Math-Net.Ru

Общероссийский математический портал

А. С. Куликов, К. Куцков, Новые верхние оценки для задачи максимальной выполнимости, *Дискрет. матем.*, 2009, том 21, выпуск 1, 139–157

DOI: <https://doi.org/10.4213/dm1043>

Использование Общероссийского математического портала Math-Net.Ru подразумевает, что вы прочитали и согласны с пользовательским соглашением
<http://www.mathnet.ru/rus/agreement>

Параметры загрузки:

IP: 3.93.64.190

26 апреля 2023 г., 15:16:19

Новые верхние оценки для задачи максимальной выполнимости

© 2009 г. А. С. Куликов, К. Куцков

В работе приводятся сравнительно простые доказательства следующих новых верхних оценок для задач максимальной выполнимости:

c^N , где константа $c < 2$, а N — число переменных, для задачи MAX-SAT для формул константной плотности;

$2^{K/6}$, где K — число кловов, для задачи MAX-2-SAT;

$2^{N/6,7}$ для задачи $(n, 3)$ -MAX-2-SAT.

Все оценки доказаны методом расщепления. Основными используемыми идеями являются техника запоминания кловов и использование комбинированных мер сложности.

Работа первого автора поддержана Российским фондом фундаментальных исследований, гранты 06-01-00502а и 08-01-00640а, и Фондом содействия отечественной науке.

1. Введение

1.1. Задачи MAX-SAT и MAX-2-SAT

Задача максимальной выполнимости (MAX-SAT) заключается в нахождении по данной формуле в конъюнктивной нормальной форме максимального числа ее кловов, которые могут быть выполнены одновременно. Вариант задачи MAX-SAT, в котором требуется ответить, можно ли выполнить заданное число кловов входной формулы, является NP-полным даже для случая, когда каждый клов входной формулы содержит не более двух литералов — данный частный случай является задачей MAX-2-SAT. Более того, известно [19], что задача остается NP-полной, даже если каждая переменная заданной формулы входит не более чем в три клова длины 2 (такой частный случай называется $(n, 3)$ -MAX-2-SAT). Задача MAX-SAT является одной из важнейших оптимизационных задач. Многие практические и теоретические задачи могут быть сформулированы в терминах MAX-SAT. Например, NP-трудные задачи о максимальном разрезе, о минимальном вершинном покрытии и о максимальном независимом множестве легко сводятся к MAX-2-SAT.

1.2. Новые верхние оценки

В табл. 1 приводятся известные оценки для трех рассматриваемых в данной работе задач относительно стандартных мер сложности, а именно: для числа переменных N и числа

Таблица 1. Лучшие известные верхние оценки

	MAX-SAT	MAX-2-SAT	$(n, 3)$ -MAX-2-SAT
N	2^N	$2^{N/1,261}$ [24]	$2^{N/6,53}$ [21]
K	$2^{K/2,465}$ [2]	$2^{K/5,88}$ [12]	$2^{K/9,795}$ [21]

Таблица 2. Известные верхние оценки для MAX-2-SAT

$2^{m/2,873}$	[16]	Нидермайер, Розманит
$2^{m/4,000}$	[7]	Гирш
$2^{m/5,000}$	[6]	Грамм, Гирш, Нидермайер, Розманит
$2^{m/5,000}$	[22]	Скотт, Соркин
$2^{m/5,217}$	[9]	Кнейс, Розманит
$2^{m/5,500}$	[10]	Кожевников, Куликов
$2^{m/5,769}$	[8]	Кнейс, Молле, Рихтер, Розманит
$2^{m/5,769}$	[21]	Скотт, Соркин
$2^{m/5,88}$	[12]	Куликов, Куцков

клов K . Практически все алгоритмы табл. 1 основаны на методе расщепления, который не может быть применен для получения нетривиальных оценок относительно числа переменных. В [24] интересным способом используется быстрое умножение матриц для доказательства оценки $2^{N/1,261}$. В табл. 2 также перечислены предыдущие известные результаты для MAX-2-SAT. Отметим, что оценки из [22, 21] применимы, на самом деле, к более широкому классу задач, а именно к MAX-2-CSP. Известна только одна оценка лучше, чем 2^n , для MAX-SAT на формулах константной плотности. Алгоритм был представлен в [3]. Данный алгоритм, однако, использует экспоненциальную память.

В данной работе мы приводим сравнительно простые доказательства следующих верхних оценок:

c^N , где константа $c < 2$, а N — число переменных, для задачи MAX-SAT для формул константной плотности;

$2^{K/6}$, где K — число клов, для задачи MAX-2-SAT;

$2^{N/6,7}$ для задачи $(n, 3)$ -MAX-2-SAT.

Все перечисленные алгоритмы используют полиномиальную память. Отметим, что из полученных результатов следует верхняя оценка $2^{N/6,7}$, где N — число вершин графа, для задачи о максимальном независимом множестве в 3-регулярном графе, поскольку последняя легко сводится к $(n, 3)$ -MAX-2-SAT (см. [17]).

1.3. Основные идеи новых алгоритмов

Метод расщепления является одним из самых популярных методов доказательства верхних оценок для SAT и MAX-SAT. Простейшая форма алгоритма расщепления представлена на рис. 1. Очевидно, что без фазы упрощения время работы такого алгоритма в худшем случае есть 2^N , так как он просто перебирает всех кандидатов (то есть наборы значений всем переменным формулы) на решение.

Алгоритм расщепления.**Вход:** формула F в КНФ.**Метод.**

- (1) Упрощать F , пока возможно.
- (2) Если решение для F очевидно (в частности, если F не содержит кловов), тогда вернуть его.
- (3) Выбрать переменную v в соответствии с некоторой эвристикой.
- (4) Произвести рекурсивные вызовы для $F[v]$ и $F[\neg v]$ и вернуть решение, основанное на решениях для обоих рекурсивных вызовов.

Рис. 1. Форма алгоритма расщепления

Естественной идеей для уменьшения времени работы такого алгоритма является введение следующего частичного порядка на кандидатах: α сильнее β , если β может быть решением только в случае, когда и α является решением. Ясно, что в такой ситуации нет необходимости рассматривать набор β , и поэтому пространство поиска сужается. На такой простой идее основано большинство известных алгоритмов расщепления. Например, если l — чистый литерал, то любой набор $\alpha \ni l$ сильнее набора $(\alpha \setminus \{l\}) \cup \{\neg l\}$. Таким образом, для доказательства того, что алгоритм расщепления является корректным, нужно показать, что любой возможный кандидат на решение либо будет рассмотрен алгоритмом, либо слабее какого-то кандидата, который будет рассмотрен. Однако такое знание, как правило, используется в одной и той же ветке дерева расщепления: если в какой-то ветке алгоритм не рассматривает β , то в той же ветке есть набор α , который сильнее β .

В данной работе мы показываем, как информация о кандидатах, рассмотренных в одной ветке, может быть использована в другой ветке. Сама по себе идея довольно естественна и не является новой: она используется, например, в теоретических алгоритмах [23, 20], практических программах [14, 15, 25] и даже в системах доказательств [1]. В алгоритмах для SAT данная идея обычно называется запоминание кловов (clause learning), что означает, что алгоритм хранит частичные наборы, делающие формулу невыполнимой. В наших новых алгоритмах мы используем естественное расширение данной идеи для задачи MAX-SAT: алгоритм запоминает частичные наборы, которые гарантированно не могут быть дополнены до оптимального (то есть выполняющего максимальное число кловов) набора. Запоминание кловов используется авторами в работе [12].

Второй ключевой идеей работы является использование комбинированных мер сложности. Время работы алгоритма оценивается с помощью меры, зависящей и от числа переменных, и от числа кловов, после чего показывается, что полученная оценка не превосходит некоторой оценки относительно только числа переменных (или же числа кловов). Основная идея заключается в том, что комбинированная мера содержит больше информации от структуре формулы, чем стандартная мера. Такой подход стал довольно популярен в последнее время, см., например, обзорную статью [5].

2. Определения

2.1. Основные определения

Пусть V — множество булевых переменных, то есть переменных, принимающих значения True и False (константы True и False обозначаются также 1 и 0, соответственно). Отри-

чение переменной $v \in V$ обозначается через $\neg v$. Литералом называется переменная или ее отрицание. Через $\text{var}(l)$ мы обозначаем переменную, соответствующую литералу l . Клозом и набором называются множества литералов, не содержащие одновременно переменную и ее отрицание. Длиной клоза называется число содержащихся в нем литералов, k -клоз — это клоз длины k . Формулой в конъюнктивной нормальной форме (КНФ) называется мультимножество клозов. Формулой в k -КНФ называется формула, содержащая только клозы длины не более k . Плотностью формулы называется отношение числа клозов к числу переменных. Следуя работе [6], мы считаем, что формула может содержать также специальный клоз $\mathcal{T}$, который выполняется любым набором.

Полный набор — это набор значений всем переменным формулы. Мы говорим, что клоз C выполняется набором α , если $C \cap \alpha \neq \emptyset$; клоз C опровергается набором α , если $\neg l \in \alpha$ для всех $l \in C$. Через $V(\alpha)$ мы обозначаем множество переменных набора α , через $\text{Cl}(F, \alpha)$ — число клозов формулы F , выполненных набором α , а через $\text{MCl}(F)$ — максимальное число одновременно выполнимых клозов, таким образом, $\text{MCl}(F) = \max_{\alpha} \text{Cl}(F, \alpha)$. Задача SAT заключается в проверке существования набора, выполняющего все клозы формулы, а задача MAX-SAT — в нахождении набора, выполняющего $\text{MCl}(F)$ клозов.

Мы говорим, что литерал содержится в клозе, если этот клоз содержит этот литерал. Однако, говоря, что переменная x содержится в клозе, мы имеем в виду, что этот клоз содержит либо литерал x , либо литерал $\neg x$. Через $\#(F, l)$ мы обозначаем общее число вхождений литерала l в клозы формулы F , а через $\#_k(F, l)$ — общее число вхождений литерала l в k -клозы формулы F .

Литерал, встречающийся в формуле ровно d раз, называется d -литералом; d^+ -литерал входит в формулу хотя бы d раз; (d_1, d_2) -литерал встречается d_1 раз положительно и d_2 раз отрицательно.

Пусть F — формула в КНФ, v — переменная F , α — набор переменным F . Наборы $\{v\}$ и $\{\neg v\}$ мы будем записывать просто как v и $\neg v$. Через $F[\alpha]$ мы обозначаем формулу, полученную из F заменой каждого выполненного набором α клоза на клоз $\mathcal{T}$ и удалением всех вхождений литералов l таких, что $\neg l \in \alpha$, из оставшихся клозов. Пусть также

$$\begin{aligned} F_v &= \{C : C \cup \{v\} \in F\}, \\ F_{\neg v} &= \{C : C \cup \{\neg v\} \in F\}, \\ F_{-v} &= \{C \in F : v, \neg v \notin C\}. \end{aligned}$$

Тогда

$$\begin{aligned} F[v] &= F_{\neg v} \cup F_{-v}, \\ F[\neg v] &= F_v \cup F_{-v}. \end{aligned}$$

Для наборов α и β таких, что $V(\beta) \subseteq V(\alpha)$, через α^β мы обозначаем набор, полученный из α изменением значений переменных $V(\beta)$ в соответствии с β . Например, α^a присваивает значение True переменной a . Заметим, что α^β всегда является расширением набора β .

Пусть α — полный набор значений переменным формулы F , a — (i, j) -литерал формулы F . Тогда

$$\text{Cl}(F, \alpha^a) = i + \text{Cl}(F[a], \alpha) = i + \text{Cl}(F_{-a}, \alpha) + \text{Cl}(F_{-a}, \alpha), \quad (1)$$

$$\text{Cl}(F, \alpha^{\neg a}) = j + \text{Cl}(F[\neg a], \alpha) = j + \text{Cl}(F_a, \alpha) + \text{Cl}(F_{-a}, \alpha). \quad (2)$$

2.2. Меры сложности

Как правило, для доказательства справедливости верхних оценок на время работы алгоритма расщепления вначале фиксируется некоторая мера сложности формул, после чего показывается, что алгоритм всегда расщепляет входную формулу на несколько формул, сложности которых меньше сложности исходной формулы. Естественными мерами сложности формул являются число переменных N , число кловов K и длина формулы L . Известно большое количество верхних оценок относительно этих мер для задач SAT и MAX-SAT и их отдельных случаев. Иногда, однако, можно использовать нестандартную меру сложности для доказательства оценок относительно стандартной меры. Такой подход используется, например, в [6] для доказательства оценки $2^{K/5}$ для MAX-2-SAT и в [13] для доказательства оценки 1.5045^N для 3-SAT. В данной работе мы также используем комбинированные меры сложности для доказательства новых оценок для MAX-SAT и MAX-2-SAT. Наша мера сложности для задачи MAX-SAT выглядит следующим образом:

$$\gamma(F) = N(F) + wK(F),$$

где w — некоторая константа. Мы показываем, что для любой формулы найдется достаточно хорошее расщепление либо относительно числа переменных, либо относительно числа кловов. После этого мы подбираем константу w , зависящую от плотности формулы, так, чтобы соответствующая оценка была меньше 2^N . Ниже мы также описываем вид меры для MAX-2-SAT.

Пусть F — формула в 2-КНФ. Через $N_i(F)$ мы обозначаем число переменных F , входящих в 2-клозы формулы F ровно i раз. Легко видеть, что общее число 2-кловов формулы $K_2(F)$ может быть выражено следующим образом:

$$K_2(F) = \sum_{i=1}^{N(F)} i N_i(F)/2.$$

В [10] использовалась мера, полученная изменением коэффициентов при первых четырех $N_i(F)$:

$$\gamma_0(F) = N_3(F) + 1,9N_4(F) + \sum_{i=5}^{N(F)} i N_i(F)/2.$$

Поясним (неформально), как именно такое изменение коэффициентов может помочь доказать более сильные оценки для MAX-2-SAT. Заметим, что расщепление по переменной веса w (весом переменной 2-КНФ формулы называется число 2-кловов формулы, в которые входит эта переменная) в общем случае является (w, w) -расщеплением относительно K_2 , поскольку в обеих ветках все 2-клозы, содержащие рассматриваемую переменную, либо выполняются, либо становятся единичными. Рассмотрим теперь переменную x веса 5. Если все соседи x различны и имеют вес не более 5, то расщепление x является $(5, 5, 5, 5)$ -расщеплением относительно γ , поскольку в обеих ветках x удаляется (γ уменьшается на 2,5) и веса всех ее соседей уменьшаются (γ уменьшается на $0,6 \cdot 5$).

Мы используем аналогичную меру с другими коэффициентами:

$$\gamma(F) = 0,925 N_3(F) + 1,94 N_4(F) + \sum_{i=5}^{N(F)} i N_i(F)/2.$$

2.3. Анализ алгоритмов расщепления

Как правило, алгоритм расщепления сначала некоторым образом упрощает формулу F , после чего делает рекурсивные вызовы для формул вида $F[\alpha]$ (что и называется расщеплением). Для оценки времени работы такого алгоритма необходимо сначала задать меру сложности формулы $\gamma(F)$. Числом расщепления $F[\alpha_1], \dots, F[\alpha_k]$ называется единственный положительный корень уравнения $1 = \sum_{i=1}^k x^{-a_i}$, где $a_i = \gamma(F) - \gamma(F[\alpha_i])$, обозначаемый $\tau(a_1, \dots, a_k)$. Такое расщепление называется $(b_1, \dots, b_k)$ -расщеплением, если $b_i \geq a_i$ для всех i . Известно, что время работы алгоритма на формуле F не превосходит $\tau_{\max}^{\gamma(F)}$, где $\tau_{\max}$ есть максимальное число расщепления алгоритма. Таким образом, для доказательства верхней оценки времени работы достаточно оценить сверху $\tau_{\max}$. У чисел расщепления есть несколько полезных свойств. В частности,

$$\begin{aligned} \tau(a, a) &= 2^{1/a}, & a &\in \mathbf{R}^+; \\ \tau(a, b) &\leq 2^{1/\sqrt{ab}}, & a, b &\in \mathbf{R}^+; \\ \tau(a_1, \dots, a_p) &\leq \tau(b_1, \dots, b_p), & \text{если } a_i \geq b_i, a_i, b_i &\in \mathbf{R}^+ \text{ для всех } i; \\ \tau(1, 2, \dots, a) &< 2, \\ \tau(1, 2, \dots, a) &< \tau(1, 2, \dots, a+1), & a &\in \mathbf{N}; \\ \tau(1, a) &\rightarrow 1 & \text{при } a &\rightarrow \infty. \end{aligned}$$

2.4. Частичный порядок на наборах

Для доказательства корректности алгоритма расщепления нужно показать, что алгоритм рассматривает всех возможных кандидатов на решения (например, если алгоритм для SAT выдает ответ “невыполнима”, значит, необходимо как-то гарантировать, что ни один из полных наборов не является выполняющим). Для этого алгоритмы расщепления, как правило, расщепляют по множеству наборов $\alpha_1, \dots, \alpha_k$ таких, что любой полный набор является расширением какого-то α_i . Если, однако, известно, что полный набор β может быть решением только в случае, когда полный набор α является решением, то нет никакой необходимости рассматривать набор β . Например, если l — чистый литерал, то нет смысла рассматривать наборы, содержащие $\neg l$. Мы формализуем эту простую идею в следующем определении (аналогичное определение используется в [11]).

Пусть α_1 и α_2 — наборы значений переменных формулы F . Мы говорим, что α_1 сильнее, чем α_2 относительно F , и пишем $\alpha_1 \succ_F \alpha_2$, если для любого полного набора переменных β набор β^{α_2} может быть решением задачи для F , только если β^{α_1} является решением (опять же, мы опускаем F , если это не ведет к двусмысленности). Под решением для SAT мы подразумеваем выполняющий набор, в то время как решением для MAX-SAT мы называем набор, выполняющий максимальное число клозов.

3. Новый алгоритм для MAX-SAT

В данном разделе мы показываем, что MAX-SAT для формул с не более чем ΔN клозами, где Δ — константа, может быть решена за время c^N , где константа $c < 2$.

Для начала мы представляем неформальные соображения, показывающие, что существует алгоритм со временем работы c^N , где $c < 2$, для SAT для формул плотности, не большей Δ . Алгоритм работает следующим образом. Если формула содержит (d^+, d^+) -литерал l для некоторой достаточно большой константы $d = d(\Delta)$, то алгоритм

просто производит расщепление по нему. Таким образом обеспечивается хорошее расщепление относительно числа клозов (в обеих ветках удаляется хотя бы по d клозов). В противном же случае алгоритм выбирает d^- -литерал a и проверяет выполнимость формулы F_a . Это может быть легко сделано за полиномиальное время: на каждом шаге мы выбираем литерал (мы, конечно, предполагаем, что он не является чистым) и расщепляем формулу на две формулы с меньшим числом клозов; поскольку F_a содержит не более d клозов, размер построенного дерева расщепления будет не более 2^d . Если F_a невыполнима, то невыполнима и $F[\neg a] = F_a \cup F_{\neg a}$, и можно просто присвоить значение True переменной a . В противном случае найдется выполняющий набор для F_a . Легко видеть, что он содержит не более d литералов. Обозначим его через $\alpha = \{l_1, \dots, l_k\}$ ($k \leq d$).

Теперь мы делаем рекурсивный вызов для $F[\neg a] = F_a \cup F_{\neg a}$. Если получен ответ “выполнима”, то алгоритм немедленно завершает работу. В противном случае мы заключаем, что α не может быть расширен до выполняющего набора формулы $F_{\neg a}$, что значит, что мы не должны рассматривать расширения α в a -ветке. Таким образом, мы расщепляем $F[a]$ на $F[a, \neg l_1]$, $F[a, l_1, \neg l_2]$, $\dots$, $F[a, l_1, \dots, l_{k-1}, \neg l_k]$. Соответствующее число расщепления относительно количества переменных не превосходит $\tau(1, 2, \dots, k+1) < 2$. Таким образом, алгоритм всегда может найти хорошее расщепление относительно либо числа переменных, либо числа клозов. Поскольку $K \leq \Delta N$, константа d может быть подобрана таким образом, чтобы общее время работы алгоритма не превосходило c^N , где $c < 2$.

Формальное описание алгоритма для MAX-SAT приведено на рис. 2. Алгоритм сначала применяет правило удаления чистых литералов до тех пор, пока это возможно. После этого он расщепляет по D^+ -литералу, если такой литерал есть. Если же такого нет, то алгоритм берет произвольный литерал и находит оптимальные наборы α_a и $\alpha_{\neg a}$ для F_a и $F_{\neg a}$, соответственно. Поскольку F_a и $F_{\neg a}$ содержат менее D клозов, это может быть сделано за полиномиальное время. Более того, можно считать, что α_a и $\alpha_{\neg a}$ содержат менее D литералов. Наконец, алгоритм расщепляет формулу, используя литералы одного из найденных наборов.

Теорема 1. Для любой константы $\Delta > 0$ найдутся константы $D > 0$ и $c < 2$ такие, что MAX-SAT-ALG(D) выдает MCI(F) для любой формулы F с не более чем $\Delta N(F)$ клозами за время $c^{N(F)}$.

Доказательство. Если текущая формула F содержит D^+ -литерал, то алгоритм просто расщепляет по нему. При этом в одной из веток удаляется хотя бы одна переменная и хотя бы один клоз, в другой — хотя бы одна переменная и хотя бы D клозов (напомним, что F не содержит чистых литералов). Рассмотрим теперь случай, когда F состоит только из D^- -литералов. Не умаляя общности, допустим, что $i + k_{\neg a} \geq j + k_a$ (напомним, что a является (i, j) -литералом). Мы утверждаем, что $\alpha_{\neg a} \cup a \not\models_F \alpha_{\neg a} \cup \neg a$. Действительно, рассмотрим произвольный полный набор β переменным формулы F . Из (1) заключаем, что $\text{Cl}(F, \beta^{\alpha_{\neg a} \cup a}) = i + k_{\neg a} + \text{Cl}(F_{\neg a}, \beta^{\alpha_{\neg a}})$, в то время как из (2) следует, что $\text{Cl}(F, \beta^{\alpha_{\neg a} \cup \neg a}) \leq j + k_a + \text{Cl}(F_{\neg a}, \beta^{\alpha_{\neg a}})$. Таким образом, $\beta^{\alpha_{\neg a} \cup a}$ выполняет хотя бы столько же клозов формулы F , сколько выполняет набор $\beta^{\alpha_{\neg a} \cup \neg a}$ (для любого β), следовательно, $\alpha_{\neg a} \cup a \not\models_F \alpha_{\neg a} \cup \neg a$. Значит, нет необходимости рассматривать расширения набора $\alpha_{\neg a}$ в $\neg a$ -ветке. Тогда алгоритм может расщепить следующим образом: $F[a]$, $F[\neg a, \neg y_1]$, $F[\neg a, y_1, \neg y_2]$, $\dots$, $F[\neg a, y_1, \dots, y_{q-1}, \neg y_q]$. Если же $i + k_{\neg a} < j + k_a$, то аналогичным образом показывается, что $\alpha_a \cup \neg a \not\models_F \alpha_a \cup a$. Из этого следует, что алгоритм находит верное решение.

Теперь необходимо оценить время работы алгоритма. Для оценки мы используем

Алгоритм MAX-SAT-ALG.**Параметры:** положительное целое число D .**Вход:** КНФ формула F .**Выход:** $\text{MCI}(F)$.**Метод.**

- (1) Присвоить значение True всем чистым литералам формулы F .
- (2) Если F не содержит никаких кловов, кроме $\mathcal{T}$, вернуть число таких кловов.
- (3) Если F содержит D^+ -литерал a , то произвести рекурсивные вызовы для $F[a]$ и $F[\neg a]$ и вернуть максимум из полученных ответов.
- (4) Пусть a — произвольный литерал F . Пусть также $i = d(a)$, $j = d(\neg a)$.
- (5) Построить оптимальные наборы для F_a и $F_{\neg a}$. Обозначим их через $\alpha_a = \{x_1, \dots, x_p\}$ и $\alpha_{\neg a} = \{y_1, \dots, y_q\}$, соответственно, и пусть

$$\begin{aligned} k_a &= \text{MCI}(F_a) = \text{Cl}(F, \alpha_a), \\ k_{\neg a} &= \text{MCI}(F_{\neg a}) = \text{Cl}(F, \alpha_{\neg a}). \end{aligned}$$

- (6) Если $i + k_{\neg a} \geq j + k_a$, то произвести рекурсивные вызовы для $F[a]$, $F[\neg a, \neg y_1]$, $F[\neg a, y_1, \neg y_2]$, $\dots$, $F[\neg a, y_1, \dots, y_{q-1}, \neg y_q]$ и вернуть максимум из полученных ответов.
- (7) В противном случае произвести рекурсивные вызовы для $F[\neg a]$, $F[a, \neg x_1]$, $F[a, x_1, \neg x_2]$, $\dots$, $F[a, x_1, \dots, x_{p-1}, \neg x_p]$ и вернуть максимум из полученных ответов.

Рис. 2. Алгоритм для MAX-SAT

следующую меру сложности ($w = w(\Delta)$ будет определено позднее):

$$\gamma(F) = \begin{cases} N(F) + wK(F), & \text{если } F \text{ содержит } D^+ \text{-литерал,} \\ N(F) & \text{в противном случае.} \end{cases}$$

Если формула содержит D^+ -литерал, то алгоритм расщепляет с числом расщепления, не большим $\tau(1 + w, 1 + wD)$. В противном случае число расщепления не превосходит $r_D = \tau(1, 2, \dots, D + 1)$, поскольку $p, q \leq D$. Возьмем теперь произвольную константу d , большую Δ . Пусть существуют константы w и D такие, что $\tau(1 + w, 1 + wD) \leq 2^{1/(1+wd)}$, $\tau(1, \dots, D + 1) \leq 2^{1/(1+wd)}$. Тогда время работы алгоритмы ограничено сверху выражением

$$2^{\gamma/(1+wd)} \leq 2^{(N+wK)/(1+wd)} \leq 2^{(N(1+wD))/(1+wd)} = c^N,$$

где константа $c = 2^{(1+wD)/(1+wd)} < 2$. Ниже мы показываем, что такие константы w и D существуют.

Легко видеть, что для любой константы $D \geq 1$ существует константа $w_D > 0$ такая, что

$$\tau(1, \dots, D + 1) = 2^{1/(1+w_D d)},$$

так как $\tau(1, \dots, D + 1) < 2$. Таким образом, достаточно подобрать целое D такое, что

$$\tau(1 + w_D, 1 + w_D D) \leq 2^{1/(1+w_D d)}.$$

Для этого достаточно показать, что

$$(1 + w_D d)^2 \leq (1 + w_D)(1 + w_D D),$$

так как $\tau(a, b) \leq 2^{1/(\sqrt{ab})}$. Последнее неравенство эквивалентно неравенству

$$w_D d^2 + 2d \leq D + 1 + w_D D,$$

которое, очевидно, выполнено для достаточно большого D .

4. MAX-2-SAT

В данном разделе мы доказываем новые верхние оценки для задач MAX-2-SAT и $(n, 3)$ -MAX-2-SAT. Сначала мы описываем правила упрощения, использующиеся алгоритмом, и доказываем важную лемму, в которой показывается, как именно алгоритм может расщеплять. После этого мы представляем алгоритм и доказываем, что его время работы в худшем случае ограничено сверху выражениями $2^{N/6,7}$ и $2^{K/6}$ для задач $(n, 3)$ -MAX-2-SAT и MAX-2-SAT соответственно.

Везде в данном разделе мы рассматриваем только 2-КНФ формулы. Весом переменной называется число ее вхождений в 2-клозы. Соседями переменной x мы называем все переменные, встречающиеся в 2-клозах с x . Для множества переменных $V_0 \subseteq V(F)$ через $N(V_0, F)$ обозначается множество всех соседей множества V_0 , не считая самого V_0 . Через $G(F)$ мы обозначаем мультиграф, в котором $V(F)$ — множество вершин, две вершины соединены ребром, если соответствующие две переменные входят в 2-клоз.

Приведем значения нескольких чисел расщепления, фигурирующих в анализе алгоритмов в данном разделе:

$$\begin{aligned}\tau(6, 8) &= 1,10487 \dots \\ \tau(10, 10, 12) &= 1,10897 \dots \\ \tau(5, 755, 9, 855, 16, 6) &= 1,12008 \dots \\ \tau(5, 64, 6, 565) &= 1,12054 \dots \\ \tau(5, 3, 10, 575, 16) &= 1,12172 \dots \\ \tau(5, 3, 10, 94, 15, 56) &= 1,12121 \dots \\ \tau(5, 3, 9, 48, 18, 2) &= 1,2229 \dots \\ \tau(6, 6) &= 1,2246 \dots\end{aligned}$$

Отметим также, что

$$\tau(6, 8) < \tau(10, 10, 12) < 2^{1/6,7}.$$

4.1. Правила упрощения

Мы используем процедуру SIMPLIFY для упрощения формулы, см. рис. 3. Легко видеть, что время работы данной процедуры полиномиально от длины входной формулы. Корректность, а также несколько других важных свойств данной процедуры приведены в лемме 1. Мы называем формулу упрощенной, если процедура SIMPLIFY не изменяет эту формулу.

Лемма 1. Пусть F — формула в 2-КНФ, и пусть $F' \in \text{SIMPLIFY}$. Тогда

- (1) $\text{MCI}(F) = \text{MCI}(F')$;
- (2) F' не содержит переменных веса не более 2;

Процедура SIMPLIFY.**Вход:** Формула F в 2-КНФ.**Выход:** Упрощенная формула F' в 2-КНФ.**Метод.**

- (1) **Правило почти одинаковые клозы.** Если $F = F_0 \cup \{C, D\}$, где $C \setminus \{l\} = D \setminus \{\neg l\}$ для литерала l , то вернуть $\text{SIMPLIFY}(F_0 \cup \{C \setminus \{x\}, \mathcal{T}\})$.
- (2) **Правило чистый литерал.** Если $\#(F, \neg l) = 0$ для литерала $l \in F$, то вернуть $\text{SIMPLIFY}(F[l])$.
- (3) **Правило доминирующий единичный клоз.** Если $\#_1(F, l) \geq \#(F, \neg l)$ для литерала $l \in F$, то вернуть $\text{SIMPLIFY}(F[l])$.
- (4) **Правило часто встречающиеся переменные.** Пусть F содержит две переменные x_1 и x_2 такие, что x_1 встречается в F в не более чем одном 2-клозе без x_2 . Заметим, что после присваивания x_2 булева значения, x_1 будет встречаться в не более чем одном 2-клозе, а значит, либо правило чистый литерал, либо правило доминирующий единичный клоз присвоит значение x_1 (перед чем, возможно, правило почти одинаковые клозы удалит несколько единичных кловов с x_1). Пусть α, β будут булевыми значениями, которые правила присваивают переменной x_1 в формулах $F[x_2]$ и $F[\neg x_2]$ соответственно. В зависимости от значений α и β правило часто встречающиеся переменные упрощает формулу F следующим образом:
 - если $\alpha = 0, \beta = 0$, то вернуть $\text{SIMPLIFY}(F[\neg x_1])$,
 - если $\alpha = 0, \beta = 1$, то вернуть $\text{SIMPLIFY}(F[x_1 = \neg x_2])$,
 - если $\alpha = 1, \beta = 0$, то вернуть $\text{SIMPLIFY}(F[x_1 = x_2])$,
 - если $\alpha = 1, \beta = 1$, то вернуть $\text{SIMPLIFY}(F[x_1])$.

Вернуть F .**Рис. 3.** Процедура упрощения формулы

- (3) $\#_2(F', x) \leq \#_2(F, x)$ для любой переменной x ;
- (4) $\gamma(F') \leq \gamma(F)$;
- (5) пусть F упрощена, a, x — соседи в F и a имеет вес 3; тогда хотя бы в одной из формул $F[x]$ и $F[\neg x]$ правила упрощения присваивают булево значение переменной a .

Доказательство. Докажем (1). Пусть F — формула, l — литерал F такой, что либо $\#(F, \neg l) = 0$, либо $\#_1(F, l) \geq \#(F, \neg l)$. Заметим, что для любого полного набора A переменных формулы F такого, что $\neg l \in A$, набор $A \setminus \{\neg l\} \cup \{l\}$ выполняет не меньше кловов формулы F , чем A . Это доказывает, что правила чистый литерал и доминирующий единичный клоз сохраняют $\text{MCI}(F)$. Корректность правила часто встречающиеся переменные следует из корректности правил чистый литерал и доминирующий единичный клоз. Правило почти одинаковые клозы корректно, поскольку любой набор всегда выполняет хотя бы один из кловов C и D и выполняет оба этих клова тогда и только тогда, когда он выполняет $C \setminus \{l\}$.

Докажем (2). F' не содержит переменных веса, не большего 2, поскольку все такие переменные могут быть удалены правилом часто встречающиеся переменные.

Докажем (3). Легко видеть, что ни присваивание значения переменной, ни замена двух кловов на их общую часть не увеличивает весов переменных. Покажем, что правило часто встречающиеся переменные тоже не увеличивает веса. Единственная переменная,

чей вес мог бы быть увеличен, есть x_2 . Однако $F[x_1 = x_2]$ (так же как и $F[x_1 = \neg x_2]$) не содержит 2-кловов формулы F , состоящих из x_1 и x_2 (в F есть хотя бы один такой клов) и содержит не более одного нового 2-клоза (поскольку x_1 встречается без x_2 не более одного раза).

Утверждение (4) следует непосредственно из (3) и определения γ .

Для доказательства утверждения (5) необходимо рассмотреть несколько случаев других вхождений переменной a (напомним, что F упрощена):

- (а) $(ax)(a?)(\neg a?)$ — после подстановки $x = 0$ правило *доминирующий единичный клов* присваивает $a = 1$;
- (б) $(ax)(\neg a?)(\neg a?)$ — после подстановки $x = 1$ правило *чистый литерал* присваивает $a = 0$;
- (в) $(ax)(a?)(a?)(\neg a)$ — после подстановки $x = 0$ правило *почти одинаковые клозы* удаляет (a) и $(\neg a)$, после чего *чистый литерал* присваивает $a = 1$;
- (г) $(ax)(a?)(a?)(\neg a)(\neg a)$ — после подстановки $x = 1$ правило *доминирующий единичный клов* присваивает $a = 0$;
- (д) $(ax)(a?)(\neg a?)(\neg a)$ — после подстановки $x = 1$ правило *доминирующий единичный клов* присваивает $a = 0$;
- (е) $(ax)(\neg a?)(\neg a?)(a)$ — после подстановки $x = 0$ правило *доминирующий единичный клов* присваивает $a = 1$.

4.2. Правила расщепления

Рассмотрим литерал a веса w . Мы называем литерал слабым, если

$$\#_2(a) + \#_1(a) - \#_1(\neg a) \geq w/2.$$

Легко видеть, что для любой переменной x либо литерал x , либо литерал $\neg x$ является слабым. Отметим также, что для упрощенной формулы $\#_2(a) + \#_1(a) - \#_1(\neg a)$ строго меньше w (иначе a был бы чистым литералом).

Лемма 2. Пусть F — упрощенная 2-КНФ формула, a — слабый литерал веса w , $d = \lfloor w \rfloor$. Пусть $b_1, \dots, b_d$ — литералы, встречающиеся с $\text{var}(a)$. Тогда

$$\{a, c_1, \dots, c_d\} \geq_F \{\neg a, c_1, \dots, c_d\},$$

где c_i — либо b_i , либо $\neg b_i$.

Доказательство. Пусть

$$i_2 = \#_2(a), \quad i_1 = \#_1(a), \quad j_2 = \#_2(\neg a), \quad j_1 = \#_1(\neg a).$$

Пусть α есть полный набор для переменных формулы F . Тогда в силу (1) и (2)

$$\begin{aligned} \text{Cl}(F, \alpha^a) &= i_1 + i_2 + \text{Cl}(F_{\neg a}, \alpha) + \text{Cl}(F_{\neg a}, \alpha), \\ \text{Cl}(F, \alpha^{\neg a}) &= j_1 + j_2 + \text{Cl}(F_a, \alpha) + \text{Cl}(F_{\neg a}, \alpha). \end{aligned}$$

Значит,

$$\begin{aligned} \text{Cl}(F, \alpha^a) - \text{Cl}(F, \alpha^{\neg a}) &= i_1 + i_2 - j_1 - j_2 + \text{Cl}(F_{\neg a}, \alpha) - \text{Cl}(F_a, \alpha) \\ &\geq \lceil w/2 \rceil - j_2 + \text{Cl}(F_{\neg a}, \alpha) - \text{Cl}(F_a, \alpha). \end{aligned}$$

Отметим, что

$$\text{Cl}(F_{\neg a}, \alpha) - \text{Cl}(F_a, \alpha) \geq -i_2,$$

так как α выполняет хотя бы 0 клозов $F_{\neg a}$ и не более i_2 клозов F_a . Если (b_i) или $(\neg b_i)$ содержится в $F_{\neg a}$, то мы присваиваем c_i так, чтобы выполнить данный единичный клоз. В противном случае, если (b_i) или $(\neg b_i)$ содержится в F_a , то мы присваиваем c_i так, чтобы опровергнуть соответствующий клоз. Тогда

$$\text{Cl}(F_{\neg a}, \alpha) - \text{Cl}(F_a, \alpha) \geq -i_2 + d$$

для любого $\alpha \ni c_1, \dots, c_d$, и следовательно,

$$\text{Cl}(F, \alpha^a) - \text{Cl}(F, \alpha^{\neg a}) \geq \lceil w/2 \rceil - j_2 - i_2 + d \geq w - i_2 - j_2 \geq 0.$$

Таким образом, для любого полного набора α такого, что $\{l_1, \dots, l_d\} \subseteq \alpha$,

$$\text{Cl}(F, \alpha^a) \geq \text{Cl}(F, \alpha^{\neg a}),$$

из чего следует, что

$$\{a, c_1, \dots, c_d\} \geq_F \{\neg a, c_1, \dots, c_d\}.$$

Приведем пример использования данной леммы. Пусть F содержит пять клозов $(al_1)(al_2)(al_3)(\neg al_4)(\neg al_5)$ (и не содержит других клозов с переменной a). Лемма 2 утверждает, что

$$\{a, \neg l_2, l_4\} \geq_F \{\neg a, \neg l_2, l_4\}.$$

Действительно, любое расширение $\{\neg a, \neg l_2, l_4\}$ выполняет не более четырех из рассматриваемых клозов (поскольку (al_2) опровергается), в то время как любое расширение $\{a, \neg l_2, l_4\}$ выполняет хотя бы четыре из них. Приведем также важное следствие из данной леммы.

Следствие 1. Пусть F — упрощенная 2-КНФ формула, a — слабый литерал F веса 3. Тогда в $\neg a$ -ветке каждому соседу $\text{var}(a)$ присваивается значение.

Доказательство. По лемме 2 для любого соседа b переменной $\text{var}(a)$

$$\{a, c\} \geq_F \{\neg a, c\},$$

где c есть либо b , либо $\neg b$. Таким образом, алгоритм может не рассматривать расширения набора $\{\neg a, c\}$, а следовательно, в $\neg a$ -ветке литералу c может быть присвоено значение False.

Алгоритм MAX-2-SAT-ALG.

Вход: Формула F в 2-КНФ.

Выход: $MCI(F)$.

Метод.

- (1) $F = \text{SIMPLIFY}(F)$.
- (2) Если F не содержит клозы кроме $(\mathcal{T})$, вернуть число таких клозов.
- (3) Если $F = F_1 \cup F_2$, где F_1 и F_2 не пересекаются по переменным, то вернуть $\text{MAX-2-SAT-ALG}(F_1) + \text{MAX-2-SAT-ALG}(F_2)$.
- (4) Расщепить формулу, произвести рекурсивные вызовы, и вернуть максимальный из полученных ответов.

Рис. 4. Алгоритм для MAX-2-SAT

4.3. Алгоритм для MAX-2-SAT

Алгоритм для MAX-2-SAT представлен на рис. 4. По форме он аналогичен алгоритму Кожевникова и Куликова [10]. В данной работе, используя идею запоминания клозов, мы доказываем более сильную оценку времени его работы. Отметим, что в описании алгоритма не указано, как именно алгоритм производит расщепления, однако это явно следует из анализа случаев, приведенного в следующих разделах.

Время работы алгоритма Кожевникова и Куликова оценивается относительно меры сложности

$$\gamma(F) = N_3 + 1,9N_4(F) + \sum_{i \geq 5} i N_i(F)/2,$$

где $N_i(F)$ — число переменных веса i формулы F . Мы изменяем коэффициенты следующим образом:

$$\gamma(F) = 0,925N_3 + 1,94N_4(F) + \sum_{i \geq 5} i N_i(F)/2.$$

Таким образом, необходимо доказать, что алгоритм всегда расщепляет с числом расщепления, не большим $\tau(6, 6)$. Заметим, что уменьшение веса переменной упрощенной формулы уменьшает γ хотя бы на 0,5 (поскольку разность любых двух коэффициентов N_i не меньше 0,5). Для начала рассмотрим случай, когда F содержит переменную x веса $w \geq 6$. В этом случае мы можем просто расщепить по этой переменной. Это уменьшит γ хотя бы на w в обеих ветках: удаление x уменьшает γ хотя бы на $w/2$, уменьшение весов соседей переменной x уменьшает γ еще хотя бы на $w/2$.

Рассмотрим теперь случай, когда F содержит только переменные веса, не большего 4. Пусть x есть переменная веса 4. Для $1 \leq i \leq j \leq 4$ пусть k_{ij} обозначает число соседей x , имеющих вес j и встречающихся i раз вместе с x . Легко видеть, что такой сосед x становится переменной веса $(j - i)$ после присвоения значения переменной x . В силу леммы 1 справедливы равенства $k_{ij} = 0$ для $j \leq 2$ и $j - i \leq 1$. Значит, только k_{13} , k_{14} и k_{24} могут быть положительными. Переменная x имеет вес 4, поэтому

$$k_{13} + k_{14} + 2k_{24} = 4.$$

Пусть теперь F' — формула, полученная из F присвоением значения переменной x . Тогда

$$\begin{aligned}\gamma(F) - \gamma(F') &= (0,925k_{13} + 1,94(k_{14} + k_{24} + 1)) - 0,925k_{14} \\ &= 0,925(k_{13} + k_{14} + 2k_{24}) + 0,09(k_{14} + k_{24}) + 1,94 \\ &= 5,64 + 0,09(k_{14} + k_{24}).\end{aligned}$$

Заметим теперь, что если $k_{14} = 4$, то

$$\gamma(F) - \gamma(F') \geq 5,64 + 0,09 \cdot 4 = 6,$$

в то время как если $k_{14} < 4$, то либо $k_{13} > 0$, либо $k_{24} > 0$. Если $k_{13} > 0$, то у x есть сосед веса 3, которому по лемме 1 хотя бы в одной из веток присваивается значение, что влечет за собой уменьшение веса еще хотя бы одной переменной в этой ветке и соответствует расщеплению $\tau(5,64, 5,64 + 0,925) = \tau(5,64, 6,565)$. Аналогично, если $k_{24} > 0$, то у x есть сосед веса 4, встречающийся вместе с x дважды. При присвоении переменной x значения, вес такого соседа уменьшается до двух. Значит, хотя бы в одной из веток ему будет присвоено значение, а тогда будет уменьшен вес еще хотя бы одной новой переменной.

Если F содержит только переменные веса, не большего трех, то, как показано в разделе 4.4, найдется $(6,7, 6,7)$ -расщепление относительно N . Это приводит к $(6,1975, 6,1975)$ -расщеплению относительно γ , поскольку $\gamma(F) = 0,925n(F)$. В оставшемся случае формула содержит хотя бы одну переменную веса 5 и все остальные переменные имеют вес не более 5, и это как раз тот случай, где мы используем запоминание клозов.

Вычислим для начала число расщепления, получающееся непосредственно по расщеплению по рассматриваемой переменной. Вычисление производится полностью аналогично случаю, когда есть переменная веса 4:

$$k_{13} + k_{14} + k_{15} + 2k_{24} + 2k_{25} + 3k_{35} = 5,$$

и

$$\begin{aligned}\gamma(F) - \gamma(F') &= (0,925k_{13} + 1,94(k_{14} + k_{24}) \\ &\quad + 2,5(k_{15} + k_{25} + k_{35} + 1)) - (0,925(k_{14} + k_{25}) + 1,94k_{15}) \\ &= 0,925k_{13} + 1,015k_{14} + 0,56k_{15} + 1,94k_{24} + 1,575k_{25} + 2,5k_{35} + 2,5 \\ &= 0,56(k_{13} + k_{14} + k_{15} + 2k_{24} + 2k_{25} + 3k_{35}) + 2,5 \\ &\quad + 0,365k_{13} + 0,455k_{14} + 0,82k_{24} + 0,455k_{25} + 0,82k_{35} \\ &= 5,3 + 0,365k_{13} + 0,455k_{14} + 0,82k_{24} + 0,455k_{25} + 0,82k_{35} \\ &> 5,3 + 0,365(k_{13} + k_{14} + 2k_{24} + k_{25} + 2k_{35}).\end{aligned}$$

Таким образом, если

$$(k_{13} + k_{14} + 2k_{24} + k_{25} + 2k_{35}) \geq 2, \quad (3)$$

то необходимая нам оценка числа расщепления верна. Это, в частности, означает, что нет необходимости в рассмотрении случаев $k_{24} > 0$ и $k_{35} > 0$.

Далее мы рассматриваем все оставшиеся случаи. Напомним, что F содержит только переменные веса не более 5. Пусть a есть переменная веса 5. Рассмотрим сначала случай, когда a имеет соседа b веса 3. По (3), $k_{13} = 1$ и $k_{15} = 4$. По лемме 1, b получает значение

хотя бы в одной из веток $F[a]$ и $F[\neg a]$, более того, все соседи b различны. Значит, расщепление по a дает число расщепления, не большее, чем

$$\tau(5,3 + 0,365, 5,3 + 0,365 + 2 \cdot 0,56) = \tau(5,665, 6,785).$$

Допустим, что a встречается с некоторой переменной b дважды. В силу (3), из этого следует, что $k_{25} = 1$ и $k_{15} = 3$. Из леммы 2 следует, что

$$\{l_a, l_b\} \geq_F \{\neg l_a, l_b\},$$

где l_a и l_b суть литералы переменных a и b . Значит, мы можем присвоить значение переменной b в одной из веток $F[a]$ и $F[\neg a]$. Соответствующее число расщепления есть

$$\tau(5,3 + 0,365, 5,3 + 0,365 + 0,925) = \tau(5,665, 6,59).$$

Далее мы предполагаем, что все пять соседей a различны. Более того, согласно (3), у не более одного из них вес может быть равен 4, в то время как все остальные имеют вес 5 (другим словами, $k_{14} \leq 1$, $k_{15} \geq 4$, $k_{14} + k_{15} = 5$). Не уменьшая общности, полагаем, что $\#_1(a) + \#_1(a) - \#_1(\neg a) \geq 3$ (в противном случае рассмотрим $\neg a$ вместо a). Пусть литералы b_1, b_2 встречаются вместе с литералом a , литерал c встречается с $\neg a$. Тогда из леммы 2 следует, что

$$\begin{aligned} \{a, \neg b_1, \neg b_2\} &\geq_F \{\neg a, \neg b_1, \neg b_2\}, \\ \{a, \neg b_1, c\} &\geq_F \{\neg a, \neg b_1, c\}. \end{aligned}$$

Тогда мы можем расщепить F на $F[a]$, $F[\neg a, b_1]$, $F[\neg a, \neg b_1, b_2, \dots, b_{i_2}, \neg c_1, \dots, \neg c_{j_2}]$, где $b_1, \dots, b_{i_2}$ и $c_1, \dots, c_{j_2}$ суть все литералы, встречающиеся с литералами a и $\neg a$, соответственно. Таким образом, нам необходимо показать, что такое расщепление всегда дает необходимую числа расщепления. Для начала переименуем все литералы так, чтобы рассматриваемое расщепление приобрело вид $F[a]$, $F[\neg a, l_1]$, $F[\neg a, \neg l_1, l_2, l_3, l_4, l_5]$. Пусть l_1, l_2, l_3, l_4 суть литералы переменных веса 5, а l_5 — литерал переменной веса d , где $d \in \{4, 5\}$.

Допустим сначала, что вес l_5 равен 4. Тогда

$$\gamma(F) - \gamma(F[a]) \geq 2,5 + 4 \cdot 0,56 + 1,015 = 5,755.$$

По лемме 2, $F[\neg a]$ можно расщепить на $F[\neg a, l_1]$ и $F[\neg a, \neg l_1, l_2, l_3, l_4, l_5]$. Допустим, что у l_1 есть k соседей веса не более 5, $0 \leq k \leq 4$. Соответствующее число расщепления равно $\tau(5,755, 5,755 + 1,94 + 0,925k + 0,54(4 - k))$, $5,755 + 4 \cdot 1,94 + 0,925 + 0,54(4 - k)$. Несложно видеть, что наибольшее значение этого выражения достигается при $k = 0$ и равно $\tau(5,755, 9,855, 16,6)$.

Далее, предполагаем, что веса всех l_i равны 5. После подстановки a значения вес каждого l_i равен 4. Значит, если в такой формуле у какого-то из l_i есть сосед веса 5, то этот сосед не совпадает ни с одним из l_i . Допустим, что у каждого из l_i не более двух соседей веса, меньшего 5. Тогда множество соседей всех l_i состоит хотя бы из десяти литералов, и соответствующее число расщепления не превосходит

$$\tau(5,3, 5,3 + 1,94 + 4 \cdot 0,56, 5,3 + 5 \cdot 1,94 + 3,2) = \tau(5,3, 9,48, 18,2).$$

Рассмотрим теперь случай, когда какой-то из литералов l_i имеет трех соседей веса, меньшего 5. Такому случаю соответствует расщепление

$$\tau(5,3, 5,3 + 1,94 + 0,56 + 3 \cdot 0,925, 5,3 + 5 \cdot 1,94 + 1) = \tau(5,3, 10,575, 16).$$

В оставшемся случае у одного из l_i есть ровно четыре соседа веса, меньшего 5. Число расщепления в таком случае не превосходит

$$\tau(5,3, 5,3 + 1,94 + 4 \cdot 0,925, 5,3 + 5 \cdot 1,94 + 0,56) = \tau(5,3, 10,94, 15,56),$$

литералы l_i не образуют замкнутую подформулу.

4.4. $(n, 3)$ -MAX-2-SAT

В данном разделе мы показываем, что время работы алгоритма MAX-2-SAT-ALG для задачи $(n, 3)$ -MAX-2-SAT в худшем случае есть $2^{N/6,7}$. Везде в данном подразделе мы рассматриваем только формулы в 2-КНФ, содержащие переменные веса, не большего 3. Более того, поскольку переменная веса два может быть удалена правилами упрощения, любая такая упрощенная формула содержит только переменные веса 3, из чего следует, что число переменных четно. Таким образом, расщепление по любой переменной является $(2k, 2l)$ -расщеплением относительно числа переменных для некоторых целых k и l . Более того, поскольку к F не применимо правило *часто встречающиеся переменные*, все три соседа любой переменной соответствуют разным переменным, и следовательно, будут удалены правилами упрощения при присваивании значения этой переменной.

Лемма 3. Пусть F — упрощенная формула, x и a суть переменные, встречающиеся вместе в F , и пусть переменной a присваиваются значения в обеих формулах $F[x]$ и $F[\neg x]$. Тогда для F существует $(6, 8)$ -расщепление относительно N .

Доказательство. Если в обеих ветках переменной a присваивается одно и то же значение, то формулу можно упростить, присвоив это значение. Допустим, что в $F[x]$ и $F[\neg x]$ переменной a присваиваются значения 1 и 0, соответственно. Значит, переменную x можно заменить на переменную a . В полученной формуле, однако, переменная a может встречаться в четырех 2-клозах. Обозначим через b, c, d, e соседей a в полученной формуле. Отметим, что все четыре соответствующие переменные различны, поскольку в противном случае было бы применимо правило *часто встречающиеся переменные*. При расщеплении по a каждой из четырех рассматриваемых переменных будет присвоено значение в одной из веток. Из отсутствия замкнутых подформул следует, что $|N(\{b, c, d, e\} \setminus \{a, b, c, d, e\})| > 0$, следовательно, в одной из веток будет удалена еще хотя бы одна дополнительная переменная. Итак, в одной из веток удаляется хотя бы шесть переменных, в другой — хотя бы семь. Вспомнив, что каждое расщепление является $(2j, 2k)$ -расщеплением, заключаем, что мы всегда можем найти $(6, 8)$ -расщепление.

Рассмотрим слабый литерал x . Заметим, что в случае $(n, 3)$ -MAX-2-SAT для любого слабого литерала x выполнено равенство

$$\#_2(x) + \#_1(x) - \#_1(\neg x) = 2.$$

Пусть $N(\{x\}) = \{a, b, c\}$. По следствию 1, переменным $\text{var}(a)$, $\text{var}(b)$, $\text{var}(c)$ присваивается значение в $\neg x$ -ветке. Если хотя бы одной из них также присваивается значение в x -ветке, то по лемме 3 существует $(6, 8)$ -расщепление. В дальнейшем мы рассматриваем несколько случаев для x, a, b, c , в каждом из которых строим необходимое расщепление.

Назовем 2-кюз смешанным, если он содержит ровно один слабый литерал. Если F содержит смешанный кюз, то, по следствию 1 и лемме 3, найдется $(6, 8)$ -расщепление. Таким образом, в дальнейшем мы предполагаем, что формула не содержит смешанных клозов.

Рис. 5. Основные случаи доказательства

Лемма 4. Пусть a — переменная формулы F , не содержащей смешанных клов, x — сосед a . Тогда при присвоении x значения либо переменной a присваивается значение, либо создается смешанный клов.

Доказательство. При присвоении значения переменной x вес переменной a уменьшается, после чего она удаляется правилами упрощения. Если ей не присваивается значения, то она удаляется либо правилом *часто встречающиеся переменные*, либо правилом *почти одинаковые кловы*. Если переменная a удаляется заменой кловов (ab) и $(\neg ac)$ на клов (bc) , то полученный клов, очевидно, является смешанным (поскольку ни (ab) , ни $(\neg ac)$ смешанными не являлись). Если же формула содержит кловы (ab) , (ac) , $(\neg a)$, то переменная a удаляется подстановкой $a = \neg b$. Легко видеть, что при такой замене появится хотя бы один смешанный клов (отметим, однако, что после подстановки значение выражение $\#_2(b) + \#_1(b) - \#_1(\neg b)$ может поменяться).

Допустим, что формула содержит три переменные, для каждой двух из которых есть клов, содержащий обе эти переменные. Такие переменные образуют треугольник в графе $G(F)$. Назовем их a, b, c . Заметим, что $|N(\{a, b, c\})| > 1$, поскольку в противном случае a, b, c и их сосед образовывали бы замкнутую подформулу. Хотя бы одна из этих переменных имеет ровно одного соседа среди $\{a, b, c\}$. Назовем эту переменную d и, не умаляя общности, будем считать, что она является соседом c ; оставшихся двух соседей d назовем e и f (см. рис. 5(a)). Мы покажем, что расщепление по d удовлетворяет необходимым нам условиям. Не умаляя общности, считаем, что в d -ветке присваиваются значения всем соседям d . После присваивания значения переменной d переменная c входит в два 2-клоза с переменными a и b . Правила упрощения либо присваивают значение переменной c , либо заменяют эти два клова на клов, содержащий переменные a и b . В каждом из этих случаев дальнейшим применением правил упрощения удаляется хотя бы одна из переменных a и b . В d -ветке присваиваются значения переменным c, e, f , после чего удаляются переменные a, b , а также еще хотя бы один сосед переменных e, f (поскольку рассматриваемые переменные не образуют замкнутую подформулу). Итак, в d -ветке удаляются c, e, f, a, b и хотя бы один сосед e и f , а в $\neg d$ -ветке — c, e, f и хотя бы одна из a, b . В итоге получаем (6, 8)-расщепление. В дальнейшем мы предполагаем, что $G(F)$ не содержит треугольников.

Допустим, что $|N(\{a, b, c\}) \setminus \{x\}| \geq 5$ (число соседей a, b, c , не считая x , равно хотя бы 5). Тогда расщепление по x является (4, 10)-расщеплением. Более того, по лемме 4 хотя бы в одной из веток появляется смешанный клов. Таким образом, получается число

расщепления, равное

$$\tau(4 + 6, 4 + 8, 10) = \tau(10, 10, 12).$$

Если же выполняется неравенство $|N(\{a, b, c\}) \setminus \{x\}| < 5$, то либо существует переменная $y \neq x$ такая, что $N(y) = \{a, b, c\}$ (см. рис. 5(b)), либо существуют две переменные $y, z \neq x$ такие, что $|N(y) \cap \{a, b, c\}| = |N(z) \cap \{a, b, c\}| = 2$ (см. рис. 5(c)).

Рассмотрим первый из этих случаев. Поскольку $G(F)$ не содержит треугольников, y b есть сосед $d \notin \{a, c, x, y\}$. При расщеплении по b в одной ветке присваиваются переменные x, y, d , после чего присваиваются также a, c и еще хотя бы одна переменная удаляется, а в другой ветке удаляются d, x, y , после чего удалится хотя бы одна из a, c . В итоге получаем (6, 8)-расщепление.

В оставшемся случае (рис. 5(c)) алгоритм расщепляет по переменной x . В одной из веток a, b, c получают значения, после чего присваиваются и y , и z . Поскольку в $G(F)$ нет треугольников и F не содержит замкнутых подформул, можно заключить, что $|N(\{x, a, b, c, y, z\})| \geq 3$, что означает, что удалятся еще хотя бы три переменные. В другой ветке удаляются переменные a, b, c . При этом, по лемме 4 либо одной из них присваивается значение, либо создается смешанный кюз. Соответствующее расщепление является либо (6, 8)-, либо (10, 10, 12)-расщеплением.

Замечание 1. В [19] доказана новая верхняя оценка для MAX-2-SAT, равная $2^{K/6,2158}$. Алгоритм получен усовершенствованием алгоритма Кожевникова и Куликова [10]. Как отмечают авторы, использование запоминания кюзов улучшило их оценку с $2^{K/6,1489}$ до $2^{K/6,2158}$.

Авторы благодарят Эдуарда Алексеевича Гирша за ценные замечания.

Список литературы

1. Beame P., Impagliazzo R., Pitassi T., Segerlind N., Memorization and DPLL: formula caching proof systems. In: *Proc. 18th IEEE Annual Conf. Computational Complexity*, 2003, pp. 248–259.
2. Chen J., Kanj I., Improved exact algorithms for MAX-SAT. *Lecture Notes Computer Sci.* (2002) **2286**, 341–355.
3. Dantsin E., Wolpert A., MAX-SAT for formulas with constant clause density can be solved faster than in $O(2^n)$ time. *Lecture Notes Computer Sci.* (2006) **4121**, 266–276.
4. Fedin S. S., Kulikov A. S., Automated proofs of upper bounds on running time of splitting algorithms. *J. Math. Sci.* (2006) **134**, №5, 2383–2391.
5. Fomin F. V., Grandoni F., Kratsch D., *A Measure & Conquer approach for the analysis of exact algorithms*, Techn. Rep. 359. Univ. Bergen, 2007.
6. Gramm J., Hirsch E. A., Niedermeier R., Rossmanith P., Worst-case upper bounds for MAX-2-SAT with an application to MAX-CUT. *Discrete Appl. Math.* (2003) **130**, №2, 139–155.
7. Hirsch E. A., *A $2^{K/4}$ -time algorithm for MAX-2-SAT: Corrected version.*, ECCC Report TR99-036, Rev. 02, 2000.
8. Kneis J., Moelle D., Richter S., Rossmanith P., Algorithms based on the treewidth of sparse graphs. *Lecture Notes Computer Sci.* (2005) **3787**, 385–396.
9. Kneis J., Rossmanith P., *A new satisfiability algorithm with applications to MAX-CUT*, Techn. Rep. AIB2005-08. Dept. Computer Sci. RWTH Aachen Univ., 2005.
10. Kojevnikov A., Kulikov A. S., A new approach to proving upper bounds for MAX-2-SAT. In: *Proc. 17th Annual ACM-SIAM Symp. Discrete Algorithms*, 2006, pp. 11–17.

11. Kulikov A. S., Automated generation of simplification rules for SAT and MAXSAT. *Lecture Notes Computer Sci.* (2005) **3569**, 430–436.
12. Kulikov A. S., Kutzkov K., New bounds for MAX-SAT by clause learning. *Lecture Notes Computer Sci.* (2007) **4549**, 194–204.
13. Kullmann O., New methods for 3-SAT decision and worst-case analysis. *Theoret. Computer Sci.* (1997) **223**, 1–72.
14. Marques-Silva J. P., Sakallah K. A., GRASP: a search algorithm for propositional satisfiability. *IEEE Trans. Comput.* (1999) **48**, №5, 506–521.
15. Moskewicz M. W., Madigan C. F., Zhao Y., Zhang L., Malik S., CHAFF: engineering an efficient SAT solver. In: *Proc. 38th Design Automation Conf.*, 2001, pp. 530–535.
16. Niedermeier R., Rossmanith P., New upper bounds for MAX-SAT. *Lecture Notes Computer Sci.* (1999) **1644**, 575–585.
17. Poljak S., Tuza Z., Maximum cuts and largest bipartite subgraphs. *DIMACS, Ser. Discrete Math. Theor. Comput. Sci.* (181–244) **20**,.
18. Raible D., Fernau H., A new upper bound for MAX-2-SAT: A graph-theoretic approach. In: *MFCS*, 2008, pp. 551–562.
19. Raman V., Ravikumar B., Srinivasa Rao S., A simplified NP-complete MAXSAT problem. *Inform. Process. Lett.* (1998) **65**, №1, 1–6.
20. Robson J. M., Algorithms for maximum independent sets. *J. Algorithms* (1986) **7**, №3, 425–440.
21. Scott A. D., Sorkin G. B., Linear-programming design and analysis of fast algorithms for MAX-2-SAT and MAX-2-CSP. *Discrete Optimization* (2007) **4**, 260–287.
22. Scott A. D., Sorkin G. B., Faster algorithms for MAX-CUT and MAX-CSP, with polynomial expected time for sparse instances. *Lecture Notes Computer Sci.* (2003) **2764**, 382–395.
23. Williams R., On computing k -CNF formula properties. *Lecture Notes Computer Sci.* (2003) **2919**, 330–340.
24. Williams R., A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoret. Computer Sci.* (2005) **348**, 357–365.
25. Zhang H., An efficient propositional prover. *Lecture Notes Computer Sci.* (1997) **1249**, 272–275.

Статья поступила 11.03.2008.