КОМБИНИРОВАННЫЙ АЛГОРИТМ ПЛАНИРОВАНИЯ КОМПЛЕКСОВ РАБОТ НА ОСНОВЕ МЕТОДА ВЕТВЕЙ И ГРАНИЦ

Гончар Д.Р.

ВЦ ФИЦ ИУ РАН, Россия, г. Москва, ул. Вавилова, д.40

dgonchar@ccas.ru

Аннотация: Рассматривается минимаксная задача составления расписания минимальной длины без прерываний для многопроцессорной системы. Для решения данной задачи предложен параллельный алгоритм на основе метода ветвей и границ ограниченной глубины и алгоритмы для проверки качества полученного решения.

Ключевые слова: многопроцессорная система, работы без прерываний, расписание минимальной длины.

Введение

Предложенная версия алгоритма сочетает использование подхода метода ветвей и границ для обхода заданной части дерева решений, а остальные задания назначаются согласно эвристическому алгоритму, что дает дополнительные возможности при выборе наиболее подходящего к условиям решения данной задачи алгоритма.

1 Постановка задачи

Рассматривается множество работ $N = \{1, 2, \ldots, n\}$, подлежащее выполнению, и вычислительная система, состоящая из m процессоров для их обработки. Время выполнения работы i на процессоре j равно t_{ij} ($i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$). При выполнении работ не допускаются переключения с одного процессора на другой и прерывания. В заданный момент времени каждый процессор может выполнять не более одной работы, а каждая работа может выполняться не более чем одним процессором.

Расписание выполнения работ N определим как разбиение множества N на m непересекающихся подмножеств $N_1, N_2, \ldots, N_m$ ($N = \bigcup_{j=1}^{m} N_j$; $N_{j_1} \bigcap N_{j_2} = \varnothing$ при $j_1 \neq j_2$). Работы из множества N_j приписываются процессору j и выполняются на нем одна за другой в произвольном порядке. Величина $Q_j = \sum_{i \in N_j} t_{ij}$ – загруженность процессора j ($j = 1, 2, \ldots, m$), а $\max_{j=1, 2, \ldots, m} Q_j$ – это длина расписания. Задача заключается в построении расписания минимальной длины, т.е. оптимального по быстродействию расписания.

Подобные задачи широко освещены в литературе. При их решении применяются, например, такие методы, как случайный и исчерпывающий поиск, методы математического программирования [1], метод ветвей и границ [2, 3], муравьиные алгоритмы, поиск с запретами, вероятностные алгоритмы, генетические алгоритмы [4], метод имитации отжига, различные эвристические алгоритмы [5], алгоритмы агрегирования и др.

2 Метод ветвей и границ

2.1 Ветвление

Множество всех расписаний (их число равно m^n) будем описывать в виде дерева расписаний. На нулевом уровне дерева находится корень, который соответствует множеству всех расписаний. На первом уровне находится m вершин, каждая из которых соответствует множеству всех расписаний, в которых первая работа назначена на определенный процессор. На втором уровне дерева находится m^2 вершин, каждая из которых соответствует множеству всех расписаний, в которых первые две работы назначены на один или два определенных процессора. На n-м уровне дерева расписаний находится m^n листьев, каждый из которых соответствует некоторому расписанию выполнения множества работ N.

Пусть x_k – некоторый узел уровня k дерева расписаний, $R(x_k)$ – множество всех расписаний, соответствующих этому узлу (т.е. множество расписаний, в которых работы 1, 2, …, k назначены на определенные процессоры), x_{k+1}^j – узел уровня $k+1$ ($k < n$), связанный с узлом x_k ребром, соответствующим процессору j. Наша цель – вычисление нижней и верхней оценок минимальной длины расписания на множестве $R(x_k)$. Имея эти оценки, можно применить стандартную схему метода ветвей и границ [6] (например, одностороннего или фронтального ветвления).

2.2 Нижняя оценка

Пусть T_j ($j = 1, \ldots, m$) – загруженность процессора j после назначения первых k работ (т.е. T_j – это суммарная длительность работ из числа 1, 2, …, k, назначенных на процессор j). Нижнюю оценку $L(x_k)$ минимальной длины расписания на множестве $R(x_k)$ будем вычислять следующим образом: $L(x_k) = \max(L_1(x_k), L_2(x_k), L_3(x_k))$, где $L_1(x_k)$, $L_2(x_k)$, $L_3(x_k)$ – это нижние оценки, вычисленные тремя различными способами.

Величина $L_1(x_k)$ вычисляется как следующий максимум: $L_1(x_k) = \max\limits_{j=1,2,\ldots,m} T_j$. При хранении величины $T_1, T_2, \ldots, T_m$ в виде обычного массива сложность вычисления $L_1(x_k)$ составляет $\theta(m)$.

Величина $L_2(x_k)$ вычисляется как следующий максимум:

$$L_2(x_k) = \max_{i=k+1,\ldots,n} \min_{j=1,\ldots,m} (T_j + t_{ij})$$

При использовании для этого обычного двумерного массива A с элементами $a_{ij} = T_j + t_{ij}$, $i = k+1, \ldots, n$; $j = 1, 2, \ldots, m$ сложность вычисления величины $L_2(x_k)$ составляет $\theta(mn)$.

Величина $L_3(x_k)$ вычисляется по формуле.

$$L_3(x_k) = \frac{1}{m}\left(\sum_{j=1}^{m} T_j + \sum_{i=k+1}^{n} \min_{j=1,\ldots,m} t_{ij}\right)$$

2.3 Верхняя оценка

В качестве верхней оценки $H(x_k)$ минимальной длины расписания на множестве $R(x_k)$ возьмем длину расписания, в котором работы 1, 2, …, k назначены на процессоры в соответствии с вершиной x_k дерева расписаний, а работы $k+1$, …, n назначаются по следующему "жадному" алгоритму. Пусть уже назначены работы 1, 2, …, p ($k \le p < n$), T_j – загруженность процессора j ($j = 1, 2, \ldots, m$) и $\min(T_1 + t_{p+1,1}, \ldots, T_m + t_{p+1,m}) = T_{j_0} + t_{p+1,j_0}$. Тогда работа $p+1$ назначается на процессор j_0. Указанная процедура повторяется для $p = k, k+1, \ldots, n-1$. Сложность процедуры вычисления величины $H(x_k)$ составляет $O(mn)$. В случае, когда процессоры идентичные (т.е. $t_{ij_1} = t_{ij_2}$ при всех $1 \le j_1, j_2 \le m$) работа $p+1$ назначается на процессор j_0, определяемый соотношением $\min(T_1, T_2, \ldots, T_m) = T_{j_0}$.

3 Распараллеливание обхода дерева в методе ветвей и границ при реализации алгоритма в многопроцессорной системе

При применении метода ветвей и границ происходит последовательное разбиение множества допустимых решений на подмножества: на каждом последующем шаге новые подмножества образуются в итоге разбиения некоторых подмножеств, полученных на предыдущих шагах. Так строится дерево решения исходной задачи. Такое разбиение продолжается до тех пор, пока для подмножеств, соответствующих концевым вершинам дерева, решение задачи уже не требует разбиения. В итоге разбиения начальная задача распадается на ряд подзадач, которые могут решаться в значительной степени независимо друг от друга.

Поскольку полная реализация метода ветвей и границ обладает высокими требованиями к объёму оперативной памяти и довольно трудоёмка в расчётах, для ряда задач, по мнению автора, имеет смысл применять комбинированный алгоритм, когда первые k уровней дерева решений получаемого расписания исследуются по методу ветвей и границ, а оставшиеся – на основе одно из известных эвристических алгоритмов [5, 7].

Это предложение было программно реализовано автором, с его помощью проведены расчёты, показавшие весьма высокую точность получаемых решений (по сравнению с длительностью рассчитываемой для соответствующих входных данных идеальной оценки длительности расписания).

Литература

1. *Кочетов Ю.А., Столяр А.А.* Использование чередующихся окрестностей для приближенного решения задачи календарного планирования с ограниченными ресурсами // Дискретный анализ и исследования операций. Сер. 2. 2003. Т. 10. № 2. С. 29–56.
2. *Алексеев О.Г.* Комплексное применение методов дискретной оптимизации. М.: Наука, 1987.
3. *Фуругян М.Г.* Некоторые алгоритмы решения минимаксной задачи составления многопроцессорного расписания. // Изв. РАН, ТиСУ. 2014, № 2. С. 50–56.
4. *Костенко В.А., Смелянский Р.Л., Трекин А.Г.* Синтез структур вычислительных систем реального времени с использованием генетических алгоритмов// Программирование. 2000. № 5. С. 63–72.
5. *Brucker P.* Scheduling Algorithms. Heidelberg, Springer, 2001.
6. *Коглер В., Штиглиц К.* Перечислительные и итеративные алгоритмы. В кн.: Теория расписаний и вычислительные машины. Под ред. Коффмана Э.Г. М.: Наука, 1984. С. 251–288
7. *Гончар Д.Р.* Параллельная реализация мультиоценочного алгоритма составления многопроцессорного расписания без прерываний. // Некоторые алгоритмы планирования вычислений и методы многокритериальной оптимизации для многопроцессорных систем. М.: ВЦ РАН, 2014. С. 21–31.