УДК 681.142.2

АЛГОРИТМЫ ФУНКЦИОНИРОВАНИЯ ОДНОРОДНЫХ УНИВЕРСАЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ В ПРОСТЕЙШИХ СИТУАЦИЯХ

Т.М. Голоскокова, В.Г. Хорошевский

Рассматриваются два алгоритма распределения конечного множества простых задач по элементарным машинам однородной универсальной вычислительной системы. Реализация алгоритмов, в отличие от алгоритмов математического программирования, не связана с большими вычислительными трудностями. Приводятся результаты статистической обработки экспериментов по моделированию алгоритмов. В приложении имеются программы алгоритмов, записанные на языке АЛГОЛ.

Постановка задачи

Имеется множество $\mathcal{J}=\bigcup_{j=1}^{n} J_j$ элементарных машин (ЭМ), образующих однородную универсальную вычислительную систему (УВС)[I], и множество $\mathcal{I}=\bigcup_{i=1}^{l} I_i$ независимых задач [2]. Для решения любой $I_i\in\mathcal{I}$ требуется одна ЭМ системы. Известно время решения t_i задачи $I_i\in\mathcal{I}$ и величина штрафа c_i за задержку решения I_i на единицу времени.

Требуется построить алгоритмы распределения задач по ЭМ системы, при помощи которых можно было бы получать оптимальные значения целевой функции, характеризующей эффективность УВС.

Алгоритм I

Допустим, что штраф за задержку решения любой задачи $I_i \in \mathcal{I}$ равен нулю. Тогда целевой функцией может быть время T решения задач множеств $\mathcal{I}$ на однородной УВС. Алгоритм I позволяет получить субминимальное значение T.

Опишем основные операции алгоритма.

Пусть $\hat{\mathcal{I}} = \{I_{i_1}, I_{i_2}, \ldots, I_{i_\ell}, \ldots, I_{i_L}\}$ – некоторая последовательность, членами которой являются элементы $I_i \in \mathcal{I}$. Подмножество $\mathcal{I}_j \subset \hat{\mathcal{I}}$ включает в себя κ_j задач, причем

$$\mathcal{I}_j = \bigcup_{\ell=K_j+1}^{K_j+\kappa_j} I_{i_\ell},$$

где

$$K_j = \sum_{z=0}^{j-1} \kappa_z, \quad j = 1,2,\ldots,n, \quad \kappa_0 = 0.$$

Если каждое множество $\mathcal{I}_j$, $j = 1,2,\ldots,n$, назначить для решения на машину $J_j \in \mathcal{J}$ $(\mathcal{I}_j \Rightarrow J_j \in \mathcal{J})$, то значение времени T будет равно

$$\hat{T} = \max_{\mathcal{I}_j \subset \hat{\mathcal{I}}} \{T_j\}, \quad T_j = \sum_{\ell=K_j+1}^{K_j+\kappa_j} t_{i_\ell}$$

Очевидно, что нижней гранью времени T будет

$$\theta = \frac{1}{n}\sum_{i=1}^{L} t_i = \frac{1}{n}\sum_{j=1}^{n} T_j .$$

Опишем правило выбора подмножеств $\mathcal{I}_j \subset \hat{\mathcal{I}}$.

Пусть построены подмножества $\mathcal{I}_z \subset \hat{\mathcal{I}}$, $z = 1,2,\ldots,j-1$. Тогда подмножество

$$\mathcal{I}_j := \begin{cases} \hat{\mathcal{I}}_j', \ \kappa_j := \kappa_j', & \text{если} \quad (\theta_\text{в})_j > \dfrac{T^j - (\theta_\text{н})_j}{N-j}, \\ \hat{\mathcal{I}}_j' \cup I_{K_j+\kappa_j'+1}, & \kappa_j := \kappa_j'+1, \ \text{в противном случае,} \end{cases}$$

где $\hat{\mathcal{I}}_j'$ – часть последовательности $\hat{\mathcal{I}}$ такая, что

$$\hat{\mathcal{I}}_j' = \{I_{K_j+1}, I_{K_j+2}, \ldots, I_{K_j+\kappa_j'}\},$$

$$T^j = \sum_{\ell=K_j+1}^{L} t_{i_\ell},$$

а для величин $(\theta_\text{в})_j$ и $(\theta_\text{н})_j$ справедливо:

$$\sum_{i=K_j+1}^{K_j+\kappa_j'+1} t_i = (\theta_\text{в})_j > \theta, \quad \sum_{i=K_j+1}^{K_j+\kappa_j'} t_i = (\theta_\text{н})_j \leq \theta .$$

Требуется выбрать такую последовательность $\hat{\mathcal{I}}^*$ задач, при которой время T принимает субминимальное значение.

В основу алгоритма отыскания последовательности $\hat{\mathcal{I}}^*$ положен метод цепей, который более эффективен, чем метод Монте-Карло [3]. Рассмотрим схему метода применительно к проблеме распределения задач по ЭМ УВС.

Некоторая последовательность $\hat{\mathcal{I}}$ объявляется базовой. Затем рассматриваются перестановки на расстоянии не больше h, $2 \leq h \leq L$, от базовой последовательности. В качестве расстояния между двумя последовательностями $\hat{\mathcal{I}}$ и $\hat{\mathcal{I}}' = \{I_{i_1}', I_{i_2}', \ldots, I_{i_\ell}', \ldots, I_{i_L}'\}$ примем число элементов множества $\mathcal{I}$, для которых предшествующие члены в $\hat{\mathcal{I}}$ и $\hat{\mathcal{I}}'$ различны. Метод получения последовательностей с расстоянием не больше h от базовой, основанный на псевдослучайных числах, описан в [3].

Если окажется, что время $\hat{T}'$ решения задач последовательности $\hat{\mathcal{I}}'$ с расстоянием h относительно $\hat{\mathcal{I}}$ меньше $\hat{T}$, то $\hat{\mathcal{I}}'$ берется в качестве базовой, то есть $\hat{\mathcal{I}} := \hat{\mathcal{I}}'(I_{i_\ell} := I_{i_\ell}', \ \ell = \overline{1,L})$.

Если же после получения g последовательностей с расстоянием h относительно $\hat{\mathcal{I}}$ базовая последовательность остается без изменения, то рассматриваются последовательности с расстоянием $h/2$ и т.д., пока не будут смоделированы последовательности с расстоянием 2.

В результате будет построена последовательность $\hat{\mathcal{I}}^*$.

Введем следующие операторы:

F_1 : формирование начальной базовой последовательности $\hat{\mathcal{I}} = \{I_{i_1}, I_{i_2}, \ldots, I_{i_\ell}, \ldots, I_{i_L}\}$, где $I_{i_\ell} := I_\ell \in \mathcal{I}$, $\ell = 1,2,\ldots, L$;

F_2 : формирование подмножеств $\mathcal{Y}_j \subset \hat{\mathcal{Y}}$, j = I,2,..., n ;

A_3 : вычисление $\hat{T}$;

$\mathcal{L}_4$: $k := h$;

$\mathcal{L}_5$: $x := 1$;

Φ_6 : формирование последовательности

$0 = x_0 = x_{t_0} \leq x_{t_1} \leq x_{t_2} \leq \ldots \leq x_{t_s} \leq \ldots \leq x_{t_{k-1}} \leq x_{t_k} = x_k = L$,
где $x_{t_s} = \xi_{t_s} \cdot L$, s = I,2,..., k – I, ξ_{t_s} – числа, формируемые генератором псевдослучайных чисел, причем $\xi_{t_s} \in$ (0,I);

F_7 : формирование частей *) $\hat{\mathcal{Y}}^s \subseteq \hat{\mathcal{Y}}$, s = I,2,..., k. В часть $\hat{\mathcal{Y}}^s$, s = I,2,..., k, включаются задачи $I_{i_\ell} \in \hat{\mathcal{Y}}$, для которых справедливо $[x_{t_{s-1}}] < i_\ell \leq [x_{t_s}]$, $[x]$ – ближайшее целое и такое, что $[x] \leq x$;

Φ_8 : формирование новой последовательности $\hat{\mathcal{Y}}' = \{I'_{i_1}, I'_{i_2}, \ldots, I'_{i_\ell}, \ldots, I'_{i_L}\}$ как случайной перестановки частей $\hat{\mathcal{Y}}^s$ в последовательности $\hat{\mathcal{Y}}$, s = I,2,..., k ;

F_9 : формирование подмножеств $\mathcal{Y}'_j \subset \hat{\mathcal{Y}}'$, j = I,2,..., n ;

A_{10} : вычисление $\hat{T}'$;

P_{11} : $P\{\hat{T}' < \hat{T}\}$ – проверка выполнения условия $\{\hat{T}' < \hat{T}\}$, $P = 0 \mapsto \mathcal{L}_{15}$ – переход на $\mathcal{L}_{15}$ при невыполнении условия;

$\mathcal{L}_{12}$: $\hat{T} := \hat{T}'$;

F_{13} : формирование базовой последовательности $\hat{\mathcal{Y}} := \hat{\mathcal{Y}}'$, то есть $I_{i_\ell} := I'_{i_\ell}$, ℓ = I,2,..., L, $\mathcal{Y}_j := \mathcal{Y}'_j$, j =I, 2,..., n ;

Π_{14} : $\to \mathcal{L}_5$;

$\mathcal{L}_{15}$: $x := x + 1$; P_{16} : $P\{x > g\}$, $P = 0 \to \Phi_6$;

$\mathcal{L}_{17}$: $k := \frac{1}{2} k$; P_{18} : $P\{k < 2\}$, $P = 0 \to \mathcal{L}_5$;

H_{19} : $\hat{\mathcal{Y}}_j \Rightarrow J_j \in \mathcal{J}$, $j = 1, 2, \ldots, n$; $\mathcal{L}_{20}$: $T := \hat{T}$;

$Я_{21}$: окончание распределения задач множества $\mathcal{Y}$ по элементарным машинам $J_j \in \mathcal{J}$, j = I,2,..., n.

Операторная схема [4] алгоритма имеет следующий вид:

$$\begin{array}{lllllll} F_1 & F_2 & A_3 & \mathcal{L}_4 & {}^{4,14,18}\mathcal{L}_5 & {}^{5,16}\Phi_6 & F_7 \\ \Phi_8 & F_9 & A_{10} & P_{11 \uparrow 15} & \mathcal{L}_{12} & F_{13} & \Pi_{14}^{5} \\ {}^{11}\mathcal{L}_{15} & P_{16 \uparrow 6} & \mathcal{L}_{17} & P_{18 \uparrow 5} & H_{19} & \mathcal{L}_{20} & Я_{21} . \end{array}$$

---

*) Некоторые части $\hat{\mathcal{J}}^s \subseteq \hat{\mathcal{J}}$ могут быть пустыми, s = I,2,..., k.

Легко заметить, что алгоритм I прост в реализации, время которой, как и качество распределения задач по машинам, зависит от значений h и g. Моделирование показывает, что даже при незначительных h и g алгоритм I осуществляет такое распределение задач по ЭМ УВС, при котором время решения T близко к своей нижней грани Θ. Это свидетельствует о том, что алгоритм I обеспечивает, по крайней мере, субминимальное значение времени решения задач на системе.

Алгоритм П

Считаем, что штраф за задержку решения задачи $I_i \in \mathcal{Y}$ составляет $c_i \neq 0$ денежных единиц в единицу времени. В качестве целевой функции целесообразно взять общую сумму штрафов F за задержку решения задач.

Требуется найти такое распределение задач $I_i \in \mathcal{Y}$ по ЭМ $J_j \in \mathcal{J}$, при котором функция F минимальна.

Прежде всего выведем формулу для функции штрафа F в предположении, что n = I. Пусть имеется последовательность решения задач $I_i \in \mathcal{Y}$, i = I,2,..., L,

$$\{I_{i_1}, I_{i_2}, \ldots, I_{i_k}, \ldots, I_{i_\ell}, \ldots, I_{i_L}\}, \qquad (I)$$

тогда

$$F(i_1, i_2, \ldots, i_\ell, \ldots, i_L) = t_{i_1}(c_{i_2} + c_{i_3} + \ldots + c_{i_L}) + t_{i_2}(c_{i_3} + \ldots + c_{i_L}) + \ldots + t_{i_{L-1}} \cdot c_{i_L} = C \cdot \mathcal{T} - f(i_1, i_2, \ldots, i_\ell, \ldots, i_L), \qquad (2)$$

где

$$C = \sum_{\ell=1}^{L} c_{i_\ell}, \quad \mathcal{T} = \sum_{\ell=1}^{L} t_{i_\ell},$$

$$f(i_1, i_2, \ldots, i_\ell, \ldots, i_L) = \sum_{\ell=1}^{L} t_{i_\ell} \sum_{z=1}^{\ell} c_{i_z}. \qquad (3)$$

Очевидно, что минимум (2) достигается при максимуме (3).

ТЕОРЕМА. Для того, чтобы последовательность решения задач (I) обес в е ч и в а л а м и н и м у м ф у н к ц и и ш т р а ф а $F(i_1, i_2, \ldots, i_\ell, \ldots, i_L)$ (2), н е о б х о д и м о и д о с т а т о ч н о, ч т о б ы в ы п о л н я л о с ь н е р а в е н с т в о

$$\frac{t_{i_1}}{c_{i_1}} \leqslant \frac{t_{i_2}}{c_{i_2}} \leqslant \ldots \leqslant \frac{t_{i_\ell}}{c_{i_\ell}} \leqslant \cdots \leqslant \frac{t_{i_L}}{c_{i_L}}. \tag{4}$$

НЕОБХОДИМОСТЬ. Докажем, что если при последовательности решения задач (I) функция (3) принимает максимальное значение, то выполняется (4).

Рассмотрим последовательность

$$\{I_{i_1}, I_{i_2}, \ldots, I_{i_\ell}, \ldots, I_{i_k}, \ldots, I_{i_L}\}, \tag{5}$$

причем (5) получена из (I) путем перестановки членов I_{i_k} и I_{i_ℓ}.

По определению

$$f(i_1, i_2, \ldots, i_k, \ldots, i_\ell, \ldots, i_L) - f(i_1, i_2, \ldots, i_\ell, \ldots, i_k, \ldots, i_L) =$$

$$= \sum_{q=1}^{L} t_{i_q} \cdot \sum_{z=1}^{q} c_{i_z} - \sum_{q=1}^{k-1} t_{i_q} \cdot \sum_{z=1}^{q} c_{i_z} -$$

$$- t_{i_\ell}\Big(\sum_{z=1}^{k-1} c_{i_z} + c_{i_\ell}\Big) - t_{i_{k+1}}\Big(\sum_{z=1}^{k-1} c_{i_z} + c_{i_\ell} + c_{i_{k+1}}\Big) - \cdots -$$

$$- t_{i_k} \cdot \Big(\sum_{z=1}^{k-1} c_{i_z} + c_{i_\ell} + c_{i_{k+1}} + \ldots + c_{i_{\ell-1}} + c_{i_k}\Big) - \sum_{q=\ell+1}^{L} t_{i_q} \cdot \sum_{z=1}^{\ell} c_{i_z} =$$

$$= (-t_{i_k} + t_{i_\ell})(c_{i_{k+1}} + \ldots + c_{i_\ell}) + (c_{i_k} - c_{i_\ell})(t_{i_{k+1}} + \ldots + t_{i_\ell}) \geqslant 0$$

Следовательно,

$$t_{i_k} \leqslant t_{i_\ell} + (c_{i_k} - c_{i_\ell}) \cdot \frac{t_{i_{k+1}} + \ldots + t_{i_\ell}}{c_{i_{k+1}} + \ldots + c_{i_\ell}}, \quad k < \ell. \tag{6}$$

Неравенство (6) является необходимым условием для того, чтобы задача $I_{i_k} \in \mathcal{J}$ решалась раньше $I_{i_\ell} \in \mathcal{J}$.

При $\ell = k+1$ условие (6) принимает следующий вид:

$$\frac{t_{i_k}}{c_{i_k}} \leqslant \frac{t_{i_{k+1}}}{c_{i_{k+1}}}$$

Учтя последнее неравенство, методом математической индукции легко доказать, что для любых двух задач I_{i_k} и I_{i_ℓ} последовательности (I), обеспечивающей максимальное значение функции (3), выполняется соотношение

$$\frac{t_{i_k}}{c_{i_k}} \leqslant \frac{t_{i_\ell}}{c_{i_\ell}}, \quad k < \ell.$$

Необходимость доказана.

ДОСТАТОЧНОСТЬ. Докажем, что если для последовательности (I) выполняется (4), то значение функции (3) будет максимальным.

Предположим противное. Пусть максимальное значение (3) соответствует некоторой последовательности

$$\{I_{s_1}, I_{s_2}, \ldots, I_{s_\ell}, I_{s_{\ell+1}}, \ldots, I_{s_L}\}, \tag{7}$$

для которой существует такое ℓ, что

$$\frac{t_{s_{\ell+1}}}{c_{s_{\ell+1}}} < \frac{t_{s_\ell}}{c_{s_\ell}} \tag{8}$$

Меняя местами задачи I_{s_ℓ} и $I_{s_{\ell+1}}$ в (7) и учтя (8), получим:

$$f(s_1, \ldots, s_{\ell-1}, s_\ell, s_{\ell+1}, \ldots, s_L) - f(s_1, \ldots, s_{\ell-1}, s_{\ell+1}, s_\ell, \ldots, s_L) =$$

$$= \sum_{q=1}^{L} t_{s_q} \sum_{z=1}^{q} c_{s_z} - \sum_{q=1}^{\ell-1} t_{s_q} \cdot \sum_{z=1}^{q} c_{s_z} - t_{s_{\ell+1}} \cdot \Big(\sum_{z=1}^{\ell-1} c_{s_z} + c_{s_{\ell+1}}\Big) -$$

$$- t_{s_\ell} \cdot \Big(\sum_{z=1}^{\ell-1} c_{s_z} + c_{s_{\ell+1}} + c_{s_\ell}\Big) - \sum_{q=\ell+2}^{L} t_{s_q} \sum_{z=1}^{q} c_{s_z} =$$

$$= t_{s_{\ell+1}} \cdot c_{s_\ell} - t_{s_\ell} \cdot c_{s_{\ell+1}} < 0.$$

Таким образом, получено противоречие с предположением о соответствии максимального значения функции (3) последовательности (7), что и доказывает достаточность условия теоремы.

Теорема доказана.[ж)]

Исходя из этой теоремы, в основу алгоритма II функционирования УВС можно положить следующий принцип: на освобождающуюся ЭМ назначать очередную задачу из последовательности (I), для которой выполняется условие (4).

Операторная схема алгоритма II проста, поэтому её приводить не будем.

Об эффективности алгоритмов

Пусть n = I0, L = I00, h = 8, q = 5; t_i и c_i – псевдослучайные числа, равномерно распределенные в промежутках (0; I0] и (0;5] соответственно, $i = \overline{I;I00}$. Результаты распределения задач по ЭМ УВС как при помощи алгоритмов I и II, так и случайным образом приведены в табл.I. T_j – время решения задач, а F_j – штраф за задержку решения задач, распределенных на элементарную машину $J_j \in \mathcal{J}$, $j = \overline{I;I0}$.

Приведем результаты статистической обработки экспериментов по моделированию алгоритмов функционирования однородных УВС.

Обозначим:

$$\vartheta^* = \frac{1}{\theta}(T^* - \theta), \ \vartheta^{**} = \frac{1}{\theta}(T^{**} - \theta), \ \vartheta = \frac{1}{\theta}(T - \theta),$$

где T^*, T^{**}, T – время решения задач, распределенных по алгоритмам I, II и случайно. В табл.2 приведены оценки математических ожиданий $\widetilde{M}$ и дисперсий $\widetilde{D}$ величин ϑ^*, ϑ^{**}, ϑ.

Оценки математических ожиданий и эмпирические дисперсии величин

$$\tau = \frac{1}{T^{**}}|T - T^{**}|, \ \varphi = \frac{1}{F^{**}}(F - F^{**}),$$

ж) В работе [5] подобный результат получен для систем массового обслуживания с несколькими входящими потоками для относительных и абсолютных приоритетов.

Таблица I

	способ \ J_j	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}	J_j \ T	θ
T_j	алгоритм I	48,60	44,43	48,50	46,05	47,58	48,38	46,14	47,10	48,20	47,02	48,50	47,2
	алгоритм II	43,83	49,58	5I,22	48,75	43,54	44,35	44,66	5I,40	46,69	47,98	5I,40	
	случайно	49,90	44,40	52,24	44,26	45,42	45,02	50,37	47,73	47,78	44,88	52,24	
F_j	алгоритм II	260,I	242,3	I97,6	258,I	247,4	227,2	233,2	2II,4	22I,9	25I,4	235I,I	
	случайно	383,0	6II,8	550,3	9I9,4	649,4	43I,9	70I,0	6II,3	545,3	795,8	6I99,2	

Таблица 2

	ϑ^*	ϑ^{**}	ϑ
$\widetilde{M}$	$I9 \cdot I0^{-3}$	$95 \cdot I0^{-3}$	$90 \cdot I0^{-3}$
$\widetilde{D}$	$78 \cdot I0^{-6}$	$670 \cdot I0^{-6}$	$500 \cdot I0^{-6}$

Таблица 3

	τ	φ
$\widetilde{M}$	$I5 \cdot I0^{-3}$	I,I3
$\widetilde{D}$	$52 \cdot I0^{-6}$	$64 \cdot I0^{-3}$

где F и F^{**} – величины штрафов при случайных назначениях и по алгоритму II, представлены в табл.3.

Из таблиц видно, что даже для малых h и g ($h = 8$, $g = 5$) алгоритм I осуществляет такое распределение задач по ЭМ УВС, при котором время решения T существенно ближе к своей нижней границе θ, чем при случайных назначениях. Алгоритм II обеспечивает минимум функции штрафа F, который, по крайней мере, в два раза меньше величины штрафа при случайном распределении задач.

Сложность реализации алгоритмов незначительна по сравнению с алгоритмами математического программирования. (В приведенном примере время реализации на вычислительной машине "М-220" алгоритма I составляет 0,5 мин, а алгоритма II – I мин.).

Выводы

I. Алгоритм I осуществляет такое распределение задач по ЭМ УВС, при котором значение времени T решения задач близко к своей нижней грани, кроме того, оно меньше значения T, получаемого при случайных назначениях.

2. Алгоритм II обеспечивает минимум функции штрафа. Время решения задач близко к значению, получаемому при распределении задач по ЭМ УВС случайным образом.

3. Рассматриваемые алгоритмы функционирования однородных УВС просты в реализации и в отличие от алгоритмов математического программирования могут быть использованы при диспетчировании.

4. Алгоритмы эффективно реализуются на однородных вычислительных системах.

ЛИТЕРАТУРА

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

2. В.Г. ХОРОШЕВСКИЙ. Об алгоритмах распределения задач по ЭЦВМ.– Труды СФТИ, Томск, Изд-во ТГУ, I965, вып.47, стр. 29 –34.

3. E.S.PAGE. On Monte-Carlo methods in congestion problems : I. Searching for an optimum in discrete situations. – Operation Research, The Journal of the Operation Research Society of America, Virginia, 1965, v.13, №2, p.291–299.

4. Н.П.БУСЛЕНКО. Моделирование сложных систем. М., "Наука", I968.

5. О.И.БРОНШТЕЙН, В.В.РЫКОВ. Об оптимальных приоритетах в системах массового обслуживания.– Изв.АН СССР, Техническая кибернетика, 1965, №6, стр. 28 – 37.

Поступила в редакцию
20.XI. 1969 г.

П р и л о ж е н и е

Приведем программы алгоритмов, записанные на АЛГОЛе. Для алгоритма I необходимо задать следующие величины: L – число задач; n – число ЭМ в УВС; h – расстояние между задачами ; g – допустимое число попыток, не улучшающих значение $\hat{T}$; $u0$, $u1$ – псевдослучайные числа в интервале (0,4); $T[1:L]$ – массив значений времени решения задач.

На печать выдаются следующие значения: θ , $T[1:n]$ – массив времен решения задач на каждой ЭМ; $K[1:n, 1:s[j]]$ – номера задач, назначенных для решения на J_j , $j = 1,\ldots, n$,где $s[j]$ – число задач, назначенных на J_j .

Для алгоритма II необходимо задать L ; n ; $t[1:L]$; а также $c[1:L]$ – массив величин штрафа за задержку решения задачи на единицу времени.

На печать выдаются значения: $T[1:n]$, $F[1:n]$ – величины штрафов по машинам; $K[1:n, 1:s[j]]$.

Программа алгоритма I

```
begin  integer L, n, h, g, i, j, z, p, d, k;   real F, θ , T1,
        T2, τ , u0, u1;
      read (L, n, h, d, u0, u1);
begin   k:= entier (L/n)+n;
begin   real array t[1:L], r2[1:L], r1[1:L], T[1:n],  X [1:L],
        v[0:h], s[1:h, 1:L-h];
        integer array a,b,[1:n, 1:k], l[0,h], ӣ [0:h];
      read (t);
begin   procedure H(f,L,n,φ , θ ,a,V); array f,a; real φ ,V;
begin   real δ1, δ2, F1; F1:= φ ;  for  i:= 1,...,n  do T[i] :=
        0;  V:= θ  ; j:= 1;  for i:= 1,...,n-1  do { p:= 1;  D:
        T[i] := T[i]+f(j);  if  T[i]< θ     then  { a[i,p] := j ;
        p:= p+1;  if  j < L  then  { j:= j+1;  go to D}} else
        {δ1:= T[i]; δ 2:= (F1-T[i]+f[j])/(n-i);  if δ1 < δ2
        then { F1:= F1-T[i] ;  a[i,p] := j;  if  T[i] > V then
        V:= T[i];  if  j < L  then  j:= j+1}  else {F1:= F1 -
        T[i] +f[j]}}};   p:= L-j+1;  for  i:= 1,...,p do {a[n ,
        i] := j; T[n]:= T[n]+f[j];  j:= j+ } ;  if V< T[n] then
        V:= T[n]
                                                          end;
        F:= 0;  for  i:= 1,...,L  do  F:= F+t[i]; θ := F/n;
        H(t,L,n,F, θ ,a,T1);  d:= 1; Q:  for  i:= 1,...,h-1 do
        { τ := u0+u1;  u0:= u1;  if  τ >4  then  τ := τ -4;
        u1:= τ ; v[i] := τ /4} ;  for  i:= 1,...,h-1  do X[i] :=
        L x v[i];  p:= 1;   A:  z:= 1;  for  i:= 1,...,h-1  do
        { if X[p]= X[i] then {if  p > i  then  z:= z+1} else
        { if X[p]> X[i]  then  z:= z+1}} ; r1[z]:= X[p]; if p<
        h-1  then { p:= p+1; go to  A } ;  for  z:= 1,..., h-1
        do  ӣ[z]:= entier (r1[z]);  ӣ[0]:= 0; ӣ[h] := L;  for
        i:= 1,...,h  do { τ:= u0+u1; u0:= u1;  if τ > 4  then
        τ := τ - 4; u1:= τ  ; v[i] := τ /4} ; p:= 1;   B: z:= 1;
        for i:= 1,...,h  do {if v[p]= v[i]  then { if p > i
        then z:= z+1}  else  {if  v[p]> v[i] then z:= z+1}};
        r1[p]:= z; l[z] := ӣ[p]- ӣ[p-i] ;  if  p < h  then {p:=
        p+1; go to B};  for i:= 1,...,h do {if ӣ [i]> ӣ [i-1]
        then {for j:= (ӣ [i-1] +1)  step 1 until ӣ[i]  do
        s[r1[i], j- ӣ[i-1]] := t[j]}};  for  j:= 1,...,L  do
        r2[j]:= 0; j:= 1; z:= 1;  R: if  l[z] > 0  then  {for
        i:= 1,...,l[z]  do {r2[j]:= s[z,i] ; j:= j+1}}; if z< h
        then {z:= z+1;  go to R};  H(r2,L,n,F,θ ,b,T2);  if
        T2 <  T1  then {T1:= T2;  for i:= 1,...,L  do {t[i] :=
        r2[i]}; d:= 0;  for i:= 1,...,n  do {for j:= 1,...,(L
        -n+1) do { a[i,j] := b[i,j]; b[i,j] := 0}}} ; d:= d+1;
        if d <  g  then {go to Q} else {h:= h/2; if h > 1  then
        go to Q } ;  print (t, θ ,a,T1)
                                  end end end end;
```

Программа алгоритма II

```
begin integer  n, L, p, i, j, z;  real  k, r;  read (n, L);
begin  l:= entier (L/n);
```

```
begin integer array  s[1:n], K[1:n, 1:2l] ; real array t[1:L],
      c[1:L], v[1:L], r1[1:L], r2[1:L], T[1:n], C[1:n],
      F[1:n];   read  (t,c);
begin for i:=1,...,L  do  v[i]:= t[i]/c[i];  p:= 1; A: z:= 1;
      for i:=1,...,L  do  { if v[p]= v[i] then { if p > i  then
      z:= z+1 }  else { if v[p] > v[i] then z:= z+1}} ;  r1[z]:=
      t[p];  r2[z]:= c[p];  if p < L then {p:= p+1; go to A} ;
      for z:= 1,...,L  do { t[z]:= r1[z]; c[z]:= r2[z]};  for
      i:= 1,...,n  do { s[i]:= 1;  K[i, s[i]]:= i; T[i]:=t[i]};
      j:= n+1;  R:  k:= T[1]; z:= 1; for i:= 2,...,n  do { if
      T[i] < k then { k:= T[i]; z:= i}} ; T[z]:= T[z]+t[j]; s[z]:=
      s[z]+1; K[z, s[z]]:= j;  if j < L then { j:= j+1;  go to
      R};  for  i:= 1,...,n do  {C[i]:= 0; z:= 0; for p:= 1, .
      ..,s[i] do { C[i]:= C[i]+c[K[i,p]]} ;  F[i]:= C[i]x T[i];
      for p:= 1,...,s[i] do { z:= z+c[K[i,p]]; F[i]:= F[i] -
      t[K[i,p]] x z}} ;  print  (T,F,t,K)
                                       end end end end ;
```