Вычислительный подход к решению задачи о поиске максимальной клики*

М. А. Грибков, А. В. Алексеевский,
С. А. Спирин, М. А. Короткова

Аннотация

В статье предлагается подход к решению задачи нахождения максимального полного подграфа в графе. Сама задача, как известно, является NP-полной. Описываемый подход позволяет находить приближенное решение за полиномиальное время. Предлагаемый подход содержит три алгоритма: каждый следующий включает предыдущие и дает более точный результат.

Введение

В математике хорошо известна так называемая «проблема клики» или задача поиска максимального полного подграфа (клики) в графе. Известно, что задача нахождения максимальной клики является NP-полной [1], что делает достаточно неэффективным сведение к ней задач практических. С другой стороны, в ряде практических задач бывает вовсе не обязательно нахождение точного решения, а вполне приемлемо некоторое приближение. В таких случаях задачу можно переформулировать и создать достаточно эффективный алгоритм ее решения.

Один из часто встречающихся классов таких задач рассматривался нами в контексте обработки пространственных структур биологических макромолекул [2]. Специфика задачи такова, что от найденной клики требуется гарантированно являться максимальной лишь тогда, когда другие клики заметно меньше нее. Ориентация на такой класс задач может встречаться во множестве прикладных областей от астрономии до экономики. Ниже предлагается разработанный нами алгоритм «приближенного» решения задачи нахождения максимальной клики.

Методика

Как уже говорилось выше, мы будем рассматривать задачу нахождения максимальной клики для случая наличия в графе клики, заметно превышающей по размеру прочие (*массивной* клики [2]). Если существует несколько

* Работа частично поддержана грантом РФФИ 03–07–90157.

клик, сильно превышающих размеры большинства прочих, то, при незначительной разнице их размеров, все такие клики считаются массивными, и нас не будет интересовать, какую именно из них выберет алгоритм.

Разработанный нами подход включает три алгоритма: каждый следующий включает предыдущие и дает более точный результат. Рассмотрим их по порядку.

Алгоритм 1 (простая итерация)

Алгоритм

На каждом шаге ищется вершина i с наименьшим числом $v(i)$ смежных ребер (наименьшей валентностью), и, если $v(i) < |G| - 1$ (где $|G|$ — мощность графа на текущем шаге), она удаляется. Когда удалять будет нечего (все вершины имеют одинаковое количество смежных ребер: $\forall\, i v(i) = |G| - 1$), оставшиеся вершины образуют полный подграф исходного графа. На рис. 1 изображена блок-схема описанного алгоритма.

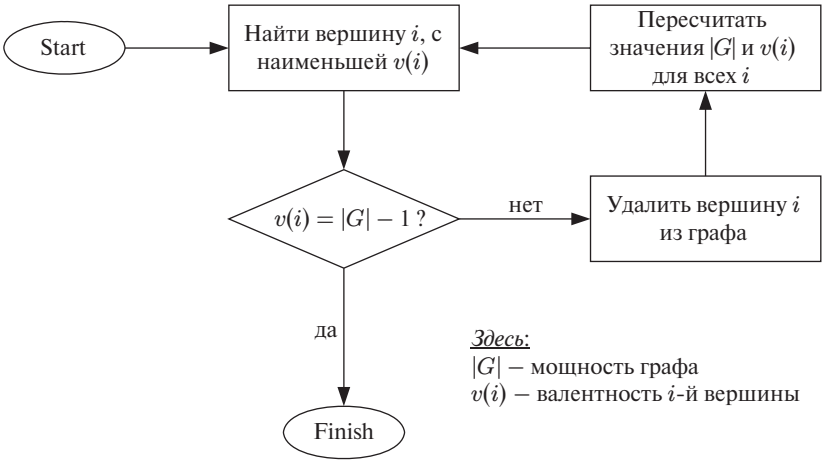

Рис. 1. Алгоритм простой итерации

Область адекватности

В достаточно большой доле реальных ситуаций данный алгоритм находит именно максимальный полный подграф, однако несложно представить ситуацию, в которой это не так. Впрочем, можно определить класс ситуаций, на которых решение находится гарантированно. Если существует одна массивная клика, то алгоритм обязательно ее найдет. Если существует несколько массивных клик, то алгоритм найдет одну из них.

Теорема 1. *Если некоторый полный подграф G_1 графа G содержит k вершин, и все вершины множества $G \setminus G_1$ имеют валентность меньшую, чем $k-1$, то подграф G_1 будет найден алгоритмом и выделен без потерь.*

Доказательство. Поскольку вначале все вершины из G_1 имеют валентность не меньше $k-1$, а все остальные вершины — меньшую, то процесс удаления вершин из графа начнется с вершин, не принадлежащих G_1. На каждом следующем шаге, поскольку удаляется вершина не из G_1, основное соотношение валентностей останется прежним: внутри G_1 валентности будут не меньше $k-1$, а вне него — строго меньше $k-1$. Таким образом, первые $N-k$ шагов алгоритма удалят из G все вершины, не принадлежащие G_1. На $N-k+1$-й итерации алгоритм остановится оставив от графа искомый полный подграф. □

Отметим, что существенную проблему для данного алгоритма составляют задачи, в которых вершины максимальной клики имеют валентность меньшую, чем у вершин, не входящих в эту клику. В решении таких задач могут помочь алгоритмы, описанные далее.

Сходимость

Легко видеть, что алгоритм сходится: поскольку на каждом шаге происходит либо удаление вершины, либо остановка, то количество шагов в любом случае меньше числа вершин исходного графа.

Временные характеристики

Рассмотрим зависимость времени работы данного алгоритма от мощности графа N. Пусть на проверку валентности каждой вершины тратится одно и то же постоянное время C_1, а на удаление вершины и пересчет валентностей — C_2N (линейная зависимость от N с некоторым постоянным коэффициентом; предположения вполне допустимые при рассмотрении зависимости от количества вершин). Тогда, обозначив за C размер искомого полного подграфа (считаем постоянным), получаем время работы алгоритма (Θ — оценка порядка роста):

$$\begin{aligned} T_1(N) &= C_1N + C_1(N-1) + \ldots + C_1C + C_2N(N-C) = \\ &= C_1\left(\sum_{i=C}^{N} i\right) + C_2N(N-C) = \\ &= N^2\left(\frac{C_1}{2} + C_2\right) + N\left(\frac{C_1}{2} - CC_2\right) - \frac{C_1}{2}\left(C^2 + C\right) = \\ &= \Theta(N^2). \end{aligned} \tag{1}$$

В худшем случае, когда $C = 1$, T_1 принимает значение:

$$T_{1\max}(N) = N^2\left(\frac{C_1}{2} + C_2\right) + N\left(\frac{C_1}{2} - C_2\right) - C_1. \tag{2}$$

Алгоритм 2 (расширенная итерация с удалением)

Алгоритм

На каждом шаге алгоритма производится поиск полного подграфа по алгоритму простой итерации. При переходе к следующему шагу из графа

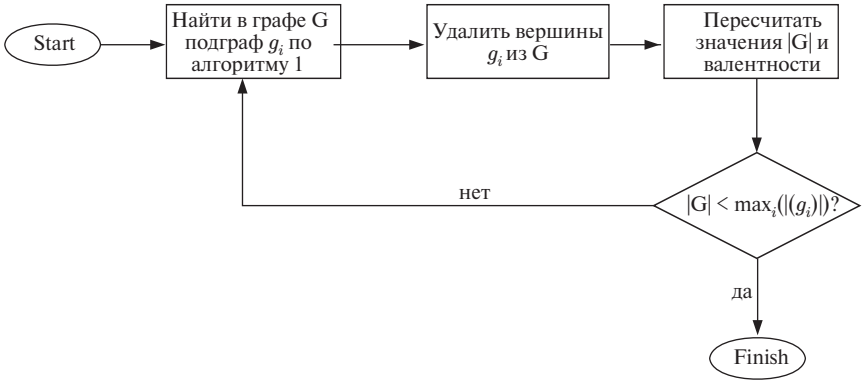

Рис. 2. Алгоритм расширенной итерации с удалением

удаляются все вершины, принадлежащие найденному полному подграфу. Таким образом, каждый раз поиск ведется в той части графа, которая не принадлежит ни одному из уже найденных полных подграфов. Алгоритм останавливается, когда мощность оставшегося графа оказывается меньше максимального из уже найденных полных подграфов. Этот подграф считается искомым. На рис. 2 представлена блок-схема расширенного алгоритма.

Область адекватности

Обладая с точки зрения адекватности всеми достоинствами первого алгоритма (поскольку включает его как один из шагов), второй позволяет решать большее число задач без явно выраженных массивных клик.

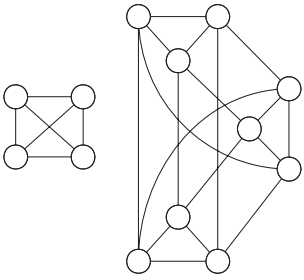

Рис. 3. Граф, неподвластный первому алгоритму, успешно обрабатывается вторым

Например, двухблочный граф на рис. 3, в случае работы алгоритма простой итерации, найдет клику из 3 вершин, в то время как алгоритм расширенной итерации успешно найдет левую клику из 4 вершин.

Трудность для алгоритма расширенной итерации представляют случаи существования вершин, принадлежащих нескольким кликам. Такие вершины будут приписаны строго одному подграфу, что исказит мощность другого (других).

Сходимость

Сходимость алгоритма опять же вытекает из ограниченности множества удаляемых вершин.

Временные характеристики

Рассмотрим зависимость максимального времени работы данного алгоритма от мощности графа N (при рассмотрении учтем формулы (1) и (2)). Пусть C_3N — время, необходимое для удаления подграфа из графа и пересчета валентностей оставшихся вершин (верхняя оценка — линейная зависимость от N). Введем еще величины A_j ($j = 1, 2, \ldots$), обозначающие мощность последовательно находимых алгоритмом полных подграфов. Заметим, что ни одна из этих величин не может быть больше N и их общее число (максимальный индекс) заведомо меньше N (число это будем считать постоянным в рамках рассмотрения зависимости только от N). Тогда время работы алгоритма:

$$T_2(N) = T_1(N) + C_3N + T_1(N - A_1) + C_3N + \\ + T_1(N - A_1 - A_2) + C_3N + \ldots = \Theta(N^3).$$

Максимальное время, очевидно, достигается при $A_1 = A_2 = \ldots = 1$. В этом случае, с учетом (2):

$$T_{2\,\max}(N) = C_3N(N-1) + \sum_{i=1}^{N} T_{1\,\max}(i) = \left(\frac{C_1}{4} - \frac{C_2}{2}\right)(N^2 + N) - \\ - C_3(N^2 - N) - C_1N + \left(\frac{C_1}{2} + C_2\right)\sum_{i=1}^{N} i^2. \qquad (3)$$

Алгоритм 3 (учет перекрытий)

Алгоритм

Сначала производится поиск подграфов по алгоритму расширенной итерации с удалением. Для каждого найденного подграфа g_i на исходном графе производится перебор вершин, которые алгоритм расширенной итерации в g_i не включил. Для каждой такой вершины производится проверка, нельзя ли включить ее в g_i в условиях исходного графа (т.е. существуют ли ребра, соединяющие эту вершину со всеми вершинами, входящими в g_i при условии рассмотрения графа, из которого ничего не удалялось). Если вершину включить можно — она включается в g_i.

Наибольший из g_i признается искомым максимальным полным подграфом.

Область адекватности

Алгоритм с учетом перекрытий еще расширяет возможности алгоритмов простой итерации, позволяя решать задачу нахождения максимальной клики для графов, в которых она может быть отнесена к нескольким кликам-«зародышам». Пример такого графа, нахождение максимальной клики на котором возможно при помощи алгоритма с учетом перекрытий, но невозможно при помощи расширенной итерации, изображен на рис. 4.

Неразрешимыми для алгоритма остаются задачи на графах, для которых ни одна из найденных алгоритмом 2 клик-зародышей не лежит полностью в настоящей максимальной клике, или же, если оставшаяся после удаления перекрытий часть клики слишком мала по сравнению с самими перекрытиями.

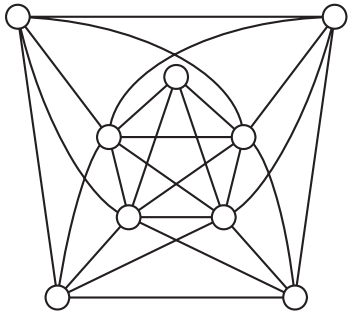

Рис. 4. Граф, не поддающийся успешному анализу алгоритмами 1 и 2, не вызывает затруднений у алгоритма 3

Сходимость

Сходимость алгоритма следует из соображений, аналогичных использованным ранее: конечность числа вершин и найденных предыдущими алгоритмами подграфов в любом случае делает конечным число удалений и проверок на включение.

Временные характеристики

Обозначим $C_4 A_i$ — время, необходимое на проверку одной вершины, нельзя ли включить ее в i-ю клику (пропорционально размерности этой клики A_i). Получаем время работы алгоритма:

$$T_3(N) = T_2(N) + C_4 A_1(N - A_1) + C_4 A_2(N - A_2) + \\ + C_4 A_3(N - A_3) + \ldots = \Theta(N^3).$$

Тогда, с учетом формулы (3), максимальное время, достигаемое при $A_1 = A_2 = A_3 = 1$, составляет:

$$T_{3\max}(N) = T_{2\max}(N) + NC_4(N-1) = \left(\frac{C_1}{4} - \frac{C_2}{2}\right)(N^2 + N) - \\ - C_3(N^2 - N) - C_1 N + \left(\frac{C_1}{2} + C_2\right)\sum_{i=1}^{N} i^2 + C_4(N^2 - N)$$

Стоит отдельно отметить, что соответствующие таким максимальным оценкам графы в реальных задачах практически не встречаются.

Апробация

Вышеописанные алгоритмы с успехом применены в системе Life Core [2, 3]. В рамках системы алгоритм применялся для выделения клик, сопоставляемых *геометрическим ядрам* семейств белков — характеризующим семейства подструктурам молекул. Определенные таким образом семейства полностью соответствуют ожидаемым (тестовая выборка на известных семействах). Работа с системой показала, что на графах, встречающихся в такого рода задачах, из трех представленных алгоритмов

Таблица 1

Сравнение скоростных характеристик алгоритмов

Размер графа	Время работы по алгоритмам, усл. мс			
	Алг. № 1	Алг. № 2	Алг. № 3	Перебор
50	5	5	15	16
100	15	15	16	391
200	15	156	157	4344
300	47	391	453	21515
400	140	1094	1125	66404
500	172	2000	2515	162844
600	406	3496	3891	349687

достаточно точный результат дает уже второй. Алгоритм № 3 улучшает результат лишь в некоторых достаточно редко встречающихся случаях.

В табл. 1 приведены сравнительные результаты испытаний разработанных алгоритмов на случайных графах различного размера. В последней колонке для сравнения приведен случай решения задачи нахождения максимального полного подграфа путем прямого полного перебора. (Размерность времени «условные миллисекунды» подчеркивает сравнительный характер данных, конкретные значения которых могут быть иными при тестировании на ином оборудовании.)

Испытания быстродействия алгоритмов производились для случайно сгенерированных графов, включающих клику размером $N/4$ и содержащих

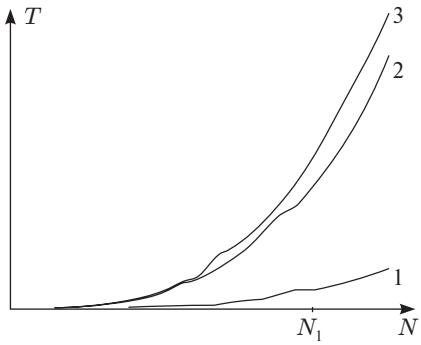

Рис. 5. Зависимость скорости роста времени работы алгоритмов от размера исходного графа

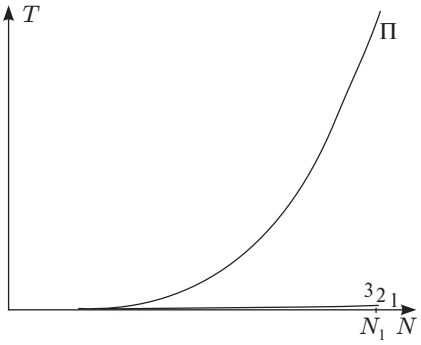

Рис. 6. Зависимость скорости роста времени работы алгоритмов от размера исходного графа (сравнение с перебором)

$3N/2$ ребер вне этой клики. Такие графы достаточно хорошо приближают встречавшиеся в реальных задачах при использовании алгоритмов в системе LifeCore.

Рис. 5 и 6 графически иллюстрируют сравнительные данные, аналогичные содержанию табл. 1 (зависимость времени работы алгоритма T от мощности исходного графа N). Кривые, помеченные цифрами «1», «2» и «3», соответствуют трем вышеописанным алгоритмам, а кривая, обозначенная «П» — случаю полного перебора (последняя колонка табл. 1). График на рис. 5 демонстрирует сравнение лишь рассматриваемых трех алгоритмов, в то время, как график на рис. 6 включает и данные для случая перебора. Конкретные величины на графиках не имеют ценности в отрыве от аппаратной платформы, но для удобства сравнения масштабов приведена некоторая общая для них точка N_1 на оси абсцисс.

Заключение

Представлена трехуровневая система алгоритмов нахождения приближенного решения «проблемы клики». Решения, предоставляемые алгоритмами, вполне соответствуют ожидаемым и выполняются за полиномиальное время.

Литература

1. *Кормен Т.*, *Лейзерсон Ч.*, *Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 1999.
2. *Gribkov M.*, *Alexeevski A.*, *Ivanova D.*, *Karyagina A.*, *Spirin S.* Life Core, program for classification of 3D structures of macromolecules // Biophysics (Moscow). 2004. Vol. 48. Suppl. 1. P. 157–166.
3. *Грибков М. А.* Программа для выявления геометрического ядра и классификации пространственных структур эволюционно родственных белков // Сборник тезисов Международной научной конференции студентов, аспирантов и молодых ученых «ЛОМОНОСОВ — 2004». М.: Изд-во МГУ, 2004. Т. 1. С. 12.