УДК 519.682.1

С. В. Листровой, д-р техн. наук
Украинская государственная академия железнодорожного транспорта
(Украина, 61050, Харьков, пл. Фейербаха, 7,
тел. 0509355042, e-mail: om1@ yandex.ru),
С. В. Минухин, канд. техн. наук
Харьковский национальный экономический университет
(Харьков, 61001, пр. Ленина, 9-А,
тел. (057) 702-18-31, e-mail: ms_vl@mail.ru)

Метод решения задач о минимальном вершинном покрытии в произвольном графе и задачи о наименьшем покрытии

Предложены приближенные алгоритмы решения задачи о наименьшем вершинном покрытии (ЗНВП) в произвольных графах и задачи о наименьшем покрытии (ЗНП) на основе сведения их соответственно к задачам квадратичного и нелинейного булевого программирования, специфика которых позволила построить алгоритмы с временной сложностью, не превышающей $O(mn^2)$, где в случае решения ЗНВП в произвольных графах n — число вершин, а m — число ребер в графе, а в случае решения ЗНП n — число столбцов, а m — число строк в матрице B. Показано, что погрешность решения этих задач предложенными процедурами $\boldsymbol{A_1}$ и $\boldsymbol{A_2}$ не превышает 5 % при плотности строк матрицы B, равной 0,5 и более.

Предложенные алгоритмы могут быть использованы для эффективного планирования распределения ресурсов в GRID-системах в масштабе реального времени при достаточно жестких ограничениях на время решения задач, если допустимое время планирования находится в диапазоне от 5 до 100 мс.

Запропоновано наближені алгоритми розв'язування задачі про найменше вершинне покриття (ЗНВП) у довільних графах і задачі про найменше покриття (ЗНП), базовані на зведенні їх відповідно до задач квадратичного та нелінійного булевого програмування, специфіка яких дозволила побудувати алгоритми з часовою складністю, що не перевищує $O(mn^2)$, де у випадку розв'язування ЗНВП у довільних графах n — число вершин, а m — число ребер у графі, а при розв'язуванні ЗНП n — число стовпців, а m — число рядків у матриці B. Показано, що похибка розв'язку цих задач запропонованими процедурами $\boldsymbol{A_1}$ і $\boldsymbol{A_2}$ не перевищує 5 % при густині рядків матриці B, що дорівнює 0,5 і більше.

Запропоновані алгоритми можна використовувати для ефективного планування розподілу ресурсів у GRID-системах в масштабі реального часу при достатньо жорстких обмеженнях в часі для розв'язування задачі, коли припустимий час планування має діапазон від 5 до 100 мс.

Ключевые слова: вершинные покрытия в графах, наименьшее покрытие столбцами строк в матрице, состоящей из единиц и нулей, GRID.

Задачи о наименьшем вершинном покрытии (ЗНВП) и задачи о наименьшем покрытии (ЗНП) имеют широкое прикладное значение в теории построения сложных систем, вычислительных систем и сетей [1], при разработке их программного и математического обеспечения, а также для планирования распределения ресурсов в GRID. Кроме того, ЗНП широко применяются при диагностике GRID-систем и сетей [2], при размещении пунктов обслуживания [1, 3], в системах информационного поиска, при назначении экипажей на транспорте [1, 3—5], проектировании интегральных схем [5] и конвейерных линий. Основным требованием к алгоритмам решения данных задач является высокая оперативность и обеспечение минимально возможной погрешности.

Задачу нахождения независимых максимальных множеств или вершинных покрытий можно решить, например, последовательным перебором независимых множеств с одновременной проверкой каждого множества на максимальность (последнее осуществляется добавлением к исследуемому множеству дополнительной вершины, не принадлежащей ему, и выяснением, сохраняется ли независимость) и запоминанием максимальных множеств. Однако с увеличением числа вершин этот способ становится весьма громоздким. На основе усовершенствования этой процедуры построены алгоритмы Брона и Кэрбоша [1].

Как показано в работе [3], задача вершинного покрытия является трудно решаемой и эффективные алгоритмы ее решения для произвольных графов неизвестны. Для двудольных графов на основе алгоритмов Хопкрофта и Карпа (с поиском в глубину) разработаны методы [3], позволяющие находить минимальное вершинное покрытие и максимальное независимое множество вершин в произвольном двудольном графе $H =< X, Y, E >$ за время $O((m+n)\sqrt{n})$, где $n = |X \cup Y|$ и $m = |E|$.

Полиномиальные алгоритмы определения числа устойчивости были получены для совершенных графов, т.е. графов, у которых для любого порожденного подграфа хроматическое число равно кликовому числу. Алгоритм вычисления числа устойчивости графа [4] основан на методе эллипсоидов с использованием процедуры отделения матриц графа. Однако в вычислительном плане этот алгоритм имеет ряд существенных недостатков, не позволяющих использовать его на практике. Как показано в работе [4], получить правильное решение при числе вершин в графе более 10 практически невозможно. Все известные детерминированные точные алгоритмы решения ЗНП [1—3, 5—8] имеют экспоненциальную сложность.

Следует заметить, что применение известных жадных алгоритмов типа Greedy-Set-Cover [9—12] может давать погрешность порядка $O(\log n)$. В работе [13] показано, что погрешность аппроксимации жадного алгоритма

для решения ЗНП составляет $lnn - lnlnn + O(1)$. Другие алгоритмы этого типа [14] построены на анализе гармонического ряда:

$$H_k = \frac{\sum_{i=1}^{k} 1}{i+c},$$

где k — мощность множества, определяющего покрытие; c — некоторая константа. Для этого гармонического ряда получены оценки аппроксимации [14], уточняющие величину константы c.

Таким образом, наиболее приемлемой для проведения сравнительного анализа имеющихся теоретических результатов эффективности аппроксимации жадных алгоритмов является оценка $O(\log n)$, пропорциональная величине логарифма от числа столбцов (вершин) в матрице B. Эти алгоритмы с трудом поддаются распараллеливанию, и следовательно, их применение в качестве средств планирования в распределенной среде, например в GRID-системах, представляется затруднительным.

Разработанный метод позволяет сократить сложность и погрешность решения ЗНВП для произвольных графов и ЗНП.

Постановка и решение задачи. Рассмотрим формализацию ЗНВП в произвольных графах, для чего введем понятие разборки и сборки графа из базовых элементов. Под базовыми элементами $\{l_i\}$ графа $G(X, E)$ с множеством вершин X и множеством ребер E будем понимать вершины графа $\{i \in X\}$ вместе с инцидентными им ребрами. Будем полагать, что базовые элементы графа l_i и l_j могут быть объединены, если они имеют общие вершины или ребра. Тогда под объединением двух базовых компонентов $l_i \cup l_j$ будем понимать подграф графа G, полученный объединением одноименных ребер и вершин в базовых элементах l_i и l_j и соединением всех вершин в объединенной конструкции в соответствии со связями между вершинами исходного графа G. Ясно, что эта операция коммутативна, т.е. $l_i \cup l_j = l_j \cup l_i$.

Объединение компонент возможно, если есть общие ребра или вершины в объединяемых компонентах. Процедуру удаления базовых компонент из графа назовем разборкой графа G. Если в процессе разборки графа удаляется базовая компонента l_i, то оставшийся подграф обозначим G_i, если удаляется две компоненты, l_i и l_j, то подграф обозначим G_{ij} и т. д. Если в графе n вершин, то число базовых компонент, из которых состоит граф G, равно n.

Например, граф G, представленный на рис. 1, содержит пять вершин и имеет пять базовых компонент. Ясно, что граф G можно рассматривать как объединение всех компонент l_i, т.е. $G = \bigcup_{1}^{n} l_i$.

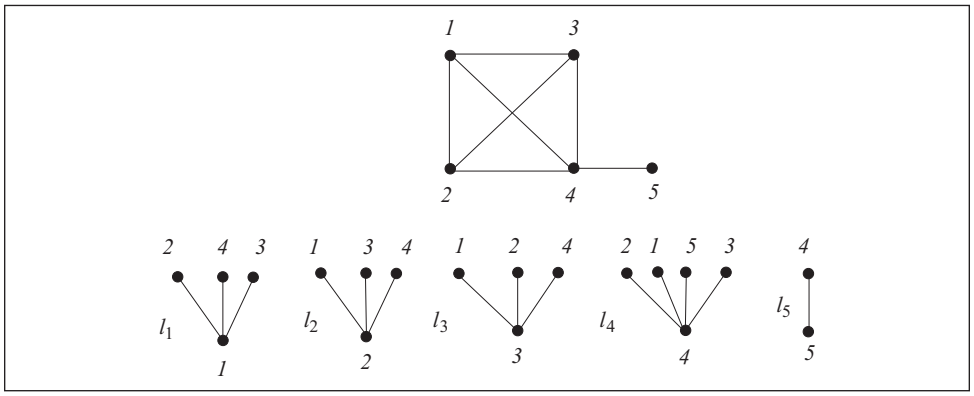

Рис. 1. Граф G и его базовые компоненты

Поставим в соответствие каждой вершине графа переменную x_i. Тогда каждую компоненту l_i можно описать в виде произведения

$$x_i \sum_{q \in m_{iq}} x_q,$$

где m_{iq} — множество вершин, смежных вершине i в компоненте l_i. Если вершину i включить в вершинное покрытие, то можно считать, что $x_i = 0$. Умножение на $x_i = 0$ в произведении

$$x_i \sum_{q \in m_{iq}} x_q$$

обращает в нуль все ребра, покрываемые вершиной i в компоненте l_i.

Рассмотрим сумму

$$x_1 \sum_{q \in m_{iq}} x_q + x_2 \sum_{q \in m_{iq}} x_q + \ldots + x_n \sum_{q \in m_{iq}} x_q. \qquad (1)$$

Если подмножество вершин $\{x_i\}$ образует покрытие, то сумма (1) будет равна нулю. Поскольку в каждой паре $x_i x_j$ одна из переменных для образования покрытия должна быть равна нулю, после перемножения в (1) будут справедливы равенства $x_i x_j + x_i x_j = x_i x_j$, и соотношение (1) можно переписать в виде

$$x_1 \sum_{q \in m_{iq}} x_q + x_2 \sum_{q \in m_{iq}} x_q + \ldots + x_n \sum_{q \in m_{iq}} x_q = \sum_{i,\, j \in E} x_i x_j. \qquad (2)$$

Тогда задачу определения минимального вершинного покрытия можно сформулировать в виде

$$\min_i \{x_i\} \qquad (3)$$

при выполнении следующего ограничения:

$$\sum_{i,j\in E} x_i x_j = 0. \tag{4}$$

Функционал (3) означает поиск минимального числа переменных $x_i = 0$, которые равенство (4) обратят в тождество, т.е. приходим к задаче квадратичного программирования. С учетом (2) задачу (3) можно переписать при выполнении следующих ограничений:

$$x_1 \sum_{q\in m_{iq}} x_q = 0;\ x_2 \sum_{q\in m_{iq}} x_q = 0 \ldots\ x_n \sum_{q\in m_{iq}} x_q = 0. \tag{5}$$

Система уравнений (5) составлена на основе базовых компонент графа. При этом число уравнений равно числу базовых компонент. Например, для графа, представленного на рис. 1, данная система уравнений имеет вид

$$\begin{gathered} x_1x_2 + x_1x_3 + x_1x_4 = 0, \\ x_2x_3 + x_2x_4 + x_2x_1 = 0, \\ x_4x_1 + x_4x_2 + x_4x_3 = 0, \\ x_5x_4 = 0. \end{gathered} \tag{6}$$

Для определения минимального вершинного покрытия в графе G необходимо найти наименьшее число переменных $x_i = 0$, обращающих в тождество систему уравнений (6).

В работе [2] показано, что справедливо следующее утверждение.

Утверждение. Если в графе G есть висячая вершина i, образованная ребром (i, j), то вершина j принадлежит одному из минимальных вершинных покрытий в графе G, если в графе G их несколько, и минимальному вершинному покрытию, если оно в графе G одно.

Наличие висячей вершины в графе означает, что есть хотя бы одна базовая компонента графа, содержащая одно ребро (см. рис. 1), ей в (6) соответствует уравнение $x_5x_4 = 0$, и следовательно, необходимо в (6) положить $x_4 = 0$, т.е. включить четвертую вершину в покрытие. При этом система примет вид

$$\begin{gathered} x_1x_2 + x_1x_3 = 0, \\ x_2x_3 + x_2x_1 = 0. \end{gathered} \tag{7}$$

Переменные x_1, x_2 и x_3 одинаково часто встречаются в оставшихся уравнениях, поэтому любую из них включаем в покрытие. Полагая $x_2 = 0$, получаем $x_1\, x_3 = 0$. Затем в покрытие включаем либо вершину *1* либо вер-

шину *3*, т.е. получаем два минимальных покрытия: вершины {*4*, *2*, *1*} либо {*4*, *2*, *3*}. Правильность решения легко проверить, перечислив все вершинные покрытия графа *G*, представив его в виде булевой функции

$$f = (x_1 \vee x_2)(x_1 \vee x_3)(x_1 \vee x_4)(x_2 \vee x_3)(x_2 \vee x_4)(x_3 \vee x_4)(x_4 \vee x_5). \quad (8)$$

Перемножив выражения в скобках и использовав правило поглощения, получим перечень вершинных покрытий графа:

$$f = x_1x_2x_4 \vee x_1x_3x_4 \vee x_2x_3x_4 \vee x_1x_2x_3x_5.$$

Следовательно, в графе *G* (см. рис. 1) имеется три минимальных покрытия: {*4*,*2*,*1*}, {*4*,*2*,*3*}, {*1*,*3*,*4*}, которые пересекаются по множеству вершин {*1*,*2*,*3*}, соответствующих множеству переменных $\{x_1, x_2, x_3\}$. Эти переменные одинаково часто встречались в уравнениях на некотором шаге поиска максимального числа переменных, обнуляющих соответствующие пары слагаемых в уравнениях (7).

Введем следующую процедуру решения задачи (3), (5).

Процедура A_1.

Ш а г 1. Проверяем, есть ли в системе уравнений переменные, встречающиеся один раз. Если да, то переменные, находящиеся с ними в паре, приравниваем нулю, а соответствующие вершины вносим в покрытие и выполняем следующий шаг, если нет, то переходим к выполнению шага 3.

Ш а г 2. Проверяем, все ли парные слагаемые в системе уравнений обнулены. Если нет, то переходим к выполнению шага 1, если иначе, то алгоритм заканчивает работу, и сформированное покрытие является минимальным.

Ш а г 3. Находим в системе уравнений переменную, которая чаще всего встречается в парных произведениях системы уравнений. Если таких переменных несколько, то выбираем любую из них и приравниваем ее нулю, а соответствующую ей вершину вносим в покрытие и переходим к выполнению шага 2.

Исследуем оптимальность работы процедуры $\boldsymbol{A_1}$. Для этого рассмотрим три следующих случая.

1. Предположим, что на каждом шаге работы процедуры появляются переменные, которые встречаются один раз. Тогда на каждом шаге, в соответствии с утверждением 1, подсоединяем к покрытию вершину, гарантировано принадлежащую минимальному вершинному покрытию, и, следовательно, полученное в результате работы процедуры $\boldsymbol{A_1}$ покрытие будет минимальным.

2. Предположим, что в исходном графе *G* существует одно минимальное покрытие и на каждом шаге 3 работы процедуры $\boldsymbol{A_1}$ существует одна

переменная, которая чаще всего встречается в парных произведениях системы уравнений. Покажем, что в данном случае также получаем минимальное вершинное покрытие.

Пусть в данном случае процедура построила покрытие мощности k. Предположим, что в графе G существует покрытие мощности $k-1$, но это возможно только в том случае, если существует переменная, встречающаяся чаще, чем переменные, включенные в покрытие мощности k. Однако это противоречит тому факту, что на каждом шаге 3 работы процедуры $\boldsymbol{A_1}$ в покрытие включалась вершина с максимальной частотой появления в парных произведениях системы уравнений (5), и, следовательно, покрытие, полученное процедурой в этом случае, также будет минимальным.

3. Предположим, что в исходном графе G существует не одно минимальное покрытие. Это означает, что в процессе работы процедуры $\boldsymbol{A_1}$ возникает ситуация, когда на некотором шаге оказывается, что максимальное число переменных, которое может быть обнулено вследствие выбора различных переменных $\{x_i\}$, одинаково. Это соответствует тому, что в графе G существует несколько минимальных покрытий, которые либо вообще не пересекаются, либо пересекаются на подмножестве вершин, соответствующих переменным $\{x_i\}$. К сожалению, в случае, когда есть несколько непересекающихся покрытий, процедура $\boldsymbol{A_1}$ может не дать оптимального решения, т.е. процедура дает приближенное решение задачи. Поэтому представляет интерес оценить погрешность ее работы.

Таким образом, процедура $\boldsymbol{A_1}$ позволяет находить приближенное решение задачи (3), (5), а следовательно, и задачи (3), (4). Число уравнений в (3), (5) не превышает n, а число пар слагаемых в уравнении m. Поскольку в процессе работы процедуры придется просмотреть систему уравнений не более чем n раз, временная сложность работы процедуры не может превысить величины $O(mn^2)$.

Формализация и решение ЗНП. Задача о наименьшем покрытии может быть сформулирована как задача линейного булевого программирования [1, 8], постановка которой в общем виде имеет вид

$$L=\sum_{j=1}^{n} x_j \to \min \tag{9}$$

при ограничениях

$$\sum_{j=1}^{n} \beta_{ij} x_j \geq 1,\ \ i=\overline{1,m};\ \ \beta_{ij} \in \{0,1\};\ \ x_j \in \{0,1\}. \tag{10}$$

Задачу (9), (10) можно рассматривать как задачу определения минимального числа столбцов в булевой матрице B, покрывающей единицами

все строки данной матрицы [2]. Произвольную булеву матрицу B будем представлять в виде булевой функции f, заданной в конъюнктивной нормальной форме [2, 6], в которой число дизъюнктов равно числу строк в матрице, а число переменных в каждом дизъюнкте — числу единиц в строке матрицы B. Тогда задачу о минимальном покрытии для произвольной матрицы B, задаваемой некоторой булевой функцией

$$f = (X_l \vee X_m \vee ... \vee X_k)(X_s \vee X_r \vee ... \vee X_t)...(X_q \vee X_d \vee ... \vee X_h), \qquad (11)$$

можно рассматривать как задачу нахождения минимального набора переменных $\{X_i = 1\}$, при которых булева функция (11) выполнима. Запишем ее в виде

$$\min_i \{X_i = 1\} \qquad (12)$$

при выполнении ограничений

$$(X_l \vee X_m \vee ... \vee X_k)(X_s \vee X_r \vee ... \vee X_t)...(X_q \vee X_d \vee ... \vee X_h) = 1. \qquad (13)$$

Перейдя к двойственной булевой функции, получим

$$\min_i \{X_i = 0\}, \qquad (14)$$

$$X_l X_m ... X_k \vee X_s X_r ... X_t \vee ... \vee X_q X_d ... X_h = 0. \qquad (15)$$

Из уравнений (14), (15) следует, что задачу о наименьшем покрытии можно рассматривать как задачу нелинейного булевого программирования, заключающуюся в нахождении наименьшего числа переменных $\{X_i = 0\}$, обращающего в нуль левую часть ограничения (15).

Справедливо следующее утверждение: если переменная $X_{i=q}$ встречается в слагаемых один раз, то ее можно положить в (15) равной единице, т.е. переменная $X_{i=q}$ не входит в минимальное покрытие.

Справедливость этого утверждения видна из постановки задачи (12), (13). Если в (13) переменная $X_{i=q}$ в дизъюнктах встречается один раз, то это значит, что все переменные, стоящие в этом же дизъюнкте, встречаются хотя бы два или более раз в других дизъюнктах, и, если их положить равными единице, то они покроют и тот дизъюнкт, в котором находится переменная $X_{i=q}$, и еще несколько дизъюнктов, в которых они присутствуют. Следовательно, эта переменная не может принадлежать минимальному покрытию и ее можно положить в (13) равной нулю. Соответственно в двойственной булевой функции она будет равна единице, что и требовалось показать.

Следует заметить, что если в результате этого в (15) появится слагаемое, состоящее из одной переменной, то данная переменная должна войти

в минимальное покрытие, так как, если этого не сделать, равенство (15) никогда не будет выполнено.

Используя данное свойство, для решения задачи (14), (15) предлагаем следующую процедуру, аналогичную процедуре A_1.

Процедура A_2.

Ш а г 1. Проверяем, есть ли в слагаемых переменные, встречающиеся один раз. Если да, то полагаем эту переменную равной единице и переходим к выполнению следующего шага.

Ш а г 2. Проверяем, есть ли слагаемые, состоящие из одной переменной. Если да, то полагаем эти переменные равными нулю и вносим их в покрытие, если нет, то переходим к выполнению следующего шага.

Ш а г 3. Проверяем, все слагаемые обнулены или нет. Если да, то процедура заканчивает работу, так как минимальное покрытие сформировано, если иначе, то переходим к выполнению следующего шага.

Ш а г 4. Среди оставшихся слагаемых находим переменную, встречающуюся чаще всего (если таких несколько, то выбираем любую из них), полагаем ее равной нулю и включаем в покрытие, после чего проверяем, все ли слагаемые обнулены. Если да, то процедура заканчивает работу, так как минимальное покрытие сформировано, если иначе, то переходим к выполнению шага 1.

Например, пусть задана матрица B вида

$$B = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 & 0 & 1 & 1 \\ 3 & 0 & 1 & 0 & 0 & 1 & 0 \\ 4 & 1 & 1 & 1 & 0 & 0 & 0 \\ 5 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}.$$

Представим ее в виде булевой функции:

$$f = (X_1 \vee X_2 \vee X_6)(X_3 \vee X_5 \vee X_6)(X_2 \vee X_5)(X_1 \vee X_2 \vee X_3)(X_3 \vee X_4).$$

Двойственная булева функция имеет вид

$$\bar{f} = X_1X_2X_6 \vee X_3X_5X_6 \vee X_2X_5 \vee X_1X_2X_3 \vee X_3X_4. \qquad (16)$$

Поскольку переменная X_4 встречается в слагаемых один раз, полагаем $X_4 = 1$, при этом образовалось слагаемое с одной переменной X_3. Полагаем $X_3 = 0$ и включаем переменную X_3 в покрытие, после чего остались слагае-

мые $X_1X_2X_6 \vee X_2X_5 = 0$. Выбираем переменную, наиболее часто встречающуюся в слагаемых, — это X_2, полагаем $X_2 = 0$ и включаем ее в покрытие. Поскольку при этом все слагаемые булевой функции (16) обнулены, сформированное покрытие из переменных $\{X_2, X_3\}$ является минимальным.

Проанализируем оптимальность работы процедуры $\boldsymbol{A_2}$. Для этого рассмотрим следующих три случая.

1. Предположим, что на каждом шаге работы процедуры появляются слагаемые, содержащие по одной переменной (такая ситуация возможна, когда в дизъюнктах булевой функции матрицы B содержатся по две переменные, что эквивалентно решению ЗНВП в произвольных графах). Тогда на каждом шаге, в соответствие с доказанным утверждением, подсоединяем к покрытию переменную, гарантировано принадлежащую минимальному покрытию. Полученное в результате работы процедуры $\boldsymbol{A_2}$ покрытие будет минимальным.

2. Предположим, что для матрицы B существует одно минимальное покрытие и на каждом шаге 4 работы процедуры $\boldsymbol{A_2}$ существует одна переменная, которая чаще всего встречается в произведениях системы уравнений. Покажем, что в данном случае покрытие также будет минимальным.

Пусть в данном случае построено покрытие мощности k. Предположим, что для матрицы B существует покрытие мощности $k - 1$, но это возможно только в том случае, если существует переменная, встречающаяся чаще, чем переменные, включенные в покрытие мощности k. Однако это противоречит тому факту, что на каждом шаге 4 работы процедуры $\boldsymbol{A_2}$ в покрытие включалась переменная с максимальной частотой появления в слагаемых двойственной булевой функции, и следовательно, покрытие, полученное в этом случае, также будет минимальным.

3. Предположим, что в матрице B существует не одно минимальное покрытие. Это означает, что в процессе работы процедуры $\boldsymbol{A_2}$ возникает ситуация, когда на некотором шаге ее работы максимальное число различных комбинаций переменных, которое может быть обнулено вследствие выбора переменных $\{X_i\}$, одинаково. Это соответствует тому, что в B существует несколько минимальных покрытий, и они либо вообще не пересекаются либо пересекаются на подмножестве переменных $\{X_i\}$. Поскольку рассуждение, приведенное в случае 2, можно применить для каждого покрытия, это означает, что и в этом случае процедура $\boldsymbol{A_2}$ построит по крайней мере одно из минимальных покрытий. Однако, когда покрытия не пересекаются, как и в ЗНВП , возможна потеря оптимального значения.

Таким образом, процедура $\boldsymbol{A_2}$ позволяет находить приближенное решение задачи (14), (15), а следовательно, и задачи (12), (13). Нетрудно видеть, что при числе переменных в дизъюнктах булевой функции, не

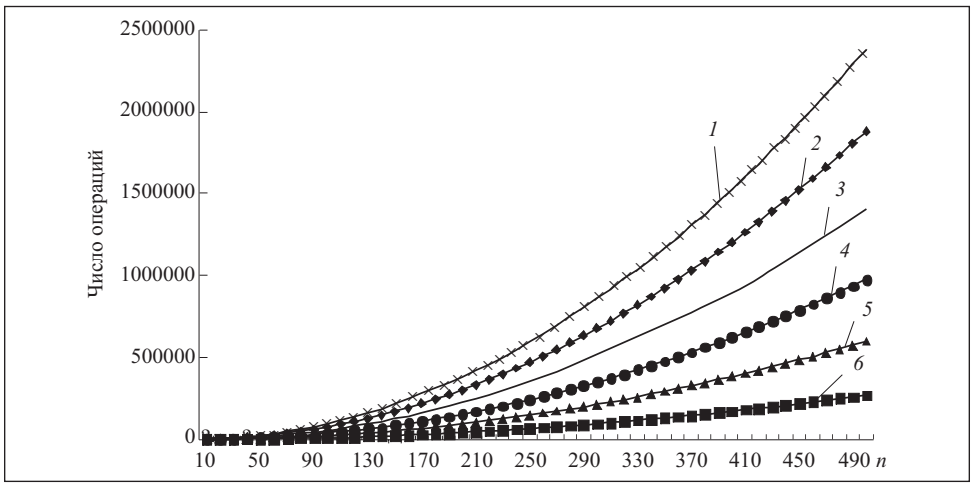

Рис. 2. Зависимость числа операций, выполняемых процедурой A_1 от числа вершин в графе при различных значениях П: *1* — $y = n^2$; *2* — П = 0,9; *3* — П = 0,7; *4* — П = 0,5; *5* — П = 0,3; *6* — П = 0,1

превышающем двух, процедура A_2 фактически вырождается в процедуру A_1. Число слагаемых в (14), (15) не превышает m, число переменных в слагаемом не превышает n, а поскольку в процессе работы процедуры придется просмотреть слагаемые не более чем n раз, то временная сложность работы процедуры не может превысить $O(mn^2)$.

Экспериментальное исследование работы процедур A_1 и A_2. Для проверки точности решения ЗНВП на основе процедуры A_1 в качестве эталонного использован точный алгоритм экспоненциальной сложности, построенный на основе идей рангового подхода [7, 15]. Размерность графа изменялась от 10 до 30 вершин, графы генерировались с различной плотностью ребер П по равномерному закону распределения, на каждую размерность графа генерировались не менее 100 реализаций и при этом измерялась погрешность. Проведена также оценка в среднем временной сложности алгоритма для размерностей графов от 10 до 490 при различных значениях П в графе. Все значения получены с доверительной вероятностью 0,95. Графики зависимости среднего значения числа элементарных операций, выполняемых алгоритмом для различных плотностей ребер в графе, приведены на рис. 2.

Зависимости вероятности решения ЗНВП для различных плотностей П за допустимое время, равное 5 мс, приведены на рис. 3, из которого видно, что в среднем временная сложность алгоритма не превышает $O(n^2)$, и в случае допустимого времени решения, равного 5 мс, могут быть эффективно решены задачи для графов с числом вершин, не превышающем 100.

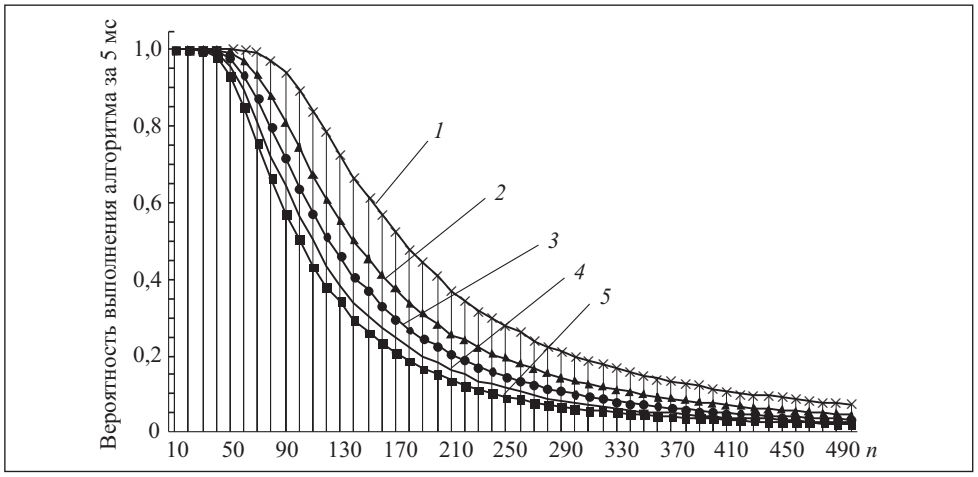

Рис. 3. Вероятность решения задачи процедурой A_1 за допустимое время 5 мс : *1* — П = = 0,1; *2* — П = 0,3; *3* — П = 0,5; *4* — П = 0,7; *5* — П = 0,9

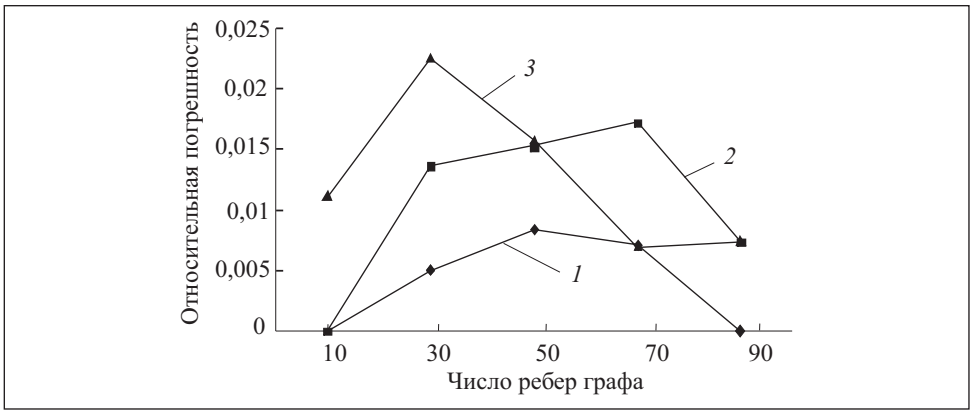

Рис. 4. Зависимость относительной погрешности поиска минимального вершинного покрытия от числа ребер в графе при различном числе вершин: *1* — $n = 10$; *2* — $n = 20$; *3* — $n = 30$

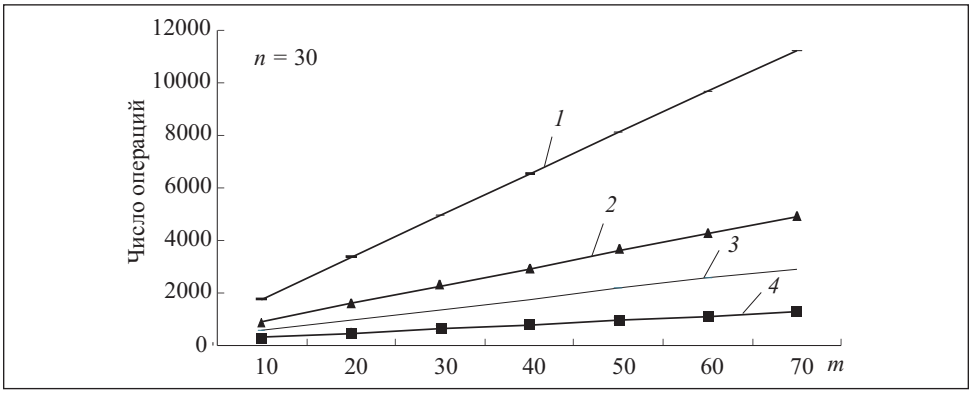

Рис. 5. Зависимость числа выполняемых операций процедурой A_2 от размерности матрицы *B*: *1* — $n \cdot m$; *2* — П = 0,9; *3* — П = 0,7; *4* — П = 0,5

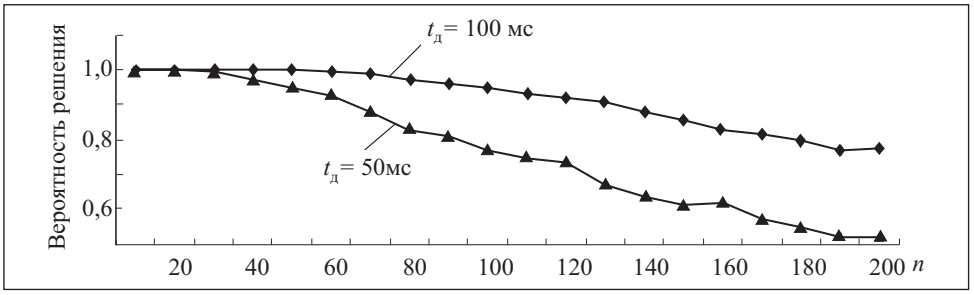

Рис. 6. Зависимость вероятности решения ЗНП процедурой A_2 от числа столбцов в матрице B при m = 300, П = 70 % и допустимом времени решения $t_д$

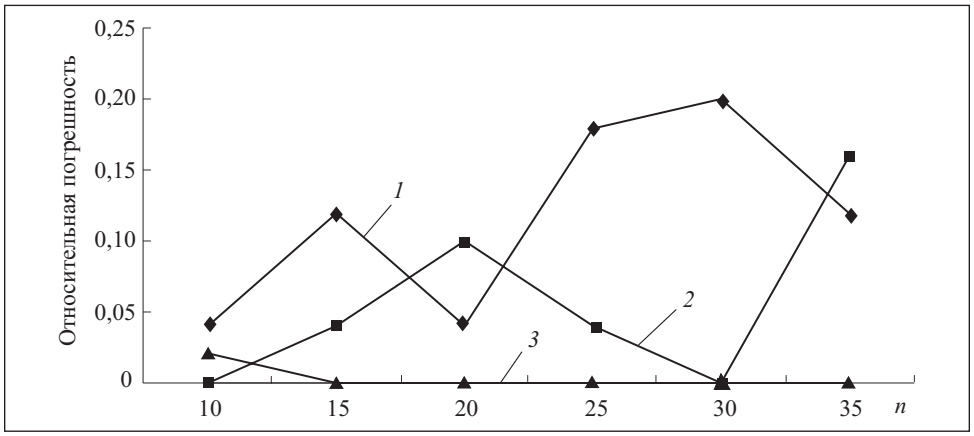

Рис. 7. Зависимость относительной погрешности работы процедуры A_2 от числа столбцов в матрице B при m = 30 и различных значениях П: *1* — П = 0,1; *2* — П = 0,3; *3* — П = 0,5

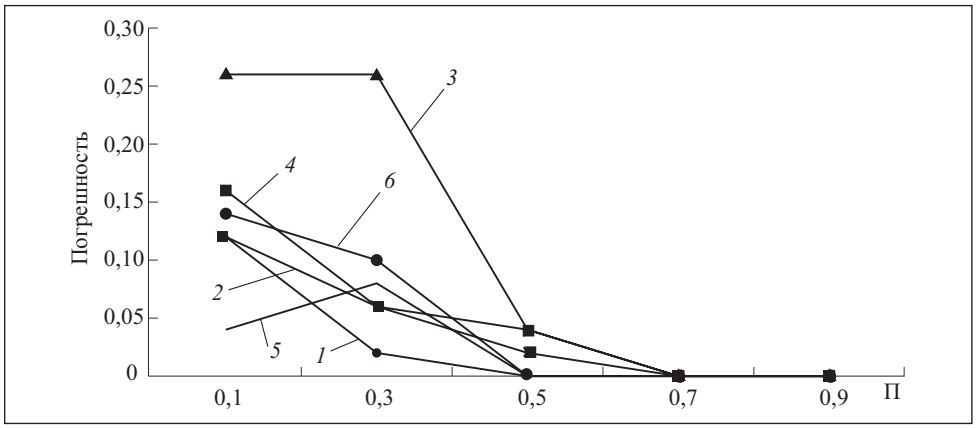

Рис. 8. Зависимость погрешности работы процедуры A_2 от плотности единиц П в матрице B при m = 40: *1* — n = 10; *2* — n = 15; *3* — n = 20; *4* — n = 25; 5 — n = 30; *6* — n = 35

На рис. 4 приведены зависимости величины погрешности работы процедуры $\boldsymbol{A_1}$ от числа ребер и вершин графа, из которых следует, что величина погрешности не превышает в среднем 2,5 %. Проверка на тестовых примерах большой размерности показала, что с увеличением размерности и плотности ребер в графе имеется тенденция к уменьшению величины погрешности.

На рис. 5—8 представлены графики, отображающие работу процедуры $\boldsymbol{A_2}$, решающей ЗНП при числе столбцов n и числе строк m в матрице B. Как видно из рис. 5, 6, временная сложность процедуры $\boldsymbol{A_2}$ в среднем не превышает $O(mn)$. Тестовые примеры решения задач при $n \geq 600$, $m = 400$ показали, что временная сложность в среднем не превышала $O(400mn)$. Анализ погрешности работы процедуры $\boldsymbol{A_2}$ (см. рис. 7, 8) свидетельствует о том, что при плотностях заполнения единицами матрицы B более чем на 50 % погрешность не превышает 5 %.

Выводы

Таким образом, предложенные приближенные алгоритмы решения ЗНВП в произвольных графах и ЗНП на основе сведения их соответственно к задачам квадратичного и нелинейного булевого программирования, позволили построить алгоритмы с временной сложностью, не превышающей $O(mn^2)$, где в случае решения ЗНВП n — это число вершин в графе, m — число ребер в графе, а в случае решения ЗНП n — это число столбцов в матрице B, m — число строк в матрице B. Погрешность решения этих задач процедурами $\boldsymbol{A_1}$ и $\boldsymbol{A_2}$ составляет 2,5—5% при $\Pi \geq 0{,}5$ и их средняя временная сложность составляет $O(n^2)$ для ЗНВП и $O(mn)$ для ЗНП, что очень важно при решении задач распределения ресурсов в современных GRID-системах.

Из рис. 3 и 6 видно, что предложенные алгоритмы могут быть использованы для эффективного решения задач планирования распределения ресурсов [2, 6] в масштабе реального времени, если допустимое время планирования находится в диапазоне от 5 до 100 мс.

The authors propose approximate algorithms for solving the problem of the minimal vertex covering of arbitrary graphs and the problem of minimal coverage on the basis of their reduction, respectively, to the problems of quadratic and nonlinear Boolean programming, their specificity allowing to construct algorithms with time complexity not exceeding $O(mn^2)$, where in the case of solving the problem of minimal vertex covering of arbitrary graphs n is the number of vertices in the graph, m is the number of edges in the graph, and in the case of solving the problem of minimal coverage n is the number of columns in the matrix, m is the number of rows in B. It is shown that this error in the solution of these problems by the proposed procedures $\boldsymbol{A_1}$ and $\boldsymbol{A_2}$ does not exceed 5 % at the density of rows of B matrix 0.5 or more.

The proposed algorithm can be used to effectively planning for the allocation of resources in GRID-systems in real time with a rather severe restrictions on the time of solving problems, if allowed time planning lies in range from 5 to 100 ms.

1. *Кристофидес Н.* Теория графов. Алгоритмический подход. — М. : Мир, 1978. — 309 с.
2. *Понаморенко В. С., Листровой С. В.* Метод решения задачи о минимальном покрытии как средство планирования в GRID // Проблемы управления. — 2008. — № 3. — С. 78—84.
3. *Липский В.* Комбинаторика для программистов. — М. : Мир, 1988. — 203 с.
4. *Шор Н. З., Стеценко С. И.* Квадратичные экстремальные задачи и недифференцируемая оптимизация. — Киев : Наук. думка, 1989. — 196 с.
5. *Пападимитриу К., Стайглиц М.* Комбинаторная оптимизация. Алгоритмы и сложность. — М. : Мир, 1985. — 512 с.
6. *Пономаренко В. С., Листровой С. В., Минухин С. В., Знахур С. В.* Методы и модели планирования ресурсов в GRID-системах.— Харьков: ИД «ИНЖЭК», 2008. — 408 с.
7. *Листровой С. В., Гуль А. Ю.* Метод решения задачи о минимальном покрытии на основе рангового подхода // Электрон. моделирование. — 1999. — **21**, № 1. — С. 58 — 70.
8. *Листровой С. В., Гуль А. Ю., Листровая Е. С.* Точный алгоритм решения задачи о минимальном покрытии // Сб. науч. тр. Информатика. Вып. 5. — Киев : Наук. думка, 1998. — С. 32 — 36.
9. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. — М. : МЦНМО, 2002. — 960 с.
10. *Gandhi R., Khuller S., Srinivasan A.* Approximation Algorithms for partial Covering Problems. // Journal of Algorithms. — 2004. — Vol. 53. — Issue 1. — P. 55—84.
11. *Alon N., Awerbuch B., Azar, Y. et. al.* The online set cover problem // Proc. STOC'03. June 9—11, 2003. — San Diego, California, USA. — P. 100—105.
12. *Chvátal V.* A greedy-heuristic for the set covering problem. // Math. Oper. Res. — 1979. — № 4. — P. 233—235.
13. *Slavik P.* A tight analysis of the greedy algorithm for set cover //Journal of Algorithms. — 1997. — № 25. — P. 237—254.
14. *Hassin R., Levin A.* A Better-than-greedy Approximation Algorithm for the Minimum Set Cover Problem. // SIAM J. Computing.— 2005.— Vol. 35, № 1. — P. 189—200.
15. *Листровой С. В., Минухин С. В.* Общий подход к решению задач оптимизации в распределенных вычислительных системах и теории построения интеллектуальных систем. — //Проблемы управления и информатики. — 2009. — № 2. — С. 47—63.

Поступила 11.07.11;
после доработки 28.10.11

ЛИСТРОВОЙ Сергей Владимирович, д-р техн. наук, профессор, профессор кафедры специализированных компьютерных систем Украинской государственной академии 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.

МИНУХИН Сергей Владимирович, канд. техн. наук, доцент, профессор кафедры информационных систем Харьковского национального экономического университета. В 1976 г. окончил Харьковский ин-т радиоэлектроники. Область научных исследований — оптимизация процессов в распределенных системах, идентификация и управление в сложных системах, управление информационно-вычислительными системами.