Записки научных
семинаров ПОМИ
Том 293, 2002 г.

А. С. Куликов

ВЕРХНЯЯ ОЦЕНКА $O(2^{0.16254n})$ ДЛЯ X3SAT: БОЛЕЕ ПРОСТОЕ ДОКАЗАТЕЛЬСТВО

1. Введение

Задача X3SAT и наш результат. Задача точной 3-выполнимости (*X3SAT*) заключается в нахождении для заданной формулы в конъюнктивной нормальной форме (КНФ), каждый клоз которой содержит не более трех литералов, полного набора истинностных значений (т.е. набора, в котором каждой переменной присвоено некоторое значение), выполняющего *ровно* по одному литералу в каждом клозе. Очевидно, что задача X3SAT является частным случаем системы линейных уравнений в 0/1-переменных. Хорошо известно, что X3SAT – NP-полна [6].

Наилучшими известными верхними оценками для XSAT (более общей проблемы – без ограничения на длину клоза) и X3SAT являются $poly(|F|) \cdot 2^{0.244499n}$ [3] и $O(2^{0.186916n})$ [4], соответственно ($poly(|F|)$ – полином от длины входной формулы). Отметим, что, в отличие от 3SAT, решая задачу X3SAT, мы не можем удалять не только "чистые" литералы, но даже и переменные, входящие в формулу ровно один раз. Наше новое правило преобразования удаляет клозы, содержащие две такие переменные.

Мы оцениваем время работы нашего алгоритма, оценивая количество листьев в дереве рекурсии. Будет показано, что в наихудшем случае количество листьев есть решение неравенства $T(n) \leq T(n-9) + T(n-4)$, то есть $O(1.119253^n) = O(2^{0.162536n}) = O(2^{n/6.152492})$.

Структура нашего алгоритма (а также статьи). Наш алгоритм является типичным алгоритмом, использующим расщепление. Именно, он использует процедуру, сводящую задачу для формулы F к задаче для двух формул с меньшим количеством переменных. Более точно, наш алгоритм строит дерево при помощи

При частичной поддержке гранта No. 1 6-го конкурса-экспертизы научных проектов молодых ученых РАН (1999).

сведения точной выполнимости формулы F к точной выполнимости формул $F[l]$ и $F[\overline{l}]$, где l – литерал, входящий в формулу F (все необходимые определения даны в разд. 2). После этого он упрощает каждую из формул $F[l]$ и $F[\overline{l}]$ при помощи правил преобразования, применение которых не влияет на точную выполнимость формул. Ясно, что время работы алгоритма с точностью до полиномиального сомножителя равно количеству листьев в дереве рекурсии. По существу, остается привести список правил преобразования (см. разд. 3), описать эвристику выбора l и доказать, что рекуррентные неравенства на количество листьев имеют необходимые решения (см. разд. 4).

Предыдущие результаты. Наша оценка $O(2^{0.162536n})$ улучшает оценку $O(2^{0.186916n})$ Поршена, Рандерата и Шпекенмейера [4]. После того, как предварительная версия [1] настоящей статьи была опубликована, авторам стало известно, что Поршен, Рандерат и Шпекенмейер независимо улучшили свой результат (см. [5]), получив ту же оценку $O(2^{0.162536n})$. Наш алгоритм немного отличается от алгоритма Поршена и др. (хотя оба алгоритма основаны на одной идее, появивишейся в [4]), но доказательства корректности и времени работы в нашей статье получились заметно короче. Более точно, оба алгоритма используют идею статьи [4], что если формула не содержит клоза, состоящего из переменной x, входящей в формулу хотя бы три раза, и двух переменных y, z, каждая из которых входит в формулу не менее двух раз, то точная выполнимость этой формулы может быть проверена за полиномиальное время. Во всех упомянутых статьях (равно как и в настоящей статье) такой клоз используется для расщепления. Однако, алгоритм Поршена и др. расщепляет лишь по переменной x, а наш алгоритм производит расщепления также и по другим переменным (см. шаг 6 нашего алгоритма). Это различие позволяет нам сократить анализ алгоритма.

2. Основные определения

Пусть V – множество булевых переменных. Отрицание переменной v обозначается $\overline{v}$. *Литерал* есть переменная или ее отрицание. Если l обозначает переменную $\overline{v}$, то $\overline{l}$ обозначает переменную v. Мы называем два литерала l_1 и l_2 *родственными* и пишем $l_1 \sim l_2$, если l_1 и l_2 соответствуют одной и той же переменной (ясно, что для каждого литерала $l \sim \overline{l}$). Переменную и соответствующий ей литерал мы называем *уникальными*, если эта переменная входит в формулу ровно один раз. В дополнение к обычным литералам мы допускаем специальный *истинный литерал* $\mathcal{T}$, который всегда имеет значение *True* и не соответствует никакой переменной. (Мы также называем его $\mathcal{T}$-литералом и не обозначаем его при помощи других символов.) В данной статье мы допускаем, что клоз может содержать $\mathcal{T}$-*литерал*. Мы называем k-*клозом* клоз, состоящий ровно из k литералов. *Формула в КНФ* (или просто *формула*) – это конечное множество клозов; *длина* формулы – суммарное количество содержащихся в ней литералов.

Для литерала l и формулы F через $F[l]$ мы обозначаем формулу, полученную из формулы F присваиванием значения *True* литералу l, т.е. удалением всех вхождений $\overline{l}$ и заменой всех вхождений l на $\mathcal{T}$-*литерал*. Для двух литералов l_1 и l_2, соответствующих различным переменным, формула $F[l_1 = l_2]$ получается заменой всех вхождений l_1 на l_2 и всех вхождений $\overline{l_1}$ на $\overline{l_2}$. Отметим, что $F[l_1 = l_2]$ содержит меньшее количество переменных, чем F.

3. Правила преобразования

Корректное правило преобразования заменяет формулу F на более простую формулу F', такую что $F \in \text{X3SAT} \Leftrightarrow F' \in \text{X3SAT}$. Ниже мы приводим список используемых в нашем алгоритме правил преобразования. Каждое правило выполнимо за полиномиальное время и не увеличивает ни количество переменных, ни количество клозов формулы. Более того, каждое правило преобразования уменьшает длину формулы. Все правила, за исключением правила 4, взяты из статьи [5].

Для каждого правила преобразования мы предполагаем, что оно применяется к формулам, к которым все предыдущие правила не применимы, т.е. перед применением любого из правил преобразования наш алгоритм применяет правила с меньшими номерами, пока хотя бы одно из них применимо.

Правило 0: клоз, содержащий $\mathcal{T}$-*литерал*. Предположим, что F содержит клоз C, содержащий хотя бы один $\mathcal{T}$-*литерал*. Мы различаем следующие случаи:

1. Если $C = (\mathcal{T})$, то *правило* 0 заменяет F на $F - \{C\}$.
2. Если $C = (\mathcal{T}v)$, то *правило* 0 заменяет F на $F[\overline{v}]$.

3. Если $C = (\mathcal{T}vu)$, где $v \not\sim u$, то *правило* 0 заменяет F на $F[\overline{v}, \overline{u}]$.
4. Если $C = (\mathcal{T}vv)$, то *правило* 0 заменяет F на $F[\overline{v}]$.
5. Если $C = (\mathcal{T}\overline{v}v)$, то *правило* 0 заменяет F на пустой клоз (это означает, что F не является точно выполнимой).
6. Если C равен одному из следующих клозов: $(\mathcal{T}\mathcal{T})$, $(\mathcal{T}\mathcal{T}\mathcal{T})$, $(\mathcal{T}\mathcal{T}v)$, то *правило* 0 заменяет F на пустой клоз.

Правило 1: клоз, содержащий родственные литералы. Предположим, что F содержит клоз C, содержащий родственные литералы. Мы различаем следующие случаи:

1. Если C равен одному из следующих клозов: (xx), (xxx), то *правило* 1 заменяет F на пустой клоз.
2. Если $C = (x\overline{x})$, то *правило* 1 заменяет F на $F - \{C\}$.
3. Если $C = (xx\overline{x})$, то *правило* 1 заменяет F на $F[\overline{x}]$.
4. Если $C = (xxy)$, где $x \not\sim y$, то *правило* 1 заменяет F на $F[\overline{x}, y]$.
5. Если $C = (x\overline{x}y)$, где $x \not\sim y$, то *правило* 1 заменяет F на $F[\overline{y}]$.

Правило 2: удаление единичных клозов. *Правило 2* удаляет 1-клозы, т.е. оно заменяет F на $F[v]$, если F содержит клоз (v).

Правило 3: удаление бинарных клозов. *Правило 3* удаляет 2-клозы, т.е. оно заменяет F на $F[v = \overline{u}]$, если F содержит клоз (vu).

Правило 4: клоз, содержащий две уникальные переменные. *Правило 4* заменяет F на $F - \{C\}$, если C содержит хотя бы две уникальные переменные.

Правило 5: связанные литералы. Предположим, что F содержит клозы (ax_1y_1), (ax_2y_2), (ax_3y_3), а также клоз C, содержащий литералы, родственные литералам x_1, x_2, x_3. Мы называем литералы a, x_1, x_2, x_3 *связанными*, если C равен одному из клозов: $(x_1x_2x_3)$, $(\overline{x_1}\,\overline{x_2}x_3)$, $(\overline{x_1}\,\overline{x_2}\,\overline{x_3})$. В таком случае *правило* 5 заменяет F на $F[\overline{a}]$.

Правило 6: связанные клозы. Мы называем клозы C_1 и C_2 *связанными*, если они содержат хотя бы две общие переменные. Мы различаем следующие случаи:

1. Если $C_1 = (xyz), C_2 = (xyt)$, то *правило* 6 заменяет F на $F[z = t]$.
2. Если $C_1 = (xyz), C_2 = (\overline{x}yt)$, то *правило* 6 заменяет F на $F[\overline{y}, t = x, z = \overline{x}]$.

3. Если $C_1 = (xyz)$, $C_2 = (\overline{xy}t)$, то *правило* 6 заменяет F на $F[\overline{z}, \overline{t}, x = \overline{y}]$.

Правило 7: малая замкнутая подформула. Предположим, что F' – замкнутая подформула формулы F. Мы легко можем проверить, является ли F' точно выполнимой, если F' содержит, скажем, не более десяти переменных. Ясно, что $F \in \text{X3SAT} \Leftrightarrow (F - F') \in \text{X3SAT} \wedge F' \in \text{X3SAT}$. *Правило 7* заменяет F на $(F - F')$, если F' является точно выполнимой, и на пустой клоз в противном случае.

Правило 8: простая формула. Мы называем формулу F *простой*, если любой клоз $C \in F$, содержащий переменную, входящую в F хотя бы трижды, содержит также уникальную переменную. Точную выполнимость такой формулы мы можем проверить за полиномиальное время (см. [4]). Таким образом, *правило* 8 заменяет простую формулу F на клоз $(\mathcal{T})$, если F является точно выполнимой, и на пустой клоз в противном случае.

Лемма 1. *Описанные выше правила преобразования корректны.*

Доказательство. Корректность правил 0–7 очевидна. Корректность правила 8 доказана в [4]. □

4. Алгоритм

В данном разделе мы представляем алгоритм, решающий задачу X3SAT за время $O(2^{n/6.15})$, где n – количество переменных во входной формуле. Мы приводим алгоритм, после чего оцениваем его время работы и доказываем его коррекность, используя несколько лемм.

Алгоритм X3SAT

Вход: Формула F в 3-КНФ.
Выход: Если F является точно выполнимой, то *True*, иначе *False*.

Метод

1. Применять *правила* 0–8 к F, пока хотя бы одно из них применимо.
2. Если F не содержит клозов, выдать *True*.
3. Если F содержит пустой клоз, выдать *False*.

4. Если F содержит переменную, входящую в формулу по крайней мере четыре раза, выдать (X3SAT($F[a]$) $\vee$ X3SAT($F[\overline{a}]$)), где a – литерал, соответствующий этой переменной.
5. Если F содержит клозы $(\overline{a}x_1y_1)$, (ax_2y_2), (ax_3y_3), выдать (X3SAT($F[a]$) $\vee$ X3SAT($F[\overline{a}]$)).
6. Если F содержит клозы (ax_1y_1), (ax_2y_2), (ax_3y_3), $(x_3t_1t_2)$, (p_1p_2z), где $p_1 \sim x_1$, $p_2 \sim x_2$, $z \sim x_3$ или $z \sim y_3$, а также все литералы первых четырех клозов соответствуют различным переменным, выдать (X3SAT($F[x_3]$) $\vee$ X3SAT($F[\overline{x_3}]$)).
7. Если F содержит клозы (ax_1y_1), (ax_2y_2), (ax_3y_3), выдать (X3SAT($F[a]$) $\vee$ X3SAT($F[\overline{a}]$)).

После применения правил преобразования к F, алгоритм X3SAT производит два рекурсивных вызова для формул с меньшим количеством переменных (до тех пор, пока F не станет тривиальной) на одном из шагов 4–7.

Замечание 1. Ясно, что общее время работы алгоритма складывается из времени работы обоих рекурсивных вызовов и времени, потраченного на эти вызовы. Таким образом, время работы с точностью до полиномиального сомножителя равно количеству листьев в дереве рекурсии. Это количество, в свою очередь, равно максимальному из решений рекуррентных неравенств, соответствующих вершинам дерева рекурсии (см., напр., [2]).

Лемма 2. *На шаге 4 алгоритм производит расщепление, соответствующее в наихудшем случае рекуррентному неравенству $T(n) \leq T(n-9)+T(n-5)$ на количество листьев в дереве рекурсии.*

Доказательство. Рассмотрим формулу F после применения правил преобразования. Ясно, что F содержит только 3-клозы, F не содержит связанных клозов, клозов с двумя или тремя уникальными переменными и замкнутых подформул.

Предположим, что F содержит клозы $(\overline{a}x_1y_1)$, $(\overline{a}x_2y_2)$, (ax_3y_3), (ax_4y_4). Заметим, что после присваивания a значения *True* правила преобразования изменяют формулу следующим образом: $x_1 = \overline{y_1}$, $x_2 = \overline{y_2}$, $x_3 = 0$, $y_3 = 0$, $x_4 = 0$, $y_4 = 0$. Полученная формула не содержит переменных, соответствующих литералам a, x_1, x_2, x_3, y_3, x_4, y_4 (отметим, что все эти переменные различны, так как формула не содержит связанных клозов). Следовательно, количество переменных полученной формулы хотя бы на 7 меньше, чем количество переменных F. Аналогично, такая же оценка получается для формулы, полученной присваиванием a значения *False*. Таким образом, этому случаю соответствует рекуррентное неравенство $T(n) \leq T(n-7)+T(n-7)$. Аналогичным образом получаем рекуррентное неравенство $T(n) \leq T(n-8)+T(n-6)$ в случае, когда F содержит клозы $(\overline{a}x_1y_1)$, (ax_2y_2), (ax_3y_3), (ax_4y_4). Случаю, когда F содержит клозы (ax_1y_1), (ax_2y_2), (ax_3y_3), (ax_4y_4), соответствует рекуррентное неравенство $T(n) \leq T(n-9)+T(n-5)$. $\square$

Заметим, что если условие шага 4 не выполнено для формулы F, то F не содержит переменных, встречающихся в ней более трех раз. Таким образом, для шагов 5–7 переменная, соответствуюшая литералу a, встречается в формуле лишь в клозах, рассматриваемых на этих шагах.

Лемма 3. *На шаге* 5 *алгоритм производит расщепление, соответствующее в наихудшем случае рекуррентному неравенству* $T(n) \leq T(n-7)+T(n-6)$ *на количество листьев в дереве расщепления.*

Доказательство. Предположим, что F удовлетворяет условию шага 5. Заметим, что формула не содержит клозов с двумя уникальными переменными, поэтому, не умаляя общности, считаем, что x_1 не являтся уникальным.

Рассмотрим формулу $F[\overline{a}]$. В силу того, что литерал x_1 не является уникальным, формула F содержит клоз, содержащий литерал, родственный x_1. Пусть этим клозом будет $(x_1t_1t_2)$ (анализ случая, когда F содержит клоз $(\overline{x_1}t_1t_2)$, аналогичен и более прост). Применим все возможные правила преобразования к формуле $F[\overline{a}]$. Тогда будут сделаны такие замены: $x_1=0$, $y_1=0$, $x_2=\overline{y_2}$, $x_3=\overline{y_3}$, $t_1=\overline{t_2}$. Обозначим полученную формулу через F_1. Сейчас мы докажем, что F_1 содержит хотя бы на 6 переменных меньше, чем F. Для этого мы должны показать, что замена $t_1=\overline{t_2}$ уменьшает количество переменных, т.е. что замена не эквивалентна ни одной из замен $x_2=\overline{y_2}$, $x_3=\overline{y_3}$ и что по крайней мере один из литералов t_1, t_2 не является родственным ни к одному из литералов a, x_1, y_1. Ясно, что F_1 не содержит переменных, соответсвующих литералам a, x_1, y_1, x_2, x_3. Заметим, что t_1 и t_2 не являются родственными ни к однму из литералов a и y_1 (в противном случае F содержала бы связанные клозы).

Остается показать, что случай, когда $x_i \sim t_1$ и $y_i \sim t_2$ (где $i = 2$ или 3), невозможен. Это верно, так как в противном случае соответствующие клозы были бы связанными.

Теперь рассмотрим формулу $F[a]$. Опять же считаем, что x_1, x_2 не являются уникальными. Формула F содержит клоз $(x_2t_3t_4)$ или клоз $(\overline{x_2}t_3t_4)$, где литералы t_3 и t_4 не являются родственными литералам a и y_2. Предположим, F содержит клоз $(x_2t_3t_4)$ (оставшийся случай аналогичен). Примненим все возможные правила преобразования $F[a]$. Это изменит формулу следующим образом: $x_2 = 0$, $y_2 = 0$, $x_3 = 0$, $y_3 = 0$, $x_1 = \overline{y_1}$, $t_3 = \overline{t_4}$. Обозначим полученную формулу через F_2. В силу того, что F не содержит связанных клозов, хотя бы один из литералов t_3 и t_4 не является родственным ни x_1, ни y_1. По той же причине случай, когда каждый из литералов t_3 и t_4 является родственным одному из литералов x_2, y_2, x_3, y_3, невозможен. Следовательно, присваивание $t_3 = \overline{t_4}$ удаляет еще одну переменную формулы. F_1 содержит хотя бы на 7 переменных меньше, чем F.

Таким образом, этому случаю соответствует рекуррентное неравенство: $T(n) \leq T(n-7) + T(n-6)$. □

Лемма 4. *На шаге* 6 *алгоритм производит расщепление, соответствующее в наихудшем случае рекуррентному неравенству* $T(n) \leq T(n-9) + T(n-4)$ *на количество листьев в дереве расщепления.*

Доказательство. Рассмотрим формулу, удовлетворяющую условию шага 6, после применения правил преобразования. Пусть $C = (p_1p_2z)$.

Предположим, C содержит литерал, родственный x_3. В силу того, что F не содержит связанных литералов и не удовлетворяет условию шага 5, $C = (\overline{x_1}x_2x_3)$ или $C = (x_1\overline{x_2}x_3)$. Не умаляя общности, будем считать, что $C = (\overline{x_1}x_2x_3)$. Правила преобразования изменяют формулу $F[x_3]$ следующим образом: $x_2 = 0$, $x_1 = 1$, $a = 0$, $y_1 = 0$, $y_2 = 1$, $y_3 = 0$, $t_1 = 0$, $t_2 = 0$. Формула, полученная после применения правил преобразования к $F[\overline{x_3}]$, содержит по крайней мере на 4 переменные меньше, чем F. Таким образом, этому случаю соответствует рекуррентное неравенство $T(n) \leq T(n-9) + T(n-4)$.

Теперь рассмотрим случай, когда C содержит литерал, родственный y_3. Необходимо рассмотреть два случая: $C = (\overline{x_1}x_2y_3)$ или $C = (x_1x_2\overline{y_3})$ (случай $C = (x_1\overline{x_2}y_3)$ аналогичен, другие случаи невозможны по причине того, что формула не содержит связанных клозов). Пусть $C = (\overline{x_1}x_2y_3)$. Таким образом, F содержит клозы (ax_1y_1), (ax_2y_2), (ax_3y_3), $(x_3t_1t_2)$, $(\overline{x_1}x_2y_3)$. Правила преобразования изменяют формулу $F[x_3]$ следующим образом: $t_1 = 0$, $t_2 = 0$, $a = 0$, $y_3 = 0$, $y_1 = \overline{x_1}$, $y_2 = \overline{x_2}$, $x_1 = x_2$. Полученная формула содержит по крайней мере на 8 переменных меньше, чем F. Обозначим теперь через F_1 формулу, полученную из F присваиванием $x_3 = 0$. Заметим, что F_1 содержит клоз (ay_3). Значит, правило 3 заменяет F_1 на $F_1[y_3 = \overline{a}]$, в частности, оно заменяет клоз $(\overline{x_1}x_2y_3)$ на клоз $(\overline{x_1}x_2\overline{a})$. Клозы $(\overline{x_1}x_2\overline{a})$ и (ax_1y_1) являются связанными. Следовательно, правило 6 изменяет формулу следующим образом: $x_2 = 0, y_1 = 0, x_1 = \overline{a}$. Таким образом, правила преобразования производят такие изменения в формуле $F[\overline{x_3}]$: $t_1 = \overline{t_2}$, $x_2 = 0$, $y_1 = 0$, $y_3 = \overline{a}$, $x_1 = \overline{a}$, $y_2 = \overline{a}$. Полученная формула содержит хотя бы на 7 переменных меньше, чем F. Значит, этому случаю соответствует рекуррентное неравенство $T(n) \leq T(n-8) + T(n-7)$. Аналогично, в случае $C = (x_1x_2\overline{y_3})$ $F[x_3]$ содержит по крайней мере на 9 переменных меньше, чем F ($a = 0$, $y_3 = 0$, $t_1 = 0$, $t_2 = 0$, $x_1 = 0$, $x_2 = 0$, $y_1 = 1$, $y_2 = 1$), а $F[\overline{x_3}]$ содержит по крайней мере на 5 переменных меньше, чем F ($y_3 = \overline{a}$, $t_1 = \overline{t_2}$, $x_1 = \overline{a}$, $y_1 = x_2$). И соответствующее этому случаю рекуррентное неравенство имеет следующий вид: $T(n) \leq T(n-9) + T(n-5)$. □

Лемма 5. *На шаге 7 алгоритм производит расщепление, соответствующее в наихудшем случае рекуррентному неравенству* $T(n) \leq T(n-9) + T(n-4)$ *на количество листьев в дереве расщепления.*

Доказательство. Рассмотрим формулу F, содержащую клозы (ax_1y_1), (ax_2y_2), (ax_3y_3) (аналогично предыдущим леммам мы предполагаем, что никакое правило преобразования не применимо к F). Не умаляя общности, будем считать, что литералы x_1, x_2, x_3 не являются уникальными. Обозначим через S множество литералов, родственных одному из литералов a, x_1, y_1, x_2, y_2, x_3, y_3.

Предположим, F содержит клоз $(\overline{x_i}t_1t_2)$, где $t_1 \notin S$, $t_2 \notin S$. Очевидно, этому случаю соответствует рекуррентное неравенство $T(n) \leq T(n-9) + T(n-4)$.

Допустим теперь, что F содержит клоз $(x_it_1t_2)$, где $t_1 \notin S$, $t_2 \notin S$. В силу того, что F не удовлетворяет условию шага 6, в F есть еще один клоз, содержащий литерал из S и литерал не из S (обозначим этот литерал через w). (Напомним, что x_1, x_2, x_3 не являются уникальными и, следовательно, должны формировать хотя бы два таких клоза; если же они формируют всего один клоз литералов, родственных им, то клоз $(x_i t_1 t_2)$ удовлетворяет условию шага 6.) Ясно, что правила преобразования удаляют w из формулы $F[a]$. Таким образом, этому случаю также соответствует рекуррентное неравенство $T(n) \leq T(n-9) + T(n-4)$.

Рассмотрим теперь оставшийся случай, то есть случай, когда F не содержит клоза, состоящего из литерала из S и двух литералов не из S. В силу того, что переменные, соответствующие литералам множества S, не формируют замкнутую подформулу и что F не удовлетворяет условию шага 6, F содержит клоз (xyt), где $x \in S$, $y \in S$, $t \notin S$. Правила преобразования изменяют формулу $F[a]$ следующим образом: $x_1 = 0$, $y_1 = 0$, $x_2 = 0$,$y_2 = 0$, $x_3 = 0$, $y_3 = 0$, $t = 0$ или $t = 1$ (это зависит от литералов x и y). Заметим, что переменные, соответствующие литералу t и литералам множества S также не формируют замкнутую подформулу, следовательно, одна из этих переменных встречается в каком-то клозе вместе с другой переменной. Ясно, что правила преобразования удаляют эту новую переменную из $F[a]$. Таким образом, формула $F[a]$ содержит по крайней мере на 9 переменных меньше, чем F. Итак, этому случаю соответствует рекуррентное неравенство $T(n) \leq T(n-9) + T(n-4)$. $\square$

Теорема 1. *Алгоритм X3SAT корректно проверяет точную выполнимость формулы в 3-КНФ, содержащей n переменных, за время* $O(\tau^n)$, *где* $\tau = 1.11925266\ldots$ *– единственный положительный корень уравнения* $x^{-9} + x^{-4} = 1$.

Доказательство. *Корректность.* Корректность правил преобразования нашего алгоритма показана в лемме 1. После применения правил преобразования формула F содержит переменную, входящую в формулу по крайней мере три раза (в противном случае формула является простой). Пусть a – литерал, соответствующий этой переменной. Ясно, что F удовлетворяет условию шага 5 или условию шага 7 нашего алгоритма (если F содержит клозы $(\overline{a}x_1y_1)$, $(\overline{a}x_2y_2)$, (ax_3y_3), то она удовлетворяет условию шага 5, а если F содержит клозы $(\overline{a}x_1y_1)$, $(\overline{a}x_2y_2)$, $(\overline{a}x_3y_3)$, то она удовлетворяет условию шага 7).

Время работы. Время работы нашего алгоритма с точностью до полиномиального сомножителя равно количеству листьев в дереве рекурсии, которое является наибольшим решением рекуррентных неравенств, соответствующих вершинам этого дерева (см. замечание 1). В предыдущих леммах доказано, что в наихудшем случае количество листьев есть решение неравенства $T(n) \leq T(n-9) + T(n-4)$, т.е. τ^n. $\square$

Благодарности. Автор выражает глубокую благодарность Э. А. Гиршу за привлечение его внимания к задаче и за ряд ценных замечаний, позволивших значительно улучшить качество работы.

Литература

1. E. A. Hirsch and A. S. Kulikov, *A $2^{n/6.15}$-time algorithm for X3SAT*. — PDMI Preprint 13/2002,
Available from: `http://www.pdmi.ras.ru/preprint/2002/02-13.html`.
2. O. Kullmann and H. Luckhardt, *Deciding propositional tautologies: Algorithms and their complexity*. Manuscript (1997),
Available from: `http://cs-svr1.swan.ac.uk/~csoliver/`.
3. B. Monien, E. Speckenmeyer, and O. Vornberger, *Upper bounds for Covering Problems*. — Methods of Operations Research **43** 1981, 419–431.
4. S. Porschen, B. Randerath, and E. Speckenmeyer, *X3SAT is decidable in time $O(2^{n/5})$*. in: Proceedings of the Fifth International Symposium on the Theory and Applications of Satisfiability Testing (SAT 2002) 2002, pp. 231–235.
5. S. Porschen, B. Randerath, and E. Speckenmeyer, *Exact 3-Satisfiability is Decidable in Time $O(2^{0.16254n})$*. — Manuscript, 28th June (2002), 26p.
6. T. J. Schaefer, *The complexity of satisfiability problems*. — in: Conference Record of the Tenth Annual ACM Symposium on Theory of Computing, San Diego, California, 1–3 May 1978, ACM (1978), pp. 216-226.

Kulikov A. S. An upper bound $O(2^{0.16254n})$ for Exact 3-Satisfiability: a simpler proof.

The exact 3-satisfiability problem (X3SAT) is: given a Boolean formula in 3-CNF, find a truth assignment, such that exactly one literal in each clause is set to true. It is well-known that X3SAT is NP-complete. In this paper, we present an exact algorithm solving X3SAT in time $O(2^{0.162536n})$, where n is the number of variables. Our proof of this bound is slightly simpler than one of Porschen, Randerath, and Speckenmeyer. These proofs are independent (and algorithms are slightly different), though they are based on the same ideas appeared in the proof of the previous bound $O(2^{0.186916n})$ by the same authors.

С.-Петербургский
государственный университет
E-mail:ask@AK8436.spb.edu

Поступило 15 декабря 2002 г.