РЕАЛИЗАЦИЯ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ О ПОКРЫТИИ МНОЖЕСТВ И АНАЛИЗ ИХ ЭФФЕКТИВНОСТИ

Г. И. Забиняко
Институт вычислительной математики и математической геофизики СО РАН, Новосибирск, Россия
e-mail:zabin@rav.sscc.ru

Consecutive and parallel algorithms for the solution of a set of the covering problems, based on the method of branches and borders are considered. Efficiency analysis for both consecutive and parallel algorithms and the comparison of the results they produce against other known algorithms of branches and borders is presented.

Введение

Задача о покрытии множеств формулируется следующим образом. Пусть $A = \{a_{ij}\}$ — матрица размера $m \times n$ с элементами a_{ij}, равными 0 или 1. Обозначим множество индексов строк через $M = \{1, \ldots, m\}$ и множество индексов столбцов — $N = \{1, \ldots, n\}$. Считается, что столбец j покрывает строку i, если $a_{ij} = 1$. Каждому столбцу j ставится в соответствие положительное число c_j, называемое весом столбца. Требуется найти подмножество $S \subseteq N$, которое покрывает все строки из M и имеет минимальный суммарный вес. Вводя переменные $x_j = 1$, если $j \in S$, и $x_j = 0$ в противном случае, получим постановку задачи целочисленного линейного программирования (ЦЛП):

$$\text{найти} \quad \min z_x = \sum_{j=1}^{n} c_j x_j, \quad A\vec{x} \geq \vec{e}, \quad x_j = 0 \text{ или } 1 \text{ для } j \in N,$$

где $\vec{e}$ — m-мерный вектор, у которого все $e_i = 1$.

Задачи о покрытии относятся к числу NP-трудных. Задача о покрытии множеств на практике возникает при размещении пунктов обслуживания, в системах информационного поиска, в проектировании интегральных схем и конвейерных линий и в других областях [1]. Обзор методов решения приведен в [1, 2], при этом обзор [2] содержит большой объем экспериментального материала.

Нами дальше рассматриваются последовательный и параллельный алгоритмы, основанные на методе ветвей и границ для задач о покрытии, предложенном в [3]. С использованием тестовых задач из библиотеки тестов OR-Library [4] проводится анализ эффективности последовательного и параллельного алгоритмов и сопоставление с результатами по методу ветвей и границ для решения задач о покрытии из [5, 6]. Программы

выполнены на языке FORTRAN, а для распараллеливания используется система параллельного программирования MPI.

1. Последовательный алгоритм

Для решения задач о покрытии разрабатываются методы ветвей и границ, в которых нижние границы z_u (значение целевой функции в двойственных задачах) и верхние z_x (значение целевой функции по исходным переменным для $x_j = 0$ или 1) определяются на основе приближенного решения двойственных задач с помощью субградиентного метода.

Если в исходной постановке ЦЛП требования $x_j = 0$ или 1 для $j \in N$ заменить на условие $x_j \geq 0,\ j \in N$, то двойственная задача будет иметь вид

$$\text{найти максимум } z_u = \sum_{i \in M} u_i, \text{ при условиях } \sum_{i \in I_j} u_i \leq c_j, \quad u_i \geq 0,\ i \in M,$$

где $I_j = \{i \in M \mid a_{ij} = 1\},\ j \in J$.

Для сокращения размерности по исходным и двойственным переменным применяются логические процедуры. Логическое тестирование проводят в каждом узле дерева ветвей и границ [3]. Определим множества $J_i = \{j \in N \mid a_{ij} = 1\}$.

1. Если $|J_i| = 0$ для некоторого $i \in M$, то задача несовместна.

2. Если $|J_i| = 1$ для некоторого $i \in M$, то устанавливаются $x_j = 1$ и соответствующие строки удаляются из рассмотрения, т. е. $M := M \setminus \{i\}$.

3. Пусть существуют $k \in N$ и $j \in N$ такие, что $I_k \subseteq I_j$ и $c_j \leq c_k$, тогда x_k присваивается 0 и индекс k удаляется из N.

4. Для каждого $i \in M$ определим $s_i = \min\limits_{i \in J_i} c_j$. Положим $x_k = 0$ и $N = N \setminus \{k\}$, если $c_k \geq \sum\limits_{i \in I_k} s_i$. Этот тест не используется, если в исходной постановке все c_j равны между собой, например, все $c_j = 1$.

Перед первым обращением к процедурам тестирования в корневом узле проводится упорядочение по возрастанию c_j с учетом заполненности столбцов, т. е. для $c_j = c_k$ вначале следует c_k, если заполненность k-го столбца больше заполненности j-го. Для быстрого просмотра индексов ненулей по столбцам и строкам используются разреженные строчный и столбцовый форматы [7, с. 33]. При этом создаются только массивы индексов, а для элементов a_{ij} нет необходимости отводить память.

Субградиентный метод. Приближенное решение двойственных задач основывается на модификации варианта субградиентного метода, предложеннного в [8]. Пусть $f(\vec{u})$ — выпуклая функция, возможно, негладкая, $f^* = \min\limits_{\vec{u} \in Q} f(\vec{u})$, Q — достаточно просто устроенное допустимое множество, например, параллелипипед, тогда в [8] для итерационного процесса

$$\vec{u}^{k+1} = P_Q\left(\vec{u}^k - \frac{\lambda_k (f^* - f(\vec{u}^k))}{[f'(\vec{u}^k)]^2} f'(\vec{u}^k)\right)$$

установлена геометрическая скорость сходимости для $\epsilon \leq \lambda_k \leq 2 - \epsilon,\ \epsilon > 0$; P_Q — оператор проектирования на допустимое множество; $f'(\vec{u}^k)$ — градиент в точке $\vec{u}^k$ либо субградиент в случае негладких f. В [8] даются также рекомендации, как приспособить

процесс к случаю, когда оптимум по целевой функции f неизвестен. Однако эти рекомендации в большей степени подходят для решения некоторой индивидуальной задачи в диалоговом режиме, а в нашем случае требуется решение последовательности задач.

Для получения двойственных оценок максимизируется функция Лагранжа

$$L(\vec{u}) = \sum_{i \in M} u_i + \sum_{j \in N} \min_{x_j = 0 \text{ или } 1} \left(c_j - \sum_{i \in I_j} u_i \right) x_j$$

с помощью итерационного процесса

$$u_i^{k+1} = \max \left\{ 0, u_i^k + \sigma_k \left(1 - \sum_{j \in J_i} x_j^k \right) \right\},$$

где $\sigma^k = \dfrac{\lambda_k (\hat{z}_x - L(\vec{u}^k))}{(1 - \sum\limits_{j \in J_i} x_j^k)^2}$. Вместо неизвестного оптимального значения целевой функции по двойственным переменным выбирают $\hat{z}_x$ — значение рекорда (минимального значения z_x, полученного к настоящему времени), а начальное значение $\lambda \leq 2$. Если в течение последовательности итераций заданной длины l не происходит увеличения $L(\vec{u}^k)$, то дробится множитель λ_k, а именно $\lambda_{k+1} := \lambda_k / 2$. Для завершения процесса используют разные критерии, например, в [5] процесс завершается, если параметр λ_k становится меньше заданной величины δ, а в [6] аналогично контролируется величина σ_k.

В рассматриваемых нами алгоритмах для завершения субградиентного процесса контролируется сходимость по целевой функции. Проверка на завершение субградиентной оптимизации проводится только после выполнения k очередных итераций. Проверяется сходимость последовательности $L(\vec{u}^{i_1}), \ldots, L(\vec{u}^{i_k})$, при этом для $i_1, \ldots, i_k$ определяются максимальное и минимальное значения $L_{\max}$ и $L_{\min}$, и если величина $\dfrac{L_{\max} - L_{\min}}{L_{\max}}$ меньше заданного значения, выполнение итерационного процесса завершается.

В методе ветвей и границ в качестве нижней границы можно использовать значение $L(\tilde{\vec{u}})$, полученное субградиентной максимизацией. Вектор $\tilde{\vec{u}}$ не является допустимым, так как не все приведенные стоимости $d_j = c_j - \sum\limits_{i \in I_j} \tilde{u}_i$ положительны. С помощью простой процедуры [6] можно за счет уменьшения отдельных $\tilde{u}_i$ получить допустимое решение двойственной задачи $\bar{\vec{u}}$, причем обычно $z_{\bar{u}} = \sum\limits_{i \in M} \bar{u}_i > L(\tilde{\vec{u}})$. В рассматриваемых здесь алгоритмах в качестве нижних оценок используются допустимые решения двойственных задач.

Если начальное значение $\vec{u}^0$ для субградиентной оптимизации не подготовлено на предшествующих шагах алгоритма, то допустимые значения $\vec{u}^0$ задаются следующей формулой: $u_i^0 = \min\limits_{j \in J_i} \dfrac{c_j}{|I_j|}, i \in M$.

Алгоритм определения z_x. Вычисление значения целевой функции по исходным переменным $\vec{x}$ проводится после выполнения процедур логического тестирования и определения двойственных оценок $\vec{u}$.

Переменные $\vec{x}$ подразделяются на фиксированные $x_j = 1$ или $x_k = 0$ и свободные, которые могут принимать допустимые значения 0 или 1. Пусть M^* обозначает набор непокрытых к настоящему времени строк, а S — набор столбцов, для которых уже определены значения x_j. Тогда алгоритм можно представить следующей схемой.

1. Для всех $j \in N \setminus S$ вычислить $\sigma_j = \gamma_i/\mu_j$, где $\gamma_i = c_j - \sum\limits_{i \in I_j \cap M^*} u_i$, а $\mu_j = |I_j \cap M^*|$. Здесь и далее $|J|$ — число элементов в множестве J.

2. Среди σ_j определить минимальное значение σ_{j^*}. Если индекс j^* определяется не единственным образом, то среди j^* выбирается индекс, которому соответствует минимальное значение c_{j^*}/μ_{j^*}.

3. Для всех $i \in I_{j^*} \cap M^*$ и всех $j \in J_i$ модифицировать $\gamma_i = \gamma_i + u_i,\ \mu_j = \mu_j - 1$.

4. Положить $S = S \cup \{j^*\},\ M^* = M^* \setminus I_{j^*}$. Проверить, если $M^* \neq 0$, то перейти к 1, иначе остановиться.

Дальше рассмотрим действия, выполняемые в корневом узле. В самом начале обращаемся к процедурам логического тестирования. Пусть M^* — набор непокрытых строк, а S — набор фиксированных столбцов после выполнения этих процедур. Определим $u_i^0 = \min \dfrac{c_j}{|I_j|},\ i \in M^*,\ j \in N \setminus S$. Исходя из $\vec{u}^0$, определим $\vec{x}$ и z_x. Используя z_x, обратимся к процедуре субградиентной максимизации со значением параметра $\lambda_0 = 2$ для определения двойственных оценок $\vec{u}$ и z_u. На основе новых оценок $\vec{u}$ определим $\vec{x}$ и z_x. Если оказывается, что $z_x \leq z_u$, то задача решена. В противном случае делается попытка исключить некоторые переменные x_j, для которых $z_x - z_u < d_j$. Если эта попытка оказывается удачной, в корневом узле повторяются вычисления с вызова процедуры субградиентной максимизации с начальным значением $\lambda_0 = 0.1$ и поиска $\vec{x}$.

Если в корневом узле не удается найти оптимального решения, то обращаемся к процедуре ветвления.

Процедура ветвления. При первом обращении к процедуре ветвления в качестве входных данных используется вектор двойственных оценок $\vec{u}$, полученный в корневом узле. Для ветвления выбирается строка i, соответствующая максимальной компоненте u_i. Элементы $j \in J_i$ упорядочиваются по возрастанию приведенной стоимости d_{j_i} для $j_1, \ldots, j_{|J_i|}$. Упорядоченный список номеров переменных записывается в целочисленный массив List_j и используется для порождения узлов для каждого $j \in J_i$. В k-м, созданном согласно упорядочению, узле полагаются $x_{j_k} = 1$ и $x_{j_1}, \ldots, x_{j_{k-1}} = 0$, а переменные $x_{j_{k+1}}, \ldots, x_{j_{|J_i|}}$ — свободные.

Кроме массива List_j, для описания дерева решений вводится главный массив дерева решений — целочисленный массив, в котором для описания каждого узла отводится восемь элементов. В этом массиве запоминается строка ветвления для данного шага ветвления. (Шаг ветвления отвечает списку переменных, полученных при одном обращении к процедуре ветвления.) Далее запоминаются следующие величины: номер переменной по порядку в списке List_j; начальный и конечный адреса в списке для данного шага ветвления; номер предшествующего узла и др. Кроме того, для каждого узла в вещественном массиве сохраняются значения z_u, z_x, соответствующие данному узлу.

После анализа всех узлов шага ветвления производится поиск открытого узла (узел открытый, если $\tilde{z}_x > z_u$ и узел еще не использовался для порождения новых узлов) с минимальным значением z_x. Предположим, что найден необходимый открытый узел, в котором в качестве строки ветвления использовалась строка i. С использованием перечисленных выше массивов восстанавливаются фиксированные и свободные переменные, вычисляются оценки двойственных переменных $\vec{u}$, соответствующие данному узлу. Узел полагается закрытым, производится обращение к процедуре ветвления. В качестве строки ветвления выбирается строка k, которой отвечает максимальная компонента u_k

и $k \neq i$. Список List_j дополняется упорядоченным списком номеров переменных данного шага ветвления. Процесс продолжается, пока не будут исчерпаны все открытые узлы.

2. Параллельный алгоритм

В параллельном режиме среди процессорных элементов выделяется нулевой процессор. Данные о задаче считываются этим процессором с дисковой памяти и передаются всем остальным процессорам. Далее все вычисления до получения решения производятся в асинхронном режиме. После получения решения для межпроцессорных обменов используются блокирующие функции. Вначале на всех процессорах выполняются операции, предусмотренные для корневого узла.

После выполнения первого шага ветвления на каждом из процессоров выбирается подмножество узлов из упорядоченного списка, которые обрабатываются на данном процессоре. На k-м процессоре выбираются из списка номера по правилу: $j_1 = k,\ j_{i+1} = j_i +$ nump для $j_{i+1} \leq l$, где l — общее число элементов в списке, а nump — количество используемых процессоров. Если для некоторых процессоров $k > l$, то эти процессоры посылают сообщение об ожидании загрузки на нулевой процессор. Это правило используется только для первого шага ветвления.

Для того чтобы произвести загрузку, нулевой процессор опрашивает задействованные процессоры о количестве открытых узлов. Для определения процессора, с которого будет производиться загрузка, данные о количестве открытых узлов на процессорных элементах упорядочиваются по убыванию. Если на каком-то из процессоров имеется более чем K открытых узлов, то нулевой процессор посылает ему команду на загрузку простаивающего процессора. На процессоре, с которого производится загрузка, определяется $K_1 = K/3$ для передачи простаивающему процессору массивов, описывающих первые по порядку K_1 открытых узлов. При этом передаются массивы: список переменных List_j, сформированный на шагах ветвления; главный массив, описывающий дерево решений; вещественный массив значений $z_u,\ z_x$. На передающем процессоре помечаются первые K_1 открытых узлов как закрытые, а на принимающем процессоре, после приема данных, решение продолжается с поиска порождающего узла.

При получении на каком-либо из процессоров нового рекорда $\hat{z}_x$ значение $\hat{z}_x$ передается всем остальным процессорам. Сообщение о рекорде имеет более высокий приоритет, нежели другие сообщения. Поступление сообщений контролируется после решения всех задач шага ветвления, а для рекорда такая проверка делается после решения каждой задачи. Процесс решения завершается, когда на всех процессорах нет открытых узлов. Решение x передается на нулевой процессор в блокирующем режиме, кроме того, на нулевой процессор передаются с каждого процессора статистические данные о количестве обращений к субградиентной максимизации, о количестве проанализированных узлов.

3. Решение тестовых задач

Все задачи решаются при одном и том же выборе управляющих параметров. Пороговое значение параметра ϵ, используемое для завершения субградиентной максимизации $\left(\dfrac{L_{\max} - L_{\min}}{L_{\max}} \leq \epsilon\right)$, принималось в корневом узле равным 10^{-5}, а в остальных узлах оно

составляло 10^{-4}. Перед обращением к процедуре ветвления (кроме первого обращения, в котором используются оценки $\vec{u}$, полученные в корневом узле) с помощью жадного алгоритма определялось начальное $\vec{u}^0$ для субградиентной максимизации. При решении задач, соответствующих одному шагу ветвления, в качестве начального $\vec{u}^0$ принимались скорректированные оценки $\vec{u}$ из предыдущей задачи. При этом компоненты u_i полагались равными 0, если соответствующая i-я строка оказалась покрытой.

Для численных экспериментов использовались те же тестовые задачи, что и в [5, 6]. Это задачи из OR-библиотеки, которые получены с помощью генератора псевдослучайных чисел. Задачи различаются размерностью и заполненностью матриц ограничений, но во всех задачах в качестве значений весов c_j выбираются целые числа из [1,100].

Рассматриваемые нами алгоритмы наиболее близки к алгоритму [3]. К сожалению, в [3] рассматривается решение простых наборов задач, поименованных в OR-библиотеке целыми числами 3, 4, 5 и 6. Рассмотрим результаты решения наиболее трудного набора 6. Набор 6 состоит из пяти задач: 6.1, 6.2, 6.3, 6.4 и 6.5. В них $m = 200,\ n = 1000$, заполненность матриц ненулевыми элементами составляет 5 %. Результаты решения сведены в табл. 1, при этом здесь и далее в таблицах k_n обозначает число исследованных узлов (найдены значения целевых функций по двойственным и исходным переменным), z_{x_0} — значение целевой функции по исходным переменным, полученное в корневом узле. Рассматриваемый нами алгоритм обозначим в таблицах через Р. Во второй и в третьей колонках таблицы приведены значения $z^*_{u_0}, z^*_x$, — соответственно оптимум целевой функции по двойственным переменным в корневом узле, оптимум целочисленного решения по исходным переменным, полученные пакетом ЦЛП [9]. В пакете ЦЛП оценочные задачи линейного программирования решаются симплекс-методом. В больших задачах о покрытии множеств базисные решения сильно вырождены, что затрудняет применение симплекс-метода. Таких трудностей не возникает при решении задач из набора 6. Результаты из [6] даются в двух вариантах расчетов, различающихся выбором числа итерации, после выполнения которых возможно дробление параметра λ_k в процедуре субградиентной максимизации. В числителе данные для $l = 7$, а в знаменателе — для $l = 30$. В нашем случае $l = 14$. Время в секундах указано для PC CELERON 850.

Рассмотрим основные различия нашего алгоритма и алгоритма из [3]. В нашем случае делаются дополнительные вычисления в корневом узле, чтобы исключить большее число переменных. Например, в задаче 6.4 после логического тестирования из 1000 переменных остаются 224 переменные, а за счет дополнительных вычислений, на основе анализа приведенных стоимостей, исключаются еще 122 переменные. Кроме исключения переменных, дополнительные вычисления позволяют улучшить двойственные оценки, получаемые в корневом узле.

В [3] начальные приближения $\vec{u}$ вычисляются с помощью "жадного" алгоритма, полу-

Таблица 1.

Имя	ЦЛП		Алгоритм [3]		Алгоритм Р				Алгоритм [5]		Алгоритм [6]	
	$z^*_{u_0}$	z^*_x	z_{u_0}	k_n	z_{u_0}	k_n	z_{x_0}	t, c	k_n	z_{x_0}	k_n	z_{x_0}
6.1	133.14	138	132.13	439	132.79	41	145	11	195	141	77/15	142/140
6.2	140.46	146	140.14	451	140.16	86	153	21	171	146	45/23	156/147
6.3	140.13	145	138.96	136	139.23	56	148	12	15	145	7/3	145/145
6.4	129.00	131	128.76	36	128.85	11	132	2	3	131	4/1	132/131
6.5	153.35	161	152.43	369	152.38	93	168	21	271	162	53/20	170/163

ченное $\vec{u}$ уточняется 3-opt процедурой. В процедуре рассматриваются тройки u_{i1}, u_{i2}, u_{i3} и делается попытка за счет уменьшения одной из переменных увеличить на ту же положительную величину две другие. Затем применяется не более 100 итераций субградиентной максимизации.

В алгоритме ветвления из [3] строка ветвления выбирается из максимума $u_i \left(\sum\limits_{j \in J_i} x_j - 1 \right)$, где x_j — компоненты исходных переменных, отвечающие рекорду. В нашем случае строка i соответствует максимальной компоненте u_i. Использование информации о рекорде может приводить к выбору строки i с малым значением компоненты u_i, кроме того, использование рекорда в процедуре ветвления затрудняет процесс распараллеливания.

При определении решения по исходным переменным в [3] отдается предпочтение тем непокрытым строкам, для которых достигается максимум $u_i|J_i|$. В нашем же случае на каждом шаге просматриваются все непокрытые строки, и алгоритм определения z_x в точности такой, как в [10].

Далее для численных экспериментов использовалась кластерная система НКС—160, вычислительные модули которой состоят из двух процессоров Intel Itanium 2 с тактовой частотой 1.6 ГГц. Вычислительные модули связаны между собой при помощи коммутатора Infini Band с производительностью 10 Гбит/с, который обеспечивает обмен данными между параллельными процессами.

Таблица 2.

Имя	z_x^*	$z_{u_0}^*$	Алгоритм [5]		Алгоритм [6]		Алгоритм P				CPLEX	MINTO
			k_n	z_{x_0}	k_n	z_{x_0}	k_n	k_{opt}	z_{x_0}	t, c	k_n	k_n
A.1	253	246.8	515	255	95/74	261/258	107	13	258	7	44	55
A.2	252	247.5	809	256	28/25	257/254	426	117	257	24	18	85
A.3	232	228.0	857	234	41/29	243/237	398	344	239	23	37	71
A.4	234	231.4	63	235	10/5	241/235	22	8	237	1	10	13
A.5	236	234.9	79	237	3/1	237/236	135	133	245	8	10	13
B.1	69	64.5	411	70	81/24	72/69	132	86	70	19	40	67
B.2	76	69.3	3273	77	939/483	78/76	358	23	77	94	173	301
B.3	80	74.2	1383	80	288/181	81/81	519	162	85	117	66	261
B.4	79	71.2	4381	80	1508/974	83/79	978	175	83	238	301	1403
B.5	72	67.7	661	72	69/47	75/72	172	44	73	29	56	93
C.1	227	223.8	1265	230	34/22	235/230	616	549	230	34	38	39
C.2	219	212.8	275	223	145/104	224/220	412	352	223	42	61	91
C.3	243	234.6	15757	251	291/458	249/248	3775	718	254	384	129	489
C.4	219	213.8	493	224	116/55	233/224	377	377	229	33	38	27
C.5	215	211.6	375	217	91/38	219/217	107	57	220	11	13	47
D.1	60	55.3	3393	61	337/328	64/61	253	143	61	122	210	324
D.2	66	59.3	6739	68	2771/1498	70/67	967	176	67	487	575	2791
D.3	72	65.1	19707	75	5120/2219	76/74	2017	390	74	774	324	2055
D.4	62	55.8	12389	64	2931/1631	67/63	669	305	65	288	315	1957
D.5	61	58.6	147	62	18/14	61/61	32	25	62	9	43	13

Примечание. Заполненность матриц ненулевыми элементами в задачах наборов A, C составляет 2 %, а в задачах наборов B и D — 5 %.

Таблица 3.

Имя	$z^*_{u_0}$	Последовательный алгоритм						Параллельный алгоритм			
		z_{u_0}	$\tilde{z}_x$	k_s	k_n	k_{opt}	t_1, c	k_s	k_n	t_{10}, c	t_1/t_{10}
E.1	21.4	21.2	29	254186	36809	2	63000	254193	36810	7114	8.9
E.2	22.4	22.1	30	493746	75533	12982	145986	648328	107915	19501	7.5
E.3	20.15	20.3	27	68796	10250	952	21882	68938	11186	3131	7.0
E.4	21.4	21.2	28	249012	37260	3198	68516	287999	45457	8774	7.8
E.5	21.3	21.1	28	72572	10618	0	21736	72586	10618	2862	7.6
F.1	8.8	8.6	14	49660	6775	0	21160	44427	6009	3023	7.0
F.2	10.0	9.6	15	29073	4002	0	13798	26248	3676	1703	8.1
F.3	9.5	9.0	14	8456	1152	79	5688	9534	1463	932	6.2
F.4	8.5	8.3	14	103874	14077	696	39416	107311	14683	5496	7.2
F.5	7.8	7.6	13	113074	16629	4356	41645	108789	17852	6009	6.9

Примечание. В задачах $m = 500,\ n = 5000$. Заполненность ненулевыми элементами в задачах набора E составляет 10 %, а в задачах набора F — 20 %.

В табл. 2, в колонке z^*_x, приводится оптимальное значение целевой функции по исходным переменным, а в колонке $z^*_{u_0}$ — оптимальное значение по двойственным переменным в корневом узле. Для наборов задач A, B, C и D, используемых далее, известны оптимальные решения. Значения величин z^*_x и $z^*_{u_0}$ взяты из обзора [2]. В табл. 2 используются те же обозначения, что и в табл. 1. Дополнительно для результатов по нашему алгоритму приводится номер узла по порядку k_{opt}, в котором было найдено оптимальное решение. В табл. 2 приводится также количество исследованных узлов при решении задач с помощью ЛП программ CPLEX и MINTO [2]. В задачах наборов A, B $m \times n = 300 \times 3000$, а в C и D — $m \times n = 400 \times 4000$.

Сравнение величин k_n и k_{opt} позволяет сделать вывод, что в нашем случае основные затраты алгоритма приходятся на обоснование оптимальности.

В табл. 3 содержатся данные для сравнения алгоритмов в последовательном и параллельном режимах. Численные эксперименты приведены с использованием наборов E и F из OR-библиотеки. Поскольку в этих задачах не найдены строго обоснованные оптимальные решения, в [2] приводятся минимальные из полученных решений. В колонке $z^*_{u_0}$ даны оптимальные значения целевой функции по двойственным переменным в корневом узле, которые получены с помощью линейного программирования [2]. В табл. 3 дается величина k_s — число обращений к субградиентной максимизации. В параллельном режиме использовалось 10 процессорных элементов.

Из табл. 3 видно, что $k_n \gg k_{\text{opt}}$ и число обращений к субградиентной максимизации может в несколько раз превосходить k_n. Полученные у нас значения $\tilde{z}_x$ совпадают с приведенными в [2].

Заключение

Разработаны последовательный и параллельный алгоритмы для решения задач о покрытии множеств. Последовательный алгоритм позволяет эффективно решать задачи

небольшой и средней размерностей. Для больших задач с высокой заполненностью матриц целесообразно использовать параллельный алгоритм.

Автор выражает благодарность Ю.А. Кочетову за полезные обсуждения.

Список литературы

[1] Еремеев А.В., Заозерская Л.А., Колоколов А.А. Задача о покрытии множества: сложность, алгоритмы, экспериментальные исследования // Дискрет. анализ и исслед. операций. Сер. 2. 2000. Т. 7, № 2. С. 22–46.

[2] Caprara A., Toth P., Fischetti M. Algorithms for the set covering problem // Anals of Oper. Res. 2000. Vol. 98. P. 353–371.

[3] Fisher M.L., Kedia P. Optimal solutions of set covering/partitioning problems using dual heuristics // Management Sci. 1990. Vol. 36. P. 674–688.

[4] Beasley J.E. OR-Library: distributing test problems by electronic mail // J. Oper. Res. Soc. 1990. Vol. 41. P. 1069–1072.

[5] Beasley J.E. An algorithm for set covering problems // European J. Oper. Res. 1987. Vol. 31. P. 85–93.

[6] Balas E., Carrera M.C. A dynamic subgradient-based branch-and-bound procedure for set covering // Oper. Res. 1996. Vol. 44, N 6. P. 875–890.

[7] Писсанецки С. Технология разложенных матриц. Пер. с англ. М.: Мир. 1988.

[8] Поляк Б.Т. Минимизация негладких функционалов // Журн. вычисл. мат. и математической физ. 1969. Т. 9, № 3. С. 509–521.

[9] Забиняко Г.И. Пакет программ целочисленного линейного программирования // Дискрет. анализ и исслед. операций. Сер. 2. 1999. Т. 6, № 2. С. 32–41.

[10] Caprara A., Fishetti M., Toth P. A heuristic method for the set covering problem // Oper. Res. 1999. Vol. 47, № 5. P. 730–743.

Поступила в редакцию 7 февраля 2007 г.,
в переработанном виде — 1 августа 2007 г.