Approximations for the maximum acyclic subgraph problem

Rafael Hassin *, Shlomi Rubinstein

Department of Statistics and Operations Research, School of Mathematical Sciences, Tel-Aviv University, Tel-Aviv 69978, Israel

Communicated by S. Zaks; received 9 May 1993; revised 9 February 1994

Перевод

Егор Вашкевич

физтех-школа ФПМИ МФТИ

Приближения для задачи о максимальном ациклическом подграфе

Рефаэль Хассин *, Шломи Рубинштейн

Факультет статистики и операционных исследований, Школа математических наук, Тель-Авивский университет, Тель-Авив 69978, Израиль

Передано С. Заксом; получено 9 мая 1993; пересмотрено 9 Февраля 1994

Аннотация

Пусть дан граф $G = (V, E)$, тогда *задача поиска максимального ациклического подграфа* состоит в вычислении подмножества рёбер A' максимального размера или общего веса так, чтобы $G' = (V, A')$ был ациклическим. Мы обсудим несколько алгоритмов приближённых решений для этой задачи. Нашим главным результатом будет алгоритм, работающий за $O(|A| + d_{max}^3)$, который выдает решение с долей как минимум $\frac{1}{2} + \Omega(\frac{1}{\sqrt{d_{max}}})$ от числа рёбер в оптимальном решении. Здесь d_{max} — максимальная степень вершины в G.

Ключевые слова: Анализ алгоритмов; Комбинаторные задачи

Введение

Пусть дан граф $G = (V, A)$, $V = \{1, \ldots, n\}$ с весами рёбер $w_{ij} > 0, (i, j) \in A$. *Задача поиска максимального ациклического подграфа* состоит в том, чтобы найти подмножество $A' \subset A$ такое, что $G' = (V, A')$ является ациклическим и значение $w(A') = \sum_{(i,j) \in A'} w_{ij}$ максимально. В альтернативной постановке этой задачи (поиск

минимального множества обратных рёбер[1]) требуется найти подмножество минимального веса $A'' \subset A$ такое, что каждый (направленный) цикл G содержит по крайней мере одно ребро в A''. Задача является NP-сложной [11]. Она принадлежит к классу задач "удаления рёбер"[17, 21]. Было показано, что она является полной для класса задач оптимизации перестановок, MAX SNP$[\pi]$, определенном в [19], которые можно приближённо вычислить с фиксированной погрешностью.

Задача полиномиально разрешима, когда G планарный ([4,12,15] и глава 8.4 в [8]). Наилучшая сложность этих алгоритмов — $O(n^3)$ [5] и $O(n^{\frac{5}{2}} \log(nW))$, где W — наибольшее значение веса ребра (предполагается, что веса являются целыми) [6]. Задача также полиномиально разрешима для более общего класса $K_{3,3}$-свободных графов [18] и классов приводимых потоковых графов [20] и слабо ациклических графов [7]. Вариация задачи, в которой целью является минимизация наибольшего значения выходной степени вершины в подграфе (V, A''), может быть решена за линейное время [16].

Задача имеет множество применений, таких как упорядочение альтернатив путем группового голосования, определение иерархии секторов экономики, определение родственных связей, анализ систем с обратной связью и определенных проблем с планированием [3,14,20]. Флуд [3] использовал связь задачи с квадратичным присваиванием для разработки эффективного метода ветвей и границ. Юнгер [14] изучал многогранник ациклического подграфа.

Задачи о максимальном ациклическом подграфе и о минимальном множестве обратных рёбер эквивалентны по отношению к их оптимальному решению. Однако полиномиальные аппроксимации с ограниченной ошибкой известны только для постановки проблемы с максимальным ациклическим подграфом. Простейший алгоритм следующий [9]: Пусть $A_1 = \{(i,j) \in A \mid i < j\}$, $A_2 = \{(i,j) \in A \mid i > j\}$. Очевидно, что и (V, A_1), и (V, A_2) являются ациклическими и, поскольку $A1 \cup A2 = A \Rightarrow \max\{w(A_1), w(A_2)\} \geqslant 0.5w(A)$. Следовательно, это 0.5-приближение к задаче. Заметим, что алгоритм имеет линейную сложность.

Корте и Хаусманн [13] доказали, что жадный алгоритм (т.е. построение решения путем многократного выбора ребра максимального веса, которое не образует направленный цикл с уже выбранными ребрами) не гарантирует какой-либо фиксированной ошибки.

Более сложный алгоритм для невзвешенной постановки задачи был предложен Бергером и Шором [2]. Они отмечают, что без ограничения общности мы можем предположить, что G не имеет циклов длины 2, поскольку любая оценка, которая может быть получена при этом предположении, также может быть получена без него путем простой модификации алгоритма. Затем они разрабатывают алгоритм, создающий ациклический подграф из как минимум $(\frac{1}{2} + \Omega(\frac{1}{\sqrt{d_{max}}}))|A|$ рёбер где d_{max} — максимальная степень вершины G. Даже когда существуют циклы длины два, решение содержит по меньшей мере $(\frac{1}{2} + \Omega(\frac{1}{d_{max}}))$ долю рёбер от их количества в оптимальном решении. Алгоритм работает за время $O(|A||V|)$.

В этой статье мы исследуем различные алгоритмы для решения задачи. Наш основной вклад — это алгоритм для задачи без весов, который гарантирует оценку, аналогичную той, которая была получена Бергером и Шором, но с временной сложностью $O(|A| + d_{max}^3)$, которая лучше, чем $O(|A||V|)$ в определенных случаях.

Получение решения из перестановки

Назовем ациклический подграф G максимальным, если он строго не содержится в другом ациклическом подграфе G. Поскольку мы предполагаем, что веса рёбер положительны, максимальный ациклический подграф также максимален. Перестановка π из $\{1, \ldots, n\}$ индуцирует ациклический подграф $G_\pi = (V, A_\pi)$, где $A_\pi = \{(i,j) \in A \mid \pi(i) < \pi(j)\}$. Заметим, что G_π может быть не максимальным, если он не связен. Однако каждый максимальный ациклический подграф G индуцируется некоторой перестановкой. (Всегда можно перенумеровать вершины ациклического графа так, чтобы каждое ребро (i,j) в нем удовлетворяло $i < j$, и, если граф максимален, тогда он индуцируется этой перестановкой.) Следовательно, задача о максимальном ациклическом подграфе — это в точности задача вычисления перестановки, индуцированный подграф которой имеет максимальный вес.

[1] В оригинале — minimum feedback arc set problem (прим. перев.).

Для начала напишем псевдо-код алгоритма, который генерирует перестановку π, такую что $w(A_\pi) > 0.5w(A)$. Для $i \in V$ и $S \subset V$, пусть $w_i^{in}(S) = \sum\limits_{j \in S} w_{ij},\ w_i^{out}(S) = \sum\limits_{j \in S} w_{ij}$.

Алгоритм 1.
1) Положим $S = V, l = 1, u = n$.
2) Возьмём $i \in S$. Присвоим $S \leftarrow S \setminus \{i\}$. Если $w_i^{in}(S) \leqslant w_i^{out}(S)$, присвоим $\pi(i) = l,\ l \leftarrow l + 1$. Если $w_i^{in}(S) > w_i^{out}(S)$, присвоим $\pi(i) = u,\ u \leftarrow u - 1$.
3) Если $u \geqslant l$, выполнить шаг 2. Иначе остановиться и вернуть π.

Вес рёбер, сохраняемых алгоритмом, составляет по меньшей мере половину от общего веса A, поскольку это свойство выполняется на каждой итерации относительно рёбер, инцидентных вершине i в подграфе, индуцированном S.

Рандомизация

Трудность в получении лучшей оценке ошибки, чем 0.5, в случае применения простых конструктивных алгоритмов возникает, когда $w_i^{in}(S) \approx w_i^{out}(S)$ для большого количества вершин. Мы постараемся преодолеть трудности, связанные с такой ситуацией, и улучшить оценку для задачи с невзвешенным графом, считая максимальную степень вершины графа параметром.

Начнём с представления рандомизированного алгоритма. Он отличается от предложенного в [2], но обеспечивает аналогичную оценку. Без ограничения общности можем считать, что G не содержит циклов длины 2. Если он содержит циклы длины 2, как это сделано в [2], то они могут быть временно удалены до выполнения алгоритма. Любая перестановка приведет к получению графа ровно с одним ребром из каждого такого цикла.

Алгоритм 2.
1) Разделить V на два подмножетсва V_1 и V_2, присваивая вершину к каждому множеству случайно с вероятностью 0.5. Пусть $A_r = \{(i,j) |\ i,j \in V_r\}$. Выполнить шаг 2 для для $r = 1, 2$.
2) Создать перестановку π_r вершин в V_r, применяя алгоритм 1 к (V_r, A_r). Вершины выбираются в порядке возрастания их индексов.
3) Определить конечную перестановку, выбирая между (π_1, π_2) и (π_2, π_1) ту перестановку, которая индуцирует подграф с бóльшим количеством рёбер.

Теорема 3. Пусть APX — матожидание числа рёбер в решении, полученном алгоритмом 2. Тогда

$$APX = \left(0.5 + \Omega\left(\frac{1}{\sqrt{d_{max}}}\right)\right)|A|$$

Доказательство. □ Рассмотрим вершину $i \in V_r$. Обозначим:

$$\begin{aligned} D_i^{in} &= |(j,i) \in A : j > i|, \\ D_i^{out} &= |(i,j) \in A : j > i|, \\ d_i^{in} &= |(j,i) : j \in V_r, j > i|, \\ d_i^{out} &= |(i,j) : j \in V_r, j > i|. \end{aligned}$$

Случайная величина d_i^{in} имеет биномиальное распределение с параметрами $(0.5, D_i^{in})$, так как для любого ребра, инцидентного i, вероятность того, что его второй конец также будет из V_r, составляет 0.5. Аналогично d_i^{out} имеет биномиальное распределение с параметрами $(0.5, D_i^{out})$.

Без ограничения общности будем считать, что $D_i^{in} \geqslant D_i^{out}$. Для $a \geqslant 0$

$$P \equiv \Pr(\left|d_i^{in} - d_i^{out}\right| \geqslant a) \geqslant \Pr(d_i^{in} - d_i^{out} \geqslant a).$$

По нашему предположению G не содержит циклов длины 2, d_i^{in} и d_i^{out} — независимые случайные величины. Рассмотрим 2 независимые случайные величины X_1, X_2, имеющие биномиальное распределение с параметрами $\left(0.5, D_i^{in}\right)$. Тогда

$$\begin{aligned} P &\geqslant Pr\left(X_1 - X_2 \geqslant a\right) \\ &\geqslant Pr\left(X_1 \geqslant 0.5D_i^{in} + a, X_2 \leqslant 0.5D_i^{in}\right) \\ &= 0.5Pr\left(X_1 \geqslant 0.5D_i^{in} + a\right), \end{aligned}$$

где первое следует из предположения $D_i^{in} \geqslant D_i^{out}$. Пусть a — дисперсия X_1, $a = \frac{1}{2}\left(D_i^{in}\right)^{\frac{1}{2}}$. Получаем, что для некоторой константы $\beta > 0$[1]:

$$Pr(\left|d_i^{in} - d_i^{out}\right| \geqslant a) \geqslant \beta$$

Пусть $D_i = D_i^{in} + D_i^{out}$, тогда $D_I^{in} \geqslant \frac{D_i}{2}$ и $a = \Omega(\sqrt{D_i}) = \Omega\left(\frac{D_i}{\sqrt{D_i}}\right) = \Omega\left(\frac{D_i}{\sqrt{d_{max}}}\right)$.

Шаг 2 присваивает вершине i следующую более низкую или более высокую позицию в перестановке в соответствии со знаком разности $d_i^{in} - d_i^{out}$. Общее число рёбер, порождённое π_r:

$$\begin{aligned} \sum_{i \in V_r} \max(d_i^{in}, d_i^{out}) &= \sum_{i \in V_r} \left(\left(\frac{d_i^{in} + d_i^{out}}{2}\right) + \frac{1}{2}\left|d_i^{in} - d_i^{out}\right|\right) \\ &= 0.5|A_r| + 0.5 \sum_{i \in V_r}^{\left|d_i^{in} - d_i^{out}\right|}. \end{aligned}$$

Приходим к заключению, то матожидание числа рёбер, порождённых π_r, удовлетворяет соотношению

$$ASX_r = 0.5|A_r| + \Omega\left(\sum_i \frac{D_i}{\sqrt{d_{max}}}\right).$$

Поскольку $\sum\limits_i D_i = |A|$, то

$$APX_1 + APX_2 = 0.5\left(|A_1| + |A_2|\right) + \Omega\left(\frac{|A|}{\sqrt{d_{max}}}\right)$$

.

Шаг 3 алгоритма упорядочивает 2 частичные перестановки так, что итоговый индуцированный подграф содержит как минимум половину рёбер, соединяющих вершины из V_1 с вершинами из V_2. Отсюда следует утверждение теоремы. ■

Замечание 4. Матожидание числа рёбер, получаемых в результате алгоритма, также составляет $\left(0.5 + \Omega\left(\frac{1}{\bar{d}}\right)\right)|A|$. где $\bar{d} = \frac{2|A|}{n}$ — средняя степень в G. Эта оценка получается исходя из того, что матожидание числа рёбер, в дополнение к $0.5|A|$, полученном на шаге 3, равно как минимум одному для каждой вершины. Эта оценка может быть лучше, чем указанная в теореме, когда средняя степень ограничена, а максимальная — нет.

Дерандомизация

Теперь мы опишем детали, необходимые для эффективного выполнения дерандомизации с помощью метода условных вероятностей (см., например, [1]). Используя этот метод, мы превращаем наш рандомизированный алгоритм в детерминированный с той же временной асимптотикой.

[1]Для случайной величины X, имеющей биномиальное распределение с параметрами $(p, n) : p \leqslant 0.5$ вероятность $p_n = Pr\left(X - np \geqslant (np(1-p))^{\frac{1}{2}}\right)$ положительна при любых значениях n. p_n сходится к положительному пределу в силу центральной предельной теоремы и поэтому при $n = 1, 2, \ldots$ имеем $\inf\limits_n p_n > 0$.

Вместо случайной генерации V_1, V_2 в алгоритме 2, метод последовательно присваивает по одной вершине за раз одному из подмножеств, так что матожидание размера решения, полученного при продолжении случайного вычисления с этого момента, максимизируется. Причина в том, что матожидание размера случайного решениия — это среднее двух полученных матожиданий, обусловленных присваиванием текущей вершины. Присваивание её множеству, дающему большее значение, гарантирует, по крайней мере, безусловное матожидание.

Заметим, что, исходя из доказательства теоремы 3, член $\Omega\left(\frac{1}{\sqrt{d_{max}}}\right)|A|$ получается из анализа матожидания количества рёбер, полученных в множествах V_1 и V_2. Шаг 3 алгоритма 2 гарантирует половину рёбер между множествами (и если мы просто рассмотрим матожидания, то оба (π_1, π_2) и (π_2, π_1) подходят под условия теоремы). Теперь покажем, как приведенное выше условие может быть гарантировано детерминистически. На шаге 3 полное решение будет иметь $\left(0.5 + \Omega\left(\frac{1}{\sqrt{d_{max}}}\right)\right)|A|$ рёбер.

Предположим, что каждая из вершин $1, \ldots, k-1$ уже были присвоены к V_1 или V_2. Теперь покажем, как эффективно вычислить матожидание количества рёбер в этих наборах, полученных на шаге 2 алгоритма 2, когда V_1, V_2 заполняются случайным присваиванием вершин $k, \ldots, n$. Как и в доказательстве теоремы 3, мы связываем каждое ребро с той его вершиной, у которой индекс меньше.

Рассмотрим данную вершину $i \in \{1, \ldots, k-1\}$. Предположим, что она была присвоена к V_r. Очевидным образом алгоритм 2 гарантирует, что приблизительное решение будет содержать по крайней мере половину рёбер внутри V_r из тех, которые связаны с i. Сейчас нас интеремует вычисление матожидания количества дополнительных таких рёбер, которые будут содержаться в нашем решении, учитывая первоначальное присваивание первых $k-1$ вершин. Пусть $E(x_i, y_i, z_i)$ обозначает матожидание количества таких дополнительных рёбер, где

$$\begin{aligned} x_i &= |(i,j) : j \in V_r, i < j < k| - |(i,j) : j \in V_r, i < j < k|, \\ y_i &= |(i,j) : j \geqslant k|, \\ z_i &= |(i,j) : j \geqslant k|. \end{aligned}$$

Тогда

$$\begin{aligned} &E(x,0,0) = 0.5|x|, \\ &x = \ldots, -2, -1, 0, 1, 2, \ldots \end{aligned}$$

и остальные значения можно вычислить рекурсивно следующим образом:

$$\begin{aligned} E(x,y,0) &= 0.5E(x+1, y-1, 0) + 0.5E(x, y-1, 0), \\ E(x,y,z) &= 0.5E(x,y,z-1) + 0.5E(x-1,y,z-1), z \geqslant 1. \end{aligned}$$

Рекурсия может быть применена путем вычисления $E(x,y,0)$ для фиксированного значения y и всех значений x, начиная с $y=0$, затем $y=1$ и так далее. После этого $E(x,y,z)$ вычисляется для фиксированного z и всех значений x, y, начиная с $z=1$, затем $z=2$ и так далее. Общие сложность вычисления этих значений для $x = -d_{max}, \ldots, d_{max}, \quad y = 0, \ldots, d_{max}, \quad z = 0, \ldots, d_{max}$ составляет $O\left(d_{max}^3\right)$, где d_{max} — максимальная степень вершины в G.

Ожидаемый выигрыш в рёбрах, связанных с вершинами $i \geqslant k$, равен просто $E(0, y_i, z_i)$.

Заметим, что приведённые выше матожидания справедливы в независимости от порядка, в котором вершины рассматриваются для объединения двух множеств.

Ожидаемый размер решения, обусловленный заданным частичным распределением вершин по двум подмножествам, равен $0.5|A| + \sum\limits_{i=1}^{k-1} E(x_i, y_i, z_i) + \sum\limits_{i=k}^{n} E(0, y_i, z_i)$.

Всякий раз, когда рассматривается неприсвоенная вершина, сравниваются две альтернативы для ее присвоения. Каждая альтернатива влияет на значение (x,y,z) своих соседей, а затем выполняется лучшее присваивание и вычисляются пересмотренные значения соседей. Общее время, затраченное на внесение этих изменений и сравнение матожиданий решения, связанных с двумя альтернативами каждой вершины, равно $O(|A|)$.

Таким образом, вычисление значений E, определяющих отнесение к одному из двух множетсв, а также выполнение шагов 2 и 3 алгоритма 2 занимают $O\left(d_{max}^3 + |A|\right)$.

Конструктивные алгоритмы

Мы упоминали, что 0.5-приближение может быть получено путём сравнения графов, определенных любой перестановкой и обратной к ней. Далее можно показать, что лучшая оценка не может быть получена путем выбора любого полиномиального множетсва перестановок (независимо от конкретного вида задачи) с последующим сравниванием решений, которые они дают. Поэтому теперь перейдем к исследованию методов, которые генерируют перестановку в зависимости от конкретной постановки задачи.

Нашим основным инструментом будет алгоритм 1. Заметим, что он не определяет порядок, в котором рассматриваются вершины на шаге 2. Рассмотрим некоторые "примечательные"правила для исследования вершин, а также приведём "плохой пример"для каждого из них. Все плохие примеры описывают невзвешенные постановки задачи.

Напомним, что $w_i^{in}(S) = \sum\limits_{j \in S} w_{ji}, \quad w_i^{out}(S) = \sum\limits_{j \in S} w_{ij}$. Пусть $W_i(S) = \max\left\{w_i^{in}(S), w_i^{out}(S)\right\}$. Наше первое правило получает на каждой итерации максимально возможный вес:

Алгоритм 5.
На шаге 2 алгоритма 1 выбирать вершину $i \in S$ с максимальным значением $W_i(S)$.

Пример 6. Пусть $n = 2k + 1$ и $A = \{(j, k + 1) \mid j = 1, \ldots, k\} \cup \{(k + 1, j) \mid j = k + 2, \ldots, 2k + 1\}$. Граф ациклический и оптимальное решение содержит A. Однако алгоритм 5 начнет с выбора $k+1$ и присвоит $\pi(k+1)$ либо 1, либо n, таким образом потеряв половину рёбер.

Пусть $w_i(S) = min\left\{w_i^{in}(S), w_i^{out}(S)\right\}$. Наше второе правило аналогично тому, которое использовали Лин и Сахни [16], чтобы решить их проблему с узким местом. Это минимизирует на каждой итерации вес потерянных рёбер:

Алгоритм 7. На шаге 2 алгоритма 1 выбирать вершину $i \in S$ с наименьшим значением $w_i(S)$.

Пример 8. Пусть $n = k^2, V = V_1 \cup V_2 \cup \ldots \cup V_k$, где $V_i = \{(i-1)k + 1, \ldots, ik\}, i = 1, \ldots, k$. Пусть $A = \{(p, q) \mid p \in V_i, q \in V_{i+1}, i = 1, \ldots, k - 1\} \cup \{(p, q) \mid p \in V_k, q \in V_1\}$. Заметим, что $A \setminus \{(p, q) \mid p \in V_k, q \in V_1\}$ является ациклическим, и оптимальное решение содержит $k^2(k - 1)$ рёбер. Однако возможна следующая последовательность выборок при использовании алгоритма 7. Первоначально $S = V$ и $w_i(S) = k$ для всех $i \in V$. Положим $\pi(1) = 1$. Теперь, $S = V \setminus \{1\}, w_i(S) = k - 1$ для вершин i в V_2 и V_k, в то время как $w_i(S) = k$ для всех остальных вершин S. Вершина $k + 1 \in V_2$ может быть выбрана следующей, а $\pi(k + 1)$ будет присвоено значение 2. Кандидатами для отбора теперь являются вершины в V_1, V_2, V_3 и V_k. Алгоритм сейчас может присвоить $\pi(2k + 1) = 3$, затем $\pi(3k + 1) = 4$ и так далее, пока $\pi((k - 1)k + 1)$ не будет присвоено значение k. На текущий момент в S осталось $k - 1$ вершин из каждого множества. Алгоритм может продолжить выбор вершин в следующем порядке: $(2, k+2, 2k+2, \ldots, (k-1)k+2, 3, k+3, \ldots)$. Таким образом, полученное решение содержит $\frac{k^2(k+1)}{2}$ ребра вида $(ik + j, (i + 1)k + l)$ для $l \geqslant j$. Асимптотически это всего лишь половина оптимального значения.

Пусть $\widehat{w}_i(S) = \left|w_i^{in}(S) - w_i^{out}(S)\right|$. Наше третье правило максимизирует на каждой итерации превышение полученного веса над потерянным.

Алгоритм 9. На шаге 2 алгоритма 1 выбирать вершину $i \in S$ с максимальным значением $\widehat{w}_i(S)$.

Пример 10. Пусть $n = 2k + 3, A = \{(j, k + 1) \mid j = 1, \ldots, k\} \cup \{(k + 1, j) \mid j = k + 2, \ldots, 2k + 3\}$. Тогда вершина $k + 1$ изначально имеет $\widehat{w}_{k+1}(V) = 2$, в то время как все остальные вершины имеют $\widehat{w}_i(S) = 1$. Следовательно, $k + 1$ будет выбрана первой в алгоритме 9 и количество потерянных рёбер в решении асимптотически равно $\frac{|A|}{2}$.

Пусть $r_i(S) = \max\left\{\frac{w_i^{out}(S)}{w_i^{in}(S)}, \frac{w_i^{in}(S)}{w_i^{out}(S)}\right\}$. Наше четвертое правило максимизирует на каждой итерации отношение полученного веса к потерянному:

Алгоритм 11. На шаге 2 алгоритма 1 выбирать вершину $i \in S$ с наибольшим значением $r_i(S)$.

Пример 12. Ещё раз рассмотрим граф из примера 8. Оптимальное решение состоит из $k^2(k-1)$ рёбер, например $\{(i,j) \mid i \in V_l, j \in V_{l+1}, l = 1, \ldots, k-1\}$. Алгоритм 11 может выбрать вершину из V_1, затем из $V_2, V_3, \ldots, V_{k-2}$, а затем в циклическом порядке множеств, начиная с V_{k-1} по две вершины из каждого подмножества. Выбранная перестановка описана для $k = 6$ в следующей таблице:

V_1	V_2	V_3	V_4	V_5	V_6
1	2	3	4	5	7
9	11	13	15	6	8
10	12	14	16	17	19
21	23	25	27	18	20
22	24	26	28	29	31
33	34	35	36	30	32

Эта перестановка приводит к решению с $\frac{k(k+1)}{2}$ рёбрами, идущими от вершины в V_l к вершине в V_{l+1} для $l = 1, \ldots, k$. Общее число рёбер, выбранных алгоритмом 11, равно $\frac{k^2(k+1)}{2}$, а отношение оптимальных решений к приближенным асимптотически равно 0.5.

Теперь опишем другой подход к генерации перестановки. Вместо определения точного значения $\pi(i)$ на шаге 2 определим только относительное расположение i относительно вершин, которые уже были исследованы. Таким образом, на каждом этапе нам задается (общий) порядок на множестве $Q = V \setminus S$. Выбираем $i \in S$ и выбираем для него наилучшее местоположение относительно этого порядка. В любой заданной точке выполнения алгоритма обозначим символом $\prec$ порядок, определённый на подмножестве V, просмотренном на данный момент. В частности, $k \preceq j$ означает, что k — это либо j, либо вершина, присвоенная предшествующей j.

Алгоритм 13.

1) Выберем $(i,j) \in A$. Пусть $Q = i, j, i \prec j, S = V \setminus Q$.
2) Полагаем, что данный шаг достигнут с порядком $\prec$ на Q. Выберем $i \in S$ и множество $S \leftarrow S \setminus i$. Вычислим для каждого $j \in Q$
$$D_j = \sum_{k \mid k \preceq j} w_{ki} + \sum_{K \mid j \preceq k} w_{ik}.$$
Положим $D_0 = \sum\limits_{j \in Q} w_{ij}$, $D_l = \max\{D_J \mid j \in Q\}$. Если $D_l \geqslant D_0$, расширим $\prec$ путём добавления таких i, которые являются непосредственным преемником l. Если $D_l < D_0$, расширим $\prec$ путём добавления i в качестве первого элемента.
3) Если $S = \varnothing$, остановиться. Иначе присвоить $Q \leftarrow Q \cup \{i\}$ и перейти на шаг 2.

Пример 14. Пусть $A = \{(1,n)\} \cup \{(n,j) \mid j = 2, \ldots, n-1\} \cup \{(j,1) \mid j = 1, \ldots, n-1\}$. Если алгоритм 13 начинается с $(1,n)$, то как минимум половина остальных рёбер должны быть потеряны. Таким образом, результатом будет $n-1$ ребро, в то время как оптимальное решение содержит $2(n-2)$ рёбер в $A \setminus (1,n)$.

Локальный поиск

Общая идея локального поиска включает определение окрестности для каждого возможного решения задачи. Затем текущее решение заменяется лучшим решением из его окрестности, если такое решение существует. В противном случае алгоритм останавливается с текущим ("локально оптимальным") решением.

Мы считаем довольно естественным определять окрестности в отношении перестановок. Пусть V_π — окрестность π.

Алгоритм 15. Применим локальный поиск с помощью V_k, определённого следующим образом: $\pi' \in V_\pi$,

если для некоторого $j \neq k$,

$$\pi' = (\pi_1, \ldots, \pi_j, \pi_k, \pi_{j+1}, \ldots, \pi_{k-1}, \pi_{k+1}, \ldots)$$

или

$$\pi' = (\pi_1, \ldots, \pi_{k-1}, \pi_{k+1}, \ldots, \pi_j, \pi_k, \pi_{j+1}, \ldots).$$

Другими словами, π'_i получается из π с помощью изменения позиции одной вершины. Например, $(1, 2, 3, 6, 4, 5, 7)$ и $(1, 3, 4, 5, 2, 6, 7)$ являются соседями $(1, 2, 3, 4, 5, 6, 7)$.
Алгоритм 15 является 0.5-приближением. Докажем это утверждение, показав, что любое решение, содержащее менее половины общего веса рёбер, не может быть получено с помощью локально оптимальной перестановки. Такое решение должно иметь по крайней мере одну вершину таким образом, чтобы общий вес рёбер, инцидентных с ней в индуцированном подграфе, был строго меньше половины общего веса рёбер, инцидентных с ней в G. Но в этом случае лучшая перестановка может быть получена путем перемещения этой вершины либо в первую, либо в последнюю позицию (один из этих вариантов добавит больше веса, чем то, что теряется при изменении). Таким образом, предлагаемое решение не может быть локально оптимальным.

Пример 16. Пусть $A = \{(i, i+1) \mid i = 1, \ldots, n-1\}$. Граф ациклический, и оптимальное решение содержит все $n-1$ рёбер. Однако перестановка $(n-1, n, n-3, n-2, \ldots, 5, 6, 3, 4, 1, 2)$ является локально оптимальной и индуцирует подграф с $\frac{n}{2}$ рёбрами $(n-1, n), (n-3, n-2), \ldots, (5, 6), (3, 4), (1, 2)$. Любое локальное изменение в перестановке может добавить не более одного ребра, но, несомненно, также потеряет одно ребро.

Алгоритм 17. Сделаем локальный поиск с V_π, определенным следующим образом: $\pi' \in V_\pi$, если для некоторого $j < k$,

$$\pi' = (\pi_1, \ldots, \pi_{j-1}, \pi_k, \pi_{j+1}, \ldots, \pi_{k-1}, \pi_j, \pi_{k+1}, \ldots).$$

Другими словами, π' получается из π путем переставления двух вершин друг с другом. Например, перестановка $(1, 6, 3, 4, 5, 2, 7)$ является соседом $(1, 2, 3, 4, 5, 6, 7)$.

Алгоритм 17 является 0.5-приближением. Докажем это утверждение, показав, что любое решение, содержащее менее половины общего веса рёбер, не может быть получено локально оптимальной перестановкой. Достаточно показать, что вес подграфа, индуцированного локально оптимальной перестановкой π, по меньшей мере так же хорош, как и вес подграфа, индуцированного обратной перестановкой $\tilde{\pi}$(поскольку сумма двух равна общему весу рёбер в G). Обратную перестановку $\tilde{\pi}$ можно получить из π, меняя местами два элемента между собой $\frac{n}{2}$ подряд: сначала поменяем местами $\pi(1)$ и $\pi(n)$, затем $p(2)$ и $\pi(n-1)$ и так далее. Влияние второй замены на вес индуцированного подграфа не зависит от первой замены, поскольку оно влияет только на рёбра, два конца которых отличаются как от $\pi(1)$, так и от $\pi(n)$. Аналогично, каждая последующая перестановка влияет на решение таким же образом, как она повлияет на него, если мы сделаем это непосредственно с исходной перестановкой π. Поскольку π является локальным максимумом, ни одна из этих перестановок не увеличивает вес индуцированного подграфа. Это доказывает наше утверждение.

Пример 18. Рассмотрим ещё раз пример 16. Перестановки, описанная там, также локально оптимальна по отношению к алгоритму 17.

Даже если мы допускаем бóльшую окрестность, в которой допускается до k перестановок для некоторого фиксированного k, расширение примера 16 все равно применимо. Пусть $V = V_1 \cup V_2 \cup \ldots \cup V_{2l}$, где $|V_i| = m \gg k$ для $i = 1, \ldots, 2l$. Из каждой вершины в V_i для четных i есть рёбра, идущие ко всем m вершинам в V_{i+1}. Из каждой вершины в V_i для нечётных i есть $m-k$ рёбер, ведущих к вершинам в V_{i+1}. Эти рёбра выбраны так, чтобы для вершины в V_i при чётном i было ровно $m-k$ рёбер, входящих из V_{i-1}. График является ациклическим и имеет $lm^2 + lm(m-k)$ рёбер. Рассмотрим перестановку, в которой вершины из каждого V_i были последовательными, а порядок множеств был $V_{2l-1}, V_{2l}, V_{2l-3}, V_{2l-2}, \ldots, V_3, V_4, V_1, V_2$. Из каждой вершины в V_i при чётных i решение, полученное перестановкой, содержит все m рёбер, исходящих из неё. Решение не содержит ни одного из рёбер, идущих от V_i к V_{i+1} для нечетных i. Таким образом, оно содержит lm рёбер, что составляет примерно половину оптимального количества. Любое изменение местоположения любого отдельного индекса влечет за собой потерю по меньшей мере k рёбер. Следовательно, при перестановке не более k вершин выигрыш невозможен.

Благодарности

Доказательство того, что алгоритм 17 имеет 0.5-приближение, принадлежит Нили Гутман.

Литература

[1] N. Alon and J.H. Spencer, *The Probabilistic Method* (John Wiley and Sons, New York, 1992).

[2] B. Berger and P.W. Shor, Approximation algorithms for the maximum acyclic subgraph problem, in: *Proc. 1st Ann. ACM-SIAM Symp. on Discrete Algorithms* (1990) 236-243 .

[3] M.M. Flood, Exact and heuristic algorithms for the weighted feedback arc set problem: A special case of the skew-symmetric quadratic assignment problem, *Networks* 20 (1990) 1-23.

[4] A. Frank, How to make a digraph strongly connected, *Combinatorica* 1 (1981) 145-153.

[5] H.N. Gabow, A representation for crossing set families with applications to submodular flow problems, *Proc. 4th Ann. ACM-SIAM Symp. on Discrete Algorithms* (1993) 202-211.

[6] H.N. Gabow, A framework for cost-scaling algorithms for submodular flow problems, 1993.

[7] M. Grötschel, M. Jünger and G. Reinhelt, On the acyclic subgraph polytope, *Mathematical Programming* 33 (1985) 28-42.

[8] M. Grötchel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization* (Springer, Berlin, 1988).

[9] B. Korte, Approximation algorithms for discrete optimization problems, *Ann. Discrete Math.* 4 (1979) 85-120.

[10] R. Kaas, A branch and bound algorithm for the acyclic subgraph problem, *European J. Oper. Res.* 8 (1981) 335-362.

[11] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, eds., *Complexity of Computer Computations* (Plenum, New York, 1972) 85-103.

[12] A. Karazanov, On the minimal number of arcs of a digraph meeting all its directed cutsets, abstract, *Graph Theory Newsletters* 8 (1979).

[13] B. Korte and D. Hausmann, An analysis of the greedy heuristic for independence systems, *Ann. Discrete Math.* 2 (1978) 65-74.

[14] M. Jünger, *Polyhedral Combinatorics and the Acyclic Subgraph Problem* (Heldermann, 1985).

[15] C.L. Lucchesi, A minimax equality for directed graphs, Ph.D. Dissertation, University of Waterloo, Waterloo, Ontario, 1976.

[16] L. Lin and S. Sahni, Fair edge deletion problems, *IEEE Trans. Comput.* 38 7(1989) 56-761.

[17] S. Miyano, Systematized approaches to the complexity of subgraph problems, *J. Inform. Process.* 13 (1990) 442-447.

[18] M, Penn and Z. Nutov, Minimum feedback are set and maximum integral dicycle packing in $K_{3,3}$-free digraphs, 1993.

[19] C.H. Papadimitriou and M. Yannakakis, Optimization, approximation, and complexity classes, *J. Comput, System Sci.* 43 (1991) 425-440 .

[20] V. Ramachandran, Finding a minimum feedback arc set in reducible flow graphs, *J. Algorithms* 9 (1988) 299-313.

[21] M. Yannakakis, Node- and edge-deletion NP complete problems, in: *Proc. 10th ACM Symp. on Theory of Computing* (ACM, New York, 1978) 253-264.