УДК 519.1+004.9

DOI: 10.21779/2542-0321-2017-32-3-74-84

А.М. Магомедов, Т.А. Магомедов

Сокращенное перечисление двудольных графов заданного порядка[1]

Дагестанский государственный университет; Россия, 367000, г. Махачкала, ул. М. Гаджиева, 43а; magomedtagir1@yandex.ru

Для проверки существования интервальной реберной раскраски связного двудольного графа и ее построения в случае существования сконструирован жадный алгоритм, который осуществляет перебор с возвратами последовательных ребер кусочно-непрерывного пути; последний представляет собой такой упорядоченный набор всех ребер графа, что подграф, порожденный любым подмножеством его ребер с номерами 1, 2, …, является связным.

Построен алгоритм, позволяющий для множества S всех двудольных графов заданного порядка H сгенерировать множество S_0 двудольных графов существенно меньшей мощности, содержащее для каждого графа G из S граф, изоморфный G. Применением жадного алгоритма к каждому графу из S_0 с привлечением компьютерных ресурсов показано, что при малых значениях H все графы из S_0 интервально раскрашиваемы.

В частности, установлено, что все двудольные графы G = (X, Y, E) порядка 16 при |X|<7 обладают интервальной раскраской; отсюда следует, что достаточно проверить интервальную раскрашиваемость двудольных графов порядка 16 при |X| = 7 и |X| = 8. Программное обеспечение, разработанное авторами на основе упомянутых алгоритмов, обладает также рядом дополнительных свойств, способствующих графическому отображению интервальной раскраски в удобной для восприятия форме.

Ключевые слова: *двудольный граф, интервальная раскраска, программа.*

1. Задачи теории расписаний и раскраска графов

1.1. Введение

Исследования в области теории расписаний приводят к появлению новых методов и даже целых направлений в теории графов. Например, знаменитая проблема четырех красок возникла в связи с задачами теории расписаний и разбиений [1, с. 110]. В свою очередь, с проблемой четырех красок связана задача о правильной вершинной раскраске графа в три цвета, т. е. задача о существовании такого отображения множества вершин графа в множество цветов {1, 2, 3}, что концевым вершинам каждого ребра сопоставляются разные цвета. Задача раскраски графа в три цвета – полна [1, с. 101], следовательно, не может быть разрешима за полиномиальное время, если верна знаменитая гипотеза «$NP \neq P$» (отметим недавнее сообщение Норберта Блюма о доказательстве данной гипотезы [2] и работу [3]).

Некоторые задачи теории расписаний формулируются в терминах *реберной* раскраски. Раскраску множества ребер E будем называть правильной реберной раскраской графа $G = (V, E)$, если в каждой вершине все *представленные в ней цвета* (цвета всех инцидентных ребер) различны. Следуя работе [4], *интервальной t-раскраской* графа $G = (V, E)$ будем называть правильную раскраску ребер G в цвета $1, 2, \ldots, t$, при

[1] Работа выполнена при поддержке Дагестанского государственного университета и отдела математики и информатики ДНЦ РАН.

которой в каждый цвет окрашено хотя бы одно ребро, а для каждой вершины $v \in V$ ребра, инцидентные v, окрашены в $d(v)$ последовательные цвета. Всюду в дальнейшем рассматриваются только реберные раскраски. Если при каком-либо t граф имеет интервальную раскраску, будем говорить, что граф *интервально раскрашиваем*. В [4] доказано, что задача об интервальной раскрашиваемости – полная.

Пусть планируется обработать множество заданий $X = \{x_0, \dots, x_{n-1}\}$ в системе специализированных процессоров $Y = \{y_0, \dots, y_{m-1}\}$; для каждого задания x_i определен набор процессоров, каждый из которых должен выполнить одну операцию по обработке x_i; длительность операции равна единице, запрещено одновременное участие объекта (задания, процессора) в двух и более операциях, условия предшествования отсутствуют. Требуется выяснить, существует ли расписание выполнения обработки без простоев объектов.

Рассмотрим двудольный граф $G = (X, Y, E)$, где $(x_i, y_j) \in E$, если только процессору y_j предписано выполнить операцию над заданием x_i. Под расписанием длины t будем понимать отображение множества ребер E в множество $\{1, 2, \dots, t\}$, где каждому ребру $(x_i, y_j) \in E$ сопоставляется $c(x_i, y_j)$ – номер дискретного момента времени, в течение которого y_j выполняет операцию над x_i. Если рассматривать $c(x_i, y_j)$ в качестве цвета ребра (x_i, y_j), получим эквивалентную задачу интервальной реберной раскраски двудольного графа. Запрет на одновременное участие объекта в двух и более операциях в терминологии теории графов означает, что раскраска должна быть *правильной*, а отсутствие простоев – что раскраска должна быть и *интервальной*, т. е. в каждой вершине графа все представленные в ней цвета должны образовать целочисленный интервал (в дальнейшем под интервалом будем понимать целочисленный интервал, интервал $\{a, a+1, \dots, b\}$ будем обозначать $[a..b]$). В [5] показано, что задача об интервальной раскрашиваемости заданного двудольного графа полная.

1.2. Свойства интервальной раскраски

Естественно ожидать, что двудольные графы с малым порядком интервально раскрашиваемы. Возникает вопрос: каково самое сильное ограничение на количество вершин, при котором задача все еще остается NP-полной? С применением компьютерных ресурсов в [6] доказано следующее утверждение.

Утверждение 1. Все двудольные графы с числом вершин не более 14 интервально раскрашиваемы.

В ряде статей (например, в [7]) указывается, что вопрос интервальной раскрашиваемости двудольных графов порядка 15, 16, 17 и 18 остается открытым.

Следствие. Если двудольный граф $G = (X, Y, E)$ с числом вершин 15 несвязный или обладает висячим ребром, то граф G интервально раскрашиваем.

Интересно, что и малая мощность только одной доли способна обеспечить двудольному графу $G = (X, Y, E)$ интервальную раскрашиваемость.

Утверждение 2 [8]. Если для двудольного графа $G = (X, Y, E)$ выполнены неравенства $|X| \leq 3$ или $|Y| \leq 3$, то граф G интервально раскрашиваем.

Утверждение 3. Пусть задана интервальная раскраска связного графа и $P = (a_0, b_0, a_1, b_1, \dots, b_{n-1}, a_n)$ – цепь, соединяющая два ребра: $b_0 = (a_0, a_1)$ и $b_{n-1} = (a_{n-1}, a_n)$ с цветами t_0 и t_{n-1} соответственно, $t_0 < t_{n-1}$. Тогда для любого τ, $t_0 < \tau < t_{n-1}$ найдется ребро цвета τ, инцидентное некоторой вершине из P.

Доказательство. Обозначим интервал цветов, представленных в вершине a_i, через $[m_i..M_i]$; $m' = \min_i m_i$, $M' = max_i M_i$. Так как каждые два соседних интервала последовательности $[m_0..M_0], [m_1..M_1], \dots, [m_{n-1}..M_{n-1}]$ имеют непустое пересечение, то каждый цвет из интервала $[m'..M']$ представлен в некоторой вершине цепи P. Поскольку $t_0 \in [m_0..M_0]$, $t_{n-1} \in [m_{n-1}..M_{n-1}]$, то t_0, $t_{n-1} \in [m'..M']$; следовательно, и $\tau \in [m'..M']$, поэтому цвет τ представлен в некоторой вершине цепи P.

Из *Утверждения 3* немедленно вытекает следующее утверждение.

Утверждение 4. Приведенное выше определение интервальной раскраски из работы [4] равносильно следующему: «Интервальной раскраской связного графа G называется раскраска ребер G в цвета $1, 2, \dots, t$, при которой найдутся ребра, окрашенные в цвета 1 и t, и для каждой вершины v цвета, представленные в v, образуют множество последовательных чисел».

Утверждение 5. [4] Если двудольный граф на k вершинах обладает интервальной -раскраской, то $t \leq k - 1$.

Двудольный граф $G = (X, Y, E)$, где $|X| = n$, $|Y| = m$, будем называть (n, m)-*графом*. Подход, предложенный в следующих разделах, в частности, позволил доказать, что все (n, m)-*графы* порядка 15 интервально раскрашиваемы.

2. Перечисление нормализованных (n, m)-графов

2.1. Нормализованное представление двудольного графа

Рассмотрим двудольный граф $G = (X, Y, E)$, где $X = \{x_0, \dots, x_{n-1}\}$, $Y = \{y_0, \dots, y_{m-1}\}$.

Подмножество вершин Y, смежных (не смежных) вершине $x_i \in X$, обозначим через Y_i ($\bar{Y}_i$); $i = 0, \dots, n-1$. Множество индексов вершин подмножества $Z \subset Y$, образованного последовательными вершинами из множества Y, будем называть *индексным интервалом* множества Z. Введем понятие *нормализованного представления* графа $G = (X, Y, E)$.

1. Автоморфизмом множества Y разместим вершины множества Y_0 в начальных позициях множества Y. Тем самым интервал $[0..n-1]$ разбивается на индексные интервалы множеств Y_0 и $\bar{Y}_0$.

2. Пусть $0 < i < n-1$ и в результате некоторого автоморфизма α интервал $[0..n-1]$ разбит на 2^i последовательных интервала – индексные интервалы некоторых множеств $Z_0, \dots, Z_{2^i-1}$, отдельные из которых могут быть пустыми. Выполним для каждого Z_j автоморфизм β_j, размещающий все вершины множества $Z_j \bigcap Y_i$ в начальных позициях множества Z_j, а вершины множества $Z_j \bigcap \bar{Y}_i$ соответственно в последних позициях множества Z_j. Суперпозицию $\alpha \cdot \beta_0 \cdot \dots \cdot \beta_{2^i-1}$ снова обозначим через α. Автоморфизм α индуцирует разбиение интервала $[0..n-1]$ на 2^{i+1} последовательных интервала – индексные интервалы множеств

$$Z_0 \bigcap Y_i,\ Z_0 \bigcap \bar{Y}_i, \dots,\ Z_{2^i-1} \bigcap Y_i,\ Z_{2^i-1} \bigcap \bar{Y}_i. \quad (1)$$

Увеличим i на единицу и переобозначим множества (1) через $Z_0, \dots, Z_{2^i-1}$ последовательно слева направо.

Пока $i < n-1$, повторим выполнение пункта **2**, по достижении равенства $i = n-1$ перенумеруем вершины множества X по невозрастанию их степеней:

$$d(x_0) \geq d(x_1) \geq \cdots d(x_{n-1}) \quad (2)$$

и полученный граф назовем *нормализованным графом* или – если необходимо подчеркнуть связь с графом $G = (X, Y, E)$ – *нормализованным представлением графа* $G = (X, Y, E)$.

2.2. Схема перечисления нормализованных (n, m)-графов

В данном разделе приведено развернутое изложение идеи работы [9]. Значение p всюду в дальнейшем выбирается равным 15 или 16. Очевидно, граф $G = (X, Y, E)$ изоморфен своему нормализованному представлению. Поэтому достаточно проверить интервальную раскрашиваемость всех нормализованных (n, m)-графов. Несмотря на то, что среди них могут встретиться изоморфные графы, компьютерный перебор всех нормализованных (n, m)-графов потребует при $n + m = p$ «значительно меньше» времени, нежели перебор всех (n, m)-графов. Существенному сокращению времени перебора способствует и то обстоятельство, что мы ограничимся связными графами без висячих вершин (см. следствие *Утверждения 1*) и случаем $3 < n$ (см. *Утверждение 2*).

Замечание 1. Задача об изоморфизме графов является «открытой» задачей, относительно которой неизвестно, является ли она полной.

При реализации алгоритма в многократных внутренних итерациях очередного вложенного цикла по $i = 0, \dots, n-1$ используются целочисленные переменные a_{ij}, которые пробегают значения из соответствующих индексных интервалов (на рис. 1 – наименьшие по длине интервалы, охватывающие a_{ij}), формируются суммы $d(x_i) = a_{i0} + a_{i1} + \cdots + a_{i,2^i-1}$. Если неравенство (2) выполнено, то $d(x_i)$ вершины множества Y с соответствующими индексными интервалами длины a_{ij} выбираются в качестве вершин, смежных вершине x_i $(i = 0, \dots, n-1)$.

Схема перечисления всех нормализованных (n, m)-графов представлена на рис. 1 (ввиду очевидной проблемы отображения деталей рисунок приведен для случая *p = 15*, распространение на случай *p = 16* тривиально). Продемонстрируем на примерах использование этой схемы.

Пример 1. Генерация списков смежности для вершины x_3.

$d(x_3) = a_{30} + a_{31} + a_{32} + a_{33} + a_{34} + a_{35} + a_{36} + a_{37}$. В список вершин, смежных x_3, выбираются по a_{3j} вершины множества Y, индексы которых суть a_{3j} значений из начала интервала $Z_{3j}, j = 0, \dots, 7$, где:

$Z_{30} = [0..a_{20} - 1]$, $Z_{31} = [a_{20}..a_{10} - 1]$, $Z_{32} = [a_{10}..a_{10} + a_{21} - 1]$, $Z_{33} = [a_{10} + + a_{21}..a_0 - 1]$, $Z_{34} = [a_0..a_0 + a_{22} - 1]$,

$Z_{35} = [a_0 + a_{22}..a_0 + a_{11} - 1]$, $Z_{36} = [a_0 + a_{11}..a_0 + a_{11} + a_{23} - 1]$, $Z_{37} = [a_0 + + a_{11} + a_{23}, \dots, m - 1]$.

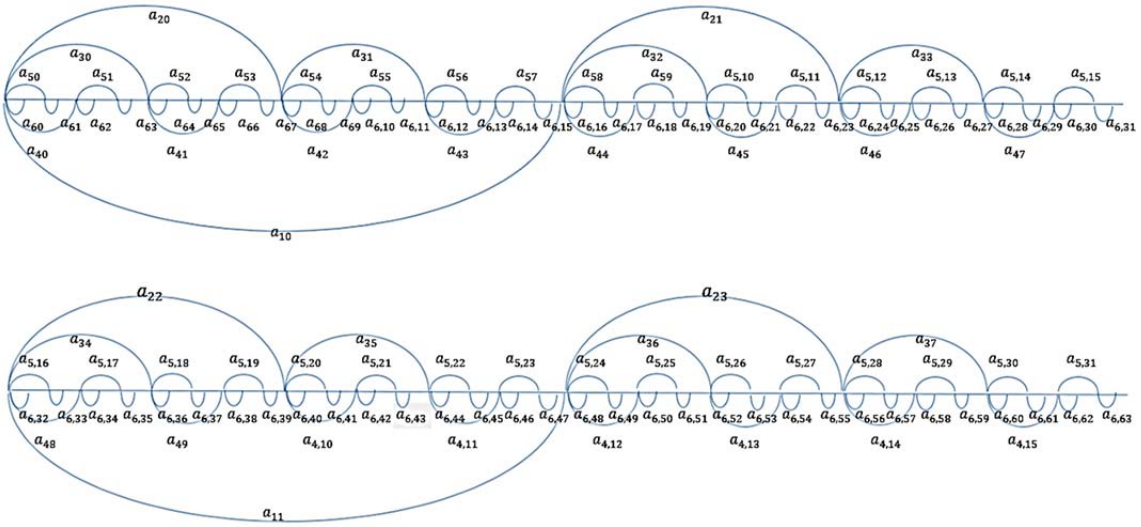


Рис. 1. Схема генерации всех нормализованных $(n,\ m)$-графов, $n \leq 7$; нижняя часть рисунка является непосредственным продолжением верхней части. При $n < 7$ рисунок корректирует-

Пример 2. Генерация списков смежности для вершины x_4.

$d(x_4) = a_{40} + a_{41} + a_{42} + a_{43} + a_{44} + a_{45} + a_{46} + a_{47} + a_{48} + a_{49} + a_{4,10} + a_{4,11} + a_{4,12} + a_{4,13} + a_{4,14} + a_{4,5}$. В список смежных x_4 вершин будем выбирать по a_{4j} вершин множества Y с индексами из начала интервала $Z_{4j}, j = 0, \dots, 15$, где:

$Z_{40} = [0..a_{30} - 1]$,
$Z_{41} = [a_{30}..a_{20} - 1]$,
$Z_{42} = [a_{20}..a_{20} + a_{31} - 1]$,
$Z_{43} = [a_{20} + a_{31}..a_{10} - 1]$,
$Z_{44} = [a_{10}..a_{10} + a_{32} - 1]$,
$Z_{45} = [a_{10} + a_{32}..a_{10} + a_{21} - 1]$,
$Z_{46} = [a_{10} + a_{21}..a_{10} + a_{21} + a_{33} - 1]$,
$Z_{47} = [a_{10} + a_{21} + a_{33}..a_0 - 1$
$Z_{48} = [a_0..a_0 + a_{34} - 1]$,
$Z_{49} = [a_0 + a_{34}..a_0 + a_{22} - 1]$,
$Z_{10} = [a_0 + a_{22}..a_0 + a_{22} + a_{35} - 1]$,
$Z_{4,11} = [a_0 + a_{22} + a_{35}..a_0 + a_{11} - 1]$,
$Z_{4,12} = [a_0 + a_{11}..a_0 + a_{11} + a_{36} - 1]$,
$Z_{4,13} = [a_0 + a_{11} + a_{36}..a_0 + a_{11} + a_{23} - 1]$,
$Z_{4,14} = [a_0 + a_{11} + a_{23}..a_0 + a_{11} + a_{23} + a_{37} - 1]$,
$Z_{4,15} = [a_0 + a_{11} + a_{23} + a_{37}..m - 1]$.

Справедливость следующего утверждения очевидна.

Утверждение 6. Пусть S – множество всех связных (n, m)-графов без висячих вершин, S_0 – множество всех нормализованных (n, m)-графов. Для каждого графа G из S в множестве S_0 найдется граф, изоморфный графу G.

3. Жадный алгоритм интервальной раскраски

3.1. Кусочно-непрерывный путь в связном графе

В разделе 3.2 изложена модификация жадного алгоритма проверки интервальной раскрашиваемости, приведенного в [10]. В формулировке алгоритма используется понятие *кусочно-непрерывного пути* в графе G. Приведем логическую (с результатом *True* или *False*) процедуру генерации кусочно-непрерывного пути.

Пусть в графе G выбрана произвольная непродолжимая цепь P и логической переменной *Result* присвоено значение *True*.

Пока не все ребра графа G включены в P, выполним следующие три действия: 1) если среди вершин цепи P найдется вершина с инцидентным ребром, не включенным в P, то последнюю из таких вершин обозначим a_k и перейдем к второму действию, в противном случае присвоим переменной *Result* значение *False* и «досрочно» завершим цикл; 2) построим непродолжимую цепь P_1, начинающуюся с вершины a_k и содержащую лишь ребра, не включенные в P; 3) результат конкатенации P и P_1 обозначим снова через P.

Если по завершении цикла значение переменной *Result* равно *True*, то последовательность ребер графа G, перечисленных в том порядке, в каком они следуют в итоговом P, будем называть кусочно-непрерывным путем в графе G, в противном случае заключаем, что граф G не является связным и кусочно-непрерывный путь не существует.

3.2. Жадный алгоритм интервальной раскраски

Напомним, что требуется проверить, существует ли интервальная раскраска заданного связного $(n,\ m)$-графа $G = (X,\ Y,\ E)$ не более чем t цветами (и выполнить раскраску в случае существования).

Количество вершин и ребер графа G обозначим через H и h соответственно ($H = n + m$). В соответствии с *Утверждением 5* положим t равным $H - 1$ (это единственный пункт алгоритма, учитывающий двудольную природу графа G). Использованные далее обозначения, присущие языкам программирования, понятны без дополнительных объяснений. Например, i++ (i– –) означает увеличение (уменьшение) значения переменной i на единицу, скобки { } используются для обозначения блока (группового оператора).

Построим кусочно-непрерывный путь $P = \{e_0,\ \dots,\ e_{h-1}\}$ и будем последовательно окрашивать ориентированные ребра $e_i = (v_i,\ w_i)$ цветами из целочисленного интервала $[1..t]$ так, что в любой момент времени в каждой вершине v представленные в v цвета образуют некоторый список L_v, составленный из последовательных целых чисел. Алгоритм раскраски, обладающий таким свойством, будем называть *жадным алгоритмом*. Массив текущих списков цветов, представленных в вершинах множества V, будем обозначать через L, $L = \{L_{v_0},\ \dots,\ L_{v_{H-1}}\}$. *Приграничным цветом* для пустого интервала назовем любой цвет из интервала $[1..t]$, для непустого интервала $[a..b]$ *нижним приграничным цветом* назовем цвет $a - 1$, если $1 \le a - 1 \le t$; *верхним приграничным цветом* – цвет $b + 1$, если $1 \le b + 1 \le t$; в случае существования нижний и верхний приграничные цвета для списка L_{v_i} будем обозначать через $low(v_i)$ и $high(v_i)$ соответственно.

Рассмотрим ситуацию, когда ребра $e_0,\ \dots,\ e_{i-1}$ уже окрашены и требуется окрасить ребро e_i, $i > 0$. Список цветов в начальной вершине v_i ориентированного ребра

$e_i = (v_i, w_i)$ не пуст, т. к. содержит цвет ребра e_{i-1}. Очевидно, что окраска ребра e_i без изменений цветов ранее окрашенных i ребер возможна только в следующих двух случаях: 1) $low(v_i)$ существует и является приграничным цветом для w_i; 2) $high(v_i)$ существует и является приграничным цветом для w_i. Соответственно допустимы две попытки окрасить ребро e_i без изменений цветов ранее окрашенных 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 предпринята). Заметим, что элементы массива q индексируются, начиная с единицы, в отличие от всех других массивов, используемых в нашем алгоритме.

Для записи и хранения цвета ребра e_i будем использовать ячейку C_i. В качестве признака успешного (с построением интервальной раскраски) или неудачного (без построения интервальной раскраски) завершения жадного алгоритма используется логическая переменная $Success$.

Перейдем к изложению жадного алгоритма, который осуществляет перебор с возвратами и кроме стадии инициализации включает итерации с двумя пунктами: «Прямой шаг» и «Возвратный шаг».

1. Инициализация

$Success := false$; всем v_i (соответственно w_i) присвоить номера начальных (конечных) вершин ориентированных ребер e_i из кусочно-непрерывного пути P, $i = 0, \dots, h-1$; для всех вершин выбрать пустые списки цветов: $L_{v_0} := \emptyset$; …; $L_{v_{H-1}} := \emptyset$; всем элементам массивов q и C присвоить нулевые значения: $q_1 := 0$; …; $q_{h-1} := 0$; $C_0 := 0$; …; $C_{h-1} := 0$; начальное ребро e_0 окрасить в цвет 1: $C_0 := 1$; $L_{v_0} := \{1\}$; $L_{w_0} = \{1\}$; номер очередного окрашиваемого ребра выбрать равным единице: $i := 1$.

2. Прямой шаг

FWRD: данный пункт выполняется сразу после инициализации или же после возвратного шага и заключается в окрашивании (более точно, в попытке окрашивания) ребра, номер которого записан в ячейке i.

2.1. Рассмотрим сначала случай $i = 0$: если $C_0 < t$, то {C_0++; добавить C_0 в списки L_{v_i} и L_{w_i}; i++; перейти к метке FWRD} иначе {$Success := false$; завершить алгоритм}.

Заметим, что при первом выполнении прямого шага сразу после инициализации значение i равно единице, однако переход к прямому шагу после возвратных шагов может быть осуществлен и при $i = 0$.

2.2. Рассмотрим случай $i > 0$.

Увеличить на единицу текущий номер попытки раскрасить ребро e_i: q_i++.

В зависимости от равенств: $q_i = 1$ или $q_i = 2$ перейти к пунктам 2.2.1 или 2.2.2 соответственно.

2.2.1. – Данный пункт выполняется при $q_i = 1$.

Если $low(v_i)$ для начальной вершины v_i рассматриваемого ребра e_i существует и является приграничным цветом для конечной вершины w_i этого ребра, то

{

$C_i := low(v_i)$; добавить $low(v_i)$ в списки L_{v_i} и L_{w_i}; если $i < h-1$, то {i++; перейти к метке FWRD}, иначе {$Success := true$; завершить алгоритм}

},

иначе {$q_i := 2$; перейти к следующему пункту 2.2.2}.

2.2.2. – Данный пункт выполняется при $q_i = 2$.

Если $high(v_i)$ для начальной вершины v_i рассматриваемого ребра e_i существует и является приграничным цветом для конечной вершины w_i этого ребра, то

{

$C_i := high(v_i)$; добавить $high(v_i)$ в списки L_{v_i} и L_{w_i};

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

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

}

иначе перейти к метке BACK.

2.3. Обратный шаг

BACK: данный пункт выполняется, если в прямом шаге не удалось окрасить ребро e_i, $0 < i,\ q_i = 2$.

Пока $(i > 0)$ и $(q_i = 2)$

{

$q_i := 0;\ i--$; удалить C_i из списков L_{v_i} и L_{w_i};

если $i = 0$, то перейти к метке FWRD;

если $q_i < 2$, то перейти к метке FWRD;

}

Конец алгоритма.

Замечание 2. Жадный алгоритм «привязан» к выбранному кусочно-непрерывному пути и не выполняет полный перебор попыток интервальной раскраски.

4. Результаты вычислений

4.1. Вычислительная схема

В авторском программном обеспечении, разработанном на языке C#, на основе алгоритмов, изложенных в разделах 2 и 3, принята следующая схема.

Для каждого $n = 4, 5, 6, 7$ *при* $p = 15$ (*и для каждого* $n = 4, 5, 6, 7, 8$ *при* $p = 16$) *генерируются все нормализованные* $(n, p - n)$*-графы, при этом непосредственно после генерации очередного графа* G *и увеличения счетчика* N *выполняется следующая процедура: если в графе* G *существует кусочно-непрерывный путь* (*связность графа* G) *и степени всех вершин графа* G *больше единицы* (*отсутствие висячих ребер*)*, то с помощью жадного алгоритма проверяется существование интервальной раскраски графа* G.

Как видно из схемы, в N подсчитывается не количество нормализованных $(n, p - n)$-графов, а число сгенерированных $(n,\ p - n)$-графов еще до выполнения проверки связности (совмещенной с проверкой существования кусочно-непрерывного пути) и наличия висячих ребер. Такие графы будем называть *преднормализованными*.

Во всех случаях результаты проверки интервальной раскрашиваемости нормализованных $(\mathrm{n}, 15 - \mathrm{n})$-графов оказались положительными. С учетом *Утверждения 6* это означает, что все двудольные графы порядка 15 интервально раскрашиваемы; при $p = 16$ интервальная раскрашиваемость установлена для всех $(n,\ p - n)$-графов с $n < 7$. Таким образом, для двудольных графов порядка 16 вопрос об интервальной раскрашиваемости остается невыясненным лишь для (7, 9)-графов и (8, 8)-графов.

Таблица 1. Информация о длительности вычислений для преднормализованных $(n,\ 15-n)$-графов порядка 15

n	$d(x_6)$	N	Время в сек.
4		366956	3,4
5		35135708	441,2
6		984643930	29275,4
7	2	1368967576	47399,0
7	3	1175207229	24086,1
7	4	341971501	15350,6
7	5	23364103	1547,1
7	6	274842	10,7
7	7	685	0,035
7	8	1	0,003

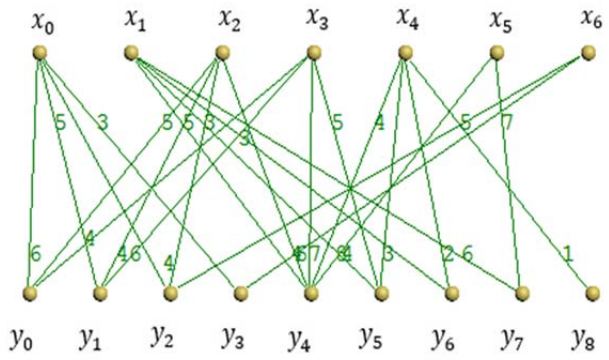
4.2. Информация о длительности вычислений

Понятно, что в режиме вытесняющей мультизадачности длительность вычислений носит условный характер и стремление к чрезмерной точности вычислений здесь неуместно. Тем не менее мы нашли допустимым записать в последней колонке таблицы 1 длительности вычислений, измеренные в одном из компьютерных экспериментов (временны́е параметры в других экспериментах несколько отличались от приведенных в таблице). Вычисления для $(n, 15-n)$-графов проводились отдельно для $n = 4,\ 5,\ 6$ и 7; при этом для $n = 7$ рассматривались семь случаев в зависимости от степени вершины x_6 (см. второй столбец таблицы). Значение N в предпоследней колонке таблицы – количество преднормализованных графов; количество нормализованных графов (для которых только и запускается жадный алгоритм), соответствующих тому или иному конкретному случаю, существенно меньше.

Заключение

Предложенный в разделах 2 и 3 подход не только позволяет доказать, что все $(n,\ 15-n)$-*графы* интервально раскрашиваемы, но и открывает перспективы для проверки интервальной раскрашиваемости двудольных графов большего порядка. Например, вычислительные эксперименты для $(n,\ 16-n)$-графов показали, что в случаях $n = 4,\ 5$ и 6 все преднормализованные $(n,\ 16-n)$-графы интервально раскрашиваемы, при этом значения N и длительности вычислений равны соответственно: при $n = 4$: 767910 графов, 7,3 сек; при $n = 5$: 126091168 графов, 1993,0 сек; при $n = 6$: 6536939854 графов, 95178,2 сек.

Характеристики использованного авторами компьютерного обеспечения: Surface, Windows 10 Pro, Processor: Intel(R) Core(TM) i5-4300U CPU © 1.90GHz 2.50 GHz, RAM: 8.00 GB, System type: 64-bit Operating System, x64-based.



$c(x_0, y_0) = 6;\ c(y_0, x_2) = 5;$
$c(x_2, y_1) = 4;\ c(y_1, x_0) = 5;$
$c(x_0, y_2) = 4;\ c(y_2, x_2) = 3;$
$c(x_2, y_4) = 6;\ c(y_4, x_1) = 5;$
$c(x_1, y_5) = 4;\ c(y_5, x_3) = 5;$
$c(x_3, y_0) = 4;\ c(x_3, y_1) = 6;$
$c(x_3, y_4) = 7;\ c(y_4, x_4) = 4;$
$c(x_4, y_5) = 3;\ c(x_4, y_6) = 2;$
$c(y_6, x_1) = 3;\ c(x_1, y_7) = 6;$
$c(y_7, x_5) = 7;\ c(x_5, y_4) = 8;$
$c(x_4, y_8) = 1;\ c(y_2, x_6) = 5;$
$c(x_6, y_3) = 4;\ c(y_3, x_0) = 3;$

Рис. 2. Пример вывода интервальной раскраски преднормализованного (7,9)-графа

Авторское программное обеспечение обладает некоторыми дополнительными функциями, не принципиальными для целей данного сообщения.

Среди них различные формы представления графов и кусочно-непрерывных путей, графический и текстовый вывод интервальной раскраски выбранного пользователем графа, в частности преднормализованного. Пример такого вывода приведен на рис. 2.

Литература

1. *Гэрри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи: пер. с анг. – М.: Мир, 1982. – 416 с.

2. *Norbert Blum*. A Solution of the P versus NP Problem // Institut fűr Informatik, Universität Bonn / arXiv:1708.03486vl [cs.CC] 11 Aug. 2017.

3. *Разборов А.А*. Алгебраическая сложность. – М.: МЦНМО, 2016. – 32 с.

4. *Асратян А.С., Камалян Р.Р*. Интервальные раскраски ребер мультиграфа // Прикладная математика, вып. 5. – Ереван: Изд-во Ереванского ун-та, 1987. – С. 25–34.

5. *Севастьянов С.В.* Об интервальной раскрашиваемости ребер двудольного графа // Методы дискретного анализа. – 1990. – Т. 50. – С. 61–72.

6. *Giaro K*. Compact task scheduling on dedicated processors with no waiting period (in Polish) // PhD thesis, Technical University of Gdansk, IETI Faculty. – Gdansk, 1999.

7. *Petrosyan P.A., Khachatrian H.H*. Interval non-edge-colorable bipartite graphs and multigraphs // Journal of Graph Theory 76. – 2014. – Issue 3. – P. 200–216.

8. *Giaro K., Kubale M. and Malafiejski M*. On the deficiency of bipartite graphs // Discrete Appl. Math. – 1999. – 94. – P. 193–203.

9. *Магомедов А.М*. Элиминация перебора двудольных графов на 15 вершинах // ДЭМИ, Дагестанский научный центр РАН. – 2016. – № 5. – С. 20–24.

10. *Магомедов А.М*. Цепочечные структуры в задачах о расписаниях // Прикладная дискретная математика. – 2016. – № 3 (33). – С. 67–77.

Поступила в редакцию 28 августа 2017 г.

UDC 519.1+004.9

DOI: 10.21779/2542-0321-2017-32-3-74-84

Abridged enumeration of bipartite graphs of preassigned order

A.M. Magomedov, T.A. Magomedov

Dagestan State University; Russia, 367000, Makhachkala, M. Gadzhiev st., 43a; magomed-tagir1@yandex.ru

Greedy algorithm is designed to test the existence of the edge-interval-coloring of a connected bipartite graph and its construction if it exists. The algorithm performs backtracking consecutive edges of the piecewise-continuous path; the latter represents an ordered set of all edges in the graph, and the subgraph generated by any subset of its edges with the numbers 1, 2, ..., is coherent.

The algorithm allowing for the set S of all bipartite graphs of given order H to generate a set S_0 bipartite graphs of less significant power, containing a graph isomorphic to *G* for each graph *G* of *S* a. The use of the greedy algorithm to each element of S_0 involving computer resources has shown, that at small values of *H* all the graphs S_0 are interval-colourable.

The research has found, in particular, that all bipartite graphs $G = (X, Y, E)$ of 16 order where $|X| < 7$ have an interval coloring; this is enough to check the interval coloring bipartite graphs of order 16 with $|X| = 7$ and $|X| = 8$. The software developed by the authors on the basis of the algorithms also provides a number of additional features contributing to a graphic display of interval coloring in a perceptible form.

Keywords: *bipartite graph, interval coloring, program.*

Received 28 August, 2017