УДК 681.142.4

АЛГОРИТМЫ ПЛАНИРОВАНИЯ ДЛЯ ДИСПЕТЧЕРА ОДНОРОДНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

Н.Н. Миренков

Возможность использования монопрограммных режимов на отдельных подсистемах сделала одной из главных функций диспетчера однородной вычислительной системы (ОВС) умение разбивать систему на подсистемы в зависимости от поступающих требований [1]. Рассматривается ряд задач и алгоритмов, планирующих работу такого диспетчера.

Дадим определения.

Р а н г о м подсистемы назовем число элементарных машин (ЭМ), входящих в подсистему.

Задачу назовем k -го р а н г а, если она требует для своего решения подсистему такого же ранга.

Любое разбиение системы на подсистемы назовем ф у н к ц и о н а л ь н ы м с о с т о я н и е м системы.

З а д а ч а 1. ОВС из ℓ ЭМ работает в режиме большой очереди задач (требований). Очередь считается большой, если для её обслуживания необходимо заведомо большее время, чем некоторый рассматриваемый промежуток. Предполагается, что в очереди достаточно задач любого ранга и время решения каждой из них произвольное. Требуется, полностью загружая систему, обслуживать указанную очередь так, чтобы в течение промежутка времени T задачи всех рангов получили одинаковое системное время. Под системным временем понимается следующее: если группа задач k -го ранга решалась на системе в течение времени t_k, занимая одну подсистему, то считается, что она занимала всю систему в течение времени $\frac{t_k \cdot k}{\ell}$.

Решение поставленной задачи осуществим с помощью специального выбора последовательности функциональных состояний и времени работы системы в каждом из них.

Первую часть указанной последовательности можно представить в виде следующей операторной схемы:

$$A_\ell(T/\ell)\underbrace{A_\kappa^{\ell-\kappa}\left(\frac{T}{\ell-\kappa}\right)P_\kappa}_{\kappa<n}A_\alpha(T/\ell), \qquad (1)$$

где $A_\ell(T/\ell)$ – оператор состояния, при котором решаются задачи ранга ℓ в течение времени T/ℓ ;

$A_\kappa^{\ell-\kappa}\left(\frac{T}{\ell-\kappa}\right)$ – оператор состояния, при котором решаются задачи рангов κ и $\ell-\kappa$ в течение времени $T/(\ell-\kappa)$;

$A_\alpha(T/\ell)=A_{\ell/2}(T/\ell)$ для четного ℓ ;

$A_\alpha(T/\ell)=\Lambda$ – в противном случае;

P_κ – оператор условного перехода, $n=\left[\frac{\ell}{2}\right]$ для нечетного ℓ и $n=\frac{\ell}{2}-1$ – для четного ℓ .

После реализации (1) все задачи с рангами $\ell-\kappa \geq \frac{\ell}{2}$ исчерпают свое время, задачи же с рангами $\kappa < \frac{\ell}{2}$ будут еще иметь квоты на системное время, равные $\frac{T(\ell-2\kappa)}{(\ell-\kappa)\cdot\ell}$.

A – операторы в (1) могут быть, вообще говоря, произвольно переставлены. Однако обслуживание задач по уменьшению ранга является более удобным для перехода от одного функционального состояния к другому.

Вторую половину общей последовательности будем заполнять состояниями, реализация которых также по очереди обслуживает самые большие задачи из оставшихся, т.е. состояние, при котором обслуживаются задачи κ -го ранга, имеет в своем составе максимально возможное число подсистем κ -го ранга при наименьшем общем числе подсистем. Функционирование системы в таком состоянии продолжается до тех пор, пока не кончится квота на время у задач с максимальным рангом. Так, при четном ℓ первое состояние во второй половине общей последовательности содержит

две подсистемы ранга $(\ell-2)/2$ и одну подсистему ранга 2. Функционирование его будет продолжаться в течение $\frac{4T}{\ell^2-4}$, после чего задачи ранга $\frac{\ell-2}{2}$ будут обслужены, а у задач ранга 2 останется квота величиною в $\frac{2T}{\ell}\left(\frac{1}{2}-\frac{1}{\ell-2}-\frac{4}{\ell^2-4}\right)$.

Предполагается, что если при смене функциональных состояний имеются прерванные задачи, то соответствующая информация запоминается.

На рис.1 приведена последовательность рассматриваемых состояний $\{a_\kappa\}$ и время их работы для систем из 6 машин.

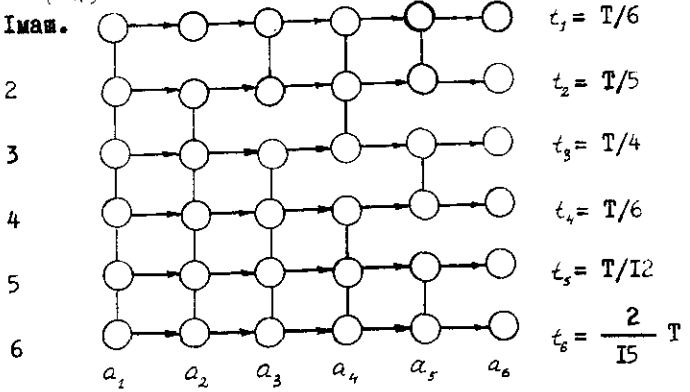


Рис.I.

Остается показать, что при реализации нашего алгоритма задачи некоторого ранга не могут получить свое время раньше задач с большими рангами.

Предположим противное, т.е. пусть имеются задачи κ-го ранга, которые исчерпывают свое время, по крайней мере, раньше задач ранга $\kappa+1$. Тогда они в любом из состояний занимают не больше одной подсистемы, а число состояний, в которых они могут встретиться, ограничено величиной $m=\frac{\ell-\kappa}{\kappa+1}$. Эти состояния представляют собой такие разбиения системы, в которых задачи ранга, большего, чем κ, занимают по I,2,3,..., m подсистем. Поэтому время, которое получат задачи κ-го ранга, даже если они присутствуют во всех этих состояниях, меньше

$$\tau=\frac{T}{\ell-\kappa}+\frac{T}{2}\left(\frac{2}{\ell-\kappa}-\frac{2}{\ell+\kappa}\right)+\frac{T}{3}\left(\frac{3}{\ell-\kappa}-\frac{3}{2\ell+\kappa}\right)+\dots$$
$$\dots+\frac{T}{m}\left(\frac{m}{\ell-\kappa}-\frac{m}{(m-1)\ell+\kappa}\right)=T\left(\frac{1}{\kappa+1}-\sum_{n=1}^{m-1}\frac{1}{n\ell+\kappa}\right)<\frac{T}{\kappa}, \qquad (2)$$

что доказывает наше утверждение. Следовательно, алгоритм обеспечивает полную загрузку системы.

П р и м е ч а н и е. Рассмотренный алгоритм без изменения можно использовать для некоторых других типов обслуживания большой очереди, в частности, если задачам разных рангов предоставляются квоты на системное время, равные $T\cdot\rho(\kappa)$, где

$$\sum_{\kappa=1}^{\ell}\rho(\kappa)=1 \quad \text{и} \quad \rho(\kappa)\geq 0$$

для всех κ от $\frac{\ell}{\kappa}-n$ до $\frac{\ell}{2}+n$ $(n=0,1,\dots,\frac{\ell}{2}-1)$; $\rho(\kappa)$, $\kappa<\frac{\ell}{2}-n$ и $\kappa>\frac{\ell}{2}+n$ – невозрастающая функция для , причем

$$\frac{\rho\left(\frac{\ell}{2}-n-1\right)}{\frac{\ell}{2}-n-1}\geq\frac{\rho\left(\frac{\ell}{2}+n+1\right)}{\frac{\ell}{2}+n+1}. \qquad (3)$$

З а д а ч а 2. На обслуживание в ОВС поступило N задач. Известно время решения t_i и ранг $\mathcal{R}_i$ каждой. Требуется распределить их по машинам и во времени так, чтобы весь набор был обслужен в минимально возможный срок.

Эта задача может быть решена методами целочисленного программирования.

Введем переменные:

$$x_{ij}=\begin{cases}1, \text{если } i\text{-я задача начинает обслуживаться в } j\text{-й момент;}\\ 0 \text{ – в противном случае,}\end{cases}$$

$i=1,2,\dots,N+1$, $N+1$ означает номер дополнительной задачи ℓ-го ранга, которая "будет ставиться" всегда последней. Тогда необходимо найти такие x_{ij}, которые бы минимизировали функцию

$$z=\sum_{\kappa=2}^{\bar{\kappa}}(\kappa-1)\,x_{N+1,\kappa} \qquad (4)$$

при следующих ограничениях:

$$\sum_{\kappa=2}^{\bar{\kappa}}(\kappa-1)x_{N+1,\kappa}\geq\sum_{\kappa=1}^{\bar{\kappa}}(\kappa-1+t_i)\,x_{i\kappa},\quad i=1,2,\dots,N,$$
$$\sum_{m=1}^{K}\sum_{i=1}^{N+1}x_{im}\,\mathcal{R}_i\,\delta_{i\kappa}^{m}\leq\ell, \qquad (5)$$

где $\delta_{ik}^{m} = \begin{cases} 0, & \text{если } m + t_i \le \kappa, \\ 1, & \text{если } m + t_i > \kappa, \end{cases} \quad \kappa \le \bar{\kappa},$

$\sum_{j=1}^{\bar{\kappa}} x_{ij} = 1, \quad i = 1,2,\dots,N+1,$

$\bar{\kappa}$ – момент времени, до которого заведомо закончатся все задачи.

Объем вычислительной работы при этом велик, поэтому оперативное диспетчирование не всегда будет возможным, хотя громоздкость задачи и уменьшается при объединении требований одинакового ранга (число их тогда будет порядка числа машин в системе). Но обычно нет необходимости учитывать любые наборы, т.к. диспетчер имеет дело только с теми, которые специфичны для работы каждой конкретной ОВС. Например, в системе "ILLIAC IV"[2] по закрепленным в её структуре правилам задачам разрешается запрашивать для своей реализации четверть, половину или всю систему. Поэтому рассмотрим решение нашей задачи, наложив некоторые ограничения на структуру поступающих наборов. Другой подход рассмотрен в [3].

Статистический анализ говорит о том, что отладочные задачи занимают порядка 30–35% [4] от общего времени работы ЭВМ. Так как программы для ОВС в основном отлаживаются на одной или нескольких машинах, то естественно предположить, что наиболее вероятными будут наборы с большим процентным содержанием задач, запрашивающих всю или почти всю систему, и задач, запрашивающих несколько машин.

Тогда алгоритм планирования будет следующим. Как и в задаче I, выберем последовательность функциональных состояний, реализация которых по очереди обслуживает самые большие из остающихся каждый раз задач или стремится к этому. Состояние, обслуживающее, в основном, задачи κ – го ранга, имеет в своем составе максимально возможное число подсистем κ – го ранга при наименьшем общем числе подсистем. Функционирование системы в таком состоянии продолжается до тех пор, пока в наборе не остается задач κ – го ранга меньше, чем максимально возможное число подсистем этого ранга в одном состоянии. После чего смена функциональных состояний происходит при окончании решения каждой из оставшихся задач κ – го ранга. Эти состояния

отличаются тем, что у них разное число подсистем упомянутого ранга и разное общее число подсистем. После того, как все задачи κ – го ранга решены, ведущее положение при реализации в следующих состояниях занимают задачи максимального ранга из оставшихся.

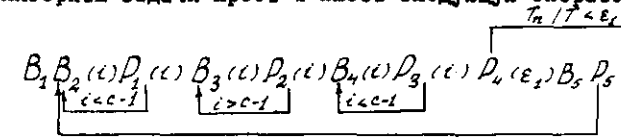
В постановке предполагалось, что время решения каждой задачи известно. Однако на практике это далеко не так. Поэтому в рассмотренном алгоритме явно учитывалось только суммарное время решения всех задач каждого ранга. Это связано с тем, что общее время решения большой группы задач оценить значительно проще, чем одной. К тому же суммарное системное время для задач разных рангов может быть просто регламентировано заранее в зависимости от того или иного способа использования ОВС. Поэтому далее исследуется более общая

З а д а ч а 3. Спланировать реализацию заданного набора, если известно только суммарное время решения задач каждого ранга. При этом на структуру набора не накладывается никаких ограничений.

Решение проведем в три этапа: на первом преобразуем заданный набор задач в набор укрупненных задач с одинаковым временем решения (задача 3а); на втором – для нового набора спланируем реализацию на системе (задача 3б); наконец, по результатам обоих этапов выработаем стратегию реализации для первоначального набора.

З а д а ч а 3а. Имеется набор задач, и известно суммарное время решения задач каждого ранга (T_i , $i \le \ell$). Необходимо найти такой наибольший промежуток времени τ, чтобы все T_i с достаточной точностью могли выражаться целым числом этих промежутков. Это и определит нам структуру нового набора.

Алгоритм задачи прост и имеет следующую операторную схему:



где c есть число различных рангов в наборе;

B_1 вычисляет $\tau = \min T_i$;

$B_2(i)$ вычисляет $\alpha_i = \frac{T_i}{\tau}$ и $\beta_i = [\alpha_i]$;

$B_3(\tau)$ присваивает t_i значение нуль, если $\alpha_i = \beta_i$, и $(\beta_i + 1)\tau - T_i$ – в противном случае;

B_4 вычисляет $T_\pi = \sum_i t_i R_i$ и $T = \sum_i T_i R_i$;

$P_4(\varepsilon_1)$ проверяет условие $T_\pi / T < \varepsilon_1$;

B_5 вычисляет $\tau = \tau / \kappa$, где $\kappa > 1$;

P_1, P_2, P_3 – операторы условного перехода;

P_5 – оператор безусловного перехода;

$Я$ – оператор конца.

Однако $\tau = \min_i T_i$ может оказаться много меньше других T_i, что в дальнейшем при реализации будет невыгодно. Поэтому задаемся дополнительными константами ε_2, ε_3 и анализируем T_i в следующих условиях: если $T_i < \varepsilon_2$, то соответствующие задачи исключаем из рассмотрения, проверяя при этом, чтобы величина $\sum_i T_i R_i$, где суммирование – только по исключенным задачам, не превосходила $\varepsilon_3 T$. Исключенные задачи обрабатываются при втором рассмотрении.

ε_1 характеризует простои машин, связанные с укрупнением задач. Очевидно, чем меньше ε_1, тем точнее весь алгоритм. Но существующие методы дают простои около 5% при реализации распараллеленного алгоритма. Поэтому нет смысла задавать $\varepsilon_1 < 0{,}05$.

ε_2 играет роль выделителя и должна задаваться так, чтобы за время τ могло решиться, по крайней мере, несколько задач из первоначального набора. Так как известны T_i, то известно и число задач, дающих вклад в T_i. Поэтому всегда можно определить среднее время решения задач в каждом наборе, среднее значение T_i и в зависимости от них задать ε_2.

ε_3 характеризует объем вычислительной работы, приходящейся на задачи, пропущенные при первом рассмотрении, и, очевидно, не должна превышать 0,1–0,2.

Рассмотренный алгоритм реализован в виде α-программы. Проведен ряд численных экспериментов, рассматривающих преобразования для больших наборов задач, предназначенных к реализации на системах высокой производительности. Результат: алгоритм достаточно эффективен и не требует больших затрат машинного времени (один вариант на ЭВМ "М-20" < 10 сек.).

З а д а ч а 3б. Имеется набор задач с одинаковым временем решения. Известно число задач каждого ранга. Необходимо составить расписание реализации этого набора, обеспечивая высокую

загрузку системы. Эту задачу можно решить методами целочисленного программирования. Однако при этом возникает ряд трудностей. Поэтому рассмотрим некоторый приближенный метод.

Составление нашего расписания тождественно упаковке достаточно высокого прямоугольника прямоугольниками с различными основаниями, но одинаковыми для всех высотами.

Упаковка производится по временным слоям. Для подбора группы, укладывающейся в один слой, производится просмотр имеющихся прямоугольников, но перебираются не все возможные ситуации, а только те, которые подчиняются определенным правилам:

1) прямоугольники с единичными основаниями не участвуют в упаковке (они используются на самом последнем этапе для заполнения "пустот");

2) прямоугольники с большими основаниями являются более приоритетными для постановки в группу;

3) для улучшения упаковки, произведенной с учетом предыдущего правила, разрешается нарушать его, пропуская некоторые прямоугольники;

4) после того, как сработало исключение, содержащееся в п.3, оставшиеся прямоугольники добираются в ту же группу с учетом правила 2.

Итак, упаковываем первый слой согласно второму правилу и запоминаем качество упаковки τ_1 (число машин, простаивающих на данном временном слое). Пусть в эту упаковку попали задачи с рангами $R_{k_1}, R_{k_2}, \ldots, R_{k_\alpha}$. Считая, что задачи с рангом R_{k_2} отсутствовали в начальном наборе, производим переупаковку и запоминаем τ_2. Если в наборе после первой упаковки ещё оставались задачи с рангами R_{k_2}, то они также не принимают участия при переупаковке. Далее производим аналогичные переупаковки, пропуская по очереди задачи с рангами $R_{k_3}, R_{k_4}, \ldots, R_{k_\alpha}$, считая что все R_{k_i} $(i = 3, 4, \ldots, \alpha)$ больше двух (если есть равные двум, то над ними указанные действия не производятся).

В итоге мы получим ряд упаковок – направлений (их число $\leq \alpha$), из которых выберем m лучших, согласно $\{\tau_i\}$. Затем для каждого из направлений упакуем следующий слой, в результате чего получится m^2 направлений. Выбрав из них m лучших, с учетом суммарного качества упаковок на всех предыдущих слоях, переходим на следующий слой и т.д., пока в каком-нибудь из напра-

влений не будут исчерпаны все задачи набора. В табл. I показаны возможные, согласно нашему рассмотрению, упаковки прямоугольника с основанием 20 прямоугольниками с основаниями: I5, I2, 8, 7, 5, 4, 4, 3, 2 при $m = 3$.

Таблица I

I слой	→	2 слой	→	3 слой	$\sum_{i=1}^{3} t_i$	Направления
I5,5	I2,8 I2,7	I5,5;I2,8	7,4,4,3,2	I5,5;I2,8; 7,4,4,3,2	0	I
I5,3,2	I2,8 I2,7	I5,3,2;I2,8	7,5,4,4	I5,3,2; I2,8; 7,5,4,4	0	2
— —	— —	I5,5;I2,7	8,4,4,3	I5,5;I2,7; 8,4,4,3, (2)	2	3

Заметим, что при необходимости число сравниваемых переупаковок для каждого слоя (а это связано с точностью алгоритма) может быть увеличено за счет пропуска различных пар задач, троек и т.д.,а также за счет переупаковок второго уровня, т.е. переупаковок,применяемых к переупаковкам.

Возникает сразу вопрос, как часто нужно учитывать последнее замечание и насколько большой нужно задавать величину m. Была написана программа для ЭВМ "Минск-22", реализующая рассмотренный алгоритм, и проведено ряд численных экспериментов. Составлялись расписания всевозможных наборов с числом задач от 50 до 200 для реализации на системе из I00 машин. При этом вышеупомянутое замечание не учитывалось. Уже при $m = 5$ среднее отклонение качества составленных расписаний от качества оптимальных не превосходило 3%. Под отклонением здесь понималось отношение $\alpha = \frac{T - T_{опт}}{T_{опт}}$, где T – время, необходимое для реализации на системе всего набора по расписанию,составленному нашей программой.Время (τ) одного эксперимента на ЭВМ "Минск-22"

было около одной-двух минут. Необходимо отметить, что для наиболее вероятных структур наборов, ожидаемых в реальных условиях, получены наилучшие результаты. В табл.2 приведены характеристики некоторых типичных экспериментов.

По расписанию реализации укрупненных задач с одинаковым временем решения легко выработать стратегию реализации первоначального набора,т.к.теперь известна последовательность функциональных состояний. Стратегия состоит в том, что мы в течение одного временного слоя посылаем на подсистемы (какие именно, известно из составленного расписания) задачи одних и тех же рангов. Затем функциональное состояние системы меняется (в частном случае оно может оставаться тем же), и задачи других рангов обслуживаются в течение временного слоя. Незаконченные задачи при смене функциональных состояний прерываются и запоминаются до тех пор, пока соответствующим подсистемам не будет выделено новое время. Если задачи одного ранга занимают одновременно несколько подсистем и по расписанию должны полностью реализоваться,то могут возникнуть машинные простои, связанные с неравномерным распределением задач по подсистемам.В этом случае поступают двояко: если временный слой значительно больше среднего времени решения одной задачи, то простоями пренебрегаем; в противном случае – некоторые задачи прерываем и, скомпоновав с подобными, доканчиваем в заключение всего процесса реализации. Информацию о времени решения отдельных задач, хотя бы и приближенную, можно использовать для уменьшения указанного эффекта путем постановки задач с большим временем решения в первую очередь.

В заключение рассмотрим распределение задач одинаковых рангов по подсистемам, если известно время реализации каждой, и приведем решение, основанное на алгоритме задачи 3б.

З а д а ч а 3в. Имеется набор задач одинаковых рангов R. Известно время решения каждой (t_i , $i = 1,2,\ldots, N$). Необходимо спланировать их реализацию на системе (в частности,распределить по подсистемам).

Рассмотренный выше алгоритм упаковки использовал тот факт, что задачи требовали для своего решения одно и то же время, но разные подсистемы. Здесь же требуются одинаковые подсистемы,но разное время для решения. Так как для упаковки не важно, что в

Таблица 2

№	Процентное содержание задач различных рангов в наборе						α	t
	I	2-5	6-I0	II-20	2I-50	5I-I00	[%]	[сек]
I	—	-	33	27	I4	26	4,8	30
2	-	-	-	I6	75	9	4,8	25
3	-	-	-	-	83	I7	5,0	30
4	3	I5	7	-	I4	7I	0	40
5	-	-	-	I7	62	2I	5,5	65
6	6	-	7	7	73	7	0	I20
7	-	-	-	-	82	I8	7,2	75
8	-	4	I3	28	48	7	0	75
9	-	-	I2	28	47	7	0	I25
I0	-	2	6	22	45	9	2,3	I50
II	-	9	20	26	35	I0	0	I55
I2	I0	I6	24	2I	8	2I	0	35
I3	I5	I6	I7	24	5	23	0	35
I4	-	I2	5	I2	60	II	2,2	I65
I5	-	-	-	I4	73	I3	7,I	I20
I6	-	42	2I	I8	I6	3	0	I50
I7	3	23	25	24	3	22	0	60
I8	-	33	I5	I8	27	7	0	I50
I9	-	-	-	20	65	I5	3,7	35
20	-	-	-	I3	75	I2	6,4	I65
2I	50	-	25	I8	-	7	0	30
22	2I	20	I4	20	5	20	0	35
23	I5	2I	I4	25	5	20	0	35
24	-	-	30	27	I6	27	5,0	35
25	-	-	I0	28	34	28	5,I	35

основании прямоугольников: время или число машин, то их можно "повернуть" на 90^0. Остается найти "число машин" ℓ. Его определяем, исходя, например, из самой плотной упаковки:

$$\ell = \frac{R \sum_{i=1}^{N} t_i}{\ell} . \qquad (7)$$

Если через ℓ / R " временных слоев " не удалось упаковать все задачи из набора, то может быть два случая:

I) оставшиеся задачи требуют для своей реализации небольшое системное время (например, < 5-I0%), тогда их можно просто добавить в подсистемы с наихудшей внешней упаковкой;

2) оставшиеся задачи требуют существенную часть общего системного времени (>I0-20%), тогда для них можно повторить основной алгоритм, определив "число машин" согласно (7), с учетом только оставшихся задач.

Так как время реализации основного алгоритма даже на ЭВМ " Минск-22 " составляет порядка минуты, то имеется возможность перебрать ряд значений параметров, от которых зависит планирование работы. В частности, "число машин" можно выбирать не только по формуле (7), но и как некоторую функцию от ℓ .

Выводы

Алгоритмы, лежащие в основе решений поставленных задач, являются достаточно эффективными, не требуют больших затрат машинного времени и могут быть использованы при оперативном диспетчировании в ОВС.

Литература

I. Н.Н. МИРЕНКОВ. Некоторые вопросы организации вычислений на однородных ВС. - Материалы ко П-ой Всесоюзной конференции по вычислительным системам. Новосибирск, 1969.

2. G.H. BARNES, R.A. STOCKES, R.M. BROWN, D.T. KUCK, D.L.SLOTNICK. The ILLIAC IV computer. - IEEE Trans.Computers, C-17, Aug.1968, 746-757.

3. В.Г.ХОРОШЕВСКИЙ. Алгоритмы функционирования однородных УВС. – Вычислительные системы, Новосибирск, "Наука" СО, в печати.

4. С.П.СУРЖИКОВ, И.В.МАКСИМЕЙ. Организация сбора статистики о структурах задач с помощью автооператора для машин М-20. – Труды I Всесоюзной конференции по программирова – нию, Киев, 1967.

Поступила в редакцию
17 марта 1970 года