УДК 519.854

С.В. Листровой, д-р техн. наук, **С.В. Моцный**, аспирант
Украинский государственный университет железнодорожного транспорта
(Украина, 61050, Харьков, пл. Фейербаха, 7,
тел. +380509355042, +380987290907,
e-mail: om1@yandex.ru, stanislav.motsnyi@gmail.com)

Алгоритм решения задачи о наименьшем вершинном покрытии произвольного графа с помощью систем квадратичных уравнений

Предложен алгоритм решения задачи о наименьшем покрытии произвольного графа с помощью систем квадратичных уравнений, которые позволяют достигать высокой степени распараллеливания операций. Для решения этой задачи на практике используются приближенные алгоритмы с различными коэффициентами аппроксимации. Приведены результаты экспериментального анализа, свидетельствующие о преимуществе описанного алгоритма по сравнению с существующими.

Запропоновано алгоритм роз'язку задачі про найменше покриття довільного графа за допомогою систем квадратичних рівнянь, які дозволяють досягати високого ступіню розпаралелювання операцій. Для розв'язку цієї задачі на практиці використовують наближені алгоритми з різними коефіцієнтами апроксимації. Наведено результати експериментального аналізу, які свідчать про перевагу описаного алгоритму в порівнянні з існуючими.

К л ю ч е в ы е с л о в а: вершинное покрытие, степень аппроксимации, системы квадратичных уравнений, частота появления слагаемых.

Проблема поиска наименьшего покрытия в произвольном графе была одной из первых задач, выделенных в категорию *NP*-полных [1]. Было сделано множество попыток разработать методы, которые позволили бы найти точное решение за полиномиальное время. Однако до сих пор известны лишь алгоритмы [2], дающие приближенный результат, т.е. воспроизводящие такое вершинное покрытие, в котором число вершин не более чем в k раз превосходит минимально возможное (k — точность приближенного алгоритма).

В последние десятилетия данной проблеме уделяется особое внимание, так как задачу о наименьшем покрытии приходится часто решать в теории и на практике. В частности, алгоритмы решения этой задачи широко используются при мониторинге телекоммуникационных сетей [3], давая возможность эффективно обнаруживать медленные и (или) неисправные участки, при выполнении процедуры выравнивания биологических последовательностей (белков, РНК, ДНК и др.) [4], которая позволяет анализировать сходные участки в этих последовательностях, их структурные и эволюционные взаимосвязи, а также при исследовании многих других проблем вычислительной биологии [5].

Данная проблема актуальна также при планировании и распределении ресурсов в гетерогенных высоконагруженных системах, где большое влияние на производительность оказывает именно выбор эффективного алгоритма нахождения наименьшего покрытия, который обеспечивал бы устойчивость системы при высокой интенсивности поступления заданий и выборку необходимых ресурсов с минимальными затратами.

Несмотря на то что точные алгоритмы решения данной задачи нецелесообразно использовать на практике ввиду их чрезвычайно высокой временной сложности, в большинстве случаев достаточно уровня аппроксимации более быстрых приближенных методов.

Анализ последних исследований. В последние годы проблема нахождения оптимального алгоритма решения задачи о наименьшем покрытии рассматривалась с учетом фиксированной параметрической сложности [6]. В основе параметрического моделирования лежит идея разделения проблемы на две составляющие. С одной стороны, имеется совокупность множества входных данных, а с другой, — множество различных параметров, так или иначе влияющих на общую вычислительную сложность исследуемого алгоритма. Это позволяет сформировать более гибкую классификацию *NP*-сложных проблем по сравнению с классическим подходом, когда сложность алгоритма зависит только от входного потока.

Многие задачи вычислительной теорий сложности могут быть представлены в следующем виде. Пусть имеется экземпляр x и неотрицательное целое число k. Обладает ли x такими свойствами, которые зависят от k? Для задачи о вершинном покрытии параметром k может быть число вершин, которое необходимо найти. Очевидно, что параметр k весьма важен при оценке сложности решения задачи. Если предположить, что $P \neq NP$, то существует большое число проблем, для решения которых требуется экспоненциальное вычислительное время.

При использовании параметрического подхода эти проблемы можно решить за полиномиальное время при условии, что входной параметр k

имеет фиксированное значение. Иными словами, если имеется некоторая функция $f(k)$, влияющая на параметрическую сложность алгоритма, и k равняется относительно небольшому целочисленному значению, то вычислительная сложность равна $O(f(k)\times n^{O(1)})$, где n — число входных данных.

Задачи, в которых используется фиксированный параметр k, называются параметризированными и относятся к классу сложности FPT (Fixed-parameter tractability — выполнимость при фиксированном параметре). Задача о вершинном покрытии также относится к данному классу. Уже давно проводятся исследования, связанные с поиском наиболее быстрого параметризированного алгоритма. В настоящее время один из самых быстрых алгоритмов решает данную задачу за $O(kn\times 1{,}2738^k)$ шагов [7].

При решении оптимизационных задач на практике часто используют приближенные алгоритмы, важнейшей характеристикой которых является коэффициент аппроксимации.

Доказано, что в случае p-аппроксимированного алгоритма [2] отношение $f(x)$ (значение—стоимость) для *NP*- трудной задачи x удовлетворяет следующим ограничениям:

$$\begin{aligned} &OPT \le f(x) \le pOPT,\ p>1,\\ &pOPT \le f(x) \le OPT,\ p<1, \end{aligned} \tag{1}$$

где OPT — точное решение поставленной задачи x; $f(x)$ — отношение значение—стоимость в решении задачи x, определяющее степень аппроксимации.

Для оценки качества работы аппроксимированного алгоритма часто используется понятие относительной погрешности. Пусть s — приближенное решение, s^* — соответствующее оптимальное решение, c и c^* — временные затраты работы алгоритма. Тогда относительная погрешность определяется так:

$$\mu_\varepsilon(s) = \frac{c(s) - c^*(s)}{c(s^*)}. \tag{2}$$

Один из наиболее оптимальных алгоритмов предложен в работе [8], где коэффициент аппроксимации удалось уменьшить в соответствии с (1) до $\left(2-\Theta\left(\frac{1}{\sqrt{\log n}}\right)\right)$. В основе данного алгоритма лежит сбалансированное разбиение графа по заданным правилам, а также использование методов полуопределенного программирования. Однако на практике при выборе

приближенного алгоритма для достижения наиболее оптимальных показателей погрешности (2) необходимо учитывать конкретную модель используемого графа. Например, для графов с низкой плотностью распределения вершин предложен метод [9], основанный на эвристическом варианте поиска с рекомбинацией решений-кандидатов. Этот алгоритм относится к разряду генетических, отличительной особенностью которых является использование механизмов, аналогичных естественному отбору в природе.

В работе [10] описан приближенный алгоритм, основанный на каноническом статистическом анализе Монте-Карло, с помощью которого можно проводить выборку вершин для занесения в покрытие с прогнозированной вероятностью. В отличие от других этот алгоритм позволяет балансировать между необходимой точностью и сложностью вычислительных операций.

К недостаткам популярных методов решения задачи о наименьшем покрытии можно отнести нерешенную проблему распараллеливания операций для повышения эффективности выполнения в распределенных средах, а также слишком высокие фиксированные коэффициенты, снижающие производительность даже при небольших входных значениях.

При разработке оптимального алгоритма решения задачи о наименьшем покрытии в произвольных графах для применения в распределенных средах и условиях высокой нагрузки с улучшенными по сравнению с существующими методологиями характеристиками особое внимание уделено улучшению показателя аппроксимации получаемых решений. Результаты экспериментального анализа позволили оценить возможность использования предлагаемой методики для решения других задач подобного класса сложности.

Постановка и решение задачи. Для общей формализации задачи о наименьшем покрытии используем понятие произвольного неориентированного графа, под которым подразумевается пара $G(V, E)$, где V — множество вершин, E — множество ребер данного графа. Ребром в неориентированном графе называют неупорядоченную пару вершин $(u, v) \in E$. Вершинным покрытием данного графа является такое подмножество вершин $V' \subseteq V$, что каждое ребро $(u, v) \in G$ удовлетворяет следующему критерию: $u \in V' \vee v \in V'$ (т.е. множеству V' принадлежит хотя бы одна из вершин u или v). В задаче о наименьшем покрытии требуется найти минимально возможный размер вершинного покрытия в исследуемом графе.

Предлагаемый алгоритм решения поставленной задачи состоит из двух процедур: ***A*** и ***B***. Процедура ***A*** состоит из 11 базовых шагов. При выполнении этих шагов периодически возникает необходимость в вызове вспомогательной процедуры ***B***, которая, в свою очередь, позволяет про-

вести проверку на наличие в графе висячих вершин. Под висячей подразумевается вершина, которая является концом ровно одного ребра.

Минимальное вершинное покрытие зависит от локальной структуры выбранного графа. Общая топология и плотность распределения позволяет проводить прогнозированную выборку пар вершин с наиболее оптимальной общей степенью. На основании этой выборки составляется система квадратичных уравнений, при решении которой, в зависимости от конкретного шага алгоритма, возвращается либо частичный $R_z^{\times}$, либо полный $R_z^{\text{п}}$ результат.

Полученный результат является наименьшим покрытием заданного графа. Переменная z в данном случае характеризует вариант прогнозирования, т.е. выбор слагаемого из одной из трех возможных пар переменных, рассматриваемых на шаге 3.

Процедура А. Шаги поиска минимального вершинного покрытия в произвольном графе следующие:

Ш а г 1. По заданному графу формируем исходное квадратичное уравнение:

$$f_z(x_i x_j) = 0, \tag{3}$$

где $x_i\ x_j$ — пары переменных, соответствующие вершинам графа. Одна из переменных для образования покрытия должна быть равна нулю.

Ш а г 2. Обрабатываем уравнение (3) с помощью вспомогательной процедуры ***В***. Если процедура ***В*** возвращает второй возможный вариант результата (все возможные результаты перечислены в описании процедуры ***В***), то минимальное покрытие найдено. Оно является полным решением $R_z^{\text{п}}$, и процедура ***А*** заканчивает работу. Если получен первый или третий возможный результат работы процедуры ***В***, то переходим к следующему шагу.

Ш а г 3. В зависимости от того, какой результат получен на предыдущем шаге, в уравнении (3) или производном от него уравнении $f_z^{|}(x_i\ x_j) = 0$, имеющем частичное решение $R_z^{\times}$, находим слагаемое $S_p^* = x_l x_m$ с максимальной суммарной частотой появления переменных в текущем уравнении ($h_s = h_l + h_m$) и формируем три прогнозируемые пары переменных: $x_l = 0$, $x_m = 0$; $x_l = 1$, $x_m = 0$; $x_l = 1$, $x_m = 1$. Затем переходим к следующему шагу.

Ш а г 4. Переменной z присваиваем значение 1 и подставляем первую пару переменных $x_l = 0$, $x_m = 0$ в текущее уравнение. Получаем новое текущее уравнение $f_1(x_i\ x_j) = 0$, а переменные x_l и x_m вносим в частичное решение $R_z^{\times}$. Применяем к текущему уравнению процедуру ***В***. Если в результате работы процедуры ***В*** получен второй вариант, то покрытие

найдено, оно определяется полным решением $R_z^{\text{п}}$. Запоминаем его и переходим к выполнению следующего шага.

Ш а г 5. В зависимости от того, какой результат получен на предыдущем шаге в результате работы процедуры ***B***, уравнение $f_1(x_i\, x_j) = 0$ (или $f_1^{|}(x_i\, x_j) = 0$) вместе с характеризующим его текущим частичным решением заносим во множество *M*, куда заносим также все полные решения $R_z^{\text{п}}$, полученные на предыдущих шагах. Переходим к следующему шагу.

Ш а г 6. Переменной *z* присваиваем значение 2 и подставляем вторую пару переменных $x_l = 1$, $x_m = 0$ в текущее уравнение. Получаем новое текущее уравнение $f_2(x_i\, x_j) = 0$. Слагаемые $(x_t\, x_j)$ уравнения (3), содержащие по одной переменной $\{x_t\}$, приравниваем нулю. В результате получаем новое текущее уравнение $f_2^{|}(x_i\, x_j) = 0$. Переменные $\{x_j\}$, соседние с $\{x_t\}$, и переменную x_m заносим в частичное решение $R_z^{\times}$. Применяем к текущему уравнению процедуру ***B***. Если при этом получаем второй вариант результата работы процедуры ***B***, то покрытие найдено, оно определяется полным решением $R_z^{\text{п}}$. Запоминаем его и переходим к выполнению следующего шага.

Ш а г 7. Уравнение $f_2^{|}(x_i\, x_j) = 0$ (или $f_2^{\|}(x_i\, x_j) = 0$, в зависимости от того, какой результат получен на предыдущем шаге процедуры ***B***) вместе с характеризующим его текущим частичным решением заносим во множество *M*, куда заносим также все полные решения $R_z^{\text{п}}$, полученные на предыдущих шагах, и переходим к выполнению следующего шага.

Ш а г 8. Переменной *z* присваиваем значение 3 и подставляем третью пару переменных $x_l = 0$, $x_m = 1$ в текущее уравнение. Получаем новое текущее уравнение $f_3(x_i\, x_j) = 0$. Полагаем слагаемые $(x_t\, x_j)$ в данном уравнении, содержащие по одной переменной $\{x_t\}$, равными нулю. Получаем новое текущее уравнение $f_3^{|}(x_i\, x_j) = 0$, а переменные $\{x_j\}$, соседние с $\{x_t\}$, и переменную x_l заносим в частичное решение $R_z^{\times}$. Применяем к текущему уравнению процедуру ***B***. Если получаем второй вариант результата работы процедуры ***B***, то покрытие найдено. Оно определяется полным решением $R_z^{\text{п}}$, которое запоминаем и переходим к выполнению следующего шага.

Ш а г 9. Уравнение $f_3^{|}(x_i\, x_j) = 0$ (или $f_3^{\|}(x_i\, x_j) = 0$, в зависимости от того, какой результат был получен на предыдущем шаге в результате работы процедуры ***B***) вместе с характеризующим его текущим частичным решением заносим во множество *M*, и туда же заносим все полные решения $R_z^{\text{п}}$, полученные на предыдущих шагах. Переходим к выполнению следующего шага.

Ш а г 10. Проверяем, все ли уравнения в *M* превращены в тождества вида 0 = 0. Если да, то среди всех полных решений $\{R_z^{\text{п}}\}$ выбираем минимальное, которое и будет определять минимальное вершинное покрытие

графа. На этом процедура заканчивает работу, иначе — выполняем следующий шаг.

Ш а г 11. Из текущих уравнений $f_1(x_i\ x_j)=0, f_2(x_i\ x_j)=0, f_3(x_i\ x_j)=0$, которые еще не обратились в тождество вида 0 = 0, выбираем уравнение $f_i^*(x_i\ x_j)=0$ с наибольшим числом слагаемых. Меняем используемое уравнение (3) на уравнение $f_i^*(x_i\ x_j)=0$ и переходим к выполнению шага 2. Вся последовательность шагов повторяется до тех пор, пока минимальное вершинное покрытие не будет найдено.

Следует заметить, что если ни одно из трех уравнений не обращается в тождество вида 0 = 0, то выбирается уравнение с максимальным числом слагаемых независимо от того, входят в него два оставшихся уравнения или нет, во множестве *M* при этом сохраняется только выбранное уравнение и частичное решение, относящееся к оставшемуся уравнению.

Введем вспомогательную процедуру ***B***, которая периодически повторяется внутри процедуры ***A***. Процедура ***B*** позволяет выполнять проверку наличия в уравнении слагаемых с переменными, встречающимися один раз. Подобные переменные ассоциированы с висячими вершинами графа. Следовательно, они удаляются из текущего решения, а смежные с ними вершины заносятся в покрытие, в результате чего удается избежать избыточности получаемых решений и улучшить показатель аппроксимации алгоритма.

Процедура B. Поиск висячих вершин.

Ш а г 1. Проверяем, есть ли в уравнении (3) слагаемые $x_q x_j \in f_*(x_i\ x_j)$ с переменными $\{x_q\}$, встречающимися один раз. Если да, то всем переменным $\{x_j\}$, соседним с $\{x_q\}$, присваиваем значение, равное нулю, и включаем их в частичное решение $R_i^{\times}$. При этом уравнение (3) преобразуется в уравнение $f_z^{|}(x_i\ x_j)=0$ с меньшим числом слагаемых. Переходим к выполнению следующего шага, иначе уравнение (3) остается неизменным и процедура заканчивает работу.

Ш а г 2. Проверяем, обратилось ли уравнение $f_z^{|}(x_i\ x_j)=0$ в тождество вида 0 = 0. Если да, то частичное решение $R_z^{\times}$ становится полным $R_z^{\text{п}}$, поскольку переменные, вошедшие в $R_i^{\text{п}}$, определяют вершинное покрытие в исходном графе, и процедура заканчивает работу. Если нет, то меняем уравнение (3) на уравнение $f_z^{|}(x_i\ x_j)=0$ и переходим к выполнению шага 1.

В результате применения процедуры ***B*** к уравнению (3) можно получить три следующих варианта:

1) уравнение $f_z(x_i\ x_j)=0$ не изменяется;

2) уравнение обращается в тождество вида 0 = 0, и получаем полное решение $R_z^{\text{п}}$;

3) получаем новое уравнение $f_z^{|}(x_i\ x_j)=0$ с меньшим числом слагаемых и некоторое частичное решение $R_z^{\times}$.

Рассмотрим пример использования предлагаемого алгоритма. Составим перечень связей между вершинами исследуемого графа:

1: 3 6 7 8 12; 2: 5 6 9 11 12; 3: 1 4 5 7 8 9 10 11; 4: 3 6 8 9 10;
5: 2 3 7 8 11 12; 6: 1 2 4 7 8 9; 7: 1 3 5 6 8; 8: 1 3 4 5 6 7 9;
9: 2 3 4 6 8 10 12; 10: 3 4 9 11; 11: 2 3 5 10; 12: 1 2 5 9.

Составляем исходное уравнение графа:

$$x_1x_3+x_1x_6+x_1x_7+x_1x_8+x_1x_{12}+x_2x_5+x_2x_6+x_2x_9+x_2x_{11}+$$
$$+x_2x_{12}+x_3x_4+x_3x_5+x_3x_7+x_3x_8+x_3x_9+x_3x_{10}+x_3x_{11}+x_4x_6+$$
$$+x_4x_8+x_4x_9+x_4x_{10}+x_5x_7+x_5x_8+x_5x_{11}+x_5x_{12}+x_6x_7+x_6x_8+x_6x_9+$$
$$+x_7x_8+x_8x_9+x_9x_{10}+x_9x_{12}+x_{10}x_{11}=0. \qquad (4)$$

Список вершинных покрытий и независимых множеств данного графа представлен в следующей таблице:

Вершинное покрытие	Независимое множество
1 2 3 5 6 8 9 10 (8)	4 7 11 12 (4)
1 2 3 4 5 6 7 9 10 (9)	8 11 12 (3)
1 2 3 4 5 6 7 9 11 (9)	8 10 12 (3)
1 2 3 4 5 6 8 9 11 (9)	7 10 12 (3)
1 2 3 4 5 6 8 10 12 (9)	7 9 11 (3)
1 2 3 4 5 7 8 9 10 (9)	6 11 12 (3)
1 2 3 4 5 7 8 9 11 (9)	6 10 12 (3)
1 2 3 4 7 8 9 11 12 (9)	5 6 10 (3)
1 2 4 5 7 8 9 10 11 (9)	3 6 12 (3)
1 3 4 5 6 7 9 11 12 (9)	2 8 10 (3)
1 3 4 5 6 8 9 11 12 (9)	2 7 10 (3)
1 3 5 6 8 9 10 11 12 (9)	2 4 7 (3)
2 3 4 5 6 7 8 10 12 (9)	1 9 11 (3)
2 3 4 6 7 8 9 11 12 (9)	1 5 10 (3)
2 3 4 6 7 8 10 11 12 (9)	1 5 9 (3)
2 3 5 6 7 8 9 10 12 (9)	1 4 11 (3)
2 3 6 7 8 9 10 11 12 (9)	1 4 5 (3)
3 4 5 6 7 8 9 11 12 (9)	1 2 10 (3)
3 5 6 7 8 9 10 11 12 (9)	1 2 4 (3)
1 4 5 6 7 8 9 10 11 12 (10)	2 3 (2)

Определяем частоты h_i^x появления переменных X_i в слагаемых уравнения (4) и на основании полученных значений составляем таблицу:

X_i	1	2	3	4	5	6	7	8	9	10	11	12
h_i^x	5	5	8	5	6	6	5	7	7	4	4	4

Выбираем слагаемое с максимальной суммарной частотой появления переменных в уравнении (4). Таким является слагаемое x_3x_8 с суммарной частотой появления 8 + 7 = 15. На основе этого слагаемого формируем систему из трех уравнений, учитывая поочередно выполнение следующих прогнозируемых равенств: $x_3 = 0$, $x_8 = 0$; $x_3 = 0$, $x_8 = 1$; $x_3 = 1$, $x_8 = 0$.

Первое уравнение получаем, учитывая в (4) выполнение первого равенства ($x_3 = 0$, $x_8 = 0$):

$$x_1x_6 + x_1x_7 + x_1x_{12} + x_2x_5 + x_2x_6 + x_2x_9 + \\ + x_2x_{11} + x_2x_{12} + x_4x_6 + x_4x_9 + x_4x_{10} + x_5x_7 + x_5x_{11} + x_5x_{12} + \\ + x_6x_7 + x_6x_9 + x_9x_{10} + x_9x_{12} + x_{10}x_{11} = 0. \quad (5)$$

В частичное решение первого уравнения включаем переменные x_3, x_8. Второе уравнение (при $x_3 = 0$, $x_8 = 1$) получаем на основании того, что переменные в (4) $x_1 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$, $x_7 = 0$, $x_9 = 0$, так как $x_8 = 1$. Следовательно, уравнение (4) примет вид

$$x_2x_{11} + x_2x_{12} + x_{10}x_{11} = 0. \quad (6)$$

При этом в частичное решение включаем переменные x_3 и переменные x_1, x_4, x_5, x_6, x_7, x_9. Третье уравнение (при $x_3 = 1$, $x_8 = 0$) получаем аналогично: из равенства $x_3 = 1$ следуют равенства $x_1 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$, $x_7 = 0$, $x_9 = 0$, $x_{10} = 0$ и уравнение (4) принимает вид

$$x_2x_6 + x_2x_{12} = 0. \quad (7)$$

В частичное решение включаем переменные x_8 и x_1, x_4, x_5, x_6, x_7, x_9, x_{10}. Поскольку слагаемые уравнений (6) и (7) входят в уравнение (5), уравнения (6) и (7) могут быть исключены из дальнейшего анализа.

Далее определяем частоты появления переменных для уравнения (5) и заносим их в следующую таблицу:

X_i	1	2	4	5	6	7	9	10	11	12
h_i^x	3	5	3	4	5	3	4	3	3	4

Выбираем слагаемое с максимальной суммарной частотой появления в (5). Таким слагаемым является x_2x_6 с суммарной частотой 5 + 5 = 10. На основе

уравнения (5) формируем новую систему из трех уравнений, учитывая поочередно выполнение следующих равенств: $x_2 = 0$, $x_6 = 0$; $x_2 = 0$, $x_6 = 1$; $x_2 = 1$, $x_6 = 0$.

Первое уравнение получаем из (5) с учетом равенства $x_2 = 0$, $x_6 = 0$. После обнуления соответствующих переменных уравнение (5) принимает следующий вид:

$$x_1x_7 + x_1x_{12} + x_4x_9 + x_4x_{10} + x_5x_7 + \\ + x_5x_{11} + x_5x_{12} + x_9x_{10} + x_9x_{12} + x_{10}x_{11} = 0. \quad (8)$$

При этом в частичное решение включаем x_2, x_6 и переменные x_3, x_8, включенные на предыдущем шаге. Второе уравнение получаем, учитывая, что в (5) $x_2 = 0$, $x_6 = 1$. Из равенства $x_6 = 1$ следуют равенства $x_1 = 0$, $x_4 = 0$, $x_7 = 0$, $x_7 = 0$, $x_9 = 0$. Таким образом, уравнение (5) принимает вид

$$x_5x_7 + x_5x_{11} + x_5x_{12} + x_{10}x_{11} = 0. \quad (9)$$

Теперь в частичное решение включаем переменные x_2 и x_1, x_4, x_7, x_9 . Третье уравнение (при $x_2 = 1$, $x_6 = 0$) получаем аналогично: из равенства $x_2 = 1$ следуют равенства $x_5 = 0$, $x_9 = 0$, $x_{11} = 0$, $x_{12} = 0$. Уравнение (5) преобразуется к виду

$$x_1x_7 + x_1x_{12} + x_2x_5 + x_4x_{10} + x_5x_7 = 0. \quad (10)$$

При этом в частичное решение включаем переменные x_6 и переменные x_5, x_9, x_{11}, x_{12}. Слагаемые уравнений (9) и (10) входят в (8), поэтому уравнения (9) и (10) могут быть исключены из дальнейшего анализа.

Далее определяем частоту появления переменных в уравнении (8) и на основании полученных результатов формируем следующую таблицу:

X_i	1	4	5	7	9	10	11	12
h_i^x	2	2	3	2	3	3	2	3

Выбираем слагаемое с максимальной суммарной частотой появления переменных в уравнении (8). Таким является слагаемое $x_9 x_{10}$ с суммарной частотой появления 3 + 3 = 6. На основе этого слагаемого формируем систему из трех уравнений, полагая поочередно выполнение следующих равенств: $x_9 = 0$, $x_{10} = 0$; $x_9 = 0$, $x_{10} = 1$; $x_9 = 1$, $x_{10} = 0$.

Первое уравнение формируем на основе (8), принимая во внимание, что $x_9 = 0$, $x_{10} = 0$. Пользуясь стандартными формулами булевой алгебры, получаем новое прогнозированное уравнение:

$$x_1x_7 + x_1x_{12} + x_5x_7 + x_5x_{11} + x_5x_{12} = 0. \quad (11)$$

Второе уравнение составляем на основе второй пары переменных: $x_9 = 0$, $x_{10} = 1$. Из равенства $x_{10} = 1$ следует, что в (8) $x_4 = 0$, $x_9 = 0$, $x_{11} = 0$. Тогда уравнение примет вид

$$x_1x_7 + x_1x_{12} + x_5x_7 + x_5x_{12} = 0. \quad (12)$$

Третье уравнение (при $x_9 = 1$, $x_{10} = 0$) получаем аналогично: из равенства $x_9 = 1$ следуют равенства $x_4 = 0$, $x_9 = 0$, $x_{12} = 0$, и уравнение (8) принимает вид

$$x_1x_7 + x_5x_7 + x_5x_{11} = 0. \quad (13)$$

Слагаемые уравнений (12) и (13) входят в уравнение (11), поэтому уравнения (12), (13) могут быть исключены из дальнейшего анализа.

Далее, для уравнения (11) с частичным решением x_3, x_8, x_2, x_6, x_9, x_{10} определяем частоту появления переменных и на основании полученных результатов формируем следующую таблицу:

X_i	1	5	7	11	12
h_i^x	2	3	2	1	2

Поскольку x_{11} встречается один раз, переменную $x_5 = 0$ включаем в частичное решение, в результате чего уравнение (11) преобразуется к виду

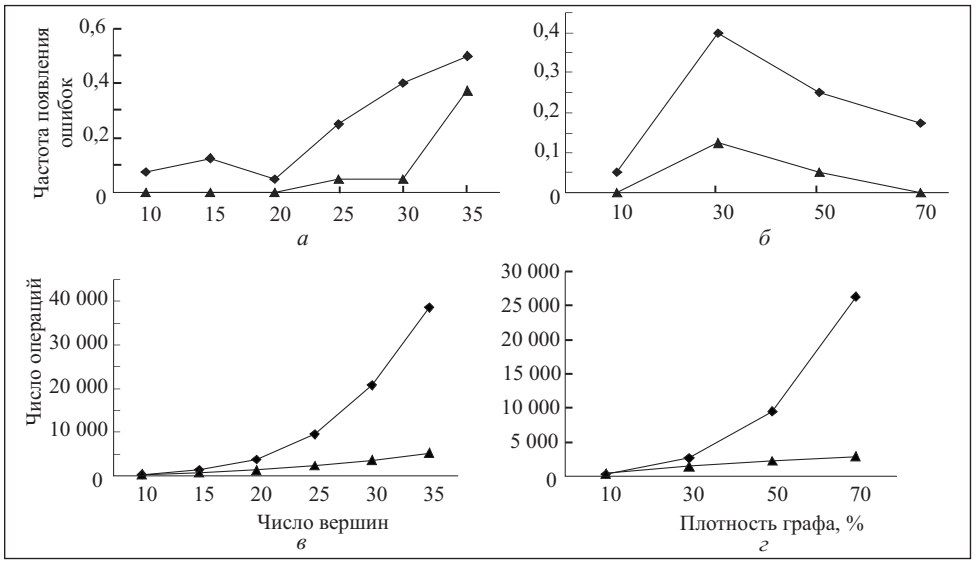
$$x_1x_7 + x_1x_{12} = 0. \quad (14)$$

Определяем частоты появления переменных в уравнении (14) с частным решением x_3, x_8, x_2, x_6, x_9, x_{10}, x_5 и заносим их в таблицу:

X_i	1	7	12
h_i^x	2	1	1

При этом появилось несколько висячих вершин (имеющих степень 1), из которых произвольно выбираем любую для обработки. В покрытие должна быть добавлена вершина, смежная с выбранной. Поскольку $x_7 x_{12}$ встречается один раз, переменная $x_1 = 0$ должна быть включена в частичное решение. Тогда уравнение (14) принимает вид $0 = 0$, т.е. получено тождество. Таким образом, набор переменных x_3, x_8, x_2, x_6, x_9, x_{10}, x_5, x_1 определяет минимальное вершинное покрытие заданного графа. Это же покрытие содержится в первой строке первой таблицы. Следовательно, решение, найденное с помощью предлагаемого алгоритма, является точным.

Результаты экспериментальных исследований. Для проведения экспериментальных исследований и дальнейшего сравнительного анализа разра-



Графики зависимостей частоты появления ошибок (*а*, *б*) и числа операций (*в*, *г*) от числа вершин (*а*, *в*) и плотности графа (*б*, *г*): ■ — частотный метод; ▲ — метод уравнений

ботано программное обеспечение на языке программирования C++, которое позволяет генерировать графы согласно равномерному закону распределения с учетом задаваемых характеристик (плотность и число вершин).

Предлагаемый метод решения задачи о наименьшем покрытии (метод уравнений) сравним с частотным методом [11, 12], который имеет временную сложность $O(mn^2)$ (где m — число ребер графа, n — число вершин) и является одним из наиболее точных методов решения данной задачи. На рисунках *а*, *б* представлены зависимости частоты появления неточных решений алгоритмов от числа вершин и плотности графа. При проведении экспериментов размерность графа менялась от 10 до 35 вершин, плотность — от 10 до 70 %.

Результаты экспериментального анализа свидетельствуют о том, что предлагаемый метод уравнений более эффективен как на начальных стадиях тестирования, так и при значительном увеличении вершинных характеристик графа, и позволяет получать более точные решения по сравнению с частотным методом. Так, при использовании частотного метода погрешность решения не превышает 4 %, а при использовании метода уравнений максимальное зафиксированное значение составило 1,74 %, т.е. данный показатель удалось улучшить практически в два раза. Из рисунков *в* и *г* видно, что при улучшенных показателях аппроксимации предлагаемый метод уравнений значительно меньше зависит от сложности топологии выбранного графа.

Обычно при увеличении показателя точности приближенного алгоритма существенно возрастает его временная сложность. Как показали результаты тестирования, временная сложность алгоритма с использованием систем квадратичных уравнений в среднем не превышает величины mn^2. При этом его эффективность значительно улучшается при использовании большого числа вычислительных ядер. Высокая степень распараллеливания инструкций достигается в результате использования возможности одновременного решения каждого из прогнозируемых вариантов с одной из трех вероятных пар переменных по ходу выполнения алгоритма. Естественно, существует необходимость в синхронизации процедур, возвращаемые результаты которых требуется получать в определенный момент времени. При реализации данного алгоритма в распределенной среде время его выполнения сокращается в 2,3 раза по сравнению с частотным методом.

Выводы

В результате проведеного сравнительного анализа установлено, что предложенный метод решения задачи о наименьшем покрытии произвольного графа с помощью систем квадратичных уравнений более эффективен по сравнению с частотным методом, который в настоящее время является одним из наиболее точных. Как свидетельствуют результаты анализа, при решении задач в масштабах реального времени предлагаемый метод превосходит существующие как по показателям погрешности, так и по времени выполнения. При этом возможность гибкого распараллеливания выполняемых процедур позволяет наиболее оптимально использовать ресурсы современных вычислительных систем. Эффективность алгоритма значительно повышается при использовании распределенных систем с большим количеством вычислительных ядер.

СПИСОК ЛИТЕРАТУРЫ

1. *Кристофидес Н.* Теория графов. Алгоритмический подход. — М. : Мир, 1978. — 309 с.
2. *Vijay V. Vazirani*. Approximation Algorithms. 2nd Ed. — Berlin: Springer-Verlag, 2003. — P. 306—334.
3. *Zhang Y., Qi G., Fleisher R. et al.* Approximating the minimum weight weak vertex cover // Elsevier: Theoretical Computer Science. —1992. — Vol. 7, Iss. 4. — P. 404—416.
4. *Roth-Korostensky C.* Algorithms for building multiple sequence alignments and evolutionary trees// Doctoral thesis. — ETH Zürich, Institute of Scientific Computing, Switzerland, 2000. — 161 p.
5. *Stege U.* Resolving conflicts in problems from computational biology// Doctoral thesis. — ETH Zürich, Institute of Scientific Computing, Switzerland, 2000. — 158 p.
6. *Jianer C., Iyad K.A., Ge X.* Improved Parameterized Upper Bounds for Vertex Cover// Elsevier: Theoretical Computer Science. — 2010. — Vol. 411. — P. 3736—3756.
7. *Jianer C., Iyad K.A., Ge X.* Simplicity is beauty: Improved upper bounds for vertex cover// Technical Report 05-008, Texas A&M University, Utrecht, Netherlands, April 2005.

8. *Karakostas G.* A better approximation ratio for the vertex cover problem// ACM Transactions on Algorithms (TALG). — 2009. — Vol. 5, Iss. 4. — P. 1—8.
9. *Cheng W., Yuren Z., Weiping T.* Hybrid genetic algorithm for vertex cover problem// Computer Engineering and Applications. — 2007. — Vol. 43, No. 14. — P. 27—29.
10. *Miehalewiez Z., Fogel D.B.* How to Solve It: Modern Heuristics. — Berlin, Heidelberg: Springer-Verlag, 2004. — 554 p.
11. *Листровой С.В., Минухин С.В.* Метод решения задач о минимальном вершинном покрытии в произвольном графе и задачи о наименьшем покрытии// Электрон. моделирование. — 2012. — **34,** № 1. — С. 29—45.
12. *Listrovoy S.V., Minukhin S.V.* Investigation of the Scheduler for Heterogeneous Distributed Computing Systems based on Minimal Cover Method// Intern. Journal of Computer Applications. — 2012. — Vol. 51, No.19. — P. 35—44.

S.V. Listrovoy, S.V. Motsnyi

THE MINIMUM VERTEX COVER PROBLEM
SOLVING ALGORITHM FOR AN ARBITRARY GRAPH
WITH USING THE SYSTEMS OF QUADRATIC EQUATIONS

This paper presents an algorithm for solving the minimum vertex cover problem for the arbitrary graphs using systems of quadratic equations that provide high level of the operations parallelization. Approximation algorithms with different approximation coefficients can be used in practice to solve such problems. Experimental analysis shows the advantages of the described methodology in comparison with existing implementations. The algorithm effectiveness can be significantly enhanced by the use of distributed systems with many cores.

K e y w o r d s : vertex cover, approximation coefficient, quadratic equations, frequency of terms occurrences.

REFERENCES

1. Christofides, N. (1978), *Teoriya grafov. Algoritmicheskiy podkhod* [Graph Theory. An Algorithmic Approach], Mir, Moscow, Russia.
2. Vijay Vazirani, V. (2003), Approximation Algorithms, 2nd ed., Springer-Verlag, Berlin, Germany, pp. 306-334.
3. Zhang, Y. and et al. (1992), “Approximating the minimum weight weak vertex cover”, *Elsevier, Theoretical Computer Science*, Vol. 7, no. 4, pp. 404-416.
4. Roth-Korostensky, C. (2000), “Algorithms for building multiple sequence alignments and evolutionary trees”, PhD thesis, ETH Zürich Institute of Scientific Computing, Zürich, Switzerland.
5. Stege, U. (2000), “Resolving conflicts in problems from computational biology”, PhD thesis, ETH Zürich Institute of Scientific Computing, Zürich, Switzerland.
6. Jianer, C., Iyad, K.A. and Ge, X. (2010), “Improved Parameterized Upper Bounds for Vertex Cover”, *Elsevier, Theoretical Computer Science*, Vol. 411, Iss. 40-42, pp. 3736-3756.
7. Jianer, C., Iyad, K.A. and Ge, X. (2005), “Simplicity is beauty: Improved upper bounds for vertex cover”, *Technical Report 05-008*, Texas A&M University, Utrecht, the Netherlands.
8. Karakostas, G. (2009), “A better approximation ratio for the vertex cover problem”, *ACM Transactions on Algorithms (TALG),* Vol. 5, Iss. 4, pp. 1-8.
9. Cheng, W., Yuren, Z. and Weiping, T. (2007), “Hybrid genetic algorithm for vertex cover problem”, *Computer Engineering and Applications*, Vol. 43, no. 14, pp. 27-29.

10. Miehalewiez, Z. and Fogel, D.B. (2004), How to solve it: Modern heuristics, 2nd ed., Springer-Verlag, Heidelberg, Berlin, Germany.
11. Listrovoy, S.V. and Minukhin, S.V. (2012), "Method for solving the minimum vertex cover problem in arbitrary graphs and the problem of minimal coverage", *Elektronnoe modelirovanie*, Vol. 34, no. 1, pp. 29-45.
12. Listrovoy, S.V. and Minukhin, S.V. (2012), "Investigation of the scheduler for heterogeneous distributed computing systems based on minimal cover method", *International Journal of Computer Applications,* Vol. 51, no. 19*,* pp. 35-44.

Поступила 06.05.15;
после доработки 05.10.15

ЛИСТРОВОЙ Сергей Владимирович, д-р техн. наук, профессор, профессор кафедры специализированных компьютерных систем Украинского государственного университета железнодорожного транспорта. В 1972 г. окончил Харьковское высшее военное командно-инженерное училище. Область научных исследований — задачи дискретной оптимизации и теории графов и их приложение к анализу вычислительных систем и сетей.

МОЦНЫЙ Станислав Владимирович, аспирант кафедры специализированных компьютерных систем Украинского государственного университета железнодорожного транспорта. В 2012 г. окончил Украинскую государственную академию железнодорожного транспорта (г. Харьков). Область научных исследований — информационная безопасность в телекоммуникационных системах и сетях, оптимизация процессов в распределенных системах.