Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет «Высшая школа экономики»

Факультет Санкт-Петербургская школа физико-математических и компьютерных наук
Департамент информатики

Основная профессиональная образовательная программа
«Прикладная математика и информатика»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
на тему
Новый алгоритм для задачи выполнения наибольшего количества дизъюнктов

Выполнил студент группы БПМ161, 4 курса,
Алфёров Василий Викторович

Научный руководитель:
Кандидат физико-математических наук,
доцент департамента информатики,
Близнец Иван Анатольевич

Санкт-Петербург
2020

Оглавление

Аннотация

Задача булевой выполнимости – исторически первая задача, для которой была доказана NP-полнота. Её оптимизационная версия, задача максимальной выполнимости, состоящая в выполнении наибольшего количества дизъюнктов в булевой формуле, также является NP-полной. Несмотря на то, что в предположении гипотезы об экспоненциальном времени эти задачи не могут быть решены за субэкспоненциальное время, задача максимальной выполнимости имеет большое количество применений, и подходы к этой задаче активно изучаются. В последние годы исследования версий задачи максимальной выполнимости, параметризованных общим количеством дизъюнктов и количеством выполненных дизъюнктов, сильно продвинулись за счёт введения сильных правил упрощения, основанных на правиле резолюции, и новых техник сведения экземпляра задачи к экземпляру задачи о покрытии множества. Другой важный результат заключается в том, что задача $(n,3)$-MAXSAT, параметризованная количеством переменных, решается гораздо быстрее, чем в общем случае [3]. В данной работе рассматривается задача максимальной выполнимости, решаемая относительно длины формулы, то есть суммарного количества литералов во всех дизъюнктах. Несмотря на то, что некоторые новые правила оказываются полезными для такой задачи, большинство из них увеличивают длину формулы и не могут быть применены. В этой работе представлены новые правила упрощения, не увеличивающие длину формулы. Также предлагается новый параметр с пониженной стоимостью 3-переменных, использующий то, что $(n,3)$-MAXSAT решается гораздо быстрее, чем общий случай задачи максимальной выполнимости. Комбинация двух методов позволяет получить алгоритм, работающий за время $O^*(1.093^L)$. Это улучшает предыдущую верхнюю оценку в $O^*(1.106^L)$, полученную Банзалом и Раманом [2].

Ключевые слова: задача максимальной выполнимости, параметризованные алгоритмы, точные экспоненциальные алгоритмы

Satisfiability problem is historically the first problem that was proven to be NP-complete. Maximum Satisfiability, being its optimization version, is also NP-complete. Though under assumption of Exponential Time Hypothesis those problems cannot be solved in subexponential time, Maximum Satisfiability have many applications, so approaches to solve its instances are nevertheless heavily studied. In the last years, research for versions of Maximum Satisfiability parameterized by the total number of clauses and the number of satisfied clauses have been pushed forward by introducing powerful resolution-based reduction rules and new techniques to reduce the problem instance to a Set Cover instance. In our work, we consider Maximum Satisfiability version parameterized by the formula length (sum of number of literals in each clause). Another important recent result is $(n,3)$-MAXSAT problem being solved in much better time than general case when parameterized by the number of variables [3]. Though some of new reduction rules appear to be very useful in this parameterization, most of them increase the length of the formula, and hence cannot be used for solving the problem in this parameterization. In our work, we introduce new reduction rules that do not increase the formula length. Then, we decrease the parameter value to discount the 3-variables, given that $(n,3)$-MAXSAT can be solved in much better time than general MAXSAT. The combination of two techniques produces an algorithm with running time $O^*(1.093^{|F|})$, improving the previous bound of $O^*(1.106^{|F|})$ by Bansal and Raman [2].

Keywords: Maximum Satisfiability, Parameterized Complexity, Exact Exponential Algorithms

Введение

Актуальность задачи

Задача о максимальной выполнимости, или, сокращённо, MAXSAT, как оптимизационная версия задачи о выполнимости (сокращённо SAT), одной из самых известных NP-полных задач, имеет широкий круг применений, от анализа данных [4] до медицины [12]. При этом не только задача MAXSAT, но многие её частные случаи, такие, как $(n, 3)$-MAXSAT, являются NP-трудными [15].

Гипотеза об экспоненциальном времени говорит, что задача 3SAT, то есть задача булевой выполнимости с дополнительным ограничением, что длина каждого дизъюнкта не более трёх, не может быть решена быстрее, чем за экспоненциальное время от количества переменных и тем более от длины входа. Как следствие, задача о максимальной выполнимости также не может быть решена за субэкспоненциальное время от длины входа, так как наличие такого алгоритма автоматически влекло бы за собой существование алгоритма для 3SAT. Поэтому основное направление исследований в этой области – уменьшение основания экспоненты.

В последние годы активно продвинулось изучение других параметризаций той же задачи: параметризация количеством выполненных дизъюнктов [8] и общим количеством дизъюнктов [16]. Для формул с большой средней длиной дизъюнкта эти алгоритмы дают хорошее время работы. Однако если в формуле большинство дизъюнктов имеют длину 1 или 2, алгоритм относительно длины применять эффективнее. При этом стоит отметить, что задача MAX-2-SAT, где все дизъюнкты имеют длину 1 или 2, уже является NP-трудной, хотя для неё существуют специальные алгоритмы, позволяющие решать её быстрее, чем в общем случае [9]. Тем не менее, если ограничения на максимальную длину дизъюнкта нет, но средняя длина небольшая, задача эффективно решается именно алгоритмом относительно длины формулы.

Условие задачи

Задача MAXSAT формулируется следующим образом:

MAXSAT	
Вход:	Булева формула F в конъюнктивной нормальной форме (КНФ) и число k
Ответ:	Означивание переменных, выполняющее хотя бы k дизъюнктов.

Длина формулы обозначается за L.

Как упоминалось выше, цель работы – создать алгоритм за $O^*(\alpha^L)$ при минимальном α. Однако, в процессе решения задачи выяснилось, что такой алгоритм проще построить для другой, уменьшенной меры $d = L - n_3$. Такая мера всегда не превосходит длины формулы L, таким образом, алгоритм, работающий за $O^*(\alpha^d)$ автоматически работает за $O^*(\alpha^L)$.

Алгоритм будет иметь следующую структуру:

1. Свести экземпляр задачи к экземпляру задачи $(n, 5)$-MAXSAT.

2. Свести экземпляр задачи к экземпляру задачи $(n, 3)$-MAXSAT.

3. Запустить на полученном экземпляре лучший известный алгоритм для $(n, 3)$-MAXSAT [3].

Уменьшенная мера позволяет использовать тот факт, что время работы алгоритма для задачи $(n, 3)$-MAXSAT гораздо меньше времени работы алгоритма в общем случае. За счёт увеличения награды за сведение переменной с большим количеством вхождений к 3-переменной достигается уменьшение чисел расщепления в правилах для переменных с большим количеством вхождений. При этом алгоритм для $(n, 3)$-MAXSAT относительно заданной меры работает за большее время, чем относительно длины, но это время всё ещё меньше времени работы алгоритма для общего случая.

Ограничения работы

Алгоритм для задачи булевой выполнимости (SAT), работающий за $O^*(1.074^L)$, был представлен Гиршем в 2000 году [10]. Несмотря на то, что целью работы является подойти ближе к этой границе, едва ли удастся её преодолеть, так как существование лучшего алгоритма для MAXSAT повлекло бы существование лучшего алгоритма для более простой задачи SAT. В частности, все худшие случаи представленного алгоритма в задаче булевой выполнимости разбирались бы тривиально.

В данной работе рассматривается лишь алгоритм относительно длины формулы, но не другие варианты параметризации задачи MAXSAT.

Определения ключевых терминов

Булевы переменные в работе обозначаются буквами x, y, z, w.

Если x — булева переменная, то выражения x и $\overline{x}$ называются литералами. Во избежание неоднозначности литералы в работе обозначаются буквами l, k, m.

Дизъюнкт — это дизъюнкция литералов, то есть выражение вида $x_1 \vee \overline{x_2} \vee x_3 \vee \ldots$. По умолчанию считается, что повторяющихся литералов в дизъюнкте нет, иначе их можно было бы сократить по правилу $l \vee l = l$. В работе дизъюнкты обозначаются буквами C, D, E.

Формула находится в конъюнктивной нормальной форме (КНФ), если она является конъюнкцией дизъюнктов (то есть имеет вид $(x_1 \vee \overline{x_2} \vee x_3) \wedge \overline{x}_1 \wedge \ldots)$.

Задача булевой выполнимости состоит в том, чтобы определить, существует ли означивание переменных, выполняющее формулу в КНФ. У неё есть вариант k-SAT с дополнительным ограничением на длину каждого дизъюнкта: она не больше k. В то время как задача 3SAT уже является NP-трудной, для задачи 2SAT известно полиномиальное решение.

Задача о максимальной выполнимости, как сформулировано выше, является оптимизационной версией этой задачи, и требует выполнения не всех, но хотя бы заданного числа дизъюнктов. У неё есть аналогичные варианты MAX-k-SAT. Кроме того, выделяют версии этой задачи (n, k)-MAXSAT, в которых каждая переменная входит в формулу не более k раз. При этом задачи MAX-2-SAT, $(n, 3)$-MAXSAT и даже $(n, 3)$-MAX-2-SAT уже являются NP-трудными [15]. Для задачи $(n, 2)$-MAXSAT существует полиномиальное решение.

Переменная называется k-переменной, если она входит в формулу ровно k раз. Если про переменную известно, что она входит в формулу хотя бы k раз, она называется k^+-переменной. Аналогично, если известно, что переменная входит в формулу не более k раз, она называется k^--переменной. Число k-переменных в формуле обозначается за n_k.

Если переменная x входит k раз положительно (то есть как литерал x) и l раз отрицательно (как литерал $\overline{x}$), она называется (k, l)-переменной. Такое же обозначение вводится и для литералов. Поскольку замена переменной x на $\overline{x}$ во всей формуле не влияет на ответ на задачу, если не указано иного, при обозначении переменных всегда считается $k \geq l$.

Алгоритм состоит из правил упрощения и правил расщепления.

Правило упрощения — полиномиальный алгоритм, преобразующий экземпляр задачи в эквивалентный ему и при этом не увеличивающий длину формулы. Такие правила применяются к формуле постоянно, пока это возможно. В правилах расщепления считается, что ни одно правило упрощения к формуле неприменимо.

Правило расщепления — полиномиальный алгоритм, преобразующий экземпляр задачи в несколько других вариантов и при этом с необходимостью уменьшающий длину формулы в каждом из них. От каждого из них алгоритм запускается рекурсивно, если хотя бы в одном из них удаётся получить положительный ответ, ответ на задачу объявляется положительным.

Если во вариантах, произведённых правилом расщепления, длина уменьшается на $a_1, \dots, a_k$, то время работы алгоритма оценивается как рекуррентное соотношение

$$T(L) = T(L - a_1) + \cdots + T(L - a_k)$$

Решением такого соотношения является $T(n) = O^*(\alpha^n)$, где α – единственный больший единицы корень уравнения

$$1 = \alpha^{-a_1} + \cdots + \alpha^{-a_k}$$

Вектор $(a_1, \ldots, a_k)$ называется вектором расщепления, а число α – числом расщепления. Асимптотика всего алгоритма оценивается как $O^*(\alpha^L)$, где α – максимальное по всем правилам расщепления число расщепления.

Расщепление по переменной x – правило расщепления, разделяющееся на случаи $x = 0$ и $x = 1$. Разумеется, такое правило исчерпывает все варианты и по ответам в каждом из вариантов можно восстановить ответ на исходную формулу.

Подформула – это подмножество дизъюнктов исходной формулы. Подформула называется замкнутой, если все переменные, литералы которых содержатся в подформуле, не имеют литералов вне этой подформулы.

Полученные результаты

Верхняя оценка на задачи максимальной выполнимости улучшена в данной работе с $O^*(1.106^L)$ до $O^*(1.093^L)$. В логарифмической шкале это даёт улучшение с $O^*(2^{0.145L})$ до $O^*(2^{0.128L})$, то есть на 11.7%.

Кроме того, в работе предложена новая мера для задачи MAXSAT с уменьшенной стоимостью 3-переменных. Продолжение исследований в этом направлении может помочь сдвинуть эту границу ещё дальше.

1. Обзор литературы

1.1. История развития области

Задача булевой выполнимости была исторически первой задачей, для которой была доказана NP-полнота (этот известный факт называется теоремой Кука-Левина). Задача о максимальной выполнимости, как оптимизационная версия этой задачи, автоматически является NP-полной и изучается с тех времён.

История развития точных алгоритмов для MAXSAT представлена в таблице 1.1.

Таблица 1.1: Развитие алгоритмов для задачи MAXSAT

Работа	**Год**	**Результат**	**Δ**
Нидермайер и Россманит [14]	1999	$O^*(1.1279^L) = O^*(2^{0.1737L})$	–
Банзал и Раман [2]	1999	$O^*(1.1057^L) = O^*(2^{0.1450L})$	16.5%

Видно, что со времён работы [2] значимого улучшения не произошло. В решении задачи в других мерах, связанных с количеством дизъюнктов при этом, как показано чуть ниже, существовал прогресс. Связано это с тем, что количество дизъюнктов меньше длины и относительно такой меры улучшение алгоритма выглядит для исследователей привлекательнее. В данном обзоре ниже показано, как развитие работ в этой области помогает в решении задачи относительно длины.

Одной из причин, затормозившей развитие алгоритма для задачи MAXSAT, стало недостаточное развитие смежного алгоритма для $(n, 3)$-MAXSAT.

Как будет продемонстрировано в данном обзоре чуть ниже, простые правила упрощения позволяют убрать из формулы 1- и 2-переменные, таким образом, можно считать, что все переменные в $(n, 3)$-MAXSAT-формуле являются 3-переменными, а длина формулы, таким образом, равняется утроенному количеству переменных. Одним из направлений исследований алгоритмов для задачи $(n, 3)$-MAXSAT является исследование алгоритмов, экспоненциальных относительно количества переменных в формуле. В силу равенства $L = 3n$ таковой алгоритм автоматически является и алгоритмом относительно длины. История этих алгоритмов приведена в таблице .

Таблица 1.2: Развитие алгоритмов для задачи $(n, 3)$-MAXSAT

Работа	**Год**	**Результат**
Раман, Равикумар и Рао [15]	1998	$O^*(1.732^n)$
Банзал и Раман [2]	1999	$O^*(1.3248^n)$
Куликов и Куцков [11]	2009	$O^*(1.2721^n)$
Близнец [6]	2013	$O^*(1.2600^n)$
Чэнь, Сюй и Ван [8]	2015	$O^*(1.237^n)$
Ли, Сюй, Ван и Ян [1]	2017	$O^*(1.194^n)$
Белова и Близнец [3]	2018	$O^*(1.191^n)$

Алгоритм, представленный Беловой и Близнецом в 2018 году, относительно длины формулы работает за $O^*(1.0600^L)$, что гораздо лучше, чем цель этой работы и подталкивает к введению уменьшённой меры. Отметим, однако, что с учётом введённой меры для достижения заявленной нами асимптотики с помощью введённой нами меры, необходимо существование алгоритма для $(n, 3)$-MAXSAT, работающего относительно количества переменных не хуже, чем за время $O^*(1.194^n)$.

В свою очередь, активное развитие алгоритмов для $(n, 3)$-MAXSAT стало возможным благодаря представленной Близнецом и Головнёвым [5] идеи сведения задачи к задаче о покрытии множества. Эта идея первоначально была применена к задаче о максимальной выполнимости, параметризованной ответом. История работ, связанных с этой формулировкой задачи, приведены в таблице 1.3.

Таблица 1.3: Развитие алгоритмов для MAXSAT относительно ответа

Работа	**Год**	**Результат**	$\boldsymbol{\Delta}$
Махаджан и Раман [13]	1999	$O^*(1.618^k) = O^*(2^{0.695k})$	–
Нидермайер и Россманит [14]	1999	$O^*(1.400^k) = O^*(2^{0.486k})$	30%
Банзал и Раман [2]	1999	$O^*(1.381^k) = O^*(2^{0.466k})$	4%
Чэнь и Кандж [7]	2004	$O^*(1.370^k) = O^*(2^{0.455k})$	2.5%
Близнец и Головнёв [5]	2012	$O^*(1.358^k) = O^*(2^{0.442k})$	2.8%
Чэнь и Сюй [8]	2015	$O^*(1.325^k) = O^*(2^{0.406^k})$	8%

Отдельно хочется отметить развитие алгоритмов для задачи о максимальной выполнимости, параметризованной общим количеством дизъюнктов. Развитие алгоритмов для этой задачи представлено в таблице 1.4.

Таблица 1.4: Развитие алгоритмов для задачи MAXSAT относительно количества дизъюнктов

Работа	**Год**	**Результат**	**Δ**
Нидермайер и Россманит [14]	1999	$O^*(1.3803^m) = O^*(2^{0.465m})$	–
Банзал и Раман [2]	1999	$O^*(1.3413^m) = O^*(2^{0.424m})$	8.9%
Чэнь и Кандж [7]	2004	$O^*(1.3248^m) = O^*(2^{0.406m})$	4.2%
Сюй, Ли, Ян, Чэнь и Ван [16]	2019	$O^*(1.2989^m) = O^*(2^{0.378m})$	7%

Лучший результат в этой области $O^*(1.2989^m)$ был получен Сюй и др. [16] в 2019 году. В то время как эта работа также использует идею сведения к задаче о покрытии множества, там также введено большое количество новых правил упрощения. В то время как большая их часть увеличивает длину формулы (и, следовательно, не может быть применена к рассматриваемой задаче), одно из них оказывается весьма полезным. Подробнее об этом рассказано в данном обзоре ниже.

1.2. Правила упрощения

Правила упрощения – одно из самых мощных средств построения алгоритмов благодаря тому, что многие из них можно переиспользовать для разных версий одной и той же задачи. В данном разделе представлены правила упрощения из литературы, используемые в представленном в работе алгоритме.

Отметим, что, несмотря на то, что формальная постановка задачи о максимальной выполнимости подразумевает, что во входных данных содержится требуемое количество выполненных дизъюнктов k, для краткости это количество будет опускаться. Это возможно благодаря тому, что представленный алгоритм сразу строит означивание, выполняющее наибольшее возможное количество дизъюнктов, игнорируя число k до момента ответа.

Правило упрощения 1.1. *Если для переменной x оба литерала x и $\overline{x}$ содержатся в одном дизъюнкте $x \vee \overline{x} \vee C$, то можно удалить этот дизъюнкт.*

Это правило корректно, поскольку выражение $x \vee \overline{x}$ верно при любом означивании переменных, и, следовательно, дизъюнкт $x \vee \overline{x} \vee C$ выполняется всегда. Удаление дизъюнкта не увеличивает длины формулы.

Правило является очевидным и относится больше к вопросу формального определения формулы, нежели к построению алгоритма. Так, в работе [2] это правило опускается, так как определение понятия дизъюнкта, используемое там, исключает возможность ситуации, в которой такое правило применимо.

Правило упрощения 1.2. *Пусть в формуле содержится литерал l такой, что литерал $\bar{l}$ в формуле отсутствует. Тогда можно означить $l = 1$.*

Правило снова является очевидным: при переходе от означивания $l = 0$ к означиванию $l = 1$ ни один дизъюнкт не перестаёт быть выполненным, а хотя бы один новый дизъюнкт, напротив, становится выполненным. В силу очевидности не вполне корректно приводить ссылки на конкретную работу, где оно введено.

Правило упрощения 1.3 (Almost common clauses, [2])**.** *Пусть для некоторой переменной x и (возможно, пустого) дизъюнкта C оба дизъюнкта $x \vee C$ и $\overline{x} \vee C$ входят в формулу. Тогда оба этих дизъюнкта можно заменить на один дизъюнкт C.*

Это правило встречается в работе [2]. Вариация с пустым C была известна и ранее.

Замечание. В частности, после того, как правило 1.3 неприменимо, для каждой переменной x лишь один из литералов x и $\overline{x}$ может входить в дизъюнкты длины 1.

Правило упрощения 1.4 (Правило резолюций)**.** *Пусть* x – $(1,1)$-*переменная, входящая в дизъюнкты* $x \vee C$ *и* $\overline{x} \vee D$*. Тогда оба этих дизъюнкта можно заменить на один дизъюнкт* $C \vee D$*.*

Это правило имеет истоки в пропозициональной логике и было, по видимому, в каком-то виде известно ещё до формулировки задачи булевой выполнимости. В представленном виде оно верно и для задачи о максимальной выполнимости.

Замечание. После того, как правила 1.2 и 1.4 неприменимы, в формуле нет 2^--переменных, так как все эти переменные элиминируются одним из указанных правил. В частности, задача $(n, 2)$-MAXSAT решается за полиномиальное время.

Правило упрощения 1.5 ([14])**.** *Пусть* l – (i, j)*-литерал, входящий в* k *дизъюнктов длины 1, причём* $k \geq j$*. Тогда можно назначить* $l = 1$*.*

Мощность этого правила заключается в том, что оно ограничивает сверху количество дизъюнктов длины 1, в которое может входить переменная, и, следовательно, ограничивает снизу суммарную длину дизъюнктов, в которые переменная входит. Кроме того, это правило существенно уменьшает разбор случаев.

Правило упрощения 1.6 (Правило 9 из [16])**.** *Пусть* $i \geq 2$ *и* x – $(i, 1)$*-переменная, такая, что все дизъюнкты, содержащие* x*, содержат также один и тот же литерал* l*. Тогда можно убрать* l *из всех этих дизъюнктов и добавить его в дизъюнкт, содержащий* $\overline{x}$*. То есть* $(x \vee l \vee C_1) \wedge \cdots \wedge (x \vee l \vee C_i) \wedge (\overline{x} \vee D) \wedge F' \to (x \vee C_1) \wedge \cdots \wedge (x \vee C_i) \wedge (\overline{x} \vee l \vee D) \wedge F'$*.*

Это правило даёт очень мощные ограничения на 3-переменные (каждая из которых после правила 1.2 является $(2, 1)$-переменной), особенно в сочетании с правилом 1.3. Такое сочетание во многих случаях будет ограничивать количество литералов одной и той же 3-переменной в наборы дизъюнктов одним вхождением.

Правило упрощения 1.7. *Пусть в формуле есть замкнутая подформула на не более чем пяти переменных. Тогда эту подформулу можно решить за полиномиальное время независимо от остальной части формулы.*

Заметим, что замкнутые подформулы можно находить за полиномиальное время, построив граф с вершинами – переменными из формулы и рёбрами, если концы входят в один дизъюнкт, и найдя в нём компоненты связности. Также заметим, что замкнутые формулы на константном количестве переменных решаются за полиномиальное время выбором произвольной переменной и расщеплением по ней: размер дерева рекурсии получается константным и в каждой вершине выполняется полиномиальное действие.

1.3. Выводы

- Работа над задачей остановилась на работе [2] из-за отсутствия алгоритмов для задачи $(n, 3)$-MAXSAT с достаточно хорошим временем работы. Такой алгоритм был получен в работах [1] и [3].
- Этому способствовала работа в других параметризациях задачи о максимальной выполнимости, в частности, идея о сведении к задаче о покрытии множества, высказанная в работе [5].
- Также современные правила упрощения, такие, как правило 1.6, позволяют сильно уменьшить пространство разбираемых случаев.

2. Уменьшенная мера

2.1. Мотивация

При разработке алгоритма для задачи $(n, 4)$-MAXSAT становится заметным дисбаланс между 3-переменными и 4-переменными: при относительно небольшой разнице в стоимости этих переменных в мере длины свойства этих переменных заметно различаются. Эта разница заключается в следующем.

Правила редукции 1.2 и 1.4 позволяют избавляться от 2^--переменных. Благодаря этому при удалении одного литерала у такой переменной моментально элиминируются и два других литерала. Благодаря этому и достигается время работы алгоритма для $(n, 3)$-MAXSAT, значительно меньшее времени работы в общем случае.

В то же время удаление одного литерала 4-переменной делает из неё 3-переменную. Для полной элиминации 4-переменной необходимо удалить из неё как минимум два литерала.

В работе [2], представляющей лучший до сих пор алгоритм для общего случая задачи о максимальной выполнимости, одним из худших случаев является случай с 4-переменной с большим количеством соседей, также являющихся 4-переменными. Хотя небольшое улучшение получить в этом случае возможно, разбор такого случая до целей, поставленных в этой работе, всё ещё представляет сложность.

Уменьшенный параметр, введённый ниже, позволяет бороться с этим дисбалансом. Практика показывает, что он оказывается также полезным для переменных с бóльшим количеством вхождений в формулу.

2.2. Определение

Определение 2.1. Уменьшенной мерой формулы F называется величина $d = L - n_3$.

Поскольку n_3 является неотрицательной величиной, для любой формулы $d \leq L$. Таким образом, любой алгоритм, работающий за $O^*(\alpha^d)$, автоматически работает за $O^*(\alpha^L)$. Следовательно, достаточно существования алгоритма за $O^*(\alpha^d)$. В дальнейшем в работе строится алгоритм относительно именно уменьшенной меры.

Отметим, что такую меру можно записать в следующем виде:

$$d = L - n_3 = \sum_{k \geq 3} k n_k - n_3 = 2n_3 + \sum_{k \geq 4} k n_k \tag{1}$$

Для задач $(n,4)$-MAXSAT и $(n,5)$-MAXSAT эта запись имеет вид $d = 2n_3 + 4n_4$ и $d = 2n_3 + 4n_4 + 5n_5$, соответственно. В частности, для мотивации, представленной в разделе 2.1, видно, что при удалении одного литерала у 4-переменной эта мера уменьшается на 2, то есть на столько же, на сколько она уменьшается при удалении одного литерала у 3-переменной. Таким образом, происходит выравнивание свойств 3- и 4-переменных и становится возможным придумать для 4-переменных правила расщепления с временем работы, почти не отличающимся от времени работы алгоритма для $(n,3)$-MAXSAT относительно этой меры.

Касательно последнего, в статье Беловой и Близнеца [3] указана асимптотика получившегося алгоритма $O^*(1.191^{n_3})$, что соответствует времени $O^*(1.0912^{2n_3}) = O^*(1.0912^d)$. В статье худшим вектором расщепления указан вектор $(2,7)$, соответствующий, в терминах d, вектору $(4,14)$.

2.3. Свойства

Важнейшим свойством уменьшенной таким образом меры является то, что в очень многих случаях правила расщепления дают не худший вектор расщепления, чем в мере длины. В частности, такое можно доказать для обычного расщепления по переменной. Это и является утверждением леммы 2.1.

Лемма 2.1. *Пусть x – (i,j)-переменная ($i+j \geq 4$) в формуле*

$$F = (x \vee C_1) \wedge \ldots \wedge (x \vee C_i) \wedge (\overline{x} \vee D_1) \wedge \ldots \wedge (\overline{x} \vee D_j) \wedge F'$$

Тогда расщепление по переменной x даёт относительно меры d вектор расщепления не хуже, чем $(j + \sum_{k=1}^{i} |C_k|, i + \sum_{k=1}^{j} |D_k|)$.

Доказательство. В случае $x = 1$ из формулы убираются все отрицательные вхождения x (как ложные они не влияют на выполнение дизъюнктов, в которые они входят) и все дизъюнкты с положительным вхождением x (как уже выполненные). Аналогично, в случае $x = 0$ из формулы убираются все положительные вхождения x и все дизъюнкты с отрицательным вхождением x. Таким образом, нашей задачей является доказать, что уменьшенная мера после применения правил упрощения изменяется не меньше, чем длина формулы до применения этих правил.

В первую очередь, отметим, что никакие правила редукции, введённые до текущего момента или после, не вводят новых переменных, но лишь убирают литералы существующих или же склеивают дизъюнкты (как правило 1.4) или иногда добавляют новые литералы, уменьшая при этом их общее количество (такие как правило 1.6). Таким образом, разницу уменьшенных мер формул можно посчитать как сумму разниц весов переменных в этих формулах.

В мере длины за каждый элиминированный литерал мера уменьшается на 1.

В уменьшенной мере за каждый элиминированный литерал 4^+-переменной длина уменьшается хотя бы на 1. При сведении же 4^+-переменной к 2^--переменной полученная переменная моментально элиминируется правилами 1.4 или 1.2. Таким образом, уменьшение числа литералов k-переменной на t уменьшает вес переменной хотя бы на t, достигая значения t при $t \leq k - 4$ или $t = k$.

Таким образом, для 4^+-переменных уменьшение веса в уменьшенной меры не меньше уменьшения веса в длине формулы. Поскольку x из условия леммы является 4^+-переменной, сказанное относится и к нему.

Остался случай 3-переменных. Для них при уменьшении количества литералов вес всегда уменьшается на 2, но в случае элиминации всех 3 литералов длина изменяется на 3. Однако, если все три литерала входили в дизъюнкты C_i или D_i, то к такой 3-переменной применимо правило упрощения 1.6. Такого быть не могло, а значит, у 3-переменных не могло элиминироваться больше двух литералов.

Таким образом, в каждом из случаев уменьшенная мера изменяется после применения правил упрощения не меньше, чем длина до применения этих правил, что и требовалось доказать. □

Оказывается, этого утверждения, вместе с объявленными выше правилами упрощения, достаточно для разбора 6^+-переменных.

Правило расщепления 2.1. *Если в формуле есть* 6^+*-переменная* x*, расщепиться по ней.*

Это даёт как минимум $(6, 10)$*-расщепление.*

Доказательство. Основная идея — доказать, что переменная входит в небольшое количество дизъюнктов длины 1, и воспользоваться леммой 2.1.

Пусть x – (i, j)-переменная (не умаляя общности, $i \geq j$). Во-первых, в силу правила сокращения 1.3, лишь один из литералов x и $\overline{x}$ может входить в дизъюнкты длины 1. Соответственно, рассмотрим два случая: когда x входит в дизъюнкты длины 1 и когда не входит.

Если x входит в дизъюнкты длины 1, то в силу правила сокращения 1.5 таких дизъюнктов может быть не более $j - 1$, и, таким образом, по лемме 2.1 вектор расщепления у нас как минимум $(2(i - j + 1) + j - 1 + j, 2j + i) = (2i + 1, 2j + i)$.

Если x не входит в дизъюнкты длины 1, то лемма 2.1 даёт расщепление хотя бы $(2i + j, i + j)$. Более того, в случае $i = j$ в силу правила сокращения 1.5 все вхождения литерала $\overline{x}$ не могут быть в дизъюнктах длины 1. Следовательно, в случае $i = j$ у нас вектор расщепления не хуже, чем $(3i, 2i + 1)$.

Если переменная является 8^+-переменной, расщепление по ней даёт вектор как минимум $(8, 8)$ (по восемь литералов такой переменной элиминируются в каждом случае). Такой вектор уже лучше вектора $(6, 10)$.

Для 6-переменных и 7-переменных вектора, полученные оценками выше, можно найти в таблице 2.1.

Таблица 2.1: Оценочные вектора расщепления для 6- и 7-переменных

(i, j)	Первый случай		Второй случай	
	Вектор	Число	Вектор	Число
$(6, 1)$	$(13, 8)$	1.0697	$(13, 7)$	1.0743
$(5, 2)$	$(11, 9)$	1.0721	$(12, 7)$	1.0777
$(4, 3)$	$(9, 10)$	1.0758	$(11, 7)$	1.0816
$(5, 1)$	$(11, 7)$	1.0816	$(11, 6)$	1.0878
$(4, 2)$	$(9, 8)$	1.0851	$(10, 6)$	1.0927
$(3, 3)$	$(7, 9)$	1.0911	$(9, 7)$	1.0911

Отметим, что второй подслучай второго случая возникает в таблице ишь для $(3, 3)$-переменных

Видно, что в таблице худшим вектором является вектор $(6, 10)$ во втором случае для $(4, 2)$-переменных. Таким образом, он же является худшим и для всех 6^+-переменных. $\square$

Хочется заметить, что такое утверждение верно и для длины формулы, не только для уменьшенной меры.

Это правило расщепления хоть и не встречается в работе [2] в точно таком же виде, но по сути является комбинацией нескольких однотипных правил, встречающихся там.

2.4. Выводы

- Предложена новая уменьшенная мера $d = L - n_3$, позволяющая сгладить разницу свойств 3-переменных и остальных переменных.

- Продемонстрировано, что для простого расщепления по 4^+-переменной в мере d вектор расщепления получается не хуже, чем для длины формулы (лемма 2.1).

- Введено правило расщепления для 6^+-переменной с вектором расщепления $(6, 10)$ в худшем случае.

3. Разбор 5-переменных

3.1. Общие наблюдения

Итак, после того, как правило расщепления 2.1 неприменимо, остался экземпляр задачи $(n, 5)$-MAXSAT. В данном разделе представлен разбор случаев, в совокупности позволяющих избавиться от 5-переменных. Прежде всего, несколько наблюдений.

Во-первых, не умаляя общности, 5-переменная может быть $(4, 1)$-переменной или $(3, 2)$-переменной. В силу правила сокращения 1.5, в первом случае переменная может встречаться в дизъюнктах длины 1 лишь отрицательно, во втором случае – либо дважды отрицательно, либо однажды положительно или отрицательно. Каждый из этих случаев разобран в разделах этой главы.

Во-вторых, вспомним доказательство леммы 2.1. В доказательстве изменение уменьшенной длины формулы считалось как сумма изменений весов всех переменных. Для 5-переменных оказывается, что случаев переменных-соседей немного. Все эти случаи приведены в таблице 3.1. В первом столбце перечислено количество вхождений во всю формулу. Во втором столбце перечислено изменение длины до применения правил сокращения, то есть количество литералов этой переменной, которые могут одновременно иметь соседями один и тот же литерал переменной, по которой мы пытаемся расщепиться. В третьем столбце приведено изменение веса этой переменной в уменьшенной длине в таком случае.

В таблице отсутствуют 3-переменные, встречающиеся среди соседей такой переменной трижды, поскольку, как указано в доказательстве леммы 2.1, правило сокращения 1.6 позволяет сократить такие случаи. Также в таблице не указаны 5-переменные, встречающиеся пятикратно, так как пять литералов такой переменной должны были бы иметь соседями один и тот же литерал, что невозможно, поскольку в силу правила сокращения 1.2 в $(n, 5)$-MAXSAT формуле каждый литерал встречается не более чем четырежды.

Таблица 3.1: Случаи переменных-соседей для $(n, 5)$-MAXSAT

Тип переменной	ΔL	Δd
3-переменная	1	2
	2	2
4-переменная	1	2
	2	4
	3	4
	4	4
5-переменная	1	1
	2	3
	3	5
	4	5

Лемма 2.1 фактически утверждает, что $\Delta L \le \Delta d$. Для $(n, 5)$-MAXSAT можно заметить к тому же, что почти во всех строках $\Delta L < \Delta d$. Равенство достигается лишь для 3-переменных, встречающихся дважды, 4-переменных, встречающихся четырежды, или 5-переменных, встречающихся однажды. Более того, второй из этих трёх случаев встречается только при разборе $(4, 1)$-переменных, но не при разборе $(3, 2)$-переменных.

Таким образом, верно неформальное усиление леммы 2.1: если среди соседей есть переменные, не являющиеся одним из этих трёх (а после разбора $(4, 1)$-переменных двух) случаев, то уменьшенная мера уменьшается даже больше, чем длина. Это позволяет строить эффективные правила расщепления для 5-переменных.

Такое утверждение позволяет разобрать огромное количество крайних случаев. В общем же случае может оказаться, что все соседи какой-то переменной – 5-переменные, встречающиеся там однажды. В таком случае можно этим воспользоваться и вывести правило расщепления, означивающее одновременно многих соседей. Это правило формально закреплено в двух следующих леммах.

Лемма 3.1. *Пусть* x *–* $(3, 2)$*-переменная следующего вида:*

$$(x \vee C_1) \wedge (x \vee C_2) \wedge (x \vee C_3) \wedge (\overline{x} \vee D_1) \wedge (\overline{x} \vee D_2) \wedge F'$$

Пусть при этом $|C_1| = 1$, а C_2 и C_3 непусты. Тогда, если существует означивание переменных, одновременно выполняющее C_2 и C_3 и не выполняющее C_1, D_1 и D_2, расщепление на следующие три случая корректно:

1. $x = 1$
2. $x = 0$, $C_1 = 1$
3. $x = 0$, $C_1 = 0$, $C_2 = C_3 = 1$, *и, если для какого-то i дизъюнкт D_i непуст,* $D_i = 0$

Доказательство. Во-первых, докажем, что существует оптимальное означивание, в котором или x равен единице, или два из трёх дизъюнктов C_i выполнены.

Рассмотрим какое-то оптимальное означивание. Пусть в нём $x = 0$ и из C_i выполнено не более одного дизъюнкта. Тогда всего из дизъюнктов, в которых встречается x, выполнено не более трёх. Означив в таком случае $x = 1$, мы из этих пяти дизъюнктов выполним хотя бы столько же, и не изменим значения других дизъюнктов.

Более того, если в оптимальном означивании хотя бы один из D_i равен единице, то утверждение усиливается до "все C_i должны быть выполнены" в силу того, что означивание $x = 1$ выполняет хотя бы четыре из пяти дизъюнктов.

Таким образом, существует оптимальное означивание, в котором выполняется одно из трёх утверждений: либо $x = 1$, либо $x = 0$ и $C_1 = 1$, либо, если $x = 0$ и $C_1 = 0$, то C_2 и C_3 равны единице. Более того, поскольку в последнем случае выполняются два из трёх C_i, в таком случае обязательно все непустые D_i равны нулю. □

Лемма 3.2. *Пусть x – $(3, 2)$-переменная такого же вида, как в лемме 3.1, где $|C_1| = 1$, а C_2 и C_3 непусты. Пусть при этом не существует означивания переменных, одновременно выполняющего C_2 и C_3 и не выполняющего C_1, D_1 и D_2. Тогда расщепление на следующие три случая корректно:*

1. $x = 1$

2. $x = 0$, $C_1 = 1$

Доказательство. В лемме 3.1 доказано, что существует оптимальное означивание переменных, удовлетворяющее одному из трёх случаев. В данной лемме дополнительно в условиях указана невозможность существования означивания, удовлетворяющего третьему из них. Значит, остаются первые два. □

Как показано ниже, двух правил расщепления из лемм 3.1 и 3.2 в совокупности с расщеплением по переменным и несколькими правилами сокращения оказывается достаточно.

3.2. Разбор $(4, 1)$-переменных

В первую очередь докажем общие правила упрощения и расщепления для $(i, 1)$-переменных.

Правило упрощения 3.1. *Пусть x – $(i, 1)$-переменная $(i \geq 2)$ в формуле вида*

$$(x \vee C_1) \wedge \ldots \wedge (x \vee C_i) \wedge (\overline{x} \vee D) \wedge F'$$

Пусть для какого-то j выполняется $|C_j| = 1$, и литерал оттуда присутствует в D. Тогда можно назначить $x = 1$.

Доказательство. Обозначим $C_j = l$.

Если в оптимальном назначении $l = 1$, то D выполнен, и, таким образом, поскольку единственный дизъюнкт с $\overline{x}$ выполнен, назначение $x = 1$ выполняет не меньше дизъюнктов в таком означивании, чем $x = 0$.

Если же $l = 0$, и при этом $x = 0$, то из дизъюнктов с переменной x выполнено в таком означивании не более i, в то время как назначение в этом означивании $x = 1$ выполнит хотя бы i. □

Лемма 3.3. *Пусть x – $(i,1)$-переменная ($i \geq 2$) в формуле вида*

$$(x \vee C_1) \wedge \ldots \wedge (x \vee C_i) \wedge (\overline{x} \vee D) \wedge F'$$

Тогда расщепление на следующие два случая корректно:

1. $x = 1$

2. $x = 0$, *если D непусто, $D = 0$, и, если для какого-то j выполняется $|C_j| = 1$, то $C_j = 1$*

Доказательство. Покажем, что существует оптимальное означивание, в котором либо $x = 1$, либо $x = 0$ и тогда все C_j выполнены, а D не выполнено.

Рассмотрим какое-то оптимальное означивание. Пусть в нём $x = 0$, но один из C_j не выполнен. Тогда из всех дизъюнктов, содержащих x, выполнены не более i. Тогда при означивании $x = 1$ из этих дизъюнктов будет выполнено хотя бы i, а значения других дизъюнктов не изменяются. Таким образом, мы получили оптимальное означивание, в котором $x = 1$.

Теперь пусть в оптимальном означивании $x = 0$ и все C_j выполнены, но также выполнен D. Тогда при означивании $x = 1$ выполнимость ни одного дизъюнкта не изменится. Таким образом, мы снова получили оптимальное означивание, в котором $x = 1$. □

Замечание. Отметим, что такое назначение всегда выполнимо. А именно, если два C_j имеют длину 1 и содержат противоположные дизъюнкты одной и той же переменной, применимо правило упрощения 1.3. Также, если C_j имеет длину 1, то этот литерал не может содержаться в D в силу правила **??**.

Мы обозначим $(4,1)$-переменную x также, как в формулировке леммы 3.3:

$$(x \vee C_1) \wedge (x \vee C_2) \wedge (x \vee C_3) \wedge (x \vee C_4)$$

Здесь, как указано выше, все C_i непусты, а D может быть пустым.

Воспользуемся только что доказанной леммой.

Правило расщепления 3.2. *Если* x – $(4,1)$*-переменная, расщепиться по правилу из леммы 3.3.*

Это даёт как минимум $(7,9)$*-расщепление.*

Доказательство. Во-первых, если $|D| > 0$, это хотя бы $(7,9)$-расщепление. В первом случае длина уменьшается хотя бы на 9 (пять литералов x и как минимум четыре литерала в C_j), а значит, по лемме 2.1, уменьшенная длина уменьшается хотя бы на 9. Во втором случае мы убираем 5-переменную x и 3^+-переменную из D, таким образом уменьшая меру d как минимум на 7.

Во-вторых, если $|D| = 0$, но существует такое j, что $|C_j| = 1$, то это точно также хотя бы $(7,9)$-расщепление: во втором случае мы убираем переменную не из D, а из C_j, а в остальном доказательство повторяет предыдущий абзац.

Наконец, если $|D| = 0$ и все $|C_j| > 1$, то $\sum_j |C_j| \geq 8$. Тогда по лемме 2.1 это хотя бы $(5,13)$-расщепление.

Так как вектор $(7,9)$ хуже, он и является оценкой для худшего случая. $\square$

После того, как указанные правила неприменимы, в формуле не осталось $(4,1)$-переменных.

Список литературы

1. An improved branching algorithm for (n, 3)-MaxSAT based on refined observations / W. Li [et al.] // International Conference on Combinatorial Optimization and Applications. — Springer. 2017. — P. 94–108.
2. *Bansal N.*, *Raman V.* Upper bounds for MaxSat: Further improved // International symposium on algorithms and computation. — Springer. 1999. — P. 247–258.
3. *Belova T.*, *Bliznets I.* Upper and Lower Bounds for Different Parameterizations of (n, 3)-MAXSAT // International Conference on Combinatorial Optimization and Applications. — Springer. 2018. — P. 299–313.
4. *Berg J.*, *Hyttinen A.*, *Järvisalo M.* Applications of MaxSAT in data analysis // Pragmatics of SAT. — 2015.
5. *Bliznets I.*, *Golovnev A.* A new algorithm for parameterized MAX-SAT // International Symposium on Parameterized and Exact Computation. — Springer. 2012. — P. 37–48.
6. *Bliznets I. A.* A new upper bound for (n, 3)-MAX-SAT // Journal of Mathematical Sciences. — 2013. — Vol. 188, no. 1. — P. 1–6.
7. *Chen J.*, *Kanj I. A.* Improved exact algorithms for Max-Sat // Discrete Applied Mathematics. — 2004. — Vol. 142, no. 1–3. — P. 17–27.
8. *Chen J.*, *Xu C.*, *Wang J.* Dealing with 4-variables by resolution: an improved MaxSAT algorithm // Workshop on Algorithms and Data Structures. — Springer. 2015. — P. 178–188.
9. *Golovnev A.*, *Kutzkov K.* New exact algorithms for the 2-constraint satisfaction problem // Theoretical Computer Science. — 2014. — Vol. 526. — P. 18–27.
10. *Hirsch E. A.* New worst-case upper bounds for SAT // Journal of Automated Reasoning. — 2000. — Vol. 24, no. 4. — P. 397–420.

11. *Kulikov A. S.*, *Kutskov K.* New upper bounds for the problem of maximal satisfiability // Discrete Mathematics and Applications. — 2009. — Vol. 19, no. 2. — P. 155–172.

12. *Lin P.-C. K.*, *Khatri S. P.* Application of Max-SAT-based ATPG to optimal cancer therapy design // BMC genomics. — 2012. — Vol. 13, S6. — S5.

13. *Mahajan M.*, *Raman V.* Parameterizing above guaranteed values: MaxSat and MaxCut // J. Algorithms. — 1999. — Vol. 31, no. 2. — P. 335–354.

14. *Niedermeier R.*, *Rossmanith P.* New upper bounds for MaxSat // International Colloquium on Automata, Languages, and Programming. — Springer. 1999. — P. 575–584.

15. *Raman V.*, *Ravikumar B.*, *Rao S. S.* A simplified NP-complete MAXSAT problem // Information Processing Letters. — 1998. — Vol. 65, no. 1. — P. 1–6.

16. Resolution and domination: an improved exact MaxSAT algorithm / C. Xu [et al.] // Proceedings of the 28th International Joint Conference on Artificial Intelligence. — AAAI Press. 2019. — P. 1191–1197.