ВОПРОСЫ ТЕОРИИ ПОСТРОЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1977 год Выпуск 70

УДК 681.306

РЕШЕНИЕ ЗАДАЧИ БЛОЧНОГО ПРОГРАММИРОВАНИЯ НА СОСРЕДОТОЧЕННОЙ ОДНОРОДНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЕ

С.А.Тарноруцкая

В связи с разработкой однородных вычислительных систем (ОВС) [1] необходимо рассмотрение возможно более широкого круга задач для определения эффективности их решения. В данной работе предлагается параллельный алгоритм задачи блочного программирования [2] и оценивается его эффективность.

1. З а д а ч а б л о ч н о г о п р о г р а м м и р о в а н и я. Требуется найти максимум линейной формы

$$C^{(1)}X^{(1)}+C^{(2)}X^{(2)}+\ldots+C^{(s)}X^{(s)} \qquad (1)$$

при условиях:

$$A_1^{(0)}X^{(1)}+A_2^{(0)}X^{(2)}+\ldots+A_s^{(0)}X^{(s)}\le B^{(0)}, \qquad (2)$$

$$\left.\begin{aligned} A^{(1)}X^{(1)} &\le B^{(1)},\\ A^{(2)}X^{(2)} &\le B^{(2)},\\ &\cdots\\ A^{(s)}X^{(s)} &\le B^{(s)}, \end{aligned}\right\} \qquad (3)$$

$$X^{(t)}\ge 0,\quad t=1,2,\ldots,s. \qquad (4)$$

Здесь $C^{(t)}=(C_1^{(t)},C_2^{(t)},\ldots,C_{n_t}^{(t)})$ – n_t-мерная вектор-строка; $B^{(0)}=(b_1^{(0)},b_2^{(0)},\ldots,b_m^{(0)})$, $B^{(t)}=(b_1^{(t)},b_2^{(t)},\ldots,b_{m_t}^{(t)})$, $X^{(t)}=(x_1^{(t)},x_2^{(t)},\ldots,x_{n_t}^{(t)})$ – m-, m_t-, n_t-мерные вектор-столбцы соответственно;

$$A_t^{(0)}=\begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)}\ldots a_{1,n_t}^{(0)}\\ a_{21}^{(0)} & a_{22}^{(0)}\ldots a_{2,n_t}^{(0)}\\ \cdots & \cdots\\ a_{m1}^{(0)} & a_{m2}^{(0)}\ldots a_{mn_t}^{(0)}\end{bmatrix};\qquad A^{(t)}=\begin{bmatrix} a_{11}^{(t)} & a_{12}^{(t)}\ldots a_{1,n_t}^{(t)}\\ a_{21}^{(t)} & a_{22}^{(t)}\ldots a_{2n_t}^{(t)}\\ \cdots & \cdots\\ a_{m_t,1}^{(t)} & a_{m_t,2}^{(t)}\ldots a_{m_tn_t}^{(t)}\end{bmatrix}.$$

Метод разложения, применяемый в задаче (1)–(4), позволяет представить ее в виде локальных подзадач ($X_\lambda^{(t)}$-задач), соответствующих отдельным блокам, и главной задачи (Z-задачи), связывающей эти подзадачи.

Главная задача строится следующим образом. Предположим, что множества, определяемые условиями (3), ограничены и выпуклы. Тогда любое допустимое решение отдельного блока $X^{(t)}$ может быть представлено как линейная комбинация его базисных решений $X_\nu^{(t)}$. Здесь ν – номер базисного плана t-го блока ($\nu=1,2,\ldots,N_t$). Таким образом,

$$X^{(t)}=\sum_{\nu=1}^{N_t} z_\nu^{(t)}X_\nu^{(t)}, \qquad (5)$$

где $$\sum_{\nu=1}^{N_t} z_\nu^{(t)}=1,\qquad z_\nu^{(t)}\ge 0.$$

Введем новые обозначения:

$$\sigma_\nu^{(t)}=(C^{(t)},X_\nu^{(t)}),\quad P_\nu^{(t)}=A_t^{(0)}X_\nu^{(t)}. \qquad (6)$$

Теперь Z-задачу можно представить так:

$$\sum_{t=1}^{s}\sum_{\nu=1}^{N_t}\sigma_\nu^{(t)}z_\nu^{(t)}\to max, \qquad (7)$$

$$\sum_{t=1}^{s}\sum_{\nu=1}^{N_t}P_\nu^{(t)}z_\nu^{(t)}\le B^{(0)}, \qquad (8)$$

$$\sum_{\nu=1}^{N_t}z_\nu^{(t)}=1,\quad t=1,2,\ldots,s, \qquad (9)$$

$$z_\nu^{(t)}\ge 0;\quad \nu=1,2,\ldots,N_t,\ t=1,2,\ldots,s. \qquad (10)$$

Задача (7)–(10) эквивалентна исходной (1)–(4). Полученные решения Z-задачи $z_\nu^{(t)}$ позволяют по формулам (5) перейти к решениям исходной задачи $X^{(t)}$.

Для решения Z-задачи не обязательно знать все векторы $P_\nu^{(t)}$. На каждом шаге необходимо иметь только $m+s$ векторов условий Z-задачи, составляющих ее текущий базис. Что касается вектора, подлежащего включению в базис, то его выбор производится с помощью решения $X_\Lambda^{(t)}$-задач.

Перейдем к построению $X_\Lambda^{(t)}$-задач, для чего рассмотрим $(m+s)$-мерный вектор $\Lambda = (\Lambda^{(1)}, \Lambda^{(2)})$ оценок условий главной задачи относительно исследуемого опорного плана. Вектор $\Lambda^{(1)} = (\lambda_1^{(1)}, \lambda_2^{(1)}, \dots, \lambda_m^{(1)})$ состоит из первых m компонентов вектора Λ (из оценок условий (8)), а $\Lambda^{(2)} = (\lambda_1^{(2)}, \lambda_2^{(2)}, \dots, \lambda_s^{(2)})$ составлен из последующих s компонентов (из оценок условий (9)).

В $X_\Lambda^{(t)}$-задаче требуется вычислить максимум линейной формы $\mathcal{L}_\Lambda^{(t)}(X^{(t)}) = (C_\Lambda^{(t)}, X^{(t)}) = (C^{(t)} - \Lambda^{(1)} A_t^{(0)}, X^{(t)})$ при условиях $A^{(t)} X^{(t)} \le B^{(t)}$, $X^{(t)} \ge 0$.

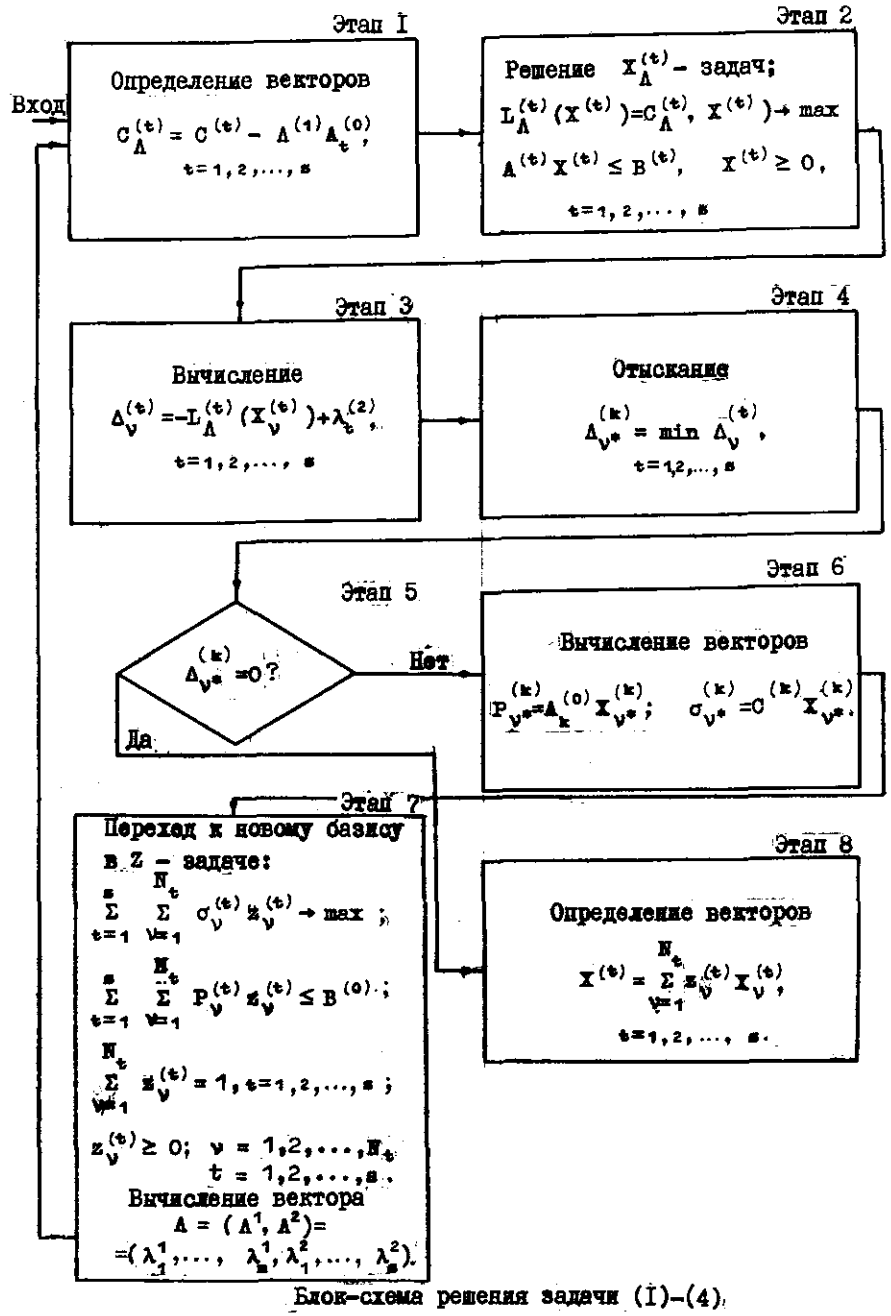
Оценка $\Delta_\nu^{(t)}$ вектора условий $P_\nu^{(t)}$ относительно базиса главной задачи вычисляется из соотношения:

$$\Delta_\nu^{(t)} = -\mathcal{L}_\Lambda^{(t)}(X_\nu^{(t)}) + \lambda_t^{(2)}.$$

Если $\min\limits_{t=1,2,\dots,s} \Delta_\nu^{(t)} = 0$, то исследуемый опорный план Z-задачи является ее решением. Если же это условие не соблюдается, то в очередной базис Z-задачи вводится вектор $P_{\nu^*}^{(k)}$, для которого $\Delta_{\nu^*}^{(k)} = \min\limits_{t=1,2,\dots,s} \Delta_\nu^{(t)}$. Компоненты вектора $P_{\nu^*}^{(k)}$ и соответствующий коэффициент линейной формы $\sigma_{\nu^*}^{(k)}$ вычисляются по формуле (6).

Блок-схему решения задачи (1)–(4) можно рассматривать как последовательность восьми этапов (см. рисунок). Нахождение исходного опорного плана в блок-схеме не отражено и в статье не рассматривается.

2. Параллельный алгоритм. При распараллеливании алгоритма блочного программирования на сосредоточенной ОВС из Q машин применяется принцип раздельной оптимизации [3], т.е. параллельный алгоритм строится для каждого из этапов в отдельности.



Блок-схема решения задачи (1)–(4).

Пусть $m_1, m_2, \ldots, m_Q$ – машины сосредоточенной ОВС и $Q < s < n$. Обозначим время счета машины m_k на этапе с номером ℓ через $t_k^{(\ell)}$ ($\ell = 1,2,\ldots,8$; $k = 1,2,\ldots,Q$), тогда "чистое" системное время счета на этапе ℓ есть $t_{cr}^{(\ell)} = \max\limits_k t_k^{(\ell)}$. Обозначим через $t_{об}^{(\ell)}$ время обменов информацией, не совмещаемых с вычислениями, между машинами на ℓ –м этапе, а через $t_{np}^{(\ell)}$ – время простоев.

Общее время работы ОВС на ℓ –м этапе определим как $T^{(\ell)} = t_{cr}^{(\ell)} + t_{об}^{(\ell)}$. На каждом этапе будем распараллеливать вычисления сначала для Q машин, где $1 < Q \le Q$, а затем находить значения Q, при которых $T^{(\ell)}$ ($\ell = 1,2,\ldots,8$) достигает минимума. Эффективность распараллеливания на каждом этапе будем оценивать величиной $\delta^{(\ell)} = \dfrac{t_{об}^{(\ell)} + t_{np}^{(\ell)}}{t_{cr}^{(\ell)}}$ (см. [4]).

Перейдем к распараллеливанию на каждом этапе.

ЭТАП I. Вычисление компонентов вектора C_λ, где $C_\lambda = (C_\lambda^{(1)}, C_\lambda^{(2)}, \ldots, C_\lambda^{(s)})$.

Реализуется формула $C_\lambda = C - \Lambda^{(1)} A^{(0)}$, где $C = (C^{(1)}, C^{(2)}, \ldots, C^{(s)})$, $A^{(0)} = (A_1^{(0)}, A_2^{(0)}, \ldots, A_s^{(0)})$.

В соответствии с методикой распараллеливания по циклам [5] в каждой машине вычисляется r_1 компонентов вектора C_λ, где

$$r_1 = \begin{cases} \left[\dfrac{n}{q}\right]+1 & \text{для } m_1, m_2, \ldots, m_\rho, \\ \left[\dfrac{n}{q}\right] & \text{для } m_{\rho+1}, m_{\rho+2}, \ldots, m_q; \end{cases}$$

$$n = \sum_{t=1}^{s} r_t$$

($0 \le \rho \le q-1$ – остаток от деления n на q; $\left[\dfrac{n}{q}\right]$ – наибольшее целое число, не превосходящее $\dfrac{n}{q}$).

К матрице $A^{(0)}$ применяется ВП-распределение [4], т.е. в каждую машину помещается r_1 столбцов матрицы $A^{(0)}$. Аналогично распределяются компоненты вектора C.

Вектор $\Lambda^{(1)}$ во время вычислений пересылается из m_q (см. этап 7) во все машины, т.е. осуществляется трансляционный обмен [4]. Очевидно, что для нахождения одной компоненты вектора C_λ необходимо произвести m операций умножения и m операций сложения. Обозначим через t_y и t_c время выполнения операций умножения и сложения. Нетрудно подсчитать, что

$$t_{cr}^{(1)} = m(t_y + t_c) r_1 \le m(t_y + t_c)\left(\left[\frac{n}{q}\right]+1\right).$$

Теперь вычислим величины $t_{np}^{(1)}$ и $t_{об}^{(1)}$. На данном этапе машины с номерами $\rho+1, \rho+2, \ldots, q$ будут простаивать в течение времени $t_{np}^{(1)} \le m(t_y + t_c)$ в связи с тем, что они вычисляют на один компонент вектора C_λ меньше, чем машины $m_1, m_2, \ldots, m_\rho$.

В рассматриваемой ОВС предполагается совмещение обмена между машинами с вычислениями. Компонент i вектора $\Lambda^{(1)}$ можно переслать из m_q во все машины во время вычисления $(i-1)$ –й частичной суммы, составляющей компонент вектора C_λ. Поэтому можно считать $t_{об}^{(1)} = 0$.

По мере вычисления компонентов C_λ машины должны обмениваться частью вычисленных компонентов. В первую очередь должны вычисляться те компоненты, которые подлежат пересылке. Вычисление одного компонента на машине занимает время, равное $m(t_y + t_c)$, а на пересылку q компонентов уходит время $q\,t_n$, где t_n – время пересылки компонента вектора из одной машины в другие. Так как рассматриваются задачи больших размерностей, для которых $q \le Q < m$, то на обмен вычисленными компонентами C_λ также не будет затрачено дополнительное время. Таким образом,

$$T^{(1)} = t_{cr}^{(1)} \le m(t_y + t_c)\left(\left[\frac{n}{q}\right]+1\right).$$

Так как $T^{(1)}$ является невозрастающей функцией от q и $q \le Q < n$, то распараллеливать вычисления этапа I следует на Q машинах. Коэффициент

$$\delta^{(1)} \le \frac{m(t_y + t_c)}{m(t_y + t_c)\left(\left[\frac{n}{q}\right]+1\right)} = \frac{1}{\left[\frac{n}{q}\right]+1},$$

т.е. для $\dfrac{n}{q} > 10$ "накладные" расходы от распараллеливания составляют незначительную величину.

ЭТАП 2. Решение s $X_\lambda^{(t)}$ – задач, на которое затрачивается основное время работы ОВС. Распараллеливание производится таким способом, чтобы каждая $X_\lambda^{(t)}$ – задача решалась на одной машине.

Так как для решения $X_\lambda^{(t)}$ – задачи на различных итерациях задачи (I)–(4) используется одна и та же матрица $A^{(t)}$ и один и тот же вектор ограничений $B^{(t)}$, то их целесообразно хранить в памя-

Таблица

Номера машин	Номера столбцов				
	1	2	...	$\left[\frac{s}{Q}\right]$	$\left[\frac{s}{Q}\right]+1$
1	1	2Q	...	$\left[\frac{s}{Q}\right]Q$	$\left[\frac{s}{Q}\right]Q+1$
2	2	2Q−1	...	$\left[\frac{s}{Q}\right]Q-1$	$\left[\frac{s}{Q}\right]Q+2$
⋮	⋮	⋮		⋮	⋮
j	j	2Q−j+1	...	$\left[\frac{s}{Q}\right]Q-j+1$	$\left[\frac{s}{Q}\right]Q+j=s$
⋮	⋮	⋮		⋮	
p	p	2Q−p+1	...	$\left[\frac{s}{Q}\right]Q-p+1$	
⋮	⋮	⋮		⋮	
Q	Q	Q+1	...	$\left(\left[\frac{s}{Q}\right]-1\right)Q+1$	

ти той машины, на которой решается соответствующая $X_\lambda^{(t)}$-задача. В результате операций обмена, произведенных на этапе I, вектор $C_\lambda^{(t)}$ находится в памяти этой же машины.

Для проведения одной итерации в $X_\lambda^{(t)}$-задаче требуется совершить $2m_t^2+m_t n_t$ операций умножения, $2m_t$ операций деления, $m_t^2+m_t n_t$ операций сложения и $m_t^2+m_t+n_t-2$ операций вычитания. Количество итераций, производимых при решении одной $X_\lambda^{(t)}$-задачи, не превосходит величины $3m_t$. Следовательно, время решения одной $X_\lambda^{(t)}$-задачи на одной машине

$$T_t \le 3m_t\left\{(2m_t^2+m_t n_t)t_y+2m_t t_g+\right.$$

$$\left.+(m_t^2+m_t n_t)t_c+(m_t^2+m_t+n_t-2)t_B\right\},$$

где t_g и t_B – время выполнения операций деления и вычитания.

При параллельном решении s $X_\lambda^{(t)}$-задач q машинами, каждой машине m_j $(1\le j\le q)$ достанется N_j задач; $\sum_{j=1}^{q} N_j = s$. Распределение этих задач по машинам производится таким образом, чтобы минимизировать величину:

$$\max_j \sum_{i=\alpha_j+1}^{\alpha_j+N_j} 3m_{t_i}\left\{(2m_{t_i}^2+m_{t_i}n_{t_i})t_y+2m_{t_i}t_g+\right.$$

$$\left.+(m_{t_i}^2+m_{t_i}n_{t_i})t_c+(m_{t_i}^2+m_{t_i}+n_{t_i}-2)t_B\right\}, \qquad (II)$$

где

$$\alpha_j=\sum_{k=1}^{j-1} N_k .$$

Нетрудно показать, что выражение (II) есть невозрастающая функция от q. Следовательно, на этапе 2 рационально распараллеливание вычислений на Q ветвей.

Точное определение максимума для (II) является громоздкой задачей, поэтому предлагается приближенный алгоритм. Упорядочим номера задач по величинам T_t. Не нарушая общности рассуждений, можно считать, что $T_1 \ge T_2 \ge \dots \ge T_s$. Произведем распределение задач по машинам в соответствии с вышеприведенной таблицей. (Предполагается, что при $\frac{s}{Q} \ne \left[\frac{s}{Q}\right]$ вводятся фиктивные задачи с $T_t = 0$; $t = s+1, s+2, \dots, \eta$; $\frac{s+\eta}{Q} = \left[\frac{s+\eta}{Q}\right]$.) Решение $X_\lambda^{(t)}$-задач, номера которых находятся в одной строке, осуществляется на одной машине. Время работы машины m_j составит величину

$$\tau_j^{(2)} = T_j + T_{2Q-j+1} + T_{2Q+j} + \dots + T_\nu,$$

где $\nu=\left[\frac{s}{Q}\right]Q+j$, если $\left[\frac{s}{Q}\right]$ – четная величина, иначе $\nu=\left(\left[\frac{s}{Q}\right]+1\right)Q-j+1$, а время работы ОВС на этапе 2 будет равно $T^{(2)}=\max_j \tau_j$. При этом $t_{об}^{(2)} = 0$, так как обмена информацией на этом этапе нет.

Теперь оценим потерю времени на простой машин из-за их неравномерной загрузки:

$$t_{np} \le (T_1-T_Q)+(T_{Q+1}-T_{2Q})+\dots+\left(T_{\left(\left[\frac{s}{Q}\right]\right)Q+1}-T_{\left[\frac{s}{Q}\right]Q}\right)+T_{\left[\frac{s}{Q}\right]Q+1} \le$$

$$\le (T_1-T_{Q+1})+(T_{Q+1}-T_{2Q+1})+\dots+\left(T_{\left(\left[\frac{s}{Q}\right]-1\right)Q+1}-T_{\left[\frac{s}{Q}\right]Q+1}\right)+T_{\left[\frac{s}{Q}\right]Q+1}=T_1.$$

Следовательно, время простоев не превышает времени решения самой длинной задачи.

Величины $T^{(2)}$ и $t_{np}^{(2)}$ можно значительно уменьшить, если при заполнении каждого столбца таблицы учитывать значения частичных сумм в (I2), т.е. к максимальной частичной сумме добавлять минимальное из оставшихся T_t.

ЭТАП 3. Вычисление по формуле $\Delta_y^{(t)}=-\mathcal{L}_\lambda^{(t)}(X_y^{(t)})+\lambda_t^{(2)}$. После реализации вычислений на этапе 2 в каждой машине находится значений $\mathcal{L}_\lambda^{(t)}(X_y^{(t)})$, где

$$r_2 = \begin{cases} \left[\frac{s}{Q}\right]+1 & \text{для } m_1, m_2, \ldots, m_\ell, \\ \left[\frac{s}{Q}\right] & \text{для } m_{\ell+1}, m_{\ell+2}, \ldots, m_Q \end{cases}$$

($0 \le \ell \le Q$ – остаток от деления s на Q).

Все s компонентов вектора $\Lambda^{(2)}$ вычисляются в m_Q на этапе 7. Так как обмена информацией на 2-м этапе нет, то во время этапа 2 можно переслать из m_Q в каждую машину по r_2 соответствующих компонентов вектора $\Lambda^{(2)}$.

Очевидно, что $st_n << T^{(2)}$. Поэтому дополнительного времени на пересылку компонентов вектора $\Lambda^{(2)}$ не потребуется. Тогда получаем

$$T^{(3)} \le \left(\left[\frac{s}{Q}\right]+1\right)t_б, \quad t_{пр}^{(3)} \le t_б, \quad t_{об}^{(3)} = 0,$$

$$\delta^{(3)} \le \frac{t_б}{\left(\left[\frac{s}{Q}\right]+1\right)t_б} = \frac{1}{\left[\frac{s}{Q}\right]+1}.$$

ЭТАП 4. Определение $\Delta_{y*}^{(k)} = \min\limits_{t=1,2,\ldots,s} \Delta_y^{(t)}$. В каждой машине перед началом этапа 4 находится $\left[\frac{s}{Q}\right]+1$ или $\left[\frac{s}{Q}\right]$ значений $\Delta_y^{(t)}$. Выполняется следующий алгоритм нахождения: $\min\limits_{t=1,2,\ldots,s} \Delta_y^{(t)}$. Сначала производится $\left[\frac{s}{Q}\right]-1$ шагов вычислений, причем на каждом шаге выполняется по Q операций сравнения.

На $\left[\frac{s}{Q}\right]$-м шаге выполняется по одному сравнению на $m_1, m_2, \ldots, m_\ell$, а полученное минимальное значение в машине с номером $\ell+1$ пересылается в m_Q. Далее в машине с номером Q находится минимум из двух имеющихся там чисел, и параллельно с этим пересылается минимальное значение, полученное в $m_{\ell+2}$, затем в $m_{\ell+3}$ и т.д. В результате в машине с номером Q получается минимальное значение.

Так как для ОВС $t_п < t_б$, то за время сравнения двух кодов можно переслать следующий код в m_Q. Обозначая через $t_{уп}$ и $t_з$ время выполнения операций условного перехода и засылки в ячейку одного кода, получаем следующие формулы:

$$t_{ст}^{(4)} = t_Q^{(4)} = \left(\left[\frac{s}{Q}\right]-1\right)(t_б + t_{уп} + t_з) + (Q-1)(t_б + t_{уп} + t_з), \quad t_{об}^{(4)} = t_п.$$

$$t_{пр}^{(4)} = (Q-1)(t_б + t_{уп} + t_з),$$

$$T^{(4)} = \left(\left[\frac{s}{Q}\right]+Q-2\right)(t_б + t_{уп} + t_з) + t_п,$$

$$\delta^{(4)} = \frac{(Q-1)(t_б + t_{уп} + t_з) + t_п}{\left(\left[\frac{s}{Q}\right]+Q-2\right)(t_б + t_{уп} + t_з)} \approx \frac{Q-1}{\left[\frac{s}{Q}\right]+Q-2}.$$

Распараллеливание на этом этапе можно провести другим способом. Сначала, так же, как и в предыдущем способе, найти минимальные значения $\Delta_y^{(t)}$ в каждой машине, а дальше параллельно выполнять сравнения [4]. Для этого система разбивается на подсистемы по 2 машины. Машины с четными номерами передают по одному коду предшествующим машинам с нечетными номерами, которые параллельно находят минимум из двух чисел в каждой подсистеме. Затем система разбивается на подсистемы по 4 машины и определяются минимальные значения уже в более крупных подсистемах и т.д., пока не будет найдено $\Delta_y^{(k)}$.

Распараллеливание на данном этапе для $q < Q$ машин не рассматривается, так как это усложняет алгоритм из-за дополнительных операций пересылок и настроек системы между этапами 3 и 4.

ЭТАП 5. Сравнение $\Delta_{y*}^{(k)}$ с нулем. Пусть работает вычислительная машина, в памяти которой хранится $\Delta_{y*}^{(k)}$. Тогда $T^{(5)} = t_б + t_{уп} + t_з$, $t_{пр}^{(5)} = t_б + t_{уп} + t_з$.

ЭТАП 6. Вычисление по формулам: $P_{y*}^{(k)} = A_k^{(0)} X_{y*}^{(k)}$, $\sigma_{y*}^{(k)} = (C^{(k)}, X_{y*}^{(k)})$.

Здесь производится умножение матрицы A_k размерности $(m+1) \times n_k$ на вектор-столбец $X_{y*}^{(k)}$ длины n_k, где

$$A_k = \begin{bmatrix} A_k^{(0)} \\ C^{(k)} \end{bmatrix}.$$

К матрице A_k применяется ГП-распределение [4], помещением в каждую машину r_3 строк матрицы, где

$$r_3 = \begin{cases} \left[\frac{m+1}{q}\right]+1 & \text{для } m_1, m_2, \ldots, m_q, \\ \left[\frac{m+1}{q}\right] & \text{для } m_{q+1}, m_{q+2}, \ldots, m_q \end{cases}$$

($0 \le q \le q-1$ – остаток от деления $m+1$ на q).

В каждой машине вычисляется r_3 компонентов вектора ($P_{y*}^{(k)}$, $\sigma_{y*}^{(k)}$). Вектор $X_{y*}^{(k)}$ перед началом этапа 6 находится в оперативной памяти машины, имеющей $\Delta_{y*}^{(k)}$. Из этой машины его необходимо переслать во все машины сосредоточенной ОВС.

Повторив рассуждения, приведенные на этапе I, можно принять, что $t_{ob}^{(6)} = 0$. Далее вычисляется

$$T^{(6)} \le \left\{ n_k t_y + (n_k - 1) t_c \right\} \left(\left[\frac{m+1}{q}\right] + 1 \right),$$

$$t_{np}^{(6)} \le n_k t_y + (n_k - 1) t_c.$$

На данном этапе так же, как и на этапе I, следует положить $= Q$. Коэффициент

$$\delta^{(6)} \le \frac{1}{\left[\frac{m+1}{Q}\right]+1}.$$

Так как рассматриваются задачи большой размерности, то $\delta^{(6)} \ll 1$.

ЭТАП 7. Переход к новому базису в Z-задаче; отыскание базисных переменных и вычисление вектора оценок $\Lambda = (\Lambda^{(1)}, \Lambda^{(2)})$. Матрицу, обратную к матрице текущего базиса Z-задачи, обозначим через $E_Z = \{e_{ij}\}$, $i,j = 1,2,\ldots,m+s$, а вектор-столбец из значений базисных переменных – через $e_0 = (e_{10}, e_{20}, \ldots, e_{m+s,0})$. Вычисления на этапе 7 разбиваются на ряд последовательных шагов.

Шаг I. Разложение вектора $\bar{P}_{y*}^{(k)}$ по векторам текущего базиса Z – задачи по формуле $U = E_Z \cdot \bar{P}_{y*}^{(k)}$. Здесь $U = (u_1, u_2, \ldots, u_{m+s})$, $\bar{P}^{(k)} = (\bar{P}_{y*}^{(k)}, e_k)$, где e_k – s-мерный единичный вектор с единицей на k-м месте.

Вектор $\bar{P}_{y*}^{(k)}$ должен находиться на шаге I каждой машине. Для этого все машины должны обменяться компонентами $P_{y*}^{(k)}$,

сленными на этапе 6. Векторы e_k ($k = 1,2,\ldots,s$) во избежание пересылок должны храниться в каждой машине.

Таким образом, в каждой машине перед началом вычислений шага I находится $r_3 + s$ компонентов вектора $\bar{P}_{y*}^{(k)}$. От других машин она должна принять $m - r_3$ компонентов вектора $P_{y*}^{(k)}$.

Матрица E_Z записывается по r_4 строк в каждой машине, где

$$r_4 = \begin{cases} \left[\frac{m+s}{q}\right]+1 & \text{для } m_1, m_2, \ldots, m_d, \\ \left[\frac{m+s}{q}\right] & \text{для } m_{d+1}, m_{d+2}, \ldots, m_q \end{cases}$$

($0 \le d \le q-1$ – остаток от деления $m+s$ на q).

В каждой машине сначала производится умножение одной строки матрицы E_Z на те компоненты $\bar{P}_{y*}^{(k)}$, которые находятся в ее памяти. Это займет время

$$t = (r_3 + s) t_y + (r_3 + s - 1) t_c \le \left(\left[\frac{m+1}{q}\right] + 1 + s \right) t_y + \left(\left[\frac{m+1}{q}\right] + s \right) t_c \approx \left(\left[\frac{m+1}{q}\right] + 1 + s \right) \left(t_y + t_c \right).$$

В это же время машины должны обмениваться частью компонентов $P_{y*}^{(k)}$. Время обмена составит величину $m t_n$. Так как $t_n < t_y + t_c$, то для больших m, s будет выполняться неравенство

$$m t_n \le \left(\left[\frac{m+1}{q}\right] + 1 + s \right) \left(t_y + t_c \right)$$

и дополнительное время на обмен не потребуется.

Следовательно,

$$T^{7(1)} \le \left\{ (m+s) t_y + (m+s-1) t_c \right\} \left(\left[\frac{m+s}{q}\right] + 1 \right),$$

$$t_{np}^{7(1)} \le (m+s) t_y + (m+s-1) t_c, \quad \delta^{7(1)} \le \frac{1}{\left[\frac{m+s}{q}\right]+1}.$$

На данном шаге следует положить $q = Q$.

Шаг 2. Вычисление по формуле

$$h_{ik} = \frac{e_{i0}}{u_i}, \quad i = 1,2,\ldots, m+s.$$

Для распараллеливания на этом шаге в каждой машине должно находиться по r_4 компонентов векторов e_0 и U. Обмен информацией отсутствует. Тогда

$$T^{7(2)} \le \left(\left[\frac{m+s}{Q} \right] + 1 \right) t_q,$$

где t_q – время выполнения операции деления, причем

$$t_{np}^{7(2)} \le t_q, \quad \delta^{7(2)} \le \frac{1}{\left[\frac{m+s}{Q} \right] + 1}.$$

Ш а г 3. Отыскание $h_{lk} = \min\limits_{i=1,2,\dots,m+s} h_{ik}$.

Распараллеливание на этом шаге подобно распараллеливанию на этапе 4. В результате получаем

$$T^{7(3)} \le \left(\left[\frac{m+s}{Q} \right] - 1 \right) t_g + (Q-1) t_g + t_n,$$

$$t_{np}^{7(3)} = (Q-1)(t_g + t_{yn} + t_g).$$

Ш а г 4. Определим матрицу $\hat{E}_x = (\hat{e}_{ij})$ следующим образом:

$$\hat{E}_x = \begin{vmatrix} e_0 & E_x \\ \chi_x & \Lambda \end{vmatrix}.$$

Строки матрицы $\hat{E}_x$ будем обозначать через $\hat{E}_{x,i}$, $i=1,2,\dots,m+s+1$. Разместим $\hat{E}_{x,\,m+s+1}$ в машине m_q.

На шаге 4 необходимы преобразования матрицы $\hat{E}_x$ по формулам:

$$\hat{E}'_{x,l} = \frac{\hat{E}_{x,l}}{h_{l,k}}, \quad \hat{E}'_{x,i} = \hat{E}_{x,l} - u_i \hat{E}_{x,l}, \quad i=1,2,\dots,m+s+1, \quad i \ne l,$$

где $\hat{E}'_{x,i}$ $(i=1,2,\dots,m+s+1)$ – преобразованные строки матрицы $\hat{E}_x$.

Предположим, что $\hat{E}_{x,i}$ находится в машине m_t $(1 \le t \le q)$. . Все компоненты $\hat{E}_{x,l}$ будем вычислять в m_t. Распараллеливание прове – дем следующим способом. Поскольку $e'_{l,0}$ уже вычислено на шаге 2, то это значение в начале шага 4 надо переслать во все машины. Затем параллельно во всех машинах вычислить новые значения базисных переменных. На их вычисление в $m_1, m_2, \dots, m_{t-1}, m_{t+1}, \dots, m_q$ уйдет время $\left(\left[\frac{m+s}{q} \right] + 1 \right)(t_y + t_g)$. За это же время m_t успеет вычислить и переслать во все машины $e'_{l,1}$. Первый столбец $\hat{E}_x$ преобразовывается так же, как e_0, и т.д. Нетрудно подсчитать, что

$$T^{7(4)} \le \left(\left[\frac{m+s}{q} \right] + 1 \right)(m+s+1)(t_y + t_g),$$

$$t_{ob}^{7(4)} = t_n, \quad t_{np}^{7(4)} \le (m+s+1)(t_y + t_g).$$

Очевидно, что на шаге 4 следует положить $q = Q$.

$$\delta^{7(4)} \le \frac{t_n + (m+s+1)(t_y + t_g)}{\left(\left[\frac{m+s}{Q} \right] + 1 \right)(m+s+1)(t_y + t_g)} \approx \frac{1}{\left[\frac{m+s}{Q} \right] + 1}.$$

ЭТАП 8. Определение вектора $X^{(t)}$ по формулам:

$$X^{(t)} = \sum_{y=1}^{N_t} z_y^{(t)} X_y^{(t)}, \quad t=1,2,\dots,s; \quad \sum_{t=1}^{s} N_t = m+s.$$

Вычисления на этапе 8 происходят всего I раз. Умножение вектора $X_y^{(t)}$ на $z_y^{(t)}$ производится в той машине, в памяти которой находится $X_y^{(t)}$. Между машинами происходит обмен значениями $z_y^{(t)}$ для того, чтобы они находились в той же машине, что и $X_y^{(t)}$.

Для нахождения n_t компонентов вектора $X^{(t)}$ необходимо $n_t N_t$ операций умножения и $n_t (N_t - 1)$ операций сложения. Следовательно,

$$t_k^{(8)} = \sum_{t \in M_k} \left\{ n_t N_t t_y + n_t (N_t - 1) t_c \right\},$$

где M_k – множество номеров $X_\lambda^{(t)}$–задач, для которых $X_y^{(t)}$ находятся в машине с номером k $(k=1,2,\dots,Q)$; $t_{cr}^{(8)} = \max t_k^{(8)}$.

Время обмена не превысит величины $Q\, t_n$ – времени пересылки Q значений $z_y^{(t)}$ в соответствующие машины. Пересылка остальных $m+s-Q$ значений $z_y^{(t)}$ совмещается с вычислениями в машинах; $t_{np}^{(8)} = t_{cr}^{(8)} - t_m^{(8)}$, где m – номер машины, которая первой заканчивает вычисления на этапе 8.

Величины $T^{(8)}$ и $t_{np}^{(8)}$ достигнут максимального значения, если все вычисления будут производиться на одной машине. Итак,

$$T^{(8)} \le Q\, t_n + \sum_{t=1}^{s} \left\{ n_t N_t t_y + n_t (N_t - 1) t_c \right\}.$$

Так как вычисления на этапе 8 происходят всего один раз, то $T^{(8)}$ и $t_{np}^{(8)}$ – незначительные величины по сравнению со временем решения всей задачи.

Из формул, выведенных на этапах I–8, следует, что предлагаемый параллельный алгоритм ускоряет счет примерно в Q раз только благодаря увеличению в Q раз количества процессоров. Кроме того, вся информация или большая часть ее будет находиться в оперативной памяти машин, что приведет к уменьшению или даже сведет к нулю время обращения к вспомогательной памяти. За счет этого выигрыш во времени при решении задачи блочного программирования может превысить величину Q.

Литература

I. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности. Новосибирск, "Наука", I966.

2. ГОЛЬДШТЕЙН Е.Г., ЮДИН Д.Б. Новые направления в линейном программировании. М., "Сов.радио", I966.

3. ГОЛУБЕВ-НОВОЖИЛОВ Ю.С. Многомашинные комплексы вычислительных средств. М., "Сов.радио", I967.

4. МИРЕНКОВ Н.Н. Параллельные алгоритмы для решения задач на однородных вычислительных системах. –В кн.: Вычислительные системы. Вып. 57. Новосибирск, I973, с. 3–32.

5. КОСАРЕВ Ю.Г. Распараллеливание по циклам. –В кн.: Вычислительные системы. Вып. 24. Новосибирск, I967, с. 3–I9.

Поступила в ред.-изд.отд.
23 апреля I974 года