МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ОПТИМАЛЬНОЕ УПРАВЛЕНИЕ

УДК 519.874

ЗАДАЧА РАСПАРАЛЛЕЛИВАНИЯ АЦИКЛИЧЕСКОГО АЛГОРИТМА

© 2008 г. ***В.В. Слободской***

Нижегородский госуниверситет им. Н.И. Лобачевского

slvital@gmail.com

Поступила в редакцию 26.06.2008

Рассматривается задача распараллеливания алгоритма, не содержащего циклов и ветвлений. Строится общая математическая модель, проводится ее исследование, ставится оптимизационная задача, предлагаются алгоритмы ее решения, ставится эксперимент.

Ключевые слова: распараллеливание, алгоритм, ветвление.

Введение

В настоящее время все большую роль играет не столько написание быстрой реализации некоторого алгоритма, сколько возможность сделать эту реализацию наиболее масштабируемой на многопроцессорной вычислительной системе. В работе рассматривается возможность распараллеливания алгоритмов, заданных в виде канонической сети взаимозависимых операций, выполнение которых может осуществляться на различных процессорах многопроцессорной вычислительной системы.

Будем рассматривать алгоритмы, представимые в виде канонической сети взаимозависимых операций. Каноничность сети означает то, что никакая операция не может быть выполнена до тех пор, пока не завершены все ей предшествующие операции. Вычислительная система – это набор вычислительных элементов (процессоров), обладающих различными скоростями выполнения операций. Предполагается, что известной является матрица скоростей передачи данных от одного процессора другому. Мы будем рассматривать случай, когда каждая операция может выполняться на любом процессоре.

Примером алгоритма, не содержащего циклов и ветвлений, может являться любой алгоритм, заданный в виде рекуррентных соотношений, – например, рекуррентные соотношения динамического программирования. Примером ациклического алгоритма является вычислительная схема расчета временных характеристик сетевой модели.

Область исследований распараллеливания задач на вычислительных системах, состоящих из нескольких вычислительных элементов с разными скоростями, с помощью решения оптимизационных задач, разбивается на две части – распараллеливание независимых операций и распараллеливание операций, зависимости между которыми представимы в виде канонической сети. Случай независимых операций достаточно хорошо исследован. В [1] и [2] предлагаются приближенные алгоритмы решения подобной задачи с приводимыми оценками погрешности. Работа [3] предлагает 2-приближенный алгоритм ее решения, также приводится и доказывается теорема о том, что для подобной задачи не существует полиномиального ε-приближенного алгоритма с $\varepsilon < 3/2$. В [4] приводятся приближенные алгоритмы для задач как с независимыми, так и с зависимыми операциями. В работе [5] приводится аналог фронтального алгоритма для задачи с зависимыми операциями. В [6] рассматривается оптимальный алгоритм для решения такой задачи при условии малого количества операций.

Цель данной работы – развитие исследования задач с зависимыми операциями, для которых до сих пор нет действительно эффективных приближенных алгоритмов с оценками погрешности.

Содержательное описание задачи распараллеливания

Имеется вычислительная система, состоящая из n вычислительных элементов (процессоров),

объединенных в сеть таким образом, что имеется возможность передачи данных от каждого вычислительного элемента любому другому. Скорость передачи данных между процессорами задается в виде матрицы. Каждый процессор имеет скорость вычисления операции. Считается, что время выполнения любой операции на процессоре обратно пропорционально скорости вычисления процессора. Скорость самого медленного процессора принимается равной единице.

Задан алгоритм, представляющий собой каноническую сеть взаимозависимых операций, где каждая операция (кроме начальных) имеет набор предшествующих операций и каждая операция (кроме завершающих) имеет набор последующих операций. Операция не может выполняться, пока не завершены все предшествующие ей операции. Для каждой операции известно ее время выполнения на самом медленном процессоре. Для любой операции и любой последующей операции задана величина, определяющая количество данных, которые необходимо передать для выполнения последующей операции. Операция на процессоре не может начинаться, пока на этот процессор не переданы данные со всех операций, предшествующих этой операции. Передача данных может начинаться только после завершения операции. Считается, что данные на процессор можно передавать независимо от его работы, т.е. параллельно с текущими вычислениями, не замедляя при этом его работу. Также считается, что выполнение более одной операции на процессоре является невозможным, поэтому каждый процессор в одно и то же время может обрабатывать лишь одну операцию.

Необходимо таким образом назначить операции на процессоры, чтобы общее время завершения обработки всех операций было минимальным.

Общая математическая модель

Исходные параметры математической модели. Пусть J – множество всех операций, которые необходимо выполнить, I – множество процессоров вычислительной системы.

Обозначим через $K(j)$ множество операций, непосредственно предшествующих операции с номером j, $K(j) \subset J$, $j \in J$; t_j – время выполнения операции j на процессоре со скоростью 1, $t_j > 0$, $j \in J$; p_i – скорость процессора i, $p_i > 0$, $i \in I$. Под скоростью процессора здесь будем понимать величину, определяющую, во сколько раз действительная скорость процессора больше действительной скорости самого медленного процессора.

Пусть $\|M\|$ – матрица размерности $|J|\times|J|$, элемент которой m_{kl} определяет количество данных, которые должны быть переданы после выполнения операции k для выполнения операции l, $k \in K(l)$, $k,l \in J$, $m_{kl} = 0$, если $k \notin K(l)$; $\|S\|$ – матрица размерности $|I|\times|I|$, элемент которой s_{uv} определяет скорость передачи данных от процессора u к процессору v, $s_{uv} > 0$, $u,v \in I$. Здесь предполагается, что скорость передачи данных с одного процессора на самого себя определяется очень большим числом.

Обозначим через $Q = \{(j,k)|$, где j – непредшествующая операция для k, k – непредшествующая операция для $j\}$; J^0 – множество начальных операций (не имеющих предшествующих), $J^0 \subseteq J$; J^D – множество завершающих операций, $J^D \subseteq J$.

Варьируемые параметры математической модели. В качестве неизвестных математической модели определим $|J|$-мерный вектор $\vec{x}$, компонента которого x_j определяет время начала выполнения операции j, $j \in J$, и булеву матрицу $\|Y\|$ размерности $|J|\times|J|$, элемент которой $y_{ji} = 1$, если операция j выполняется на i-м процессоре, $j \in J$, $i \in I$; 0 – в противном случае.

Ограничения математической модели.

$$x_k \geq \max_{j \in K(k)} \left(x_j + t_j \sum_{l \in I} \frac{y_{jl}}{p_l} + m_{jk} \sum_{i \in I} \sum_{l \in I} \frac{y_{ji} y_{kl}}{s_{il}} \right), \quad k \in J. \tag{1}$$

Начало выполнения операции j возможно лишь после завершения выполнения и передачи необходимых данных всех непосредственно предшествующих ей операций.

$$x_k \geq x_j + t_j \sum_{l \in I} \frac{y_{jl}}{p_l} \text{ или}$$

$$x_j \geq x_k + t_k \sum_{l \in I} \frac{y_{kl}}{p_l}, \text{ если } (j,k) \in Q, y_{jl} = y_{kl}, \quad \text{для всех } l \in I; k, j \in J. \tag{2}$$

Одновременно на одном и том же процессоре не может выполняться более одной операции.

$$\sum_{l \in I} y_{jl} = 1, \; j \in J. \tag{3}$$

Каждая операция должна выполняться на одном процессоре.

$$x_j \geq 0,\ j \in J^0. \qquad (4)$$

Ограничения для начальных операций.

$$y_{ji} \in \{0,1\},\ j \in J, i \in I, \quad x_j \geq 0,\ j \in J. \qquad (5)$$

Естественные ограничения на варьируемые переменные.

Задача минимизации общего времени выполнения операций. Требуется найти такое решение системы ограничений (1)–(5), на котором принимает минимальное значение критерий, определяющий общее время выполнения операций:

$$F\ (\vec{x}, Y) = \max_{j \in J^D} \left(x_j + t_j \sum_{l \in I} \frac{y_{jl}}{p_l} \right). \qquad (6)$$

Методы решения задачи распараллеливания

Дискретность переменных и существенная нелинейность ограничений (2) в случае большого числа переменных и ограничений, соответствующих реальным вычислительным задачам, не позволяют для решения поставленной задачи использовать точные методы. В работе для этого предлагается эвристический алгоритм фронтального типа.

Фронтальный алгоритм. Фронтальный алгоритм строит решения оптимизационных задач, используя идеологию «жадных алгоритмов» (аналог метода градиента для непрерывных задач): выбранная из фронта операций очередная операция включается в строящееся решение и исключается из дальнейшего рассмотрения (схема отсутствия обратных связей).

Для произвольного момента времени t обозначим через

$$F(t) = ((J^0 \cap J^t) \cup N(t)) \cap A(t)$$

фронт операций на момент времени t – множество тех операций, которые еще не выполнены и могут начать выполняться в момент времени t с учетом всех ограничений задачи. Здесь J^t – множество операций, на момент времени t еще не включенных в строящееся решение, $J^t \subseteq J$, $N(t)$ – множество еще не включенных к моменту времени t в строящееся решение операций, для любой из которых к моменту времени t все ей предшествующие операции уже включены в строящееся решение и время их завершения, а также передачи данных, установлено не позже момента t:

$$N(t) = \{k \mid \max_{j \in K(k) \setminus J^t} \left(\tau_j + \frac{t_j}{p_{e(j)}} + \frac{m_{jk}}{s_{e(j)e(k)}} \right) \leq t,$$
$$k \in J^t \setminus J^0\},$$

где τ_j – момент начала выполнения уже включенной в строящееся решение операции j, $j \in J \setminus J^t$, $e(j)$ – процессор, на котором выполняется уже включенная в решение операция j, $e(j) \in I$, $j \in J \setminus J^t$.

$A(t)$ – множество тех операций, для выполнения которых в момент времени t существует свободный процессор:

$$A(t) = \{k \mid k \in J, \text{существует } i \in I,$$
$$\text{что для всех } s \in J \setminus J^t, \text{таких что } e(s) = i,$$
$$\text{выполняется } [\tau_s; \tau_s + \frac{t_s}{p_i}] \cap [t; t + \frac{t_k}{p_i}] = \emptyset\}.$$

Пусть t – текущий такт планирования, а $F(t) = \{j_1, j_2, ..., j_k\}$ – фронт операций на момент времени t. Тогда обозначим через $\vec{\pi}(t) = (\pi_1, \pi_2, ..., \pi_k)$, $\pi_l \neq \pi_q$, $\pi_l, \pi_q \in F(t), l = \overline{1,k}$, $q = \overline{1,k}$ – вектор очередности операций из фронта $F(t)$, где π_q – номер операции, рассматриваемой при построении решения q-й по порядку. Операция π_1 включается в строящееся решение на тот из свободных в момент времени t процессор, время окончания операции π_1 на котором будет минимальным. Операция π_1 из вектора $\vec{\pi}(t)$ исключается. Процесс повторяется до тех пор, пока не будут исключены все операции из вектора $\vec{\pi}(t)$ или не останется свободных процессоров к моменту времени t. Затем рассматривается фронт операций на момент времени t', $t' > t$, – момент времени, к которому завершится выполнение любой из операций, включенных в строящееся решение. Процесс повторяется, пока все операции из множества J не будут включены в строящееся решение.

Упорядочение операций при построении вектора $\vec{\pi}(t)$ может быть осуществлено по различным схемам, однако с учетом специфики рассматриваемой задачи наиболее эффективным видится использование стратегии упорядочивания операций по убыванию суммарного времени выполнения последующих операций: $U(j)$ – множество всех последующих для j операций, $T(j) = \sum_{l \in U(j)} t_l$ – суммарное время выполнения всех последующих для j операций.

Применимость этой стратегии (схемы сравнения) для поставленной задачи объясняется

построением как можно более сбалансированного в смысле использования вычислительных ресурсов решения – операции, имеющие большее суммарное время выполнения последующих операций, назначаются на наиболее быстрые процессоры, те же, что имеют меньшее время выполнения последующих операций, назначаются на более медленные процессоры.

Будем рассматривать две модификации фронтального алгоритма, основанные на использовании схемы сравнения: *Фронт/А* – фронтальный алгоритм без схемы сравнения, *Фронт/Б* – фронтальный алгоритм со схемой сравнения.

Метод ветвей и границ

Различают четыре основные вычислительные процедуры метода ветвей и границ, которые делятся на универсальные (общие для любых решаемых задач) и индивидуальные (зависящие от специфики задачи). К универсальным процедурам относятся процедура отсева (отбрасывания неперспективных направлений) и процедура останова (определение оптимальности найденного решения). К индивидуальным процедурам относятся процедура оценок (в общем случае нахождение верхней и нижней оценок) и процедура ветвления.

Рассмотрим индивидуальные процедуры для задачи распараллеливания.

Процедура ветвления для рассматриваемой задачи заключается в построении дерева ветвления по следующей схеме. В качестве корневого узла дерева ветвления рассматривается решение, в основе которого нет ни одной включенной операции. К этому узлу добавляются дочерние узлы в количестве всех возможных комбинаций назначения начальных операций на каждый процессор. Далее к каждому узлу дерева ветвления добавляются дочерние узлы в количестве всех возможных комбинаций назначения всех последующих операций для операций, уже назначенных для этого узла, на каждый процессор. Процесс повторяется до тех пор, пока не будут достигнуты листовые узлы дерева, в которых уже назначены все операции задачи.

Для формализации процедуры оценок введем дополнительные обозначения. Пусть U – множество вершин дерева ветвления, $S(u)$ – множество операций, еще не включенных в решение для вершины u, $u \in U$, $M(u)$ – множество операций, для которых включены в решение все предшествующие операции $M(u) \subseteq S(u)$, $u \in U$.

В качестве верхней (достижимой) оценки значения критерия оптимальности принимается значение критерия на решении задачи фронтальным алгоритмом. Для определения нижней оценки рассмотрим релаксированную задачу R с ограничениями (1), (3)–(5) и критерием (6). При такой постановке предполагается, что на тот же самый процессор можно ставить одновременно несколько операций, тем самым можно перейти к задаче с одним процессором $w_{\max}$ – процессором с максимальной скоростью. Тогда оптимальное решение задачи R находится с помощью соотношений

$$x_j^0 = \max_{k \in K(j)} \left(x_k^0 + \frac{t_k}{p_{w_{\max}}} + \frac{m_{jk}}{s_{w_{\max} w_{\max}}}\right), \quad j \in J,$$

$$y^0{}_{ji} = \begin{cases} 1, & i = w_{\max} \\ 0 - \text{иначе} \end{cases}, i \in I, j \in J,$$ и значение нижней оценки определяется как $H_1 = F\ (\vec{x}^0, Y^0)$.

Замечание. Так как скорость передачи данных с одного процессора на самого себя определяется большим числом, величиной $\frac{m_{jk}}{s_{w_{\max} w_{\max}}}$ в оптимальном решении задачи R можно пренебречь.

Результаты вычислительного эксперимента показали, что нижняя оценка, вычисляемая подобным образом, позволяет производить отсев неперспективных направлений только на окололистовых узлах дерева метода ветвей и границ, т.е. на этапе завершения работы алгоритма. Учитывая то, что каждая операция может выполняться на каждом процессоре, а также то, что процессоры могут иметь различную скорость, можно предложить более эффективную нижнюю оценку для рассматриваемой задачи:

$$H = \frac{\sum_{j \in J} t_j}{\sum_{i \in I} p_i}. \qquad (7)$$

Действительно, предположим от противного, что существует решение задачи (1)–(6) $(\vec{x}', Y')$, для которого $F(\vec{x}', Y') < \sum_{j \in J} t_j \Big/ \sum_{i \in I} p_i$. При этом, согласно (6),

$$F(\vec{x}', Y') = \max_{j \in J^D} \left(x'_j + t_j \sum_{l \in I} \frac{y'_{jl}}{p_l} \right) \geq$$

$$\geq \max_{i \in I} \left(\frac{\sum_{j \in J} t_j y'_{ji}}{p_i} \right),$$

отсюда следует, что $\max_{i\in I}\left(\sum_{j\in J} t_j y'_{ji} \Big/ p_i\right) < \sum_{j\in J} t_j \Big/ \sum_{i\in I} p_i$ и $\sum_{j\in J} t_j y'_{jl} \Big/ p_l < \sum_{j\in J} t_j \Big/ \sum_{i\in I} p_i$, $l \in I$. Тогда $\sum_{j\in J} t_j y'_{jl} < \dfrac{p_l \sum_{j\in J} t_j}{\sum_{i\in I} p_i}, l \in I$. Просуммировав неравенства и учитывая (3), получим противоречие: $\sum_{j\in J} t_j < \sum_{j\in J} t_j$, которое доказывает условие (7).

В процессе вычислений оценку (7) можно улучшать, учитывая те простои процессоров, которые уже невозможно занять выполнением операций. Тогда

$$H_2 = \frac{\sum_{j\in J} t_j + t^n(u)}{\sum_{i\in I} p_i},\ u \in U, \qquad (8)$$

где $t^n(u)$ – суммарное количество простоев процессоров на интервале от начала планирования до минимального времени начала выполнения операций из множества операций, для которых все предшествующие операции уже назначены на процессоры в частичном решении, заданном узлом u.

Нетрудно показать, что неравенство (8) является нижней оценкой для частично построенного решения, заданного узлом $u \in U$.

Действительно, считая каждый простой процессора, который не может быть занят выполнением операции, дополнительной операцией длительности величины простоя, получаем формулу (7).

Будем рассматривать три модификации метода ветвей и границ, основанные на способе вычисления нижних оценок: *МВГ/А* – метод ветвей и границ с вычислением нижней оценки H_1 как решение релаксированной задачи *R*, *МВГ/Б* – метод ветвей и границ с вычислением нижней оценки H_2 с помощью неравенства (8), *МВГ/В* – метод ветвей и границ с вычислением нижней оценки H_3=max(H_1, H_2).

Вычислительный эксперимент

Реализация алгоритмов осуществлена с использованием платформы NET. Дополнительно были реализованы решатель задачи, генератор задач, позволяющий по заданным параметрам случайным образом генерировать задачу, и визуализатор задачи. Решатель позволяет находить решения задач с использованием фронтального алгоритма или метода ветвей и границ, а также отображает найденное решение в виде графика Ганта.

Рассматриваемые задачи можно классифицировать следующим образом:

1) КР – задача с большой конкуренцией за ресурсы, в ней в каждый момент времени во фронте находится большое количество операций, которые могут выполняться;

2) БКР – задача без конкуренции за ресурсы, в каждый момент времени во фронте находятся одна-две операции.

Все задачи генерировались случайным образом с помощью написанного генератора задач. Число в указании задачи – общее количество операций в ней (табл. 1).

Каждая ячейка таблицы содержит два числа: первое – результат работы алгоритма (значение критерия на найденном решении), второе – его время работы. Для приведенных схем метода ветвей и границ результат работы алгоритма определяется «рекордом» – значением критерия на полученном лучшем допустимом решении задачи. Символ -------* означает, что алгоритм был остановлен по истечении 10 минут. При этом наличие решения с такой пометкой говорит о том, что оно не является оптимальным и было получено в течение 10 минут, а его отсутствие – о том, что за 10 минут не было найдено ни одного решения.

Таблица 2 для каждого эксперимента содержит минимальное значение нижней оценки (НО) среди всех полученных и неотсеянных (в результате процедуры отсев) решений в течение 10 минут. Процент отклонения (ПО) вычисляется по следующей формуле:

$$\text{ПО} = \frac{\text{рекорд} - \text{нижняя оценка}}{\text{нижняя оценка}} \cdot 100.$$

Результаты и их обсуждение

По результатам вычислительного эксперимента видно, что фронтальный алгоритм со схемой упорядочивания операций во фронте по убыванию времени выполнения последующих операций (Б) находит решения лучшие, чем без использования схемы (А), однако делает это медленнее, что хорошо заметно на задачах большой размерности. Хорошо заметна разница в результатах алгоритма «жадного» типа и алгоритма, осуществляющего поиск оптимального решения. Однако фронтальный алгоритм находит решение существенно быстрее, более того, результат метода ветвей и границ для задач большой размерности получить так и не удалось из-за большой вычислительной сложности вычисления оценок.

Таблица 1

Вычислительный эксперимент

Алгоритм / Задача	Фронт/А	Фронт/Б	МВГ/А	МВГ/Б	МВГ/В
10 КР	11 ч 00 м 00 с/ 00:00	7 ч 17 м 8 с/ 00:00	7 ч 08 м 34 с/ 00:01	7 ч 08 м 34 с/ 00:00	7 ч 08 м 34 с/ 00:00
10 БКР	14 ч 13 м 12 с/ 00:00	16 ч 10 м 54 с/ 00:00	7 ч 12 м 0 с/ 00:00	7 ч 12 м 0 с/ 00:00	7 ч 12 м 0 с/ 00:00
15 КР	36 ч 0 м 0 с/ 00:00	36 ч 0 м 0 с/ 00:00	18 ч 37 м 30 с/07:04	18 ч 37 м 30 с/ 02:08	18 ч 37 м 30 с/ 02:10
15 БКР	34 ч 59 м 40 с/ 00:00	34 ч 59 м 40 с/ 00:00	10 ч 24 м 0 с/ 00:04	10 ч 24 м 0 с/ 00:48	10 ч 24 м 0 с/ 00:04
20 КР	19 ч 15 м 0 с/ 00:00	18 ч 15 м 0 с/ 00:00	15 ч 12 м 0 с/-------*	15 ч 12 м 0 с/ -------*	15 ч 12 м 0 с/ -------*
20 БКР	66 ч 26 м 47 с/ 00:00	66 ч 26 м 47 с/ 00:00	16 ч 29 м 24 с/ 00:14	16 ч 29 м 24 с/ -------*	16 ч 29 м 24 с/ 00:15
100 КР	40 ч 21 м 27 с/ 00:00	37 ч 6 м 37 с/ 00:00	35 ч 20 м 0 с/-------*	35 ч 24 м 0 с/ -------*	35 ч 30 м 36 с/ -------*
100 БКР	503 ч 23 м 56 с/ 00:00	186 ч 33 м 48 с/ 00:00	68 ч 37 м 28 с/-------*	72 ч 35 м 18 с/-------*	66 ч 36 м 0 с/-------*
300 КР	242 ч 50 м 34 с/ 00:00	185 ч 34 м 15 с/ 00:00	113 ч 2 м 18 с/-------*	112 ч 35 м 32 с/-------*	111 ч 55 м 1 с/ -------*
300 БКР	1354 ч 37 м 24 с/ 00:00	722 ч 36 м 38 с/ 00:00	665 ч 19 м 20 с/-------*	373 ч 43 м 0 с/ -------*	664 ч 55 м 33 с/ -------*
1000 КР	556 ч 29 м 14 с/ 00:00	499 ч 22 м 2 с/ 00:00	499 ч 22 м 2 с/-------*	415 ч 47 м 28 с/ -------*	499 ч 22 м 2 с/-------*
1000 БКР	2196 ч 49 м 50 с/ 00:00	1544 ч 25 м 51 с/ 00:00	1544 ч 25 м 51 с/ -------*	1091 ч 7 м 3 с/-------*	1544 ч 25 м 51 с/-------*
10000 КР	2903 ч 20 м 58 с/ 00:09	2447 ч 33 м 38 с/ 00:22	2447 ч 33 м 38 с/ -------*	2333 ч 3 м 59 с/ -------*	2447 ч 33 м 38 с/ -------*
10000 БКР	24261 ч 38 м 56 с/ 00:05	15362 ч 4 м 31 с/ 01:13	15362 ч 4 м 31 с-------*	15294 ч 49 м 24 с/ -------*	15362 ч 4 м 31 с-------*

Таблица 2

Процент отклонения «рекорда» относительно нижней оценки

Алгоритм / Задача	МВГ/А		МВГ/Б		МВГ/В	
	НО	ПО	НО	ПО	НО	ПО
20 КР	13 ч 42 м 27 с	10.89	15 ч 4 м 36 с	0.82	15 ч 4 м 36 с	0.82
100 КР	13 ч 18 м 0 с	165.66	35 ч 6 м 0 с	0.85	35 ч 6 м 0 с	1.17
100 БКР	62 ч 15 м 1 с	10.24	54 ч 23 м 21 с	33.46	64 ч 51 м 1 с	2.70
300 КР	29 ч 48 м 0 с	279.32	110 ч 16 м 17 с	1.43	110 ч 15 м 35 с	2.40
300 БКР	343 ч 24 м 45 с	93.74	236 ч 42 м 5 с	57.89	324 ч 53 м 36 с	104.66
1000 КР			412 ч 45 м 0 с	0.74		
1000 БКР			204 ч 43 м 24 с	440.14		
10000 КР			2313 ч 1 м 20 с	0.87		
10000 БКР			1690 ч 57 м 34 с	804.51		

Метод ветвей и границ с нижней оценкой, основанной на неравенстве (8) (Б), вычисляет задачи с конкуренцией за ресурсы быстрее, чем это делает МВГ с нижней оценкой, основанной на задаче релаксации (А), и качественнее, что видно по небольшому значению процента отклонения найденного решения от нижней оценки. Однако оценка Б не так хороша, как оценка А в случае малоразмерной задачи без конкуренции за ресурсы. Алгоритм, основанный на композитной схеме вычисления нижней оценки (В), используя преимущества обеих схем, находит решения со скоростью, сравнимой с лучшим результатом. Однако на задачах большей размерности схема (А) становится неэффективной из-за возрастающей цены вычисления нижней оценки. Схема (Б) за 10 минут успевает перебирать существенно больше вариантов, за счет чего находит лучшее решение.

Заключение

В работе применен новый способ распараллеливания алгоритмов, основанный на решении оптимизационной задачи, поставленной в рамках общей математической модели. Для решения задачи рассмотрен эвристический алгоритм фронтального типа, предложены индивидуальные процедуры для поиска оптимального решения задачи методом ветвей и границ. Проведен вычислительный эксперимент.

Данный подход можно использовать для оценки уровня параллелизма задачи. Варьируя вычислительные ресурсы, можно исследовать поведение алгоритма на различных вычислительных системах.

Список литературы

1. Jansen K., Porkolab L. Improved approximation schemes for scheduling unrelated parallel machines // Proceedings of the thirty-first annual ACV symposium on theory of computing. Atlanta, Georgia, United States, May 01–04 1999. P. 408–417.

2. Serna M., Xhafa F. Approximating scheduling unrelated parallel machines in parallel // Computational Optimization and Applications. 2002. V. 21. P. 325–338.

3. Lenstra J.K., Shmoys D.B., Tardos E. Approximation algorithms for scheduling unrelated parallel machines // Mathematical Programming. 1990. V. 46. P. 259–271.

4. Samadzadeh F.A., Hedrick G.E. Near-optimal multiprocessor scheduling // Proceedings of the 1992 ACV annual conference on communications. Kansas City, Missouri, United States, March 03–05 1992. P. 477–484.

5. Samadzadeh F.A., Hedrick G.E. A heuristic multiprocessor scheduling algorithm for creating near-optimal schedules using task system graphs // Proceedings of the 1992 ACV/SIGAPP symposium on applied computing: technological challenges of the 1990′s. Kansas City, Missouri, United States, March 1992. P. 711–718.

6. Sinnen O., Kozlov A.V., Shahul A.Z.S. Optimal scheduling of task graphs on parallel systems // 25th LASTED International Multi-Conference Parallel and Distributed Computing and Networks. 2007.

7. Корбут А.А., Финкельштейн Ю.Ю. Дискретное программирование. М.: Наука, 1969. 368 с.

ACYCLIC ALGORITHM PARALLELIZATION PROBLEM

V.V. Slobodskoy

A parallelization problem of an algorithm without cycles and branches is considered. A general mathematical model is built and investigated. An optimization problem is set up and its solution algorithms are proposed. An experiment is carried out.