DOI: 10.25728/iccss.2023.17.45.060

Гончар Д.Р.

Решение методом ветвей и границ минимаксной задачи составления расписания с параллельной реализацией

Аннотация: Описывается алгоритм решения минимаксной задачи составления расписаний и обсуждается его параллельная реализация на многопроцессорном вычислительном комплексе.

Ключевые слова: построение расписания, минимаксная задача, параллельная реализация

Построение оптимальных расписаний необходимо при планировании производственной деятельности, сложных и масштабных транспортных систем, в ходе различных видов медицинского, промышленного, экологического мониторинга и в ряде других случаев.

1. Постановка задачи

Пусть имеется множество заданий $N = \{1, 2, \ldots, n\}$, которое необходимо выполнить с помощью m обработчиков, составляющих производственную (транспортную и т.д.) систему для их обработки. Длительность выполнения задания i на обработчике j равно t_{ij} ($i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$). Каждое задание в каждый миг времени может выполняться не более чем одним обработчиком, а каждый обработчик может выполнять не более одного задания. Переключения с одного обработчика на другой и прерывания при выполнении заданий не допускаются.

Расписание выполнения заданий 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 $j_1 \neq j_2$). Задания из множества N_j приписываются обработчику j и выполняются на нем одна за другой в произвольном порядке. Под загруженностью обработчика j ($j = 1, 2, \ldots, m$) будем понимать величину $Q_j = \sum_{i \in N_j} t_{ij}$, а $\max_{j=1, 2, \ldots, m} Q_j$ – это

длина расписания. Задача заключается в построении расписания минимальной длины.

При решении подобных задач применяются, например, такие подходы, как случайный и исчерпывающий поиск, методы математического программирования [1], метод ветвей и границ [2, 7], муравьиные алгоритмы, поиск с запретами, вероятностные алгоритмы, генетические алгоритмы [3], метод имитации отжига, различные эвристические алгоритмы [4, 5], алгоритмы агрегирования и др.

2. Метод ветвей и границ

Для решения вышеприведенной задачи предлагается метод ветвей и границ, основанный на результатах работы [2]. Этот метод подразумевает структуру поиска оптимального решения в виде дерева (что выполняется для нашей задачи), последовательное разбиение исходного множества решений на подмножества (ветви дерева решений) и применение оценочных процедур для определения перспективности подробного исследования очередной ветви дерева решений. На каждом последующем шаге новые подмножества образуются в результате разбиения некоторых подмножеств, полученных на предыдущих шагах, пока для подмножеств, соответствующих концевым вершинам дерева, решение задачи уже не требует разбиения.

В итоге указанного разбиения мы получаем множество подзадач, которые могут обрабатываться независимо (совмещено, одновременно по времени)

.

2.1. Дерево решений

Опишем множество всех расписаний (их число равно m^n) в виде дерева решений. Корень дерева находится на нулевом уровне. Корень 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 [7] (например, одностороннего или фронтального ветвления).

2.2. Вычисление нижней оценки

Пусть T_j (j = 1, …, 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:

$$L(x_k) = \max(L_1(x_k), L_2(x_k), L_3(x_k)), \qquad (1)$$

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, …, T_m в виде обычного массива сложность вычисления $L_1(x_k)$ составляет $\theta(m)$.

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

$$L_2(x_k) = \max_{i=k+1,\ldots,n} \min_{j=1,\ldots,m} (T_j + t_{ij}) \qquad (2)$$

При использовании для этого 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:

$$L_3(x_k) = \frac{1}{m}\left(\sum_{j=1}^{m} T_j + \sum_{i=k+1}^{n} \min_{j=1,\dots,m} t_{ij} \right). \qquad (3)$$

Величину $\min\limits_{j=1,\dots,m} t_{ij}$ вычислим для всех i = 1, 2, …, n сразу до начала вычисления нижних оценок. Тогда сложность вычисления величины $L_3(x_k)$ составляет $O(n + m)$. Перейдем в дереве расписаний от узла x_k к узлу $x_{k+1}^{j_0}$, $k < n$ (т.е. будем считать, что работа $k+1$ назначена на процессор j_0).

Тогда:

$$L_3(x_{k+1}^{j_0}) = \frac{1}{m}\left(\sum_{j=1}^{m} T_j + t_{k+1,j_0} + \sum_{i=k+1}^{n} \min_{j=1,\dots,m} t_{ij} \right). \qquad (4)$$

Вычислим разность:

$$L_3(x_{k+1}^{j_0}) - L_3(x_k) = \frac{1}{m}\left(t_{k+1,j_0} - \min_{j=1,\dots,m} t_{k+1,j} \right). \qquad (5)$$

Таким образом,

$$L_3(x_{k+1}^{j_0}) = L_3(x_k) + \frac{1}{m}\left(t_{k+1,j_0} - \min_{j=1,\dots,m} t_{k+1,j} \right) \qquad (6)$$

и с помощью данного рекуррентного соотношения, используя $L_3(x_k)$, величина $L_3(x_{k+1}^{j_0})$ вычисляется за время $O(1)$.

2.3. Вычисление верхней оценки

В качестве верхней оценки $H(x_k)$ минимальной длины расписания на множестве $R(x_k)$ возьмем длину расписания, в котором работы 1, 2, …, k в соответствии с вершиной x_k дерева расписаний распределены на процессоры, а работы k+1, …, n распределяются по следующему «жадному» алгоритму. Пусть уже распределены работы 1, 2, …, p ($k \le p < n$), T_j – загруженность процессора j (j = 1, 2, …, m) и

$$\min(T_1 + t_{p+1,1}, \ldots, T_m + t_{p+1,m}) = T_{j_0} + t_{p+1,j_0}. \qquad (7)$$

Тогда работа p+1 назначается на процессор j_0.

Указанные действия повторяются для $p = k, k + 1, \ldots, n - 1$. Сложность процедуры вычисления величины $H(x_k)$ составляет $O(mn)$.

2.4. Ветвление

Последовательное разбиение множества допустимых решений на подмножества при применении метода ветвей и границ происходит следующим образом: на каждом последующем шаге новые подмножества получаются в результате разбиения некоторых подмножеств, полученных на предыдущих шагах. Так образуется выше упоминавшееся 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 заметной степени вне зависимости друг от друга.

При этом представляется целесообразным поддерживать определенные связи между порожденными подзадачами, что связано с тем, что дерево решения может оказаться не вполне хорошо уравновешенным, что приводит к тому, что какая-то часть процессоров вычислительной системы оказывается загруженными неравномерно. Другая причина в том, что возникающие при попытке уравновешивания нагрузки зависимости по данным между подзадачами, связанные с передачей оценок, наилучших значений оптимизируемого функционала и других подобных сведений, могут приводить к большим накладным расходам на взаимодействие процессов, препятствующих повышению параллельной эффективности.

Для преодоления перечисленных причин снижения эффективности распараллеливания решения задачи применяется распределение обменов по вычислительному пространству, методы оптимизации загрузки процессов и минимизации обменов данными [7, 8].

Литература:

1. *Алексеев О.Г.* Комплексное применение методов дискретной оптимизации. – М.: Наука, 1987. – 250 с.

2. *Фуругян М.Г.* Некоторые алгоритмы решения минимаксной задачи составления многопроцессорного расписания // Известия РАН. ТиСУ. – 2014. – № 2. – С. 50-56.

3. *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.

4. *Brucker P.* Scheduling Algorithms. – Heidelberg, Springer, 2001. – 371 p.

5. *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.

6. *Тимошевская Н.Е.* Параллельные методы обхода дерева // Математическое моделирование. – 2004. – Т. 16. № 1. – С. 105-114.

7. *Посыпкин М.А., Сигал И.Х., Галимьянова Н.Н.* Алгоритмы параллельных вычислений для решения некоторых классов задач дискретной оптимизации. – М.: ВЦ РАН, 2005. – 43 с.

8. *Посыпкин М.А., Сигал И.Х., Галимьянова Н.Н.* Параллельные алгоритмы в задачах дискретной оптимизации: вычислительные модели, библиотека, результаты экспериментов. – М.: ВЦ РАН, 2006. – 50 с.

9. *Кнут Д.* Искусство программирования для ЭВМ. Т. 1. – М.: Мир, 1976. – 734 с.
