x — переменная степени 3. Тогда $F = G \wedge (x \vee A) \wedge (x \vee B) \wedge (\bar{x} \vee C)$. Если длина A или B меньше двух, тогда формула F может быть упрощена.

Доказательство. Если длина A равна 1 (то есть A — это литерал), то

$$\text{MAX-SAT } (F, k) = \text{MAX-SAT } (G \wedge (\bar{A} \vee B \vee C) \wedge (A \vee C), k - 1)$$

Если длина A не равна единице, тогда A пусто и можно просто положить $x = 1$. □

Замечание 1. Все упрощающие правила могут быть применены к формуле за полиномиальное время.

Правило расщепления 1 (BR1). Для любого литерала l есть расщепление $(F[l], k - \#_F(l)), (F[\bar{l}, k - \#_F(\bar{l})])$.

Правило расщепления 2 (BR2). Если x — переменная степени 3: $F = G \wedge (x \vee A) \wedge (x \vee B) \wedge (\bar{x} \vee C)$, то существует расщепление:

- $(G \wedge (A \vee C) \wedge (B \vee C), k - 1)$
- $(G', k - 2)$, где G' получается из G присваиванием значения 0 всем литералам, содержащимся в дизъюнктах A, B, C.

Доказательство. Положим $R = (A \vee C) \wedge (B \vee C)$. Заметим, что если если оптимальное означивание удовлетворяет s дизъюнктов из R, где $s > 0$, тогда мы можем удовлетворить $s + 1$ дизъюнкт в $F - G$, но не можем удовлетворить $s + 2$. Однако, если оптимальное означивание не выполняет ни одного дизъюнкта в R, тогда оно выполняет два дизъюнкта среди $(x \vee A) \wedge (x \vee B) \wedge (\bar{x} \vee C)$, надо просто положить $x = 1$. □

Следствие 2.1. Если $A \vee B \vee C$ содержит комплементарные литералы, то достаточно рассмотреть только первую ветку в выше предложенном расщеплении.

Замечание 2. Мы пишем $BR2(x)$, если мы применяем Расщепляющее правило 2 для переменной x. Расщепление по переменной означает применение Расщепляющего правила номер 1. Мы пишем $SRj(x)$, для $1 \leq j \leq 5$, если мы применяем Упрощающее правило j к переменной x.

Лемма 2.1. Если литерал y доминирует литерал x, тогда мы можем рассмотреть расщепление:

- $x = 1, y = 0$
- $x = 0$

Доказательство. Если в некотором означивании переменные x и y обе принимают значение 1, тогда точно такое же означивание, только при $x = 0$ выполяет не меньшее количество дизъюнктов. Ведь все дизъюнкты, которые выполнены за счет того что $x = 1$, также будут выполнены ввиду $y = 1$. □

Лемма 2.2. Пусть $x - (t, 1)$ переменная, при этом отрицание не образует отдельного дизъюнкта. Тогда расщепление по переменной $x - (t, 2)$ расщепление.

Доказательство. Пусть y — сосед литерала $\bar{x}$ (то есть литерал встречается с $\bar{x}$ в одном и том же дизъюнкте). В ветке $x = 1$ мы выполним как минимум t дизъюнктов. А в ветке $x = 0$ мы можем положить $y = 0$ и тем самым выполним как минимум еще один дизъюнкт. □

Лемма 2.3. Если формула F содержит переменную степени ≥ 6, тогда число расщепления по переменной x не больше $\tau(1, 5)$.

Доказательство. Этот факт следует из того что $\tau(1, 5) > \tau(2, 4) > \tau(3, 3)$. □

2.2. Алгоритм для $(n,3)-$MAX-SAT за время 1.2721^k

Через $(n,3)-$MAX-SAT обозначим частный случай задачи максимальной выполнимости, где каждая переменная встречается не более трёх раз. В этом разделе представлен простой алгоритм для $(n,3)-$MAX-SAT . Время работы алгоритма равняется 1.2721^k. Эта верхняя оценка лучше верхней оценки данной Ченем и Канжем, доказанной для общего случая. На протяжении всего раздела мы предполагаем, что F — это $(n,3)$-MAX-SAT формула.

Лемма 2.4. Возьмем переменную x степени 3, тогда $F = G \wedge (x \vee A) \wedge (x \vee B) \wedge (\bar{x} \vee C)$. Если $|A| < 2$, тогда найдется $(2,4)$-расщепление и новые формулы будут $(n,3)-$MAX-SAT формулами.

Доказательство. Ввиду Упрощающего правила 5 мы убираем один дизъюнкт и получаем новую формулу $F' = G \wedge (\bar{A} \vee B \vee C) \wedge (A \vee C)$. Переменные степени 4 в формуле F' могут появится только за счет A и C. Расщепившись по переменной A мы в каждой ветке получим снова $(n,3)-$MAX-SAT формулу. При этом мы получим как минимум $(1,3)-$расщепление поскольку $\tau(1,3) > \tau(2,2)$ и A в новой формуле было переменной степени 4. Учитывая, что до этого мы уже выполнили один дизъюнкт получаем итоговое расщепление $(2,4)$. □

Лемма 2.5. Если каждая переменная в формуле F встречается один раз, отрицательно и дважды положительно и все отрицательные литералы образуют единичные дизъюнкты. Тогда MAX-SAT (F) может быть найден за полиномиальное время.

Доказательство. Построим граф $G_F = (V, E)$ следующим образом. Каждая вершина будет соответствовать дизъюнкту состоящему из положительных литералов, каждое ребро соединяет две вершины, если им соответствующие дизъюнкты содержат общую переменную. Для каждой общей

переменной мы используем свое ребро. Тогда MAX-SAT $(F) = n + v(G_F)$, где $v(G_F)$ — это размер наибольшего паросочетания. □

Algorithm 1 $(n,3)$-MAX-SAT-Alg — решение $(n,3)$-MAX-SAT за время 1.2721^k.

Input: F — $(n,3)$-MAX-SAT формула.
Parameter: k — число дизъюнктов, которое надо выполнить.
Output: 1, если k дизъюнктов может быть одновременно выполнено; 0 иначе.

1: применить Упрощающие правила 1–4.
2: **if** все отрицательные литералы образуют отдельные дизъюнкты **then**
3: **return** $k \leq$ MAX-SAT (F) (лемма 2.5).
4: выбрать x, т.ч. у $\bar{x}$ есть сосед: $(x \vee A)(x \vee B)(\bar{x} \vee C), |C| > 0, |A| \leq |B|$.
5: **if** $|A| \leq 1$ **then**
6: использовать лемму 5 для расщепления.
7: **if** $|A| \geq 2$ **then**
8: расщепиться BR $1(x)$.

Теорема 2.1. Алгоритму $(n,3)$-MAX-SAT-Alg для решения $(n,3)$-MAX-SAT требуется не более 1.2721^k шагов.

Доказательство. По лемме 2.5 время работы 3 шага занимает полиномиальное время. Шаг 6 дает $(2,4)$-расщепление по лемме 3. Расщепление на шаге 8 это как минимум $(4,2)$-расщепление. Действительно, $|C| > 0$ и из лемма 2.1 следует, что в случае $x = 0$ мы выполняем не менее 2 дизъюнктов: один это $\bar{x} \vee C$ и еще один с помощью литерала $\bar{y}$, где y некоторый литерал из C. В виду SR4, переменные из A встречаются как минимум в 4 дизъюнктах. В ветке $x = 1$ два дизъюнкта выполнены за счет x и не менее двух дизъюнктов в виду SR4 примененных к различным литералам из A. То есть расщепившись по x мы получим $(4,2)$-расщепление.
Время работы алгоритма составляет $\max(\tau(2,4), \tau(3,3))^k = \tau(2,4)^k < 1.2721^k$.
□

Замечание 3. Иногда мы не делаем различий между случаями, когда мы выполнили 4 дизъюнкта и когда мы уменьшили параметр задачи на 4. К примеру, когда мы применяем $SR2(x)$, мы уменьшаем параметр задачи, но пишем, что удовлетворили один дизъюнкт.

Глава 3. Избавление от переменных степени три

В этой части мы покажем, что если переменная содержит переменную степени три, тогда мы можем или уменьшить параметр или найти хорошее расщепление. Предположим, x встречается три раза в формуле и никакое правило упрощения не применимо к формуле. Неограничив общности будем считать, что формула F содержит дизъюнкты $(x \vee A), (x \vee B), (\bar{x} \vee C)$, где A, B, C — это дизъюнкции литералов. Мы рассмотрим только случаи когда, $|A|, |B| \geq 2$, поскольку иначе мы можем применить $SR5(x)$ или присвоить переменной x значение 1.

Определение 3.1. Обозначим через $LN(!A_1, \ldots, !A_k)$ множество всех дизъюнктов, содержащих отрицание хотя бы одного литерала из $A_1 \vee A_2 \vee \cdots \vee A_k$.

На протяжении этого раздела мы предположим, что $A \vee B \ \vee C$ не содержит противоречивых литералов. Иначе, по следствию 2.1 формула может быть упрощена.

Лемма 3.1. Пусть y, z — такие литералы, что $y, z \in A \vee B \vee C$, $\bar{y}$ встречается не менее двух раз и доминирует $\bar{z}$(назовем эту ситуацию первый случай доминирования). В этом случае существует $(2, 4)$-расщепление.

Доказательство. Напомним, что литералы $\bar{y}, \bar{z}$ встречаются в некоторых дизъюнктах из $LN(!A, !B, !C)$. Рассмотрим $F[y], F[\bar{y}]$. Во втором случае два дизъюнкта выполнены литералом $\bar{y}$ и переменную z можно означить 1, поскольку z станет чистым литералом. $z = 1$ выполнит один из дизъюнктов $x \vee A, x \vee B, \bar{x} \vee C$. Поэтому можно будет применить правило $SR2(x)$ (x встречается не более 2 раз). Таким образом мы выполнили 4 дизъюнкта. В ветке расщепления $y = 1$ мы можем применить $SR2(x)$ и тем самым уменьшим параметр задачи на 2. □

Лемма 3.2. Пусть y, z — литералы из $A \vee B \vee C$ такие, что $\bar{y}$ встречается один раз и доминирует $\bar{z}$ (назовем эту ситуацию вторым случаем доминирования). В подобной ситуации существует $(3,3)$-расщепление.

Доказательство. Как и в прошлой лемме в ветке $F[\bar{y}]$ можно положить $z = 1$. Литерал z встречается в формуле как минимум два раза, поскольку $\bar{z}$ втречается один раз и все переменные имеют степень больше 2. А значит $z = 1$ выполняет еще два дизъюнкта. Итого в этой ветке мы выполнили три дизъюнкта. При $y = 1$ мы выполнили как минимум два дизъюнкта с литералом y и после этого мы можем применить $SR2(x)$. Тем самым мы получили $(3,3)$ расщепление. □

Лемма 3.3. Если $|LN(!A, !B, !C)| < 3$ тогда мы имеем специальный случай доминирования или $A = B = y \vee z$.В первом случае у нас есть хорошее расщепление, а во втором параметр может быть уменьшен.

Доказательство. Мы знаем, что $|A|, |B| \geq 2$. Значит $A \cup B$ либо содержит более двух различных литералов, либо равняется $y \vee z$. В первом случае у нас получается, что три литерала должны содержаться в двух дизъюнктах, но это не возможно без доминирования. А во втором случае можем заменить все дизъюнкты с переменной x на $y \vee z \vee C$ и уменьшить параметр на два. □

НУО мы можем предположить, что сейчас мы будем рассматривать только формулы, в которых $|LN(!A, !B, !C)| > 2$. Если $|LN(!A, !B, !C)| > 3$, тогда используя $BR2(x)$ мы сразу получим $(1,6)$-расщепление. Полагаем до окончания секции $|LN(!A, !B, !C)| = 3$.

Лемма 3.4. Если $|A \vee B \vee C|$ содержит более трех различных литералов, тогда у нас будет один из специальных случаев доминирования и значит одна из лемм 3.1,3.2 применима. То есть в подобном случае у нас есть $(2,4)-$ или $(3,3)-$ расщепление.

Доказательство. Как минимум отрицание четырех литералов должно быть помещено в три дизъюнкта, но это невозможно без одного из специальных случаев доминирования. □

Из предыдущей леммы следует, что достаточно рассматривать только формулы, в которых $|A \vee B \vee C|$содержит не более 3 различных литералов.

Лемма 3.5. Если $\min\{|A|, |B|\} \geq 3$ тогда или есть специальный случай доминирования, или параметр задачи может быть уменьшен.

Доказательство. Если $|A \cup B \cup C| > 3$ тогда по предыдущей лемме мы имеем специальный случай доминирования. Иначе из $|A \cup B \cup C| \geq 3$ и $min\{|A|, |B|\} \geq 3$ следует, что |A|=|B| и более того:

$$x \vee A = x \vee B = x \vee y_1 \vee y_2 \vee y_3.$$

А это значит можно заменить $(x \vee A) \wedge (x \vee B) \wedge (\bar{x} \vee C)$ на $A \vee C$ и уменьшить параметр на 2, после таких манипуляций ответ не измениться. □

Сейчас без ограничения общности можно считать, что $x \vee A = x \vee y \vee z$.

Лемма 3.6. Если нам надо решить задачу (F, k) и для всех переменных x, что встречаются ровно три раза, выполнено следующее:

- x участвует в дизъюнктах $x \vee A_x, x \vee B_x, \bar{x} \vee C_x$
- $LN(!A_x, !B_x, !C_x) = 3$

тогда можно уменьшить параметр или применить одно из расщеплений: $(3, 3), (2, 4)$ или лучше.

Доказательство. Предполагаем, что ни одна из предыдущих лемм не применима, иначе просто применим одну из них. Таким образом мы можем выбрать переменную x, которая встречается три раза и $A_x = y \vee z$. Заметим,

что если $\bar{y}$ встречается три раза, то у нас будет специальный случай доминирования и, значит, хорошее расщепление будет найдено. Рассмотрим два случая: $\bar{y}$ встречается один раз, $\bar{y}$ встречается два раза.

Случай 1: $\bar{y}$ встречается ровно один раз.

Случай 1.1: у $\bar{y}$ есть сосед.

F содержит следующие дизъюнкты:

$$(x \vee y \vee z), \quad (x \vee \dots), \quad (\bar{x} \vee \dots), \quad (\bar{y} \vee w \vee \dots)$$

В $F[\bar{y}]$ по лемме мы можем положить $w = 0$ и использовать $SR5(x)$ или $SR2(x)$, тем самым мы уменьшим параметр на три. В $F[y]$ мы выполним два дизъюнкта, а затем используя $SR2(x)$ уменьшим параметр на 1. Таким образом получено $(3, 3)$-расщепление. Значит в дальнейшем можно считать, что $(\bar{y})$ образует отдельный дизъюнкт.

Случай 1.2: литерал y встречается более 2 раз и есть дизъюнкт с y,

в котором нет переменной x

После $F[y]$ и SR2(x) мы выполним как минимум 4 дизъюнкта. В $F[\bar{y}]$ применим $SR5(x)$ и в итоге получим уменьшение параметра как минимум на 2.

Случай 1.3: литерал y встречается во всех дизъюнктах с переменно

Формула содержит следующие дизъюнкты:

$$(x \vee y \vee z), \quad (x \vee y \vee \dots), \quad (\bar{x} \vee y \vee \dots), \quad (\bar{y})$$

Переменная y не встречается в оставшейся части формулы, поскольку иначе это был бы случай 1.2. В подобной ситуации достаточно рассмотреть только случай $y = 0$. Поскольку означивание

$x = 1, y = 0$ не хуже означиваний $x = y = 1$ и $x = 0, y = 1$. Это означает, что мы можем удовлетворить один дизъюнкт и удалить одну переменную без расщепления.

Случай 1.4: y встречается ровно дважды

Ипользуя симметрию можно заключить, что дизъюнкт с $\bar{x}$ не содержит других переменных. Иначе у нас получился бы случай 1.1, но только относительно y. Снова используя симметрию можем заключить, что или $\bar{z}$ встречается дважды или $\bar{z}$ встречается только как отдельный дизъюнкт и литерал z появляется в формуле два раза. Рассмотрим два этих слуйчая по отдельности.

Случай 1.4.1: $\bar{z}$ встречается один раз, а z дважды

$$(x \vee y \vee z), \quad (x \vee \dots), \quad (\bar{x}), \quad (\bar{y}), \quad (\bar{z})$$

В $F[x]$ используя SR1(y), SR1(z) мы уменьшим параметр на 4. В $F[\bar{x}]$ легко видеть, что мы можем положить $y = \bar{z}$. В итоге мы получаем $(4, 4)$-расщепление.

Случай 1.4.2: $\bar{z}$ встречается дважды.

В этом случае формула содержит дизъюнкты:

$$(x \vee y \vee z), \quad (x \vee \dots), \quad (\bar{x}), \quad (\bar{y}), \quad (\bar{z} \vee \dots), \quad (\bar{z} \vee \dots)$$

В ветке $F[z]$ применив SR1(x), SR1(y) мы уменьшим параметр на 3. В $F[\bar{z}]$ мы можем положить $x = \bar{y}$, получив тем самым $(3, 4)$-расщепление.

Случай 2: $\bar{y}$ встречается ровно дважды.

Учитывая симметрию можно предположить, что $\bar{z}$ также встречается дважды, иначе мы будем иметь ситуацию описанную в случае 1, только используя z вместо y. Таким образом наша формула содержит следующее семейство дизъюнктов:

$$(x \vee y \vee z), \quad (x \vee B), \quad (\bar{x} \vee \dots), \quad (\bar{y} \vee \dots), \quad (\bar{y} \vee \dots)$$

Предположим $y \in B$ (случай $z \in B$ рассматривается аналогично). В ветке $F[y]$ можно положить $x = 0$ и это удаляет 3 дизъюнкта. А в ветке $F[\bar{y}]$ можно применить $SR5(x)$ и это также удалит три дизъюнкта. Если y, z не встречаются в B, тогда или $|B| < 2$ — противоречие или $|A \vee B| \geq 4$ и существует специальный случай доминирования.

□

Из выше доказанных лемм вытекает:

Теорема 3.1. Если переменная встречается в формуле ровно 3 раза, тогда параметр задачи может быть уменьшен или существует $(1, 6), (2, 4), (3, 3)$-расщепление.

Глава 4. Решение MAX-SAT за 1.358^k

В этой главе мы представим простой алгоритм, который улучшает верхнюю оценку для параметризованного MAX-SAT (Algorithm MAX-SAT-Alg). Сложный случай задачи — это ситуация, когда все переменные $(1,3)-$одиночки или $(1,4)-$одиночки. Мы рассмотрим два этих случая отдельно.

Мы сведем этот частный случай задачи MAX-SAT к задаче о минимальном покрытии множествами. В задаче о минимальном покрытии множествами, дано универсальное множество U и семейство $\mathcal{S}$ подмножеств множества U и необходимо найти минимальное количество подмножеств $S' \subset \mathcal{S}$, которые покрывают U: $\bigcup_{S_i \in S'} S_i = U$. Для $e \in U$, $f(e)$ (частота e) обозначает число подмножеств из $\mathcal{S}$ в которых присутствует e.

Эффективные точные алгоритмы для задачи Минимального покрытия множествами были придуманы Фомином, Грандони и Кратчем [16]. А ван Руй и Бодлайнер [17] сконструировали алгоритмы для задач поиска Минимального доминирующего множества и Максимального независимого множества. Время алгоритма ван Руя и Бодлайнера для задачи о Минимальном покрытии множествами состовляет $1.28759^{k(U,\mathcal{S})}$, где

$$k(U,\mathcal{S}) = \sum_{e \in U} v(f(e)) + \sum_{S_i \in \mathcal{S}} w(|S_i|),$$

и v, w — это функции весов. Максимальное значение, которое может принимать v, — это 0.595723 и максимальное значение w — 1. Заметим, что для множеств мощности 4 максимальное значение w есть 0.866888. Поэтом, мы будем использовать следующую лемму, показанную ван Руем и Бодлайнером [17].

Теорема 4.1. Алгоритм MSC решает задачу Минимального покрытия множествами, где мощность каждого множества в S не больше 4, за время

$O^*(1.28759^{0.595723|U|+0.866888|\mathcal{S}|}) \geq O^*(1.29^{0.6|U|+0.9|\mathcal{S}|})$.

Следующая теорема была доказана Лиеберхером и Спекером [18]. Позже Янакакис [19] дал простое доказательство, используя вероятностный метод.

Теорема 4.2. Если любые три дизъюнкта в формуле F можно выполнить одновременно, тогда как минимум $\frac{2m}{3}$ дизъюнктов можно одновременно выполнить в F.

Теорема 4.1 используется для примеров задач, когда $m < 1.5k$, а теорема 4.2 для случая $m \geq 1.5k$. Сейчас мы приведем верхнюю оценку.

Algorithm 2 MAX-SAT-Alg — решает MAX-SAT за время 1.3579^k.

Input: F — экземпляр задачи MAX-SAT .
Parameter: k — число дизъюнктов, которое надо удовлетворить.
Output: 1, если можно одновременно выполнено k дизъюнктов; 0 иначе.

применим упрощающие правила 1–5.
if существует x, что $deg(x) \geq 6$ **then**
  расщепиться по x.
if существует x степени 3 **then**
  расщепиться по x согласно теореме 3.1.
{Сейчас у нас есть только переменные степени 4 и 5.}
if F содержит $(3,2)$, $(3,1)$ или $(4,1)$ переменную x и нет дизъюнкта $\bar{x}$ **then**
  расщепимся по x.
if все переменные $(3,1)$-одиночки или $(4,1)$-одиночки **then**
  if $k \leq n$ **then**
    return 1
  if $m < 1.5k$ **then**
    return $k \leq \text{MSC}(F)$
  if существует дизъюнкт длины 2: $(x \vee y)$ **then**
    расщепиться $F[x,y]; F[x = \bar{y}]$.
  else
    return 1.
{Сейчас у нас есть только одиночки и $(2,2)$-переменные. Также, у нас есть общительная переменная, а значит существует $(2,2)$-переменная x.}
if у x есть $(4,1)$-одиночка сосед y **then**
  расщепиться по y.
else
  расщепиться по x.

Теорема 4.3. Алгоритм MAX-SAT-Alg решает MAX-SAT за время 1.3579^k.

Доказательство.

- Шаг 3. Если $deg(x) \geq 6$ тогда по лемме 2.3 мы получаем $(1,5)$-расщепление. $\tau(1,5) \approx 1.3248 < 1.3579$.

- Шаг 5. Если $deg(x) = 3$ тогда по теореме 3.1 мы получаем $(1, 6)$-расщепление. $\tau(1, 6) \approx 1.2852 < 1.3579$.
- Шаг 7. $(3, 2)$-переменная дает $\tau(3, 2) \approx 1.3248 < 1.3579$.
- Шаг 7. По следствию 2.2, расщепление по $(4, 1)$-общительной или $(3, 1)$-общительной переменной дает как минимум $\tau(3, 2) \approx 1.3248 < 1.3579$.
- Шаг 8. У нас есть только переменные $(3, 1)$- или $(4, 1)$-одиночки.
  - Шаг 10. Сейчас у нас все переменные одиночки. Это значит мы можем выполнить n дизъюнктов положив все переменные равные 0. Если $k \leq n$, то задача решена.
  - Шаг 12. Предположим, что каждая переменная встречается 3 или 4 раза положительно и один раз отрицательно, при этом образовывая отдельный дизъюнкт. Заметим, что тогда все дизъюнкты состоят только из положительных литералов или дизъюнкт состоит из одного отрицательного литерала. Мы утверждаем, что в таком случае есть оптимальное означивание, которое выполняет все дизъюнкты с положительными литералами. Действительно, если один из таких дизъюнктов не выполнен, тогда изменив значение любой переменной в него вхождящей, мы не уменьшим количество выполненных дизъюнктов, но увеличим количество выполненных дизъюнктов с положительными литералами. То есть наша задача эквивалентна поиску означивания с наименьшим количеством единиц в нем и при этом все положительные дизъюнкты были выполнены. В действительности это задача о минимальном покрытии множествами. Мы построим экземпляр задачи о минимальном покрытии множествами следующим образом. Пусть U множество всех положительных дизъюнктов ($|U| = m - n$). $\mathcal{S}$ содержит n подмножеств. Положим $S_i \in \mathcal{S}$ состоящим из всех дизъюнктов содержащих литерал x_i.

Сейчас мы хотим покрыть U минимальным количеством подмножеств из $\mathcal{S}$. Если t это минимальное количество подмножеств требуемых для покрытия U, тогда максимальное количество выполненных дизъюнктов это $m-t$. Мы можем сравнить это число с k и выдать ответ. По теореме 4.1, алгоритм для задачи о минимальном покрытии множествами для подмножеств множности ≤ 4 имеет время работы не больше $\leq O^*(1.29^{(0.6(m-n)+0.9n)})$. Мы знаем, что $k > n$ и $m < 1.5k$. Значит $T(F) \approx 1.3574^k < 1.3579^k$.

  - Шаг 14. Мы знаем, что формула содержит дизъюнкты $(\bar{x})$ and $(\bar{y})$. Если есть дизъюнкт $(x \vee y)$,тогда некоторое оптимальное означивание выполняет $(x \vee y)$. Значит, мы можем расщепиться $x = y = 1$ и $x = \bar{y}$. В первой ветке мы выполним не менее 3 дизъюнктов, поскольку x есть $(3,1)-$ или $(4,1)-$ переменная. Во второй ветке мы выполняем $(x \vee y)$ и ввиду упрощающего правила 3 мы выполняем один их дизъюнктов (x) и $(\bar{x})$. Мы получили $(2,3)$-расщепление.
  - Шаг 16. Сейчас у нас формула с $m \geq 1.5k$ дизъюнктами. F не содержит дизъюнктов длины 2. А значит любые три дизъюнкта могут быть выполнены одновременно. По теореме 4.2 существует означивание, которое выполняет $\frac{2m}{3} \geq k$ дизъюнктов.

- Шаг 18. Пусть x это $(2,2)$-переменная, y —$(4,1)$-одиночка и сосед x. Расщепление по y дает $\tau(4,1)$ и в ветке $y = 1$ у нас есть переменная степени 3. Поэтому проведя еще одну итерацию в ветке $y = 1$ мы получим $\tau(4+1, 4+6, 1) = \tau(5, 10, 1) < 1.3579$.
- Шаг 20. Пусть x это $(2,2)$-переменная. Соседи у x это переменные степени 4. То есть обе ветки $F[x]$ и $F[\bar{x}]$ содержат переменные не выше 3. По теореме 3.1,переменная степени 3 дает $(1,6)$,- $(2,4)$- или $(3,3)$-расщепление. Значит в худшем случае $\tau(2+1, 2+6, 2+1, 2+6) = \tau(3,8,3,8) \approx 1.3480 < 1.3579$.

□

Замечание 4.1. Чтобы найти означивание, которое выполняет как минимум k дизъюнктов, мы можем использовать самосводимость SAT. $\#_x F$ — число дизъюнктов, содержащих литерал x. Пусть $F_0 = F$. Для i от 1 до n подставляем в F_{i-1} значение x_i равное 1. Если алгоритм говорит, что в новой формуле можно выполнить $k - \#_x F_{i-1}$ дизъюнктов, тогда означиваем $x_i = 1$ и $F_i = F_{i-1}[x_i = 1]$, иначе $x_i = 0$ и $F_i = F_{i-1}[x_i = 0]$.

Глава 5. Новая верхняя оценка для $(n,3)$-MAX-SAT относительно переменных

На данный момент не известно алгоритма решающего задачу MAX-SAT за время меньшее 2^n. Однако такие верхние оценки известны для некоторых частных случаев. В данной главе мы приведем подобную верхнюю оценку для частного случая $(n,3)$-MAX-SAT .

5.1. Алгоритм

Алгоритм представлен ниже. Анализ будет представлен в следующей части.

Алгоритм N3MaxSat

Input: Формула F в КНФ

Output: Максимальное число одновременно выполнимых дизъюнктов

Нормализационные правила:

N1 Убрать все пустые дизъюнкты

N2 Заменить все дизъюнкты содержащие противоречивые литералы на (T)

N3 If $(F = F_1 \wedge F_2)$, где F_1, F_2 замкнутые формулы,
then return $\text{N3MaxSat}(F_1) + \text{N3MaxSat}(F_2)$

N4 Переименовать все переменные, так, что для любой переменной x, литерал $\bar{x}$ встречался ровно один раз

Упрощающие правила:

S1 If (F содержит чистый литерал l), **then** return $\text{N3MaxSat}(F[l])$

S2 If $(F = (x \vee A) \wedge (\bar{x} \vee B) \wedge F_1)$, где A, B дизъюнкты, F_1 — формула, которая не содержит x и $\bar{x}$, **then** return $\text{N3MaxSat}((A \vee B) \wedge F_1) + 1$

S3 If (F содержит дизъюнкт (x)), **then** return $\text{N3MaxSat}(F[x])$

S4 If $(F = (x \vee y) \wedge (x \vee y) \wedge (\bar{x} \vee A) \wedge (\bar{y} \vee B) \wedge F_1)$,
then return $\text{N3MaxSat}((A \vee B) \wedge F_1) + 3$

S5 If $(F = (x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee A) \wedge (y \vee B) \wedge F_1)$,
then return $\text{N3MaxSat}(F_1 \wedge A) + 3$

S6 If $(F = (\bar{x} \vee y) \wedge C \wedge F_1$ и $(l_x, l_y \in C)$), где C — дизъюнкт, F_1 — формула
then return $\text{N3MaxSat}(F[x = y])$

Правила расщепления:

B1 If $((F = C \wedge F_1)$ и $(\bar{x}, \bar{y} \in C))$, где C это дизъюнкт, F_1 — формула,
then return $\max(\text{N3MaxSat}(F[x]), \text{N3MaxSat}(F[\bar{x}]))$

B2 If $((F = C \wedge F_1)$ и $(\bar{x}, y, z \in C))$, где C это дизъюнкт, F_1 формула,
then return $\max(\text{N3MaxSat}(F[x]), \text{N3MaxSat}(F[\bar{x}]))$

B3 If $(F = (y \vee \bar{x}) \wedge F_1)$,
then return $\max(\text{N3MaxSat}(F[x, y]), \text{N3MaxSat}([\bar{x}, \bar{y}]))$

Паросочетание:
Комментарий: F зависит от n переменных и содержит k дизъюнктов (T)
Построим граф G_F:
— вершина будет соответствовать дизъюнкту с положительными литералами;
— ребро соединяет две вершины, если соответствующие дизъюнкты содержат
общую переменную, для каждой такой переменной есть свое ребро (G_F мультиграф).
Найти максимальное паросочетание M в графе G_F, return $k + n + |M|$

5.2. Анализ алгоритма

Мы опустим доказательства корректности правил нормализации и упрощающих правил $S1 - S5$, поскольку в их корректности несложно убедиться.

Лемма 5.1. Если $F = (\bar{x} \vee y) \wedge F_1$, тогда существует оптимальное означивание τ такое, что $\tau(x) = \tau(y)$.

Доказательство. Достаточно доказать, что $Opt(F[x, y]) \geq Opt(F[x, \bar{y}])$ и $Opt(F[x, y]) \geq Opt(F[y, \bar{x}])$. Первое утверждение следует из того, что если $x = 1$,то можно положить $y = 1$ согласно упрощающему правилу три, то есть $Opt(F[x, y]) \geq Opt(F[x, \bar{y}])$. Второе утверждение вытекает из того что, если $y = 1$, то в виду **S1** мы можем означить $x = 1$, получаем $Opt(F[x, y]) \geq Opt(F[\bar{x}, y])$. □

Корректность правила **B3** следует из выше доказанной леммы.

Лемма 5.2 (корректность **S6**)**.** Пусть $F = (\bar{x} \vee y) \wedge C \wedge F_1$ и $l_x, l_y \in C$. Тогда $Opt(F[x = y]) = Opt(F)$. После нормализации, формула $F[x = y]$ содержит переменную x максимум три раза.

Доказательство. Согласно лемме 5.1, достаточно рассмотреть означивания в которых $x = y$. Заменим y на x в формуле F. В новой формуле переменная x встречается 6 раз. Но мы можем элиминировать дизъюнкт $(x \vee \bar{x})$ и как минимум один литерал из дизъюнкта C. □

5.2.1. Корректность последнего шага

Если правила нормализации, упрощения и расщепления не применимы к функции F, тогда любой дизъюнкт содержащий отрицательный литерал имеет длину ровно 1. По такой формуле F, алгоритм строит граф G_F и находит максимальное паросочетание в нем. Заметим, что число переменных в F равно числу дизъюнктов с отрицательными литералами. Более того это число равняется числу ребер в G_F.

Теорема 5.1. Если каждая переменная формулы F встречается один раз отрицательно и дважды положительно и все отрицательные литералы встречаются в дизъюнктах длины один, тогда

$$Opt(F) = V(G_F) + E(G_F) - \rho(G_F) = E(G_F) + \nu(G_F),$$

где $E(G_F), V(G_F), \nu(G_F), \rho(G_F)$ соответственно число ребер, число вершин, размер максимального паросочетания, размер минимального реберного покрытия в графе G_F.

Доказательство. Рассмотрим оптимальное означивание τ для F. Не ограничив общности предположим, что τ выполняет все дизъюнкты с положительными литералами (иначе мы можем изменить значение переменной в невыполненном дизъюнкте с положительными литералами без уменьшения числа выполненных дизъюнктов). Заметим, что все ребра помеченные переменными означенными в true означиванием τ формирует реберное покрытие графа G_F. Таким образом, $Opt(F) = V(G_F) + E(G_F) - \rho(G_F)$. Для завершения доказательства воспользуемся, тем что: $\rho(G_F) + \nu(G_F) = V(G_F)$. □

5.3. Время работы

В этой секции, мы докажем верхнюю оценку на время работы представленного алгоритма, оценив число переменных в двух формулах полученных после расщепления. Далее предполагаем, что к формуле уже не применимы правила нормализации и упрощения.

Определение 5.1. $VN(x)-$окрестость — это множество переменных встречающихся в одном дизъюнкте вместе с литералом x.

Лемма 5.3. В формуле $F[x]$ мы можем элиминировать как минимум две переменные.

Доказательство. Рассмотрим VN-окрестность x. Заметим, что $|VN(x)| \geq 2$, так как иначе одно из правил **S3**, **S4**, **S5**, **N3** можно было бы применить к F. Тогда в формуле $F[x]$, существует как минимум две переменные встречающиеся не более двух раз. Они будут удалены с помощью правил нормализации и упрощения. □

Лемма 5.4. Если $F = C \wedge F_1$ и $\bar{x}, \bar{y} \in C$, где C — это дизъюнкт, F_1 — формула, тогда в формулах $F[x]$, $F[\bar{x}]$ мы можем элиминировать(используя правила нормализации или упрощения) две переменные.

Доказательство. Утверждение для $F[x]$ вытекает из леммы 5.3. Сейчас рассмотрим $F[\bar{x}]$. Если мы присвоим значение 0 переменной x, тогда y можно присвоить значение 1 согласно упрощающим правилам. Значит мы снова получили специальный случай леммы 5.3, но вместо x мы используем y. □

Ниже мы везде предполагаем, что **B1** не применимо к формуле F.

Лемма 5.5. Если $F = C \wedge F_1$ и $\bar{x}, y, z \in C$, где C — дизъюнкт, F_1 — формула, тогда в формулах $F[x]$, $F[\bar{x}]$ можно убрать (в виду правил упрощения и нормализации) как минимум две переменные.

Доказательство. Чтобы получить желаемый результат, достаточно увидеть, что $2 \leq \min\{|VN(x)|, |VN(\bar{x})|\}$. После этого доказательство идентично доказательству леммы 5.3. □

Ниже в этой секции мы предположим, что правила **B1** и **B2** не применимы к формуле F.

Лемма 5.6. Пусть $F = (\bar{x} \vee y) \wedge F_1$. Тогда или две переменные можно элиминировать из $F[x]$ и $F[\bar{x}]$ или одну переменную элиминировать в $F[\bar{x}]$ и четыре переменные в $F[x]$.

Доказательство. Лемма 5.1 утверждает, что существует оптимальное означивание τ такое, что $\tau(x) = \tau(y)$

Рассмотрим следующие случаи:

- Все переменные, содержащие литералы x, y, кроме $(\bar{x} \vee y)$, имеют длину как минимум три. В этом случае $|VN(x) \cup VN(y)| \geq 5$, поскольку эти дизъюнкты содержат только положительные литералы и их суммарная длина как минимум 9. То есть , если $x = 1$, тогда $y = 1$ и мы можем дополнительно убрать 3 переменные. Если $x = 0$, то $y = 0$, значит мы можем элиминировать одну переменную.
- В формуле есть дизъюнкт $(x \vee \bar{z})$ или $(y \vee \bar{z})$ и переменная z отличается от x, y тогда достаточно рассмотреть два случая $x = y = z = 0$ или $x = y = z = 1$ (как это было описано ранее). Значит в этом случае мы убираем две переменные в формулах $F[x], F[\bar{x}]$.
- Единственный оставшийся случай, когда вместе с дизъюнктом $(y \vee \bar{x})$ у нас есть дизъюнкт $x \vee z$ или $y \vee z$. Если $x = 1$, мы элиминируем 2 переменные в $F[x]$ согласно 5.3. Если $x = 0$, тогда $y = 0$. Отсюда по упрощающему правилу **S3**, имеем $z = 1$.

□

Теорема 5.2. Время работы алгоритма N3MaxSat есть $O^*(2^{\frac{n}{3}})$.

Доказательство. Обозначим через $T(n)$ время работы алгоритма в худшем случае для формул зависящих от n переменных. Заметим, что правила нормализации, упрощения и расщепления применимы за время полиномиальное от n (очевидно, длина формулы линейна). Более того, если мы применим любое правило нормализации, упрощения или расщепления, то длина формулы уменьшится, а количество переменных не возрастет.

Сейчас мы рассмотрим правила расщепления. Когда мы применяем расщепляющее правило мы получаем две меньшие формулы. Из лемм 5.4, 5.5, 5.6 мы получаем следующее реккурентное соотношение на время работы

$$T(n) \leq T(n-2) + T(n-5) + poly(n),$$
$$T(n) \leq 2T(n-3) + poly(n).$$

Финальная часть нашего алгоритма: построение графа G_F и нахождение максимального паросочетания занимает лишь полиномиальное время.

Хорошо известно, что в этом случае $T(n) = O^*(\lambda^n)$, где λ это наибольший корень уравнений $x^3 - 2 = 0$, $x^5 - x^3 - 1 = 0$. Наибольший корень первого уравнения $1.25993\ldots$ и он больше, чем наибольший корень второго уравнения $1.23652\ldots$. Это завершает доказательство.

□

Заключение

Основной частью работы являются два новых алгоритма, улучшающие предыдущие лучшие верхние оценки. В частности, был построен новый алгоритм для параметризованного MAX-SAT со временем работы 1.358^k, до этого лучший результат был 1.3695^k, полученный Ченем и Канжем в 2002 году. Второй алгоритм решает задачу $(n, 3)$-MAX-SAT за время 1.259^n, n — количество переменных, предыдущее лучшее время составляло 1.273^n. Для достижения полученных результатов наряду со стандартным методом расщепления использовались новые правила упрощения и рассмотрение специальных окрестностей переменных.

Список литературы

[1] Patrizia Asirelli, Michele De Santis, and Maurizio Martelli. Integrity constraints in logic databases. *J. Log. Program.*, 2(3):221–232, October 1985.

[2] Roberto Battiti and Marco Protasi. Reactive search, a history-based heuristic for max-sat. *ACM Journal of Experimental Algorithmics*, 2, 1996.

[3] Henning Fernau and Rolf Niedermeier. An efficient exact algorithm for constraint bipartite vertex cover. *Journal of Algorithms*, 38(2):374 – 410, 2001.

[4] Pierre Hansen and Brigitte Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44(4):279–303, April 1990.

[5] Frederick Hayes-Roth, Donald A. Waterman, and Douglas B. Lenat. *Building expert systems.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1983.

[6] T. A. Nguyen, W. A. Perkins, T. J. Laffey, and D. Pecora. Checking an expert systems knowledge base for consistency and completeness. In *Proceedings of the 9th international joint conference on Artificial intelligence - Volume 1*, IJCAI'85, pages 375–378, San Francisco, CA, USA, 1985. Morgan Kaufmann Publishers Inc.

[7] Richard J. Wallace. Enhancing maximum satisfiability algorithms with pure literal strategies. In *In 11th Canadian Conference on Artificial Intelligence, AI'96*, pages 388–401. Springer, 1996.

[8] Richard Wallace and Eugene C. Freuder. Comparative studies of constraint satisfaction and davis-putnam algorithms for maximum satisfiability problems. In *Cliques, Coloring and Satisfiability*, pages 587–615. American Mathematical Society.

[9] Takao Asano and David P. Williamson. Improved approximation algorithms for max sat. In *In Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'00*, pages 96–105. ACM, 2000.

[10] Jianer Chen and Iyad Kanj. Improved Exact Algorithms for Max-Sat. In Sergio Rajsbaum, editor, *LATIN 2002: Theoretical Informatics*, volume 2286 of *Lecture Notes in Computer Science*, pages 98–119. Springer Berlin / Heidelberg, 2002.

[11] Meena Mahajan and Venkatesh Raman. Parameterizing above Guaranteed Values: MaxSat and MaxCut. *Journal of Algorithms*, 31(2):335 – 354, 1999.

[12] Rolf Niedermeier and Peter Rossmanith. New Upper Bounds for MaxSat. In Jiri Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Automata, Languages and Programming*, volume 1644 of *Lecture Notes in Computer Science*, pages 705–705. Springer Berlin / Heidelberg, 1999.

[13] Nikhil Bansal and Venkatesh Raman. Upper bounds for maxsat: Further improved. In *Proceedings of the 10th International Symposium on Algorithms and Computation*, ISAAC '99, pages 247–258, London, UK, 1999. Springer-Verlag.

[14] B. Monien and E. Speckenmeyer. Solving satisfiability in less than 2n steps. *Discrete Applied Mathematics*, 10(3):287 – 295, 1985.

[15] Evgeny Dantsin and Alexander Wolpert. MAX-SAT for Formulas with Constant Clause Density Can Be Solved Faster Than in 2n Time. In Armin Biere and Carla Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006*, volume 4121 of *Lecture Notes in Computer Science*, pages 266–276. 2006.

[16] Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. A measure & conquer approach for the analysis of exact algorithms. *J. ACM*, 56:1–32, August 2009.

[17] Johan M.M. van Rooij and Hans L. Bodlaender. Exact algorithms for dominating set. *Discrete Applied Mathematics*, 159(17):2147 – 2164, 2011.

[18] K. J. Lieberherr and E. Specker. Complexity of Partial Satisfaction. *J. ACM*, 28:411–421, April 1981.

[19] Mihalis Yannakakis. On the approximation of maximum satisfiability. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, SODA '92, pages 1–9, Philadelphia, PA, USA, 1992. Society for Industrial and Applied Mathematics.