Записки научных
семинаров ПОМИ
Том 293, 2002 г.

А. С. Куликов, С. С. Федин

РЕШЕНИЕ ЗАДАЧИ О МАКСИМАЛЬНОМ СЕЧЕНИИ ЗА ВРЕМЯ $2^{|E|/4}$

1. Введение

Экспоненциальные оценки времени работы в наихудшем случае для NP-трудных задач. NP-трудные задачи возникают естественным образом в различных областях информатики. Известны различные экспоненциальные верхние оценки времени работы алгоритмов, решающих подобные задачи. Большинство полученных результатов касается SAT – задачи выполнимости пропозициональной формулы в конъюнктивной нормальной форме (*КНФ*). Как правило, алгоритмы, решающие SAT, используют локальный поиск или расщепление (обзор этих методов см., например, в [3]).

Задача MAX-CUT. В настоящей статье мы рассматриваем другую NP-трудную задачу, а именно, MAX-CUT. Задача состоит в нахождении для заданного неориентированного графа $G = (V, E)$ (в котором между вершинами могут быть кратные ребра) сечения максимального веса, то есть такого разбиения множества вершин V на два подмножества V_1 и V_2, что количество ребер $(u_1, u_2) \in E$, для которых $u_1 \in V_1$ и $u_2 \in V_2$, было бы максимально. Задача MAX-CUT была предложена Карпом [5] и давно известна как NP-полная. Более того, MAX-CUT является MAX-SNP-трудной задачей [7], то есть существует такая константа $c < 1$, что существование c-приближенного алгоритма для этой задачи влечет за собой равенство $P = NP$ [1].

Помимо теоретической важности, задача MAX-CUT имеет приложения в разработке схем и в статистической физике (Барахона и др. [2]). Подробный обзор приложений задачи MAX-CUT можно найти в статье Поляка и Тузы [6].

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

Верхние оценки в наихудшем случае для MAX-CUT. Простая оценка $O(2^{nd/(d+1)})$, где n – это количество вершин, а d – это максимальная степень вершины, дана в [8]. Недавно Грамм и др. [4] построили алгоритм, решающий MAX-CUT за время $O(2^{M/3})$, где M – количество ребер в графе. Эта оценка особенно интересна для графов с ограниченной степенью вершин: если $d = 3$, то MAX-CUT может быть решена за время $O(2^{n/2})$, а если $d = 4$ – то за время $O(2^{2n/3})$. В данной статье мы улучшаем эту оценку до $O(2^{M/4})$.

Основные идеи нашего алгоритма. Как и алгоритм из [4], наш алгоритм является типичным алгоритмом, использующим метод расщепления. Именно, он использует процедуру, называемую процедурой расщепления, которая сводит задачу для графа G к задаче для двух графов с меньшим количеством ребер. Перед расщеплением каждого из двух полученных графов алгоритм может упростить их, используя правила преобразования.

Алгоритм из [4] преобразует индивидуальную задачу MAX-CUT в индивидуальную задачу MAX-2-SAT и решает последнюю. В то время, как анализ этого алгоритма использует тот факт, что входная формула получена из индивидуальной задачи MAX-CUT, сам алгоритм этот факт не использует. В отличие от вышеупомянутого алгоритма наш алгоритм решает непосредственно задачу MAX-CUT. Так как соответствующие графам формулы в 2-КНФ имеют специфическую структуру, мы можем использовать правила преобразования, применимые лишь к этим формулам, вместо обычных правил преобразования, применимых к любым индивидуальным задачам MAX-2-SAT.

Наш алгоритм работает с небольшим обобщением задачи MAX-CUT: каждой вершине графа мы приписываем два числа Δ_1 и Δ_2, такие что Δ_i (где $i \in \{1,2\}$) обозначает вес, который будет добавлен к максимальному сечению, если эта вершина попадет в соответствующее множество V_i. Говоря неформально, эти два числа соответствуют единичным клозам формулы в 2-КНФ, полученным после расщепления индивидуальной задачи MAX-CUT, представленной формулой в 2-КНФ. Однако, несмотря на то, что единичные клозы не считаются в алгоритме из [4], они не могут быть удалены в MAX-2-SAT (и это накладывает некоторые ограничения на правила упрощения формулы). Напротив, Δ_1 и Δ_2 позволяют упрощать графы с Δ_i-ми аналогично обычным графам. В частности, наш алгоритм использует новое правило преобразования, которое удаляет конечные вершины (в графе с Δ_i-ми). Это правило также может быть сформулировано в терминах задачи MAX-2-SAT: оно удаляет переменную, появляющуюся лишь в двух 2-клозах равного веса (эти два 2-клоза соответствуют ребру) и в нескольких 1-клозах. Это правило помогает нам находить максимальное сечение в некоторых специальных графах. Например, мы за полиномиальное время можем найти максимальное сечение в графе (с Δ_i-ми), в котором каждая вершина имеет степень 2.

Мы покажем, что почти на каждом шаге алгоритма можно сделать расщепление, соответствующее рекуррентному неравенству $T(|E|) \leq 2 \cdot T(|E|-4)+\mathrm{poly}(|E|)$ для времени работы (где $T(|E|)$ обозначает максимальное возможное время, которое алгоритм будет работать на графе с $|E|$ ребрами). Единственным графом, в котором такого расщепления не существует, является граф, в котором каждая вершина имеет ровно трех соседей, и не содержащий кратных ребер. Мы докажем, что такой случай может появиться не более одного раза на любой из веток дерева расщепления.

Организация статьи. Статья организована следующим образом: в разделе 2.1 даются основные определения. В разделах 2.2 и 2.3 мы описываем процедуры инициализации и расщепления. В разделе 2.4 описываются правила преобразования, которые мы используем. И наконец, основной алгоритм и его анализ даны в разделе 2.5.

2. Наш результат

2.1. Основные определения. Для представления графа с кратными ребрами мы рассматриваем граф, в котором каждому ребру присвоен целочисленный вес (таким образом, в дальнейшем мы предполагаем, что между любыми двумя вершинами существует не более одного ребра). При помощи $|F|$ мы обозначаем суммарный вес ребер множества F. Пусть $W(u,v)$ обозначает вес ребра между вершинами u и v. $W(v)$ обозначает суммарный вес ребер между вершиной v и ее соседями, а $W(G)$ – вес графа $G=(V,E)$: $W(G)=\frac{1}{2}\sum_{v \in V} W(v)$. Через $Deg(v)$ мы обозначаем *степень* вершины v, то есть количество ее соседей.

Вес сечения для разбиения (V_1,V_2) множества вершин V – это сумма весов тех ребер (u_1,u_2), для которых $u_1 \in V_1$, $u_2 \in V_2$. Разбиение (V_1, V_2) называется *оптимальным*, если оно соответствует сечению максимального возможного веса.

В начале нашего алгоритма множества V_1 и V_2 пусты. На каждом шаге мы переносим некоторые вершины из множества V в одно из множеств разбиения (при удалении вершины из множества V мы также удаляем все смежные с ней ребра из множества E). Таким образом, некоторые ребра добавляются в сечение. $CurrentCut$ – это переменная, обозначающая вес этого сечения.

Мы используем Δ_1 и Δ_2 – целочисленные функции на множестве вершин. Интуитивный смысл этих функций заключается в следующем: $\Delta_1(v)$ (или $\Delta_2(v)$) – это число, которое будет добавлено к текущему значению сечения при добавлении вершины v во множество V_1 (соответственно V_2). Эти функции помогают нам сохранять информацию о весе удаленных ребер следующим образом. Предположим, граф содержит вершины u и v, такие что $(u, v) \in E$, и на некотором шаге нашего алгоритма мы переносим v из V в V_1. В этом случае мы должны запомнить, что если мы перенесем u в V_2, то $W(u, v)$ будет добавлено к весу сечения. С этой целью мы изменяем $\Delta_2(u)$ таким образом: $\Delta_2(u) = \Delta_2(u) + W(u, v)$.

2.2. Процедура инициализации. Мы предполагаем, что каждый появляющийся в нашем алгоритме граф G состоит из множества вершин $G.V$, множества ребер $G.E$ (каждому ребру присвоен целочисленный вес), переменной $G.CurrentCut$ и двух функций $G.\Delta_1$ анд $G.\Delta_2$ (из описания алгоритма легко следует, что на каждом его шаге $0 \le \Delta_1(v), \Delta_2(v) \le |E|$ для любой вершины v). Вес сечения (V_1, V_2) для такого графа равен $\sum_{(u,v)\in E, u\in V_1, v\in V_2} W(u, v) + \sum_{u\in V_1} \Delta_1(u) + \sum_{u\in V_2} \Delta_2(u)$. Перед применением алгоритма к графу G мы применяем следующую процедуру:

Процедура *Init*(граф G)

$G.CurrentCut = 0$

FOR each $u \in G.V$

  $\Delta_1(u) = 0, \Delta_2(u) = 0$

end FOR

2.3. Расщепление. При помощи расщепления мы сводим задачу для графа G к задаче для двух графов $G[v \in V_1]$ и $G[v \in V_2]$ (где $v \in V$). Мы производим расщепление, используя следующую процедуру:

Процедура *Split*(граф G, вершина v)
FOR $i = 1, 2$
(a) $G[v \in V_i].CurrentCut = G.CurrentCut + \Delta_i(v)$
(b) $G[v \in V_i].E = G.E$
(c) $G[v \in V_i].V = G.V - \{v\}$
(d) FOR each $u \in G[v \in V_i]$
$G[v \in V_i].\Delta_1(u) = G.\Delta_1(u)$,
$G[v \in V_i].\Delta_2(u) = G.\Delta_2(u)$
end FOR
(e) FOR each $u \in G.V$
IF $(u, v) \in G.E$
$G[v \in V_i].\Delta_{3-i}(u) = G[v \in V_i].\Delta_{3-i}(u) + W(u, v)$,
$G[v \in V_i].E = G[v \in V_i].E - \{(u, v)\}$
end IF
end FOR
end FOR
return $G[v \in V_1]$ and $G[v \in V_2]$

Процедура расщепления удаляет вершину v и все ее ребра из графа и сохраняет в $\Delta_i(u)$ информацию о том, что если u и v принадлежат разным множествам разбиения, то (u, v) входит в сечение. Очевидно, $MaxCut(G) = \max\{MaxCut(G[v \in V_1]), MaxCut(G[v \in V_2])\}$. Отметим, что $W(G[v \in V_1]) = W(G[v \in V_2]) = W(G) - W(v)$.

2.4. Правила преобразования. Для нахождения максимального сечения достаточно рассмотреть все сечения и выбрать из них оптимальное. Таким образом, для каждой вершины v мы должны рассмотреть два случая: $v \in V_1$ и $v \in V_2$. Но в некоторых ситуациях можно доказать, что по крайней мере в одном оптимальном разбиении v принадлежит одному (конкретному) из этих множеств. В такой ситуации нет необходимости в разборе случая, когда v принадлежит другому множеству. *Правила преобразования* помогают нам обрабатывать такие вершины.

Корректное правило преобразования заменяет граф G на более простой граф G', такой что $MaxCut(G) = MaxCut(G')$, то есть корректное правило преобразования сохраняет вес максимального сечения. Каждое правило преобразования выполнимо за полиномиальное время и не увеличивает суммарного веса ребер.

Ниже мы приводим список используемых в нашем алгоритме правил преобразования и доказываем их корректность.

Правило 1: гарантированное значение. Данное правило изменяет граф следующим образом:

FOR each $u \in G.V$

$$G.CurrentCut = G.CurrentCut + min\{\Delta_1(u), \Delta_2(u)\}$$
$$G.\Delta_1(u) = G.\Delta_1(u) - min\{\Delta_1(u), \Delta_2(u)\}$$
$$G.\Delta_2(u) = G.\Delta_2(u) - min\{\Delta_1(u), \Delta_2(u)\}$$

end FOR

Лемма 1. Правило 1 *корректно.*

Доказательство. Заметим, что в листе дерева расщепления каждая вершина должна находиться в одном из множеств V_1 или V_2. Таким образом, когда вершина v добавляется в одно из множеств разбиения, по крайней мере $min\{\Delta_1(v), \Delta_2(v)\}$ прибавляется к $CurrentCut$.

Правило 2: изолированная вершина. Если граф содержит вершину v веса 0, **Правило 2** преобразует граф таким образом: $G.V = G.V - \{v\}$, $G.CurrentCut = G.CurrentCut + \max\{\Delta_1(v), \Delta_2(v)\}$. Корректность этого правила очевидна.

Правило 3: конечная вершина. Предположим, что G содержит вершины v и u, такие что $(u, v) \in E$ и $Deg(v) = 1$. Заметим, что хотя бы одно из чисел $\Delta_1(v)$ и $\Delta_2(v)$ равно 0 (в противном случае применим **правило 1**). Не умаляя общности, будем считать, что $\Delta_2(v) = 0$.

Если $\Delta_1(v) \geq W(u, v)$, то **правило 3** преобразует граф следующим образом: $\Delta_2(u) = \Delta_2(u) + W(u, v)$, $CurrentCut = CurrentCut + \Delta_1(v)$, $E = E - \{(u, v)\}$, $V = V - \{v\}$.

Если же $\Delta_1(v) < W(u, v)$, то так: $\Delta_2(u) = \Delta_2(u) + \Delta_1(v)$, $CurrentCut = CurrentCut + W(u, v)$, $E = E - \{(u, v)\}$, $V = V - \{v\}$.

Лемма 2. Правило 3 *корректно.*

Доказательство. Рассмотрим случай, когда $\Delta_1(v) \geq W(u, v)$. Предположим, в каком-то из оптимальных разбиений $v \in V_2$. Перенесем тогда v из V_2 в V_1. Ясно, что при этом вес сечения не уменьшился, то есть полученное сечение также является оптимальным. Это означает, что мы не должны рассматривать случай, когда $v \in V_2$. Таким образом, мы просто сохраняем в графе информацию о том, что (v, u) будет добавлено к сечению, если $u \in V_2$. После чего мы прибавляем $\Delta_1(v)$ к $CurrentCut$ и удаляем v из графа.

Предположим теперь, что $\Delta_1(v) < W(u, v)$. Заметим, что если существует оптимальное разбиение, в котором u и v принадлежат одному и тому же множеству V_i, то при переносе v из V_i в V_{3-i} мы увеличим вес сечения. Следовательно, любое оптимальное сечение содержит ребро (u, v). Таким образом, мы сохраняем информацию о том, что если в оптимальном разбиении вершина u принадлежит множеству V_2, то v будет в V_1 и, следовательно, $\Delta_1(v)$ будет прибавлено к $CurrentCut$. Далее мы прибавляем $W(u, v)$ к $CurrentCut$ и выкидываем v из графа.

Правило 4: малая компонента связности. Мы легко можем найти оптимальное сечение для малой (скажем, не более, чем с четырьмя ребрами) компоненты связности графа G. Таким образом, **правило 4** выкидывает такую компоненту из G и прибавляет вес максимального сечения этой компоненты к $G.CurrentCut$.

2.5. Алгоритм. В этом разделе мы представляем Алгоритм MaxCut, который решает задачу MAX-CUT за время $\mathrm{poly}(|E|) \cdot 2^{|E|/4}$, где $|E|$ обозначает суммарный вес ребер во входном графе. Мы приводим сам алгоритм и далее доказываем его корректность и оцениваем время его работы.

Алгоритм MaxCut

Вход: Граф G после применения Процедуры Init.

Выход: Вес максимального сечения для G.

Метод

1. Применять **правила 1–4** к G, пока хотя бы одно из них применимо.
2. Если G не содержит ребер, выдать $CurrentCut$.
3. Если G содержит нетривиальную компоненту связности G_1, выдать $MaxCut(G_1) + MaxCut(G - G_1)$.
4. Если G содержит вершину v веса хотя бы 4, то выдать[1] $\max\{MaxCut(G[v \in V_1]), MaxCut(G[v \in V_2])\}$.

---

[1]В дальнейшем через $G[v \in V_1]$ и $G[v \in V_2]$ мы обозначаем графы, построенные Процедурой Split.

5. Если G содержит вершины u и v, такие что $W(u,v)=2$, выдать $\max\{MaxCut(G_1), MaxCut(G_2)\}$, где G_1 и G_2 – графы, полученные из $G[v \in V_1]$ и $G[v \in V_2]$, соответственно, после применения правил преобразования.
6. Если каждая вершина G имеет степень 2, выбрать любую вершину v и выдать $\max\{MaxCut(G_1), MaxCut(G_2)\}$, где G_1 и G_2 – графы, полученные из $G[v \in V_1]$ и $G[v \in V_2]$, соответственно, после применения правил преобразования.
7. Если G содержит вершины u, v, такие что $Deg(v)=3$, $Deg(u) = 2$, $(u,v) \in G.E$, выдать $\max\{MaxCut(G_1), MaxCut(G_2)\}$, где G_1 и G_2 – графы, полученные из $G[v \in V_1]$ и $G[v \in V_2]$ соответственно, после применения правил преобразования.
8. Если каждая вершина G имеет степень 3, а каждое ребро имеет вес 1, выдать $\max\{MaxCut(G[v \in V_1]), MaxCut(G[v \in V_2])\}$ для произвольной вершины v.

Теорема 1. *Для заданного неориентированного графа $G=(V,E)$, в котором ребрам присвоены целочисленные веса, алгоритм MaxCut находит вес макцимального сечения для G за время* $\mathrm{poly}(|E|)\cdot 2^{|E|/4}$, *где $|E|$ – это суммарный вес ребер графа G.*

Доказательство.

Корректность. Корректность правил преобразования показана в разделе 2.4. Корректность шагов алгоритма MaxCut очевидна. Ниже мы покажем, что любой граф после применения правил преобразования удовлетворяет одному из условий шагов 2–8.

Предположим, что граф G не удовлетворяет ни одному из условий шагов 2–7. В таком случае G связен. G не содержит ребер веса хотя бы 3, так как в нем нет вершин веса хотя бы 4 и конечных вершин. Также в G нет ребер веса 2, так как он не удовлетворяет условию шага 5. Таким образом, каждое ребро имеет вес 1, а каждая вершина имеет степень 2 или 3. Если степень каждой вершины равна 2, то G удовлетоворяет условию шага 6. Если G содержит как вершины степени 2, так и вершины степени 3, то в нем есть вершины u и v, такие что $Deg(u)=2, Deg(v)=3, (u,v)\in E$ (так как G связен). В этом случае G удовлетворяет условию шага 7. В оставшемся случае каждая вершина графа G имеет степень 3, то есть G удовлетворяет услвоию шага 8.

Время работы. После применения правил преобразования к G алгоритм MaxCut делает (до тех пор, пока G не станет тривиальным) два рекурсивных вызова для графов с меньшими суммарными весами ребер на одном из шагов 3–8. Ясно, что 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 вершин (или листьев) в дереве рекурсии.

Теперь покажем, что на шагах 3–7 мы можем сделать рекурсивный вызов для графов G_1 и G_2, таких что $|G.E| - |G_1.E| \geq 4, |G.E| - |G_2.E| \geq 4$ (это соответствует рекуррентному неравенству $L(M) \leq 2 \cdot L(M-4)$ на количество листьев в дереве расщепления, где $L(m)$ обозначает максимальное возможное количество листьев в дереве расщепления для графа с m ребрами, построенном алгоритмом MaxCut). Это утверждение очевидно для шагов 3 и 4. Если G удовлетворяет условиям шагов 5 или 7, тогда $W(v) = 3$ в графе G (в силу того, что в G нет конечных вершин и вершин веса хотя бы 4) и $W(u) = 1$ в графах $G[v \in V_1]$ и $G[v \in V_2]$ (т.е. u является конечной, и **правило 2** удаляет ее из этих графов). Таким образом, для обоих графов G_1 и G_2, получающихся из графа G на шагах 5 или 7, $|G_i.E| = |G.E| - 4$ (где $i = 1, 2$) и неравенство на время работы выполняется. Если G удовлетворяет условию шага 6, то вершины графа образуют цикл. Тогда после расщепления мы можем удалить все оставшиеся вершины посредством **правила 2**.

Докажем теперь, что если G удовлетворяет условию шага 8 и не удовлетворяет ни одному из условий других шагов, то все получающиеся при дальнейших вычислениях графы этому условию удовлетворять не будут. Предположим противное: каждая вершина графа G имеет степень 3, и после применения некоторых правил преобразования и шагов нашего алогритма G преобразован в G_1, который удовлетворяет тому же условию. Нетрудно видеть, что ни правила преобразования, ни шаги нашего алгоритма не увеличивают степени вершин. Значит, G_1 был компонентой связности в G, что противоречит факту, что G был связен.

Как мы показали выше, каждая ветка дерева расщепления содержит не более одного графа, удовлетворяющего условию шага 8. Следовательно, количество листьев этого дерева составляет $O(2^{|E|/4})$.

Замечание 1. Ясно, что полиномиальный сомножитель, появляющийся в оценке времени работы нашего алгоритма, является линейным в unit-cost RAM модели и имеет небольшую степень в любой другой разумной модели вычислений.

Благодарности. Авторы признательны Э. А. Гиршу за привлечение их внимания к задаче и за ценные замечания. Многочисленные комментарии анонимных рецензентов также значительно улучшили качество статьи.

Литература

1. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof verification and hardness of approximation problems.* — in: Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science (1992), 14–23.
2. F. Barahona, M. Grotschel, M. Jünger, and G. Reinelt, *An application of combinatorial optimization to statistical physics and circuit layout design.* — Operations Research, **36** (1988), 493–513.
3. М. А. Всемирнов, Э. А. Гирш, Е. Я. Данцин, С. В. Иванов, *Алгоритмы для пропозициональной выполнимости и верхние оценки их сложности.* — Зап. научн. семин. ПОМИ, **277** (2001), 14–46.
4. J. Gramm, E. A. Hirsch, R. Niedermeier, and P. Rossmanith, *New worst-case upper bounds for MAX-2-SAT with application to MAX-CUT.* — Technical Report 00-037, Electronic Colloquium on Computational Complexity (2000). To appear in *Discrete Applied Mathematics.*
5. R. M. Karp, *Reducibility among combinatorial problems.* — in: R. Miller and J. Thatcher, eds., Complexity of Computations, Plenum Press, New York (1972), 85–103.
6. S. Poljak and Z. Tuza, *Maximum cuts and large bipartite subgraphs.* — DIMACS Series in Discrete Mathematics and Theoretical Computer Science, **20** (1995), 181–244.
7. C. H. Papadimitriou and M. Yannakakis, *Optimization, approximation, and complexity classes.* — J. Computer and System Sciences, **43** (1991) 425–440.
8. U. Zwick, *Сообщение авторам статьи* [4] (2000).

Fedin S. S., Kulikov A. S. A $2^{|E|/4}$-time Algorithm for MAX-CUT.

In this paper we present an exact algorithm solving MAX-CUT in time $\mathrm{poly}(|E|) \cdot 2^{|E|/4}$, where $|E|$ is the number of edges (there can be multiple edges between two vertices). This bound improves the previously known bound $\mathrm{poly}(|E|) \cdot 2^{|E|/3}$ of Gramm et al. (2000).

Санкт-Петербургский
государственный университет
E-mail: ask@AK8436.spb.edu
fss@SF8469.spb.edu

Поступило 8 декабря 2002 г.