ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ
1975 год Выпуск 63

УДК 681.142.4

АЛГОРИТМЫ ПЛАНИРОВАНИЯ ФУНКЦИОНАЛЬНЫХ СОСТОЯНИЙ ОДНОРОДНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

Э.Г.Крылов, Н.Н.Миренков

Одной из задач диспетчера однородной вычислительной системы (ОВС) [1] является планирование функциональных состояний, то есть разбиение на подсистемы и назначение подсистемам режимов работы [2]. Для идентификации функциональных состояний системы вводятся управляющие функции $\Psi(i,j,t)$ и $\xi(\tau(t),\mathcal{J}(t))$, позволяющие задавать универсальную схему функционирования ОВС в рамках возможностей, предоставляемых управляемыми связями:

$\Psi(i,j,t)$ определяет разбиение системы на подсистемы*), причем $\Psi(i,j,t)=1$, если i-я машина принадлежит j-й подсистеме в момент времени t, иначе $\Psi(i,j,t)=0$; $1 \le i \le L$; $L = \sum_{i,j} \Psi(i,j,t)$ – число машин в ОВС, $1 \le j \le \mathcal{J}(t) \le L$; $\mathcal{J}(t)$ – число подсистем в момент времени t;

$\xi(\tau(t),\mathcal{J}(t))$ задает каждой подсистеме режимы работы на некотором промежутке времени $[T_0, T_1]$:

$$\xi(\tau(t),\mathcal{J}(t)) = \left\{ \xi^1(\tau^1(t)), \xi^2(\tau^2(t)),\dots, \xi^{\mathcal{J}(t)}(\tau^{\mathcal{J}(t)}(t)) \right\},$$

где $\tau(t)=\{\tau^j(t)\}$, $j=1,2,\dots,\mathcal{J}(t)$; при этом $\tau^j(t)$ принимает множество значений $\{\tau^j_k(t)=t^j_{k+1}(t)-t^j_k(t)\}$, $k=1,2,\dots,M^j-1$; $\{t^j_k\}$ – множество точек, зависящих от времени и выбираемых на $[T_0, T_1]$, $t^j_1 = T_0$, $t^j_k \le t^j_{k+1}$, $t^j_{Mj} = T_1$;

*) Подсистема – подмножество машин ОВС, обеспечивающее реализацию заданных схем обмена [3].

$$\xi^j(\tau_k^j)=\begin{cases} 0, & \text{если} \quad \tau_k^j = 0; \\ \sum\limits_{h=1}^{Y} \varphi_h(\tau_k^j), & \text{если } \tau_k^j \neq 0; \end{cases}$$

$\varphi_h(\tau_k^j) = \nu_h \tau_k^j$, если $\left[\frac{k+Y-h}{Y}\right] = \frac{k+Y-h}{Y}$ ($[x]$ – целая часть x); иначе $\varphi_h(\tau_k^j) = 0$; ν_k – коэффициент (признак), определяющий h-й режим работы; Y – число различных режимов.

Таким образом, подсистема j работает в момент времени $t \in [t_k^j \,.\, t_{k+1}^j]$ $(t_k^j \neq t_{k+1}^j)$ в режиме ν_h , если $\xi^j(\tau_k^j)/\tau_k^j = \nu_h$.

Функционирование системы называется с т а т и ч е с к и м, если оно задается управляющими функциями, не зависящими от времени; д и н а м и ч е с к и м , если управляющие функции зависят от времени и зависимость известна в момент $t \leq T_0$; с т о х а с т и ч е с к и м , если эта зависимость неизвестна (формируется под воздействием внешних условий, случайных факторов).

Управляющие функции в каждый момент времени представляются в оперативной памяти с помощью УФ-таблицы из L строк следующего вида:

H	K	ν	АДРЕС

,

где H и K – признаки машин с минимальным и максимальным относительными номерами в подсистеме, соответственно; ν – признак режима, в котором работает подсистема; АДРЕС – номер строки УФ-таблицы, принадлежащий машине со следующим относительным номером в подсистеме, если признак K не отмечен, иначе АДРЕС есть номер стандартной программы (СП), определяющий режим-преемник и условия окончания текущего режима.

Зависимость УФ-таблицы от времени в некоторых случаях может полностью переноситься на указанные СП, что устанавливает следующее

УТВЕРЖДЕНИЕ. Динамическое функционирование системы может быть задано управляющими функциями, из которых $\Psi(i,j,t) = \Psi(i,j,t')$ при $t' \leq T_0$, если соседние подсистемы [2], работающие в одном и том же режиме, рассматривать как одну подсистему суммарного ранга*).

*) Ранг – число ЭМ в подсистеме.

Действительно, пусть последовательность функциональных состояний системы задается функциями $\Psi_1(i,j,t)$ и $\xi_1(\tau(t), \mathcal{J}(t))$, реализующими в некоторый момент времени $t^* \in [T_0, T_1]$ состояние F_1 , при котором

$$\sum_{i=1}^{L} \Psi_1(i,j,t^*) = R_j \quad \text{и} \quad \sum_{j=1}^{\mathcal{J}(t)} R_j = L;$$

причем для каждой из подсистем выполняются равенства:

$$\xi_1^j(\tau_k^j(t^*))/\tau_k^j(t^*) = \nu_{\alpha_j}.$$

Для доказательства достаточно

I) задать в момент времени $t' \leq T_0$ функцию $\Psi_2(i,m)$, удовлетворяющую условию

$$\sum_{i=1}^{L} \Psi_2(i,m) = 1,$$

$$m = 1,2,\ldots,L;$$

2) для каждой из полученных подсистем выбрать свою функцию $\xi_2^m(\tau(t))$, у которой множество значений $\{\tau_k^m\}$ совпадает с множеством значений $\{\tau_k^j\}$ и $\xi_2^m(\tau_k^m(t^*))/\tau_k^m(t^*) = \nu_{\alpha_j}$, если машина m принадлежит j-й подсистеме состояния F_1 .

I. З а д а ч и д и н а м и ч е с к о г о ф у н к ц и о н и р о в а н и я . Задать динамическое функционирование системы – значит определить для каждого момента времени разбиение системы на подсистемы и режимы работы для каждой из подсистем. В связи с этим диспетчеру*) ОВС приходится решать ряд задач, типичные представители которых рассматриваются ниже.

В систему поступил набор заданий (задач) $Z = \bigcup_{i=1}^{N} Z_i$, известны ограничения на порядок их выполнения, ранги R_i и время t_i , необходимое для реализации каждого задания.

Требуется для решения поступившего набора выбрать размер подсистемы и для неё определить последовательность функциональ-

*) Имеется в виду диспетчер в программном обеспечении общего назначения.

ных состояний, чтобы обеспечить одно (или оба) из следующих условий:

- максимальную загрузку машин подсистемы;
- реализацию набора в заданный промежуток времени.

Решению подобных задач посвящен ряд работ. В [4–6] они рассматриваются для случая $R_i = 1$, $t_i = 1$; в [7,8] для произвольных t_i, но одинаковых R_i, в [9,10] для произвольных R_i и t_i, но при отсутствии ограничений на порядок выполнения заданий. Здесь рассматривается решение задачи в более общей постановке.

Предполагается, что порядок реализации поступившего набора заданий определяется ориентированным графом без петель и циклов. Этот порядок специфицируется приоритетами заданий или реальными, например, технологическими процессами. Исходный граф, где каждому заданию соответствует вершина, преобразуется в граф, где с каждым заданием может связываться несколько вершин в зависимости от числа временных квантов, содержащихся в t_i. Алгоритм выбора эффективного размера временного кванта приводится в [9]. С каждой вершиной графа (рис.I) связывается ранг R соответствующего задания, порядковый номер (индекс при R) и метка, определяющая у р о в е н ь вершины, то есть наибольшее расстояние до конечной вершины (так как конечных вершин может быть несколько, то всегда вводится дополнительная фиктивная вершина с первым номером, нулевым уровнем и рангом, равным числу ЭМ в системе). Алгоритм присвоения указанных меток имеется в [4].

Две вершины графа считаются независимыми, если ни одна из них не предшествует другой.

ОПРЕДЕЛЕНИЕ I. Максимальная совокупность попарно независимых между собой вершин называется н а б о р о м б л и ж а й ш е г о р а с с м о т р е н и я, если любое из заданий, соответствующих этим вершинам, может быть реализовано в данный момент.

ОПРЕДЕЛЕНИЕ 2. Пусть L – некоторое целое положительное число, $R = \bigcup_i R_i$ – множество рангов, соответствующих набору ближайшего рассмотрения. Процесс выбора подмножества элементов $R_s = \bigcup_{j=1}^{s} R_{i_j} \subset R$, обеспечивающего минимум функции $D(s) = L - f(s)$ при $f(s) = \sum_{j=1}^{s} R_{i_j} \leq L$, называется у п а к о в к о й L-местного слоя.

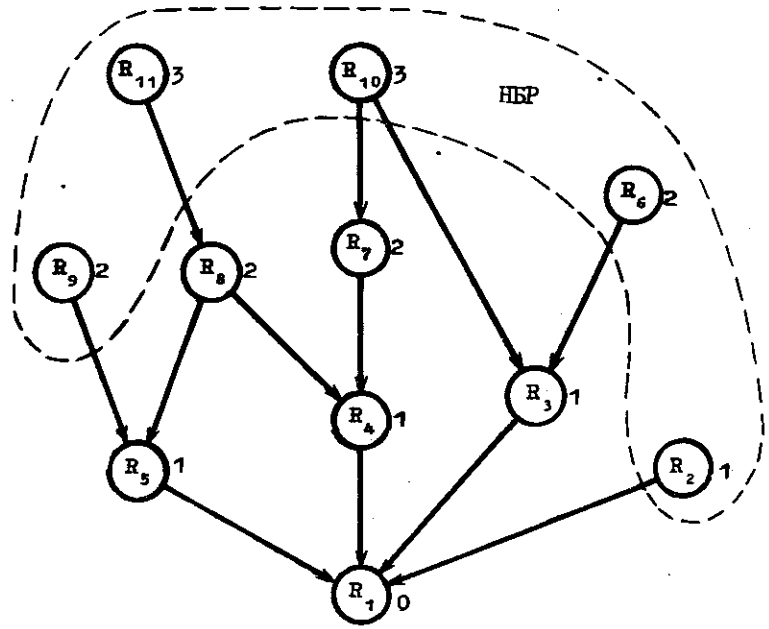

Рис. I

Решение поставленной задачи осуществляется применением для различных значений L следующего алгоритма

A_0 A_1 A_2 P_0 A_3 A_4 A_5 P_1 A_6 P_2 P_3 A_7 Я,

A_0 – $\mathcal{J} := 0$; $B := 0$; $\mathcal{D} := 0$;

A_1 – $n := L$; $\mathcal{J} := \mathcal{J} + 1$; $B := B + \mathcal{D}$;

A_2 – выделяет вершины набора ближайшего рассмотрения;

P_0 – передает управление на A_7, если набор ближайшего рассмотрения содержит первую вершину;

A_3 – разбивает набор ближайшего рассмотрения на группы с одинаковыми уровнями, упорядочивает группы по убыванию уровней и присваивает группам убывающие номера;

A_4 – $\rho := m$ (m – число различных групп);

A_5 – реализует упаковку n-местного слоя, используя ρ-ю группу;

P_1 – передает управление на A_4, если $D = 0$;

A_6 – $n := D$; $\rho := \rho - 1$;

P_2 – передает управление на A_5, если $\rho > 0$;

P_3 – безусловно передает управление на A_4;

A_7 – $\mathcal{T} := \mathcal{T} - 1$;

Я – оператор конца.

После выполнения алгоритма $\alpha = \frac{B}{\mathcal{T} \cdot L}$ характеризует качество загрузки L машин в течение $\mathcal{T}$ квантов времени, за которое решается заданный набор; $B = \sum_i D_i$, $i = 1,2,\ldots,\mathcal{T}$.

При практической реализации алгоритма порядок выполнения поступивших задач, их уровни и ранги определялись матрицей $A[i,j]$ ($i,j = 1,2,\ldots,N+1$, где N – число вершин, соответствующих реально существующим заданиям), в которой $A[i,i] = R_i$, $A[j,i] = -1$, если i-я вершина предшествовала j-й, иначе $A[j,i] = 0$, за исключением $A[1,j]$, равной уровню вершины с номером j.

Выделение набора ближайшего рассмотрения сводилось к выявлению строк, элементы которых, за исключением диагональных, равнялись нулю. Строки и столбцы матрицы $A[i,j]$ вычеркивались из рассмотрения, если соответствующие вершины оказывались упакованными.

Для работы с набором ближайшего рассмотрения организовывались три массива Q, R и K. Первый содержал упорядоченные по убыванию уровни вершин, второй – ранги, упорядоченные для каждого уровня, третий – номера вершин, причем i-й элемент каждого массива соответствовал одной и той же вершине. (При назначении номеров вершинам с большими рангами при прочих равных условиях присваивались большие номера.)

Оператор упаковки A_5 выполнялся в две стадии. На первой стадии из упорядоченного по убыванию множества рангов $R^6 = \{R^6_1, R^6_2, \ldots, R^6_R\}$, соответствующих наибольшему уровню 6, слева подряд набиралось s элементов, до тех пор пока не выполнялось неравенство $\sum_{i=1}^{s} R^6_i \geq n$. В случае равенства процесс упаковки слоя считался законченным, в случае неравенства – последний выбранный элемент пропускался, а процесс подбора продолжался. После просмотра элементов множества R^6 упаковка пе-

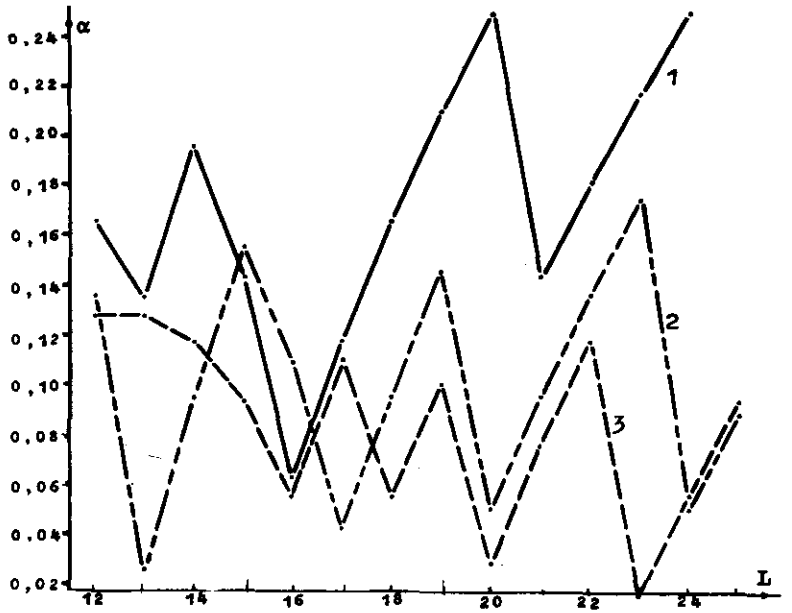

Рис. 2

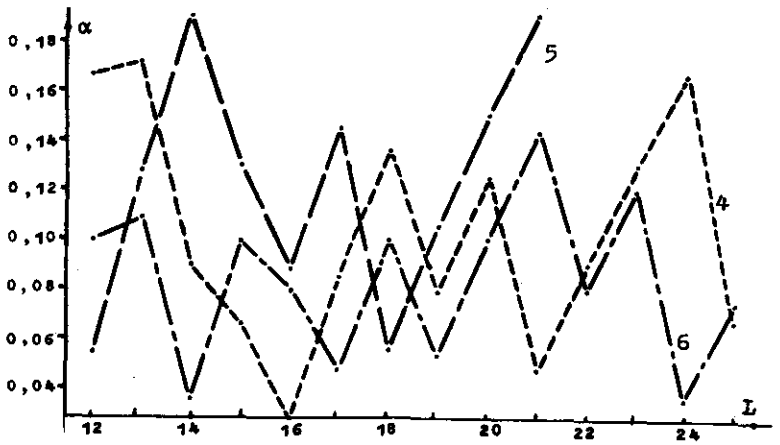

Рис. 3

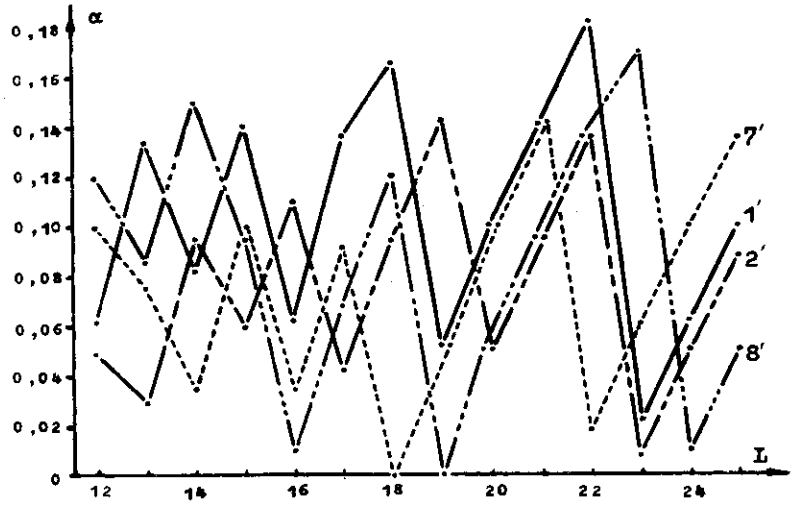

Рис. 4

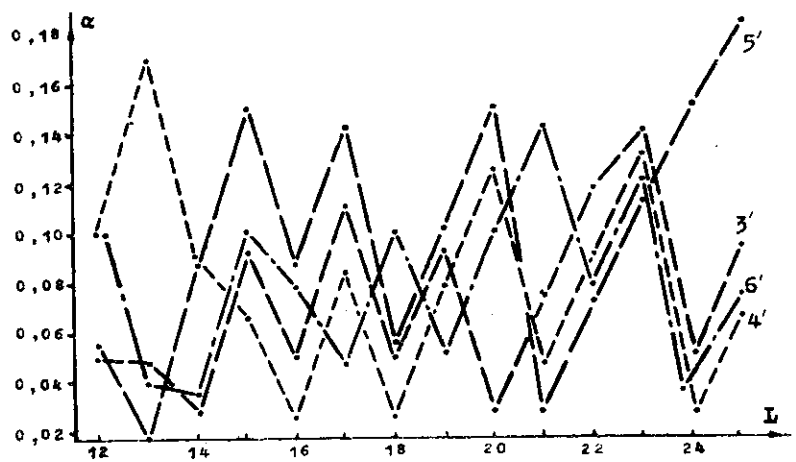

Рис. 5

реходила во вторую стадию, если величина $D = n - \sum_{j}^{s} R^{\sigma}_{ij}$ (суммирование по элементам, попавшим в набор) была больше нуля. На второй стадии создавалась последовательность множеств, каждое из которых получалось исключением из множества R^{σ} элемента R^{σ}_{ij}, $j = 2,3,\ldots,s$, принадлежащего подмножеству $\{R^{\sigma}_{i_1}, R^{\sigma}_{i_2}, \ldots, R^{\sigma}_{i_s}\}$, выделенному для упаковки на первой стадии. Для каждого такого множества выполнялись действия первой стадии. Процесс упаковки заканчивался, если обнаруживался набор, при котором $D = 0$; в противном случае проверялись все созданные множества и упакованным считался набор элементов с минимальным D.

Результаты численных экспериментов, основанных на статистическом моделировании работы рассмотренного алгоритма, представлены на рис.2,3 и табл.I – для 6 наборов с ограничениями порядка выполнения задач; на рис.4,5 и табл.2 – для 8 наборов без ограничений. Число задач в наборах менялось от I6 до 32, максимальный ранг задач не превышал I0, время выполнения каждой задачи – один квант. Расписания составлялись для подсистем с числом ЭМ от I2 до 25, причем на каждом шаге регистрировалось число α – качество загрузки подсистемы. Используя подобные результаты, можно выбрать размер подсистемы режима параллельной обработки, обеспечивая реализацию задач к нужному сроку с высокой загрузкой подсистемы.

Из экспериментов можно сделать следующий вывод. Для любых набора заданий и числа L машин всегда находится число L_0, такое что $|L - L_0| \le X$ (как правило, $X \approx 2$), для которого предложенный алгоритм обеспечивает достаточную степень загрузки подсистемы, то есть варьирование числа L позволяет простому эвристическому алгоритму давать удовлетворительные для практики результаты.

2. Задачи стохастического функционирования. Задать стохастическое функционирование системы – значит определить СП УФ–таблицы, то есть правила преобразования структуры ОВС при возникновении той или иной мультипрограммной ситуации. В качестве основных выделяются правила распознавания в подсистемах нужных структур [3], перевода машин из одного режима работы в другой [II], а также правила организации подсистем при обслуживании потока заданий.

Временные кванты, за которые реализуются наборы при различном числе L машин

Таблица I

№ эксперимента \ L	I2	I3	I4	I5	I6	I7	I8	I9	20	2I	22	23	24	25
I	9	8	8	7	6	6	6	6	6	5	5	5	5	5
2	II	9	9	9	8	7	7	7	6	6	6	6	5	5
3	I3	I2	II	I0	9	9	8	8	7	7	7	6	6	6
4	I4	I3	II	I0	9	9	9	8	8	7	7	7	7	6
5	9	9	9	8	7	7	6	6	6	6	6	6	6	6
6	I5	I4	I2	I2	II	I0	I0	9	9	9	8	8	7	7

Таблица 2

№ эксперимента \ L	I2	I3	I4	I5	I6	I7	I8	I9	20	2I	22	23	24	25
I′	8	8	7	7	6	6	6	5	5	5	5	4	4	4
2′	I0	9	9	8	8	7	7	7	6	6	6	5	5	5
3′	I2	II	I0	I0	9	9	8	8	7	7	7	6	6	6
4′	I2	I2	II	I0	9	9	8	8	8	7	7	7	6	6
5′	9	8	8	8	7	7	6	6	6	5	5	5	5	5
6′	I5	I3	I2	I2	II	I0	I0	9	9	9	8	8	7	7
7′	I0	9	8	8	7	7	6	6	6	5	5	5	5	5
8′	9	8	8	7	6	6	6	5	5	5	5	5	4	4

Сложность работы с потоком заданий обусловливается как случайным характером поступления заданий, так и случайным временем их обслуживания. Задача состоит в том, чтобы определить стратегию оперативного, связанного с конкретной ситуацией, выбора размера подсистемы, обслуживающей параллельную обработку, и назначения последовательности функциональных состояний для этой подсистемы.

Предлагается следующая стратегия. В каждый момент времени в системе обычно существует набор заданий, известны их ранги, предполагаемое время выполнения, приоритеты (ограничения на порядок выполнения), а также максимально возможное время реализации (израсходовав которое задание снимается с обслуживания).

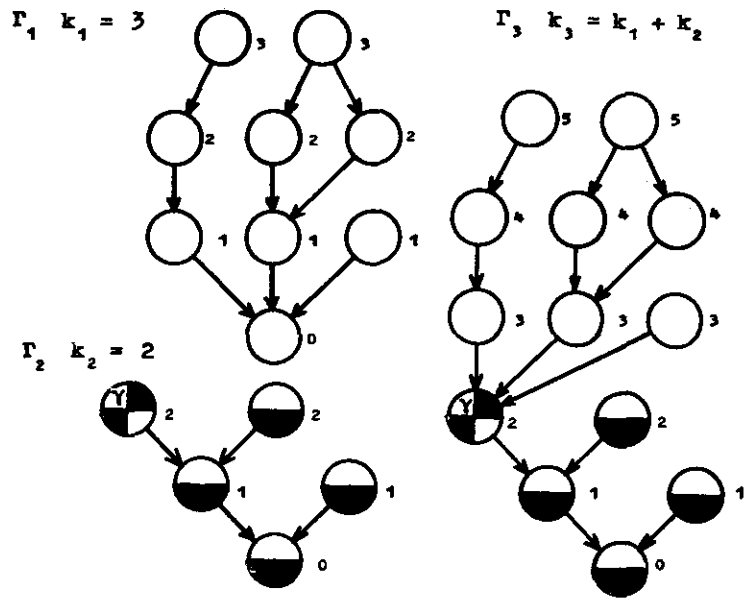

Рис. 6

Используя эти данные и алгоритмы из [9], выбирается размер τ временного кванта, в течение которого не предполагается менять функциональное состояние системы.

На основании полученного значения τ с помощью алгоритма, рассмотренного в предыдущем пункте, выбирается размер подсистемы и последовательность функциональных состояний.

Поскольку со временем ситуация изменяется в частности может появиться новый набор заданий, то необходима коррекция тактики обслуживания. Пересмотр расписания, определенного старым набором заданий, реализуется в следующих случаях:

I) если отсутствуют в старом наборе задания для загрузки некоторых подсистем до завершения временного кванта, что связано с несоответствием предполагаемого и фактического времени решения задач;

2) если имеются в очередном функциональном состоянии незагруженные подсистемы;

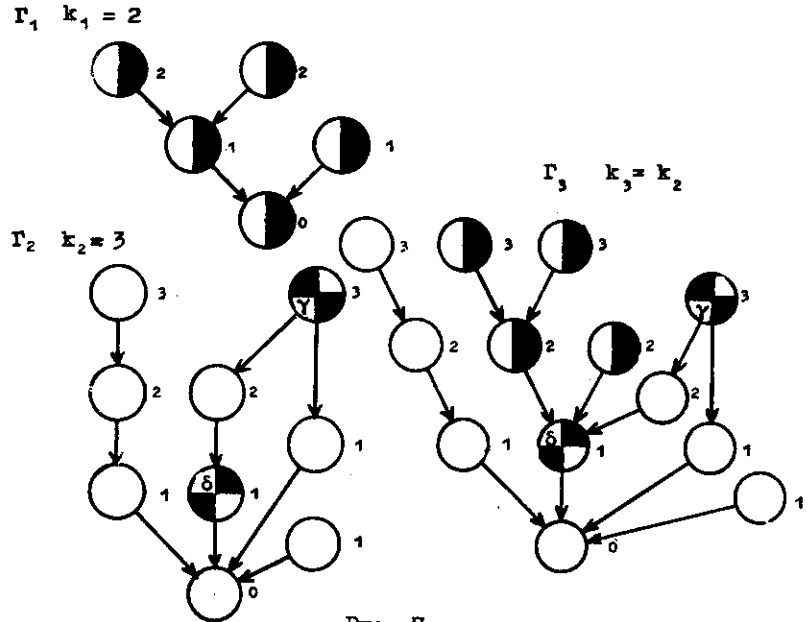

Рис. 7

3) если имеются в новом наборе задания с высоким приоритетом.

Пусть в одном из этих случаев остаток от старого набора заданий представляется графом Γ_1, а новый набор – графом Γ_2 (рис. 6,7). При объединении этих графов для выработки новой тактики обслуживания выполняется следующее. В графах Γ_1 и Γ_2 определяются величины k_1 и k_2, равные максимальным уровням вершин; в графе Γ_2 из вершины с уровнем k_2 выбирается вершина γ с максимальным рангом и, если $k_1 < k_2$, вершина δ с уровнем $k_2 - k_1$, находящаяся на максимальном пути из γ.

Объединение Γ_1 и Γ_2 в Γ_3 выполняется совмещением конечной вершины Γ_1 с вершиной γ графа Γ_2, если $k_1 \geq k_2$ (рис.6), иначе конечная вершина Γ_1 совмещается с вершиной δ графа Γ_2 (рис.7). На основе полученного графа и упомянутых выше алгоритмов вырабатывается новая тактика обслуживания поступивших наборов заданий.

Отметим, что при практической реализации переход к новой тактике, связанный с неполной загрузкой, может выполняться не каждый раз, а только в случае недогрузки сверх допустимой нормы, причем в этом случае возможны прерывания текущего функционального состояния и переход к очередному состоянию, запланированному на основании старого набора заданий.

Типичные результаты численных экспериментов, выполненных с помощью специальной моделирующей программы, приведены на рис.8 и 9, первый из которых характеризует среднюю загрузку подсистемы потоками задач в зависимости от ранга подсистемы, второй – загрузку α' подсистемы ранга $L = 12$ в зависимости от Δt (числа машин, определяющего сегмент $[L-\Delta L, L]$, в пределах которого разрешено варьирование размера подсистемы). Максимальный ранг задач в приведенных экспериментах равнялся 6. Представлены пять потоков по 30 случайных наборов задач в каждом. Набор задач интерпретировался сетевым графом с числом вершин в графе от 20 до 50. Характеристики потоков приведены в табл. 3.

Таблица 3

Номер потока	Шкала рангов в наборах	Наличие связей в наборах	Суммарное число вершин в потоке	Длительность реализации потока в квантах времени при $L = 12$
I	2,3,4,5,6	да	1123	399
2	2,3,4,5,6	нет	1078	372
3	2,3,5,6	да	1044	377
4	2,3,6	да	1085	375
5	2,6	да	1136	393

Из экспериментов можно сделать следующие выводы: I) наилучшая загрузка подсистем получается при выполнении условия $L \geq 2 \cdot R_{max}$, где L – размер подсистемы, R_{max} – максимальный ранг задач в поступающих наборах; 2) основной эффект от динамического выбора размера подсистемы, реализующей поток задач, проявляется при $\Delta L \approx 1 \div 2$.

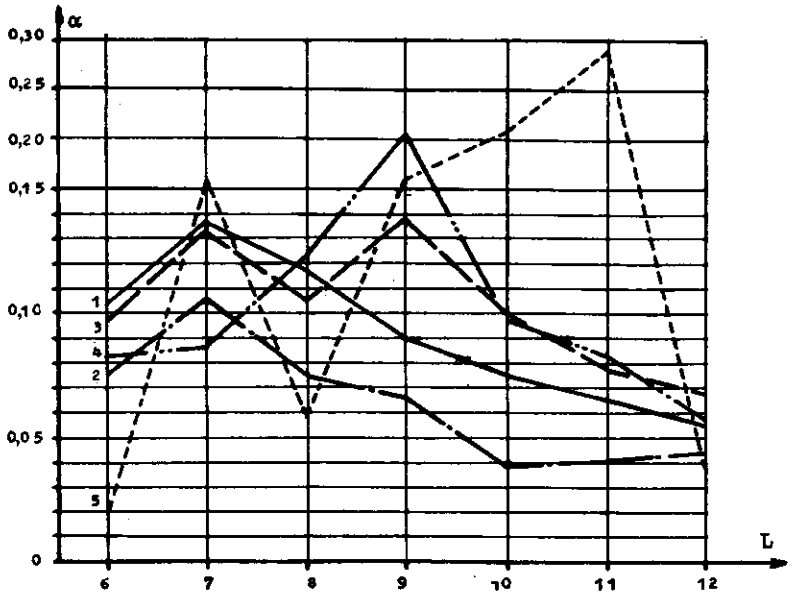

Рис. 8

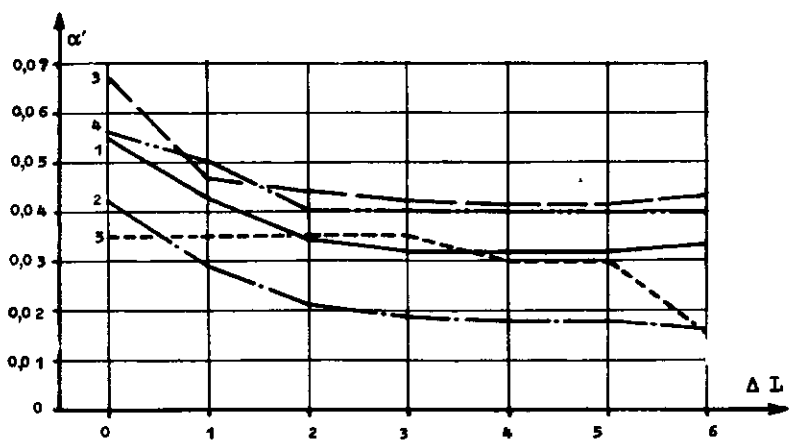

Рис. 9

Описанные алгоритмы планирования функциональных состояний ОВС позволяют с единых позиций оценивать мультипрограммную ситуацию и выбирать стратегию планирования как при обработке заданного набора, так и при обслуживании случайного потока задач.

Разработанные методы положены в основу управляющей программы, создаваемой для системы МИНИМАКС.

Литература

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

2. МИРЕНКОВ Н.Н. Диспетчирование в однородных вычислительных системах. -В кн.: Материалы 3-й Всесоюзной конференции по однородным вычислительным системам и средам. Таганрог, 1972, с. 51-54.

3. МИРЕНКОВ Н.Н., ФИШЕРМАН С.Б. Алгоритмы распознавания подсистем заданных структур в ОВС. -Настоящий сборник, с.44-53.

4. ХУ Т.С. Параллельное упорядочивание и проблемы линии сборки. -"Кибернетический сборник", 1967, № 4, с.115-132.

5. КИКНАДЗЕ Д.Н. Метод анализа структуры параллельных алгоритмов, реализуемых на однородной вычислительной системе. -В кн.: Труды симпозиума "Вычислительные системы", Новосибирск, 1967, с.89-96.

6. ВЕНКОВА Н.Б., ТУШКИНА Т.А., ШАХБАЗЯН К.В. О расписании, минимизирующем число процессов однородной вычислительной системы. -"Техническая кибернетика", 1971, № 5, с. 115-117.

7. БАРСКИЙ А.Б. Минимизация числа вычислителей при реализации вычислительного процесса в заданное время. -"Техническая кибернетика", 1968, № 6, с.69-74.

8. RAMAMORTHY C.V., CHANDY K.M., GONZALEZ M.I. Optimal Scheduling Strategies in a Multiprocessor System. - "IEEE Trans. Computers", 1972, v.C-21, N 2, p.137-146.

9. МИРЕНКОВ Н.Н. Алгоритмы планирования для диспетчера однородной вычислительной системы. -В кн.: Вычислительные системы. Вып.42. Новосибирск, 1970, с. 34-46.

10. ХОРОШЕВСКИЙ В.Г., СЕДУХИНА Л.А. Стохастические алгоритмы функционирования однородных вычислительных систем. -В кн.: Вычислительные системы. Вып.51. Новосибирск, 1972, с.3-19.

11. КЕРБЕЛЬ В.Г., КОРНЕЕВ В.Д., МИРЕНКОВ Н.Н., ЩЕРБАКОВ Е.В. Управляющая программа системы МИНИМАКС. -В кн. Вычислительные системы. Вып. 60. Новосибирск, 1974, с. 129-143.

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