УДК 681.3-181.48 DOI 10.12737/4546

Сравнительный анализ алгоритмов раскраски обыкновенного взвешенного графа*

А. С. Мерзленко, В. Г. Кобак

Рассматриваются и сравниваются алгоритмы решения задачи поиска «минимаксной» раскраски взвешенного по вершинам графа. Приведена математическая постановка задачи, указаны критерии оптимальности решения. Описан точный алгоритм, всегда находящий минимальные по количеству цветов раскраски методом Магу и затем отыскивающий среди них «минимаксный» вариант с помощью 3 модификаций алгоритма «критического пути». Описаны три быстрых эвристических алгоритма: алгоритм, работающий с упорядоченным по локальным степеням списком вершин; алгоритм, основанный на удалении вершин и смежных рёбер; алгоритм, использующий степень насыщения вершин. Все алгоритмы рассмотрены с примерами. Для оценки эффективности алгоритмов поставлен вычислительный эксперимент на нескольких сотнях случайно сгенерированных графов. Алгоритмы сравнивались по скорости работы и близости результата к «минимаксному» варианту раскраски.

Ключевые слова: *взвешенные графы, раскраска взвешенного графа, теория расписаний.*

Введение. Задача раскраски взвешенного графа возникает в том случае, когда в вершинах графа необходимо выполнить некоторые работы различной целочисленной длительности, причём в смежных вершинах работы не могут выполняться одновременно. Хорошим примером этой задачи может служить проблема распределения команд в многопроцессорной системе, когда требуется минимизировать загрузку каждого процессора.

Математическая постановка задачи. Имеется неориентированный связный взвешенный граф $G = (V, E)$, где V — множество вершин, E — множество рёбер, $W(V_i)$ — вес i-й вершины. Необходимо раскрасить вершины графа таким образом, чтобы никакие две смежные вершины не были окрашены в один цвет, и при этом максимальная сумма весов вершин одного цвета стремилась к минимуму:

$$\max_{i\in(1..P)} \sum_{j\in B_i} W\left(v_j\right) \to \min, \qquad (1)$$

где B_i — подмножество несмежных вершин графа, окрашенных в i-й цвет.

Формально

$$\max T_j \to \min,$$
$$T_j = \sum_{i\in N_j} t_{i,j}$$

при условиях

$$t_{i,j} \geq 0;\ i = 1..m,\ j = 1..n,\ N_k \cap N_l = \varnothing;\ k, l \neq 1..n,\ k \neq l.\ \bigcup_{j=1}^{n} N_j = M,$$

где M — множество вершин, которые необходимо распределить по цветам; N — количество цветов; $t_{i,j}$ — вес вершины i при цвете j.

Данная задача принадлежит к классу *NP*-полных задач [1, 2].

«Точный» алгоритм. Точный (здесь и далее — без кавычек, тем не менее, этот алгоритм нельзя назвать полностью «точным», т. к. третий этап решения содержит эвристический метод) алгоритм [3] поиска раскраски минимального веса можно разделить на три этапа. На первом этапе решается задача поиска всех максимально внутренне устойчивых подмножеств графа C_l, $L = 1, K$, где K — число вершин в этом подмножестве. На втором этапе выбираются кортежи T_e, длина ко-

* Работа выполнена в рамках инициативной НИР.

торых равна хроматическому числу, а элементами являются максимально внутренне устойчивые подмножества, охватывающие все вершины (т. е. их объединение даёт множество V). На этих этапах используется метод Магу [4]. На третьем этапе l раз (l — число кортежей T_e) решается задача распределения программ по однородным вычислительным системам. Матрица загрузки приборов (процессоров) формируется следующим образом: строки соответствуют вершинам графа, столбцы — цветам раскраски. Если вершина $a_m \in C_l$, а C_l является элементом кортежа T_e, то на пересечении m-й строки и номера, под которым C_l стоит в кортеже T_e, ставится вес вершины $W(a_m)$. В противном случае ставится заведомо большее число («бесконечность»), чтобы вершина не могла быть назначена другому цвету.

На третьем этапе используется метод «критического пути». Однако, метод подразумевает упорядочивание строк матрицы, а почти в каждой строке находится «бесконечный» элемент. Приведём три модификации алгоритма «критического пути» для матриц с «бесконечными» элементами [5].

Первая модификация:

все операторы, имеющие время $a_i > 0$, но не равное бесконечности, записываются под тем же индексом в матрицу-строку $A1$ [1, m], где m — число строк в матрице A;

элементы матрицы $A1$ упорядочиваются в порядке убывания, формируя список приоритетов;

строки матрицы A упорядочиваются в соответствии со списком приоритетов;

решается задача распределения согласно критерию (1).

Вторая модификация:

в каждой строке матрицы A ищется количество бесконечностей и записывается в матрицу-строку $A1$;

элементы матрицы $A1$ упорядочиваются в порядке убывания количества бесконечностей, формируя список приоритетов;

строки матрицы A упорядочиваются в соответствии с получившимся списком приоритетов;

решается задача распределения согласно критерию (1).

Третья модификация:

в каждой строке матрицы A ищется количество бесконечностей и записывается в матрицу-строку $A1$;

элементы матрицы $A1$ упорядочиваются в порядке убывания количества бесконечностей;

строки матрицы A упорядочиваются в соответствии с матрицей $A1$;

если строки матрицы A имеют одинаковое количество бесконечностей, то происходит упорядочение в порядке убывания элементов, отличных от бесконечности;

решается задача распределения согласно критерию (1).

Рассмотрим пример, иллюстрирующий решение всей задачи точным алгоритмом.

Граф, для которого решается задача, изображён на рис. 1.

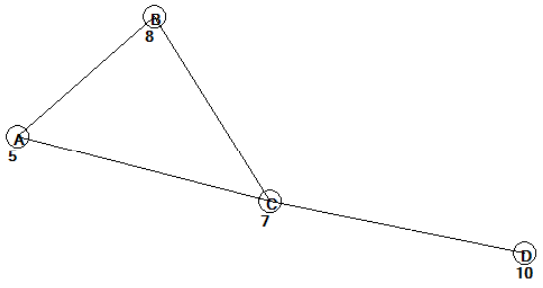

Рис. 1. Рассматриваемый граф

Найдём все максимально внутренне устойчивые подмножества графа методом Магу. Согласно этому методу составляется булево уравнение — сумма произведений отрицаний смежных вершин. Формула упрощается по законам $A\&A=A$ и $A+(A\&B)=A$. Вершины, отсутствующие в элементах получившегося многочлена, образуют максимально внутренне устойчивые подмножества. Для рассматриваемого графа булева формула будет иметь вид:

$$F_s=\left(\overline{a}+\overline{b}\right)\left(\overline{a}+\overline{c}\right)\left(\overline{b}+\overline{c}\right)\left(\overline{c}+\overline{d}\right)=\left(\overline{a}+\overline{bc}\right)\left(\overline{c}+\overline{bd}\right)=\overline{ac}+\overline{bc}+\overline{abd},$$

следовательно, максимально внутренне устойчивые подмножества: {*B*, *D*}, {*A*, *D*}, {*C*}.

Определим хроматическое число по методу Магу:

$$F_s=F_1+F_2+\ldots+F_n,\ F_1=\overline{ac},\ F_2=\overline{bc},\ F_3=\overline{abd}.$$

Определим, в каких элементах многочлена *F* отсутствуют те или иные вершины:

$$\overline{a}\notin F_2,\overline{b}\notin F_1,\overline{c}\notin F_3,\overline{d}\notin F_1,F_2,$$

$$y_a=y_2,y_b=y_1,y_c=y_3,y_d=y_1+y_2.$$

Составив и упростив формулу, получаем единственное покрытие графа максимально внутренне устойчивыми подмножествами:

$$\Psi=y_2y_1y_3\left(y_1+y_2\right)=y_1y_2y_3,$$

$$\Psi_1=y_1y_2y_3.$$

Хроматическое число равно количеству множителей в минимальном по длине элементе ψ_n. В данном случае элемент один и хроматическое число равно трём.

Составим матрицу загрузки:

$$A=\begin{vmatrix}\infty & 5 & \infty\\ 8 & \infty & \infty\\ \infty & \infty & 7\\ 10 & 10 & \infty\end{vmatrix}.$$

Определим расписание по первой модификации алгоритма «критического пути»:

$$A'=\begin{vmatrix}10 & 10 & \infty\\ 8 & \infty & \infty\\ \infty & \infty & 7\\ \infty & 5 & \infty\end{vmatrix}.$$

Загрузка процессоров (веса цветов в раскраске) будет следующей: $T_1 = 18$, $T_2 = 5$, $T_3 = 7$.

Для второй модификации:

$$A'=\begin{vmatrix}\infty & 5 & \infty\\ 8 & \infty & \infty\\ \infty & \infty & 7\\ 10 & 10 & \infty\end{vmatrix}.$$

Загрузка приборов: $T_1 = 8$, $T_2 = 15$, $T_3 = 7$.

Для третьей модификации:

$$A'=\begin{vmatrix}8 & \infty & \infty\\ \infty & \infty & 7\\ \infty & 5 & \infty\\ 10 & 10 & \infty\end{vmatrix}.$$

Загрузка приборов: $T_1 = 8$, $T_2 = 15$, $T_3 = 7$.

Таким образом «минимаксная» раскраска такова: вершины *A* и *D* красятся в один цвет, *B* во второй и *C*, соответственно, в третий.

Быстрый алгоритм. Описанный алгоритм находит «минимаксную» раскраску, но он весьма трудоёмок из-за нахождения всех максимально внутренне устойчивых подмножеств и раскрасок графа. Другой подход состоит в том, чтобы попробовать использовать известные эвристические алгоритмы поиска единственной раскраски графа, например более быстрый алгоритм [6, 7], который, однако, не всегда находит минимальную раскраску. Алгоритм упорядочивает вершины по степеням. Составляется список L вершин графа в порядке убывания степеней вершин. Выбирается цвет $j = 1$. Первая вершина x_i из списка красится в цвет j и удаляется из списка. Список просматривается на предмет наличия несмежных с x_i вершин, которые так же окрашиваются в цвет j и удаляются. Когда список просмотрен до конца и если он не пуст, то $j = j + 1$, x_i назначается очередная первая вершина списка и т. д. Алгоритм продолжается, пока список не опустеет.

Рассмотрим этот алгоритм на примере того же графа (рис. 1). Пусть $x_1 = A$, $x_2 = B$, $x_3 = C$, $x_4 = D$. $L = x_3, x_1, x_2, x_4$. В табл. 1 описаны шаги алгоритма.

Таблица 1

Выполнение быстрого алгоритма

Цвет j	Окрашенные вершины	Список L
1	$X_1 = \{x_3\}$	x_1, x_2, x_4
2	$X_2 = \{x_1\}$	x_2, x_4
2	$X_2 = \{x_1, x_4\}$	x_2
3	$X_3 = \{x_2\}$	

Значения T считаются по полученной раскраске и равны, соответственно: $T_1 = 7$, $T_2 = 15$, $T_3 = 8$.

Алгоритм с удалением вершин. В основе этого эвристического алгоритма [8] лежит идея удаления из графа вершин вместе с инцидентными рёбрами. Вводится счётчик цветов $p := 1$. Из графа удаляется вершина с максимальной локальной степенью вместе с инцидентными рёбрами. Процесс продолжается пока множество рёбер графа не пусто. Затем все не удалённые вершины красятся в цвет p, счётчик цветов увеличивается на единицу $p := p + 1$ и, пока в графе присутствуют не окрашенные вершины, алгоритм повторяется вновь. Необходимо добавить, что в начале каждого шага нужно восстановить в графе связи (рёбра) между неокрашенными вершинами.

Рассмотрим подробнее работу алгоритма на примере рассматриваемого графа (рис. 1).

Вершина C имеет максимальную степень и удаляется из графа. Множество рёбер не пусто и из получившегося графа удаляется очередная вершина с максимальной степенью — A (т. к. степени вершин A и B равны, не имеет значения какую из них удалять). Оставшиеся вершины B и D красятся в цвет 1. Неокрашенные вершины C и A образуют граф с единственным ребром. Их степени одинаковы, поэтому удаляем любую вершину, например A. Оставшаяся вершина C красится в цвет 2, счётчик цветов увеличивается. Единственная неокрашенная вершина A, соответственно, окрашивается в цвет 3.

Значения T рассчитываются по полученной раскраске и равны, соответственно: $T_1 = 18$, $T_2 = 7$, $T_3 = 5$.

Алгоритм Д. Этот весьма эффективный эвристический алгоритм [9] последовательной раскраски основан на использовании такой характеристики вершин, как степень насыщения. Степень насыщения вершины v — это количество различных цветов на смежных с v вершинах.

На начальном этапе алгоритма выбирается вершина с максимальной степенью (локальной, не насыщения) и окрашивается в цвет $p = 1$. На каждом последующем шаге выбирается вершина с максимальной степенью насыщения и окрашивается в минимально возможный для неё цвет. Под «минимально возможным» подразумевается минимально возможное значение цвета, исходя из цветов смежных вершин. Стоит отметить, что если вершины будут просматриваться в

порядке уменьшения их степеней, то такая жадная раскраска использует максимум $d + 1$ цветов, где d — максимальная степень вершин в графе [10].

Проиллюстрируем алгоритм на примере рассматриваемого графа (рис. 1).

Вершина с максимальной степенью — C окрашивается в цвет 1. У оставшихся вершин степени насыщения станут равными единице, поэтому всё равно, какую из них выбирать. Выберем вершину D. Минимально возможный для неё цвет — 2. У оставшихся неокрашенными вершин степени насыщения по-прежнему одинаковы. Выбираем вершину A — для неё минимально возможный цвет тоже 2. Для оставшейся вершины C — 3.

Значения T рассчитываются по полученной раскраске и равны, соответственно $T_1 = 7$, $T_2 = 15$, $T_3 = 8$.

Вычислительный эксперимент. Сравним эффективность предложенных подходов к решению задачи.

Сравнение проводится на 100 сгенерированных обыкновенных (без петель и кратных рёбер) графах с количеством вершин 12 и 14. Связи и веса вершин распределяются случайным образом по равномерному закону. Веса вершин находятся в диапазоне от 1 до 50.

Из сотни графов были выбраны те, для которых все быстрые алгоритмы нашли минимальные по количеству цветов раскраски.

Для 12 вершин из 500 графов нашлось 139, в которых все быстрые алгоритмы дали оптимальные по количеству цветов раскраски. Средние T_{max} для этих случаев приведены в табл. 2.

Среднее время работы алгоритмов для 12-вершинных графов приведены в табл. 3.

Таблица 2

Средние T_{max} в случае 12-вершинных графов

Алгоритм (модификация)	Среднее T_{max}
1-я модификация	78
2-я модификация	76
3-я модификация	75
Быстрый алгоритм	101
Алгоритм с удалением вершин	110
Алгоритм Д	102

Таблица 3

Время работы в случае 12-вершинных графов

Алгоритм (модификация)	Среднее время работы (мс)
Точный алгоритм	519,082
Быстрый алгоритм	0,004
Алгоритм с удалением вершин	0,054
Алгоритм Д	0,002

Таблица 4

Средние T_{max} в случае 14-вершинных графов

Алгоритм (модификация)	Среднее T_{max}
1-я модификация	81
2-я модификация	79
3-я модификация	79
Быстрый алгоритм	111
Алгоритм с удалением вершин	129
Алгоритм Д	115

Для 14 вершин из 300 графов нашлись 56, в которых все быстрые алгоритмы дали оптимальные по количеству цветов раскраски. Средние T_{max} для этих случаев приведены в табл. 4.

Среднее время работы алгоритмов для 14-вершинных графов приведены в табл. 5.

Таблица 5

Время работы в случае 14-вершинных графов

Алгоритм (модификация)	Среднее время работы (мс)
Точный алгоритм	40940,303
Быстрый алгоритм	0,010
Алгоритм с удалением вершин	0,003
Алгоритм Д	0,003

Выводы. Как продемонстрировал вычислительный эксперимент, средняя точность быстрых алгоритмов значительно ниже результатов точного алгоритма: на 35–40 % в случае быстрого алгоритма и алгоритма Д и на 40–60 % в случае алгоритма с удалением вершин. Однако вычислительные затраты точного алгоритма существенно превосходят аналогичные затраты быстрых алгоритмов и быстро растут по мере увеличения числа вершин.

Для небольших графов (примерно до 10 вершин) вполне можно использовать точный алгоритм. Для среднего размера графов (примерно от 12 вершин) выбор алгоритма уже не так очевиден, потому что точность решения обеспечивается длительной работой, что не всегда может быть приемлемо. Наконец, для больших графов (десятки вершин) задача может оказаться практически не решаемой за адекватное время точным алгоритмом и здесь, по-видимому, придётся выбирать быстрое решение.

Библиографический список

1. Карп, Р. М. Сводимость комбинаторных проблем / Р. М. Карп // Кибернетический сборник, вып. 12. — Москва : Мир, 1975. — С. 16–38.

2. Гэри, М. Вычислительные машины и труднорешаемые задачи / М. Гэри, Д. Джонсон. — Москва : Мир, 1982. — 416 с.

3. Букин, В. В. Алгоритм раскраски взвешенного графа / В. В. Букин, В. Г. Кобак // Известия СКНЦ ВШ. Техн. науки. — 1998. — №2. — С. 12–14.

4. Кофман, А. Введение в прикладную комбинаторику / А. Кофман. — Москва : Наука, 1975. — 480 с.

5. Кобак, В. Г. Модификация алгоритма обслуживания «по критическому пути» для систем с избирательными свойствами приборов / В. Г. Кобак // Микропроцессорные и цифровые системы. — 2003. — № 2 (6). — С. 156–162.

6. Емельянов, В. В. Теория и практика эволюционного моделирования / В. В. Емельянов, В. М. Курейчик, В. В. Курейчик. — Москва : ФИЗМАТЛИТ, 2003. — 432 с.

7. Кристофидес, Н. Теория графов. Алгоритмический подход / Н. Кристофидес. — Москва : Мир, 1988. — 432 с.

8. Койнов, Р. В. Практикум по дискретной математике / Р. В. Койнов, Л. С. Лисицына. — Санкт-Петербург : Изд-во СПбГУ ИТМО, 2004. — 78 с.

9. Евстигнеев, В. А. Применение теории графов в программировании / В. А. Евстигнеев. — Москва : Наука, 1985. — 352 с.

10. Welsh, D. J. A. An upper bound for the chromatic number of a graph and its application to timetabling problems / D. J. A. Welsh, M. B. Powell // The Computer Journal. — № 10 (1), 1967. — С. 85–86.

Материал поступил в редакцию 12.03.2014.

References

1. Karp, R. M. Svodimost kombinatornykh problem. [Reducibility of combinatorial problems.] Kiberneticheskiy sbornik, iss. 12. Moscow: Mir, 1975, pp. 16–38 (in Russian).

2. Garey, M., Johnson, D. Vychislitelnyye mashiny i trudnoreshayemyye zadachi. [Computers and Intractibility.] Moscow: Mir, 1982, 416 p. (in Russian).

3. Kobak, V. G., Bukin, V. V. Algoritm raskraski vzveshennogo grafa. [Algorithm of weighted graph coloring.] Izvestiya SKNTs VSh. Tekhnicheskiye nauki. 1998, no. 2, pp. 12–14 (in Russian).

4. Kauffman, A. Vvedeniye v prikladnuyu kombinatoriku. [Introduction to applied combinatorics.] Moscow: Nauka, 1975, 480 p. (in Russian).

5. Kobak, V. G. Modifikatsiya algoritma obsluzhivaniya «po kriticheskomu puti» dlya sistem s izbiratelnymi svoystvami priborov. [Operation algorithm modification "through the critical path" for systems with specific equipment properties.] Mikroprotsessornyye i tsifrovyye sistemy, 2003, no. 2 (6), pp. 156–162 (in Russian).

6. Yemelyanov, V. V., Kureychik, V. M., Kureychik, V. V. Teoriya i praktika evolyutsionnogo modelirovaniya. [Theory and practice of evolutionary modeling.] Moscow: FIZMATLIT, 2003, 432 p. (in Russian).

7. Christofides, N. Teoriya grafov. Algoritmicheskiy podkhod. [Graph theory. An algorithmic approach.] Moscow: Mir, 1988, 432 p. (in Russian).

8. Koynov, R. V., Lisitsyna, L. S. Praktikum po diskretnoy matematike. [Workshop on discrete mathematics.] Saint Petersburg: Izdatelstvo SPbGU ITMO, 2004, 78 p. (in Russian).

9. Yevstigneyev, V. A. Primeneniye teorii grafov v programmirovanii. [Application of graph theory in programming.] Moscow: Nauka, 1985, 352 p. (in Russian).

10. Welsh, D. J. A., Powell, M. B. An upper bound for the chromatic number of a graph and its application to timetabling problems. The Computer Journal, 1967, no. 10 (1), pp. 85–86.

COMPARATIVE ANALYSIS OF COLORING ALGORITHMS FOR ORDINARY WEIGHTED GRAPH*

A. S. Merzlenko, V. G. Kobak

Algorithms of solving the "minimax" weighted graph coloring problem are considered and compared. The mathematical formulation of the problem is presented; the solution optimality criteria are stated. The exact algorithm that always finds the minimum count of colors through Maghout method and then finds the "minimax" option using 3 versions of the critical path method is described. Three fast heuristic algorithms are described: the algorithm that works with the list of vertexes arranged by the local degrees; the algorithm based on the removal of the points and adjacent edges; the algorithm using the vertex saturation rate. All algorithms are considered with examples. To evaluate the algorithm efficiency, a computational experiment on several hundred of randomly generated graphs is set up. The algorithms were compared by the operating speed and the proximity of the result to the "minimax" version of coloring.

Keywords: *weighted graphs, weighted graph coloring, scheduling theory.*

* The research is done within the frame of the independent R&D.