УЧРЕЖДЕНИЕ РОССИЙСКОЙ АКАДЕМИИ НАУК
ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР
ИМ. А.А. ДОРОДНИЦЫНА

НЕКОТОРЫЕ АЛГОРИТМЫ ПЛАНИРОВАНИЯ ВЫЧИСЛЕНИЙ И ОРГАНИЗАЦИИ КОНТРОЛЯ В СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ

ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР ИМ. А.А. ДОРОДНИЦЫНА
РОССИЙСКОЙ АКАДЕМИИ НАУК
МОСКВА 2011

УДК 519.86

Ответственный редактор
канд. физ.-матем. наук Я.И. Рабинович

В сборнике содержится ряд алгоритмов планирования вычислений в многопроцессорных систем реального времени и методов анализа интервальных графов. Приведено доказательство 2-интервальности графов, имеющих единственный чередующийся цикл, и описано строение графов с единственным циклом без триангуляторов. Для случая смешанного набора работ приведен алгоритм, минимизирующий максимальное запаздывание. Рассмотрена задача составления допустимого расписания с прерываниями в многопроцессорной системе в случае, когда длительности выполнения работ зависят от количества выделенного им ресурса. Разработан псевдополиномиальный алгоритм построения многопроцессорного оптимального по быстродействию расписания без прерываний и переключений. Определено необходимое число процессоров для эффективного распараллеливания алгоритма. Рассмотрена задача оптимизации структуры подсистемы контроля в вычислительных системах реального времени.

Ключевые слова: многопроцессорная система, оптимальное расписание, прерываемые и непрерываемые работы, ресурс, контроль, интервальный граф, чередующийся цикл, триангулятор.

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

Научное издание

2-ИНТЕРВАЛЬНОСТЬ ГРАФОВ С ЕДИНСТВЕННЫМ ЧЕРЕДУЮЩИМСЯ ЦИКЛОМ

В.П. Козырев, Е.А. Соколова

В работе приведено доказательство 2-интервальности графов, имеющих единственный чередующийся цикл.

1. Основные определения и утверждение

Создание систем жесткого реального времени и автоматизация их создания требует исследования структур графов определенного вида. В таких системах осуществляется назначение директивных сроков начала и окончания обработки конкретных вычислительных задач. Близкими задачами являются упаковка в контейнеры, динамическое распределение памяти, задача о разбиении, задача о рюкзаке и другие.

Рассматриваются неизоморфные графы $G(V,E)$ без петель и кратных ребер (т.е. простые графы), имеющие n вершин, $n = |V|$, для удобства изложения материала будем вводить нумерацию вершин. Ребра дополнительного графа $\overline{G}$ будем называть неребрами графа G и наоборот, ребра графа G – это неребра графа $\overline{G}$. Будем использовать следующие обозначения: (u,v) – ребро, соединяющее вершины u и v, $\overline{(u,v)}$ – неребро, соединяющее вершины u и v. Граф называется интервальным, если его вершинам можно взаимно однозначно поставить в соответствие отрезки на вещественной прямой таким образом, что две вершины смежны тогда и только тогда, когда отрезки, соответствующие этим вершинам, пересекаются. Если граф и его дополнительный граф являются интервальными, то граф будем называть 2-интервальным. Клика K графа G – это полный подграф графа, замкнутый по включению.

Теорема 1 [1]. Система клик $K_1,\ldots,K_n$ множества вершин графа G является системой клик интервального графа G тогда и только тогда, когда существует такая нумерация клик, что если $K_i \cap K_j \neq \emptyset$, то это пересечение принадлежит всем кликам с номерами от i до j.

Определение 1. Чередующейся цепью (кратко П-цепью) графа G называется последовательность ребер и неребер, в которой за каждым ребром следует неребро и за каждым неребром следует ребро; имеются два крайних элемента. Замкнутая чередующаяся цепь называется чередующимся циклом (кратко П-циклом).

Графы, не содержащие П-циклов, охарактеризованы в работе [3]. Показано, что вершины связного графа G можно разбить на множества $V_1,V_2,\ldots,V_k$ следующим образом: вершины множества V_1 имеют в графе G максимальную степень $(n\text{-}1)$. Вершины множества V_2 имеют степень ноль в графе $G \setminus V_1$, полученном из графа G удалением вершин множества V_1 и инцидентных им ребер. Вершины множества V_3 имеют в графе $G\setminus\{V_1,V_2\}$ максимальную степень и смежные со всеми оставшимися вершинами, и так далее. Таким образом, вершины множеств с нечетными номерами смежны между собой и со всеми вершинами из множеств с большими номерами; вершины множеств с четными номерами смежны только с вершинами из множеств с меньшими нечетными номерами.

Пусть G – связный граф без П-циклов, множество вершин которого разбито на множества $V_1,V_2,\ldots,V_k$, $|V_i| = n_i$, $1 \le i \le k$, L – граф на множестве вершин a, b, c, d, содержащий единственный П-цикл. Таким образом, граф L имеет вид $(a,b),(b,c),(c,d),\overline{(a,c)},\overline{(a,d)},\overline{(b,d)}$. Причем $V(G) \cap V(L) = \emptyset$. Введем определение. Граф F – граф, получающийся в результате доопределения ребер и неребер на вершинах графов G и L таким образом, что граф F содержит единственный П-цикл. Степенью вершины v в графе G по графу L называется число вершин графа L, смежных с вершиной v.

Приведем **процедуру** α, описывающую строение графа F [2]. Граф F строится одним из трех способов:

1) степень вершин из множеств с нечетными номерами по графу L равна 4, а степень вершин из множеств с четными номерами по графу L равна нулю.
2) найдется такой номер i, что вершины из множеств с нечетными номерами от 1 до (i-1) имеют степень по графу L, равную 4, а вершины из множеств с четными номерами от 2 до (i-1) имеют степень по графу L, равную нулю. Здесь возможны два случая:
   a) если $i = k$, то при нечетном k существует ровно одна вершина степени 0 по графу L, остальные вершины в этом множестве имеют степень по графу L, равную 4; при четном k существует ровно одна вершина, степень которой по графу L равна 4, остальные вершины в этом множестве имеют степень 0 по графу L.
   b) если $1 \le i \le k$, то в множестве с номером i существует t вершин степени 2 по графу L (смежных с вершинами b и c), $1 \le t \le n_i$, остальные вершины в этом множестве имеют степень по графу L, равную либо 4 (i – нечетно), либо 0 (i – четно). У вершин множеств с номерами, большими i, степень по графу L равна 2 (также смежных с вершинами b и c).

2. Основной результат

Теорема 2. Любой граф F с единственным П-циклом является 2-интервальным.

Для доказательства построим для каждого случая, описанного в процедуре α, клики графов F и $\overline{F}$ и упорядочим их так, чтобы были выполнены условия теоремы 1.

Рассмотрим первый случай. Построим табл. 1, строками которой являются множества V_i, число вершин в множестве, степень вершин графа F по графу L и степень вершин графа $\overline{F}$ по графу $\overline{L}$.

Таблица 1. Степени вершин для пункта 1 процедуры α

	V_1	V_2	V_3	…	V_k	
	n_1	N_2	n_3	…	n_k	
					$k = 2k'$	$k = 2k'+1$
F	4	0	4	…	0	4
$\overline{F}$	0	4	0	…	4	0

Построим все клики графа F. Вершины множества V_1 смежны со всеми вершинами графов G и F, поэтому в каждой клике будут вершины этого множества. Исходя из этого, строим клики, начиная с множества V_1. Первую группу клик составляют все вершины множества V_1 и одна вершина из множества V_2, число таких клик равно числу вершин в множестве V_2, то есть n_2. Других вершин в эти клики добавить нельзя, так как вершины множества V_2 смежны только с вершинами V_1 и не смежны со всеми остальными вершинами графа F. Добавляя к вершинам множества V_1 вершины множества V_3, получим полный подграф, а добавляя одну вершину из множества V_4, получим клику, других вершин добавить нельзя, так как они не будут смежны с вершиной множества V_4. Построенные клики образуют вторую группу клик, число которых равно n_4. Далее проводим аналогичные рассуждения. В зависимости от четности числа k рассмотрим, что будет происходить в последних множествах. Пусть k четно. Тогда получаем группу клик, состоящую из вершин множеств с нечетными номерами $V_1, V_3, \ldots, V_{k-1}$ и одной вершины из множества V_k (n_k клик). Вершины графа L смежны со всеми вершинами множеств с нечетными номерами, при этом в самом графе L попарно смежны вершины a и b, b и c, c и d. Следовательно, получим три клики, состоящие из всех вершин множеств с нечетными номерами и одной из пар вершин: $\{a,b\}$, $\{b,c\}$, $\{c,d\}$. Выпишем все клики графа и упорядочим их в соответствии с условием тео-

ремы 1. Клики упорядочим по группам, клики внутри каждой группы могут быть упорядочены произвольным образом, так как каждая клика состоит из некоторой общей для данной группы части вершин и одной вершины, которая не содержится ни в какой другой клике графа.

$$
\begin{aligned}
&V_1 \cup \{v\}, \quad v \in V_2, \quad \textit{клик}\, n_2,\\
&V_1 \cup V_3 \cup \{v\}, \quad v \in V_4, \quad \textit{клик}\, n_4,\\
&\ldots\\
&\left.\begin{aligned}
&V_1 \cup V_3 \cup \ldots \cup V_{k-1} \cup \{v\}, \quad v \in V_k, \quad \textit{клик}\, n_k\\
&V_1 \cup V_3 \cup \ldots \cup V_{k-1} \cup \{a,b\}\\
&V_1 \cup V_3 \cup \ldots \cup V_{k-1} \cup \{b,c\}\\
&V_1 \cup V_3 \cup \ldots \cup V_{k-1} \cup \{c,d\}
\end{aligned}\right\} k = 2k',\\
&\left.\begin{aligned}
&V_1 \cup V_3 \cup \ldots \cup V_{k-2} \cup \{v\}, \quad v \in V_{k-1}, \quad \textit{клик}\, n_{k-1}\\
&V_1 \cup V_3 \cup \ldots \cup V_k \cup \{a,b\}\\
&V_1 \cup V_3 \cup \ldots \cup V_k \cup \{b,c\}\\
&V_1 \cup V_3 \cup \ldots \cup V_k \cup \{c,d\}
\end{aligned}\right\} k = 2k'+1.
\end{aligned}
$$

Теперь рассмотрим граф $\overline{F}$. Вершины множества V_1 имеют в этом графе степень ноль, поэтому каждая вершина составляет клику. Следовательно, в дальнейшем мы их не рассматриваем. Вершины множества V_2 смежны со всеми вершинами графа $\overline{F}$ кроме вершин множества V_1, и, следовательно, они входят в любую клику графа $\overline{F}$. Проводя для графа $\overline{F}$ аналогичные рассуждения с учетом того, что вершины графа $\overline{L}$ смежны с вершинами множеств с четными номерами, получим следующий упорядоченный набор клик:

$\{v\}, \quad v \in V_1, \quad$ *клик* n_1,

$V_2 \cup \{v\}, \quad v \in V_3, \quad$ *клик* n_3,

$V_2 \cup V_4 \cup \{v\}, \quad v \in V_5, \quad$ *клик* n_5,

…

$$\left.\begin{array}{l} V_2 \cup V_4 \cup \ldots \cup V_{k-2} \cup \{v\}, \quad v \in V_{k-1}, \quad \text{клик } n_{k-1} \\ V_2 \cup V_4 \cup \ldots \cup V_k \cup \{c,a\} \\ V_2 \cup V_4 \cup \ldots \cup V_k \cup \{a,d\} \\ V_2 \cup V_4 \cup \ldots \cup V_k \cup \{d,b\} \end{array}\right\} k = 2k',$$

$$\left.\begin{array}{l} V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup \{v\}, \quad v \in V_k, \quad \text{клик } n_k \\ V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup \{c,a\} \\ V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup \{a,d\} \\ V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup \{d,b\} \end{array}\right\} k = 2k'+1.$$

Рассмотрим второй случай. Приведем табл. 2 степеней для этого случая.

Таблица 2. Степени вершин для пункта 2а процедуры α

	V_1	V_2	V_3	…	V_k			
					$k = 2k'$		$k = 2k'+1$	
					V_k'	V_k''	V_k'	V_k''
	n_1	N_2	n_3	…	1	$n_k - 1$	1	$n_k - 1$
F	4	0	4	…	4	0	0	4
$\overline{F}$	0	4	0	…	0	4	4	0

Этот случай отличается от предыдущего только тем, что в последнем множестве существует особая вершина (для четного k – это вершина степени 4 по графу L, для нечетного – вершина степени 0 по графу L). Разобьем множество V_k на два

непересекающихся подмножества V_k' и V_k'', где V_k' – это множество, состоящее из особой вершины, V_k'' – остальные вершины множества V_k. Так как табл. 1, 2 степеней для первого и второго случаев совпадают за исключением последнего множества, то, проведя рассуждения, аналогичные приведенным выше, видим, что клики, в которых нет вершин из последнего множества, остаются и в этом случае. Рассмотрим клики, содержащие вершины множества V_k. Пусть k четно. Множество вершин множеств с нечетными номерами $V_1, V_3, \ldots, V_{k-1}$ образуют полный подграф, добавляя к этому множеству вершин одну вершину множества V_k'', получим клику. Число таких клик равно числу вершин множества V_k'', то есть $(n_k - 1)$.

Если к множеству вершин $V_1, V_3, \ldots, V_{k-1}$ добавить множество V_k', то для того, чтобы получить клику, необходимо добавить одну из пар вершин *a, b* или *b, c*, или *c, d*. Теперь пусть k нечетно. В этом случае клику образуют вершины множеств с нечетными номерами $V_1, V_3, \ldots, V_{k-2}$, вершины множества V_k'' и либо вершина множества V_k', либо одна из пар вершин *a, b* или *b, c*, или *c, d*. Таким образом, мы построили все клики графа *F*. Переходя к дополнительному графу, можно заметить, что вершины множества V_1, как и в предыдущем случае, рассматривать не надо, так как они имеют в графе $\overline{F}$ нулевую степень. В оставшемся графе (после исключения вершин первого множества) клики строятся аналогично кликам графа *F*. Выпишем упорядоченные наборы клик.

Для графа *F*:

$V_1 \cup \{v\}, \quad v \in V_2, \quad$ *клик* n_2,

$V_1 \cup V_3 \cup \{v\}, \quad v \in V_4, \quad$ *клик* n_4,

…

…

$$\left.\begin{array}{l} V_1 \cup V_3 \cup \ldots \cup V_{k-1} \cup \{v\}, \quad v \in V_k'', \quad клик\ (n_k - 1) \\ V_1 \cup V_3 \cup \ldots \cup V_{k-1} \cup V_k' \cup \{a,b\} \\ V_1 \cup V_3 \cup \ldots \cup V_{k-1} \cup V_k' \cup \{b,c\} \\ V_1 \cup V_3 \cup \ldots \cup V_{k-1} \cup V_k' \cup \{c,d\} \end{array}\right\} k = 2k',$$

$$\left.\begin{array}{l} V_1 \cup V_3 \cup \ldots \cup V_{k-2} \cup V_k'' \cup V_k' \\ V_1 \cup V_3 \cup \ldots \cup V_k \cup V_k'' \cup \{a,b\} \\ V_1 \cup V_3 \cup \ldots \cup V_k \cup V_k'' \cup \{b,c\} \\ V_1 \cup V_3 \cup \ldots \cup V_k \cup V_k'' \cup \{c,d\} \end{array}\right\} k = 2k' + 1.$$

Для графа $\overline{F}$:

$\{v\}, \quad v \in V_1, \quad клик\ n_1,$

$V_2 \cup \{v\}, \quad v \in V_3, \quad клик\ n_3,$

$V_2 \cup V_4 \cup \{v\}, \quad v \in V_5, \quad клик\ n_5,$

…

$$\left.\begin{array}{l} V_2 \cup V_4 \cup \ldots \cup V_{k-2} \cup V_k'' \cup V_k' \\ V_2 \cup V_4 \cup \ldots \cup V_{k-2} \cup V_k'' \cup \{c,a\} \\ V_2 \cup V_4 \cup \ldots \cup V_{k-2} \cup V_k'' \cup \{a,d\} \\ V_2 \cup V_4 \cup \ldots \cup V_{k-2} \cup V_k'' \cup \{d,b\} \end{array}\right\} k = 2k',$$

$$\left.\begin{array}{l} V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup \{v\}, \quad v \in V_k'', \quad клик\ (n_k - 1) \\ V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup V_k' \cup \{c,a\} \\ V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup V_k' \cup \{a,d\} \\ V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup V_k' \cup \{d,b\} \end{array}\right\} k = 2k' + 1.$$

Осталось рассмотреть последний случай. В этом случае существует такой номер i, $1 \le i \le k$, что вершины множеств с номерами от 1 до $(i - 1)$ имеют степень 0 или 4 по графу L, в множестве V_i существует по крайней мере одна вершина сте-

пени 2 по графу L, а все остальные вершины из множеств с номерами, большими i, имеют степень 2 по графу L. По аналогии с предыдущими случаями приведем табл. 3 степеней вершин графов F и $\overline{F}$. Множество V_i разобьем на два непересекающихся подмножества V_i' и V_i'', где V_i'' – множество вершин степени 2 по графу L, V_i' – множество всех остальных вершин множества V_i (если i – четно, то это вершины степени 0 по графу L, иначе – степени 4 по графу L).

Таблица 3. Степени вершин для пункта 2б процедуры α

	V_1	V_2	…	V_i				V_{i+1}	…	V_k
				$i = 2i'$		$i = 2i'+1$				
				V_i'	V_i''	V_i'	V_i''			
	n_1	n_2	…	$n_i - t$	t	$n_i - t$	t	n_{i+1}	…	n_k
F	4	0	…	0	2	4	2	2	…	2
$\overline{F}$	0	4	…	4	2	0	2	2	…	2

Пусть i четно. Применяя рассуждения, аналогичные приведенным выше, получаем все клики, последняя из которых состоит из вершин множеств $V_1, V_3, \ldots, V_{i-3}$ и одной вершины множества V_{i-2}. Теперь рассмотрим вершины, образующие полный подграф и состоящие из вершин множеств $V_1, V_3, \ldots, V_{i-1}$, обозначим это множество вершин через M. Добавим к множеству M вершину множества V_i'. Мы не можем добавить более одной вершины из V_i', чтобы полученный подграф оставался полным. По этой же причине мы не можем добавить ни одной другой вершины. Тогда множество M и одна вершина множества V_i' образуют клику. Теперь, добавив к множеству M одну из пар вершин a,b или c,d, мы также получим клику, так как в графе F больше нет вершин, смежных с вершинами a или d. Если к множеству M добавить вершины b и c, то клика образу-

ется, если к рассматриваемым вершинам добавить одну вершину множества V_i''. Так как вершины b и c смежны с вершинами множеств, номера которых нечетны и меньше i, с вершинами множества V_i'' и со всеми вершинами множеств с номерами, строго большими i, следовательно, оставшиеся клики образуются следующим образом: вершины нечетных множеств $V_1, V_3, \ldots, V_{i-r}$, где $i+1 \le r \le k$, $r = 2r'+1$, и одна вершина множества V_{r+1}. Выпишем упорядоченные наборы клик графа F:

$V_1 \cup \{v\}$, $v \in V_2$, *клик* n_2,

$V_1 \cup V_3 \cup \{v\}$, $v \in V_4$, *клик* n_4,

…

$V_1 \cup V_3 \cup \ldots \cup V_{i-3} \cup \{v\}$, $v \in V_{i-2}$, *клик* n_{i-2},

$V_1 \cup V_3 \cup \ldots \cup V_{i-3} \cup V_{i-1} \cup \{v\}, v_i \in V_i'$, *клик* $(n_i - t)$,

$V_1 \cup V_3 \cup \ldots \cup V_{i-3} \cup V_{i-1} \cup \{a, b\}$,

$V_1 \cup V_3 \cup \ldots \cup V_{i-3} \cup V_{i-1} \cup \{b, c\} \cup \{v\}, v \in V_i''$, *клик* t,

$V_1 \cup V_3 \cup \ldots \cup V_{i-1} \cup V_{i+1} \cup \{b, c\} \cup \{v\}, v \in V_{i+2}$, *клик* n_{i+2},

…

$V_1 \cup V_3 \cup \ldots \cup V_{k-1} \cup \{b, c\} \cup \{v\}, v \in V_k$, *клик* n_k $\}k = 2k'$,

$V_1 \cup V_3 \cup \ldots \cup V_{k-2} \cup \{b, c\} \cup V_k$ $\}k = 2k' + 1$,

$V_1 \cup V_3 \cup \ldots \cup V_{i-3} \cup V_{i-1} \cup \{c, d\}$.

Рассмотрим теперь граф $\overline{F}$. Каждая вершина множества V_1 образует клику, так как они имеют в графе степень ноль, а вершины множества V_2 входят во все остальные клики. Как и раньше, определяются клики вида: вершины множеств с четными номерами $V_2, V_4, \ldots, V_r$, где $2 \le r \le i\text{-}2$, $r = 2r'$ и одна вершина из множества V_{r+1}. Рассмотрим множество вершин, состоящее из всех вершин множеств $V_2, V_4, \ldots, V_{i-2}, V_i'$, обозначим его через M. Вершины этого множества смежны с верши-

нами графа $\overline{L}$, причем, только вершины множества M смежны с вершинами c и b. Следовательно, вершины множества M и одна из пар вершин $\{c,a\}$ или $\{d,b\}$ образуют клику. Пара вершин $\{a,d\}$ смежна: со всеми вершинами множеств, номера которых четны и меньше i; с вершинами V_i; с вершинами множеств, номера которых больше i. Следовательно, клики образуются так: вершины четных множеств $V_2,V_4,\ldots,V_r$, где $i \le r \le k$, $r = 2r'$, одна вершина множества V_{r+1} и пара вершин $\{a,d\}$. Выпишем упорядоченные наборы клик графа $\overline{F}$:

$\{v\}, \quad v \in V_1, \quad$ *клик* n_1,

$V_2 \cup \{v\}, \quad v \in V_3, \quad$ *клик* n_3,

$V_2 \cup V_4 \cup \{v\}, \quad v \in V_5, \quad$ *клик* n_5,

$\ldots$

$V_2 \cup V_4 \cup \ldots \cup V_{i-2} \cup \{v\}, \quad v \in V_{i-1}, \quad$ *клик* n_{i-1},

$V_2 \cup V_4 \cup \ldots \cup V_{i-2} \cup V_i' \cup \{c,a\}$,

$V_2 \cup V_4 \cup \ldots \cup V_{i-2} \cup V_i \cup \{a,d\} \cup \{v\}, v \in V_{i+1}, \quad$ *клик* n_{i+1},

$\ldots$

$V_2 \cup V_4 \cup \ldots \cup V_k \cup \{a,d\}$ $\}k = 2k'$,

$V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup \{a,d\} \cup \{v\}, v \in V_k,$ *клик* n_k $\}k = 2k' + 1$,

$V_2 \cup V_4 \cup \ldots \cup V_{i-2} \cup V_i' \cup \{d,b\}$.

Пусть теперь i нечетно. Клики графа F в этом случае строятся аналогично кликам графа $\overline{F}$ для четного i, а клики дополнительного графа $\overline{F}$ – аналогично кликам графа F для четного i. Поэтому приведем только результат. Для графа F:

$V_1 \cup \{v\}, \quad v \in V_2, \quad$ *клик* n_2,
$V_1 \cup V_3 \cup \{v\}, \quad v \in V_4, \quad$ *клик* n_4,
…
$V_1 \cup V_3 \cup \ldots \cup V_{i-2} \cup \{v\}, \quad v \in V_{i-1}, \quad$ *клик* n_{i-1},
$V_1 \cup V_3 \cup \ldots \cup V_{i-2} \cup V_i' \cup \{a,b\}$,
$V_1 \cup V_3 \cup \ldots \cup V_{i-2} \cup V_i \cup \{b,c\} \cup \{v\}, v \in V_{i+1}, \quad$ *клик* n_{i+1},
…
$V_1 \cup V_3 \cup \ldots \cup V_{k-1} \cup \{b,c\} \cup \{v\}, v \in V_k,$ *клик* n_k $\}k = 2k'$,
$V_1 \cup V_3 \cup \ldots \cup V_{k-2} \cup \{b,c\} \cup V_k$ $\}k = 2k'+1$,
$V_1 \cup V_3 \cup \ldots \cup V_{i-2} \cup V_i' \cup \{c,d\}$.

Для графа $\overline{F}$:

$\{v\}, \quad v \in V_1, \quad$ *клик* n_1,
$V_2 \cup \{v\}, \quad v \in V_3, \quad$ *клик* n_3,
$V_2 \cup V_4 \cup \{v\}, \quad v \in V_5, \quad$ *клик* n_5,
…
$V_2 \cup V_4 \cup \ldots \cup V_{i-3} \cup \{v\}, \quad v \in V_{i-2}, \quad$ *клик* n_{i-2},
$V_2 \cup V_4 \cup \ldots \cup V_{i-3} \cup V_{i-1} \cup \{c,a\}$,
$V_2 \cup V_4 \cup \ldots \cup V_{i-3} \cup V_{i-1} \cup \{a,d\} \cup \{v\}, v \in V_i'', \quad$ *клик* t,
$V_2 \cup V_4 \cup \ldots \cup V_{i-3} \cup V_{i-1} \cup \{a,d\} \cup \{v\}, v \in V_{i+2}, \quad$ *клик* n_{i+2},
…
$V_2 \cup V_4 \cup \ldots \cup V_k \cup \{a,d\}$ $\}k = 2k'$,
$V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup \{a,d\} \cup \{v\}, v \in V_k,$ *клик* n_k $\}k = 2k'+1$,
$V_2 \cup V_4 \cup \ldots \cup V_{i-3} \cup V_{i-1} \cup \{d,b\}$.

Таким образом, мы рассмотрели все случаи. Так как в каждом из них были построены все клики, для которых мы указали способ упорядочивания, то по теореме 1 графы F и $\overline{F}$ являются интервальными.

Теорема 3. Число доопределений ребер и неребер на вершинах графов L и G равно $n+2$.

Доказательство. Обозначим через $D(n_1,n_2,\ldots,n_k)$ число доопределений ребер и неребер на вершинах графа L и вершинах множеств $V_1,V_2,\ldots,V_k$. Рассмотрим множество V_1. Согласно процедуре α возможны два случая:

1. Все вершины в множестве V_1 имеют степень $(n\text{-}1)+4$ в графе F, и тогда вершины этого множества на дальнейшее доопределение не влияют, т.е. $D(n_1,n_2,\ldots,n_k) = D(n_2,n_3,\ldots,n_k)$.
2. В множестве V_1 существует $t, 1 \le t \le n_1$, вершин степени 2 по графу L. Тогда однозначно доопределяются ребра и неребра на вершинах множеств $V_2,V_3,\ldots,V_k$ и вершинах графа L. Следовательно, $D(n_1,n_2,\ldots,n_k)$ равно числу способов выбрать t, т.е. $D(n_1,n_2,\ldots,n_k) = n_1$.

Так как эти случаи независимы, то

$$D(n_1,n_2,\ldots,n_k) = n_1+D(n_2,n_3,\ldots,n_k).$$

Рассуждая аналогичным образом для всех множеств V_i, где $2 \le i \le k-1$, получаем, что $D(n_i,n_{i+1},\ldots,n_k) = n_i+D(n_{i+1},n_{i+2},\ldots,n_k)$.

Теперь пусть $i = k$, пусть для определенности k четно (случай нечетного k рассматривается аналогично). Тогда возможны два случая:

1. В множестве V_k существует единственная вершина степени 4 по графу L. Тогда все остальные вершины в этом множестве имеют степень 0 по графу L. Получаем, что $D(n_k) = 1$.
2. В множестве V_k существует $t, 0 \le t \le n_k$, вершин степени 2 по графу L. Следовательно, $D(n_k)$ равно числу способов выбрать t, т.е. $D(n_k) = n_k+1$.

Таким образом, $D(n_k) = n_k+2$ и общее число доопределений

$$D(n_1,\ldots,n_k) = n_1 + D(n_1,\ldots,n_k) = n_1 + n_2 + D(n_3,\ldots,n_k) = \ldots =$$
$$= n_1 + n_2 + \ldots + n_{k-1} + D(n_k) = n_1 + n_2 + \ldots + n_k + 2 = n + 2.$$

Литература

1. *Boland J.Ch., Lekkerkerker C.G.* Representation of a finite graph by a set of intervals on the real line // Fund. Math., 51(1962). P. 45-64.

2. *Козырев В.П.* Составление многопроцессорных расписаний для 2-совместимых и 2-несовместимых работ // Проблемы теоретической кибернетики. Тез. докл. XIII междунар. конф. (Казань, 31 мая 2002 г.), ч. I, М.: изд-во мехмата МГУ. С. 89.

3. *Козырев В.П.* Характеризация графов, не имеющих чередующихся циклов // Тез. докл. научн. конф. "Математические модели сложных систем и междисциплинарные исследования" (23-24 октября 2002 г.). М.: ВЦ РАН, 2002. С. 33.

4. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.

5. *Зыков А.А.* Теория конечных графов. Новосибирск: Наука, 1969. Т. I.

6. *Козырев В.П.* Описание и порождение всех минимальных раскрасок интервального графа и решение смежных вопросов// Журнал вычислительной математики и математической физики, 1996. Т. 36, № 5. С. 146-153.

ГРАФЫ С ЕДИНСТВЕННЫМ ЦИКЛОМ БЕЗ ТРИАНГУЛЯТОРОВ

В.П. Козырев, Е.А. Соколова

В работе описывается строение графов с единственным циклом без триангуляторов (цикл длины четыре). Приведено доказательство интервальности дополнительного графа.

1. Основные определения и утверждение

Создание систем жесткого реального времени и автоматизация их создания требуют исследования структур графов определенного вида. В таких системах осуществляется назначение директивных сроков начала и окончания обработки конкретных вычислительных задач. Близкими задачами являются упаковка в контейнеры, динамическое распределение памяти, задача о разбиении, задача о рюкзаке и другие.

Рассматриваются неизоморфные графы *G(V,E)* без петель и кратных ребер (т.е. простые графы), имеющие *n* вершин, $n = |V|$, для удобства изложения материала будем вводить нумерацию вершин. Ребра дополнительного графа $\overline{G}$ будем называть неребрами графа *G* и наоборот, ребра графа *G* – это неребра графа $\overline{G}$. Будем использовать следующие обозначения: *(u,v)* – ребро, соединяющее вершины *u* и *v*, $\overline{(u, v)}$ – неребро, соединяющее вершины *u* и *v*. Граф называется интервальным, если его вершинам можно взаимно однозначно поставить в соответствие отрезки на вещественной прямой таким образом, что две вершины смежны тогда и только тогда, когда отрезки, соответствующие этим вершинам, пересекаются. Если граф и его дополнительный граф являются интервальными, то граф будем называть 2-интервальным. Клика *K* графа *G* – это полный подграф графа, замкнутый по включению.

Теорема 1 [1]. Система клик $K_1,\ldots,K_n$ множества вершин графа G является системой клик интервального графа G тогда и только тогда, когда существует такая нумерация клик, что если $K_i \cap K_j \neq \emptyset$, то это пересечение принадлежит всем кликам с номерами от i до j.

Определение 1. Чередующейся цепью (кратко П-цепью) графа G называется последовательность ребер и неребер, в которой за каждым ребром следует неребро и за каждым неребром следует ребро; имеются два крайних элемента. Замкнутая чередующаяся цепь называется чередующимся циклом (кратко П-циклом).

Графы, не содержащие П-циклов, охарактеризованы в работе [3]. Показано, что вершины связного графа G можно разбить на множества $V_1,V_2,\ldots,V_k$ следующим образом: вершины множества V_1 имеют в графе G максимальную степень $(n\text{-}1)$. Вершины множества V_2 имеют степень ноль в графе $G\backslash V_1$, полученном из графа G удалением вершин множества V_1 и инцидентных им ребер. Вершины множества V_3 имеют в графе $G\backslash\{V_1,V_2\}$ максимальную степень и смежные со всеми оставшимися вершинами, и так далее. Таким образом, вершины множеств с нечетными номерами смежны между собой и со всеми вершинами из множеств с большими номерами; вершины множеств с четными номерами смежны только с вершинами из множеств с меньшими нечетными номерами. Триангулятором цикла в графе называется такое ребро графа, которое образует треугольник вместе с некоторой парой последовательных ребер цикла.

Пусть G – связный граф без П-циклов, множество вершин которого разбито на множества $V_1,V_2,\ldots,V_k$, $|V_i| = n_i$, $1 \le i \le k$, C – граф на множестве вершин *a, b, c, d*, представляющий собой цикл без триангуляторов длины четыре*:* $(a,b),(b,c),(c,d),(d,a),\overline{(a,c)},\overline{(b,d)}$. Причем $V(G) \cap V(C) = \emptyset$. Введем определение. Граф F – граф, получающийся в результате доопределе-

ния ребер и неребер на вершинах графов *G* и *C* таким образом, что граф *F* содержит П-цикл только на вершинах графа *C*. Следовательно, граф F содержит единственный цикл без триангуляторов, так как цикл без триангуляторов содержит П-цикл. Рассмотрим граф *C* и введем понятие эквивалентных вершин. Назовем вершину *u* эквивалентной вершине *v*, если для любой вершины *i* из существования ребра (*i,u*), следует существование ребра (*i,v*), а из существования неребра $\overline{(i,u)}$ – неребро $\overline{(i,v)}$. Две вершины *u* и *v* эквивалентны, если *u* эквивалентна *v*, а также *v* эквивалентна *u*. Степенью вершины *v* в графе *G* по графу *C* называется число вершин графа *C*, смежных с вершиной *v*.

2. Вспомогательные утверждения

Лемма 1. В графе *F* вершины *a, b, c, d* эквивалентны.

Доказательство. Докажем, что вершина *a* эквивалентна вершине *b*. Аналогично доказывается эквивалентность остальных вершин. Фиксируем произвольную вершину *u* графа *C* и пусть (*u,a*) – ребро. Тогда если предположить, что вершины *u* и *b* несмежны, т.е. $\overline{(u,b)}$ – неребро, то образуется П-цикл: $(u,a),\overline{(a,c)},(c,b),\overline{(b,u)}$, что противоречит определению графа *F* (граф содержит П-цикл только на вершинах подграфа *C*). Следовательно, (*u,b*) – ребро. Теперь фиксируем произвольную вершину *v*, которая соединена с вершиной *a* неребром. Предположим, что вершины *v* и *b* смежны, тогда образуется П-цикл: $\overline{(b,a)},(a,d),\overline{(d,b)},(b,v)$, получили противоречие с определением графа *F*. Следовательно, $\overline{(v,b)}$ – неребро, и согласно введенному выше определению вершина *a* эквивалентна вершине *b*.

Так как все вершины графа *C* эквивалентны, то далее все доказательства будем проводить только для вершины *a*.

Лемма 2. Любая вершина графа G имеет степень по графу C либо 0, либо 4.

Доказательство следует непосредственно из леммы 1.

Лемма 3. Вершины графа C смежны с вершинами множеств с нечетными номерами графа G за исключением, быть может, вершин последнего множества, и не смежны с вершинами множеств с четными номерами за исключением, быть может, вершин последнего множества.

Доказательство. Рассмотрим произвольное множество V_{2p+1}, $1 \le 2p+1 < k$, в этом множестве выберем произвольную вершину u_1. Так как выбранное множество не является последним, то существуют вершины u_2 и u_3 из множеств с большими номерами, соединенные с вершиной u_1 ребрами (u_1,u_2), (u_1,u_3), а между собой – неребром $\overline{(u_2,u_3)}$. Выбрать вершины можно следующим образом: если $2p+2 = k$, то $u_2, u_3 \in V_{2p+2}$ (последнее множество содержит не менее двух вершин [3]); если $2p+2 < k$, то $u_2 \in V_{2p+2}, u_3 \in V_{2p+3}$. Предположим, что вершины a и u_1 несмежны. Тогда получаем П-цикл L_6: $\overline{(u_3,u_2)},(u_2,u_1),\overline{(u_1,a)},(a,b),\overline{(b,d)},(d,a),\overline{(a,u_1)},(u_1,u_3)$, чего быть не может. В силу произвольности выбора множества V_{2p+1} и вершины u_1 получаем, что любая вершина из множества с нечетным номером, если только это множество не является последним, соединена с вершиной a ребром. Теперь рассмотрим произвольное множество V_{2s}, $2 \le 2s < k$, в этом множестве выберем произвольную вершину v_1. Так как выбранное множество не является последним, то существуют вершины v_2 и v_3 из множеств с большими номерами, соединенные с вершиной v_1 неребрами $\overline{(v_1,v_2)}, \overline{(v_1,v_3)}$, а между собой – ребром (v_2,v_3). Выбрать вершины можно следующим образом: если $2s+1 = k$, то $v_2, v_3 \in V_{2s+1}$; если $2s+1 < k$, то $v_2 \in V_{2s+1}, v_3 \in V_{2s+2}$. Допустим, что вершины a и v_1 смежны. Тогда вершина v_1 имеет степень 4 по графу C (лемма 2), значит, (v_1,c) – ребро и образуется

П-цикл L_5: $(v_3,v_2),\overline{(v_2,v_1)},(v_1,a),\overline{(a,c)},(c,v_1),\overline{(v_1,v_3)}$, опять приходим к противоречию. Следовательно, $\overline{(v_1,c)}$ – нереброо. В силу произвольности выбора множества V_{2s} и вершины v_1 получаем, что любая вершина из множества с четным номером, если только это множество не является последним, соединена с вершиной a неребром.

Лемма 4. Если k четно, то в множестве V_k может существовать не более одной вершины степени 4 по графу C. Если k нечетно, то в множестве V_k может существовать не более одной вершины степени 0 по графу C.

Доказательство. Рассмотрим случай, когда последнее множество имеет четный номер. Предположим, что в множестве V_k существует две вершины v_1, v_2 степени 4 по графу C. Тогда возникает П-цикл: $(v_1,a),\overline{(a,c)},(c,v_2),\overline{(v_2,v_1)}$, чего быть не может. Следовательно, наше предположение неверно, и в последнем множестве может существовать не более одной вершины степени 4 по графу C. Случай, когда k – нечетно, доказывается аналогично.

Приведем процедуру β, описывающую структуру графа F. Граф F строится одним из двух способов:

1) все вершины множеств с нечетными номерами имеют степень 4 по графу C, а все вершины множеств с четными номерами имеют степень 0 по графу C;
2) все вершины множеств с нечетными номерами, за исключением последнего множества, имеют степень 4 по графу C, все вершины множеств с четными номерами, за исключением последнего множества, имеют степень 0 по графу C. Если k – нечетно, то существует ровно одна вершина, степень которой по графу C равна нулю, а остальные вершины в этом множестве имеют степень по графу C, равную 4. Если k четно, то существует ровно одна вершина, степень которой по графу C равна 4, а ос-

тальные вершины в этом множестве имеют степень по графу C, равную нулю.

Введем обозначения. Пусть F_1 – граф, полученный процедурой β в п. 1, F_2 – граф, полученный процедурой β в п. 2.

3. Основные утверждения

Теорема 2. Графы, описанные процедурой β, содержат П-цикл только на вершинах графа C.

Доказательство. И в графе F_1, и в графе F_2 вершины множества V_1 смежны со всеми вершинами графов, следовательно, они не могут принадлежать П-циклу. Удалим это множество вершин со всеми инцидентными ребрами. В новых графах $F_1 \setminus \{V_1\}$ и $F_2 \setminus \{V_1\}$ вершины множества V_2 имеют степень 0, поэтому они не могут принадлежать П-циклу. Эти вершины также исключаем. Применяя дальше эти рассуждения, можем исключить в каждом графе все множества до V_{k-1} включительно. Полученные в результате графы обозначим F_1' и F_2'. В графе F_1' вершины множества V_k либо смежны со всеми вершинами графа, либо имеют степень 0 (в зависимости от четности числа k), в любом из этих случаев вершины множества V_k не принадлежат П-циклу. Исключая эти вершины и инцидентные им ребра, получаем граф C. Таким образом, П-цикл в графе F_1 не включает вершины графа G и теорема для этого случая доказана. Рассмотрим теперь граф F_2'. В множестве V_k существует единственная вершина v, степень которой по графу C равна нулю, если k нечетно, и 4, если k четно. Остальные вершины множества V_k либо смежны со всеми вершинами графа F_2', либо имеют в этом графе степень 0 (в зависимости от четности числа k). Следовательно, ни одна вершина этого множества не может принадлежать

П-циклу, удалим эти вершины и инцидентные им ребра. В оставшемся графе вершина v имеет степень либо 0, либо 4 (то есть смежна со всеми вершинами), тогда она также не может принадлежать П-циклу. Таким образом, П-цикл в графе F_2' не включает вершины графа G.

Теорема 3. Процедура β описывает все графы F без пропусков и повторений.

Доказательство. По лемме 3 для вершин, не принадлежащих последнему множеству V_k, верно, что любая вершина множества с нечетным номером имеет степень 4 по графу C, а вершины множеств с четными номерами имеют степень 0 по графу C. Рассмотрим вершины последнего множества. Либо в этом множестве существует вершина степени 4 по графу C для четного k, и степени 0 по графу C для нечетного k, тогда по лемме 4 такая вершина единственная и полученный в результате граф описывается п. 2 процедуры β. Либо такой вершины не существует, и тогда степень всех вершин множества V_k либо 0, либо 4 по графу C, т.е. этот граф описывается п. 1 процедуры β. Так как разобраны все возможные случай, то процедура β строит все графы F. Докажем теперь, что графы F_1 и F_2 не являются изоморфными. Приведем табл. 1 степеней:

Таблица 1. Степени вершин в графах F_1 и F_2

	в графе F_1	в графе F_2
V_1	$(n-1)+4$	$(n-1)+4$
V_2	n_1	n_1
V_3	$(n-1)-n_2+4$	$(n-1)-n_2+4$
V_4	n_1+n_3	n_1+n_3
…	…	

Продолжение таблицы 1

$k = 2k'$	$v \in V_k$	$n_1 + n_3 + \ldots + n_{k-1}$	$n_1 + n_3 + \ldots + n_{k-1} + 4$
	$V_k \setminus \{v\}$		$n_1 + n_3 + \ldots + n_{k-1}$
	$\{a,b,c,d\}$	$n_1 + n_3 + \ldots + n_{k-1} + 2$	$n_1 + n_3 + \ldots + n_{k-1} + 3$
$k = 2k'+1$	$v \in V_k$	$(n-1) - n_2 - \ldots - n_{k-1} + 4$	$(n-1) - n_2 - \ldots - n_{k-1}$
	$V_k \setminus \{v\}$		$(n-1) - n_2 - \ldots - n_{k-1} + 4$
	$\{a,b,c,d\}$	$n_1 + n_3 + \ldots + n_k + 2$	$n_1 + n_3 + \ldots + n_k + 1$

Как видно из табл. 1, степени вершин множеств с нечетными номерами строго уменьшаются, а степени вершин из множеств с четными номерами строго возрастают. Причем максимальная степень вершин множеств с четными номерами (для четного k: $n_1 + n_3 + \cdots + n_{k-1} + 4$) меньше минимальной степени вершин из множеств с нечетными номерами (для четного k: $(n-1) - n_2 - \cdots - n_{k-2} + 4 = n_1 + n_3 + \cdots + \ + n_{k-1} + n_k + 3$), так как $n_k \geq 2$. Следовательно, степени вершин из различных множеств различны, при этом в графе F_2 в последнем множестве существует вершина (обозначим ее v), степень которой отлична от степеней других вершин. А в графе F_1 не существует вершины, степень которой равна степени вершины v. Следовательно, графы F_1 и F_2 неизоморфны.

Исследуем графы F_1 и F_2, и их дополнительные графы на интервальность. Графы F_1 и F_2 не являются интервальными, так как они содержат цикл без триангуляторов. Рассмотрим дополнительные графы $\overline{F_1}$ и $\overline{F_2}$.

Теорема 4. Графы $\overline{F_1}$ и $\overline{F_2}$ являются интервальными.

Доказательство. Построим клики графов $\overline{F_1}$ и $\overline{F_2}$, упорядочив их так, чтобы были выполнены условия теоремы 1. Проведем рассуждение только для графа $\overline{F_1}$, для графа $\overline{F_2}$ доказательство аналогично. Вершины множества V_1 имеют в гра-

фе $\overline{F_1}$ степень 0, поэтому каждая вершина составляет клику. Следовательно, в дальнейшем мы их не рассматриваем. Вершины множества V_2 смежны со всеми вершинами графа $\overline{F_1}$ кроме вершин множества V_1, и, следовательно, они входят во все клики графа за исключением клик, образованных вершинами множества V_1. Исходя из этого, будем строить клики, начиная с множества V_2. Первую группу клик составляют все вершины множества V_2 и одна вершина из множества V_3, число таких клик равно числу вершин в множестве V_3, т.е. равно n_3. Других вершин в эти клики добавить нельзя, так как вершины множества V_3 смежны только с вершинами множества V_2. Добавляя к вершинам множества V_2 вершины множества V_4, получим полный подграф, а добавляя одну вершину из множества V_5, получим клику, других вершин добавить нельзя, так как они не будут смежны с вершиной множества V_5. Построенные клики образуют вторую группу клик, число которых равно n_5. Повторяем эти рассуждения для всех множеств до V_{k-1} включительно. В зависимости от того, четно или нечетно число k, рассмотрим, что будет происходить в последнем множестве. Пусть k нечетно. Тогда получаем группу клик, состоящую из вершин множеств с четными номерами $V_2, V_4, \ldots, V_{k-1}$ и одной вершины из множества V_k, их число равно n_k. Вершины графа C смежны со всеми вершинами множеств с четными номерами, при этом в самом графе C попарно смежны вершины: a и c, b и d. Следовательно, получим две клики, состоящие из всех вершин множеств с четными номерами и одной из пар вершин: $\{a,c\}$, $\{b,d\}$. Выпишем все клики графа и упорядочим их в соответствии с условием теоремы 1. Клики упорядочим по группам, клики внутри каждой группы могут быть упорядочены произвольным образом, так как каждая клика состоит из некоторой общей для данной группы час-

ти вершин и одной вершины, которая не содержится ни в какой другой клике графа. Для графа $\overline{F_1}$:

$\{v\}, \quad v \in V_1, \quad$ *клик* n_1,

$V_2 \cup \{v\}, \quad v \in V_3, \quad$ *клик* n_3,

$V_2 \cup V_4 \cup \{v\}, \quad v \in V_5, \quad$ *клик* n_5,

…

$$\left.\begin{array}{l} V_2 \cup V_4 \cup \ldots \cup V_{k-2} \cup \{v\}, \quad v \in V_{k-1}, \quad \text{клик } n_{k-1} \\ V_2 \cup V_4 \cup \ldots \cup V_{k-2} \cup V_k \cup \{a, c\} \\ V_2 \cup V_4 \cup \ldots \cup V_{k-2} \cup V_k \cup \{b, d\} \end{array}\right\} k = 2k',$$

$$\left.\begin{array}{l} V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup \{v\}, \quad v \in V_k, \quad \text{клик } n_k \\ V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup \{a, c\} \\ V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup \{b, d\} \end{array}\right\} k = 2k' + 1.$$

Перед тем как выписать упорядоченный набор клик для графа $\overline{F_2}$ введем обозначение. Пусть $V_k = \{v\} \cup V_k'$, где v – особая вершина (степени 4 но графу C, если k нечетно, и степени 0 по графу C, если k четно), V_k' – все остальные вершины множества V_k.

$\{v\}, \quad v \in V_1, \quad$ *клик* n_1,

$V_2 \cup \{v\}, \quad v \in V_3, \quad$ *клик* n_3,

$V_2 \cup V_4 \cup \{v\}, \quad v \in V_5, \quad$ *клик* n_5,

…

…

$$\left.\begin{array}{l} V_2 \cup V_4 \cup \ldots \cup V_{k-2} \cup \{v\}, \quad v \in V_{k-1}, \quad \textit{клик } n_{k-1} \\ V_2 \cup V_4 \cup \ldots \cup V_{k-2} \cup V_k' \cup \{w\} \\ V_2 \cup V_4 \cup \ldots \cup V_{k-2} \cup V_k' \cup \{a, c\} \\ V_2 \cup V_4 \cup \ldots \cup V_{k-2} \cup V_k' \cup \{b, d\} \end{array}\right\} k = 2k',$$

$$\left.\begin{array}{l} V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup \{v\}, \quad v \in V_k, \quad \textit{клик } n_k \\ V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup \{w, a, c\} \\ V_2 \cup V_4 \cup \ldots \cup V_{k-1} \cup \{w, b, d\} \end{array}\right\} k = 2k' + 1.$$

Таким образом, графы $\overline{F_1}$ и $\overline{F_2}$ являются интервальными.

Замечание. Если вместо графа C рассмотреть цикл без триангуляторов большей длины (граф C'), то граф $\overline{F}$ уже не будет интервальным при любом доопределении ребер и неребер на вершинах графов G и C'. Так как если мы возьмем цикл длины пять C_5, то дополнительный граф $\overline{F}$ будет также содержать цикл без триангуляторов длины пять $\overline{C_5}$, и следовательно, $\overline{F}$ не является интервальным. Если взяли цикл C_k, $k \geq 6$, то дополнительный граф содержит цикл без триангуляторов длины четыре и также не является интервальным.

Литература

1. *Boland J.Ch., Lekkerkerker C.G.* Representation of a finite graph by a set of intervals on the real line // Fund. Math., 51(1962). P. 45-64.

2. *Козырев В.П.* Составление многопроцессорных расписаний для 2-совместимых и 2-несовместимых работ // Проблемы теоретической кибернетики. Тез. докл. XIII междунар. конф. (Казань, 31 мая 2002 г.), ч. I. М.: изд-во мехмата МГУ. С. 89.

3. *Козырев В.П.* Характеризация графов, не имеющих чередующихся циклов // Тез. докл. научн. конф. "Математиче-

ские модели сложных систем и междисциплинарные исследования" (23-24 октября 2002 г.). М.: ВЦ РАН. С. 33.

4. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.

5. *Зыков А.А.* Теория конечных графов. Новосибирск: Наука, 1969. Т. I.

6. *Козырев В.П.* Описание и порождение всех минимальных раскрасок интервального графа и решение смежных вопросов // Журнал вычислительной математики и математической физики, 1996. Т. 36, № 5. С. 146-153.

АЛГОРИТМЫ ПЛАНИРОВАНИЯ ВЫЧИСЛЕНИЙ В МНОГОПРОЦЕССОРНЫХ СИСТЕМАХ С НЕОДНОРОДНЫМ МНОЖЕСТВОМ РАБОТ И ДИРЕКТИВНЫМИ ИНТЕРВАЛАМИ

Д.Р. Гончар, М.Г. Фуругян

Рассматривается задача составления многопроцессорного расписания с директивными интервалами для случая, когда часть работ допускает прерывания и переключения с одного процессора на другой, а часть не допускает. Предлагается два приближенных алгоритма, минимизирующих максимальное запаздывание. Приводятся результаты численных экспериментов.

1. Постановка задачи

Рассматривается вычислительная система, состоящая из m идентичных процессоров. Имеется множество работ $N = N_1 \cup N_2$ ($N_1 \cap N_2 = \varnothing$), где N_1 – непрерываемые работы, N_2 – работы, допускающие прерывания и переключения с одного процессора на другой. Предполагается, что прерывания и переключения не требуют временных затрат. В фиксированный момент времени каждая работа может выполняться не более чем одним процессором, а каждый процессор может выполнять не более одной работы. Работы N_1 могут выполняться только процессорами $j = 1,\ldots,m_1$, $m_1 < m$ (процессорами первой группы), а работы N_2 – как процессорами первой, так и процессорами второй группы ($j = m_1 + 1,\ldots,m$).

Заданы длительности t_i выполнения работ $i \in N$. Для работ $i \in N_1$ установлен единый директивный интервал [0; T], который не может быть нарушен. Для каждой работы $i \in N_2$ установлен директивный интервал $[b_i, f_i]$ ($f_i - b_i \geq t_i$). Выполне-

ние работы $i \in N_2$ может быть начато не ранее момента b_i, а если она завершится в момент $\bar{f}_i$, то штраф за несвоевременное выполнение работ N_2 составит величину $F = \max_{i \in N_2} \max(\bar{f}_i - f_i, 0)$. Требуется найти такое расписание выполнения работ *N*, при котором все работы N_1 выполняются в интервале [0; *T*] и при этом штраф F минимален.

Подробный обзор литературы для случая, когда все работы являются непрерываемые, а также для случая, когда все работы допускают прерывания и переключения, содержится в [1]. Задачи со смешанным типом работ мало освещены в литературе. Так, например, в [2, 3] предполагается, что каждая работа строго закреплена за конкретным процессором, на множестве работ задан частичный порядок выполнения и, кроме того, только один из приборов допускает прерывания. В [4] рассмотрены случаи, когда директивные интервалы одинаковые, а также, когда директивные интервалы могут различаться, но с рядом дополнительных ограничений. В [1] рассмотрена задача на быстродействие для случая смешанного множества работ и идентичных процессоров.

2. Планирование выполнения непрерываемых работ

Для построения расписания выполнения работ N_1 на m_1 процессорах первой группы используется приближенный оптимизационный мультиоценочный алгоритм с калибровкой [5], который, как показали численные эксперименты, имеет высокую точность и является достаточно быстрым. Если длина полученного расписания не превышает *T*, строится расписание прерываемых работ (разд. 3). Далее будем предполагать, что построено расписание выполнения работ N_1 и его длина не превосходит *T*. Введем следующие обозначения:

$M_1 = \{j : j = 1, \ldots, m_1\}$; $M_2 = \{j : j = m_1 + 1, \ldots, m\}$; $M = M_1 \cup M_2$.

Для $j \in M_1$ определим величины: Q_j – длина интервала загруженности процессора j; $L_j = T - Q_j (L_j \geq 0)$; $N_1(j)$ – номера работ из N_1, назначенных на процессор j; $a_{ij} (i \in N_1(j))$ – момент начала выполнения работы i процессором j.

3. Планирование выполнения прерываемых работ

Для выполнения работ N_2 используются процессоры второй группы и частично процессоры первой группы. Каждой работе $i \in N_2$ приписывается фиксированная величина p_i, характеризующая ее срочность. Работы с меньшей величиной p_i являются более срочными. Рассмотрим два варианта вычисления величин p_i: в первом варианте $p_i = f_i$; во втором $p_i = f_i - b_i - t_i$.

Пусть $d_1 < d_2 < \ldots < d_s$ – все различные величины b_i, $i \in N_2$. Для интервала $[d_k; d_{k+1}]$ $(k = 1, \ldots, s-1)$ введем следующие обозначения:

$N_2(d_k)$ – список номеров работ $i \in N_2$, готовых к выполнению в интервале $[d_k; d_{k+1}]$ (т.е. $b_i \leq d_k$); список $N_2(d_k)$ будем хранить упорядоченным по неубыванию величин p_i (т.е. первая работа в списке самая срочная);

τ_j – момент времени, начиная с которого на процессор j можно назначить очередную работу из $N_2(d_k)$;

$v_j (j \in M)$ – максимальное время, которое может быть выделено процессором j на выполнение работ в интервале $[d_k; d_{k+1}]$;

$M_{11}(d_k)$ – номера процессоров первой группы, на которые можно дополнительно назначить работы из $N_2(d_k)$ в интервале $[d_k; d_{k+1}]$, корректируя при этом построенное ранее расписание выполнения работ N_1;

$M_{12}(d_k)$ – номера процессоров первой группы, на которые можно дополнительно назначить работы из $N_2(d_k)$ в интервале $[d_k;d_{k+1}]$, не корректируя при этом построенное ранее расписание выполнения работ N_1.

При описании алгоритма составления расписания выполнения работ N_2 будут использованы следующие процедуры.

1. Процедура $\Pi_1(i_0, j_0)$ назначает работу $i_0 \in N_2(d_k)$ на процессор $j_0 \in M_2$ в интервале $[d_k;d_{k+1}]$.

Положить $\tau = \min(t_{i_0}; v_{j_0})$. Работу i_0 назначить на процессор j_0 в интервале $[\tau_{j_0};\tau_{j_0}+\tau]$. Положить $\tau_{j_0} = \tau_{j_0} + \tau$; $t_{i_0} = t_{i_0} - \tau$; $v_{i_0} = v_{i_0} - \tau$. Работу i_0 исключить из $N_2(d_k)$. Если $t_{i_0} \neq 0$ и $k < S$, то работу i_0 включить в $N_2(d_{k+1})$.

2. Процедура $M_{11}(d_k)$ строит множество $M_{11}(d_k)$ и вычисляет величины v_j и τ_j, $j \in M_{11}(d_k)$.

Множество $M_{11}(d_k)$ определяется по правилу:

$$M_{11}(d_k) = \{ j : j \in M_1,\ L_j > 0,\ Q_j \geq d_{k+1};\ \exists\, i \in N_1(j),\ d_k \leq a_{ij} \leq d_{k+1} \}$$

Для каждого $j_0 \in M_{11}(d_k)$ положить

$$\tau_{j_0} = \min_{i \in N_1(j_0)} \{ a_{ij_0} : d_k \leq a_{ij_0} \};\quad v_{j_0} = \min(d_{k+1} - \tau_{j_0}; L_{j_0}).$$

3. Процедура $\Pi_2(i_0, j_0)$ назначает работу $i_0 \in N_2(d_k)$ на процессор $j_0 \in M_{11}(d_k)$ в интервале $[d_k;d_{k+1}]$.

Положить $\tau = \min(t_{i_0}; v_{j_0})$.

Для всех работ $i \in N_1(j_0)$, у которых $a_{ij_0} \geq \tau_{j_0}$, положить $a_{ij_0} = a_{ij_0} + \tau$.

Назначить работу i_0 на процессор j_0 в интервале $[\tau_{j_0};\tau_{j_0}+\tau]$.

Положить $t_{i_0}=t_{i_0}-\tau;\quad L_{j_0}=L_{j_0}-\tau;\quad \tau_{j_0}=\tau_{j_0}+\tau;$ $Q_{j_0}=Q_{j_0}+\tau;\ v_{j_0}=v_{j_0}-\tau.$

Работу i_0 исключить из $N_2(d_k)$. Если $t_{i_0}\neq 0$ и $k<s$, то работу i_0 включить в $N_2(d_{k+1})$.

Если для j_0 условие принадлежности множеству $M_{11}(d_k)$ не выполняется, то исключить j_0 из $M_{11}(d_k)$.

4. Процедура $M_{12}(d_k)$ строит множество $M_{12}(d_k)$ и вычисляет величины v_j и τ_j, $j\in M_{12}(d_k)$.

Множество $M_{12}(d_k)$ определяется по правилу:

$M_{12}(d_k)=\{j: j\in M_1,\ Q_j<d_{k+1}\}$.

Для каждого $j_0\in M_{12}(d_k)$ положить

$\tau_{j_0}=\max\{Q_{j_0};d_k\};$

$v_{j_0}=d_{k+1}-\tau_{j_0}.$

5. Процедура $\Pi_3(i_0,j_0)$ назначает работу $i_0\in N_2(d_k)$ на процессор $j_0\in M_{12}(d_k)$ в интервале $[d_k;d_{k+1}]$.

Положить $\tau=\min(t_{i_0};v_{j_0}).$

Работу i_0 назначить на процессор j_0 в интервале $[\tau_{j_0};\tau_{j_0}+\tau]$.

Положить $\tau_{j_0}=\tau_{j_0}+\tau;\quad v_{j_0}=v_{j_0}-\tau;\quad Q_{j_0}=Q_{j_0}+\tau_{j_0}.$

Перейдем к описанию алгоритма 1 распределения прерываемых работ и построения окончательного расписания вы-

полнения работ *N*. Алгоритм является обобщением известного однопроцессорного алгоритма относительной срочности (RU-алгоритм Э.Г. Коффмана [6]), который при $N_1 = \varnothing, m = 1$ и $p_i = f_i$ находит допустимое расписание, если оно существует.

***Алгоритм* 1**

1. Положить $k = 1$.
2. Включить в $N_2(d_k)$ все работы $i \in N_2$, для которых $b_i = d_k$.
3. Если $k < s$, перейти на шаг 4,
   если $k = s$, перейти на шаг 10.
4. Положить $\tau_j = d_k, \quad v_j = d_{k+1} - d_k$ для $j \in M_2$.
5. С помощью процедуры $M_{11}(d_k)$ построить множество $M_{11}(d_k)$ и вычислить величины τ_j и v_j, $j \in M_{11}(d_k)$.
   С помощью процедуры $M_{12}(d_k)$ построить множество $M_{12}(d_k)$ и вычислить величины τ_j и v_j, $j \in M_{12}(d_k)$.
   Для $j \in M_1 \setminus (M_{11}(d_k) \cup M_{12}(d_k))$ положить $v_j = 0$.
6. Если $N_2(d_k) = \varnothing$, перейти на шаг 9;
   если $N_2(d_k) \neq \varnothing$, перейти на шаг 7.
7. Пусть $\max\limits_{j \in M} v_j = v_{j_0}$.
   Если $v_{j_0} = 0$, перейти на шаг 9;
   если $v_{j_0} \neq 0$, перейти на шаг 8.
8. Пусть i_0 - номер первой в списке $N_2(d_k)$ работы.
   Если $j_0 \in M_2$, выполнить процедуру $\Pi_1(i_0, j_0)$;
   если $j_0 \in M_{11}(d_k)$, выполнить процедуру $\Pi_2(i_0, j_0)$;
   если $j_0 \in M_{12}(d_k)$, выполнить процедуру $\Pi_3(i_0, j_0)$.
9. Положить $k = k + 1$. Перейти на шаг 2.
10. Положить $\tau_j = max(Q_j; d_s)$ для $j \in M_1$ и $\tau_j = d_s$ для

$j \in M_2$.

11. Если $N_2(d_s) = \varnothing$, остановиться, расписание построено; если $N_2(d_s) \neq \varnothing$, перейти на шаг 12.
12. Пусть i_0 - первая в списке $N_2(d_s)$ работа; $\min\limits_{j \in M} \tau_j = \tau_{j_0}$. Назначить работу i_0 на процессор j_0 в интервале $[\tau_{j_0}; \tau_{j_0} + t_{i_0}]$; исключить работу i_0 из $N_2(d_s)$; положить $\tau_{j_0} = \tau_{j_0} + t_{i_0}$; перейти на шаг 11.

Дадим некоторые пояснения к алгоритму 1. На шаге 2 к работам, которые могли быть включены в $N_2(d_k)$ при выполнении процедур Π_1 и Π_2, добавляются работы с начальным директивным сроком b_i, равным d_k. На шаге 5 определяются процессоры $j \in M_1$, которые могут выполнять в интервале $[d_k; d_{k+1}]$ прерываемые работы. На шаге 7 определяется процессор, который может выделить в интервале $[d_k; d_{k+1}]$ наибольший объем процессорного времени. На шаге 8 выбирается наиболее срочная из числа готовых к выполнению работ из N_2 и назначается на процессор, выбранный на шаге 8. Шаги 10–12 описывают построение расписания после момента d_s, которое строится по «жадному» алгоритму.

Алгоритм 2 отличается от алгоритма 1 тем, что расписание выполнения работ N_1 остается неизменным, а процессор $j \in M_1$ может использоваться только после момента Q_j. На шаге 5 вызывается только процедура $M_{12}(d_k)$ (процедура $M_{11}(d_k)$ никогда не вызывается) и полагается $v_j = 0$ для $j \in M_1 \setminus M_{12}(d_k)$. На шаге 8 проверка $j_0 \in M_{11}(d_k)$ не выпол-

няется (и поэтому процедура Π_2 вообще никогда не вызывается). Остальные шаги алгоритма 1 не изменяются.

4. Вычислительная сложность алгоритма

Определим сначала вычислительную сложность процедур, используемых в предложенных алгоритмах. Вычислительная сложность процедуры $\Pi_1(i_0, j_0)$ есть $O(1)$, процедуры $\Pi_2(i_0, j_0)$ – $O(|N_1|)$, процедуры $\Pi_3(i_0, j_0)$ – $O(1)$, процедуры $M_{11}(d_k)$ – $O(m_1|N_1|)$, процедуры $M_{12}(d_k)$ – $O(m_1)$. Кроме того, $s = O(|N_2|)$. Шаги 2 – 10 выполняются s раз, поэтому сложность этой части алгоритма составляет $O(m_1|N_1||N_2|)$. Сложность части алгоритма, соответствующей шагам 11, 12, составляет $O(m|N_2|)$. Таким образом, вычислительная сложность алгоритмов 1 и 2 составляет $O(|N_2|\cdot\max(m_1|N_1|, m))$.

5. Результаты численных экспериментов

Были проведены численные эксперименты. Число работ n полагалось равным 100, 400 и 1000, а число процессоров m – 20, 60 и 100. Результаты экспериментов приведены в табл., в которой указаны значения величины штрафа F. Здесь n_1 – это число непрерываемых работ, а столбцы p_1 и p_2 соответствуют первому и второму вариантам вычисления значений p_i, характеризующих срочность работ.

Для получения входных данных программы использовался стандартный генератор случайных чисел, порождающий числа, равномерно распределенные на заданном отрезке. При этом были сделаны следующие допущения:

1. Директивный срок начала выполнения каждой работы не превышает 50% от идеальной оценки времени выполнения всего набора заданий при возможности обработки всех заданий на любом процессоре и отсутствии директивных сроков; т.е. $b_i \le \alpha(\sum_{i=1}^{N} t_i / M)$, $\alpha = 0{,}5$.

Таблица. Результаты численных экспериментов

				Алгоритм 1		Алгоритм 2	
n	n_1	m	m_1	p_1	p_2	p_1	p_2
100	01	**20**	1	32.00	31.30	44.60	40.70
	10		3	22.70	15.70	21.70	14.70
	20		3	26.70	25.70	23.10	22.00
	30		5	25.60	24.60	30.40	21.50
	40		7	22.90	19.60	22.10	20.60
	50		9	17.70	8.50	17.40	13.60
	60		11	28.40	19.20	23.40	22.10
	70		14	17.50	18.10	36.50	27.40
	80		15	2.70	0.0	0.0	6.60
	90		18	7.60	0.0	40.00	47.00
	99		19	0.0	0.0	0.0	0.0
400	001	**60**	1	80.30	76.60	89.70	84.00
	050		7	85.10	75.90	86.20	78.90
	100		14	78.10	73.10	81.10	74.30
	150		22	78.40	72.00	76.40	70.90
	200		29	74.20	64.00	75.20	66.20
	250		37	67.50	66.90	72.50	67.90
	300		44	56.70	53.50	61.90	61.20
	350		52	52.70	48.20	79.70	78.20
	399		59	0.0	0.0	0.0	0.0
1000	001	**100**	1	152.90	145.50	164.60	156.40
	100		9	152.50	142.20	151.50	141.50
	200		20	149.90	142.50	155.90	146.80
	300		29	159.60	148.70	158.60	148.70
	400		39	145.20	133.10	152.20	139.80
	500		50	136.10	131.40	139.10	137.40
	600		59	143.40	136.80	145.40	137.80
	700		69	123.40	119.60	138.50	136.10
	800		80	127.80	121.50	161.80	156.10
	900		90	64.50	59.00	169.90	169.80
	999		99	0.0	0.0	0.0	0.0

2. Директивный срок окончания каждой работы не должен быть меньше, чем директивный срок начала ее выполнения плюс ее длительность, умноженная на некий коэффициент, больший единицы, т.е. $f_i \geq b_i + \beta * t_i \mid \beta \geq 1$.

Возможность задания различных значений множителей α и β позволяет исследовать более широкий диапазон генерируемых значений директивных сроков работ.

3. Длительность выполнения каждой работы не меньше 1. Более точно, указанная величина вычислялась по формуле t_i = random (V_{max}) + 1, где random (V_{max}) – обращение к генератору случайных чисел, V_{max} – предельная величина длительности выполнения работ (она задается среди входных данных программы). Для проведенной серии расчетов величина V_{max} полагалась равной 26. Таким образом, порождаемые длительности работ колебались от 1 до 27.

Из результатов численных экспериментов можно сделать следующие выводы.

1. Алгоритм 1 в 87 % случаев дал меньшие значения штрафа, чем Алгоритм 2, и поэтому является более точным.

2. Для обоих алгоритмов в подавляющем числе случаев при вычислении срочности выполнения работ по второму варианту были получены меньшие значения штрафа по сравнению с первым вариантом.

3. При увеличении доли числа непрерываемых работ в общем числе работ значение штрафа уменьшается.

Литература

1. *Фуругян М.Г., Гончар Д.Р.* Минимаксная задача планирования вычислений в многопроцессорной системе со смешанным набором работ. // Системы управления и информационные технологии. 2009, № 2 (36). С. 36–39. Москва-Воронеж: Научная книга, 2009.

2. *Буланже Д.Ю., Сушков Б.Г.* Алгоритмы управления вычислительными системами жесткого реального времени. // Изв. АН СССР, Техн. кибернетика, 1982, № 6. С. 160–169.

3. *Буланже Д.Ю.* Оптимальная коррекция директивных интервалов для задачи одного прибора. М.: ВЦ АН СССР, 1983.

4. *Скиндерев С.А., Фуругян М.Г.* Алгоритмы планирования вычислений в многопроцессорных системах с неоднородным множеством работ. М.: ВЦ РАН, 2006.

5. *Гончар Д.Р.* Мультиоценочный алгоритм решения минимаксной задачи составления расписания // Системы управления и информационные технологии, 2007. № 1.3 (27). Москва-Воронеж: Научная книга, 2007. С. 324–328.

6. *Коффман Э.Г.* Теория расписаний и вычислительные машины. М.: Наука, 1984.

АЛГОРИТМЫ РАСПРЕДЕЛЕНИЯ РЕСУРСОВ В МНОГОПРОЦЕССОРНЫХ СИСТЕМАХ С НЕФИКСИРОВАННЫМИ ПАРАМЕТРАМИ

Е.О. Косоруков, М.Г. Фуругян

Рассматривается задача составления допустимого расписания с прерываниями в многопроцессорной системе в случае, когда заданы директивные интервалы, а длительности выполнения работ линейно зависят от количества выделенного им дополнительного ресурса.

1. Постановка задачи

Рассматривается вычислительная система, состоящая из m идентичных процессоров, и множество работ $N = \{1, 2, ..., n\}$. Предполагается, что в каждый момент времени каждый процессор может выполнять не более одной работы, а каждая работа выполняется не более чем одним процессором. При выполнении работ допускаются прерывания и переключения с одного процессора на другой. Предполагается, что прерывания и переключения не сопряжены с временными затратами. Для работы $i \in N$ установлен директивный интервал $(b_i, f_i]$ (т.е. работа i может выполняться только в этом интервале). Помимо процессоров в системе имеется дополнительный ресурс невозобновляемого типа. Суммарное количество этого ресурса составляет R. Если работе i выделено r_i единиц дополнительного ресурса, то ее длительность составляет $t_i = d_i - a_i r_i$, где

$$r_i \in [0, \overline{r_i}], i = 1, ..., n, \quad (1)$$

$$\sum_{i \in N} r_i \le R, \quad (2)$$

$a_i, d_i, \bar{r}_i$ – заданные величины, $a_i > 0$, $d_i > 0$, $0 \le \bar{r}_i < d_i / a_i$. Таким образом, $t_i \in [d_i - a_i \bar{r}_i, d_i]$ причем $d_i - a_i \bar{r}_i > 0$. Требуется найти такое распределение ресурсов $(r_1^0, r_2^0, ..., r_n^0)$, при котором существует допустимое расписание (каждая работа выполняется в своем директивном интервале). При этом распределение ресурсов $(r_1^0, r_2^0, ..., r_n^0)$ должно удовлетворять ограничениям (1), (2). В случае, когда указанного распределения ресурсов не существует, требуется определить директивные интервалы $(b_i, f_i']$, при которых указанное распределение ресурсов существует и величина $\max_{i \in N}(f_i' - f_i)$ принимает минимальное значение.

Подобная задача для случая, когда дополнительный ресурс отсутствует, рассматривалась в [1] и была сведена к задаче о максимальном потоке в сети специального вида. Задача с произвольными процессорами и произвольными директивными интервалами рассматривалась в [2], а задача с произвольными процессорами и одинаковыми директивными интервалами – в [3]. В обеих этих работах дополнительный ресурс не рассматривался. Задача минимизации времени выполнения сетевого комплекса работ, когда длительности выполнения работ являются функциями от вектора распределения ресурсов, а число процессоров не ограничено, рассматривалась в [4] и была сведена к задаче нелинейного программирования.

2. Сведение исходной задачи к задаче о потоке минимальной стоимости

В общем случае, когда директивные интервалы произвольные, по аналогии с тем, как это сделано в [1], построим потоковую сеть $G = (V, A)$ (рис. 1) 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 табл. 1.

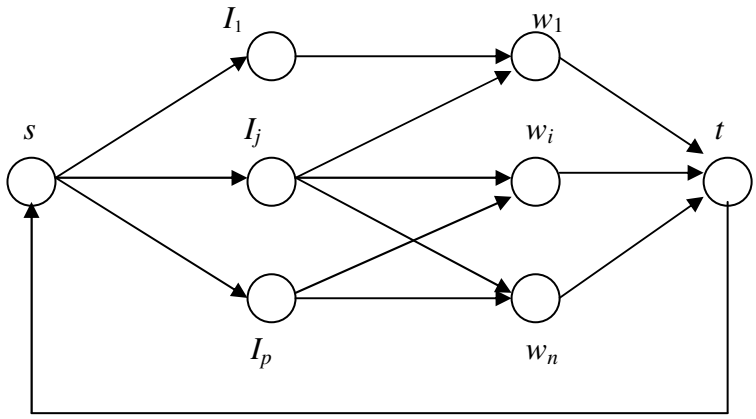

Рис.1. Потоковая сеть G

Таблица 1. 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

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

$$c(g) \le R - \sum_{i \in N} d_i / a_i \,. \qquad (3)$$

Доказательство теоремы содержится в [5, 6]. Предположим, что $c(g) > R - \sum_{i \in N} d_i / a_i$. Поскольку общее количество ресурса фиксировано, то добиться выполнения неравенства (3) за счет увеличения величины R нельзя.

В дальнейшем будем предполагать, что директивные интервалы имеют вид $(0, f_i]$, $i \in N$. Без ограничения общности можно считать, что $f_i < f_{i+1}$ при всех $i \in N$, т.е. работы упорядочены по возрастанию правых границ их директивных интервалов. Построим для этого случая потоковую сеть и будем обозначать ее G^* (рис. 2). Заметим, что $I_1 = (0, f_1]$, $I_2 = (f_1, f_2]$,..., $I_n = (f_{n-1}, f_n]$. Таким образом, особенностью данной сети является то, что из узла I_j исходят дуги, ведущие в узлы $\{w_j, w_{j+1}, ..., w_n\}$. Пусть g – циркуляция минимальной стоимости в сети G^*, а $c(g)$ – ее стоимость. Обозначим потоки по дугам $\langle s, I_j \rangle$, $\langle I_j, w_i \rangle$ и $\langle w_i, t \rangle$ через $g_{\langle s, I_j \rangle}$, $g_{\langle I_j, w_i \rangle}$ и $g_{\langle w_i, t \rangle}$ соответственно.

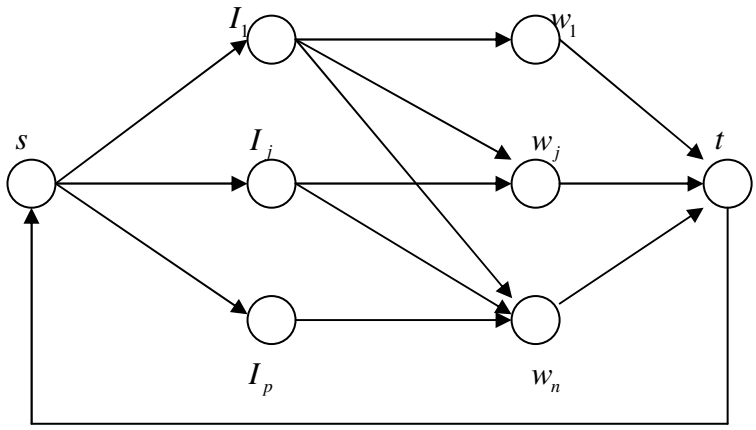

Рис. 2. Потоковая сеть G^*

Увеличим правые границы директивных интервалов всех работ на величину $\varepsilon > 0$: $f_i' = f_i + \varepsilon$. Из постановки задачи следует, что правые границы всех директивных интервалов можно увеличивать на одну и ту же величину. Получим, что $\Delta_1^* = \Delta_1 + \varepsilon$, $\Delta_j^* = \Delta_j$, $j \in N \setminus \{1\}$. Это означает, что после такого увеличения всех директивных интервалов изменились пропускные способности дуг $\langle s, I_1 \rangle$ и $\langle I_1, w_i \rangle$, $i \in N$. Увеличение верхней границы пропускной способности дуг $\langle I_1, w_i \rangle$, $i \in N$ означает возможность увеличения потока по дугам $\langle w_i, t \rangle$, а поскольку стоимость потока по дугам $\langle w_i, t \rangle$ – величина отрицательная, то увеличение потока по ним снижает его стоимость.

Пусть $\delta = c(g) - \left[R - \sum_{i \in N} \frac{d_i}{a_i} \right]$. В дальнейшем будем рассматривать сеть G^*. Обозначим через $g^*_{\langle I_1, w_i \rangle}$ модифицированный поток по дугам $\langle I_1, w_i \rangle$. По дугам вида $\langle I_j, w_i \rangle$ $(j \geq 2)$ поток изменяться не будет. Имеем

$$\begin{aligned} g^*_{\langle I_1, w_i \rangle} &= g_{\langle I_1, w_i \rangle} + x_i, \\ g^*_{\langle I_j, w_i \rangle} &= g_{\langle I_j, w_i \rangle},\ j \in N \setminus \{1\}. \end{aligned} \tag{4}$$

Из неравенств $g_{\langle I_1, w_i \rangle} \leq \Delta_1$ и $g^*_{\langle I_1, w_i \rangle} \leq \Delta^*_1 = \Delta_1 + \varepsilon$ получаем первое ограничение на величины x_i:

$$x_i \leq \Delta_1 + \varepsilon - g_{\langle I_1, w_i \rangle}. \tag{5}$$

Из условия сохранения потока в каждом узле сети следует, что величина потока, входящего в вершину I_1, изменилась на величину $\sum_{i \in N} x_i$, откуда следует второе ограничение:

$$\sum_{i \in N} x_i \leq m\varepsilon. \tag{6}$$

Поскольку ненулевую стоимость потока имеют только дуги $\langle w_i, t \rangle$, то общая стоимость потока изменится на величину $\sum_{i \in N} x_i \cdot \left(-\frac{1}{a_i} \right)$, откуда следует третье ограничение на x_i:

$$\sum_{i \in N} x_i \cdot \left(-\frac{1}{a_i} \right) + \delta \leq 0. \tag{7}$$

Минимизируя ε при указанных ограничениях (5) – (7), получаем следующую задачу линейного программирования:

$$\varepsilon \to \min,$$
$$x_i \le \Delta_1 + \varepsilon - g_{\langle I_1, w_i \rangle}, \quad i \in N,$$
$$\sum_{i \in N} x_i \le m\varepsilon,$$
$$\sum_{i \in N} x_i \cdot \left(-\frac{1}{a_i} \right) + \delta \le 0,$$
$$\varepsilon > 0, x_i \ge 0, i \in N.$$

3. Необходимое и достаточное условие существования допустимого расписания

Для произвольного подмножества $\overline{N} \subseteq N$ определим множество номеров l интервалов I_l

$$I(\overline{N}) = \{ l : 1 \le l \le p, \text{существует } i \in \overline{N}, I_l \in (b_i, f_i] \}$$

и величину $\overline{n}(l) = \left| \{ i \in \overline{N} : I_l \in (b_i, f_i] \} \right|$. Как следует из [1], допустимое расписание существует тогда и только тогда, когда для любого подмножества $\overline{N} \subseteq N$ выполняется неравенство

$$\sum_{i \in \overline{N}} t_i \le \sum_{l \in I(\overline{N})} \Delta_l \, min\left(m, \overline{n}(l)\right). \qquad (8)$$

При этом достаточно рассмотреть только такие подмножества $\overline{N}$, для которых $I(\overline{N}) = \{l_1, l_1 + 1, ..., l_2\}$ при некоторых l_1, l_2. Поскольку директивные интервалы имеют вид $(0, f_i]$, $i \in N$ и $f_i < f_{i+1}$ при всех $i \in N$, достаточно рассматривать только подмножества $\overline{N}$ следующего вида: $\{1\}$, $\{1,2\}$, $\{1,2,3\}$, ..., $\{1,2,...,n\}$. Для подмножества $\{1\}$ условие (8) имеет вид $t_1 \le \Delta_1$; для подмножества $\{1,2\}$ – $t_1 + t_2 \le \Delta_1 \cdot 2 + \Delta_2$. В общем случае для подмножества

$\{1,2,...,k\}$ имеем $\sum_{i=1}^{k} t_i \le \sum_{i=1}^{k} \Delta_i \cdot \min(m, k-i+1)$. Увеличим правые границы директивных интервалов всех работ на ε. Получим, что $\Delta_1^* = \Delta_1 + \varepsilon$ и $\Delta_j^* = \Delta_j$, $j \in N \setminus \{1\}$. Таким образом, для некоторого фиксированного распределения ресурсов, которое можно найти, например, с помощью алгоритма, описанного в разд. 2, получаем следующую задачу линейного программирования:

$$\varepsilon \to \min,$$

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

$$\sum_{i=1}^{k} t_i \le \sum_{i=1}^{k} \Delta_i^* \cdot \min(m, k-i+1), k = \overline{1,n}.$$

4. Приближенный алгоритм

В разд. 2 задача поиска модифицированного потока была сведена к задаче линейного программирования. В настоящем разделе мы рассмотрим вопрос о приближенном решении поставленной задачи.

Суть алгоритма заключается в последовательном насыщении дуг в соответствии с определенным приоритетом. Действительно, при достаточно большом значении ε для выполнения условия (3) можно ограничиться насыщением потока лишь по одной дуге. При уменьшении ε число таких дуг может возрасти. Фрагмент сети G^*, который будет рассматриваться в дальнейшем, изображен на рис. 3.

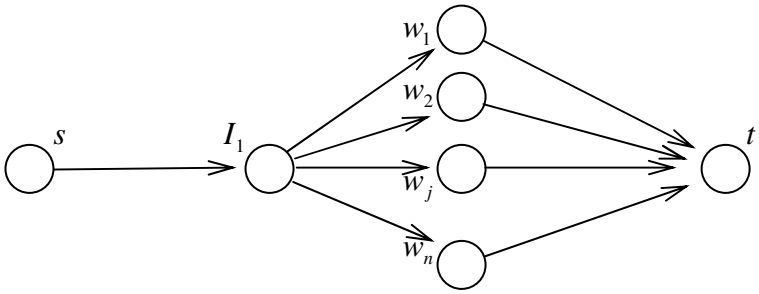

Рис. 3. Фрагмент сети G^*

Пусть по дугам $\langle s, I_1\rangle, \langle I_1, w_i\rangle, i \in N$ уже имеется некоторый поток (например, поток, который был получен для проверки условия (3)). Обозначим $\alpha_0 = m\Delta_1 - g_{\langle s, I_1\rangle}$ и $\alpha_i = \Delta_1 - g_{\langle I_1, w_i\rangle}$, $i \in N$. Величины α_0 и α_i характеризуют максимально допустимое увеличение потока по дугам $\langle s, I_1\rangle, \langle I_1, w_i\rangle, i \in N$. Отсюда следует, что величины $\left[-\frac{\alpha_i}{a_i}\right]$ характеризуют максимально возможное уменьшение стоимости потока по дугам $\langle w_i, t\rangle$. Без ограничения общности можно считать, что выполнено следующее соотношение:

$$\frac{\alpha_1}{a_1} \geq \frac{\alpha_2}{a_2} \geq \ldots \geq \frac{\alpha_n}{a_n}. \qquad (9)$$

В случае, когда условие (9) не выполнено, предлагаемый алгоритм легко может быть соответствующим образом модифицирован.

Рассмотрим случай, когда для выполнения условия (3) достаточно полностью насытить одну дугу. Пусть $\varepsilon_1 = \delta \cdot a_1$. Назначим по дуге $\langle I_1, w_1\rangle$ дополнительно $x_1 = \delta a_1 + \alpha_1$ единиц потока, а по дугам $\langle I_1, w_i\rangle, i \neq 1$ дополнительный поток не назначается, т.е. $x_i = 0$. Суммарная стоимость всего потока

уменьшится на величину $\sum_{i\in N}\frac{x_i}{a_i}=\delta+\frac{\alpha_1}{a_1}\geq\delta$. Это значит, что $c(g)-c(g^*)\geq\delta$. Поскольку $\delta=c(g)-\left[R-\sum_{i\in N}\frac{d_i}{a_i}\right]$, то $c(g^*)\leq R-\sum_{i\in N}\frac{d_i}{a_i}$, а значит допустимое расписание существует.

На первом шаге величина $\varepsilon_1=\delta\cdot a_1$ может оказаться достаточно большой и при этом полного насыщения достигает только дуга $\langle I_1,w_1\rangle$, в то время как для дуги $\langle s,I_1\rangle$ $\alpha_0=(m-1)\,\delta\cdot a_1-\alpha_1$, т.е. дуга $\langle I_1,w_1\rangle$ насыщается не полностью. На этом шаге будем предполагать, что насыщаются k наиболее приоритетных дуг, т.е. дуги $\langle I_1,w_1\rangle,...,\langle I_1,w_k\rangle$. Насыщая большее количество дуг, величина ε уменьшается. На *k-м шаге* $\varepsilon_k=(\delta-\sum_{i=1}^{k}\frac{\alpha_i}{a_i})/\sum_{i=1}^{k}\frac{1}{a_i}$. Аналогично тому, как показано на шаге 1, можно показать, что $c(g^*)\leq R-\sum_{i\in N}\frac{d_i}{a_i}$. Условие $k<m$ необходимо для того, чтобы, насыщая k дуг, поток по дуге $\langle s,I_1\rangle$ не превысил ее пропускную способность. Суммарный поток по дугам $\langle I_1,w_1\rangle,...,\langle I_1,w_k\rangle$ может увеличиться не более чем на величину $k\varepsilon+\sum_{i=1}^{k}\alpha_i$, а по дуге $\langle s,I_1\rangle$ – не более чем на $m\varepsilon+\alpha_0$, где $\alpha_0\geq\sum_{i=1}^{n}\alpha_i\geq\sum_{i=1}^{k}\alpha_i$.

Для ε_m имеем

$$\varepsilon_m = \frac{\delta - \sum_{i=1}^{m} \frac{\alpha_i}{a_i}}{\sum_{i=1}^{m} \frac{1}{a_i}}, \quad g_{\langle I_1, w_m \rangle} = \varepsilon_m - \max\left(-\alpha_m, \varepsilon_m - \alpha_0 + \sum_{k=1}^{m} \alpha_m \right).$$

Перейдем теперь к описанию алгоритма.

Шаг 1. Положить $\varepsilon = \delta \cdot a_1$, $x_1 = \delta a_1 + \alpha_1$. (Как показано выше, в этом случае $c\left(g^*\right) \le R - \sum_{i \in N} \frac{d_i}{a_i}$.)

Шаг 2. Положить $\varepsilon := \varepsilon / 2$.

Шаг 3. Если $\varepsilon \le \varepsilon_m$, перейти к *шагу 4.* Если $\varepsilon > \varepsilon_m$, перейти к *шагу 2.*

Шаг 4. Насытить дуги в порядке приоритета, пока выполнено условие $m\varepsilon + \alpha_0 \ge \sum_{i=1}^{s} x_i, s \le m$. Если последнее условие не выполнено и $c\left(g^*\right) \le R - \sum_{i \in N} \frac{d_i}{a_i}$, перейти к *шагу 6,* в противном случае – к *шагу 5.*

Шаг 4. Положить $\varepsilon := 1{,}5\varepsilon$. Перейти к *шагу 3.*

Шаг 5. Завершение алгоритма.

Данный алгоритм позволяет в случае, когда m не намного меньше n, с небольшими вычислительными затратами получать хорошие оценки для ε. В табл. 2 показано, на сколько процентов значения ε, полученные приближенным алгоритмом, отличаются от точных значений, полученных методом, описанным в разд. 2.

Таблица 2. Сравнение точного и приближенного методов

Параметры		Приближенный
m/n	δ	Отклонение (%)
0,01	10	300-1000
0,1	10	100-400
0,5	10	10-120
0,9	10	1-20
0,1	1	80-200
0,9	1	10-35
0,1	0,1	40-90
0,9	0,1	1-8

Литература

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

2. *Federgruen A., Groenevel H.T.* Preemptive scheduling of uniform machines by ordinary network flow technique// Management Science, 1986, 32, 3. Pp. 341 –349.

3. *Gonzales T., Sahni S.* Preemptive scheduling of uniform processor systems//Journal of the Association for Computing Machinery, 1978, 25, 1. Pp. 92 – 101.

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

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

6. *Косоруков Е.О., Фуругян М.Г.* Некоторые алгоритмы распределения ресурсов в системах с нефиксированными длительностями работ. М.: ВЦ РАН, 2009.

ПАРАЛЛЕЛЬНЫЙ ПСЕВДОПОЛИНОМИАЛЬНЫЙ АЛГОРИТМ РЕШЕНИЯ МИНИМАКСНОЙ ЗАДАЧИ ТЕОРИИ РАСПИСАНИЙ

М.Г. Фуругян

Предлагается псевдополиномиальный алгоритм построения многопроцессорного оптимального по быстродействию расписания без прерываний и переключений при фиксированном числе процессоров. Определяется необходимое число процессоров для эффективного распараллеливания алгоритма.

1. Постановка задачи

Имеется вычислительная система, состоящая из m идентичных процессоров. Задано множество работ $N = \{1,\ldots, n\}$, каждая из которых может выполняться на любом процессоре. В фиксированный момент времени каждая работа может выполняться не более чем одним процессором, и каждый процессор может выполнять не более одной работы. При выполнении работ не допускаются прерывания и переключения с одного процессора на другой. Длительность выполнения работы $i \in N$ равна t_i. Задан общий директивный срок T выполнения работ N. Предполагается, что $t_i \le T$ при всех $i \in N$. Требуется определить, существует ли допустимое расписание выполнения работ N (т.е. такое расписание без прерываний и переключений, при котором выполнение каждой работы завершается не позднее момента времени T), и построить его, если оно существует.

Как известно [1], данная задача является *NP*-трудной даже при фиксированном m, а при произвольном m – *NP*-трудной в сильном смысле. Алгоритмы решения аналогичной задачи на быстродействие широко освещены в литературе. Отметим, например, такие методы, применяемые при их реше-

нии, как случайный и исчерпывающий поиск [2], методы математического программирования [3], метод ветвей и границ [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. В настоящей работе будет описан псевдополиномиальный алгоритм и его параллельная реализация.

2. Псевдополиномиальный алгоритм

Будем строить допустимое расписание с помощью (n+1)-уровневого дерева решений, каждый лист которого соответствует одному из возможных вариантов распределения работ по процессорам при заданном директивном сроке. Корень дерева (нулевой уровень) соответствует множеству всех вариантов распределения. С корнем ребрами связаны m вершин первого уровня, соответствующих множеству всех вариантов распределения, в которых первая работа назначена на определенный процессор. Каждой такой вершине соответствует m-мерный вектор, j-я компонента которого равна временной загруженности j-го процессора. Таким образом, вершинам первого уровня дерева решений соответствуют m-мерные векторы (t_1, 0, …, 0), (0, t_1, 0, …, 0), …, (0, …, 0, t_1) (всего m векторов). Каждая вершина первого уровня связана ребрами с m вершинами второго уровня, соответствующими множеству всех вариантов распределения работ по процессорам, в которых первые две работы закреплены за определенными процессорами. Например, вершина первого уровня (0, …, 0, t_1, 0, …, 0) связана с m вершинами второго уровня, которым соответствуют m-мерные векторы (t_2, 0, …, 0, t_1, …, 0), (0, t_2, 0, …, 0, t_1, …, 0), …, (0, …, t_2, t_1, 0, …,0), (0, …, t_1+t_2, 0,…,0),…,(0, …, t_1,t_2, …, 0), (0, …, t_1, 0, …, t_2). Далее, с вершинами второго уровня дерева решений связаны вершины третьего уровня и т.д.

Если хотя бы одна компонента вектора, приписываемого вершине, превышает величину T, то данная вершина в дерево решений не включается и дальнейшее ветвление из нее не производится. Допустимое расписание в поставленной задаче существует в том случае, если построенное дерево решений содержит хотя бы одну вершину n-го уровня (т.е. если вектор $(\tau_1,...,\tau_m)$, соответствующий некоторой вершине n-го уровня, содержится в m-мерном кубе с ребром T, или $\tau_j \le T$ при всех j=1, …, m). Для построения допустимого расписания следует построить путь, соединяющий вершину n-го уровня с корнем дерева решений. Если вершина k-го уровня (k = 1, 2, …, n) в этом пути соответствует процессору j, то работа k назначается на процессор j.

Определим вычислительную сложность предложенного алгоритма. Число вершин дерева решений, которые могут быть получены в результате работы алгоритма, не превосходит числа точек с целочисленными координатами в m-мерном кубе с ребром T, т.е. $(T+1)^m$. Для каждой вершины строится не более m вершин следующего уровня. Таким образом, сложность алгоритма составляет $O(m(T+1)^m)$ операций с m-мерными векторами. Выполнение одной операции заключается в сложении одной компоненты вектора с числом t_i ($i \in N$). Максимальное значение одной компоненты каждого вектора равно $\sum_{i\in N} t_i$, т.е. длина слова, соответствующего одной компоненте каждого вектора, с которым работает алгоритм, есть $O(\log_2(\sum_{i\in N} t_i))$. Таким образом, предложенный алгоритм является псевдополиномиальным при фиксированном значении m.

3. Параллельная реализация алгоритма

Параллельная реализация предложенного алгоритма заключается в том, что все вершины одного уровня дерева решений обрабатываются одновременно, причем с каждой верши-

ной параллельно работают m процессоров и генерируют вершины следующего уровня. Вершины уровня $k+1$ ($k = 0, 1, \ldots, n-1$) обрабатываются только после того, как будут обработаны все вершины уровня k (т.е. после того, как будут получены все вершины уровня $k+1$).

Пусть L_k – число вершин k-го уровня в дереве решений и $L = \max\limits_{k=0,\ldots,n} L_k$. Тогда необходимое число процессоров для реализации данного алгоритма составляет mL. Если предположить, что все вершины обрабатываются одно и то же время, то при обработке вершин k-го уровня ускорение (по сравнению с однопроцессорным алгоритмом) составит mL_k ($k = 1, 2, \ldots, n$).

Поскольку при работе данного алгоритма несколько процессоров будут одновременно обрабатывать одну и ту же вершину дерева решений и, кроме того, возможна ситуация, когда результатом работы нескольких процессоров является один и тот же m-мерный вектор (точка m-мерного куба), то предложенный алгоритм может быть реализован на CRCW-машине (машине с одновременным чтением и одновременной записью общего значения [12]).

Перейдем к вычислению величин L_k. Для этого сначала вычислим все возможные значения, не превосходящие T, которые могут принимать суммы $\sum\limits_{i\in\tilde{N}} t_i$ для всех подмножеств $\tilde{N} \subseteq N$. Воспользуемся алгоритмом, предложенным в [1].

Пусть $P(i, j)$ ($i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, T$) – массив размером $n \times T$, такой что $P(i, j) = 1$, если существует подмножество $\overline{N} \subseteq \{1, 2, ..., i\}$, для которого $\sum\limits_{i\in\overline{N}} t_i = j$, и $P(i, j) = 0$ в противном случае. Если массив P будет построен, то множество всех возможных значений величины $\sum\limits_{i\in\tilde{N}} t_i$ ($\tilde{N} \subseteq N$) можно определить как множество $\{ j : 1 \le j \le T, P(n, j) = 1\}$.

Опишем алгоритм построения массива P. Вычисления проводятся последовательно для каждой строки, начиная с первой.

1) Положить $P(i, j) = 0$ для всех $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, T$.

2) Положить $P(1, t_1) = 1$.

3) Для всех $i = 2, \ldots, n$ выполнять шаги 4 – 6.

4) Если $P(i\text{-}1, j) = 1$, то положить $P(i, j)=1$ $(j = 1, 2, \ldots, T)$.

5) Если $P(i\text{-}1, j) = 1$ и $j + t_i \le T$, то положить $P(i, j+t_i) = 1$ $(j = 1, 2, \ldots, T)$.

6) Положить $P(i, t_i) = 1$.

Описанный алгоритм для каждого $i = 1, 2, \ldots, n$ вычисляет все возможные значения сумм $\sum_{k \in \overline{N}} t_k$, $\overline{N} \subseteq \{1,2,...,i\}$. Для определения вычислительной сложности предложенного алгоритма отметим, что для каждого элемента массива P выполняется не более фиксированного числа операций, а число элементов массива равно nT. Таким образом, вычислительная сложность алгоритма построения массива P составляет $O(nT)$ операций сложения, сравнения и присваивания с исходными данными.

Пусть $\{ j : 1 \le j \le T, P(n, j) = 1\} = \{a_1, \ldots, a_v\}$, причем $0 < a_1 < a_2 < \ldots < a_v \le T$, т.е. a_j, $j = 1, 2, \ldots, v$ – это все возможные значения, не превосходящие T, которые могут принимать суммы $\sum_{i \in \tilde{N}} t_i$, $\tilde{N} \subseteq N$. Пусть, далее, $C(S, m, p)$ – это количество разбиений натурального числа S на m неотрицательных целых слагаемых, каждое положительное из которых совпадает с одним из значений a_j $(1 \le j \le p)$. Каждое разбиение может содержать одинаковые слагаемые, в частности несколько слагаемых, равных нулю. Тогда число L_k вершин уровня k

в дереве решений может быть вычислено как $L_k=C(\sum_{i=1}^{k} t_i, m, v)$.

Отметим, что $\sum_{i=1}^{k} t_i = a_p$ при некотором p, $1 \le p \le v$.

Приведем рекуррентные соотношения для вычисления величин $C(S, m, p)$ $(S \le \sum_{i \in N} t_i,\ S \in \{a_1,...,a_v\},\ 1 \le p \le v)$. Число $C(S, m, a_p)$ равно сумме следующих величин:

- числа разбиений S на m неотрицательных целых слагаемых, каждое положительное из которых совпадает с одним из a_j $(1 \le j \le p\text{-}1)$, т.е. $C(S, m, p\text{-}1)$;

- числа разбиений S на m неотрицательных целых слагаемых, в каждом из которых t слагаемых равны a_p ($t = 1, 2, \ldots;\ t \le m$), а остальные $m - t$ слагаемых не превосходят $a_{p\text{-}1}$ (т.е. равны величинам a_j ($j = 1, 2, \ldots, p - 1$) или 0).

Таким образом,

$$C(S, m, p) = C(S, m, p\text{-}1) + C_m^1 . C(S - a_p, m - 1, p\text{-}1) +$$
$$+ C_m^2 . C(S - 2a_p, m - 2, p\text{-}1) + C_m^3 . C(S - 3a_p, m - 3, p\text{-}1) + \ldots$$

Суммирование ведется до того значения $t \le m$, для которого выполняются оба соотношения $S - ta_p \ge 0$ и $m - t \ge 0$. Начальные условия определяются следующим образом:

$C(0, j, p) = 1$ при всех $j = 1, 2, \ldots, m;\ p = 1, 2, \ldots, v$;

$C(u, j, p) = 0$ при $u > p\,j;\ j = 1, 2, \ldots, m;\ p = 1, 2, \ldots, v$;

$C(u, t, 1) = C_u^t$. при $u \le t;\ t = 1, 2, \ldots, m$.

Если считать, что биномиальные коэффициенты C_u^t затабулированы в заранее созданном массиве чисел, т.е. для их определения требуется время $O(1)$, то вычислительная сложность нахождения величин $C(S, m, a_p)$ для всех $S \in \{a_1, \ldots, a_v\}$ и $p = 1, 2, \ldots, v$ составляет $O(v \max(v,m))$ или $O(T \max(T,m))$.

Рассмотрим пример вычисления величины C(S, m, a_p) при $S = 4$, $m = 3$, $p = 2$, $a_1 = 1$, $a_2 = 2$. Вычисления проводятся следующим образом:

$C(0, 1, 1) = 1,\ C(0, 1,2) = 1,\ C(0, 2, 1) = 1,\ C(0, 2, 2) = 1,$
$C(1, 1, 1) = 1,\ C(1, 2, 1) = 2,\ C(1, 3, 1) =3,$
$C(2, 1, 1) = 0,\ C(2, 2, 1) = 1,\ C(2, 3, 1) =3,$
$C(3, 1, 1) = 0,\ C(3, 2, 1) = 0,\ C(3, 3, 1) =1,$
$C(4, 1, 1) = 0,\ C(4, 2, 1) = 0,\ C(4, 3, 1) =0,$
$C(4, 3, 2) = C(4, 3, 1) + C_3^1 \cdot C(2, 2, 1) + C_3^2 \cdot C(0, 1, 1) =$
$= 0 + 3 \cdot 1 + 3 \cdot 1 = 6.$

Литература

1. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.

2. *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.

3. *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.

4. *Алексеев О.Г.* Комплексное применение методов дискретной оптимизации. М.: Наука, 1986.

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 исполь-

зованием 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. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы. Построение и анализ. М.: МЦНМО, 2002.

АЛГОРИТМЫ ОРГАНИЗАЦИИ КОНТРОЛЯ В СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ

М.Г. Фуругян

Рассматривается задача оптимизации структуры подсистемы контроля в вычислительных системах реального времени. Разработаны алгоритмы, позволяющие определить такое расположение модулей контроля и модулей-буферов, при котором математическое ожидание суммарного времени, затраченного на выполнение всех модулей (включая повторное их выполнение при возникновении ошибок), минимальное. Исследован случай нескольких параллельных цепочек рабочих модулей.

1. Введение

При функционировании вычислительных систем реального времени большое значение имеет подсистема организации рестартов, позволяющая в случае возникновения сбоев определить те программные модули, которые подлежат повторному запуску. Контроль поступающих в систему данных, а также данных, которыми обмениваются программные модули, осуществляется специальными модулями контроля, каждый из которых определяет, удовлетворяет ли множество значений контролируемых им параметров определенным условиям. Кроме того, в вычислительную систему реального времени вводятся дополнительные модули-буфера, сохраняющие промежуточную информацию для того, чтобы при рестарте системы воспользоваться данными из этих буферов, а не проводить все вычисления заново. Таким образом, наличие модулей контроля и модулей-буферов может существенно улучшить надежность системы и уменьшить временные задержки, вызванные несистематическими сбоями и ошибками. С другой стороны, избыточное использование этих модулей может при-

вести к существенному уменьшению быстродействия и увеличению стоимости системы. Поэтому возникает задача оптимальной частоты расположения модулей-буферов.

В настоящей работе рассматривается задача оптимальной расстановки модулей контроля и модулей-буферов в заданном графе, вершинами которого являются рабочие модули. Рассмотрен случай, когда граф зависимости по данным состоит из нескольких параллельных цепочек. Доказательство приводимых в статье утверждений содержится в [1], а подробный обзор литературы по данной тематике – в [2].

2. Постановка задачи

Дан ориентированный граф $G = <V, E>$, вершины V которого – рабочие модули $M_1, M_2, \ldots, M_n$, ребра E – зависимости по данным между ними (ребро (i, j) означает, что модуль i передает данные модулю j). Рабочие модули – это главные модули системы, выполняющие вычисления. Время выполнения каждого рабочего модуля предполагается одинаковым и равным единице. При выполнении каждого рабочего модуля с вероятностью $a > 0$ может возникнуть ошибка, и тогда его необходимо выполнить повторно. Помимо рабочих модулей в системе имеются модули контроля C_i, $i = 1,2,...,k$, которые проверяют рабочие модули на наличие ошибок, и модули-буфера B_i, $i = 1,2,...,m$, которые сохраняют полученные данные и передают их последующим модулям. Предполагается, что время работы каждого модуля контроля равно нулю, и он указывает на ошибку в том и только том случае, если хотя бы в одном из предшествующих ему по графу модулей произошла ошибка. Время работы каждого модуля-буфера также равно нулю. Для повторного выполнения рабочего модуля M_i нужно повторно выполнить все рабочие модули, которые непосредственно ему предшествуют, чтобы получить данные для M_i. Для каждого из его предшественников, в свою очередь, необходимо повторно

выполнить рабочие модули, предшествующие ему, и т. д., пока не дойдем до модулей-буферов.

Рабочие модули должны выполниться без ошибки. Каждый модуль контроля и модуль-буфер может располагаться в начале или в конце некоторого рабочего модуля. Такое соответствие между модулями будем называть расстановкой. Требуется расположить модули контроля и модули-буфера так, чтобы математическое ожидание суммарного времени, затраченного на выполнение всех модулей (включая повторное их выполнение при возникновении ошибок), было минимальным. В дальнейшем такую расстановку модулей контроля и модулей-буферов будем называть оптимальной. Минимум ищется по всем допустимым расстановкам. Расстановка называется допустимой, если ошибка в любом рабочем модуле может быть обнаружена некоторым модулем контроля и все данные могут быть восстановлены. Иными словами, для каждого рабочего модуля существует ориентированный путь, ведущий из этого модуля в некоторый модуль контроля, и ориентированный путь, ведущий из некоторого модуля-буфера в этот рабочий модуль (в том числе учитываются и пути, состоящие из одной вершины, для случаев, когда модуль контроля помещен в конец рабочего модуля или модуль-буфер помещен в начало рабочего модуля). Предполагается, что имеется только один процессор, т. е. параллельные вычисления невозможны. Кроме того, предполагается, что выполнены следующие условия.

1. Граф G состоит из l не связанных между собой цепочек, причем в i-й цепочке n_i рабочих модулей, $\sum_{i=1}^{l} n_i = n$.

2. Число рабочих модулей достаточно велико, т.е. $n >> k$. Это позволит считать, что с определенной степенью точности можно располагать модули контроля и модули-буфера в любой точке временного отрезка $[0, T]$, где T – это суммарное время выполнения всех модулей при условии, что ошибки не произойдет. При этих предположениях время выполнения рабоче-

го модуля равно единице, модуля контроля или модуля-буфера – нулю, и, следовательно, $T = n$.

3. $a << 1/n$, т. е. ошибка возможна, но маловероятна. В этом случае можно считать, что при работе всей системы ошибка может произойти не более одного раза. Случаем, когда после перезапуска системы ошибка произойдет снова, мы пренебрегаем как маловероятным. Это позволит в дальнейшем существенно упростить вычисления и получить точные формулы для места расположения модулей контроля и модулей-буферов.

Требуется расставить k модулей контроля и m модулей-буферов так, чтобы математическое ожидание M суммарного времени, затрачиваемого на выполнение всей совокупности модулей (включая повторное их выполнение при возникновении ошибок), было минимальным.

Из условия допустимости расстановки следует, что в начале каждой цепочки рабочих модулей необходимо расположить модуль-буфер, а в конце каждой цепочки необходим модуль контроля. Это значит, что при $k < l$ или $m < l$ допустимой расстановки не существует. В дальнейшем мы будем предполагать, что $k \geq l$, $m \geq l$, и рассматривать только допустимые расстановки.

Поскольку $n_i >> k$, $a << \frac{1}{n_i}$ $i = 1,...,l$, т.е. в каждой цепочке достаточно много рабочих модулей, а ошибка при их выполнении возможна, но маловероятна, то для оптимальной расстановки модулей контроля и модулей-буферов внутри каждой цепочки можно использовать алгоритм, описанный в [2]. Следовательно, надо только решить задачу оптимального распределения модулей контроля и модулей-буферов между цепочками рабочих модулей.

Пусть на i-ю цепочку рабочих модулей выделено k_i модулей контроля и m_i модулей-буферов. Произвольную допус-

тимую расстановку этих модулей внутри данной цепочки будем обозначать через $A'(k_i, m_i, n_i)$, а оптимальную расстановку– через $A'_{опт}(k_i, m_i, n_i)$. Через $MA'(k_i, m_i, n_i)$ (и соответственно $M'_{опт}(k_i, m_i, n_i)$) обозначим математическое ожидание суммарной длительности выполнения всех рабочих модулей данной цепочки (включая их повторное выполнение при возникновении ошибок). Общую допустимую расстановку k модулей контроля и m модулей-буферов по всей совокупности цепочек рабочих модулей будем обозначать $A(k, m, n)$, а оптимальную расстановку – $A_{опт}(k, m, n)$. Соответствующие им математические ожидания суммарной длительности выполнения всех модулей графа G будем обозначать $MA(k, m, n)$ и $M_{опт}(k, m, n)$.

Важно отметить, что, несмотря на появление независимых цепочек рабочих модулей, которые в принципе можно было бы выполнять параллельно, мы будем предполагать, что имеется только один процессор, который выполняет рабочие модули. В этом случае время, необходимое для выполнения всех рабочих модулей равняется сумме времен для каждой из цепочек, т.е.

$$MA(k, m, n) = \sum_{i=1}^{l} MA'(k_i, m_i, n_i).$$

Из этой формулы, в частности, следует, что в общей оптимальной расстановке $A_{опт}(k, m, n)$ расстановка внутри каждой цепочки тоже должна быть оптимальной. Кроме того, мы должны так распределить k модулей контроля и m модулей-буферов по l цепочкам, чтобы минимизировать $MA(k, m, n)$. Следовательно, для оптимальной расстановки получаем следующее выражение:

$$M_{опт}(k, m, n) = \min_{k_i, m_i : \sum k_i = k, \sum m_i = m} \sum_{i=1}^{l} M'_{опт}(k_i, m_i, n_i). \qquad (1)$$

В последующих разделах мы рассмотрим различные подходы к решению этой задачи.

3. Разделение модулей контроля по одинаковым цепочкам

В этом разделе мы рассмотрим частный случай поставленной задачи. Пусть имеется l одинаковых цепочек, в каждой из которых – по s рабочих модулей ($n = ls$), и на каждую из цепочек выделено по w модулей-буферов ($m = lw$). Нужно оптимально распределить k модулей контроля по l цепочкам.

Предположим сначала, что $m = l$, т.е. $w = 1$. В этом случае в начале каждой цепочки находится модуль-буфер и других модулей-буферов нет. Из результатов, полученных в [2], следует, что в этом случае внутри каждой цепочки модули контроля следует располагать равномерно, а минимальное математическое ожидание времени выполнения задается формулой

$$M'_{onm}(k_i,1,s) = s + a \cdot s^2 \frac{k_i + 1}{2k_i}.$$

В этом случае

$$\begin{aligned} M_{onm}(k,m,n) &= \min_{k_i:\sum k_i = k} \sum_{i=1}^{l} M'_{onm}(k_i,1,s) = \\ &= n + l \cdot a \cdot s^2 \min_{k_i:\sum k_i = k} \sum_{i=1}^{l} \frac{k_i + 1}{2k_i}. \end{aligned} \qquad (2)$$

Рассмотрим функцию $f(x) = \dfrac{x+1}{2x}$. Несложно заметить, что эта функция является выпуклой при $x > 0$. Действительно, $f'(x) = -0{,}5x^{-2}$, $f''(x) = x^{-3} > 0$ при $x > 0$.

Лемма 1. Пусть $f(x)$ – строго выпуклая функция. Рассмотрим задачу оптимизации $\min\limits_{k_i:\sum k_i = k} \sum\limits_{i=1}^{l} f(k_i)$, где $k_1, k_2, ..., k_l$ –

натуральные числа. Тогда для оптимального набора $k_1,k_2,...,k_l$ выполнено неравенство $\max\limits_i k_i - \min\limits_i k_i \le 1$.

Если $f(x)$ – нестрого выпуклая функция, то таким свойством обладает как минимум один из оптимальных наборов $k_1, k_2,..., k_l$.

Из леммы 1 и выпуклости функции $f(x) = \dfrac{x+1}{2x}$ следует, что минимум в выражении (2) достигается, если модули контроля распределять равномерно между цепочками рабочих модулей.

Рассмотрим теперь случай, когда w – произвольное. Тогда выражение (1) можно записать в виде

$$M_{опт}(k,m,n) = \min_{k_i:\sum k_i=k} \sum_{i=1}^{l} M'_{опт}(k_i, w, s), \qquad (3)$$

где $M^i_{опт}(k_i, w, s)$ можно вычислить по формулам, приводимым в [2]:

$$M_{опт}(k,m,n) = n + \frac{a\cdot n^2}{2m} + \frac{a\cdot n^2}{2k} + a\cdot n^2 \frac{r(m-r)}{2km(m(b^2+2b)+r)}. \qquad (4)$$

Лемма 2. Для любых натуральных k,h,m,n $(k > h)$ выполняется неравенство

$$M'_{опт}(k-h,m,n) + M'_{опт}(k+h,m,n) \ge 2M'_{опт}(k,m,n).$$

Из лемм 1, 2 следует теорема.

Теорема 1. Пусть имеется l одинаковых цепочек, каждая из которых состоит из s рабочих модулей ($n = ls$), и на каждую из них выделено по w модулей-буферов ($m = lw$). Пусть $k = l\cdot\overline{b} + \overline{r}$, $\overline{r} < l$. Тогда математическое ожидание времени выполнения всех модулей задается формулой

$$M_{опт}(k,m,n) = \overline{r}M'_{опт}(\overline{b}+1, w, s) + (l-\overline{r})M'_{опт}(\overline{b}, w, s),$$

где $M'_{onm}(\bar{b}+1,w,s)$ и $M'_{onm}(\bar{b},w,s)$ задаются формулой (4).

Из теоремы 1 следует способ построения оптимальной расстановки. Для этого следует назначить по $\bar{b}+1$ модулей контроля на некоторые $\bar{r}$ цепочек, по $\bar{b}$ модулей контроля – на остальные $l-\bar{r}$ цепочек и применить алгоритм оптимальной расстановки для каждой цепочки.

4. Распределение модулей контроля по цепочкам при заданном распределении модулей-буферов

Как и в предыдущем разделе, будем предполагать, что m модулей-буферов уже некоторым образом распределены по l цепочкам (m_i модулей-буферов в i-й цепочке). В предыдущем разделе мы доказали, что если $n_1 = n_2 = ... = n_l$ и $m_1 = m_2 = ... = m_l$, то модули контроля также следует располагать между цепочками равномерно. В этом разделе числа $n_1, n_2, ..., n_l$ и $m_1, m_2, ..., m_l$ не предполагаются равными.

В рассматриваемом случае выражение (1) можно записать в виде

$$M_{onm}(k,m,n) = \min_{k_i, m_i : \sum k_i = k} \sum_{i=1}^{l} M'_{onm}(k_i, m_i, n_i).$$

Поскольку величины $n_1, n_2, ..., n_l$ и $m_1, m_2, ..., m_l$ фиксированы, $M'_{onm}(k_i, m_i, n_i)$ можно обозначить как $q_i(k_i)$, и задача сводится к нахождению $\min_{k_i, \sum k_i = k} \sum_{i=1}^{l} q_i(k_i)$. В следующей лемме мы сформулируем алгоритм для решения этой задачи.

Лемма 3. Пусть $q_i(a)$ ($a \in N, i = 1,...,l$) – функции натурального аргумента, такие, что для каждой из них

$$q_i(a) - q_i(a+1) \le q_i(a-1) - q_i(a) \qquad (5)$$

при всех $a \geq 2$. Тогда для нахождения величины $\min\limits_{k_i:\sum k_i=k} \sum\limits_{i=1}^{l} q_i(k_i)$ существует алгоритм сложности не более $Cl^2 \ln^2 k$.

Доказательство. Для каждой пары натуральных a и i введем следующие обозначения: $r(a,i) = q_i(a) - q_i(a+1)$, $Q = \{(j,i) : j \leq k_i\}$. Тогда $q_i(k_i) = q_i(1) - \sum\limits_{j=1}^{k_i} r(j,i)$, $\sum\limits_{i=1}^{l} q_i(k_i) = \sum\limits_{i=1}^{l} q_i(1) - \sum\limits_{(j,i)\in Q} r(j,i)$. Поскольку величина $\sum\limits_{i=1}^{l} q_i(1)$ постоянна, то надо решить задачу $\max\limits_{Q} \sum\limits_{(i,j)\in Q} r(j,i)$. В последнем выражении содержится k слагаемых, так как $\sum\limits_{i=1}^{l} k_i = k$. Из (5) следует, что

$$r(j,i) \geq r(j+1,i) \tag{6}$$

при всех i, j. Это означает, что последняя задача эквивалентна задаче нахождения k максимальных чисел среди $r(j,i), j = 1,...,k, i = 1,...,l$, которая легко решается за время $P(l,k)$. Мы решим ее за время $P(l,\ln k)$, используя условие (6). Найдем величину k_l. Для этого определим функцию $U(c)$, $c = 1,...,k$, равную количеству таких пар (j,i), что $r(j,i) \leq r(c,l)$. Заметим, что при фиксированном i количество значений j таких, что $r(j,i) \leq B$ ($B = const$), определяется за время $C\ln k$, а значит, значение $U(c)$ вычисляется за время $Cl\ln k$. Поскольку $r(k_l,l)$ входит в множество k максимальных среди $r(j,i)$ чисел, то количество чисел, не меньших $r(k_l,l)$, не превышает k, т.е. $U(k_l) \leq k$. С другой стороны, $r(k_l+1,l)$ уже не входит в множество k максимальных среди

$r(j,i)$ чисел, поэтому $U(k_l+1)>k$. Таким образом, k_l можно найти, вычисляя $C\ln k$ раз функцию U. Вычисление одного значения функции U требует $Cl\ln k$ операций, поэтому общее число операций для нахождения числа k_l составляет $Cl\ln^2 k$, а для нахождения всех чисел $k_1,k_2,...,k_l$ требуется $Cl^2\ln^2 k$ операций. Лемма доказана.

Из леммы 3 следует выполнение условия (4) для функций $q_i(k_i)=M'_{опт}(k_i,m_i,n_i)$ и алгоритм решения поставленной в этом разделе задачи.

5. Приближенный алгоритм расстановки модулей

В предыдущих разделах предполагалось, что m модулей-буферов некоторым образом распределены по l цепочкам рабочих модулей, и были разработаны алгоритмы для распределения по этим цепочкам модулей контроля. В этом разделе мы рассмотрим общий случай сформулированной задачи: требуется распределить k модулей контроля и m модулей-буферов по l цепочкам (k_i модулей контроля и m_i модулей-буферов на i-ю цепочку, $\sum k_i=k$, $\sum m_i=m$) так, чтобы минимизировать величину $MA(k,m,n)=\sum_{i=1}^{l} MA'(k_i,m_i,n_i)$. Другими словами, надо решить оптимизационную задачу (1). Будем предполагать, что $k\ge m$. В этом разделе мы рассмотрим приближенный алгоритм решения этой задачи.

Лемма 4. Для случая одной цепочки при $k\ge m$

$$M'_{опт}(k,m,n)=n+\frac{a\cdot n^2}{2m}+\frac{a\cdot n^2}{2k}+\delta(k,m,n),$$

где $\delta(k,m,n)<0{,}03\left(\frac{a\cdot n^2}{2m}+\frac{a\cdot n^2}{2k}\right)$.

Из леммы 4 следует, что $\frac{M'_{опт}(k,m,n)-n}{\alpha} \approx \frac{n^2}{2m}+\frac{n^2}{2k}$. Поэтому вместо исходной мы будем рассматривать следующую задачу: найти такие целые неотрицательные числа $m_1,m_2,...,m_l$, $k_1,k_2,...,k_l$ ($\sum_{i=1}^{l} m_i = m$, $\sum_{i=1}^{l} k_i = k$), при которых величина $\sum_{i=1}^{l}\left(\frac{n_i^2}{2m_i}+\frac{n_i^2}{2k_i}\right)$ минимальна. Преимущество данного приближения в том, что можно отдельно решать задачи $\min_{m_i}\sum_{i=1}^{l}\frac{n_i^2}{2m_i}$ и $\min_{k_i}\sum_{i=1}^{l}\frac{n_i^2}{2k_i}$.

Рассмотрим набор функций $q_i(a)=\frac{n_i^2}{2a}$. Покажем, что для каждой из них выполнено условие (5). Это следует из приводимых ниже эквивалентных неравенств:

$$-q(a+1,b)+q(a,b) \le -q(a,b)+q(a-1,b);$$

$$-\frac{n_i^2}{2(a+1)}+\frac{n_i^2}{2a} \le -\frac{n_i^2}{2a}+\frac{n_i^2}{2(a-1)};$$

$$-\frac{1}{a+1}+\frac{1}{a} \le -\frac{1}{a}+\frac{1}{a-1};\ \frac{1}{a(a+1)} \le \frac{1}{a(a-1)}.$$

Применяя лемму 3 для функций $q_i(a)$ и учитывая ограничение $\sum_{i=1}^{l} m_i = m$, получаем числа $m_1,m_2,...,m_l$, а учитывая ограничение $\sum_{i=1}^{l} k_i = k$, получаем числа $k_1,k_2,...,k_l$. Из условия $m \le k$ и леммы 1 следует, что $m_i \le k_i$ при всех $i=1,...,l$. После этого для *i*-й цепочки ($i=1,2,...,l$) следует расставить m_i модулей-буферов и k_i модулей контроля согласно алгоритму для случая одной цепочки.

Оценим относительную погрешность данного алгоритма. В результате его работы мы получаем расстановку $A(k,m,n)$, для которой

$$MA(k,m,n) = \sum_{i=1}^{l} M_{опт}(k_i,m_i,n_i) = n +$$

$$+ \sum_{i=1}^{l}\left(\frac{a\cdot n_i^2}{2m_i} + \frac{a\cdot n_i^2}{2k_i} + \delta(k_i,m_i,n_i)\right).$$

Пусть в оптимальной расстановке в i-й цепочке назначено m_i^* модулей-буферов и k_i^* модулей контроля. Тогда

$$M_{опт}(k,m,n) = n + \sum_{i=1}^{l}\left(\frac{a\cdot n_i^2}{2m_i^*} + \frac{a\cdot n_i^2}{2k_i^*} + \delta(k_i^*,m_i^*,n_i)\right).$$

Из леммы 4 следует, что $\delta(k_i,m_i,n_i) < 0{,}03\left(\frac{a\cdot n_i^2}{2m_i} + \frac{a\cdot n_i^2}{2k_i}\right)$.

Суммируя, получаем $\sum_{i=1}^{l}\delta(k_i,m_i,n_i) < 0{,}03\sum_{i=1}^{l}\left(\frac{a\cdot n_i^2}{2m_i} + \frac{a\cdot n_i^2}{2k_i}\right)$.

Отсюда следует, что $\frac{M(k,m.n)-n}{\alpha} \le 1{,}03\sum_{i=1}^{l}\left(\frac{n_i^2}{2m_i} + \frac{n_i^2}{2k_i}\right)$. Но согласно выбору m_i и k_i имеем

$$\sum_{i=1}^{l}\left(\frac{n_i^2}{2m_i} + \frac{n_i^2}{2k_i}\right) \le \sum_{i=1}^{l}\left(\frac{n_i^2}{2m_i^*} + \frac{n_i^2}{2k_i^*}\right) \le$$

$$\le \sum_{i=1}^{l}\left(\frac{n_i^2}{2m_i^*} + \frac{n_i^2}{2k_i^*} + \delta(m_i^*,k_i^*,n_i)\right) = \frac{M_{опт}(k,m,n)-n}{\alpha}.$$

Следовательно, при применении описанного выше приближенного алгоритма математическое ожидание количества рабочих модулей, которые придется выполнить повторно, может увеличиться не более чем на 3%.

Оценка сложности приведенного алгоритма следует из леммы 3 и составляет $O(Cl^2 \ln^2 k)$.

Литература

1. *Гречук Б.В.* Алгоритмы планирования вычислений и организации рестартов в системах реального времени.// Дисс. на соискание ученой степени канд. физ.–матем. наук. М.: 2006.

1. *Фуругян М.Г.* Алгоритмы организации контроля в системах реального времени. // Некоторые алгоритмы планирования вычислений в многопроцессорных системах реального времени. М.: ВЦ РАН, 2010. С. 37 – 54.

Содержание