Федеральный исследовательский центр «Информатика и управление» Российской академии наук

Вычислительный центр им. А.А. Дородницына

Некоторые алгоритмы планирования вычислений в многопроцессорных системах

2017

ФЕДЕРАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЦЕНТР
«ИНФОРМАТИКА И УПРАВЛЕНИЕ»
РОССИЙСКОЙ АКАДЕМИИ НАУК

Некоторые алгоритмы планирования вычислений в многопроцессорных системах

Москва
ФИЦ ИУ РАН
2017

УДК 519.86

Печатается по решению Учёного совета Федерального исследовательского центра «Информатика и управление» Российской академии наук

Некоторые алгоритмы планирования вычислений в многопроцессорных системах / *Фуругян М.Г., Гончар Д.Р., Мирошник С.Н., Рабинович Я.И.* / под ред. к.ф.-м.н. *Л.Л. Вышинского*. М.: ФИЦ ИУ РАН, 2017. – 38 с. – ISBN 978-5-91993-070-9.

Сборник посвящен решению некоторых дискретных оптимизационных задач, возникающих при разработке многопроцессорных систем реального времени. В частности, в сборнике получены следующие результаты:

– описаны алгоритмы построения допустимого расписания с прерываниями в многопроцессорной системе с несколькими типами дополнительных ресурсов;

– разработан алгоритм построения многопроцессорных расписаний без прерываний, основанный на методе ветвей и границ;

– разработан алгоритм оптимизации структуры базы данных реального времени.

Кроме того, в сборнике исследован ряд численных методов оптимизации, применяемых при разработке систем реального времени.

Ключевые слова: многопроцессорная система, оптимальное расписание, метод ветвей и границ, параллельный алгоритм, избыточность базы данных, численные методы оптимизации.

Рецензенты: *А.А. Белолипецкий*, *В.А. Костенко*

ISBN 978-5-91993-070-9

НЕКОТОРЫЕ АЛГОРИТМЫ СОСТАВЛЕНИЯ МНОГОПРОЦЕССОРНЫХ РАСПИСАНИЙ С ДОПОЛНИТЕЛЬНЫМИ РЕСУРСАМИ

М.Г. Фуругян

Рассматривается задача составления допустимого расписания с прерываниями в многопроцессорной системе при заданных директивных интервалах. Помимо процессоров в системе имеются дополнительные ресурсы. Длительности выполнения работ линейно зависят от количества выделенного им дополнительного ресурса. Разработаны полиномиальные алгоритмы, основанные на сведе́нии исходной задачи к потоковым.

1. Введение

Составление допустимых расписаний является одной из наиболее важных задач при разработке математического и программного обеспечения, используемого в системах реального времени, которые находят широкое применение в различных областях человеческой деятельности. Для функционирования таких систем необходимо иметь заранее составленное расписание работы всех программных модулей. С появлением новой многопроцессорной вычислительной техники важность таких задач еще более возрастает.

Настоящая статья содержит обзор работ автора по разработке алгоритмов планирования вычислений в многопроцессорных системах с дополнительными ресурсами. Задача составления многопроцессорного расписания для случая, когда дополнительный ресурс отсутствует, рассматривалась в [1 − 4]. Задача с одним типом дополнительного ресурса и идентичными процессорами рассмотрена в [5], а с

произвольными процессорами – в [6] . Задача с идентичными процессорами и несколькими типами дополнительных ресурсов рассмотрена в [7]. Некоторые методы построения расписаний без прерываний, основанные на имитационном моделировании, описаны в [8], а основанные на муравьиных алгоритмах – в [9].

2. Постановка задачи

Рассматривается вычислительная система, состоящая из m процессоров t типов. Производительность процессоров j-го типа равна s_j, а их число составляет m_j ($j = 1, 2, ..., t$, $\sum_{j}^{t} m_j = m$). Имеется множество заданий (работ) $N = \{1, 2, \ldots, n\}$. Предполагается, что в каждый момент времени каждый процессор может выполнять не более одного задания, а каждое задание выполняется не более чем одним процессором. При выполнении заданий допускаются прерывания и переключения с одного процессора на другой. Предполагается, что прерывания и переключения не сопряжены с временны́ми затратами. Для задания $i \in N$ установлен директивный интервал $(b_i, f_i]$ (т.е. работа i может выполняться только в этом интервале). Помимо процессоров в системе имеется K типов дополнительных ресурсов невозобновляемого типа. Суммарное количество k-го типа этого ресурса составляет R_k, $k = 1, 2, ..., K$. Если работе i выделено r_{ik} единиц дополнительного ресурса k-го типа, $i \in N$; $k = 1, 2, ..., K$, то объем работы процессоров по выполнению задания i составляет $Q_i = d_i - \sum_{k=1}^{K} a_{ik} r_{ik}$, где

$$r_{ik} \in [0, \bar{r}_{ik}], \quad i \in N, \ k = 1, 2, ..., K, \qquad (1)$$

$$\sum_{i \in N} r_{ik} \le R_k, \quad k = 1, 2, ..., K, \tag{2}$$

a_{ik}, d_i, $\bar{r}_{ik}$ – заданные величины, $a_{ik} \ge 0$, $d_i > 0$, $\bar{r}_{ik} \ge 0$, $d_i - \sum_{k=1}^{K} a_{ik} \bar{r}_{ik} > 0$. Таким образом, $Q_i \in [d_i - \sum_{k=1}^{K} a_{ik} \bar{r}_{ik};\ d_i]$. Требуется найти такое распределение ресурсов r_{ik}^0, $i \in N$; $k = 1, 2, ..., K$, при котором существует допустимое расписание (т.е. такое расписание, когда каждая работа полностью выполняется в своем директивном интервале), или установить, что такого распределения ресурсов не существует. При этом указанное распределение ресурсов должно удовлетворять ограничениям (1), (2). Примеры подобных задач содержатся в [6].

3. Идентичные процессоры, один тип дополнительного ресурса

В этом разделе предполагается, что производительности всех процессоров совпадают. В этом случае задаются длительности работ. Имеется один тип дополнительного ресурса ($K = 1$) в количестве R.

Рассмотрим сначала случай, когда все директивные интервалы совпадают. Без ограничения общности можно считать, что $b_i = 0$, $f_i = F$ $(i \in N)$. В этом случае постановка задачи следующая [5]. Если работе i выделено r_i единиц дополнительного ресурса, то ее длительность составляет $t_i = d_i - a_i r_i$, где

$$r_i \in [0, \bar{r}_i], \ i = 1, ..., n, \tag{3}$$

$$\sum_{i \in N} r_i \le R, \tag{4}$$

$a_i, d_i, \bar{r}_i$ – заданные величины, $a_i > 0$, $d_i > 0$, $0 \le \bar{r}_i < d_i / a_i$, $t_i \in [d_i - a_i \bar{r}_i, d_i]$, $d_i - a_i \bar{r}_i > 0$. Требуется найти такое распре-

деление ресурсов $(r_1^0, r_2^0, ..., r_n^0)$, при котором существует допустимое расписание. При этом распределение ресурсов $(r_1^0, r_2^0, ..., r_n^0)$ должно удовлетворять ограничениям (3), (4). В [5] показано, как такая задача сводится к задаче линейного программирования.

В общем случае, когда директивные интервалы произвольные, в [5] строится потоковая сеть $G = (V, A)$ (см. рисунок) с источником s и стоком t (V – множество узлов сети, A – множество дуг). Пусть $y_0 < y_1 < ... < y_p$ $(p < 2n)$ – все различные по величине значения b_i, f_i $(i \in N)$. Определим $V = \{s, t, I_1, I_2, ..., I_p, w_1, ..., w_n\}$, где узел I_j соответствует интервалу $(y_{j-1}, y_j]$ $(j = 1, 2, ..., p)$, а узел w_i – работе $i \in N$. Множество дуг A сети G определим следующим образом: (s, I_j) $(j = 1, 2, ..., p)$; (w_i, t) $(i \in N)$; (I_j, w_i) в случае, если $(y_{j-1}, y_j] \subseteq (b_i, f_i]$ (отметим, что для любых j $(1 \le j \le p)$ и $i \in N$ интервал $(y_{j-1}, y_j]$ либо не пересекается с интервалом $(b_i, f_i]$, либо целиком лежит в нем); определим также возвратную дугу (t, s). Пусть $\Delta_j = y_j - y_{j-1}$ – длина интервала I_j. Для каждой дуги определим три параметра: L, U, C, где L – нижняя граница потока по дуге, U – верхняя граница потока по дуге, C – стоимость единицы потока по дуге. Параметры дуг сети G указаны в таблице.

Таблица. Параметры дуг сети G

Дуга	L	U	C
(s, I_j)	0	$m\Delta_j$	0
(I_j, w_i)	0	Δ_j	0
(w_i, t)	$d_i - a_i \bar{r}_i$	d_i	$-1/a_i$
(t, s)	0	$\sum_{i \in N} d_i$	0

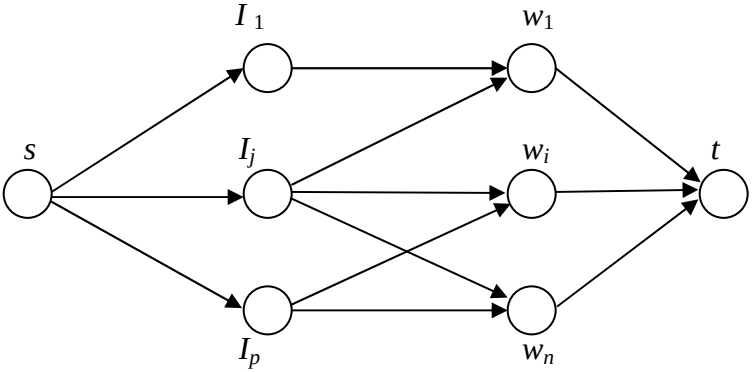


Рис. 1. *Потоковая сеть* G.

Лемма 1. Для существования допустимого расписания необходимо и достаточно существование в сети G циркуляции g, стоимость которой

$$c(g) \le R - \sum_{i \in N} d_i / a_i . \qquad (5)$$

Доказательство содержится в [5].

Опишем *алгоритм* решения поставленной задачи, основанный на доказанной теореме.

Шаг 1. Построить потоковую сеть G.

Шаг 2. Найти циркуляцию g минимальной стоимости в сети G (для этого можно применить, например, алгоритм дефекта). Пусть $c(g)$ – ее стоимость, а g_{it} – величина потока по дуге (w_{it}, t).

Шаг 3. Если $c(g) > R - \sum_{i \in N} d_i / a_i$, то допустимого расписания не существует. Если $c(g) \le R - \sum_{i \in N} d_i / a_i$, то допустимое расписание существует. При этом величина g_{it} потока по дуге (I_j, w_i) равна величине процессорного времени, выделяемого работе i в интервале I_j, $i \in N$; $j = 1, 2, \ldots, p$. Для построения расписания в интервале I_j следует применить алгоритм упаковки, а для построения искомого распи-

сания следует совместить расписания, построенные для всех интервалов I_j, $j = 1, 2, \ldots, p$.

4. Произвольные процессоры, один тип дополнительного ресурса

Результаты этого раздела основаны на работах [2, 3, 6]. В случае одинаковых директивных интервалов исходная задача непосредственно сводится к задаче линейного программирования [6]. Для случая произвольных директивных интервалов по аналогии с [2, 6] строится потоковая сеть, для которой справедливо утверждение леммы 1 и разработан алгоритм, аналогичный алгоритму из разд. 3.

5. Идентичные процессоры, несколько типов дополнительных ресурсов

В этом случае длительности выполнения работ задаются как $t_i = d_i - \sum_{k=1}^{K} a_{ik} r_{ik}$, где

$$r_{ik} \in [0,\ \bar{r}_{ik}], \quad i \in N, \quad k = 1, 2, \ldots, K, \tag{6}$$

$$\sum_{i \in N} r_{ik} \le R_k, \tag{7}$$

$$a_{ik} \ge 0, \ \ d_i > 0, \ \ \bar{r}_{ik} \ge 0, \ \ d_i - \sum_{k=1}^{K} a_{ik} r_{ik} > 0,$$

$$t_i \in [d_i - \sum_{k=1}^{K} a_{ik} r_{ik};\ d_i].$$

В случае одинаковых директивных интервалов ($b_i = 0, \quad f_i = F, i \in N, F > 0$), как показано в [7], задача заключается в поиске такого распределения ресурсов

r_{ik} $(i \in N, \quad k = 1, 2, ..., K)$, которое при $\sum_{i \in N} d_i - mF = B$ удовлетворяет системе ограничений

$$\sum_{i \in N} \sum_{k=1}^{K} a_{ik} r_{ik} \geq B,$$
$$\sum_{i \in N} r_{ik} \leq R_k, \qquad (8)$$
$$r_{ik} \in [0, \bar{r}_{ik}], \quad i \in N, \quad k = 1, 2, ..., K.$$

Если задача (8) имеет решение r_{ik}^0 $(i \in N, \quad k = 1, 2, ..., K)$, то допустимое расписание существует и определяется, положив $t_i^0 = d_i - \sum_{k=1}^{K} a_{ik} r_{ik}^0$ и применив алгоритм, описанный в [1].

В случае произвольных директивных интервалов [7] строится сеть G, дуги которой имеют один параметр − пропускную способность U: $U(s, I_j) = m\Delta_j$, $U(I_j, w_i) = \Delta_j$, $U(w_i, t) = d_i$. Сначала рассмотрим задачу без дополнительных ресурсов, т.е. предположим, что длительности выполнения работ фиксированы и равны t_i, $i \in N$.

Лемма 2. *Для существования допустимого расписания с длительностями работ* $i \in N$, *равными* t_i, *необходимо и достаточно существование в сети* G *потока* g, *для которого*

$$g(w_i, t) = t_i. \qquad (9)$$

Доказательство этого утверждения содержится в [7]. Теперь снова рассмотрим задачу с дополнительными ресурсами. Из леммы 2 следует, что для существования допустимого расписания необходимо и достаточно существования потока g в сети G и распределения ресурсов r_{ik}, $(i \in N,$

$k = 1, 2, ..., K)$, для которых $g(w_i, t) = d_i - \sum_{k=1}^{K} a_{ik} r_{ik} > 0$, $i \in N$, и выполнены ограничения (6),(7). Запишем эти условия в виде следующей задачи линейного программирования [7]. Определить величины $g(s, I_j)$, $g(I_j, w_i)$, $g(w_i, t)$, r_{ik}, $i \in N$, $j = 1, 2, ..., p$, $k = 1, 2, ..., K$, такие, что

$$g(s, I_j) = \sum_{i \in N} g(I_j, w_i), \quad j = 1, 2, ..., p,$$

$$g(w_i, t) = \sum_{j=1}^{p} g(I_j, w_i), \ i \in N,$$

$$g(s, I_j) \le m\Delta_j, \quad j = 1, 2, ..., p,$$

$$g(I_j, w_i) \le \Delta_j, \quad i \in N, \quad j = 1, 2, ..., p,$$

$$g(w_i, t) = d_i - \sum_{k=1}^{K} a_{ik} r_{ik}, \quad i \in N,$$

$$r_{ik} \le \bar{r}_{ik}, \quad i \in N, \quad k = 1, 2, ..., K,$$

$$\sum_{i \in N} r_{ik} \le R_k, \quad k = 1, 2, ..., K,$$

$$g(s, I_j) \ge 0, \ g(I_j, w_i) \ge 0, \ g(w_i, t) \ge 0, \ r_{ik} \ge 0, \quad i \in N,$$

$$j = 1, 2, ..., p, \ k = 1, 2, ..., K.$$

Допустимое расписание существует тогда и только тогда, когда в этой задаче линейного программирования существует решение. Допустимое расписание строится с помощью алгоритма упаковки [1] в каждом интервале I_j, $j = 1, 2, ..., p$.

5. Заключение

Исследована задача составления допустимого расписания с прерываниями в многопроцессорной системе в случае, когда заданы директивные интервалы, а длительности выполнения работ линейно зависят от количества выделен-

ного им дополнительного ресурса. Разработаны полиномиальные алгоритм, основанные на сведе́нии исходной задачи к потоковой и задаче линейного программирования.

Литература

1. *Танаев В.С., Гордон В.С., Шафранский Я.М.* Теория расписаний. Одностадийные системы. М.: Наука, 1984.

2. *Federgruen A., Groenevel H.* Preemptive Scheduling of Uniform Machines by Ordinary Network Flow Technique // Management Science. 1986. V. 32. No. 3. P. 341 – 349.

3. *Gonzales T., Sahni S.* Preemptive Scheduling of Uniform Processor Systems // J. Association for Computing Machinery. 1978. V. 25. No. 1. P. 92 – 101.

4. *Давыдов Э.Г.* Исследование операций. М.: Высшая школа, 1990.

5. *Косоруков Е.О., Фуругян М.Г.* Некоторые алгоритмы распределения ресурсов в многопроцессорных системах // Вестн. МГУ. 2009. Сер. 15. № 4. С. 34 – 37.

6. *Фуругян М.Г.* Планирование вычислений в многопроцессорных АСУ реального времени с дополнительным ресурсом. // АиТМ. 2015. № 3. С. 144 – 150.

7. *Фуругян М.Г.* Составление расписаний в многопроцессорных системах с несколькими дополнительными ресурсами. //Изв. РАН. ТиСУ. 2017. № 2. С. 70 –79.

8. *Костенко В.А.* Алгоритмы построения расписаний для вычислительных систем реального времени, допускающие использование имитационных моделей // Программирование. 2013. № 5. С. 53 – 71.

9. *Костенко В. А., Плакунов А. В.* Алгоритм построения одноприборных расписаний, основанный на схеме муравьиных колоний // Изв. РАН. ТиСУ. 2013. № 6. С. 87 – 96.

ОБ ОДНОМ ОБОБЩЕНИИ ПОНЯТИЯ ВОГНУТОЙ ФУНКЦИИ

Я.И. Рабинович

В деле совершенствования систем реального времени важную роль играет качество применяемых численных методов оптимизации. Исследование сходимости численных методов максимизации функции тесно связано с понятием вогнутой функции и целым рядом обобщений этого понятия. Ниже рассматривается одно из подобных обобщений, обладающее рядом полезных свойств.

Введем следующие стандартные определения.

Определение 1. Заданная на выпуклом множестве $A \subset \mathbf{R}^s$ функция f называется *вогнутой* на A, если при всех $x, y \in A$, $z \in [x, y]$ выполняется неравенство

$$\|y - x\| f(z) \geq \|y - z\| f(x) + \|z - x\| f(y).$$

Определение 2. Дифференцируемая на выпуклом множестве $A \subset \mathbf{R}^s$ функция $f(x)$ называется *псевдовогнутой* на A, если для любых $x, y \in A$ неравенство

$$\langle \nabla f(x), y - x \rangle \leq 0$$

влечет неравенство $f(x) \geq f(y)$.

Определение 3. Заданная на выпуклом множестве $A \subset \mathbf{R}^s$ функция f называется *квазивогнутой* на A, если при всех $x, y \in A$, $z \in [x, y]$ выполняется неравенство

$$f(z) \geq \min\{ f(x), f(y) \}.$$

В согласии с определениями 1–3 всякая вогнутая и всякая псевдовогнутая функция квазивогнута. Вогнутые и псевдовогнутые функции не могут иметь точку перегиба; квазивогнутая функция (а также родственные ей строго и сильно квазивогнутые функции) точку перегиба иметь могут [1].

Для различных приложений может представлять интерес функция, занимающая промежуточное положение между псевдовогнутыми и квазивогнутыми функциями (необязательно дифференцируемая, но не имеющая точек перегиба). Этим требованиям соответствует функция, удовлетворяющая следующему определению.

Определение 4. Заданная на выпуклом множестве $A \subset \mathbf{R}^s$ функция f называется ω *– вогнутой* на A, если для любых фиксированных $x, y \in A$ при условии $f(y) \geq f(x)$ можно указать величину $\omega = \omega(x, y) \in (0, 1]$ такую, что при всех $z \in [x, y]$ выполняется неравенство

$$\|y - x\|\big[f(z) - f(x)\big] \geq \omega\|z - x\|\big[f(y) - f(x)\big].$$

Определение 4 при условии $\omega \equiv 1$ совпадает с определением 1 вогнутой функции, причем каждая ω – вогнутая функция в согласии с определениями 3, 4 квазивогнута.

Следующая лемма устанавливает связь между ω – вогнутыми и псевдовогнутыми функциями.

Теорема. 1^0 . *Если функция* f *дифференцируема и* ω *– вогнута на выпуклом множестве* $A \subset \mathbf{R}^s$, *то она псевдовогнута на* A .

2^0 . *Если функция* f *псевдовогнута на выпуклом множестве* $A \subset \mathbf{R}^s$, *то она* ω *– вогнута на* A .

Доказательство. 1^0. Предположим от противного, что для некоторой пары точек $x, y \in A$, $x \neq y$ выполняется соотношение

$$\langle \nabla f(x), y - x \rangle \leq 0, \quad f(y) > f(x).$$

Тогда для любой точки

$$z = x + \rho(y - x) \in A, \quad \rho > 0,$$

справедливо утверждение

$$\frac{f(z) - f(x)}{\rho} \leq \frac{o(\rho)}{\rho},$$

и для достаточно малых $\rho = \frac{\|z - x\|}{\|y - x\|} > 0$ из-за соотношения $\lim\limits_{\rho \to 0} \frac{o(\rho)}{\rho} = 0$ выполняется неравенство

$$\frac{\|y - x\|}{\|z - x\|}\left[f(z) - f(x) \right] \leq \frac{\omega(x, y)}{2}\left[f(y) - f(x) \right],$$

что невозможно ввиду ω – вогнутости функции f; это доказывает первое утверждение леммы.

2^0. Предположим от противного, что для некоторой пары точек $x, y \in A$, $f(y) \geq f(x)$ при любом $\omega \in (0, 1]$ можно указать точку $z^\omega \in [x, y]$, удовлетворяющую неравенству

$$\|y - x\|\left[f(z^\omega) - f(x) \right] < \omega \|z^\omega - x\|\left[f(y) - f(x) \right]. \qquad (1)$$

Если выполняется хотя бы одно из равенств

$$z^\omega = x, \quad f(y) = f(x),$$

то из неравенства (1) следует утверждение

$$f(z^\omega) < f(x) = \min\{ f(x), f(y) \},$$

тогда как ввиду квазивогнутости псевдовогнутой функции f при любых $y \in A_0(x)$, $z^{\omega} \in [x, y]$ выполняется противоположное соотношение

$$f(z^{\omega}) \geq \min\{f(x),\ f(y)\} = f(x).$$

Тем самым с необходимостью из утверждения (1) вытекает утверждение

$$f(y) > f(z^{\omega}) \geq f(x), \quad x \neq y, \quad z^{\omega} \in (x, y],$$

$$\frac{f(z^{\omega}) - f(x)}{\|z^{\omega} - x\|} < \omega \frac{f(y) - f(x)}{\|y - x\|}, \tag{2}$$

где первое неравенство вытекает из (1), поскольку выполняются соотношения

$$0 < \omega, \ \frac{\|z^{\omega} - x\|}{\|y - x\|} \leq 1.$$

Ввиду компактности отрезка $[x, y]$ из произвольной сходящейся последовательности величин

$$\{\omega(t)\}_{t=1}^{\infty} \subset (0, 1], \quad \lim_{t \to \infty} \omega(t) = 0$$

можно выбрать такую бесконечную подпоследовательность

$$\{\omega(t)\}_{t \in T}, \quad \lim_{t \in T} \omega(t) = 0, \quad T \subset \{1, 2, ...\}, \quad |T| = \infty,$$

что последовательность векторов $\{z^{\omega(t)}\}_{t \in T} \subset (x, y]$ сходится к точке отрезка $[x, y]$:

$$\lim_{t \in T} z^{\omega(t)} = z^{*} \in [x, y].$$

Если $z^{*} \neq x$, то переходя в последнем неравенстве *следствия* к пределу по $\omega = \omega(t)$, $t \in T$, с учетом последних двух предельных соотношений можно утверждать

$$f\left(z^*\right) < f\left(x\right) = \min\left\{ f\left(x\right),\, f\left(y\right)\right\},\quad z^* \in \left(x, y\right],$$

что опять-таки противоречит квазивогнутости псевдовогнутой функции f, так что с необходимостью $z^* = x$. Но тогда утверждение (2) влечет соотношения

$$z^{\omega(t)},\ y \neq x,\quad \frac{f\left(z^{\omega(t)}\right) - f\left(x\right)}{\left\| z^{\omega(t)} - x \right\|} < \omega\left(t\right)\frac{f\left(y\right) - f\left(x\right)}{\left\| y - x \right\|},\quad t \in T;$$

переходя в последних неравенствах к пределу по $t \in T$, с учетом двух последних предельных соотношений можно утверждать, что выполняется неравенство $\left\langle \nabla f\left(x\right), y - x \right\rangle \leq 0$. Это, в согласии с определением псевдовогнутой функции, влечет неравенство $f\left(x\right) \geq f\left(y\right)$, тогда как согласно (2) из неравенства (1) следует противоположное утверждение $f\left(y\right) > f\left(x\right)$. Из противоречия следует второе утверждение леммы. Лемма доказана.

Литература

1. *Базара М., Шетти К.* Нелинейное программирование. М.: Мир, 1986.

НЕКОТОРЫЕ АЛГОРИТМЫ РЕШЕНИЯ МИНИМАКСНОЙ ЗАДАЧИ ТЕОРИИ РАСПИСАНИЙ

Д.Р. Гончар

Рассматривается минимаксная задача составления расписания минимальной длины без прерываний для многопроцессорной системы. Для решения данной задачи предложен метод ветвей и границ и алгоритмы для проверки качества полученного решения.

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 заклю-

чается 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 [1], методы математического программирования [2], метод ветвей и границ [3, 4], муравьиные алгоритмы [5], поиск с запретами [6], вероятностные алгоритмы [7], генетические алгоритмы [8], метод имитации отжига [9], различные эвристические алгоритмы [10], алгоритмы агрегирования [11] и др.

2. Метод ветвей и границ

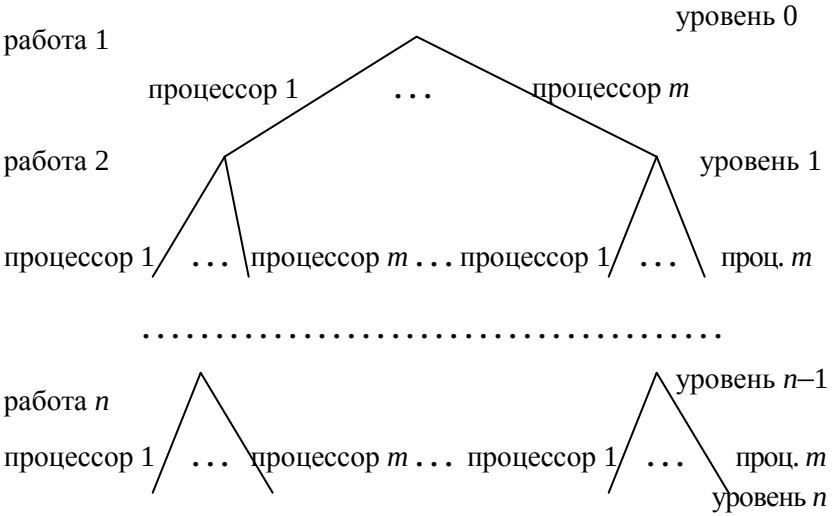
Для решения поставленной задачи предлагается метод ветвей и границ, основанный на результатах работы [3, 4].

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, соот-

ветствующим 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 [12] (например, одностороннего или фронтального ветвления).



2.2. Нижняя оценка

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

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

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. Необходимость поддерживать определенные связи (зависимости) между полученными подзадачами объясняется следующими двумя причинами:

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 [15, 16].

Из известных параллельных методов решения задач дискретной оптимизации, одними из наиболее известных и широко применяемых являются два подхода: метод оптимального поиска (best-first search, сокращенно – BFS), также называемый методом фронтального ветвления, и метод поиска в глубину (depth-first search, сокращенно DFS), также называемый методом одностороннего ветвления.

По методу DFS на каждом шаге для ветвления выбирается одна из концевых вершин дерева, положенных на предыдущем шаге. Если полученная вершина стала допустимым решением или отсеяна по правилам отсева, то производится возврат на один уровень вверх, и ветвлению подвергаются очередные, подходящие для этого вершины.

По методу BFS вершина для ветвления выбирается не только из числа вершин, полученных на предыдущем шаге, а из числа всех вершин, порожденных ранее. Для выбора вершины разными авторами предлагаются стратегии [2], основанные на оценке того, насколько быстро ветвление данной вершины может привести к решению.

В данной работе применен параллельный алгоритм поиска в глубину, предложенный в [10]. На первом шаге этого алгоритма один из процессоров получает начальную вершину и осуществляет поиск в глубину, пока количество ветвлений станет не меньше числа свободных процессоров. После этого процесс, не имеющий задания, выбирает некоторый другой процесс и отправляет ему запрос на получение задания. Выбор процесса может осуществляться различными способами. Выбранный процесс называется *процессом-донором*, а процесс, пославший запрос, называется *процессом-акцептором*. Если процесс-донор располагает необработанными концевыми вершинами, то он посылает несколько таких вершин процессу, пославшему запрос. В противном случае, процесс-акцептор повторяет запрос, но уже к другому процессору. Выполнение приложения завер-

шается тогда, когда в системе не осталось необработанных концевых вершин.

В [7] рассматриваются три различные стратегии выбора процесса-донора:

Асинхронная циклическая стратегия (asynchronous round robin, сокращенно ARR) состоит в том, что процессор, оставшийся без задания, запрашивает процессор, имеющий номер, вычисляемый по формуле *label* mod *p*, где через *p* обозначено общее число процессоров, участвующих в вычислении, а номер *label* увеличивается на 1 при каждом новом запросе. Недостатком этой схемы является возможность обращения к одному и тому же процессору-донору нескольких процессоров-акцепторов одновременно, что приводит к потере времени на ожидание обработки посланного запроса.

Глобальная циклическая стратегия (global round robin, сокращенно GRR), при которой каждый процессор считывает номер процессора-донора из разделяемой всеми процессорами переменной target, которая увеличивается на 1 после каждого запроса. Это позволяет избежать одновременных запросов к одному и тому же процессору-донору со стороны нескольких процессоров-акцепторов. Однако, при такой схеме узким местом становится общая переменная target.

Случайная стратегия (Random polling, сокращенно RP), при которой процесс-донор выбирается случайным образом.

Вычислительные опыты показывают, что с точки зрения масштабируемости наилучшие результаты дает стратегия RP, а наихудшие – стратегия GRR. Низкая эффективность стратегии GRR объясняется необходимостью считывания общей переменной target, конкурентный доступ со стороны нескольких процессоров к которой приводит к большим задержкам. Авторы предлагают аналитическую вероятностную модель для анализа

эффективности предложенных стратегий распараллеливания, которая объясняет результаты экспериментов.

Начальное распределение вычислений на NPr физических процессоров системы представляется разумным осуществить, когда число ветвей дерева решений станет не менее NPr. Если при этом часть ветвей остается не распределенной на процессоры, то они становятся в очередь на обработку к первому из освободившихся процессоров.

Обмен рекордами между разными физическими процессорами представляется разумным осуществлять не чаще обновления каждого такого значения в каждой из обрабатываемых ветвей. При этом известны несколько подходов к устройству таких обменов, каждый из которых имеет как свои сильные стороны, так и некоторые недостатки (накладные расходы).

Так, при использовании схемы «управляющий-рабочие» с выделенным главным процессом, который принимает сообщения обо всех значимых событиях в ходе вычислений (обновление рекордов в каждом процессе, завершение обработки, возникновение ошибок и т.п.) и, со своей стороны, передает всем остальным действующим процессам соответствующие обновленные сведения (например, обновленную верхнюю границу или рекорд вычислений), производительность главного процесса в ряде случаев может стать узким местом всей системы и время на согласование работы ветвей при этом превышает сбережение времени от самого согласования.

При использовании ряда распределенных схем согласования работы процессов без единого главного процесса, подобные узкие места возникают менее часто и затрагивают меньшее число взаимодействующих процессов. С другой стороны, действительно важные сообщения (к примеру, о достижении глобального оптимума), в этом случае несколько дольше доставляются до каждого из продолжающих об-

работку своих ветвей процессов, что также влияет на общую длительность вычислений задачи.

4. Стратегии обхода дерева в методе ветвей и границ

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.

5. Контрольные расчеты

Для проверки качества распараллеливания разработан алгоритм полного обхода дерева решения. В нем вводится структура данных текущего распределения заданий на процессоры, с начальным распределением всех заданий на 1-й процессор. Для данного распределения (расписания) вычисляется длительность выполнения заданий на каждом процессоре, выбирается наибольшая из этих величин и запоминается отдельно как начальное приближение общего минимума полного расписания.

Далее планомерно производится увеличение номера процессора, на который распределена последняя из задач. При этом если полученное значение превышает общее количество моделируемых процессоров, то номер процессора в текущей строке сбрасывается до наименьшего значения

(единицы), а величина номера процессора в строке на единицу с меньшим номером увеличивается на единицу, опять производится проверка и т.д. Данный процесс продолжается вложенным образом, пока не будет получено очередное ближайшее допустимое распределение, либо получен признак завершения обхода дерева. В первом случае вычисляется локальный максимум загрузки по процессорам в рамках данного расписания (т.е. полная длина расписания), который сравнивается с общим минимумом и запоминается наилучшее (наименьшее из двух) решение. Во втором случае вычисления завершаются.

В дальнейшем планируется сравнить длительность работы однопроцессорного и многопроцессорного варианта расчетов, а также длительность работы алгоритма полного перебора (для небольших размерностей задачи) с длительностью работы по методу ветвей и границ (как в однопроцессорном, так и в многопроцессорном варианте).

Литература

1. *Гончаров Е.Н., Кочетов Ю.А.* Вероятностный поиск с запретами для дискретных задач безусловной оптимизации // Дискретный анализ и исследования операций. Сер. 2. 2002. Т. 9. № 2. С. 13–30.

2. *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.

3. *Алексеев О.Г.* Комплексное применение методов дискретной оптимизации. М.: Наука, 1987.

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

5. *Штовба С.Д.* Муравьиные алгоритмы // ExponentaPro. Математика в приложениях. 2003. № 4(4). С. 70–75.

6. *Glover F., Laguna M.* Chapter 3: Tabu search/ Ed. *R. Colin Reeves*, Modern Heuristics Techniques for Combinatorial Problems. Oxford, Blackwell Scientific Publications, 1993. P. 70–150.

7. *Raghavan R.* Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs // J. Computer and System Sciences. 1988. V. 37. P. 130–143.

8. *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.

9. *Shen C., Pao Y., Yip P.* Scheduling multiple job problems with guided evolutionary simulated annealing approach // Proc. First IEEE Conf. on Evolutionary Computations. Orlando, 1994. P. 702–706.

10. *Brucker P.* Scheduling Algorithms. Heidelberg, Springer, 2001.

11. *Красовский Д.В.* Алгоритмы решения задачи составления оптимального расписания без прерываний. Диссертация на соискание ученой степени канд. физ.-матем. наук. М.: МФТИ, 2007.

12. *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.

13. *Кормен Т., Лейзерсон Ч., Ривест., Штайн К.* Алгоритмы: построение и анализ. М.: Вильямс, 2005.

14. *Гончар Д.Р.* Параллельная реализация мультиоценочного алгоритма составления многопроцессорного расписания без прерываний. // Некоторые алгоритмы планирова-

ния вычислений и методы многокритериальной оптимизации для многопроцессорных систем. М.: ВЦ РАН, 2014. С. 21–31.

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

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

АЛГОРИТМ ОПТИМИЗАЦИИ СТРУКТУРЫ БАЗЫ ДАННЫХ С НЕОГРАНИЧЕННЫМ ЧИСЛОМ ФАЙЛОВ И МИНИМАЛЬНОЙ ИЗБЫТОЧНОСТЬЮ ИНФОРМАЦИИ ДЛЯ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

С.Н. Мирошник

Рассматривается проблема избыточности информации в базе данных реального времени. Такая БД необходима для минимизации времени поиска информации при решении задач в реальном времени. В данной работе подробно исследуются различные аспекты алгоритма для уменьшения всевозможных переборов.

1. Введение

Настоящая работа является продолжением работ [1-4], посвященных методам создания такой структуры БД, в которой избыточность информации минимальная. Данная БД есть составляющая часть большой системы реального времени. В зависимости от того, насколько хорошо создана эта система, во многом зависит эффективность решение конкретных задач в реальном времени. Необходимость минимизации избыточности информации вызвана уменьшением времени поиска информации в БД. В данной работе основное внимание уделено математической проблеме в рамках определенной постановки задачи.

2. Постановка задачи

Задано множество независимых программных модулей $M_1,\ldots, M_n$. Каждый модуль использует поля из общего набора полей $\phi_1,\ldots, \phi_r$ файла Ф, все поля пронумерованы натуральным рядом чисел $1,\ldots, r$. Запись модуля M_i есть набор полей $\{\phi\}_i$, идущих подряд из набора $\phi_1,\ldots, \phi_r$. Дли-

на записи модуля M_i есть l_i. Записи разных модулей могут использовать одинаковые поля. Природа полей и их длина не рассматривается. В данной работе в записи модулей входят только используемые поля. Неиспользуемые поля не рассматриваются, так как они не влияют на работу алгоритма. Число файлов в БД неограниченно.

Идея минимизации избыточности информации состоит в том, чтобы определенным способом объединить модули в различные группы, оформляемые в виде файлов со своими атрибутами. Объединение осуществляется таким образом, чтобы каждый модуль попал только в один файл. Длина записи файла определяется количеством различных полей всех модулей, образующих этот файл. Теперь каждый модуль для своей работы использует все поля записи своего файла. Различные модули могут использовать одинаковые поля файла. Объединение модулей в файлы следует проводить таким образом, чтобы минимизировать избыточность информации, как в каждом файле, так и для всей совокупности файлов.

3. Основные определения и обозначения

В работах [1-4] предлагается определение избыточности информации трех типов. Внутрифайловая избыточность образуется из-за разности длины l_i модуля M_i, входящий в файл и длиной L самого файла F. Эта избыточность обозначается как $I_1(F)$ и вычисляется по формуле:

$$I_1(F) = L \cdot t - \sum_{i=1}^{t} l_i , \qquad (1)$$

где t – число модулей в файле F.

Полная внутрифайловая избыточность есть:

$$I_1 = \sum_{s=1}^{k} (L_s \cdot s - \sum_{i=1}^{s} l_i^s) \qquad (2)$$

для всех файлов $F_1,\ldots, F_k$.

Далее, среди полей записей файлов $F_1,\ldots, F_k$ есть повторяющиеся поля. Эти поля образуют межфайловую избы-

точность, которая обозначается как I_2 и вычисляется по формуле:

$$I_2 = \sum_{s=1}^{k} L_s - r \qquad (3)$$

Таким образом, файлы необходимо сформировать так, чтобы минимизировать как суммарную внутрифайловую информацию I_1, так и число повторяющихся полей в файлах $F_1,\ldots, F_k$, т.е. I_2 . Естественно, I_2 вычисляется только после того, как набор файлов построен.

Сумма $I = I_1 + I_2$ – есть количество избыточной информации БД, состоящей из файлов $F_1,\ldots, F_k$.

Объединение модулей в файлы может быть реализовано разными способами. В отличие от работы [1] в данной работе используется вариант, предложенный в работе [4]. Вводится понятие близости двух модулей M_i и M_j, а также близость модуля M_i к набору модулей $\{M\}_s$. Эта близость определяется сравнением двух типов избыточностей, которое определяет возможное включение модуля M_{s+1} в набор $\{M\}_s$. Здесь и далее s – число модулей в наборе $\{M\}_s$.

Включение модуля M_{s+1} в набор $\{M\}_s$ изменяет внутрифайловую избыточность на величину $\Delta I_1 = I_1^{s+1} - I_1^s$, где I_1^s – внутрифайловая избыточность набора $\{\mathrm{M}\}_\mathrm{s}$, I_1^{s+1} – внутрифайловая избыточность набора $\{M\}_{s+1}$ после включения M_{s+1} в $\{M\}_s$. С другой стороны важной информацией для близости M_{s+1} к $\{M\}_s$ является количество совпадающих полей модуля M_{s+1} и $\{M\}_s$. Для вычисления совпадающих полей можно воспользоваться формулой (3): I_2 ($\{M\}_{s+1}$, $\{M\}$) в несколько модифицированном виде.

Определение. Модуль M_{s+1} и набор $\{M\}_\mathrm{s}$ являются близкими, и может быть включен в состав набора $\{M\}_s$, если

$$\Delta I_1 \le I_2(M_{s+1}, \{M\}) \qquad (4)$$

Введем понятие внешний модуль и внутренний.

Рассмотрим модуль M^*. Его поля $\{\phi\}^*$ и длина l^*. Модуль M^* является внешним относительно некоторого модуля M (его поля $\{\phi\}$ и длина l), если $\{\phi\} \cap \{\phi\}^* \neq 0$, т.е. среди полей $\{\phi\}^*$ есть поля, отличные от полей $\{\phi\}$, также M^* является внешним относительно набора $\{M\}$. Модуль M^* является внутренним относительно M, если $\{\phi\}^* \subset \{\phi\}$. Аналогично M^* является внутренним относительно набора $\{M\}$. Обозначим символом d^* количество внешней части полей модуля M^* относительно полей набора $\{M\}$. Таким образом, длина l^* модуля M^* делится на две части ($d^* \neq 0$): $l^* - d^*$ целиком содержится в l или в L. Отсюда, в частности, следует, что

$$I_2(M^*, M) = l^* - d^*$$

есть число совпадающих полей модулей M^* и M, или M^* и $\{M\}$.

Предварительные замечания

Алгоритм состоит из выполнения последовательности шагов с использованием многократного перебора модулей $M_1,\ldots, M_n$ для поиска близких модулей.

Пусть задан модуль M длиной l. Оставшиеся модули можно разделить на две группы: внутренние относительно M и внешние относительно модуля M. Внешние модули имеют внешнюю часть, т.е. для них $d \neq 0$. Эти внешние модули могут быть упорядочены по возрастанию $d_i : d_{i+1} \geq d_i$. Естественно, для всех внутренних модулей, $d = 0$ и их упорядочивание не имеет смысла. Следует отметить, что внешний модуль M^* длиной l^* и $d^* \neq 0$ относительно M может оказаться внутренним относительно набора модулей $\{M\}$. И тогда $d^* = 0$ и $I_2(M^*, \{M\}) = l^*$.

Далее, многократный перебор модулей для поиска близких к M значительно упрощается, если выполнить несколько замечаний и утверждений. Введем некоторые обозначения: M^{+-} является близким, M^{-} является неблизким.

Утверждение 1.

Для всех модулей M_i для которых $l_i < [1/2 \cdot l]$ и $d_i = 0$: $M_i \rightarrow M^-$, т.е. M_i не являются близкими к M. Очевидно, что в этом случае $I_2(M; M) \le [1/2 \cdot l]$ при этом $\Delta I_1(M_i; M) > [1/2 \cdot l]$. Отсюда: $\Delta I_1(M_i; M) > I_2(M_i; M)$.

Утверждение 2.

Пусть модуль M_i являются внутренним модуль к M или к набору $\{M^+\}_s$, причем $l_i \ge [1/2 \cdot l]$ или $l_i \ge [1/2 \cdot L_s]$. Тогда M_i является близким к M или $\{M^+\}_s$, т.е. $M_i \rightarrow M_i^+$.

Утверждение 3.

Рассмотрим модуль M_i, для которого $l_i = l$, $d_i \neq 0$. Тогда, $M_i \rightarrow M_i^+$ если $d_i \le [1/3 \cdot l]$.

Пусть $d_i = [1/3 \cdot l]$. Вычислим $\Delta I_1(M_i, M)$ и $I_2(M_i)$.

Получаем: $I_1(M_i, M) = (l+d_i)_2 - (l+l_i)$, но $I_1(M) = 0$, поэтому $\Delta I_1 = I_1(M_i, M)$.

Вычислим $I_2(M_i)$: $I_2(M_i) = (l+l_i) - (l+d_i)$.

Условие $I_2(M_i) \ge \Delta I_1$ преобразуется к виду $2l_i - l \ge 3d_i$. По условию $l_i = l$ и $d_i \le [1/3 \cdot l]$ получаем: M_i^+

Следствие. Для $l_i < l$ и $d_i = [1/3 \cdot l]$ получаем: M_i^-.

Утверждение 4.

Рассмотрим набор модулей $\{M^+\}_s$ ($M_1^+ = M$) таких, что $l = l_i$, $i = 1,\ldots, s$, $d_{i+1} = d_i + 1$ ($d_1 = 0$), $d_s =]1/2 \cdot l[$.

Тогда для $i = s+1, \ldots, n$: $M_{s+1}^-, \ldots, M_n^-$.

Доказательство

Покажем, что $\Delta I_1^{s+1} > I_2^{s+1}$ для $s + 1, \ldots, n$.

Вычислим I_1^s и I_1^{s+1}:

$I_1^s = (l + d_s) \cdot s - s \cdot l$ или $I_1^s = d_s \cdot s$.

$I_1^{s+1} = (l + d_{s+1}) \cdot (s+1) - (\sum_{i=1}^{s} l_i + l_{s+1}) \cdot$

Получаем: $\Delta I_1^{s+1} = (s+1)d_{s+1} - d_s \cdot s$.

Но $d_{s+1} = d_s + 1$, и $\Delta I_1^{s+1} = d_s + (s+1)$.

По условию $I_2^{s+1} = l - 1$, $d_s =]1/2 \cdot l[+ 1$.

Покажем, что $\Delta I_1^{s+1} > I_2$ т.е. $]l/2[+(s+1) > l-1$.

Нетрудно заметить, что $s =]1/2 \cdot l[+ 1$.

Теперь 1. l – четное: $2]1/2 \cdot l[= l$ и $l+2 > l-1$.

2. l – нечетное: $2]1/2 \cdot l[= l-1$ и $l+1 > l-1$.

Утверждение доказано.

Следствие.

1. Максимальное число близких модулей в наборе $\{M^+\}_s$ не превышает числа s.

2. При всех прочих условиях можно допустить: $l_i < l$, если при этом модули остаются близкими в наборе $\{M^+\}_s$, т.к. в этом случае ΔI_1^{s+1} только увеличивается, в то время как I_2^{s+1} уменьшается.

Утверждение 5.

Максимально возможное число модулей в наборе $\{M^+\}_s$ для $l_i < l$ и $d_i = 0$ есть

$\max |\{M^+\}| \le 1/2]1/2 \cdot l[\cdot (\ ]1/2 \cdot l[+1)$

Знак «=» нарушается, если $d_i \neq 0$.

Доказательство.

Число модулей M_i^+ для $l_i < l$ и $d_i = 0$ есть

$|l_i| = i$, $i = 1,\ldots, s$. Здесь $s =]1/2 \cdot l[$.

Сумма чисел натурального ряда:

$\sum_{i=1}^{s} |l_i| = 1/2]1/2 \cdot l[\cdot (]1/2 \cdot l[+1)$, то есть $\max \left|\{M^+\}_s\right| = \sum_{i=1}^{s} |l_i| \cdot$

В процессе перебора модулей в поисках близких модулей M^+ к набору $\{M^+\}$ может оказаться, что ранее отвергнутый модуль M^- становится близким, т.е. $M^- \to M^+$.

Утверждение 6.

Задан набор модулей $\{M^+\}_s$, длиной L_s. Рассмотрим модуль M_t длиной l_t и $d_t \neq 0$. Найдем условие, при котором $M_t \to M_t$.

Вычислим ΔI_1 для $\{M^+\}_s$ и M_t.

Для этого выражения $I_1^s(\{M\}_s)$ и $I_1^t(\{M\}_s, M_t)$ есть:

$$I_1^s(\{M^+\}_s) = L_s \cdot s - \sum_{i=1}^{s} l_i, \quad I_1^t(\{M^+\}_s \cup M_t) = (L_s + d_t)(s+1) - (\sum_{i=1}^{s} l_i + l_t) \cdot$$

Тогда $\Delta I_1 = (L_s - l_t) + d_t(s+1)$.

Вычислим: $I_2^t = l_t - d_t$.

Условие $\Delta I_1 > I_2^t$ есть:

$$L_s - 2l_t + 2d_t + d_t \cdot s \geq 0. \qquad (6)$$

Это неравенство есть условие, при котором $M_t \to M_t^-$

Найдем условие, при котором $M_t \to M_t^+$.

Утверждение 7.

Пусть $\exists M_{s+1}^+$ такой, что $d_{s+1} \geq d_t$ длиной l_{s+1}. В этом случае M_t становится внутренним модулем для набора $\{M^+\}_s \cup M_{s+1}^+$, для которого $d_t = 0$ и $I_2(M_t) = l_t$.

Здесь $d_t = 0$, т.к. из-за того, что модули упорядочены: $d_{s+1} \geq d_t$ и потому модуль M_t не добавляет новые поля в набор модулей $\{M_s\} \cup M_{s+1}$.

Вычислим $I_1(\{M^+\}_s \cup M_{s+1}^+)$ и $I_1(\{M^+\}_s \cup M_{s+1}^+, M_t)$.

$$I_1^{s+1}(\{M^+\}_s \cup M_{s+1}^+) = (L_s + d_{s+1})(s+1) - (\sum_{i=1}^{s} l_1 + l_{s+1}).$$

$$I_1^t(\{M^+\}_s \cup M_{s+1}^+, M_t) = (L_s + d_{s+1}) \cdot (s+2) - (\sum_{i=1}^{s} l_i + l_{s+1} + l_t).$$

Получаем: $\Delta I = L_s + d_{s+1} - l_t$.

Условие, при котором $M_t \to M_t^+$ есть: $\Delta I \leq I_2$ есть

$$L_s + d_{s+1} \leq 2l_t \text{ или } L_{s+1} - l_t \leq l_t \qquad (7)$$

Пусть модуль M_t внешний относительно $\{M^+\}_s$ является $M_t^-, (d_t \neq 0)$. Докажем, что добавления в набор $\{M^+\}_s$ модуля M_{s+1}, для которого $d_{s+1} \geq d_t$ преобразует $M_t^- \rightarrow M_t^+$, т.е. результат Утверждения 6 в результат Утверждения 7 для модуля M_t.

Из Утверждения 6 имеем:

$$L_s - 2l_t + 2d_t + d_t \cdot s > 0 \text{ т.е. } M \rightarrow M^-$$

Но для набора $\{M^+\}_s \cup M_{s+1}^+$ модуль M_t^- становится внутренним, т.е. d_t=0. Длина набора $\{M^+\}_s \cup M_{s+1}^+$ есть $L_{s+1} = L_s + d_{s+1}$. После несложных преобразований неравенство для набора $\{M\}_s \cup M_{s+1}^+ \cup M_t$ есть:

$$L_{s+1} - l_t \leq l_t.$$

т.е. получена та же формула, что и в Утверждении 7. Знак неравенства определяется из условия Утверждения 2, в котором, если $l_t \geq [1/2 \cdot L_{s+1}]$, тогда $L_{s+1} - l_t \leq l_t$. Получаем: $M_t^- \rightarrow M_t^+$.

Теперь модуль M_t^+ может быть включен в набор близких модулей $\{M^+\}_{s+1} \cup M_t^+$.

Описание алгоритма

Алгоритм построения множества близких для набора различных файлов $F_1, \ldots, F_n$ состоит из последовательности шагов.

Для формирования набора близких модулей $\{M^+\}$ требуется многократный перебор всех модулей. Однако, с учетом приведенных выше замечаний и утверждений, этот перебор может быть значительно сокращен.

1. Прежде всего, путем перебора находим модуль M, для которого $l = \max(l_1, \ldots, l_n)$

2. Все модули разделим на две группы: внутренние $\{M\}^*$ относительно M, для которых $d = 0$, и внешние $\{M\}^{**}$ относительно M, для которых $d \neq 0$.

Согласно Утверждению 1 все внутренние модули можно разделить на $\{M^{+}\}^{*}$ и $\{M^{-}\}^{*}$. Для этого необходимо выполнить условия на длины этих модулей, указанных в Замечании 1. Путем перебора внутренних модулей находим $\{M^{+}\}^{*}$. Близкие модули из набора $\{M^{+}\}^{*}$ определяются только относительно M и не зависит от уже найденных близких к M группе внутренних модулей. Максимальное возможное число близких внутренних модулей в наборе $\{M^{+}\}^{*}$, у которых $l_i \le l$ определяется в Утверждении 5.

3. Рассмотрим оставшиеся внешние модули, у которых $d \neq 0$.

Упорядочим эти внешние модули в порядке возрастания внешней части этих модулей, т.е. $d_{i+1} \ge d_i$.

Согласно Утверждению 4 эти модули ограничением на числа d делятся на две части: в первой части это могут быть модули, близкие к M во второй части это модули, которые не могут быть близкими к M, и потому при переборе модулей их логично не рассматривать.

Возможное число модулей первой части определяется в Утверждении 5.

4. В процессе определения близких модулей из первой части, некоторые из них могут быть отвергнуты. Но, согласно Утверждению 7, некоторые из них при определенных условиях могут оказаться близкими по набору близких модулей $\{M^{+}\}$. Отсюда следует, что, как только найден неблизкий модуль из первой части, а следующим модуль с большим d оказывается близким к набору $\{M^{+}\}$, этот неблизкий модуль следует проверить на его возможную близость к найденному набору.

После того, как сформирована группа близких модулей $\{M^{+}\}$ можно вычислить $I_1(\{M^{+}\})$. Исключим эту группу из общего набора модулей $M_1,\ldots, M_n$ и повторим процедуры, описанные в п.п. 1-4 для формирования других близких групп модулей.

Пусть сформулированы группы модулей $F_1,\ldots, F_k$. Для каждой вычислим $I_1(F_1),\ldots, I_1(F_k)$. После этого вычисляется межфайловая избыточность $I_2(F_1,\ldots, F_k)$.

Сумма: $I = \sum_{i=1}^{k} I_1(F_i) + I_2(F_1,...,F_k)$ есть показатель качества построенной БД, состоящей из файлов $F_1,\ldots, F_k$.

Учитывая неоднозначность выбора первого модуля (п. 1) может быть построено множество различных БД с их показателями качества, из которых можно выбрать наилучшую БД.

Литература

1. *Мирошник С.Н., Фуругян М.Г.* Оптимизация структуры базы данных реального времени. // Некоторые алгоритмы планирования вычислений в многопроцессорных системах. М.: ВЦ РАН, 2012. С. 24-37.
2. *Мирошник С.Н.* Алгоритм оптимизации базы данных реального времени для двух файлов // Некоторые алгоритмы планирования вычислений и оптимизации структуры баз данных в многопроцессорных системах. М.: ВЦ РАН, 2013. С. 41-48.
3. *Мирошник С.Н.* Алгоритмы оптимизации базы данных реального времени для неиспользованных полей модулей. // Некоторые алгоритмы планирования и методы многокритериальной оптимизации для многопроцессорных систем. М.: ВЦ РАН, 2014. С. 32-40.
4. *Мирошник С.Н.* Алгоритмы оптимизации структуры базы данных реального времени с минимальной избыточностью информации. // Некоторые алгоритмы составления расписаний в многопроцессорных системах М.: ВЦ РАН, 2015. С. 25-34.

Содержание

Научное издание

Фуругян М.Г., Гончар Д.Р., Мирошник С.Н., Рабинович Я.И.

Некоторые алгоритмы планирования вычислений в многопроцессорных системах

Под ред. кандидата физ.-матем. наук Л.Л. Вышинского

Техническая редакция и оригинал-макет авторов

Подписано в печать 10.01.2017

Тираж 50 экз.

Заказ 17-01

Издано ФИЦ ИУ РАН,

119333, Москва, ул. Вавилова, д. 44 кор. 2

ФИЦ ИУ РАН

ISBN 978-5-91993-070-9