Work-Scheduling Algorithms: A Nonprobabilistic Queuing Study (with Possible Application to No. 1 ESS)

By JOSEPH B. KRUSKAL

(Manuscript received November 7, 1968)

In many large computer systems with real-time use (such as the No. 1 Electronic Switching System), the central processing unit handles much of its work through queues. It may spend much of its time cycling through the queues, performing the work requests it finds there. To accomodate varying degrees of urgency, the cycle may visit some hoppers more often than others. (No. 1 ESS strongly relies on this procedure.) This paper provides an approximate method for evaluating different cycles.

Using the evaluation method and some approximations, we obtain a formula for the optimum relative frequency with which different queues should be visited.

The model used is nonprobabilistic, and treats requests as continuous rather than discrete. The model also ignores certain interdependencies between queues. Despite these drastic simplifications, the results probably provide useful guidance, if interpreted cautiously.

I. INTRODUCTION

In many large computer systems, especially those with real-time use, the central processor handles much of its work through queues, which contain work requests. (The queues may also be called hoppers, buffers, waiting lines, files, and so forth. In this paper we call them hoppers.) The processor examines each hopper in turn, and performs some or all of the work requests if any, which it finds there.

Some work requests require processing more urgently than others. One method of providing appropriate response times is to examine more frequently hoppers which contain urgent work, and other hoppers less frequently. For example, the No. 1 ESS (Electronic Switching

System) has many hoppers which it groups into five different urgency classes.[1,2] The five classes are examined (or "visited") in a fixed recurring cycle, of length 30, during which the classes are visited 15, 8, 4, 2, and 1 times, respectively. (During a single visit to a single class, the individual hoppers are visited once each, in a fixed sequence.)

This paper contains a practical approximate model for evaluating various alternative cycles. The conceptual basis for the evaluation is the expected time each work request must wait in the hopper before being serviced by the central processor. (Such times depend not only on the cycle, but also on the times required to process requests, and on the rates at which new requests are initiated. These are all assumed given.) The expected waiting times for different hoppers are multiplied by frequencies and also by weights w_i , the "average penalty per second of delay," and added. The resulting sum is called P, the "expected total penalty per second." The weights w_i , which reflect the relative importance of delaying different work requests, are assumed given, and we seek to minimize P by choosing the cycle wisely. By way of illustration, the calculations required to evaluate any given cycle are given for two very simple cycles.

When applied to general cycles, our model yields the plausible conclusion that visits to the same hopper should be spaced as evenly as possible around the cycle (in terms of elapsed time between visits). Furthermore, the model permits us to estimate how sensitive P is to deviations from this ideal.

Our most important conclusion is an explicit formula for how frequently each hopper should be visited. To obtain this formula, we assume that visits to each hopper are evenly spaced around the cycle. Then P becomes a function of the visit frequency (and not of detailed visit pattern). We explicitly optimize this function, to obtain a formula for visit frequencies.

The time required to examine a hopper, whether or not it contains any work requests, is small but highly significant, and is an important consideration in the problem. Our model explicitly reflects this fact. (Indeed, it is known though sometimes overlooked that the No. 1 ESS central processor finds most hoppers empty on a majority of its visits, even when it is heavily loaded with work and operating near its capacity limits. This can occur because the number of hoppers is so large, and because each work request requires a relatively long time to service compared with the time to visit a single hopper.)

In this study, we assume that work enters the hoppers as a result of some outside process, which is independent of how the hoppers are being served. In No. 1 ESS, as in many other situations, much work does enter hoppers in this manner. However, it is also true that servicing a request from one hopper may place work, directly or indirectly, in another hopper. This interdependence may well be important in choice of a cycle. Nevertheless, the present model, which ignores such interdependence, is probably usable if we are suitably cautious about interpreting our results.

Service requests are discrete items and enter the hoppers according to an exceedingly complicated random process. Our model, however, assumes that each kind of request comes in at a constant rate, with no statistical fluctuation whatsoever. Furthermore, we treat the number of requests as a continuous quantity (so that requests keep trickling in like water) rather than a discrete quantity.

Despite the drastic nature of all these simplifications, we believe that this analysis is better than no analysis at all. Furthermore, we feel that our conclusions are probably valid approximations. It also seems plausible that our model could provide the jumping-off place for a more realistic study. Both interdependence and statistical fluctuation could be introduced in a limited way. (Since this was first written, R. W. Landgraff has done a study which extends this model to include interdependence.[3]) This might well permit their main effects to enter the model, without opening the Pandora's box of an extremely general stochastic process with one server and many interdependent queues.

II. SOME ASSUMPTIONS AND NOTATION

We suppose that there are I hoppers. For each hopper i we assume that we have three parameters:

s_i = service time = average time to service one request in this hopper,

r_i = request time = average time between occurrence of requests $\gg s_i$, and

w_i = weight = average penalty per second of delay for a single request of type i.

We also use

$$\lambda_i = \frac{s_i}{r_i} \ll 1, \qquad \Lambda = \sum_{i=1} \lambda_i \, .$$

(To permit a steady-state solution, we assume $\Lambda < 1$.) Note that the definition of w_i implies that *on the average* the penalty for delaying one kind of task is *proportional* to the delay time. The w_i are the proportionality constants. This simple assumption could be refined somewhat without too much trouble if desired.

In No. 1 ESS, one major penalty caused by hopper delays is the extra waiting time they cause to the telephone user at various stages of his call. For some hoppers, such as those involved during the process of dialing, undue delays can cause mishandling of the call. (Also, the delays tie up memory capacity and indirectly cause a need for extra memory equipment. However, this effect is probably minor.) By considering the loss incurred by the user due to various waiting periods, and the loss due to the probability of mishandled call, it would be possible to assign sensible values to the w_i. Although a truly realistic appraisal of the losses would require a quite elaborate study, some fairly reasonable simplifying assumptions which would make this study much simpler are available. Furthermore, assignment of the w_i on a direct intuitive basis would probably be adequate for many purposes.

To measure the total delay penalty paid by any work-scheduling algorithm, we combine the various penalties into a single number P:

d_i = expected delay for a request of type i,
p_i = expected penalty per request of type $i = w_i d_i$, and
P = expected total penalty per second

$$= \sum_{i=1}^{I} \frac{1}{r_i} p_i = \sum_{i=1}^{I} \frac{w_i}{r_i} d_i .$$

(Of course, $1/r_i$ is the expected number of requests of type i in one second.) We seek to minimize P by proper choice of a work-scheduling algorithm. Only the delays d_i may be influenced in this way, so we concentrate on evaluating the d_i.

A model which, like ours, treats requests as continuous has the danger of "discovering" that the hoppers are serviced infinitely fast, accumulating only an infinitesimal amount of work between visits. The following assumption, which in any case reflects an important reality, avoids this collapse.

To examine the ith hopper, whether or not it contains any work, requires a certain amount of time. We assume this amount of time is H_i. For simplicity we shall assume all the H_i are equal, and shall call their common value H, although it would be easy to work with unequal values if desired. Thus if x requests are serviced during one visit to hopper i, this visit requires $H_i + xs_i$ seconds.

It will turn out later that the value chosen for H is not very important in the context of this model. The comparison between different work-scheduling algorithms is unaffected by the (nonzero) value used.

III. WORK-SCHEDULING AND SERVICE POLICY

We suppose that the hoppers are visited in a fixed cycle of length N, namely,

$$(i_1, i_2, \cdots, i_N).$$

This means that hopper i_1 is visited first, then hopper i_2, and so on. After i_N is visited, the cycle starts over again with hopper i_1. One simple cycle with $I = 4$ and $N = 6$ is (1, 4, 2, 4, 3, 4). No. 1 ESS uses $I = 5$ hoppers (classes of hoppers, actually), and a cycle of length $N = 30$:

$$\overbrace{\overbrace{\overbrace{1\;2\;1}\;3\;\overbrace{1\;2\;1}}\;4\;\overbrace{\overbrace{1\;2\;1}\;3\;\overbrace{1\;2\;1}}}\;5\;\overbrace{\overbrace{\overbrace{1\;2\;1}\;3\;\overbrace{1\;2\;1}}\;4\;\overbrace{\overbrace{1\;2\;1}\;3\;\overbrace{1\;2}}}$$

If i is any given hopper, we shall let $V(i)$ indicate the set of all visits to hopper i. Thus for the cycle (1, 4, 2, 4, 3, 4), we have

$$V(1) = [1], \quad V(2) = [3], \quad V(3) = [5], \quad \text{and} \quad V(4) = [2, 4, 6].$$

In the No. 1 ESS cycle,

$$V(1) = [1, 3, 5, \cdots, 29], \qquad V(2) = [2, 6, 10, 14, 18, 22, 26, 30],$$

$$V(3) = [4, 12, 20, 28], \qquad V(4) = [8, 24], \qquad V(5) = [16].$$

For any visit n, the last previous visit to the *same* hopper is called $b(n)$ ("b" for before). Thus in the cycle (1, 4, 2, 4, 3, 4), visit 6 is to hopper 4, and the last previous visit to the same hopper is on visit 4. Thus $b(6) = 4$. Because "last previous" is understood in a cyclic sense, $b(2) = 6$. We have

$$b(1) = 1, \qquad b(2) = 6, \qquad b(3) = 3,$$

$$b(4) = 2, \qquad b(5) = 5, \qquad b(6) = 4.$$

Whenever a hopper is visited, we suppose that all work requests there are serviced. However, during the period when the hopper is being serviced new requests can enter it. What about these requests which enter the hopper while it is being serviced? These can either be handled when they are reached during the same visit, which we call the "come-right-in" policy, or they can be left for the next visit to the hopper, which we call the "please-wait" policy. We shall treat both of these hopper service policies, because their solutions are very similar.

IV. HOW TO EVALUATE P

As there is no statistical variation left in our model, it is easy to analyze. Let

t_n = time spent emptying the hopper i_n during visit n.

Let C be the time spent during an entire cycle, so that C consists of N hopper visits. Hopper visit n consists of time H to examine the hopper, and time t_n to service it. Thus

$$C = \sum_{n=1}^{N} (H + t_n) = NH + \sum_{n=1}^{N} t_n .$$

Now consider the requests which are serviced during t_n. Let

T_n = the interval during which they enter hopper i_n.

Recalling that $b(n)$ is the last prior visit to hopper i_n, we see from Fig. 1 that

$$T_n = \begin{cases} \displaystyle\sum_{p=b(n)+1}^{n} (H + t_p) = [n - b(n)]H + \sum_{b(n)+1}^{n} t_p , & \text{"come-right-in;"} \\ \displaystyle\sum_{p=b(n)}^{n-1} (t_p + H) = [n - b(n)]H + \sum_{b(n)}^{n-1} t_p , & \text{"please-wait."} \end{cases} \tag{1}$$

Note that $b(n)$ and the summation indices must be understood in a suitable "cyclic" sense, so that (for example) if $b(n) = n$, then $n - b(n)$ means once around the cycle and hence equals N, not 0. Now it is easy to see that

$$\begin{aligned} &(\text{the number of requests served during } t_n) = t_n/s_{i_n} \\ &\qquad = (\text{the number of requests initiated during } T_n) = T_n/r_{i_n} , \end{aligned}$$

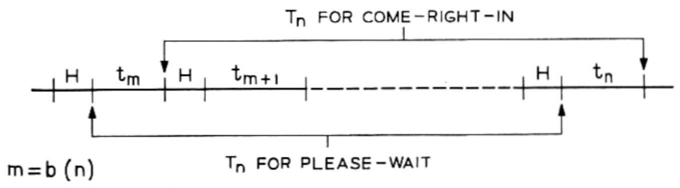

Fig. 1 —Time flow diagram illustrating "please-wait" and "come-right-in" policies.

so

$$t_n = \lambda_{i_n} T_n . \tag{2}$$

By using equation (2), we can eliminate either all T_n or all t_n from equations (1). This will leave us with N linear equations in N unknowns, which in fact turn out to be linearly independent. By solving these equations and using equation (2), we can find the T_n and the t_n , and from them all else will follow, as we show below. For convenient reference, we state the equations after eliminating the t_n :

$$T_n = \begin{cases} [n - b(n)]H + \sum_{p=b(n)+1}^{n} \lambda_{i_p} T_p , & \text{“come-right-in;”} \\ [n - b(n)]H + \sum_{p=b(n)}^{n-1} \lambda_{i_p} T_p , & \text{“please-wait.”} \end{cases} \tag{3}$$

Recall the special cyclic interpretation of $n - b(n)$ and the summations.

It is worth digressing briefly to derive an explicit formula for C, and to show how the N equations (3) can be reduced to $N - I$ equations in $N - I$ unknowns by using it. It is easy to see that if we sum T_n over all visits to some particular hopper j, the result must equal C:

$$\sum_{n \text{ in } V(j)} T_n = C \quad \text{for every } j. \tag{4}$$

Now sum equation (3) over all n in $V(j)$, and use equation (4) several times:

$$\begin{aligned} \sum_{n \text{ in } V(j)} T_n &= \sum_{n \text{ in } V(j)} \left\{ [n - b(n)]H + \sum_{p=b(n)+1}^{n} \lambda_{i_p} T_p \right\} , \\ C &= NH + \sum_{p=1}^{N} \lambda_{i_p} T_p \\ &= NH + \sum_{j=1}^{I} \lambda_j [\sum_{p \text{ in } V(j)} T_p] \\ &= NH + \left(\sum_{j=1}^{I} \lambda_j \right) C \\ &= NH + \Lambda C . \end{aligned}$$

This yields

$$C = \frac{NH}{1 - \Lambda} . \tag{5}$$

Since C is now given directly in terms of known quantities, we can use equation (4) to solve for one T_n in terms of others. We can do this separately for each $j = 1$ to I, and thereby reduce the number of unknowns and equations to $N\text{–}I$.

Once we have the values of T_n (and hence of t_n), we may easily evaluate e_n, the average delay for requests serviced during visit n. (Each delay is reckoned from occurrence of request to when its processing starts.) By elementary reasoning, we see that

$$\left.\begin{aligned} e_n &= \tfrac{1}{2}(T_n - t_n), \quad \text{“come-right-in,”} \\ e_n &= \tfrac{1}{2}(T_n + t_n), \quad \text{“please-wait.”} \end{aligned}\right\} \tag{6}$$

Of course T_n/r_{i_n} requests are serviced in visit n. Thus the average delay per request of type j is

$$d_j = \frac{\sum_{n \text{ in } V(j)} \frac{T_n}{r_j} e_n}{\sum_{n \text{ in } V(j)} \frac{T_n}{r_j}}. \tag{7}$$

Using equations (6), (2), and (4), we get

$$\left.\begin{aligned} d_i &= \frac{1 - \lambda_i}{2C} \sum_{n \text{ in } V(i)} T_n^2, \quad \text{“come-right-in,”} \\ d_i &= \frac{1 + \lambda_i}{2C} \sum_{n \text{ in } V(i)} T_n^2, \quad \text{“please-wait.”} \end{aligned}\right\} \tag{8}$$

Now let

$$F_n = T_n/C = \text{fraction of a cycle used by } T_n,$$

so that

$$\sum_{n \text{ in } V(i)} F_n = 1, \qquad \text{all } i. \tag{9}$$

Then

$$d_i = \begin{cases} \tfrac{1}{2}(1 - \lambda_i)C \sum_{n \text{ in } V(i)} F_n^2, & \text{“come-right-in,”} \\ \text{same, but with } 1 + \lambda_i \text{ for } 1 - \lambda_i, & \text{“please-wait.”} \end{cases} \tag{10}$$

Using equation (5) and the definition of P, we now easily find a formula for the penalty P, which is the key quantity we use to evaluate work-scheduling algorithms:

$$P = \begin{cases} \dfrac{NH}{2(1-\Lambda)} \left\{ \sum_i \dfrac{w_i}{r_i} (1-\lambda_i) \sum_{n \text{ in } V(i)} F_n^2 \right\}, & \text{“come-right-in,”} \\ \text{same, but with } 1+\lambda_i \text{ for } 1-\lambda_i, & \\ 1-\Lambda \text{ is unaffected} & \text{“please-wait.”} \end{cases} \tag{11}$$

(However, note that the values of the F_n may differ for the two policies.) We note that the work-scheduling algorithm influences equation (11) in only two ways: through N, and through the fractions F_n. From this formula we can evaluate and compare different work-scheduling algorithms. Also we can compare “come-right-in” with “please-wait.”

V. SOME EXAMPLES

If there are $I = 3$ different hoppers, the simplest possible cycle is (1, 2, 3), for which $N = 3$. In this case we see trivially that $F_1 = F_2 = F_3 = 1$, for either “come-right-in ”or “please-wait.” Thus equation (10) for cycle (1, 2, 3) is:

$$P = \begin{cases} \dfrac{3H}{2(1-\Lambda)} \sum_1^3 \dfrac{w_i}{r_i} (1-\lambda_i), & \text{“come-right-in,”} \\ \text{same, but with } 1+\lambda_i \text{ for } 1-\lambda_i, & \\ 1-\Lambda \text{ is unaffected} & \text{“please-wait.”} \end{cases}$$

Given the three input parameters s_i, r_i, and w_i for each hopper, this can be evaluated numerically.

Now suppose we use the cycle (1, 2, 1, 3), for which $N = 4$, with the “come-right-in” policy. Then equations (3) for cycle (1, 2, 1, 3) become the following four equations:

$$\begin{aligned} T_1 &= \lambda_1 T_1 + \lambda_3 T_4 + 2H, \\ T_2 &= \lambda_1(T_1 + T_3) + \lambda_2 T_2 + \lambda_3 T_4 + 4H, \\ T_3 &= \lambda_2 T_2 + \lambda_1 T_3 + 2H, \\ T_4 &= \lambda_1(T_1 + T_3) + \lambda_2 T_2 + \lambda_3 T_4 + 4H. \end{aligned}$$

However, taking C as known, and using equation (4) for cycle (1, 2, 1, 3) namely,

$$T_1 + T_3 = C, \qquad T_2 = C, \qquad T_4 = C,$$

we eliminate the unknowns T_2, T_3, and T_4, leaving one equation in one unknown, T_1 :

$$T_1 = \lambda_1 T_1 + \lambda_3 C + 2H.$$

We find

$$T_1 = \frac{1}{1 - \lambda_1} [2H + \lambda_3 C].$$

Dividing by C, and using $C = 4H/(1 - \Lambda)$ from equation (5), we see

$$F_1 = \frac{\frac{1}{2}(1 - \Lambda) + \lambda_3}{1 - \lambda_1} = \frac{1}{2} \frac{1 - \lambda_1 + \lambda_3 - \lambda_2}{1 - \lambda_1}$$
$$= \frac{1}{2}\left[1 + \frac{\lambda_3 - \lambda_2}{1 - \lambda_1}\right].$$

As $F_3 = 1 - F_1$, we find

$$F_1^2 + F_3^2 = \frac{1}{2}\left[1 + \left(\frac{\lambda_3 - \lambda_2}{1 - \lambda_1}\right)^2\right],$$

and also

$$F_2^2 = 1, \qquad F_4^2 = 1.$$

Thus equation (11) for cycle (1, 2, 1, 3) with the "come-right-in" policy is

$$P = \frac{4H}{2(1 - \Lambda)} \left\{ \frac{w_1}{r_1}(1 - \lambda_1)\frac{1}{2}\left[1 + \left(\frac{\lambda_3 - \lambda_2}{1 - \lambda_1}\right)^2\right] + \frac{w_2}{r_2}(1 - \lambda_2) + \frac{w_3}{r_3}(1 - \lambda_3) \right\}.$$

Through special circumstances which would not hold in general, the values for F_n using this cycle are all the same for "please-wait" as for "come-right-in," so P for "please-wait" is the same as the above but with $1 + \lambda_i$ substituted for $1 + \lambda_i$ in three places. Given the parameters s_i , r_i , and w_i for each hopper, this can be evaluated numerically.

VI. CONCLUSIONS

If we compare cycles of the same length and with the same number of visits to each hopper, then equation (11) yields the following conclusion: *The visits to a given hopper should be spaced as evenly around the cycle as possible.*

By this we mean that the values of T_n (and hence of F_n) pertaining to this hopper should be as equal as possible. This follows because the minimum of

$$\sum_{n \text{ in } V(i)} F_n^2 \quad \text{subject to} \quad \sum_{n \text{ in } V(i)} F_n = 1$$

occurs when the F_n with n in $V(i)$ are all equal. Furthermore, equation (11) can be used to estimate how serious any given deviation from equality is.

Suppose a cycle has N_i visits to hopper i, so that $N = \Sigma N_i$, and suppose that the N_i visits are spaced approximately evenly around the cycle for every i. Then for each visit n to hopper i,

$$F_n \approx \frac{1}{N_i}.$$

Thus

$$P \approx \frac{HN}{2(1 - \Lambda)} \sum_i \frac{w_i}{r_i} (1 - \lambda_i) \frac{1}{N_i}, \qquad \text{“come-right-in.”}$$

Either using a Lagrange multiplier to handle the constraint that $\Sigma N_i = N$, or by direct argument (see the appendix), it is easy to deduce that the values of N_i which minimize this satisfy

$$N_i \text{ proportional to } \left[\frac{w_i}{r_i}(1 - \lambda_i)\right]^{1/2}$$

so

$$\frac{N_i}{N} = \frac{\left[\frac{w_i}{r_i}(1 - \lambda_i)\right]^{1/2}}{\sum_j \left[\frac{w_j}{r_j}(1 - \lambda_j)\right]}.$$

This yields our most important conclusion: *The above approximate formula gives the optimum relative frequency of visits to each hopper in the cycle.*

By obtaining values for r_i, s_i, and less easily for w_i, it is possible to compare different work-scheduling algorithms with each other and with the “ideal” schedule with perfect spacing implied above. Notice that the actual value of H does not enter into this comparison. (If we had used unequal values for the H_i, only the ratios H_i/H_j would enter into the comparison, not the actual values of the H_i themselves.)

It would probably be worthwhile to analyze the actual work-scheduling algorithm used for ESS No. 1 in these terms. It would be interesting to compare this actual algorithm with the "ideal" algorithm.

Our model, with its highly simplified assumptions, cannot possibly provide the last word on work-scheduling evaluations, even with regard to delay times. However, this kind of approach is probably desirable. If greater realism is desired, the most important aspects are statistical variability and interdependence of hoppers.

APPENDIX

Direct Argument to Replace the LaGrange Multiplier Argument

Henry Pollak has pointed out a simple direct argument which shows that $\Sigma(a_i/N_i)$ is minimized, subject to the constraint $\Sigma N_i = N$, if N_i is proportional to $(a_i)^{\frac{1}{2}}$. Using $a_i = w_i(1 - \lambda_i)/r_i$, this yields the formula given above for N_i.

First, let $q = N/[\Sigma(a_i)^{\frac{1}{2}}]$. Now, we multiply the quantity to be minimized by q^2, and express it:

$$\Sigma \frac{q^2 a_i}{N_i} = \Sigma\left(q\left(\frac{a_i}{N_i}\right)^{1/2} - (N_i)^{1/2}\right)^2 + 2q\Sigma(a_i)^{1/2} - \Sigma N_i .$$

The middle term is constant by definition, and the last term is constant by constraint. The first term cannot be less than 0. The first term is 0 if

$$\frac{q^2 a_i}{N_i} = N_i \quad \text{or} \quad N_i = q(a_i)^{1/2}.$$

Since these values satisfy the constraint, we obtain the desired result.

REFERENCES

1. Keister, W., Ketchledge, R. W., and Vaughn, H. E., "No. 1 ESS: System Organization and Objectives," B.S.T.J., *43*, No. 5 (September 1964), pp. 1831–1844.
2. Harr, J. A., Hoover, Mrs. E. S., and Smith, R. B., "Organization of the No. 1 ESS Stored Program," B.S.T.J., *43*, No. 5 (September 1964), pp. 1923–1959.
3. Landgraff, R. W., unpublished work.