КОМБИНИРОВАННЫЙ АЛГОРИТМ ПЛАНИРОВАНИЯ КОМПЛЕКСОВ ВЫЧИСЛИТЕЛЬНЫХ РАБОТ НА ОСНОВЕ МЕТОДА ВЕТВЕЙ И ГРАНИЦ

Гончар Д.Р.
ВЦ ФИЦ ИУ РАН
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 сочетания метода ветвей и границ ограниченной глубины с эвристикой и алгоритмы для проверки качества полученного решения.

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

Введение

Задачи построения оптимальных расписаний достаточно широко встречаются на практике при планировании производственной деятельности, управление энергетическими объектами, работе различных систем мониторинга (например, экологических систем) и множества других. Различные области применения в ряде случаев обусловливают и разные требования к точности и скорости получения искомого решения.

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

Пусть имеется множество работ $N = \{1, 2, \ldots, n\}$, которое необходимо выполнить с помощью m процессоров, составляющих вычислительную систему для их обработки. Длительность выполнения работы i на процессоре j равно t_{ij} ($i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$). Переключения с одного процессора на другой и прерывания при выполнении работ не допускаются. В каждый момент времени каждая работа может выполняться не более чем одним процессором, а каждый процессор может выполнять не более одной работы.

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

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.

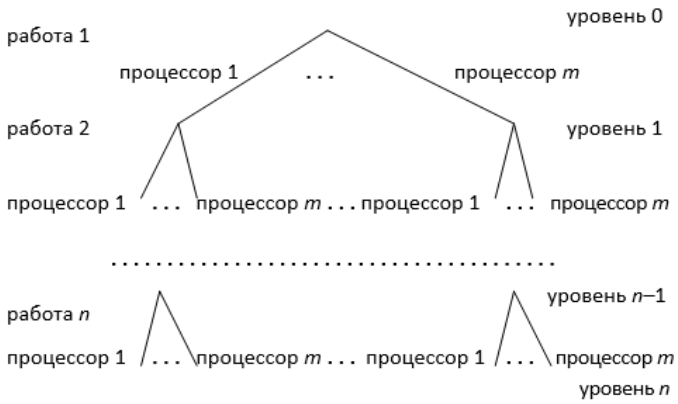
2 Метод ветвей и границ

Для решения поставленной задачи предлагается метод ветвей и границ, основанный на результатах работы [2, 3].

2.1 Ветвление

Множество всех расписаний (их число равно m^n) будем описывать в виде дерева расписаний, изображенного на рисунке. 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, …, T_m в виде обычного массива сложность вычисления $L_1(x_k)$ составляет $\theta(m)$. Будем хранить эти величины в виде двоичной кучи (пирамиды) с максимальным элементом в корне [12].

Корню x_0 дерева расписаний соответствуют величины $T_1 = 0$, $T_2 = 0, \ldots, T_m = 0$ и $L_1(x_0) = 0$. Тогда сложность получения максимального элемента составит $\theta(1)$. Если работа $k+1$ будет назначена на процессор j (т.е. если от узла xk в дереве расписаний перейти к узлу x_{k+1}^j), то будет получен новый массив T_1, …, T_{j-1}, $T_j+t_{k+1,j}$, T_{j+1} …, T_m. Для преобразования этого массива в кучу следует поднять элемент $T_j + t_{k+1,j}$ до нужного уровня в куче. Сложность такой процедуры составляет $O(\log_2(m))$ [13]. В полученной куче в корне по-прежнему будет находиться максимальный элемент.

Таким образом, зная $L_1(x_k)$, можно вычислить $L_1(x_{k+1}^j)$ за время $O(\log_2(m))$.

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 $\theta(mn)$. Будем хранить каждую строку массива A в виде двоичной кучи с минимальным элементом в корне. Для корня x_0 дерева расписаний $a_{ij} = t_{ij}$, $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$. Тогда сложность вычисления величины $L_2(x_k)$ составит $\theta(n)$. Если работа $k+1$ будет назначена на процессор j (т.е. если от узла x_k в дереве расписаний перейти к узлу x^j_{k+1}), то из массива A следует исключить $(k+1)$-ю строку, а в каждой оставшейся строке i увеличить элемент a_{ij} на величину $t_{k+1,j}$. Для того, чтобы каждая строка массива A по-прежнему образовывала кучу, следует поднять элемент $a_{ij}+t_{k+1,j}$ до нужного уровня [12]. Сложность такой процедуры составляет $O(n \log_2(m))$. Каждая строка массива A по-прежнему будет

образовывать кучу с минимальным элементом в корне. Таким образом, зная $L_2(x_k)$, можно вычислить $L_2(x^j_{k+1})$ за время $O(n\log_2(m))$.

Величина $L_3(x_k)$ вычисляется по формуле.

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

Величину $\min_{j=1,\dots,m} t_{ij}$ вычислим сразу для всех $i = 1, 2, \dots, n$ 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 $x^{j_0}_{k+1}$, $k < n$ (т.е. будем считать, что работа k+1 назначена на процессор j_0).

Тогда

$$L_3(x^{j_0}_{k+1}) = \frac{1}{m}((\sum_{j=1}^{m} T_j + t_{k+1,j_0}) + \sum_{i=k+1}^{n} \min_{j=1,\dots,m} t_{ij})$$

Вычислим разность

$$L_3(x^{j_0}_{k+1}) - L_3(x_k) = \frac{1}{m}(t_{k+1,j_0} - \min_{j=1,\dots,m} t_{k+1,j})$$

Таким образом,

$$L_3(x^{j_0}_{k+1}) = L_3(x_k) + \frac{1}{m}(t_{k+1,j_0} - \min_{j=1,\dots,m} t_{k+1,j})$$

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

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, …, m) и $min(T_1+t_{p+1,1}, \dots, T_m+t_{p+1,m}) = T_{j0}+t_{p+1,j0}$. Тогда работа p+1 назначается на процессор j_0. Указанная процедура повторяется для $p = k, k + 1, \dots, n - 1$. Сложность процедуры вычисления величины $H(x_k)$ составляет $O(mn)$. В случае, когда процессоры идентичные (т.е. $t_{ij1}=t_{ij2}$ при всех $1 \le j_1, j_2 \le m$) работа $p + 1$ назначается на процессор j_0, определяемый соотношением $min(T_1, T_2, \dots, T_m) = T_{j0}$.

В этом случае величины $T_1, T_2, \dots, T_m$ можно хранить в виде двоичной кучи с минимальным элементом в корне. Корню x_0 дерева расписаний соответствуют величины $T_1 = 0, T_2 = 0, \dots, T_m = 0$. Тогда при переходе от узла x_k к узлу x^j_{k+1} применима процедура, аналогичная той, которая описана для вычисления величины $L_1(x_k)$, с той лишь разницей, что теперь в корне находится минимальный элемент и поэтому элемент $T_{j0}+t_{p+1,j0}$ следует опустить в дереве расписаний до нужного уровня. Сложность такой процедуры составляет $O(\log_2(m))$, а сложность вычисления величины $H(x_k)$ составляет $O(n\log_2(m))$.

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

Дерево решения исходной задачи при применении метода ветвей и границ строится следующим образом: множества допустимых решений последовательно разбиваются на подмножества: новые подмножества на каждом следующем шаге создаются в итоге разбиения некоторых подмножеств, полученных на предыдущих шагах. 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 независимо друг от друга.

В тоже время желательно поддерживать определенные связи (зависимости) между полученными подзадачами, что связано со следующими двумя обстоятельствами:

1) дерево решения может оказаться плохо уравновешенным, что приводит к тому, что процессоров вычислительной системы оказываются неравномерно загруженными;
2) 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 успешности распараллеливания решения задачи применяются методы оптимизации загрузки процессов, минимизации обменов данными, а также распределения обменов по вычислительному пространству [8, 9].

При реализации метода ветвей и границ известны несколько стратегий обхода дерева, в частности, фронтальный обход, когда исследуются все узлы на каждом уровне дерева (последовательно или параллельно, в зависимости от реализации алгоритма и доступных вычислительных средств) или обход в глубину, когда на каждом следующем уровне выбирается на основе предпочтений какая-то одна из подветвей, при этом уточняются реально достижимые верхние и нижние оценки конкретного решения, пока оно не будет получено полностью. В каждом из подходов есть свои преимущества и недостатки. Например, при использовании фронтального обхода наблюдается быстро возрастающие требования алгоритма к необходимой оперативной памяти для его работы. В данном случае автор программно реализовал обход дерева решений «в глубину».

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 одного 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).

4 Расчеты по реализованному алгоритму в многопроцессорной системе

Для проверки качества полученных с помощью комбинированного алгоритма на основе метода ветвей и границ решений могут применяться разные подходы. Наиболее очевидный состоит в точном решении задачи другим способом (например, полным перебором). Но по понятным причинам экспоненциального роста трудоёмкости такой проверки с увеличением размерности задачи, такой подход больше подходит для сравнительно небольших размерностей задачи. Поэтому автором вычислялась идеальная оценка оптимального расписания для заданных входных данных (что заметно менее трудоемко, чем точное решение задачи) и эта оценка сравнивалась с решением, полученным с помощью предложенного комбинированного алгоритма.

Подробные итоги расчётов предполагается представить на конференции.

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

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