MODELING OF PRIORITYLESS PROCESSING IN AN INTERLEAVED MEMORY WITH A PERFECTLY INFORMED PROCESSOR

A. Iványi and I. Kátai

UDC 681.327.2:62-501.72

We consider a mathematical model of prioritylessqueue processing under complete information for the case of two queues with customers of two types. A method is proposed for designing a polynomial optimal algorithm. The service speed is introduced, its existence is proved, and some bounds for the case of equiprobable customers are given.

1. INTRODUCTION

In various applications, e.g., when modeling a paged virtual memory [1-3] or an interleaved computer memory [4, 5], it is useful to know the service speed of prioritylessqueues under complete information, since this speed characterizes the limiting possibilities of the service algorithms.

In this article we first formulate the problem and then prove the existence of a polynomial [6] algorithm to find the optimal service strategy. A measure of efficiency of the optimal algorithm is proposed (its speed) and some bounds on this speed are derived.

The problem arises in the analysis of interleaved computer memories [7], when the main memory consists of two blocks with two programs executing concurrently. If the current commands require accessing different memory blocks, they can be executed concurrently; if the commands access the same block, they are executed sequentially.

2. THE PROBLEM

Consider two natural numbers m, n and queues of customers of two types (0 and 1) $B_m = b_1 \dots b_m$ and $C_n = c_1 \dots c_n$, where $b_i \in \{0, 1\}$ and $c_j \in \{0, 1\}$ $(i = \overline{1, m}, j = \overline{1, n})$.

These queues are served under the following restrictions.

1. Service occurs at the moments 0, 1, 2, ... The duration of service is one time unit.
2. As long as there is at least one unserved customer, at least one customer is being served at each moment of time.
3. The moment when a customer x is served will be denoted by $\mu(x)$. The service is sequential, i.e., $1 \leqslant i < k \leqslant m$ implies that $\mu(b_i) \leqslant \mu(b_k)$ and $1 \leqslant j < l \leqslant n$ implies that $\mu(c_j) \leqslant \mu(c_l)$.
4. Customers of the same type cannot be served simultaneously, i.e., $b_i = b_k$, $b_i = c_j$, $c_j = c_l$ imply $\mu(b_j) \neq \mu(b_k)$, $\mu(b_i) = \mu(c_j)$, and $\mu(c_j) = \mu(c_l)$, respectively.

Constraints 1-4 imply that the following instances of service are possible in the first moment: a) b_1; b) c_1; c) b_1 and b_2 (only if $b_1 \neq b_2$); d) b_1 and c_1 (only if $c_1 \neq b_1$) e) c_1 and c_2 (only if $c_1 \neq c_2$). If the customer indexes 1, 2,... characterize the unserved customers, the same instances are possible at each moment in time.

Denote the service time of the queues B_m, C_n with the algorithm A by $\tau(B_m, C_n)$. Then the above conditions imply that $(m + n)/2 \leqslant \tau(B_m, C_n) \leqslant m + n$. An algorithm which for any pair B_m, C_n finds the minimum service time is regarded as optimal (it is denoted by OPT). The number of possible service strategies for a fixed pair B_m, C_n is finite: There are at most five instances of service at each moment and the service time does not exceed m + n, so that the number of possible strategies is at most 5^{m+n} and the minimum service time always can be determined (e.g., by enumerating all the possible strategies). This means that the algorithm OPT always exists.

Budapest, Hungary. Translated from Avtomatika i Telemekhanika, No. 4, pp. 129-136, April 1985. Original article submitted December 20, 1983.

0005-1179/85/4604-0520$09.50

From (17) and (9),

$$\frac{T(M_j, N_j)}{M_j+N_j} \leqslant (L+\delta)\frac{h_j(m_i+n_i)}{M_j+N_j} + \frac{h_j n_i|\delta_j| + \alpha r_j + h_j n_i|\varepsilon_i| + r_j|\delta_j| + n_i}{M_j+N_j}. \quad (18)$$

If now $j \to \infty$ in (18), we obtain

$$\frac{T(M_j, N_j)}{M_j+N_j} \to U, \quad \frac{h_j(m_i+n_i)}{M_j+N_j} \to 1, \quad \frac{\alpha r_j + r_j|\delta_j| + n_i}{M_j+N_j} \to 0,$$

$$\frac{h_j n_i|\delta_j|}{M_j+N_j} \leqslant |\delta_j| \to 0, \quad \frac{h_j n_i}{M_j+N_j} \leqslant |\varepsilon_i|.$$

If $i \to \infty$, then $\varepsilon_i \to 0$; since δ is arbitrarily small, this gives $U \leqslant L$. QED.

LITERATURE CITED

1. O. I. Aven, N. N. Gurin, and Ya. A. Kogan, Performance Assessment and Optimization of Computing Systems [in Russian], Nauka, Moscow (1982).
2. O. I. Aven and Ya. A. Kogan, Managing the Computational Processes [in Russian], Energiya, Moscow (1978).
3. G. G. Avetisyan, Minimization of Overhead Costs in the Realization of Shared-Resource Processes, Abstract of Thesis [in Russian], Inst. Kibern. Akad. Nauk UkrSSR, Kiev (1984).
4. A. M. Ivanyi, "Minimization of overhead costs for concurrent program execution," Programmirovanie, No. 1, 83-88 (1984).
5. A. Iványi and I. Kátai, "Processing of independent Markov Chains," Annales Univ. Sci. Budapest, Sectio Computatorica, 3, 33-46 (1982).
6. V. S. Mikhalevich and A. I. Kuksa, Methods of Sequential Optimization [in Russian], Nauka, Moscow (1983).
7. L. N. Korolev, Computer Structure and Software [in Russian], Nauka, Moscow (1978).
8. A. A. Korbut and Yu. Yu. Finkel'shtein, "Approximate methods of discrete programming," Izv. Akad. Nauk SSSR, Tekh. Kibern., No. 1, 165-176 (1983).
9. É. Z. Lyubimskii, V. V. Martynyuk, and N. P. Trifonov, Programming [in Russian], Nauka, Moscow (1980).

CHOOSING A MEMORY HIERARCHY CONFIGURATION

A. S. Yurchenko

UDC 681.327.2

An analytical model of the hierarchical memory in a modern computer is used to determine an optimal memory hierarchy configuration. It is shown that necessary and sufficient conditions are satisfied for minimizing the mean access time to the hierarchical memory.

Modern computer architecture provides multilevel hierarchical memories which allow the central processor to access each of the memory levels. The computer throughput in these cases is largely determined by the memory speed, i.e., the mean time to process a single access to memory. The mean access time depends on the size of each level in the memory hierarchy, on page size, and on the memory access cycle in each level of the hierarchy. Modern computers manipulating large data files require memory devices with different speeds and correspondingly different costs. Such hierarchical memories are very costly to install, and it is very important to assess their speeds already in the development stage. A number of recent studies have assessed the efficiency of multilevel memories [1-6].

In this article, we propose a solution of the problem formulated in [5], viz., choosing an optimal configuration of a hierarchical memory operating under the control of one processor.

Kiev. Translated from Avtomatika i Telemekhanika, No. 4, pp. 137-139, April, 1985. Original article submitted November 28, 1983.

0005-1179/85/4604-0526$09.50

As m and n increase, the number of possible service strategies increases very rapidly and enumerative algorithms become unpracticable [8]. Therefore we must determine whether the problem of finding the minimum service time and the optimal service strategy is NP-complete [6, 8] and try to design a polynomial algorithm to solve this problem.

Let $\{\xi_i,\ i = \overline{1, m}\}$ and $\{\eta_j,\ j = \overline{1, n}\}$ be sequences of independent random variables which take the values 0 and 1 with equal probabilities. Let $T_A(m, n)$ be the mathematical expectation of the service time of the pair of queues (ξ_i, η_j) by the algorithm A. We denote the ratio $T_A(m, n)/(m + n)$ by $t_A(m, n)$. This value may be interpreted as the mean time to serve one customer from queues of length m and n.

In order to avoid using the queue lengths as parameters, we define for a nonnegative real α

$$\varphi_A(\alpha) = \lim_{\substack{y\to\infty \\ x/y\to\alpha}} t_A(x, y).$$

This $\varphi_A(\alpha)$ is called the speed of the algorithm A for relative queue length α. The speed may be interpreted as the mean time to serve one customer in long queues with a given length ratio α. We will show that $\varphi_{OPT}(\alpha)$ exists for all $\alpha \geqslant 0$ and obtain a bound for this quantity.

3. DESIGN AND ANALYSIS OF A POLYNOMIAL OPTIMAL ALGORITHM

Consider the following algorithm DIN, which associates with every possible service state some point in the Cartesian space; conversely, the point (i, j) ($i = \overline{1, m}$; $j = \overline{1, n}$) corresponds to a state in which $b_1,\ldots,b_i$, $c_1,\ldots,c_j$ have been served while $b_{i+1},\ldots,b_m$, $c_{j+1},\ldots,c_n$ await service.

Step 1. Define a Cartesian coordinate system; to integer points on the x axis assign the customers b_i ($i = \overline{1, m}$) and to integer points on the y axis assign the customers c_j ($j = \overline{1, n}$). Let $L = 0$.

Step 2. To the point (0, 0) assign the service time $\tau_{DIN}(0, 0) = 0$.

Step 3. Repeat step 4 until the point (m, n) is assigned the service time $\tau(B_m, C_n)$.

Step 4. Suppose that some points in the rectangle π with the services (0, 0), (m, 0), (0, n), and (m, n) have already been assigned the corresponding service times 0, 1,.... Then points with service time ν are determined by the following procedure.

The points (i, 0), (i, 1),...,(i, n) are called the column i. In all the columns 0, 1,...,L sequentially find the top point P which has the service time ν. Denote the point above P by Q and the point above Q by R. Denote the point to the right of P by S, the point above S by U, and finally the point to the right of S by T (Fig. 1).

A customer corresponding to the point X in B_m and C_n will be denoted, respectively, by B(X) and C(X); $\tau(X)$ is the service time of the point X, ρ is the set of points from π which so far have not been assigned service times. The following relationships hold:

a) if $Q \in \rho$, then $\tau(Q) := \nu+1$,

b) if $S \in \rho$, then $\tau(S) := \nu+1$,

c) if $R \in \rho$ and $C(Q) \neq C(R)$, then $\tau(R) := \nu+1$,

d) if $U \in \rho$ and $C(Q) \neq B(S)$, then $\tau(U) := \nu+1$,

e) if $T \in \rho$ and $B(S) \neq B(T)$, then $\tau(T) := \nu+1$,

f) if $S \in \pi$, then $L := L + 1$,

g) if $T \in \pi$ and $B(S) \neq B(T)$, then $L := L + 1$.

For example, for $B_6 = 101000$ and $C_4 = 1110$ we obtain Fig. 2, which shows that $\tau(B_6, C_4) = 5$. For this algorithm we have the following propositions.

THEOREM 1. For all pairs of queues B_m, C_n we have $\tau_{OPT}(B_m, C_n) = \tau_{DIN}(B_m, C_n)$ and the algorithm DIN is polynomial.

THEOREM 2. If $\alpha \geqslant 0$, the limit $\lim_{\substack{y\to\infty \\ x/y\to\alpha}} \varphi_{OPT}(x)$

TABLE 1

			A	B	C	D	E	F	G	H	I
A	3:3		5/16	3/16	1/8	1/16	1/32	1/16	1/8	1/32	1/16
B	2:2	X X	1/4	1/8	–	–	1/16	1/8	1/4	1/16	1/8
C	1:1	XX XX	1/4	–	–	–	–	–	1/2	–	1/4
D	1:1	XX XX	–	–	–	–	1/4	1/2	–	1/4	–
E	2:0	XX XX	–	–	–	–	1/4	1/2	–	1/4	–
F	2:0	XX XY	1/4	–	–	–	–	–	1/2	–	1/4
G	2:0	XY XY	1/2	1/2	–	–	–	–	–	–	–
H	2:0	XY XY	1/2	1/2	–	–	–	–	–	–	–
I	2:0	XX YX	1/4	–	1/2	1/4	–	–	–	–	–

H: 27/668, I: 54/668. These stationary probabilities can be used to determine γ.

The third class includes "alternative" algorithms $\{ALT\,k\}_{k=1}^{\infty}$ with memory consisting of $k-1$ pairs of locations and a counter. The algorithm also examines k pairs each time and services two elements if possible. If no two customers can be served, one customer is served so as to ensure that two customers will be served soon thereafter.

Let α_t and β_t denote the number of served customers in t time units for the first and the second queue, respectively. Then the contents of the counter at time moment $(t-1)$ will be defined as $\alpha_t - \beta_t$.

If these rules do not uniquely identify the queue from which one (or two) customers can be served, we try to serve the queues "uniformly" (so as to minimize $|\alpha_t - \beta_t|$). When $\alpha_t - \beta_t = 0$, the first queue is served.

If at least one of the queues contains fewer than k elements, the algorithm switches to one-by-one service. In the extreme case, if there remain $n + k - 1$ elements for one-by-one service, they are ignored in the computation of γ and we focus on the "main part" of the service procedure, which is again described by a homogeneous Markov chain.

For ALT1, the chain has two states, A and B. In state A different elements are examined and both are served in one time unit. In state B identical elements are examined; one of them (selected by the counter) is served and the contents of the counter is incremented by 1 (if it was negative) or decremented by 1 (if it was nonnegative). The transition probability matrix contains the elements 1/2; the solution of the system of equations is $p(A) = 1/2$, $p(B) = 1/2$. Hence we obtain $\gamma_{ALT1} = 2/3$.

For ALT2, the chain has 5 states. In state A identical elements occupy the second places and different elements occupy the first places. In state A both the first and the second pair

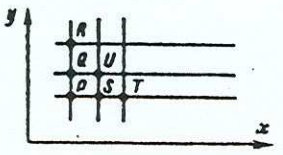

Fig. 1. The configuration of points.

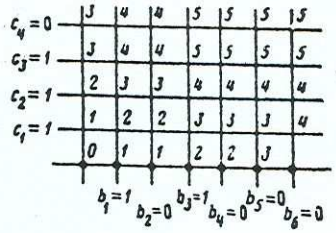

Fig. 2. Application of the algorithm.

exists and depends only on α.

4. SPEED BOUNDS

For any pair of queues B_m, C_n we have the inequality

$$\tau(B_m, C_n) \leqslant \tau(B_m, \phi) + n,$$

so that $T(m, n) = T(m, 0) + O(n)$, and for $\alpha = 0$ we obtain $t(m, n) = t(m, 0)$. Therefore for $\alpha = 0$ it suffices to consider the service of one queue only. We get $\varphi(0) = 2/3$ and the trivial bounds

$$1/2 \leqslant \varphi_{OPT}(\alpha) \leqslant 2/3.$$

In what follows we assume that $\alpha = 1$. Let $\varphi_A(1) = \gamma_A$.

We will consider three classes of approximate algorithms. The efficiency of these algorithms with the increase of their parameter k goes to γ_{OPT}.

The first class includes the "simple" algorithms $\{SIMk\}_{k=1}^{\infty}$. The algorithm SIMk serves the first k pairs of customers $(b_1, c_1, b_2, c_2, \ldots, b_k, c_k)$ in an optimal fashion, then the next k pairs, and so on [n/k] times. If at the end of the queues there remain Q pairs $(0 < Q < k)$, then for simplicity these customers are served one by one, i.e., in 2Q time units.

It is easily seen that

$$t_{SIM\,1}(n, n) = 3/4 = 0.75;$$

$$t_{SIM\,2}(n, n) = \frac{4[n/2]11/16 + 2(n - 2[n/2])}{2n};$$

$$t_{SIM\,3}(n, n) = \frac{6[n/3]21/32 + 2(n - 3[n/3])}{2n}$$

Already with four pairs the service time of a group of pairs may be greater than the number of zeros and ones in the group. Therefore from [4] we have

$$t_{SIM\,4}(n, n) = \frac{8[n/4]655/1024 + 2(n - 4[n/4])}{2n}$$

From these equalities it follows that $\gamma_{SIM1} = 3/4 = 0.75$; $\gamma_{SIM2} = 11/16 \approx 0.678$; $\gamma_{SIM3} = 21/32 \approx 0.656$; $\gamma_{SIM4} = 655/1024 = 0.639$. The second class includes algorithms with memory consisting of $k - 1$ pairs of locations. The algorithm MEMk $(k = 1, 2, \ldots)$ examines k pairs and serves as many pairs as can be served optimally. If none of the pairs can be served optimally, then the first pair is served in two time units. The unserved pairs are stored in the memory, augmented with new pairs, and k pairs are again examined. If fewer than k pairs remain, they are served one by one.

In computing γ, we can ignore the individually served customers, which gives $\gamma_{MEM1} = 3/4 = 0.75$; $\gamma_{MEM2} = 13/20 = 0.65$; $\gamma_{MEM3} = 371/594 \approx 0.624$.

For example, the last result was observed in the following way. The execution of the algorithm MEM3 is described by a homogeneous Markov chain with 9 states. In state A all the 6 elements are served; in state B, 4 elements are served, and in the other states 2 elements are served. In states C and D different customers are served in one unit of time, and in states E, F, G, H, I identical elements are served in two units of time. The transition probabilities for the algorithm MEM3 are given in Table 3.

Solving the corresponding system of linear equations, we obtain the following stationary probabilities: A: 200/668, B: 120/668, C: 52/668, D: 26/668, E: 27/668, F: 54/668, G: 108/668,

TABLE 2

			A	B	C	D	E
A	1:1	2:0	–	–	2/4	1/4	1/4
B	1:1	1:1	2/4	2/4	–	–	–
C	2:0	1:1	1/4	1/4	1/4	–	1/4
D	2:0	0:2	1/4	1/4	1/4	1/4	–
E	2:0	2:0	–	–	1/2	–	1/2

consist of different elements; in state C one of the queues starts with different elements, while the other starts with identical elements, and the first elements are identical; in state D both queues start with different elements, and the first elements are identical; in state E all the elements are identical. Table 2 presents the transition probabilities.

The stationary state probabilities are the following: A: 3/16; B: 3/16; C: 5/16, D: 1/4, E: 1/4. Hence $\gamma_{ALT2} = 4/7 \approx 0.572$, which leads to the following bound.

THEOREM 3. $1/2 \leqslant \gamma_{OPT}(1) \leqslant 4/7$.

Note that for ALT3 we have 15 states and $\gamma_{ALT3} = 9/16 \approx 0.567$.

Our results may be interpreted in the following way. The criterion φ in the form $1/\varphi$ represents a certain generalized "speed" of computer processing [9] expressed, say, in natural units of "number of executed operations per second." If we can show that $\gamma_{OPT}(1) = 1/2$, this would imply that increasing the number of memory blocks from one to two doubles the processing speed, whereas for $\varphi_{OPT}(1) > 1/2$, the speed will increase sublinearly.

APPENDIX

Proof of Theorem 1. Optimality of the algorithm DIN is proved by induction. Since step 4 examines all the 5 possible cases, ν is assigned to all the points with unit minimal service time. Now assume that ν has been assigned to all the points for which the minimal service time is ν (or at least to one point above these points). Since Step 4 examines all the 5 possible instances of service, $\nu + 1$ is assigned to all the points for which the minimal service time is $\nu + 1$ [if there are still points with ν under a point type P, it is possible that the service time $\nu + 1$ will not be assigned to some points under S, e.g., the point (6, 0) in Fig. 2, but these points do not affect $\tau(m, n)$].

Note that although the algorithm DIN determines the minimum service time, it does not provide a strategy for attaining this minimum. If in Step 4, together with the service time $\nu + 1$ we also assign the coordinates of the point P to the points Q, R, S, T, U, then having computed τ we can use the point (m, n) to reconstruct a single optimal strategy, by examining only $\tau_{DIN}(m, n)$ points.

Since the number of columns examined in each unit of time is at most doubled, in the ν-th unit of time we examine at most $2\nu + 1$ columns, i.e., at most $5(2\nu + 1)$ points. Since for every pair $\tau(B_m, C_n)$ for any algorithm A we have $\tau(B_m, C_n) \leqslant m + n$, the determination of $\tau(B_m, C_n)$ involves examining at most $\sum_{\nu=1}^{m+n} 5(2\nu+1) = 5(m+n)(m+n+2)$ points, i.e., if the size of our problem is characterized as $m + n = s$ and the complexity A(s) of the algorithm A is defined as the maximum number of points combined, we have $DIN(s) \leqslant 5s(s + 2)$.

Proof of Theorem 2. If $\alpha = 0$, the effect of the first queue may be ignored and the existence of the limit is directly obtained for the second queue. For $\alpha > 0$, the proof is based on the following inequalities:

a) if k is a natural number, then

$$T(k, m, kn) \leq kT(m, n); \tag{1}$$

b) if a and b are nonnegative integers, then

$$0 \leq T(m+a, n+v) \leq T(m, n)+a+b. \tag{2}$$

Since the inequality $0.5 \leqslant \tau(B_m, C_n)/(m + n) \leqslant 1$ holds for all possible pairs B_m, C_n, $t(x, y)$ is bounded from above and from below, i.e., finite U and L exist in the form

$$L = \lim_{\substack{y\to\infty \\ x/y\to\alpha}} t(x, y), \tag{3}$$

$$U = \overline{\lim_{\substack{y\to\infty \\ x/y\to\alpha}}} t(x, y). \tag{4}$$

Let $U \neq L$, i.e., $U > L$. We will show that $U \leqslant L$, so that $U = L$.

By (3), there exist sequences m_i $(i = 1, 2, \ldots)$ and n_i $(i = 1, 2, \ldots)$ such that

$$\lim_{i\to\infty} m_i/n_i = \alpha \quad \text{and} \quad \lim_{i\to\infty} t(m_i, n_i) = L. \tag{5}$$

By (4), there exist sequences M_j $(j = 1, 2, \ldots)$ and N_j $(j = 1, 2, \ldots)$ such that

$$\lim_{j\to\infty} M_j/N_j = \alpha \quad \text{and} \quad \lim_{j\to\infty} t(M_j, N_j) = U. \tag{6}$$

By (5), there exists a sequence ε_i $(i = 1, 2, \ldots)$ such that

$$m_i = n_i(\alpha+\varepsilon_i) \quad \text{and} \quad \lim_{i\to\infty} \varepsilon_i = 0. \tag{7}$$

By (6), there exists a sequence δ_j $(j = 1, 2, \ldots)$ such that

$$M_j = N_j(\alpha+\delta_j) \quad \text{and} \quad \lim_{j\to\infty} \delta_j = 0. \tag{8}$$

By (3), for every $\delta > 0$ there is a threshold z such that

$$\frac{T(m_i, n_i)}{m_i+n_i} < L+\delta \qquad (i = z, z+1, \ldots). \tag{9}$$

Now let n_i be a fixed number such that (9) holds for n_i, n_{i+1}. Then every N_j is uniquely representable in the form

$$N_j = h_j n_i + r_j \quad (0 \leq r_j < n_i). \tag{10}$$

Let

$$N_j^* = h_j n_i. \tag{11}$$

$$M_j^* = h_j m_i. \tag{12}$$

Now consider $T(M_j, N_j)/(M_j + N_j)$. By (2) we obtain

$$T(M_j, N_j) \leq T(M_j^*, N_j^*)+|M_j-M_j^*|+|N_j-N_j^*|, \tag{13}$$

where from (10) and (11) we have

$$|N_j-N_j^*| = r_j < n_i, \tag{14}$$

and from (12), (7), (10), and (8) we have

$$|M_j-M_j^*| \leq h_j n_i|\delta_j| +\alpha r_j + h_j n_i|\varepsilon_i| + r_j|\delta_j|. \tag{15}$$

From (11), (12), and (1) we obtain

$$T(M_j^*, N_j^*) = T(n_j m_i, h_j n_i) \leq T(m_i, n_i). \tag{16}$$

From (13), (14), (15), and (16) we get

$$\frac{T(M_j, N_j)}{M_j+N_j} < T(m_i, n_i)\frac{h_j}{M_j+N_j} + \frac{h_j n_i|\delta_j| +\alpha r_j + h_j n_i|\varepsilon_i| + r_j|\delta_j| + n_i}{M_j+N_j}. \tag{17}$$