УДК 681.142.1

А. М. Магомедов

Алгоритм раскраски ребер простого графа

Построен алгоритм проверки существования у двудольного графа такой реберной раскраски, что в каждой вершине v цвета, представленные в v , образуют множество последовательных целых чисел.

Библиография: 4 названия.

The algorithm checks the existence of bipartite graph edge-coloring such that at each vertex v the colors presented at v form a set of consecutive integers.

Bibliography: 4 items.

Ключевые слова: двудольный граф, множество, алгоритм, раскраска.

Keywords: bipartite graph, set, algorithm, colors.

Введение

Пусть в связном двудольном графе $G = (X, Y, E)$ ребро $(x_i, y_j) \in E$ указывает на предписание учителю y_j провести один урок в классе x_i . Составить однодневное расписание учебных занятий по исходным данным G – значит выбрать для каждого $(x_i, y_j) \in E$ целое положительное число $c(x_i, y_j)$ – номер академического часа, в течение которого учитель y_j проводит урок в классе x_i . Рассмотрение $c(x_i, y_j)$ в качестве цвета ребра (x_i, y_j) сводит задачу к задаче об *интервальной реберной раскраске* графа G (далее кратко: и.р.).

ОПРЕДЕЛЕНИЕ 1. Интервальной t -раскраской связного графа G называется правильная раскраска ребер G в цвета $1, 2, \dots, t$, при которой в каждый цвет окрашено хотя бы одно ребро, а ребра, инцидентные каждой вершине v , окрашены в $d(v)$ последовательных цветов [1].

Запрет на одновременное участие класса/учителя в двух и более уроках означает, что раскраска должна быть *правильной*, т.е. в каждой вершине все представленные в ней цвета [2] должны быть различны. А требование отсутствия «окон» у классов и учителей означает, что раскраска должна быть не только правильной, но и интервальной, т.е. в каждой вершине графа все представленные в ней цвета должны образовывать целочисленный интервал.

Работа выполнена при поддержке проекта № 1.6395.2017/БЧ «Теоретико-графовые и компьютерные аспекты исследования расписаний и устойчивость решений стохастических функционально-дифференциальных уравнений» в рамках базовой части государственного задания Минобрнауки России.

В данной статье приведен алгоритм проверки существования и.р., представляющий результат глубокой переработки алгоритма, приведенного в [3]; достаточно сказать, что количество попыток, необходимое для раскраски очередного ребра без перекрашивания остальных ребер, удалось уменьшить с 8 (в [3]) до 2 (в данной статье).

1. Подготовительный материал

1.1. Формулировка задачи. Рассматривается связный двудольный граф без висячих вершин $G = (V, E)$, $|V| = n$, $|E| = m$. Требуется проверить, существует ли и.р. графа G цветами $1, \dots, t$.

Согласно [1], t не может быть более $n - 1$; положим t равным $n - 1$.

ЗАМЕЧАНИЕ 1. Выбор $t = n - 1$ является единственным пунктом алгоритма, учитывающим двудольную природу графа G .

В случае несвязного графа задача сводится к рассмотрению каждой связной компоненты графа. Справедливость следующих утверждений доказывается элементарно.

УТВЕРЖДЕНИЕ 1. Пусть задана и.р. связного графа G и P — цепь, соединяющая два ребра с цветами t_1 и t_2 , где $1 \leq t_1 < t_2 \leq t$. Тогда для любого t_3 , $t_1 \leq t_3 < t_2$ найдется ребро цвета t_3 , инцидентное вершине из P .

УТВЕРЖДЕНИЕ 2. Определение 1 равносильно следующему: «интервальной t -раскраской связного графа G называется раскраска ребер G в цвета $1, \dots, t$, при которой найдутся ребра, окрашенные в цвета 1 и t , и для каждой вершины v цвета, представленные в v , образуют множество последовательных чисел».

УТВЕРЖДЕНИЕ 3. Если все связные графы $G = (V, E)$, $|V| = n$, $n < N$, обладают и.р., то любой связный граф с $|V| = N$ и висячей вершиной также обладает и.р.

1.2. Подход к решению. В результате анализа многочисленных примеров построения и.р. «вручную» предлагается следующий эвристический подход:

- сначала построить связное множество ребер P , начиная с произвольного ребра и постепенно расширяя множество P до всего E ; ребра нумеруются в порядке включения их в P и ориентируются с началом в той концевой вершине, которая включена в P ранее другой (или одновременно с ней);
- затем последовательно раскрашивать ребра множества P цветами из $[1..t]$ так, что в любой момент времени в каждой вершине v цвета, представленные в v (далее — кратко: «цвета вершины v ») — суть последовательные целые числа, иными словами, образуют интервал.

Любой алгоритм раскраски, обладающий таким свойством, будем называть *перманентно-интервальным* алгоритмом (или, сокращенно, *пи-алгоритмом*).

Далее рассматриваются только пи-алгоритмы.

1.3. Определения и обозначения.

- 1.3.1 Будем обозначать ориентированное ребро e_i множества $P = \{e_0, \dots, e_{m-1}\}$ через $e_i = (v1[i], v2[i])$; очевидная избыточность обозначений принята с целью упрощения кодирования пи-алгоритма. Например, равенства $v1[3] = 4$ и $v2[3] = 8$ означают, что $e_3 = (v_4, v_8)$.
- 1.3.2 Массив текущих списков цветов в вершинах множества V будем обозначать через L , $L = (L[0], \dots, L[n-1])$. Например, если список $L[10]$ цветов в вершине v_{10} равен $\{3, 4, 5\}$, это означает, что три ребра, инцидентные v_{10} , в данный момент раскрашены в цвета 3, 4 и 5 соответственно (вершина v_{10} может иметь и другие инцидентные ребра, нераскрашенные в настоящее время). Если список $L[10]$ пуст, то в настоящий момент времени не раскрашено ни одно ребро, инцидентное вершине v_{10} .
- 1.3.3 Приграничным цветом для пустого интервала назовем любой цвет из интервала $[1..t]$. *Нижним приграничным цветом* для непустого интервала $[a..b]$ назовем цвет $a - 1$, если $1 \leq a - 1 \leq t$, *верхним приграничным цветом* — цвет $b + 1$, если $1 \leq b + 1 \leq t$.
- 1.3.4 Пусть ребра $e_0, \dots, e_{i-1}$ уже раскрашены пи-алгоритмом и требуется раскрасить ребро e_i , $i > 0$. Понятно, что список цветов в начальной вершине ориентированного ребра $e_i = (v1[i], v2[i])$ не пуст. Очевидно также, что раскраска ребра e_i без изменения цветов ранее раскрашенных i ребер возможна только в следующих двух случаях:
- 1) нижний приграничный цвет списка цветов $L[v1[i]]$ начальной вершины ребра является приграничным цветом списка цветов $L[v2[i]]$ конечной вершины ребра;
 - 2) верхний приграничный цвет списка цветов $L[v1[i]]$ начальной вершины ребра является приграничным цветом списка цветов $L[v2[i]]$ конечной вершины ребра.
- Соответственно можно предпринять две попытки раскрасить ребро e_i без изменения цветов ранее раскрашенных ребер; номера 1 или 2 предпринятых попыток будем сохранять в ячейке q_i (удобно использовать и третье значение $q_i = 0$ в качестве признака, что ни одна из двух попыток пока не использована).
- Элементы массива q индексируются, начиная с единицы, в отличие от всех других массивов в нашем тексте; понятно, что подход к раскраске начального ребра $e_0 \in P$ существенно проще (см. ниже п. 2.2.1).
- 1.3.5 Для записи и хранения цветов i -го ребра множества P будем использовать ячейку $C[i]$. В качестве признака успешного (с генерацией интервальной раскраски) или неудачного (без получения интервальной раскраски) завершения пи-алгоритма используется логическая переменная Success.

ЗАМЕЧАНИЕ 2. В тексте использованы некоторые обозначения языков программирования, понятные без дополнительных объяснений. Наряду с привычным заключением индексов в квадратные скобки из соображений удобочитаемости текста допускается и запись индексов «справа-вниз».

Подготовительная работа для формулировки пи-алгоритма завершена. Алгоритм осуществляет перебор с возвратами и включает стадии «Инициализация», «Прямой шаг» и «Возвратный шаг».

2. Формулировка пи-алгоритма

2.1. Инициализация. Интервальная раскраска отсутствует: Success=false;

всем $v1[i]$ ($v2[i]$) присвоены номера начальных (конечных) вершин ориентированных ребер e_i множества P , $i = 0, \dots, m - 1$;

списки цветов всех вершин пусты: $L[0] = L[1] = \dots = L[n - 1] = \emptyset$;

все элементы массивов q и C равны нулю:

$q[1] = \dots = q[m - 1] = 0$, $C[0] = \dots = C[m - 1] = 0$;

начальное ребро e_0 раскрашено в наименьший допустимый цвет:

$C[0] = 1$; $L[v1[0]] = L[v2[0]] = \{1\}$;

номер очередного ребра установлен равным единице: $i = 1$.

2.2. Прямой шаг. Данный пункт выполняется сразу после инициализации и после возвратного шага и заключается в раскраске (попытке раскраски) ребра, номер которого находится в ячейке i .

FWRD:

Введем рабочие переменные без индексов: $v1i := v1[i]$; $v2i := v2[i]$;

2.2.1. Рассмотрим сначала случай $i = 0$:

если $C[0] < t$, то $\{C[0] ++$; добавить $C[0]$ в списки $L[v1i]$ и $L[v2i]$; $i ++$; перейти к FWRD}

иначе $\{\text{Success}=\text{false}$; завершить алгоритм}.

Замечание к случаю $i = 0$: при первом выполнении прямого шага сразу после инициализации i равно единице, однако переход к прямому шагу после возвратных шагов может быть осуществлен при $i = 0$.

2.2.2. Рассмотрим случай $i > 0$.

Увеличим на 1 текущий номер попытки раскрасить ребро e_i : $q_i ++$.

В зависимости от $q_i = 1$ или $q_i = 2$ перейти к пунктам 2.2.2.1 или 2.2.2.2 соответственно.

2.2.2.1. Выполняется при $q_i = 1$.

Если

нижний приграничный цвет $s = L[v1i].\text{Min}() - 1$ для начальной вершины $v1i$ рассматриваемого ребра e_i существует ($s \geq 1$) и является приграничным цветом для конечной вершины $v2i$ этого ребра,

то

{

$C[i] := s$; добавить s в списки $L[v1i]$ и $L[v2i]$;

если $i < m - 1$, то $\{i ++$; перейти к метке FWRD},

иначе $\{\text{Success}=\text{true}$; завершить алгоритм}

}

иначе $\{q_i := 2$; перейти к следующему пункту 2.2.2.2}.

2.2.2.2. Выполняется при $q_i = 2$.

Если верхний приграничный цвет $s = L[v1i].Max() + 1$ для начальной вершины $v1i$ рассматриваемого ребра e_i существует ($s \leq t$) и является приграничным цветом для конечной вершины $v2i$ этого ребра,

то

{

$C[i] := s$; добавить s в списки $L[v1i]$ и $L[v2i]$;

если $i < m - 1$, то $\{i++\}$; перейти к метке FWRD},

иначе {Success:=true; завершить алгоритм}

}

иначе перейти к метке BACK.

2.3. Обратный шаг. Данный пункт выполняется, если в прямом шаге не удалось раскрасить ребро e_i , $i > 0$, $q_i = 2$.

BACK:

$q_i := 0$; $i--$; удалить $C[i]$ из списков $L[v1[i]]$ и $L[v2[i]]$;

перейти к FWRD.

Конец пи-алгоритма

Заключение

Масштабные компьютерные эксперименты (алгоритм протестирован на более 366 тыс. графах) показали 84%-ю надежность пи-алгоритма. На рис. 1 приведен пример и.р., найденной алгоритмом.

Рис. 1. Цвета и.р. выписаны рядом с соответствующими ребрами.

Известно, что для любого двудольного графа на 14 вершинах и.р. существует [4]; отсюда и из Утверждения 3 следует существование и.р. для двудольного графа на 15 вершинах (с висячей вершиной y_{10}), изображенного на рис. 2. Однако пи-алгоритм оказался не в состоянии найти эту и.р.

Рис. 2. Известно, что и.р. для этого графа существует.

Список литературы

- [1] Асратян А.С., Камалян Р.Р. Интервальные раскраски ребер мультиграфа // Прикладная математика, вып. 5. Ереван: Изд-во Ереванского ун-та. 1987. С. 25–34.
- [2] Свами М., Тхуласираман К. Графы, сети и алгоритмы. Москва: Мир. 1984. 455 с.
- [3] Магомедов А.М. Цепочечные структуры в задачах о расписаниях // Прикладная дискретная математика. 2016. Т. 3 (33). С. 67–77.
- [4] Giaro K. Compact task scheduling on dedicated processors with no waiting period (in Polish) // PhD thesis, Technical University of Gdansk, IETI Faculty, Gdansk. 1999.

А. М. Магомедов (A. M. Magomedov)

Дагестанский научный центр РАН

E-mail: magomedtagiri@yandex.ru

Поступила в редакцию

01.07.2016