ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

УДК 681.142.1+621.316

ЦЕПОЧЕЧНЫЕ СТРУКТУРЫ В ЗАДАЧАХ О РАСПИСАНИЯХ[1]

А. М. Магомедов

Дагестанский государственный университет, г. Махачкала, Россия

Разработаны алгоритмы построения однопроцессорного расписания с условиями частичного предшествования и мультипроцессорного расписания без простоев и условий частичного предшествования.

Ключевые слова: *расписание, граф, алгоритм, раскраска, сложность.*

DOI 10.17223/20710410/33/5

CHAIN STRUCTURES IN SCHEDULES TASKS

A. M. Magomedov

Dagestan State University, Makhachkala, Russia

E-mail: magomedtagir1@yandex.ru

Some algorithms for constructing the following shedules are developed: 1) one-processor schedule with a partial precedence, 2) a multiprocessor schedule without idle time and conditions of partial precedence.

Keywords: *schedule, graph, algorithm, colors, complexity.*

Введение

В исследованиях задач составления расписаний учебных занятий [1, с. 403], их приведения к «безоконному» виду [2], а также в задачах о расписаниях обслуживания множества заданий в мультипроцессорной системе без простоя процессоров и прерываний обработки каждого задания [3] нередко используются методы теории графов [4 – 6].

Пусть $G = (V, E)$ — связный граф. Цепью в G называется [7] такая последовательность вершин a_i и попарно различных ребер b_i (вершины могут повторяться)

$$P_0 = (a_0, b_0, a_1, b_1, \ldots, b_{k-1}, a_k), \tag{1}$$

что каждые два соседних ребра b_{i-1} и b_i имеют общую вершину a_i; если все вершины в (1) попарно различны, то цепь называется простой. Если концевые вершины a_0 и a_n (простой) цепи (1) совпадают, то (простая) цепь (1) называется (простым) циклом. В настоящей работе обсуждается использование цепочечных структур (цепей и конкатенаций нескольких цепей): в п. 1 — для построения однопроцессорного расписания

[1]Работа выполнена при финансовой поддержке отдела математики и информатики ДНЦ РАН и проекта № 2588 в рамках базовой части госзадания Минобрнауки России.

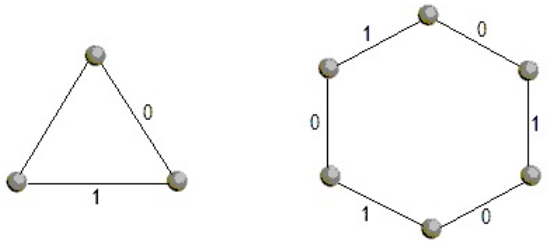
с условиями предшествования, в п. 2 — для проверки существования интервальной рёберной раскраски двудольных графов (графической интерпретации многопроцессорного расписания без простоев и условий предшествования).

Интервальной рёберной раскраской графа $G = (V, E)$ называется такое отображение множества ребер E в множество целых чисел («цветов»), при котором для каждого v из множества вершин V цвета рёбер, инцидентных v, различны и образуют целочисленный интервал (далее будем писать кратко «интервальная раскраска»).

Пример 1.

1) Простой цикл нечётной длины не имеет интервальной раскраски. Смежные рёбра на рис. 1, *а* раскрашены в цвета a и $a + 1$ ($a = 0$). Тогда цвет третьего ребра при интервальной раскраске должен быть отличным от a и $a + 1$ целым числом, соседним как с a, так и с $a + 1$ (а такого числа не существует).

2) Простой цикл чётной длины интервально раскрашиваем (достаточно взглянуть на раскраску рёбер графа на рис. 1,*б*).



а *б*

Рис. 1. Циклы нечётной длины не допускают интервальной раскраски, чётной длины — допускают

Задача об интервальной раскраске заданного графа $G = (V, E)$ NP-полна [2]. Свойство NP-полноты сохраняется и в случае двудольного графа [8]. Сложность задачи об интервальной раскрашиваемости может различаться для разных классов двудольных графов. Ниже показано, что любой регулярный двудольный граф допускает интервальную раскраску (результат Р. Камаляна [9]), при этом задача об интервальной раскрашиваемости обыкновенного регулярного графа NP-полна [2].

Общеизвестно, что текстуально близкие задачи теории графов нередко принадлежат разным классам сложности. Из примеров, приведённых в [10, с. 104], уместно указать на следующие две задачи: «Кратчайший путь между двумя вершинами» и «Самый длинный путь между двумя вершинами» в заданном графе $G = (V, E)$; первая, как известно, разрешима за время $O(|V|^2)$, вторая же NP-полна — к этой задаче сводится задача «Гамильтонов путь между двумя вершинами» [10, с. 269].

Замечание 1. Для задачи «Кратчайший путь между двумя вершинами» неизвестен алгоритм с оценкой $O(|E|)$ [11, с. 236].

Сложность задачи о гамильтоновом маршруте (цикле или пути) принципиально различна для графов разных классов. Рассмотренные далее задачи тесно связаны с вычислением гамильтоновых маршрутов в графах простой структуры: в связном регулярном графе чётной степени и ациклическом ориентированном графе.

1. Расписание выполнения заданий в однопроцессорной системе с условиями частичного предшествования

1.1. Достаточные условия интервальной раскрашиваемости

Из теоремы Эйлера [7, с. 192] следует, что связный регулярный граф G чётной степени обладает эйлеровым циклом (под степенью регулярного графа будем понимать степень вершины графа); теорема Ю. Петерсена [12] утверждает, что связный регулярный граф чётной степени $2n$ допускает разложение на n 2-факторов (каждая связная компонента 2-фактора является гамильтоновым циклом). Если каждый из этих 2-факторов является набором циклов чётной длины в графе G, то, помечая последовательные рёбра каждого цикла i-го набора метками

$$2i,\ 2i+1,\ 2i,\ 2i+1,\ \ldots\ (i=0,1,\ldots,n-1),$$

можно получить интервальную раскраску. Из перечисленных свойств графа сохраним лишь требование чётности длины любого цикла. Но последнее, согласно теореме Кенига, означает двудольность графа, а как уже отмечалось ранее, не всякий двудольный граф интервально раскрашиваем. Более того, проверка интервальной раскрашиваемости является NP-полной задачей, следовательно, не существует эффективно проверяемых необходимых и достаточных условий (в предположении истинности гипотезы P≠NP). Поэтому для обеспечения свойства интервальной раскрашиваемости следует добавить некоторое дополнительное условие (разумеется, менее слабое, чем регулярность с чётной степенью). Потребуем выполнение свойства регулярности без условия чётности степени графа.

В случае нечётной степени Δ достаточно найти в заданном регулярном двудольном графе $G=(X,Y,E)$ полное паросочетание X с Y (существование такого паросочетания следует из следствия 8.16.1 [1, с. 165]), раскрасить его рёбра цветом $\Delta-1$ и удалить его, затем воспользоваться теоремой Ю. Петерсена для интервальной раскраски оставшегося графа $(\Delta-1)$ цветами: $0,1,\ldots,\Delta-2$. Таким образом, мы пришли к известному результату [9] об интервальной раскрашиваемости любого регулярного двудольного графа.

1.2. Перечисление допустимых расписаний с условиями частичного предшествования

Пусть однопроцессорной системе выделены для обработки объекты $0,1,\ldots,n-1$ и заданы условия частичного предшествования: для некоторых пар объектов указано, что один из объектов пары должен быть обработан раньше другого. Требуется построить допустимое расписание обработки, другими словами, упорядочить все объекты в соответствии с условиями частичного предшествования. Рассмотрим орграф $G=(V,E)$, вершины $v_0,v_1,\ldots,v_{n-1}$ которого соответствуют заданным объектам, дуги — условиям частичного предшествования: из вершины v_i в вершину v_j дуга проведена тогда и только тогда, когда объект с номером j предшествует объекту с номером i. Если в орграфе G имеется цикл, то искомое расписание, очевидно, не существует (ориентированный граф имеет цикл в том и только в том случае, когда алгоритм поиска в глубину находит обратную дугу [13]). Если G — ациклический орграф, то для построения допустимого расписания достаточно применить алгоритм топологической сортировки [1, с. 96]. Предлагается близкий по идее алгоритм 1, на основе которого удаётся не только построить допустимое расписание, но и выполнить перечисление таких расписаний. В алгоритме используются следующие обозначения: $m(v)$ — целочисленные метки вершин $v\in V$; $L[i]$ — список всех вершин с i исходящими дугами,

$i = 0, 1, \ldots, n-1$ (сначала все списки пусты). Нетрудно видеть, что сложность алгоритма 1 не превышает $O(n^2)$.

Алгоритм 1. Иерархическая сортировка вершин графа

Вход: связный ациклический орграф $G = (V, E)$ на n вершинах

Выход: такая упорядоченная последовательность T вершин множества V, что для каждой дуги $(v_i, v_j) \in E$ вершина v_j предшествует вершине v_i в последовательности T

1: **Для всех** $v \in V$ выполнить:
   $m(v) := 0$; $i := d^-(v)$; добавить v в $L[i]$.
2: **Пока** $E \neq \varnothing$:
3: выбрать произвольную вершину v из $L[0]$ и дугу (u, v).
4: **Если** $m(v) + 1 > m(u)$, **то**
   $m(u) := m(v) + 1$.
5: $i := d^-(u)$.
6: Удалить дугу (u, v) из E.
7: Переместить вершину u из $L[i]$ в $L[i-1]$.
8: **Если** $d^+(v) = 0$, **то**
   удалить вершину v из графа и из $L[0]$.
9: Восстановить исходный граф $G = (V, E)$.
10: $m_0 := \max\limits_{v \in V} m(v)$, $T := \varnothing$.
11: **Для** $i = 0, \ldots, m_0$ выполнить:
12: **Для всех** $v \in V$ выполнить:
13: **Если** $m(v) = i$, **то**
   добавить v к концу последовательности T.

Утверждение 1. Упорядочение объектов, индуцированное последовательностью T, является допустимым расписанием.

Доказательство. Отсутствие вершины с нулевой полустепенью исхода в орграфе G (или в подграфе, полученном в каждой итерации цикла) означало бы присутствие в орграфе цикла, что противоречит условию применения иерархической сортировки.

Для каждой дуги (v_i, v_j) метка, присваиваемая вершине v_i алгоритмом 1, превышает метку, присваиваемую вершине v_j: $m(v_i) \geqslant m(v_j) + 1$. Следовательно, если известно, что j-й объект предшествует i-му объекту, то в списке, индуцированном последовательностью T, объект с номером j располагается раньше объекта с номером i, т. е. получено допустимое расписание. ■

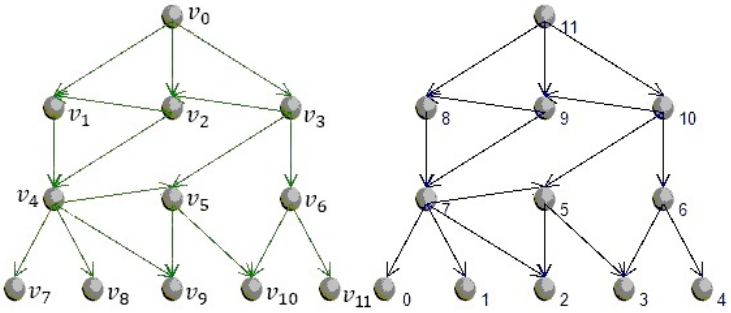
В следующем следствии через V_i обозначено множество всех вершин v с меткой $m(v) = i$, $i = 0, \ldots, m_0$.

Следствие 1. Количество допустимых расписаний равно $|V_0|! \ldots |V_{m_0}|!$

Замечание 2. Задача иерархической сортировки текстуально близка к задаче поиска ориентированного гамильтонова пути в каждой компоненте ориентированного графа. Уже отмечалось, что в общем случае задача о гамильтоновом пути NP-полна, но для ациклических ориентированных графов задача разрешима за полиномиальное время [14].

Пример 2. Пусть $n = 12$, для каждого объекта в скобках указаны предшествующие ему объекты: $0(1, 2, 3)$, $1(4)$, $2(1, 4)$, $3(2, 5, 6)$, $4(5, 7, 8, 9)$, $5(9, 10)$, $6(10, 11)$. В част-

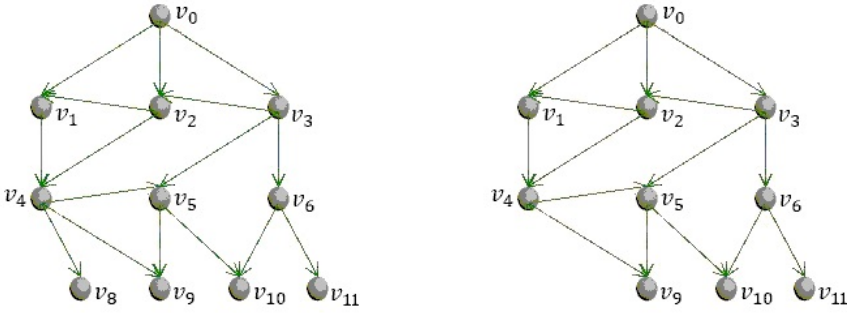
ности, отсюда видно, что у объектов $7, \ldots, 10$ нет предшественников. На рис. 2, *а* приведён орграф G.



а *б*

Рис. 2. Орграф G (*а*); результат алгоритма 1 — вершины G упорядочены по неубыванию меток (*б*)

На рис. 3, *а* отображены результаты выполнения первой итерации цикла 2, где выбраны вершина $v_7 \in L[0]$ и дуга (v_4, v_7); метке $m(v_4)$ присвоено значение 1, переменной i — значение 2; дуга (v_4, v_7) удалена; вершина v_4 перемещена из $L[2]$ в $L[1]$; вершина v_7 удалена из графа и из $L[0]$. Во второй итерации цикла 2 (рис. 3, *б*) для вершины v_8 выполнены аналогичные действия. Окончательный результат приведён на рис. 2, *б*.



а *б*

Рис. 3. Первая (*а*) и вторая (*б*) итерации цикла 2 алгоритма 1

2. Жадный алгоритм интервальной раскраски

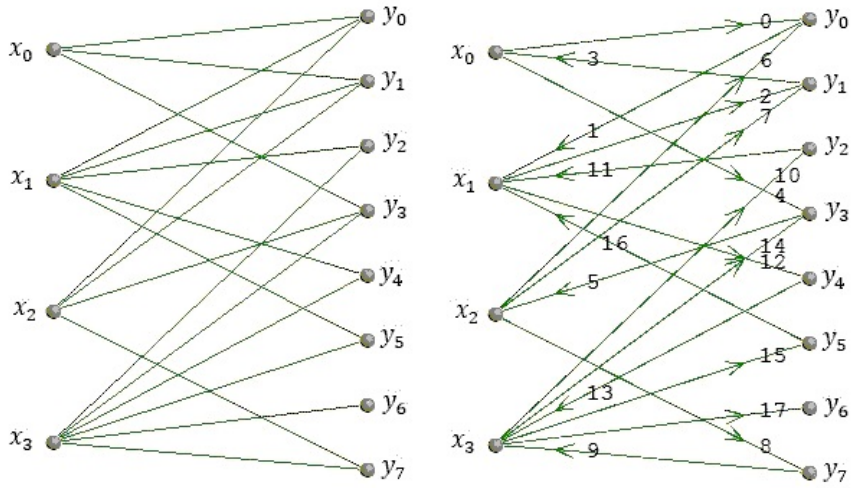
2.1. Кусочно-непрерывный путь

Цепь (1) в графе G называется *непродолжимой*, если P_0 содержит все рёбра графа G, инцидентные вершине a_k. Обозначим через G' копию графа G и определим кусочно-непрерывный путь в G следующим образом.

Пока имеется ребро графа G', не входящее в P_0, выполним действия: для графа, полученного удалением из G' всех рёбер, включённых в P_0, сохраним обозначение G' и выберем a_k — самую правую из вершин P_0, имеющих в графе G' инцидентные рёбра; построим в графе G' непродолжимую цепь P_1, начинающуюся с вершины a_k, затем выполним конкатенацию P_0 и цепи P_1 и обозначим результат конкатенации через P_0. Последовательность рёбер графа G, перечисленных в том порядке, в каком они следуют в последнем P_0, будем называть кусочно-непрерывным путём в графе G и обозначать через P. Очевидно, кусочно-непрерывных путей в графе может быть несколько.

Поясним на примере графа G, приведённого на рис. 4, *а*. Выберем в G непродолжимую цепь $P_0 = (a_0, b_0, a_4, b_1, a_1, b_2, a_5, b_3, a_0, b_4, a_7, b_5, a_2, b_6, a_4)$. Затем удалим из копии G' графа G рёбра $b_0, \ldots, b_6$, включенные в P_0. Самой правой вершиной P_0, обладающей в оставшемся графе G' инцидентными рёбрами, будет вершина a_2 (инцидентные рёбра b_7 и b_8). Построим в G' непродолжимую цепь $P_1 = (a_2, b_7, a_5)$; для результата конкатенации P_0 и P_1 сохраним обозначение P_0: $P_0 = (a_0, b_0, a_4, b_1, a_1, b_2, a_5, b_3, a_0, b_4, a_7, b_5, a_2, b_6, a_4; a_2, b_7, a_5)$ — и удалим из графа G' ребро b_7. Самой правой вершиной P_0, обладающей инцидентными ребрами, снова является вершина a_2, $P_1 = (a_2, b_8, a_{11}, b_9, a_3, b_{10}, a_6, b_{11}, a_1, b_{12}, a_8, b_{13}, a_3, b_{14}, a_7)$ — непродолжимая цепь графа G', начинающаяся в вершине a_2. Результат конкатенации P_0 и P_1 обозначим через P_0: $P_0 = (a_0, b_0, a_4, b_1, a_1, b_2, a_5, b_3, a_0, b_4, a_7, b_5, a_2, b_6, a_4; a_2, b_7, a_5; a_2, b_8, a_{11}, b_9, a_3, b_{10}, a_6, b_{11}, a_1, b_{12}, a_8, b_{13}, a_3, b_{14}, a_7)$. Непродолжимая цепь, начинающаяся в вершине a_3: $P_1 = (a_3, b_{15}, a_9, b_{16}, a_1)$. Выполним конкатенацию с P_0, после чего в качестве заключительного шага выберем непродолжимую цепь $P_1 = (a_3, b_{17}, a_{10})$ и выполним её конкатенацию с P_0. В результате получим кусочно-непрерывный путь в графе G: $P = (b_0, b_1, \ldots, b_{17})$.

Замечание 3. Естественным представляется вместо «просто» непродолжимой цепи рассматривать самую длинную цепь в графе. Однако, как отмечено выше, её построение затруднено NP-полнотой соответствующей задачи распознавания.



а *б*

Рис. 4. Двудольный граф G (*а*); кусочно-непрерывный путь P (*б*). Номера рёбер соответствуют порядку их следования в P

2.2. Жадный алгоритм интервальной раскраски

Пусть в двудольном графе $G = (V, E)$, $V = (X, Y)$, $|E| = m$, $|V| = n$, построен кусочно-непрерывный путь P из последовательных рёбер $e_i = (v_i, w_i)$, $i = 0 \ldots, m-1$. В прикладных задачах нередко требуется построить интервальную раскраску, удовлетворяющую дополнительному условию — все цвета принадлежат некоторому «допустимому» диапазону. Допустимый для раскрашивания рёбер диапазон цветов обозначим через $[-t..t]$, где t — заданное натуральное число. Цвет ребра e_i будем обозначать $C[i]$, $i = 0, \ldots, m-1$. Цвета всех раскрашенных рёбер, инцидентных вершине v_i, будем называть цветами, *представленными в вершине* v_i; для их хранения используем список $L[v_i]$, $i = 0, \ldots, n-1$.

Предлагается раскрасить последовательные рёбра кусочно-непрерывного пути P цветами из допустимого диапазона так, чтобы в каждый момент времени для каждой вершины v_i цвета, представленные в v_i, составляли (целочисленный) интервал. Алгоритм раскрашивания, обладающий таким свойством, будем называть *жадным алгоритмом*.

Для заданного непустого целочисленного интервала цветов $[a..b] = \{a, a+1, \ldots, b\}$ цвета из множества $\{a-1, b+1\} \cap [-t..t]$ будем называть *приграничными цветами*; для пустого интервала приграничным цветом считается каждый цвет из допустимого диапазона. Для ребра $e_i = (v_i, w_i)$ через $M_1 = [min_1..max_1]$ и $M_2 = [min_2..max_2]$ будем обозначать интервалы цветов, представленных в вершинах v_i и w_i, M_1' и M_2' — множества цветов, приграничных для M_1 и M_2 соответственно. Из соображений краткости мы не будем снабжать эти обозначения индексом i, поскольку всюду в дальнейшем индекс i однозначно определяется из контекста.

Утверждение 2. Пусть ребро $e_i = (v_i, w_i)$ не раскрашено. Для раскраски ребра e_i в соответствии с жадным алгоритмом может быть выбран любой цвет из пересечения $M_1' \cap M_2'$, и никакой другой.

Доказательство. Справедливость утверждения следует из того, что жадный алгоритм сохраняет свойство интервальности как для цветов, представленных в вершине v_i, так и для цветов, представленных в вершине w_i. ■

Пусть заданный допустимый интервал $[-t..t]$ достаточно большой, списки цветов $L[v_i]$ и $L[w_i]$, представленных в концевых вершинах ребра $e_i = (v_i, w_i)$, образуют непустые интервалы M_1 и M_2 соответственно. Цвет $r \in M_1' \cap M_2'$, допустимый для раскрашивания ребра e_i, может: 1) отсутствовать (например, при $M_1 = [1..2]$, $M_2 = [5..10]$); 2) определяться однозначно (например, $r = 3$ при $M_1 = [1..2]$, $M_2 = [4..5]$); 3) принимать одно из двух значений (например, при $M_1 = M_2 = [3..4]$ можно выбрать $r = 2$ или $r = 5$).

Для формулировки алгоритма 2 нам понадобится способ перебора «состояний» поиска допустимого цвета для рассматриваемого ребра e_i: $0, 1, \ldots$; текущее состояние ребра $e_i = (v_i, w_i)$ будем обозначать q_i; логическое условие, выражающее взаимное расположение интервалов M_1 и M_2 (и случаи, когда один или оба интервала M_1 и M_2 пусты), — $p_i[q_i]$; цвет, выбираемый для присвоения ребру e_i в зависимости от $p_i[q_i]$, — $c_i[q_i]$.

При принятом подходе количество таких «состояний» равно 9. Соотнесём каждому $q_i = 0, 1, \ldots, 8$ соответствующие $p_i[q_i]$ и $c_i[q_i]$.

С п и с о к с о с т о я н и й

$p_i[0] := (L[v_i] = \varnothing) \,\&\, (L[w_i] = \varnothing);$	$c_i[0] := 0;$
$p_i[1] := (L[v_i] = \varnothing) \,\&\, (L[w_i] \neq \varnothing) \,\&\, (min_2 > -t);$	$c_i[1] := min_2 - 1;$
$p_i[2] := (L[v_i] = \varnothing) \,\&\, (L[w_i] \neq \varnothing) \,\&\, (max_2 < t);$	$c_i[2] := max_2 + 1;$
$p_i[3] := (L[v_i] \neq \varnothing) \,\&\, (L[w_i] = \varnothing) \,\&\, (min_1 > -t);$	$c_i[3] := min_1 - 1;$
$p_i[4] := (L[v_i] \neq \varnothing) \,\&\, (L[w_i] = \varnothing) \,\&\, (max_1 < t);$	$c_i[4] := max_1 + 1;$
$p_i[5] := (L[v_i] \neq \varnothing) \,\&\, (L[w_i] \neq \varnothing) \,\&\, (min_1 > -t) \,\&\, (min_1 = min_2);$	$c_i[5] := min_1 - 1;$
$p_i[6] := (L[v_i] \neq \varnothing) \,\&\, (L[w_i] \neq \varnothing) \,\&\, (max_1 = max_2) \,\&\, (max_1 < t);$	$c_i[6] := max_1 + 1;$
$p_i[7] := (L[v_i] \neq \varnothing) \,\&\, (L[w_i] \neq \varnothing) \,\&\, (max_1 + 2 = min_2);$	$c_i[7] := max_1 + 1;$
$p_i[8] := (L[v_i] \neq \varnothing) \,\&\, (L[w_i] \neq \varnothing) \,\&\, (max_2 + 2 = min_1);$	$c_i[8] := max_2 + 1;$

Например, для ребра $e_i = (v_i, w_i)$ при $q_i = 8$ условие $p_i[q_i]$ означает, что списки цветов, представленных в вершинах v_i и w_i, не пусты, интервал M_2 располагается на числовой оси левее интервала M_1, при этом наибольшее значение (max_2) из интервала M_2 на 2 меньше наименьшего значения (min_1) из интервала M_1. Поэтому множество $M_1' \cap M_2'$ цветов, приграничных как для M_1, так и для M_2, содержит единственный элемент, равный $max_2 + 1$. Понятно, что он и выбирается в качестве цвета, присваиваемого ребру e_i: $c_i[8] := max_2 + 1$.

В основе алгоритма 2 лежит идея перебора с возвратами. Каждая итерация цикла по раскрашиванию рёбер $e_i (i = 0, \ldots n-1)$ состоит из двух шагов — прямого и возвратного. Прямой шаг включает действия по раскрашиванию ребра e_i кусочно-непрерывного пути P с сохранением цветов предшествующих рёбер. Возвратный шаг выполняется, когда прямой шаг по раскрашиванию очередного ребра e_i не увенчался успехом, и включает действия по изменению цветов некоторых рёбер, предшествующих e_i, с целью подготовить условия для успешного раскрашивания ребра e_i. Нумерация рёбер соответствует порядку их следования в кусочно-непрерывном пути P.

Алгоритм 2 открывает перспективы для решения известной проблемы: верно ли, что любой двудольный граф с числом вершин $n = 15, 16, 17, 18$ допускает интервальную раскраску? Известно [15], что при $n < 15$ ответ положителен, а при $n > 18$ — отрицателен.

На рис. 5 представлена интервальная раскраска, сгенерированная алгоритмом 2 для двудольного графа рис. 4, *а*. Время счёта — приблизительно 2 мс на компьютере с характеристиками 2,5 ГГц, 8 Гб. Отметим, что первый возвратный шаг выполнен для ребра $e_7 = (x_2, y_1)$, а общее число возвратных шагов — 24.

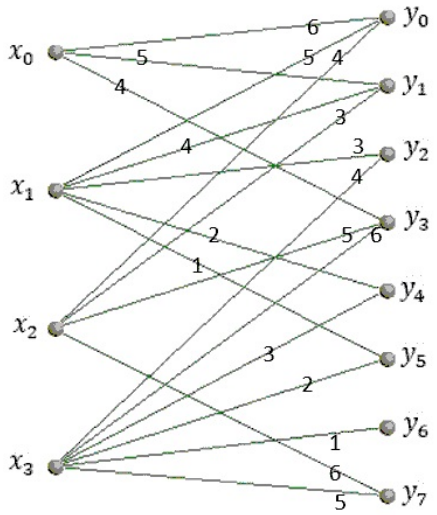


Рис. 5. Интервальная раскраска графа G

Алгоритм 2. Жадный алгоритм интервальной раскраски

Вход: $G = (V, E)$ — связный двудольный граф, $|V| = n$, $|E| = m$; кусочно-непрерывный путь P в графе G

Выход: массив $C[0..m-1]$ для цветов рёбер и логическая переменная $Success$: $Success =$ `true`, если интервальная раскраска получена, и $Success =$ `false` в противном случае

1. *Инициализация*

1: **Для** $k = 0, \ldots, m-1$ выполнить:
   $q_k := -1$
2: **Для** $k = 0, \ldots, n-1$ выполнить:
3: $L[v_k] := \varnothing$
4: $i := 0$

2. *Прямой шаг*

5: **Если** $q_i > 8$, **то**
   переход к пункту *Возвратный шаг.*
6: **Если** $L[v_i] \neq \varnothing$, **то**
7: переменным min_1 и max_1 присвоить соответственно минимальное и максимальное значения списка $L[v_i]$.
8: **Если** $L[w_i] \neq \varnothing$, **то**
9: переменным min_2 и max_2 присвоить соответственно минимальное и максимальное значения списка $L[w_i]$.
10: **Для** $k = q_i + 1, \ldots, 8$ выполнить:
11: **Если** $p_i[k] =$ `true`, **то**
      переход к строке 13
12: переход к пункту *Возвратный шаг.*
13: $C[i] := c_i[k]$;
    добавить $C[i]$ в списки $L[v_i]$ и $L[w_i]$
14: **Если** $i < m-1$, **то**
      $i := i + 1$; переход к пункту *Прямой шаг*
15: **иначе**
16: $Success :=$ `true`; завершить алгоритм.

3. *Возвратный шаг*

17: $q_i := -1$; $i := i - 1$;
18: **Если** $i = -1$, **то**
      $Success :=$ `false`; завершить алгоритм.
19: Удалить $C[i]$ из $L[v_i]$ и $L[w_i]$; $q_i := q_i + 1$; переход к пункту *Прямой шаг.*

Заключение

Алгоритмы 1 и 2 воплощены в компьютерные программы, получены свидетельства о государственной регистрации в Реестре программ для ЭВМ (2016 г.). Из п. 1.2 видно, что алгоритм иерархической сортировки вершин имеет фактически линейную вычислительную сложность относительно длины входа. Вычислительные трудности жадного алгоритма интервальной раскраски коренятся в NP-полноте задачи распознавания об интервальной рёберной раскрашиваемости двудольного графа. В худшем случае время работы данного алгоритма оценивается экспоненциально, однако ком-

пьютерные эксперименты подтверждают его эффективность для практической работы с графами, в которых число вершин не превышает 20.

ЛИТЕРАТУРА

1. *Свами М., Тхуласираман К.* Графы, сети и алгоритмы. М.: Мир, 1984. 455 с.
2. *Асратян А. С., Камалян Р. Р.* Интервальные раскраски ребер мультиграфа // Прикладная математика. Вып. 5. Ереван: Изд-во Ереванского ун-та, 1987. С. 25–34.
3. *Танаев В. С., Сотсков Ю. Н., Струсевич В. А.* Теория расписаний. Многостадийные системы. М.: Наука, 1989. 328 с.
4. *Магомедов А. М.* К вопросу об интервальной Δ-раскраске двудольных графов // Автоматика и телемеханика. 2015. № 1. С. 101–109.
5. *Магомедов А. М., Магомедов Т. А.* Реберно-вершинные инцидентные паросочетания в задачах расписаний // Прикладная дискретная математика. 2015. № 1(27). С. 92–95.
6. *Магомедов А. М., Магомедов Т. А.* Последовательное разбиение ребер двудольного графа на паросочетания // Дискретная математика. 2016. Т. 28. Вып. 1. С. 78–86.
7. *Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И.* Последовательное разбиение ребер двудольного графа на паросочетания. М.: Книжный дом «Либроком», 2009. 392 с.
8. *Севастьянов С. В.* Об интервальной раскрашиваемости ребер двудольного графа // Методы дискретного анализа. 1990. Т. 50. С. 61–72.
9. *Камалян Р. Р.* Интервальные раскраски полных двудольных графов и деревьев / Препринт ВЦ АН АрмССР. Ереван, 1989. 11 с.
10. *Гэри М, Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.
11. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.
12. *Petersen J.* Die Theorie der regularen Graphs // Acta Math. 1891. No. 15. P. 193–220.
13. *Tucker A.* Ch. 2. Covering Circuits and Graph Colorings // Appl. Combinat. 5th Ed. Hoboken: John Wiley & Sons, 2006. P. 49.
14. *Lawler E. L.* Combinatorial Optimization: Networks and Matroids. N. Y.: Holt, Rinehart and Winston, 1976.
15. *Giaro K.* Compact task scheduling on dedicated processors with no waiting period. PhD thesis. Technical University of Gdansk, IETI Faculty, Gdansk, 1999. (in Polish)

REFERENCES

1. *Swamy M. N. S. and Thulasiraman K.* Graphs, Networks, and Algorithms. Wiley-InterScience, 1981.
2. *Asratyan A. S. and Kamalyan R. R.* Interval'nye raskraski reber mul'tigrafa [Interval coloring of multigraph edges]. Prikladnaya Matematika, iss. 5, Yerevan SU Publ., 1987, pp. 25–34. (in Russian)
3. *Tanaev V. S., Sotskov Yu. N., and Strusevich V. A.* Teoriya raspisaniy. Mnogostadiynye sistemy [Shedules Theory. The Multi-Stage Systems]. Moscow, Nauka Publ., 1989. (in Russian)
4. *Magomedov A. M.* K voprosu ob interval'noy Δ-raskraske dvudol'nykh grafov [On interval Δ-coloring of bipartite graphs]. Avtomatika i Telemekhanika, 2015, no. 1, pp. 101–109. (in Russian)

5. *Magomedov A. M. and Magomedov T. A.* Reberno-vershinnye intsidentnye parosochetaniya v zadachakh raspisaniy [Edge-vertex incident matchings in scheduling]. Prikladnaya Diskretnaya Matematika, 2015, no. 1(27), pp. 92–95. (in Russian)
6. *Magomedov A. M. and Magomedov T. A.* Posledovatel'noe razbienie reber dvudol'nogo grafa na parosochetaniya [Sequential partitioning of bipartite graph edges on matching]. Diskr. Mat., 2016, vol. 28, iss. 1, pp. 78–86. (in Russian)
7. *Emelichev V. A., Mel'nikov O. I., Sarvanov V. I., and Tyshkevich R. I.* Posledovatel'noe razbienie reber dvudol'nogo grafa na parosochetaniya [Sequential Partitioning of Bipartite Graph Edges on matching]. Moscow, Librokom Publ., 2009. (in Russian)
8. *Sevast'yanov S. V.* Ob interval'noy raskrashivaemosti reber dvudol'nogo grafa [On the interval coloring of edges of a bipartite graph]. Metody Diskretnogo Analiza, 1990, vol. 50, pp. 61–72. (in Russian)
9. *Kamalyan R .R.* Interval'nye raskraski polnykh dvudol'nykh grafov i derev'ev [Interval Coloring of Complete Bipartite Graphs and Trees]. Preprint VTs AN ArmSSR, Yerevan, 1989. (in Russian)
10. *Garey R. and Johnson D.* Computers and Intractability. N. Y., USA, W. H. Freeman & Co, 1979.
11. *Aho A., Hopcroft J., and Ullman J.* The Design and Analysis of Computer Algorithms. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1974.
12. *Petersen J.* Die Theorie der regularen Graphs. Acta Math., 1891, no. 15, pp. 193–220.
13. *Tucker A.* Ch. 2. Covering Circuits and Graph Colorings. Appl. Combinat., 5th Ed., Hoboken, John Wiley & Sons, 2006, p. 49.
14. *Lawler E. L.* Combinatorial Optimization: Networks and Matroids. N. Y., Holt, Rinehart and Winston, 1976.
15. *Giaro K.* Compact task scheduling on dedicated processors with no waiting period. PhD thesis. Technical University of Gdansk, IETI Faculty, Gdansk, 1999. (in Polish)