Покрытие ребер графа наименьшим числом клик: алгоритм Келлермана

А. О. Махорин*

Декабрь 2009 г.

1 Введение

Пусть задан (неориентированный) граф $G = (V, E)$, где V — множество вершин, E — множество ребер. *Кликой* графа называется порожденный подграф, в котором любые две вершины смежны. Будем говорить, что семейство клик C_1, C_2, ..., C_k графа G образует *реберное покрытие* этого графа, если каждое ребро $e \in E$ принадлежит хотя бы одной клике из указанного семейства. Задача состоит в минимизации числа клик, покрывающих все ребра заданного графа.

Пример покрытия ребер графа кликами показан на рис. 1.

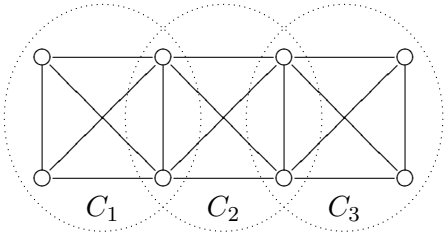

Рис. 1. Пример покрытия ребер графа кликами.

Известно, что в случае произвольных графов данная задача является NP-полной [2]. Поэтому практический интерес представляют в основном эвристические алгоритмы решения этой задачи, к числу которых относится алгоритм Келлермана.

В своем первоначальном виде алгоритм, предложенный Келлерманом, был предназначен для решения одной комбинаторной задачи, связанной с определением конфликтов в ключевых словах [1]. Однако в

*Кафедра прикладной информатики, Московский авиационный институт, Москва, Россия. E-mail: <mao@gnu.org>.

работе [3] было показано, что указанная комбинаторная задача эквивалентна рассматриваемой задаче о покрытии ребер графа наименьшим числом клик. Поэтому далее алгоритм Келлермана, который относится к числу «жадных» эвристик, излагается применительно к рассматриваемой задаче.

2 Описание алгоритма

Пусть $V = \{1, 2, \ldots, n\}$, т. е. вершины заданного графа $G = (V, E)$ занумерованы целыми числами от 1 до n. В алгоритме Келлермана вершины обрабатываются последовательно. После выполнения $(i-1)$-й итерации мы имеем порожденный подграф $G_{i-1} \subseteq G$ со множеством вершин $\{1, 2, \ldots, i-1\}$, для которого уже построено семейство клик C_1, C_2, $\ldots$, C_k, образующих реберное покрытие этого подграфа. На i-й итерации мы добавляем вершину i вместе с инцидентными ребрами из E к подграфу G_{i-1}, в результате чего получается подграф $G_i \subseteq G$ со множеством вершин $\{1, 2, \ldots, i\}$. Цель i-й итерации состоит, таким образом, в покрытии новых ребер, которые появляются в подграфе G_i, как это изображено на рис. 2.

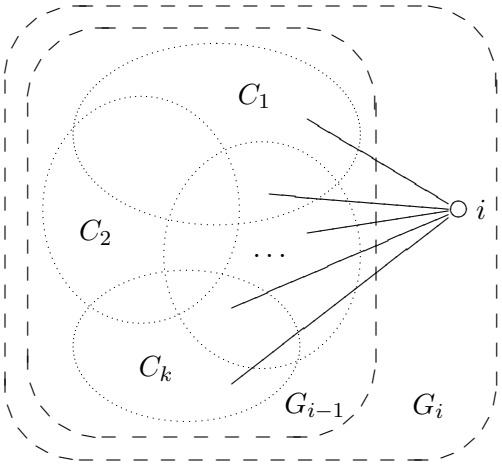

Рис. 2. Появление непокрытых ребер на i-й итерации.

Понятно, что все новые ребра, которые появляются на i-й итерации, имеют вид (i, j), где $i > j$. Пусть $W = \{j : (i, j) \in E,\ i > j\}$ — подмножество вершин подграфа G_{i-1}, смежных с вершиной i. Если $C_m \subseteq W$, где C_m — некоторая клика, $1 \leq m \leq k$, то вершина i смежна со всеми вершинами $j \in C_m$. В этом случае мы *расширяем* каждую такую клику, включая в нее вершину i, что позволяет покрыть соответствующие ребра.

Пусть теперь C_1, C_2, ..., C_k — семейство клик с учетом их расширения за счет вершины i. Положим $W := W \backslash (C_1 \cup C_2 \cup \ldots \cup C_k)$. Тогда каждая вершина $j \in W$ будет соответствовать ребру (i, j), которое осталось непокрытым. Эти непокрытые ребра уже нельзя включить в существующие клики, поэтому необходимо создать новые клики. Заметим, что для каждого непокрытого ребра (i, j) мы могли бы создать новую клику, содержащую только вершины i и j, которая покрывает это ребро. Однако, чтобы покрыть новой кликой возможно большее число оставшихся непокрытых ребер и тем самым уменьшить общее число клик, мы пытаемся воспользоваться частями уже существующих клик. Для этого мы находим существующую клику C_m, $1 \leq m \leq k$, у которой мощность пересечения $|W \cap C_m|$ является максимальной (если таких клик несколько, то мы берем клику с наименьшим значением m). Так как C_m — клика, то $W \cap C_m$ тоже будет кликой, а поскольку вершина i смежна со всеми вершинами из W, то добавляя ее в $W \cap C_m$ мы сформируем новую клику, которая позволяет покрыть $|W \cap C_m|$ непокрытых ребер (рис. 3). Далее мы полагаем $W := W \backslash C$, чтобы вычеркнуть из W вершины, для которых соответствующие ребра покрыты кликой C, увеличиваем k на единицу и включаем новую клику $C_k = C$ в семейство клик. Если в множестве W остались вершины, мы формируем еще одну новую клику и т. д. до тех пор, пока W не станет пустым, т. е. пока не будут покрыты все ребра подграфа G_i.

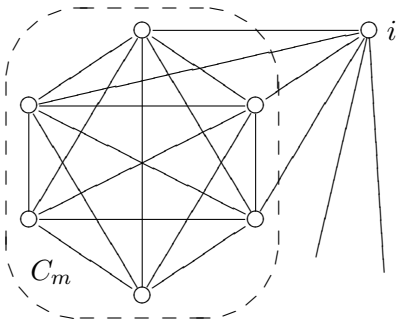

Рис. 3. Формирование новой клики.

Алгоритм заканчивает работу после выполнения n-й итерации, на которой будут покрыты все ребра графа $G_n = G$.

Формальное описание алгоритма Келлермана приведено рис. 4.

АЛГОРИТМ КЕЛЛЕРМАНА

Вход. Граф $G = (V, E)$, где $V = \{1, 2, \ldots, n\}$.

Выход. Семейство клик C_1, C_2, $\ldots$, C_k, покрывающих все ребра заданного графа.

$k := 0$;
/* основной цикл */
for $i := 1, 2, \ldots, n$ **do**
  $W := \{j : i > j,\ (i, j) \in E\}$;
  /* цель i-й итерации — покрыть ребра (i, j) для всех $j \in W$ */
  if $W = \varnothing$ **then**
    /* специальный случай */
    $k := k + 1$;
    $C[k] := \{i\}$;
    continue for i;
  end if
  /* попытаться включить вершину i в существующие клики */
  $V := \varnothing$;
  for $m = 1, 2, \ldots, k$ **while** $V \neq W$ **do**
    if $C[m] \subseteq W$ **then**
      $C[m] := C[m] \cup \{i\}$;
      $V := V \cup C[m]$;
    end if
  end for
  /* исключить из W вершины, инцидентные покрытым ребрам */
  $W := W \backslash V$;
  /* покрыть оставшиеся ребра, формируя новые клики */
  while $W \neq \varnothing$ **do**
    Найти клику $C[m]$, $1 \leq m \leq k$, для которой мощность пересечения $|W \cap C[m]|$ является максимальной (если имеется несколько таких клик, взять клику с наименьшим номером m);
    /* сформировать новую клику, используя часть $C[m]$ */
    $k := k + 1$;
    $C[k] := (W \cap C[m]) \cup \{i\}$;
    /* исключить из W вершины, инцидентные покрытым ребрам */
    $W := W \backslash C[k]$;
  end while
end for

Рис. 4. Алгоритм Келлермана.

3 Реализация алгоритма

3.1 Алгоритм Келлермана (базовый вариант)

Спецификация

```
#include "glpnet.h"
int kellerman(int n, int (*func)(void *info, int i, int ind[]),
      void *info, glp_graph *H);
```

Назначение

Подпрограмма `kellerman` реализует эвристический алгоритм Келлермана для нахождения реберного покрытия заданного графа $G = (V, E)$ минимальным числом клик.

Параметр $n \geq 0$ задает $|V|$, число вершин графа G.

Формальная подпрограмма `func` задает множество ребер E графа G следующим образом. В процессе работы подпрограмма `kellerman` вызывает формальную подпрограмму `func`, передавая ей номер некоторой вершины i, $1 \leq i \leq n$, графа G. В ответ подпрограмма `func` должна поместить номера вершин, смежных с вершиной i, в элементы массива $ind[1]$, $ind[2]$, ..., $ind[len]$ и вернуть значение len — общее число смежных вершин, $0 \leq len \leq n$. Петли допускаются, но игнорируются. Кратные ребра не допускаются.

Параметр *info* является транзитным указателем (magic cookie), который передается формальной подпрограмме `func` в качестве ее первого параметра.

Результатом выполнения подпрограммы `kellerman` является двудольный граф $H = (V \cup C, F)$, определяющий найденное реберное покрытие заданного графа G. (Программный объект типа `glp_graph`, заданный указателем H, должен быть предварительно создан с помощью подпрограммы `glp_create_graph`. В самом начале работы подпрограмма `kellerman` стирает текущее содержимое этого объекта, используя подпрограмму `glp_erase_graph`. Подробнее см. [4].) Вершины первой доли V указанного двудольного графа соответствуют вершинам исходного графа G и имеют те же порядковые номера 1, 2, ..., n. Вершины второй доли C соответствуют найденным кликам и имеют порядковые номера $n + 1$, $n + 2$, ..., $n + k$, где k — общее число клик в покрытии. Каждое ребро двудольного графа $f \in F$ в программном объекте H представлено дугой вида $f = (i \rightarrow j)$, где $i \in V$ и $j \in C$, которая означает, что вершина i исходного графа принадлежит клике C_j, $1 \leq j \leq k$. (Таким образом, если две вершины исходного графа принадлежат одной и той же клике, то эти две вершины смежны в G, а соответствующее ребро покрыто этой кликой.)

Возвращаемое значение

Подпрограмма `kellerman` возвращает значение k — общее число клик, образующих реберное покрытие заданного графа G.

3.2 Алгоритм Келлермана (стандартный вариант)

Спецификация

```
#include <glpk.h>
int glp_kellerman(glp_graph *G, glp_graph *H);
```

Назначение

Подпрограмма `glp_kellerman` эквивалентна подпрограмме `kellerman` (см. предыдущий подраздел) за исключением того, что в данном случае исходный граф $G = (V, E)$ представлен явно в виде программного объекта типа `glp_graph`, заданного указателем G. Каждая дуга $(i \rightarrow j)$ в этом программном объекте рассматривается как ребро (неупорядоченная пара вершин) $(i, j) \in E$ графа G. При этом допускаются как петли (которые игнорируются), так и кратные ребра (которые рассматриваются как простые ребра).

Литература

[1] E. Kellerman, "Determination of keyword conflict." IBM Tech. Disclosure Bull. 16, 2 (July 1973), pp. 544-46.

[2] J. Orlin, "Contentment in graph theory: Covering graphs with cliques." Indagationes Math. 39 (1977), pp. 406-24.

[3] L. T. Kou, L. J. Stockmeyer , C. K. Wong, "Covering edges by cliques with regard to keyword conflicts and intersection graphs." Comm. of the ACM, Vol. 21, No. 2 (1978), pp. 135-39.

[4] GNU Linear Programming Kit: Graph and Network Routines.