1975 год ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ Выпуск 63

УДК 681.142.2

СЕГМЕНТИРОВАНИЕ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

В.М.Гамидов, Н.Н.Миренков

Средства программирования для однородной вычислительной системы (ОВС), опирающиеся на методику крупноблочного распараллеливания [1-3], позволяют разрабатывать параллельные программы в виде совокупности идентичных параллельных ветвей (р-ветвей). При этом одновременное выполнение одинаковых р-ветвей не требует обязательного хранения соответствующей программы целиком в каждой машине. Обычно достаточно иметь одну программу, распределенную по системе, и предоставлять одновременно всем машинам только ту часть (сегмент), которая реализуется в данный момент.

Сегментирование р-ветви ведет, с одной стороны, к уменьшению объема оперативной памяти (ОП), необходимой для хранения параллельной программы, а с другой стороны,- к возникновению накладных расходов, связанных с синхронизацией машин [4] и загрузкой сегментов.

Задача сегментирования р-ветви аналогична задаче сегментирования последовательной программы [5] при наличии многоуровневой памяти. Здесь, однако, она имеет специфические особенности:

1) все сегменты во время реализации р-программы находятся в оперативной памяти ОВС;

2) объемы ОП, отводимые под сегменты в каждой машине, одинаковые;

3) загрузка очередного сегмента требует взаимодействия машин вычислительной системы.

П о с т а н о в к а з а д а ч и . Имеется р-ветвь параллельной программы объема V. Под разбиением Ω программы

р-ветви на сегменты понимается набор целых чисел $\{V_1, V_2, \dots, V_{k(\Omega)}\}$, где $V_i > 0$, $1 \le i \le K(\Omega)$ и $\sum_{i=1}^{k} V_i = V$ (V_i –объем i-го сегмента в разбиении Ω). Найти разбиение Ω, минимизирующее целевую функцию:

$$F_{(\Omega)} = \alpha \max_i \{V_i\} - \min_i \{V_i\}, \quad \alpha > 1, \tag{0.1}$$

и удовлетворяющее ограничениям

$$V_0 + \max_i \{V_i\} < \frac{1}{2} V, \tag{0.2}$$

где V_0 – объем диспетчера по загрузке сегментов;

$$t < (\varepsilon_1 + \varepsilon_2) T. \tag{0.3}$$

здесь T – время реализации программы; $t = t_1 + t_2$, t_1, t_2 – значения времени накладных расходов, связанных соответственно с загрузкой сегментов и синхронизацией машин при их выполнении; ε_1, ε_2 – константы, характеризующие доли допустимых расходов времени соответственно на загрузку и синхронизацию, $\varepsilon_1 + \varepsilon_2 < 1$.

Реализация задачи сводится к трем основным этапам:

1) эффективному разбиению на сегменты;

2) настройке их на специальное поле ОП;

3) распределению сегментов по машинам и диспетчированию их загрузкой.

§1. Разбиение на сегменты

Под эффективным разбиением на сегменты понимается разбиение, удовлетворяющее условиям (0.2) и (0.3). Точное решение поставленной задачи методами математического программирования вряд ли возможно в настоящее время. Поэтому предлагается алгоритм, ориентированный на оперативную человеко-машинную систему сегментации р-ветвей, записанных на языках высокого уровня.

1.1. О с н о в н ы е п о н я т и я и о п р е д е л е н и я . Известно, что любой последовательный алгоритм может быть представлен в виде направленного логического графа $G(W, U)$, где $W = \{\omega_i\}$ – множество вершин (операторов), $U = \{u_{ij} = (\omega_i, \omega_j) \mid \omega_i \in W, \omega_j \in W\}$ – множество направленных дуг. Мы будем иметь дело с логическими графами, на выходе предикатных вершин которых осуществляется логическая операция "Исключающее ИЛИ". Логика, ассоциированная с каждой предикатной дугой, используется для указания управляющих условий в программе.

Выполнение программы над заданным множеством входных данных может быть описано следующим образом.

Пусть ω_i – вершина с выходной логикой "Исключающее ИЛИ" и пусть $\omega_{i+1}, \omega_{i+2}, \ldots, \omega_{i+m}$ – её непосредственные преемники. Каждой предикатной дуге $u_{ij} = (\omega_i, \omega_j)$ ставится в соответствие вероятность q_{ij} достижения вершины ω_j в случае, если достигнута вершина ω_i; причем $\sum_{j=i+1}^{i+m} q_{ij} = 1$.

Оценки этих вероятностей могут быть получены непосредственно программистом или при многократном прогоне р-ветви для различных данных. В противном случае значения этих вероятностей будут выбраны сегментирующей программой самостоятельно. Не нарушая общности, можно считать, что индексация вершин в W обладает следующим свойством. Для ω_j –непосредственного преемника ω_i $j > i$, если дуга (ω_i, ω_j) не является дугой обратной связи; $j \neq i$ – в противном случае.

Введем обозначение: $C_{\alpha,\beta}\{W', U'\}$ – подграф, для которого $W' = \{\omega_i / \omega_i \in W, \alpha \leq i \leq \beta\}$, $U' = \{(\omega_i, \omega_j) / \omega_i \in W', \omega_j \in W'\}$.

Подграф $C_{\alpha,\beta}\{W', U'\}$ называется циклической (Ц–) структурой, ограниченной вершинами ω_α и ω_β, если каждая вершина в W' является сама себе предшественником и преемником.

Ц–структура $C_{\alpha,\beta}\{W', U'\}$ вложена в Ц–структуру $C_{\gamma,\delta}\{W'', U''\}$, если для ограничивающих вершин [6] выполняются условия: $(\alpha > \gamma, \beta \leq \delta)$ или $(\alpha \geq \gamma, \beta < \delta)$.

Ц–структура $C_{\alpha,\beta}\{W', U'\}$ зависима от Ц–структуры $C_{\gamma,\delta}\{W'', U''\}$, если для ограничивающих вершин выполняются условия: $(\alpha < \gamma < \beta < \delta)$ или $(\gamma < \alpha < \delta < \beta)$.

Ц–структуры $C_{\alpha,\beta}\{W', U'\}$ и $C_{\gamma,\delta}\{W'', U''\}$ не пересекаются, если $\beta < \gamma$ либо $\delta < \alpha$.

Ц–структура $C_{\alpha,\beta}\{W', U'\}$ называется простой, если не существует Ц–структуры $C_{\gamma,\delta}\{W'', U''\}$, вложенной в $C_{\alpha,\beta}\{W', U'\}$ или зависимой от нее.

В реальных программах, записанных на языках типа ФОРТРАН, БЭЙСИК, циклические структуры образуются с помощью операторов цикла (FOR ... NEXT), условного перехода (IF ... THEN), безусловного перехода (GOTO).

Ц–структуру, образованную оператором цикла, назовем циклом первого типа (ЦПТ); Ц–структуру, образованную операторами

условного и (или) безусловного переходов, – циклом второго типа (ЦВТ).

Циклы первого типа:

– могут содержаться один в другом, но не могут пересекаться;

– допускают преждевременные выходы с помощью операторов условного и безусловного переходов;

– разрешают вход в цикл только через оператор FOR .

Циклы второго типа:

– могут пересекаться с ЦВТ;

– допускают пересечение с ЦПТ, если дуга обратной связи ЦВТ подходит к вершине, являющейся предшественником всех вершин, содержащихся в этом ЦПТ.

I.2. Алгоритм сегментирования. Для алгоритма сегментирования необходима следующая информация: программа на входном языке, результат трансляции и таблица соответствия (ТС) операторов для исходной и объектной программ. ТС содержит номера операторов исходной программы и их начальные адреса в готовой программе. При реализации алгоритма предполагается диалог машины с человеком для консультаций относительно вероятностей переходов, коэффициентов повторения циклических структур и др. Чем больше информации об указанных величинах сообщит пользователь, тем эффективнее будет сегментирование.

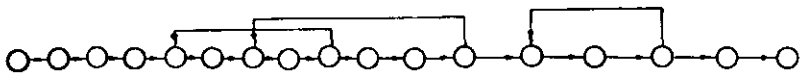
Путем просмотра "снизу вверх" программы, записанной на входном языке, накапливается в таблице Т данные о её логической и циклической структурах. Ниже дается пример, иллюстрирующий вышесказанное. Просмотр осуществляется с использованием методики R –таблиц [7].

Пример программы на ОВС–языке [8]

```
10  DIM N,I,N1
20  INPUT 1, A(N)
30  LET N1=N +1
40  DIM A(N), B(N), C(N), D(N), X(N1),G(N1)
50  INPUT 1, A(A,B,C,D)
60  LET G(1), X(1)=0
70  FOR I=2 TO N+1
```

```
 80  LET G(1)=B(I)+C(I)xG(I-1)
 85  IF G1=3 THEN 50
 90  LET G(I)=-A(I)/G(1)
100  LET X(I)=-(C(I)xX(I-1)+D(I)) / G(1)
110  NEXT I
120  FOR I=N+1 TO 3 STEP -1
130  LET X(I-1)=X(I) x G(I-1)+ X(I-1)
140  NEXT I
150  OUTPUT 5, A(X)
160  END
```

Граф – схема программы



Максимальные непересекающиеся Ц-структуры: I-я с 50-го по II0-й операторы, 2-я с I20-го по I40-й.

Таблица T:

FOR-NEXT	I20	I40
FOR-NEXT	70	II0
IF-THEN	50	85

Предполагается, что объем программы р-ветви не превосходит объем ОП машины (иначе сегментирование производится отдельно для каждого модуля загрузки).

Возможны два случая: I) программа не содержит Ц-структур, 2) программа содержит Ц-структуры.

П е р в ы й с л у ч а й. По таблице ТС определяем объем программы V. Набор операторов в сегмент производим после-довательно,используя объективную программу до тех пор, пока не выполнится условие (0.3), связанное с синхронизацией. Сог-ласно [4] и (0.2) объем сегмента в этом случае должен удовлет-ворять следующему соотношению:

$$\frac{V}{2} - V_0 > V_i \geq \frac{2(L-1)}{\varepsilon_2^2} \frac{\sum_{k=1}^{s} \cdot \tau_k \, \mathcal{D}\tau_k}{\left(\sum_{k=1}^{s} \tau_k \, M\tau_k\right)^2}, \qquad (I.1)$$

где L – число машин в ОВС; s – число операций, время выполне-ния которых зависит от операнд; τ_k – число операций k-го типа;

τ_k – случайная величина, характеризующая потери на синхрони-зацию при выполнении команд k-го типа; $M\tau_k$ – математическое ожидание τ_k ; $\mathcal{D}\tau_k$ – дисперсия τ_k (для каждой конкретной ма-шины величины $\mathcal{D}\tau_k$ и $M\tau_k$ могут быть заранее установлены экспе-риментально [4]).

После проверки для каждого сегмента условия (I.I) прове-ряется условие, связанное с расходами на загрузку. Выполнение его является достаточным для реализации сегментирования (про-цесс вычисления t и T описывается ниже при анализе программ с Ц-структурами).

В т о р о й с л у ч а й. По таблицам ТС и Т оп-ределяются максимальные непересекающиеся циклические структуры и их объемы. Эти Ц-структуры принимаются за первоначальные сег-менты (Ц-сегменты). Линейные участки не учитываются, поправка на них вводится позднее.

Для каждого Ц-сегмента проверяются условия (0.2) и усло-вие

$$t_n < (\varepsilon_1 + \varepsilon_2) T_n, \qquad (I.2)$$

где T_n – время реализации n-го Ц-сегмента; t_n – время на-кладных расходов, связанных с загрузкой n-го Ц-сегмента и син-хронизацией машин при его выполнении. Наиболее сложной являет-ся проверка второго условия.

Возможны четыре ситуации.

I. Ц-сегмент является п р о с т о й Ц - с т р у к т у-р о й. Алгоритм вычисления T_n для него имеет следующую опе-раторную схему:

$$\overbrace{A_1 \; P_1 \; A_2 \; A_3 \; A_4 \; A_5 \; P_2}^{\text{Я}} \; A_6 \; \text{Я}. \qquad (I.3)$$

где A_1 – вызывает очередной оператор Ц-структуры;

P_1 – передает управление (ПУ) на A_3, если вызванный опе-ратор не является предикатным;

A_2 – вычисляет область действия предикатного оператора;

A_3 – определяет предикатные операторы, в область действия которых попадает вызванный;

A_4 – вычисляет время работы ϑ_j вызванного j-го опера-тора, используя объектную программу и таблицы оценочного вре-мени счета подпрограмм,связанных с реализацией этого оператора;

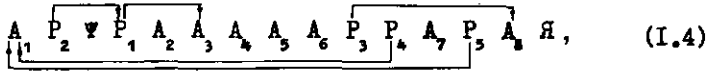
A_5 – вычисляет $\mathcal{D}_j = \sigma_j \prod_i q_i$, где q_i – вероятности соответствующих предикатных дуг, в области действия которых находится рассматриваемый оператор;

A_6 – вычисляет время счета Ц-структуры по формуле $N \sum_j \mathcal{D}_j$, где N – ожидаемое число повторений Ц-структуры;

P_2 – ПУ на A_1, если рассматриваемый оператор не последний в Ц-структуре;

Я – оператор выхода из Ц-структуры.

2. Ц-сегмент является **вложенной Ц-структурой** (ВЦ-структура – совокупность вложенных друг в друга, но независимых Ц-структур). Алгоритм вычисления T_n в этом случае имеет следующую операторную схему:



$$A_1\ P_2\ \Psi\ P_1\ A_2\ A_3\ A_4\ A_5\ A_6\ P_3\ P_4\ A_7\ P_5\ A_8\ Я, \qquad (I.4)$$

где A_1 – вызывает очередной оператор ВЦ-структуры;

P_2 – ПУ на P_1, если вызванный оператор не является началом некоторой Ц-структуры;

Ψ – определяет число k Ц-структур, начинающихся с вызванного оператора, и заносит k нулей в выталкивающий стек;

P_1, A_2, A_3, A_4, A_5, Я – аналогичны соответствующим операторам (I.3);

A_6 – увеличивает элемент, содержащийся в вершине стека, на $\mathcal{D}_j$;

P_3 – ПУ на A_8, если рассматриваемый оператор – последний в ВЦ-структуре;

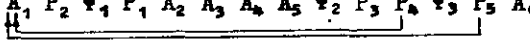
P_4 – ПУ на A_1, если рассматриваемый оператор не является последним для некоторой внутренней Ц-структуры;

A_7 – считывает содержимое C вершины стека и увеличивает элемент, содержащийся в новой вершине, на $C \cdot N$, где N – число повторений просмотренной Ц-структуры;

P_5 – ПУ безусловно на A_1;

A_8 – вычисляет $T_n = C \cdot N_1$, где N_1 – ожидаемое число повторений исходной ВЦ-структуры.

3. Ц-сегмент является **зависимой Ц-структурой**. (ЗЦ-структура – цепочка зависимых друг от друга Ц-структур). Алгоритм вычисления T_n для ЗЦ-структуры имеет следующую операторную схему:



$$A_1\ P_2\ \Psi_1\ P_1\ A_2\ A_3\ A_4\ A_5\ \Psi_2\ P_3\ P_4\ \Psi_3\ P_5\ A_6\ Я, \qquad (I.5)$$

где A_1 – вызывает очередной оператор ЗЦ-структуры.

P_2 – ПУ на P_1, если вызванный оператор не является началом некоторой Ц-структуры;

P_1, A_2, A_3, A_4, A_5, Я – аналогичны соответствующим операторам (I.3);

Ψ_1 – записывается нуль в стек (с проталкиванием);

Ψ_2 – увеличивает на $\mathcal{D}_j$ каждый элемент, содержащийся в стеке;

P_3 – ПУ на A_6, если рассматриваемый оператор – последний в ЗЦ-структуре;

P_4 – ПУ на A_1, если рассматриваемый оператор не является последним для некоторой Ц-структуры;

Ψ_3 – вычисляет $L_i = C \cdot N - C_H$, где C – содержимое вершины стека, C_H – новое содержимое вершины стека после считывания C; i – порядковый номер элемента в ЗЦ-структуре;

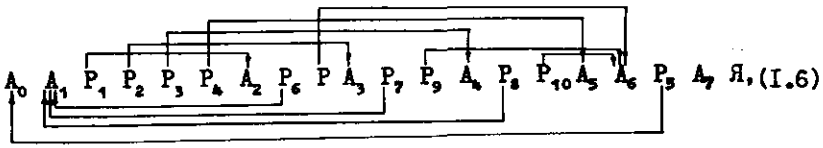
P_5 – ПУ безусловно на A_1;

A_6 – вычисляет $T_n = \sum_i L_i$,

4. Ц-сегмент является **смешанной Ц-структурой** (СЦ-структурой). Вложенно-зависимая Ц-структура (ВЗ-структура) – совокупность Ц-структур (среди которых могут быть как вложенные друг в друга, так и зависимые), вложенная в охватывающую их Ц-структуру. СЦ-структура – совокупность ВЗ-структур, охватывающие Ц-структуры которых образуют ЗЦ-структуру.

Вычисление T_n для СЦ-структуры производится в два этапа. На первом этапе вычисляется время реализации каждой ВЗ-структуры, на втором этапе – время СЦ-структуры, представляющее собой сумму времен ВЗ-структур без времени их перекрытия.

Алгоритм вычисления времени СЦ-структур может быть представлен операторной схемой:



A_0 – выбирает очередную ВЗ–структуру;

A_1 – определяет значение α ($\alpha := 1$, если очередная Ц–структура, вложенная в охватывающую Ц–структуру, является простой; $\alpha := 2$, если это ВЦ–структура; $\alpha := 3$, если – ЗЦ–структура; $\alpha := 4$, если – СЦ–структура);

P_1 – ПУ на A_{i+1}, если $\alpha = i$, $i = 1,2,3,4$;

A_j – реализует алгоритм $(1.j+1)$, $j = 2,3,4$;

A_5 – реализует алгоритм (I.6) для СЦ–структуры, вложенной в охватывающую Ц–структуру;

P_6, P_7, P_8 – ПУ на A_1, если не все Ц–структуры, вложенные в охватывающую, просмотрены;

P_9, P_{10}, P – ПУ безусловно на A_6;

A_6 – вычисляет время выбранной ВЗ–структуры;

P_5 – ПУ на A_0, если выбранная ВЗ–структура не является последней;

A_7 – реализует второй этап;

Я – оператор конца.

Вычисление t_n из (I.2) для Ц–сегментов любого из вышеперечисленных типов, производится одинаково:

$$t_n \leq V_n t_0 + \theta_n ,$$

где t_0 – время обмена одним кодом между машинами;

$$\theta_n = \sqrt{\frac{L-1}{\tau_n^2} \sum_{k=1}^{5} \tau_k D\tau_k} ,$$

где τ_k – число операций k–го вида в Ц–сегменте, с учетом развертки Ц–структур; V_n – объем сегмента; $\tau_n = \sum_{k=1}^{5} \tau_k$.

При выполнении условий (0.2) и (I.2) для всех Ц–сегментов дальнейшее разрезание реализуется только для тех, которые не являются простой Ц–структурой*). Разрезание производится на два сегмента по дугам, суммарное число повторений которых минимально. При нескольких возможностях разбиение выполняется там, где получаются сегменты с минимальной разностью $|V_i' - V_i''|$ своих объемов. Для полученных сегментов проверяется условие (I.2), выполнение которого позволяет повторить процедуру разрезания для сегментов, не содержащих простых Ц–структур. Невыполнение условия (I.2) на некотором шаге означает прекращение разрезания выбранного Ц–сегмента.

*) Это связано с большими накладными расходами на загрузку сегментов, не содержащих Ц–структур.

В результате реализации описанной процедуры (пр.I) над всеми Ц–сегментами получается некоторое множество сегментов, удовлетворяющих условиям (0.2) и (I.2) и не допускающих дальнейшего разрезания с соблюдением (I.2). Из полученного множества выбирается сегмент, имеющий наибольший объем, и разрезается на две части. Для нового множества сегментов проверяется условие (0.3), выполнение которого позволяет наибольший по объему сегмент в новом множестве разбить на две части (пр.2). В противном случае процесс разбиения Ц–структур заканчивается.

При невыполнении для некоторых Ц–сегментов условий (0.2) и (I.2) возможны три ситуации: один Ц–сегмент не удовлетворяет обоим условиям; один Ц–сегмент не удовлетворяет (0.2), но удовлетворяет (I.2); несколько Ц–сегментов удовлетворяют (0.2), но не удовлетворяют (I.2).

В первом случае выделенный Ц–сегмент разрезается на два сегмента и для полученного множества сегментов проверяется условие (0.3). При выполнении (0.3) для дальнейшего разбиения применяется, описанная ранее, процедура пр.2 разрезания наибольшего по объему сегмента. При невыполнении (0.3) сегментирование исходной р–ветви нецелесообразно.

Во втором случае выделенный Ц–сегмент разрезается на два сегмента и для каждого проверяется условие (I.2). При выполнении (I.2) для обоих сегментов и (0.2) и (I.2) для других Ц–структур выполняются процедуры пр.I и пр.2; иначе находимся в условиях третьей ситуации.

В третьем случае проверяется условие (0.3). При его выполнении для сегмента с максимальным объемом применяется процедура пр.2, описанная выше. При невыполнении (0.3) выбирается пара соседних сегментов, для которой суммарный объем меньше $\frac{V}{2} - V_0$ и является наименьшим среди всех пар, содержащих, по крайней

мере,один не удовлетворяющий условию (I.2) сегмент (при отсутствии таких пар сегментирование нерационально). Выбранная пара рассматривается как новый сегмент. Для полученного множества сегментов проверяется условие (0.3),выполнение которого позволяет считать полученный набор искомым (с точностью до выравнивания линейными участками). При невыполнении (0.3) - в сегмент объединяется другая пара, объем которой минимален среди оставшихся пар и которая удовлетворяет условию (0.2).При отсутствии пары,обеспечивающей выполнение условия (0.3),переходим к подобному объединению (троек, четверок и т.д.) сегментов и продолжаем до тех пор, пока для них выполняется условие (0.2).

Для получения окончательного набора сегментов анализируются линейные участки программы, чтобы присоединить их полностью или частично к имеющимся сегментам либо организовать новые самостоятельные сегменты.

Линейная (Л-)структура (или её часть) присоединяется к смежному сегменту, если I) размер объединенного сегмента не превосходит размера максимального сегмента, 2) в линейной части,предшествующей циклической, не существует операторов перехода за пределы объединенного сегмента, 3) в линейную часть, следующую за циклической, не существует передач управления из-за пределов объединенного сегмента. Л-структура (или её часть) не присоединяется к смежным сегментам, если нарушаются указанные условия или остается часть Л-структуры, размер которой меньше объема минимального сегмента. Из неприсоединенных Л-структур организуются самостоятельные сегменты, объемы которых не превосходят объем максимального сегмента и, при возможности, не меньше объема минимального сегмента.

§ 2. Настройка сегментов

Под настройкой сегментов понимается реализация для каждого из них абсолютного формата с базовым адресом, являющимся первой ячейкой буферного поля. При осуществлении настройки учитывается тот факт, что в итоге решения задачи сегментирования должна получиться распределенная по машинам программа, эквивалентная исходной программе. Это достигается

- выделением констант в отдельный сегмент и дублированием его во всех машинах,

- переработкой адресных частей команд, использующих эти константы;

- переработкой адресных частей команд, использующих адреса, принадлежащие другим сегментам исходной программы.

Для реализации этого достаточно одного просмотра объектной программы. При обнаружении указанной команды её адресная часть заменяется на абсолютный адрес,являющийся адресом команды, которая передает управление диспетчеру,осуществляющему загрузку сегментов.После команды передачи управления формируется таблица информации,содержащая номер сегмента,который будет продолжать вычисления,и абсолютный адрес команды в нем,на которую передается управление.Содержимое таких таблиц информации в дальнейшем будет использоваться диспетчером по загрузке сегментов.

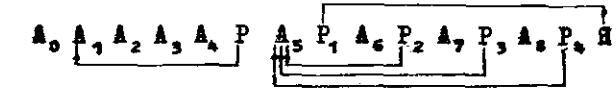
§3. Распределение сегментов по машинам и диспетчирование их загрузкой

Сегменты программы требуется распределить по машинам так, чтобы суммарный объем ОП, занимаемый ими в каждой машине, был одинаков.Другими словами, множество $\{V_i\}$ $(i=1,2,\dots,n)$ целых чисел распределить на L подмножеств так, чтобы минимизировать функцию

$$z = \max_j \Big\{ \sum_{i=1}^{n} x_{ij} V_i \Big\} - \min_j \Big\{ \sum_{i=1}^{n} x_{ij} V_i \Big\}, \qquad (3.1)$$

где $x_{ij}=1$,если элемент V_i принадлежит подмножеству j ,иначе $x_{ij}=0$.

Точное решение этой задачи методами целочисленного программирования является громоздким, поэтому предлагается эвристический алгоритм, ориентированный на практическое оперативное выполнение. Алгоритм может быть представлен следующей операторной схемой:



где A_0 – упорядочивает по убыванию множество $\{V_i\}$ и разбивает его на последовательные подмножества из L элементов в каж-

дом; если $\left[\frac{n}{L}\right] \neq \frac{n}{L}$, то к множеству $\{V_i\}$ добавляет $n-\left[\frac{n}{L}\right] L$ нулевых элементов; $c := \left]\frac{n}{L}\right[$;

A_1 – вычисляет для каждого j-го подмножества ($j = 1,\ldots, c$) значение

$$S_j = \sum_{k=L\cdot(j-1)+2}^{j\cdot L} (V_{j_0} - V_k), \qquad (3.2)$$

где $j_0 = L\cdot(j-1)+1$, $j = 1,2,\ldots,\omega/L$.

$$\omega = \begin{cases} n, & \text{если } \frac{n}{L} = \left[\frac{n}{L}\right], \\ L\cdot\left(\left[\frac{n}{L}\right]+1\right), & \text{если } \frac{n}{L} \neq \left[\frac{n}{L}\right], \end{cases}$$

и объединяет попарно элементы подмножества, у которого S_j максимально, с соответствующими элементами остальных подмножеств. Соответствие устанавливается по следующему правилу: элемент $L(j_1-1)+1+k$, входящий в подмножество с номером j_1, объединяется с элементом $j_2 L - k$, входящим в подмножество с номером j_2, $k = 0,1,\ldots, L-1$; $j_1 \neq j_2$;

A_2 – выбирает из подмножеств с объединенными элементами то, для которого функция (3.2) принимает минимальное значение; выбранное подмножество упорядочивает по убыванию;

A_3 – формирует множество из выбранного объединенного подмножества и из не попавших в последнее подмножеств множества, полученного оператором A_1;

A_4 – определяет число подмножеств c в сформированном множестве;

P – передает управление на A_1, если $c > 1$;

A_5 – определяет разность между максимальным и минимальным элементами последнего сформированного множества;

P_1 – передает управление на Я, если разность равна 0;

A_6 – находит элементы исходного множества в максимальном объединенном элементе, меньшие вычисленной разности, и минимальный из них переводит в минимальный объединенный элемент;

A_7 – определяет пару элементов исходного множества (первый принадлежит максимальному элементу выбранного объединенного множества, второй минимальному), разность между которыми меньше разности полученной при реализации A_5;

P_2 – ПУ на A_3, если A_6 находит искомый элемент;

P_3 – передает управление на A_5, если A_7 находит нужную пару элементов;

A_8 – воспринимает всевозможные комбинации из m исходных элементов ($m = 2,3\ldots$) за один элемент;

P_4 – передает управление на A_5, если $m \leq 3$;

Я – оператор конца.

Численные эксперименты показали, что практически процесс распределения сегментов по машинам можно остановить при m, равном 2 или 3.

Диспетчирование загрузкой сегментов производится следующим образом. Первоначально сегмент с номером I загружается в буферное поле каждой машины, ему передается управление и счет выполняется до тех пор, пока все машины не затребуют новых сегментов. На основе запросов выявляются машины, содержащие эти сегменты, и из выявленных машин производится поочередно пересылка затребованных сегментов. Обращение некоторых машин к диспетчеру по загрузке сегментов может совпадать с обращением других машин к системным операторам обменных взаимодействий. В этом случае в машинах с системными операторами реализуется передача управления диспетчеру по загрузке сегментов, после чего выполняется указанная выше пересылка сегментов и управление передается системным операторам. Диспетчер по загрузке сегментов представляет собой параллельную программу из идентичных р-ветвей, каждая из которых может быть включена в диспетчер элементарной машины системы или загружаться на время счета как стандартная программа.

Численные эксперименты показали, что разработанные алгоритмы не требуют больших затрат машинного времени, позволяют находить рациональное разбиение ветвей р-программ на сегменты, эффективно распределять их по машинам системы.

Выделение функций сегментирования в специальный программный модуль позволит включить их в любую операционную систему.

Литература

I. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности. Новосибирск, "Наука", 1966.

2. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. О решении задач на универсальных вычислительных системах. – В кн.: Вычислительные системы. Вып.17. Новосибирск, 1965, с. 106–164.

3. МИРЕНКОВ Н.Н. Параллельные алгоритмы для решения задач на однородных вычислительных системах. –В кн.: Вычислительные системы. Вып. 57. Новосибирск, 1973, с.3–32.

4. КОСАРЕВ Ю.Г., НАГАЕВ С.В. О потерях времени на синхронизацию в однородных вычислительных системах. –В кн.: Вычислительные системы. Вып. 24. Новосибирск, 1967, с. 21–39.

5. ПОСПЕЛОВ Д.А. Введение в теорию вычислительных систем. М., "Сов.радио", 1972.

6. BOVET D., ESTRIN G. A dynamic memory allocation algorithm. – "IEEE Trans.Comput.", 1970, vol.19, № 5, p.403–411.

7. ВЕЛЬБИЦКИЙ И.В., КЮНЕНКО Е.Л. Метаязык, ориентированный для синтаксического анализа и контроля. –"Кибернетика", Киев, 1970, № 2, с. 50–53.

8. ГРИШАКИНА Н.К., КЕРБЕЛЬ В.Г., КОЛОСОВА Ю.И., КОНСТАНТИНОВ В.И., КОРНЕЕВ В.Д., ЛЕВАГИНА Т.А., МИРЕНКОВ Н.Н., ФИШЕРМАН С.Б. Язык параллельных алгоритмов. –В кн.: Вычислительные системы. Вып. 57. Новосибирск, 1973, с. 33–54.

Поступила в ред.-изд.отд.
9 июня 1973 года