УДК 519.174.7, 004.75(575.2)(04)

АНАЛИЗ ЛОКАЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ АЛГОРИТМОВ ДЛЯ РАСКРАСКИ ГРАФОВ, ИСПОЛЬЗУЮЩИХ СТРАТЕГИЮ ЖАДНОГО АЛГОРИТМА

В.А. Евстигнеев, Турсунбай кызы Ы.

Рассматривается один из способов улучшения выполнения локального алгоритма – представление стратегии раскраски в алгоритм, который является эффективным в нераспределенных алгоритмах. Показано, что применение некоторых эвристик последовательного алгоритма раскраски, таких, как наибольшие-первые (НП), наименьшие-последние (ПН) и наибольшие-первые насыщенности (НПН), для некоторых классов графов и для частных случаев вершинной раскраски в локальных алгоритмах дают оптимальную или почти оптимальную раскраску.

Ключевые слова: раскраска графов; распределенный алгоритм; локальный алгоритм; жадный алгоритм; w -совершенные графы; *T*-раскраска; суммирующая раскраска.

1. Введение

1.1. Постановка задачи. Пусть G – конечный неориентированный граф, обозначим через $n = |V|$, $m = |E|$ и для каждой вершины v определим открытую окрестность $N(v) = \{u : (u, v) \in E\}$ и степень $\deg_G v = |N(v)|$.

Раскраской вершин графа G называется такое приписывание цветов его вершинам, что никакие две смежные вершины не получают одинаковые цвета. *Хроматическое число* $\chi(G)$ графа G определяется как наименьшее количество цветов, необходимых для раскраски G, а раскраска графа $\chi(G)$ цветами называется *оптимальной раскраской* графа G. Задача раскраски графа заключается в нахождении оптимальной раскраски.

Последовательные алгоритмы, обсуждаемые в данной работе, имеют практическое значение из-за числа классов графов, для которых они всегда дают оптимальные или почти оптимальные результаты. Поэтому естественно сформулировать задачу построения локальных алгоритмов с аналогичными свойствами.

1.2. Локальные алгоритмы. Теория локальных алгоритмов ведет свое начало от работ Ю.И. Журавлева по локальным алгоритмам решения задач дискретной оптимизации, в том числе на графах [1, 2]. Дальнейшее развитие этого подхода привело к понятию последовательного, параллельного, а также вычислительного локального алгоритма [3].

Полное определение локального алгоритма можно найти в работах [1, 3]. Здесь мы приведем более строгое определение локального алгоритма. Вначале сформулируем принцип локальности.

Принцип локальности. Вычисление предикатов и функций на элементе (вершине или ребре) графа производится на основе изучения строения окрестности элемента с учетом вы-

численных на предыдущих шагах значений данной функции или предиката, а также значений других вспомогательных функций и предикатов на элементах этой окрестности; при этом вычисленные для элемента x значения хранятся в элементе y, становясь доступными для вычисления информации в элементе y тогда и только тогда, когда элемент x войдет в окрестность данного фиксированного порядка элемента y.

Локальным алгоритмом в общем смысле называется алгоритм, удовлетворяющий принципу локальности.

Локальный алгоритм называется *параллельным локальным алгоритмом*, если один шаг его состоит в одновременном и независимом перевычислении меток на всех вершинах графа на основании информации, полученной на предыдущем шаге [3].

1.3. Модель вычислений. Перейдем к описанию реализации локальных алгоритмов, а именно рассмотрим реализацию локальных алгоритмов на сети процессоров, моделирующих структуру исследуемой граф-модели. Вычислительная система, моделирующая топологию данного графа, есть множество однотипных процессоров, каждый из которых соответствует вершине графа. Два процессора связаны каналом связи, если соответствующие вершины в графе смежны.

Исследование данных систем привело к появлению так называемых эхо- или распределенных алгоритмов решения некоторых теоретико-графовых задач. В последние годы все большее внимание уделяется распределенным системам и распределенной обработке информации. В работах [4, 5] доказано, что распределенный алгоритм есть локальный вычислительный алгоритм порядка I, однако понятие локального алгоритма шире и включает в себя возможности работы с полной окрестностью первого порядка, а также с окрестностями более высокого порядка, чего нет в распределенных алгоритмах.

1.4. Алгоритм жадной раскраски в контексте локальных алгоритмов. Для упорядоченного множества вершин $v_1,\ldots,v_n$ графа G *последовательной раскраской*, отвечающей этому порядку, называется раскраска, использующая каждый из цветов $c_1,\ldots,c_k$ и определяемая следующей стратегией жадного алгоритма:

а) вершине v_1 приписан цвет c_1;

б) если подграф $H_{\langle v_1,\ldots,v_{i-1}\rangle}$, порожденный вершинами $v_1,\ldots,v_{i-1}$, k' – раскрашен, $k' \le i-1$, то вершина v_i получает цвет c_m, где $m \le k'+1$, т.е. цвет с наименьшим номером, не встречающийся на смежных с вершиной v_i вершинах.

Рассмотрим различные стратегии при организации последовательных раскрасок.

- *НП-алгоритмом* называется алгоритм, основанный на НП-упорядочении (“наибольшие – первыми”) вершин в соответствии с убыванием их степеней и нахождения по нему раскраски графа G.
- *ПН-алгоритмом* называется алгоритм, основанный на ПН-упорядочении (“последними – наименьшие”) вершин и строится следующим образом:

а) для $n = |V|$ в качестве v_n выбирается вершина минимальной степени в графе G;

б) для $i = n-1, n-2,\ldots,2,1$ в качестве v_i выбирается вершина минимальной степени в подграфе $H_{\langle V\setminus\{v_n,\ldots,v_{i+1}\}\rangle}$; и нахождения по нему раскраски графа G.

- Степенью насыщения вершины v в частично раскрашенном графе называется число различных цветов на смежных с v вершинах. *НПН (DSATUR)-алгоритмом* называется алгоритм, в котором последовательно окрашиваются вершины с наибольшей степенью насыщения (в начале алгоритма цвета получают вершины, имеющие наибольшую степень).

2. Основные результаты

Заметим, что не всегда легко добиться и скорости, и эффективности алгоритма. В работе [6] представлен распределенный алгоритм для раскраски графа в $(\Delta+1)$ цветов, где Δ – наибольшая степень вершины в графе. Время работы алгоритма O(log n). Будем называть этот алгоритм тривиальным. Данный алгоритм достаточно простой и быстрый, но не оптимальный. Действительно, количество цветов, используемых алгоритмом, близко к Δ, даже если граф двудольный. Неудивительно, что тривиальный алгоритм не имеет механизма экономии цветов. Дальнейшее усовершенствование тривиального алгоритма предложено в работе [7].

3. Локальный ПН-алгоритм для раскраски w-совершенных графов

Одной из важных характеристик, связанных с хроматическим числом, является число Секереша–Вилфа $w(G) = \max\limits_{G'\subseteq G} \mathrm{d}(G') + 1$, где $\mathrm{d}(G') = \min\limits_{x\in V(G')} d_{G'}(x)$ – минимальная степень

графа G', а $d_{G'}(x)$ – степень вершины x в G'. $w(G)$ в качестве верхней оценки хроматического числа впервые рассмотрена в работе Секереша и Вилфа [8].

Важность этой характеристики заключается в том, что, во-первых, $w(G)$ является довольно нетривиальной верхней оценкой для $\chi(G)$, т.е. класс графов, для которых $w(G)=\chi(G)$ довольно большой и содержит в себе много практически интересных классов, и, во-вторых, она легко вычисляемая.

Граф, обладающий таким свойством, что хроматическое число и вырожденность (число Секереша-Вилфа) равны не только у самого графа, но и у каждого его порожденного подграфа, называется w*-совершенным графом*.

Для вычисления $w(G)$ применяется ПН-упорядочение вершин [9, 10] графа G. По своему строению ПН-алгоритм приводит к раскраске не более чем $w(G)$ цветами. Для класса w-совершенных графов существует линейный по времени последовательный алгоритм минимальной раскраски [10].

Применим эвристику последовательного ПН-алгоритма для раскраски w-совершенных графов в классе параллельных локальных алгоритмов.

Теорема 1. *Задача раскраски w-совершенных графов ПН-алгоритмом разрешима в классе параллельных локальных алгоритмов.*

Доказательство. Для доказательства построим соответствующий локальный алгоритм. Пусть каждая вершина графа имеет следующие параметры:

- степень: ***deg(v);***
- случайное значение: ***rndvalue (v);***
- номер, соответствующий ПН-упорядочению: ***SLnumber (v)*** (в начале номера всех вершин равны единице, т.е. SLnumber (v)=1);
- показатель состояния параметра SL-number: ***state (v),*** который имеет либо значение I (intermediate) – промежуточное, либо значение F (final) – конечное (в начале все вершины имеют промежуточное состояние, state (v)=I);
- количество соседних вершин, для которых не установлены конечные номера, т.е. cond (v) = I: ***ddeg (v***) (в начале ddeg (v) = deg (v));
- палитра "запрещенных" цветов, т.е. цвета, которые были использованы соседними вершинами: ***usedcolor (v)*** (в начале пустая).

Пусть $v_1, v_2 \in V$. Мы говорим, что вершина v_1 имеет более высокий приоритет, чем v_2, если: ddeg (v_1) < ddeg (v_2) или (ddeg (v_1) = ddeg (v_2)) и (rndvalue (v_1) < rndvalue (v_2)).

На каждом шаге алгоритма все неокрашенные вершины параллельно и независимо друг от друга проделывают следующие действия:

1. Вершина v выбирает параметр rndvalue (v) $\in$ [0..1].

2. Посылает всем соседям следующие параметры: ddeg (v), rndvalue (v).

3. Сравнивает свои параметры с полученными от соседей параметрами и проверяет, которая из вершин имеет более высокий приоритет. Если вершина v имеет высокий приоритет, то она оставляет себе текущее значение параметра SLnumber и меняет значение параметра state (v) с промежуточного на конечный. В противном случае увеличивает на единицу значение параметра SLnumber.

4. Пересчитывает параметр ddeg (v). Если ddeg (v) = 0, т.е. всем смежным вершинам назначены конечные ПН-номера, увеличить на единицу значение параметра SLnumber и изменить значение параметра state (v) с промежуточного на конечный, переход к шагу 5, иначе переход к шагу 2.

5. Посылает всем соседям параметры: SLnumber (v) и первый предполагаемый цвет с наименьшим номером (не находящийся в списке "запрещенных").

6. Сравнивает свои параметры с полученными от соседей, проверяет, какая вершина имеет наибольший SLnumber, если вершина имеет наибольший номер, то оставляет предполагаемый цвет и завершает все действия.

7. В противном случае обновляет список usedcolor (v), переход к шагу 5.

Алгоритм заканчивается, когда все вершины графа приходят в состояние бездействия.

ПН-алгоритм можно условно разделить на два этапа.

Прямой ход. Каждой вершине, имеющей минимальную степень, устанавливается номер, соответствующий ПН-упорядочению (1–4 шаги).

Обратный ход. Производится раскраска графа G, начиная с вершин, которые имеют большие ПН-номера (5–7 шаги).

В работе [11] показано, что данный алгоритм действительно решает поставленную задачу. *Теорема доказана.*

Лемма 2. *Параллельный локальный ПН-алгоритм оптимально или почти оптимально ($c(G)-\chi(G)\le 1$) красит графы из класса*

w-совершенных графов не более чем $w(G)$ цветами за время $O(\Delta^2 \log n)$.

4. Разновидности раскраски графов

4.1. T-раскраска графов. T-раскраска графа G является обобщением вершинной раскраски графа, она была введена в качестве модели для назначения передатчикам радиочастот [12, 13].

Предположим, что G – простой граф и T – конечное множество неотрицательных целых чисел, включая 0. *T*-раскраска графа G – это некоторая функция *c*, которая назначает целое число (цвет) каждой вершине G так, что, если $\{u,v\} \in E$, тогда $c(u) - c(v) \notin T$. Назовем множество *T* множеством запрещенных расстояний.

Отметим, что вершинная раскраска и T-раскраска графа взаимосвязаны, каждая T-раскраска G является вершинной раскраской G и каждая вершинная раскраска G является его {0}-раскраской.

Алгоритм T-DSATUR – это применение одного хорошо известного полиномиального НПН (DSATUR)-алгоритма для T-раскраски графов. Он начинается назначением цвета 1 вершине с максимальной степенью. Затем, используя принцип жадного алгоритма, раскрашиваются вершины с наибольшей степенью насыщения $\deg_s(v)$. Если в графе имеются, по крайней мере, две вершины с одинаковыми степенями насыщения, то алгоритм выберет вершину с максимальной степенью.

Теорема 3. *Задача T-раскраски графов НПН (DSATUR)-алгоритмом разрешима в классе параллельных локальных алгоритмов.*

Доказательство. Алгоритм получается некоторой модификацией локального ПН-алгоритма, которая включает введение нового параметра вершины $\deg_s(v)$, соответствующего степени насыщения; изменение приоритетов, а также включение элементов множества *T* в список использованных цветов *usedcolor(v)*. Таким образом, данный список будет содержать не только цвета, использованные соседями, но и все запрещенные цвета. Например, если T={0,2} и вершина v имеет двух раскрашенных соседей: u_1 в цвет 1 и u_2 в цвет 3, тогда список *usedcolor(v)* состоит из {1, 3, 5}.

Главным отличием данных алгоритмов является то, что если локальный ПН-алгоритм условно состоит из двух этапов (прямого и обратного хода), то локальный НПН (DSATUR)-алгоритм состоит всего из одного этапа (прямого хода), т.е. после сравнения вершиной v своих и полученных от смежных вершин параметров назначаются цвета вершинам, имеющим высокие приоритеты.

Доказательство проводим аналогично доказательству Теоремы 1. *Теорема доказана.*

Покажем, что самые популярные последовательные алгоритмы раскраски налагают строгие нижние границы на ожидаемое время вычислений в распределенной среде.

Лемма 4 [14]. *Для любого исполнения локального НП или НПН (DSATUR)-алгоритма требуется $\Omega(n)$ времени.*

Хотя параллельный локальный НПН (DSATUR)-алгоритм требует больше времени на выполнение, но при этом он использует меньшее число цветов для раскраски произвольных графов и дает меньший интервал, чем другие последовательные алгоритмы.

Двудольным графом называется граф, у которого существует такое разбиение множества вершин на две части (доли), что концы каждого ребра принадлежат разным долям.

Лемма 5. *Параллельный локальный НПН (DSATUR)-алгоритм оптимально или почти оптимально красит все двудольные графы для произвольных множеств T за время $\Omega(n)$.*

4.2. Суммирующая раскраска графов. Задача суммирующей раскраски состоит в нахождении правильной вершинной раскраски графа G, в которой минимизируется общая сумма цветов всех вершин графа G. Эта минимальная общая сумма называется хроматической суммой графа, $\sum(G)$, т.е. $\sum G := \min \sum(G,c)$, где $\sum(G,c) = \sum_{v \in V(G)} c(v)$ и $c \stackrel{c}{:=} V(G) \to \mathrm{N}$ – правильная вершинная раскраска графа G. Раскраска графа, при которой достигнута минимальная хроматическая сумма, называется оптимальной раскраской.

Рассмотрим еще одну стратегию последовательной раскраски графа G, а именно НП-алгоритм для суммирующей раскраски в классе параллельных локальных алгоритмов. Из определения НП-алгоритма известно, что данный алгоритм раскрашивает вершины графа в порядке убывания их степеней. Наш алгоритм является обратным к НП-алгоритму, т.е. вершины графа раскрашиваются в порядке возрастания их степеней.

Для рассматриваемого алгоритма справедлива следующая теорема.

Теорема 6. *Задача суммирующей раскраски графов, с помощью обратного НП-алгоритма разрешима в классе параллельных локальных алгоритмов.*

Доказательство теоремы проводится аналогично доказательствам теорем 1 и 2. Строим локальный обратный НП-алгоритм для суммирующей раскраски, который состоит из одного этапа (прямого хода). Отличием данного алгоритма от предыдущих алгоритмов является изменение приоритетов вершин. На каждом шаге цвета параллельно назначаются неконфликтующим вершинам, имеющим наименьшие степени в своей окрестности.

Приведем некоторые классы графов, для которых параллельный локальный обратный НП-алгоритм дает оптимальную суммирующую раскраску.

Теорема 7. *Параллельный локальный обратный НП-алгоритм оптимально красит полные k-дольные графы.*

Доказательство. При $k = 1$ доказательство очевидно. Для доказательства истинности теоремы при $i = k$ предположим, что теорема справедлива для $i = 1,2,\ldots,k-1$. Пусть $V = P_1 \cup P_2 \cup \ldots \cup P_k$, где P_i является независимым множеством графа G. Пусть $v \in P_i$ является вершиной, получившей цвет на первом шаге. Заметим, что первый цвет будет присвоен только вершинам из P_i. Таким образом, после первого шага будет раскрашено некоторое количество вершин из P_i. На втором шаге вершины из $\overline{P_i} = V \setminus P_i$ добавляют цвет с номером 1 в список "запрещенных" цветов. Так, вершины из P_i, не раскрашенные на первом шаге, получат цвет 1 без всякого конфликта на втором шаге. Теперь мы можем отдельно рассмотреть вершины из $\overline{P_i}$, где $\overline{P_i}$ образует полный $(k-1)$–дольный граф, который будет раскрашен $k-1$ цветами. *Теорема доказана.*

Теорема 8. *Параллельный локальный обратный НП-алгоритм оптимально красит двудольное колесо BW_k, для $k \geq 2$, не более чем за три шага.*

Доказательство. Двудольное колесо – это граф, получаемый из четного цикла длиной 2k путем добавления центральной вершины, соединенной с каждой второй вершиной цикла.

На первом шаге цвет 1 получат вершины, не соединенные с центральной вершиной и имеющие наименьшие степени в графе, на втором шаге цвет 2 получат вершины, соединенные с центральной вершиной, и на третьем шаге центральная вершина без всяких конфликтов получит цвет 1. *Теорема доказана.*

Теорема 9. *Параллельный локальный обратный НП-алгоритм оптимально красит двойную звезду за три шага.*

Доказательство. Двойная звезда – это дерево с *n-2* листьями.

На первом шаге будут раскрашены все листья в цвет 1, на втором и на третьем шагах будут раскрашены оставшиеся две вершины в цвета 2 и 3. *Теорема доказана.*

Заключение. В настоящей работе рассмотрена задача вершинной раскраски неориентированных графов в классе параллельных локальных алгоритмов. Выполнены вычислительные эксперименты для случайных графов применительно к предлагаемым алгоритмам. Все тесты сделаны в параллельной среде, где каждая вершина разделяет процесс. Все процессы работают независимо, без центрального управления. Эксперименты показали, что применение известных стратегий последовательного алгоритма, опирающихся на упорядочение вершин в определенном порядке для раскраски определенных классов графов, а также для частных случаев вершинной раскраски графов дают оптимальные или почти оптимальные результаты.

Литература

1. *Журавлев Ю.И.* Локальные алгоритмы вычисления информации // Кибернетика. 1965. № 1. С. 12–19.
2. *Журавлев Ю.И.* Алгоритмы построения минимальных дизъюнктивных нормальных форм для функций алгебры логики // Дискретная математика и математические вопросы кибернетики. Т. 1. М.: Наука, 1974. С. 67–98.
3. *Евстигнеев В.А.* О некоторых свойствах локальных алгоритмов на графах // Комбинаторно-алгебраические методы в прикладной математике. Горький: Изд-во ГГУ, 1983. С. 72–105.
4. *Евстигнеев В.А.* Локальные алгоритмы на графах и проблема децентрализованной обработки информации // II Всес. конф. по прикладной логике. Тез. докл. Новосибирск, 1988. С. 75–77.
5. *Евстигнеев В.А.* Локальные алгоритмы и распределенные вычисления // Проблемы теоретической кибернетики. Тез. докл. VIII Всес. конф. Горький, 1988. С. 113–114.
6. *Johansson Ö.* Simple distributed $\Delta+1$ - coloring of graphs // Inf. Process. Letters, 1999. Vol. 70. P. 229–232.

7. *Grable D.A., Panconesi A.* Fast distributed algorithms for Brooks-Vizing colorings // J. Algorithms. 2000. Vol. 37. P. 85–120.
8. *Szekeres G., Wilf H.S.* An inequality for the chromatic number of a graph // J. Combin. Theory. 1964. Vol. 4. P. 1–3.
9. *Евстигнеев В.А.* Применение теории графов в программировании. М.: Наука, 1985. 352 с.
10. *Matula D.W., Bleck L.L.* Smallest-last ordering and dustering and graph coloring algorithms // J. Assoc. Comput. Math. 1983. Vol. 30. N. 3. P. 417–427.
11. *Евстигнеев В.А., Турсунбай кызы Ы.* Динамический распределенный ПН-алгоритм для раскраски w-совершенных графов // Методы и инструменты конструирования программ. Новосибирск, 2007. С. 24–30.
12. *Hale W.K.* Frequency assignment: theory and applications // Proc. of the IEEE, 1980. Vol. 68. N. 12. P. 1497–1514. 38.
13. *Cozzens M.B., Roberts F.S.* T-colorings of graphs and the channel assignment problem // Congressus Numerantium. 1982. Vol. 35. P. 191–208.
14. *Kosowski A., Kuszner L.* On greedy graph coloring in the distributed model // Proc. of EuroPar. Lect. Notes Comput. Sci. Springer-Verlag, 2006. Vol. 4128. P. 592–601.