Waiting and Improvement Factor in Time-Sharing Systems

YOSHITERU ISHII*

1. *Introduction*

Data processing by computers, particularly by business computers, consists mainly of repetitious input-computing-output operations. For these ergodic processes, time-sharing is an effective way of improving computer's data processing capabilities. Improvement factor is defined as the ratio of the expected amount of data processed in a unit time by a system with time-sharing to that without time-sharing.

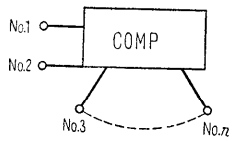

Fig. 1. Multiple connections to a time-sharing system

In a time-sharing computing system multiple connections numbered 1 through n are made from the computer to input/output devices and /or external storage devices (Fig. 1). This paper deals only with a computer which is connected to input/output devices though the results would also be quite general with respect to external storage devices. With such a time-sharing system, waiting is bound to occur, of which there are two types: waiting by input/output and that by computations. With each connection, 1 through n, in Fig. 1, wec an associate a flow of requests for input/output or a flow of requests for computations. Regarding the computer as a channel through which the resultant flow of the various component requests will pass, we can apply the theory for single-channel queues.

2. *Waiting by Input/Output Devices*

2.1. *Conditions for the occurrences of input/output waiting*

Assuming the capacity of our buffer register to be m characters and assuming the time required for transfer of m characters between the buffer register and the main memory is small enough to be ignored, we have the following conditions for occurrences of input/output waiting.

(i) *Waiting by an input device*

Assume that m characters from an input device have been transferred to the buffer register. Waiting occurs when a request from the input starting with the $(m+1)$st character meets with the buffer resister still

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 3, No. 3 (1962), pp. 7–17.

* Nippon Electric Co., Ltd., Tokyo.

filled with the first m characters.

(ii) *Waiting by an output device*

Similarly if a request from an output starting with the $(m+1)$st character meets with the buffer register being empty, waiting occurs.

2.2. *Probability for an occurrence of waiting and the expected value of waiting time*

Let T_1 be the time period between the 0th and 1st I/O operations. And let T_2 be the time period between #1 and #$(m+1)$. As can be seen in

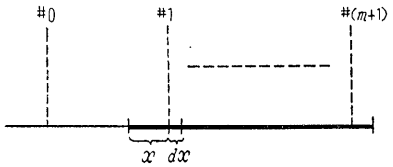

Fig. 2. Computing time

Fig. 2, the necessary and sufficient condition that #1 can be the first input/output after the start of calculations is

$$0 \leqq x < T_1 . \tag{1}$$

And if t is the time duration of computations, the necessary and sufficient condition for a waiting to occur is

$$t - x > T_2 . \tag{2}$$

As in Fig. 2 the probability p' that dx lies within the computing time and that the computing time has the duration t is

$$\left.\begin{aligned} p' &= -\lambda dx\, dF(t) && (t > x) \\ &= 0 && (t \leqq x) \end{aligned}\right\}, \tag{3}$$

where $F(t)$ is the probability that computing time is greater than t, and λ is a parameter assigned to the resultant of all computational request flows of input/output devices excepting the device under consideration. From (1), (2), and (3) we obtain the probability p of waiting as

$$p = -\lambda \int_{-\infty}^{\infty} dG_2(T_2) \int_{-\infty}^{\infty} dG_1(T_1) \int_0^{T_1} dx \int_{T_2+x}^{\infty} dF(t) . \tag{4}$$

$G_2(T_2)$ and $G_1(T_1)$ show the distribution functions for T_2 and T_1 respectively. Now letting $T_1+T_2=T_3$ and the corresponding distribution function $G_3(T_3)$, we get a general probability equation for an occurrence of waiting as:

$$p = \lambda \int_{-\infty}^{\infty} dG_2(T_2) \int_{T_2}^{\infty} (t-T_2)\, dF(t) - \lambda \int_{-\infty}^{\infty} dG_3(T_3) \int_{T_3}^{\infty} (t-T_3)\, dF(t) . \tag{5}$$

Now to obtain the expected value t_w of waiting time, we notice from

(2) that waiting time is given by $t-x-T_2$. Therefore, the general equation for t_w is given by

$$t_w=\frac{\lambda}{2}\left[\int_{-\infty}^{\infty} dG_2(T_2)\int_{T_2}^{\infty}(t-T_2)^2\,dF(t)-\int_{-\infty}^{\infty} dG_3(T_3)\int_{T_3}^{\infty}(t-T_3)^2\,dF(t)\right]. \quad (6)$$

2.3. *Waiting by input/output connected with a serial computer*

To examplify our results above, we will consider a model system which consists of a serial computer with a random access internal memory.

2.3.1. *Input/output time intervals*

The time intervals between characters transferred from mechanical input/output devices are considered to be constant. Also the intervals for manually operated keyboard input are observed as conforming to the normal distribution through χ^2 determination:

$$\left.\begin{aligned}\bar{T}&=0.187\ [\text{sec}],\\ {\sigma_T}^2&=0.196\times10^{-2},\\ \sigma_T&=0.0442\ [\text{sec}].\end{aligned}\right\} \quad (7)$$

Therefore, all the input/output can be considered as conforming to the normal distribution. Constant intervals for mechanical input/output would be special cases with the standard deviations of zero:

$$g(T)=\frac{1}{\sqrt{2\pi}\sigma_T}e^{-\frac{(T-\bar{T})^2}{2{\sigma_T}^2}}. \quad (8)$$

2.3.2. *Computing time*

The computing time is given by the sum of the execution times of instructions in a program.

(1) *Execution times of instructions*

The execution time of an instruction can be thought as being composed of two components, one being the time required for the actual computation by the arithmetic-logical circuit and the other being the time required for fetching and decoding of the instruction and for fetching and storing of data. With a random access memory the latter time can be assumed to be constant for any given instruction. We also consider the actual execution times for addition, subtraction, shift, and comparison as being constant, disregarding the variations caused by the number of the shifted digits as being insignificant in the total execution time. We denote the actual addition and subtraction times to be b and c respectively. Multiplication and division times have normally large variations.

(i) *Actual execution time for multiplication*

We consider the most commonly used method of repeated additions for

performing multiplication. The time required for shifting is ignored. If the sum of all digits in the multiplier is x, the actual execution time t' for multiplication is

$$t'=xb. \tag{9}$$

Assuming that the number system of the computer is based on the radix p, each of m digits in the multiplier can have any one of p values randomly. The number of combinations where the sum of the values of m equals x is given by the coefficients of y^x in the generating function.

And since the number of all possible combinations is p^m, the probability $P(x)$ that the sum of m digits equals x is approximated as a normal distribution obtained by integrating the generating function in the neighborhood of its saddle point.

Therefore, if we consider the constant time required for noncomputational part of the multiplication execution time to be a, the probability distribution function for the total multiplication execution time t becomes as follows:

$$\begin{aligned}\bar{t}&=\bar{t}'+a\\&=a+\frac{m(p-1)}{2}b,\end{aligned} \tag{10}$$

$$\begin{aligned}\sigma^2&=\sigma_x^2b^2\\&=\frac{mb^2(p^2-1)}{12},\end{aligned} \tag{11}$$

$$f(t)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(t-\bar{t})^2}{2\sigma^2}}. \tag{12}$$

(ii) *Actual execution time for division*

For division we consider the most commonly used restoring method. This method is the exact opposite of the repeated addition method for multiplication and requires $q+1$ substractions and one addition for a quotient of q digits. Similar to (i) we have

$$\bar{t}=\frac{mp}{2}c+b+a, \tag{13}$$

$$\sigma^2=\frac{mc^2(p^2-1)}{12}. \tag{14}$$

Thus we have shown that to a very good approximation, the execution times for the serial computer instructions with a random access memory can be expressed as the normal distribution functions. In case where the execution time is constant (such as with addition, subtraction, shift, and comparison) the variance of the function is simply zero.

(2) *Computing time for a constant number of instructions*

If there are m instructions to be performed, the computing time be-

comes the sum of the execution times for the m instructions, and this is also expressed as a normal distribution with

$$\bar{t}=\sum_{i=1}^{m}\bar{t}_i \tag{15}$$

and

$$\sigma^2=\sum_{i=1}^{m}\sigma_i^2 . \tag{16}$$

(3) *Computing time for a variable number of instructions*

Let us suppose that there are r sets of instructions which are iterated in a program, with $\bar{x}_i$ $(i=1, 2, \cdots r)$ the mean number of iterations for i, and $\sigma_{x_i}^2$ the corresponding variance. For each such set of instructions let $\bar{t}_i$ be the mean execution time and σ_i^2 the variance. Let $\bar{t}_0$ and σ_0^2 be the mean execution time and the variance respectively for the non-iterated part of the program. Using a moment generating function, the mean total computing time $\bar{t}$ and variance σ_t^2 are obtained as

$$\bar{t}=\bar{t}_0+\sum_{i=1}^{r}\bar{x}_i\bar{t}_i , \tag{17}$$

$$\sigma_t^2=\sigma_0^2+\sum_{i=1}^{r}\bar{x}_i\sigma_i^2+\sum_{i=1}^{r}\bar{t}_i^2\sigma_{x_i}^2 . \tag{18}$$

The results shown are general and applicable to any program.

2.3.3. *Probability for the occurrence of waiting and the expected value of waiting time*

For our model computer, $G_2(T_2)$, $G_3(T_3)$ and $F(t)$ in equations (5) and (6) can be obtained as the normal distribution functions. T_2 is the sum of input/output intervals for m characters. Therefore, if $\bar{T}$ and σ_T^2 are the mean and the variance respectively for one interval, the mean and the variance for T_2 become $m\bar{T}$ and $m\sigma_T^2$ respectively. Similarly T_3 has the mean $(m+1)\bar{T}$ and the variance $(m+1)\sigma_T^2$. Thus from (5) and (6) we get the following results.

$$p=\lambda\left[\frac{\bar{t}-m\bar{T}}{2}\left\{1+\Phi\left(\frac{\bar{t}-m\bar{T}}{\sqrt{2(\sigma_t^2+m\sigma_T^2)}}\right)\right\}+\sqrt{\frac{\sigma_t^2+m\sigma_T^2}{2o}}e^{-\frac{(\bar{t}-m\bar{T})^2}{2(\sigma_t^2+m\sigma_T^2)}}-\frac{\bar{t}-(m+1)\bar{T}}{2}\right.$$
$$\left.\left\{1+\Phi\left(\frac{\bar{t}-(m+1)\bar{T}}{\sqrt{2\{\sigma_t^2+(m+1)\sigma_T^2\}}}\right)\right\}-\sqrt{\frac{\sigma_t^2+(m+1)\sigma_T^2}{2\pi}}e^{-\frac{\{\bar{t}-(m+1)\bar{T}\}^2}{2\{\sigma^2+(m+1)\sigma_T^2\}}}\right], \tag{19}$$

$$t_w=\frac{\lambda}{2}\left[\frac{1}{2}(\bar{t}-m\bar{T})^2+m\sigma_T^2+\sigma_t^2\right\}\left\{1+\Phi\left(\frac{\bar{t}-m\bar{T}}{\sqrt{2(m\sigma_T^2+\sigma_t^2)}}\right)\right\}$$
$$+(\bar{t}-m\bar{T})\sqrt{\frac{m\sigma_T^2+\sigma_t^2}{2\pi}}e^{-\frac{(\bar{t}-m\bar{T})^2}{2(m\sigma_T^2+\sigma_t^2)}}$$
$$-\frac{1}{2}\left(\{\bar{t}-(m+1)\bar{T}\}^2+(m+1)\sigma_T^2+\sigma_t^2\right)\left\{1+\Phi\left(\frac{\bar{t}-(m+1)\bar{T}}{\sqrt{2\{(m+1)\sigma_T^2+\sigma_t^2\}}}\right)\right\}$$

$$-\{\bar{t}-(m+1)\bar{T}\}\sqrt{\frac{(m+1)\sigma_T^2+\sigma_t^2}{2\pi}}e^{-\frac{\{\bar{t}-(m+1)\bar{T}\}^2}{2\{(m+1)\sigma_T^2+\sigma_t^2\}}}\Bigg] \tag{20}$$

where $\Phi(r)$ is the probability integral

$$\Phi(r)=\frac{2}{\sqrt{\pi}}\int_0^r e^{-t^2}dt\,.$$

In Fig. 3 we have t_w for $m=1$ when

① $\sigma_T^2=\sigma_t^2=0$, ② $\left(\frac{\sigma_t}{\bar{t}}\right)^2=\frac{1}{20}$, $\sigma_T^2=0$, ③ $\left(\frac{\sigma_t}{\bar{t}}\right)^2=\frac{1}{10}$, $\sigma_T^2=0$,

④ $\left(\frac{\sigma_t}{\bar{t}}\right)^2=\frac{1}{5}$, $\sigma_T^2=0$.

By choosing $\bar{t}/\bar{T}$ properly, it is possible to reduce the waiting time.

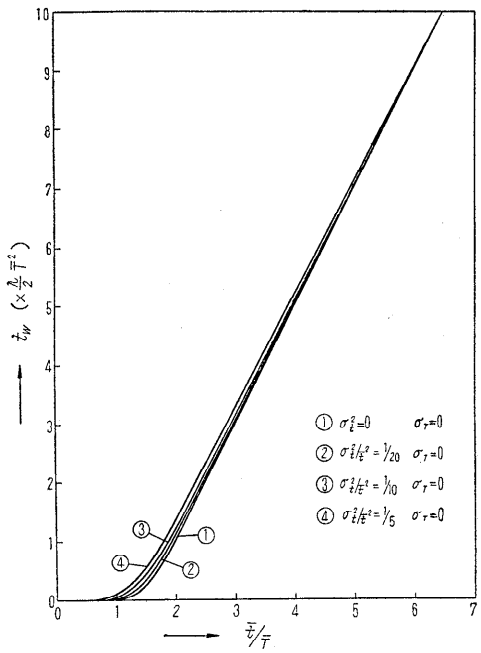

Fig. 3. Mean waiting time intervals by input/output (1)

3. *Waiting by Computations*

3.1. *Cases where groups of computations are executed sequentially with time-sharing*

We assume that at any point a desired group of computations must wait for the current group of computations to be completed before it is executed. The groups are executed sequentially according to the chronological order of requests.

A flow of requests for computations from an input/output device is called a component flow. When the number of component flows is large, the resultant flow (of component flows) approaches a simple flow conforming to the Poisson distribution.

Ordinarily, however, the number of component flows hardly ever exceeds numbers order 10, in which case the Poisson distrubution becomes questionable.

(1) *When the number of component flows is small*

The worst possible case is when a request is sent immediately after the beginning of the previous computations while at the same time all the other component flows are sending in requests. If we consider the n-th request, with $\bar{s}_i$ $(i=1, 2, \cdots, n-1)$ as the mean execution time for each of the $(n-1)$ preceding groups of computations, the probability that the computer is busy at the time of the n-th request is

$$\alpha_n=\sum_{i=1}^{n-1}\lambda_i\bar{s}_i\,. \tag{21}$$

In the worst possible case mentioned above the expected waiting time $\bar{r}_{n\,\max}$ becomes

$$\bar{r}_{n\,\max}=\alpha_n\sum_{i=1}^{n-1}\bar{s}_i\,. \tag{22}$$

(2) *When the number of component flows is large*

In this case the Poisson distribution holds and if $\bar{s}$ and σ^2 are the mean and the variance of the execution times, we get the expected value $\bar{r}$ of waiting time as

$$\bar{r}=\frac{\alpha}{2(1-\alpha)}\,\frac{\bar{s}^2+\sigma^2}{\bar{s}} \tag{23}$$

where $\alpha=\lambda\bar{s}$, and $\bar{s}$ and σ^2 are obtained according to (3).

3.2. *Case where groups of computations are divided into smaller batches to be executed sequentially with time-sharing*

For this case we consider only a small number of component flows. The execution times are likely to vary a great deal for different component flows. In order to even them out, the groups of computations are divided into batches of a same length. Assuming the n-th component flow to be divided into a mean of $\bar{q}_n$ batches and assuming $\bar{s}'$ to be the mean execution time for those batches, we get a situation similar to sec. 3.1 paragraph (1), with $\alpha_n=\sum_{i=1}^{n=1}\lambda_i\bar{s}_i$. And for the worst possible case, $\bar{r}_{n\,\max}$ becomes

$$\begin{aligned}\bar{r}_{n\,\max}&=\alpha_n\bar{q}_n(n-1)\bar{s}'\\&=\alpha_n(n-1)s_n\,.\end{aligned} \tag{24}$$

4. *Improvement Factor by Time-Sharing*

4.1. *Improvement factor for a single program*

Without time-sharing, for the i-th component flow, suppose λ_{i0} computations and the associated input/output are performed in a given unit time. With time-sharing λ_i computations and the associated input/output are performed. We then have a relation $\lambda_i \leqq \lambda_{i0}$ because λ_i includes waiting times. For our purpose here, we will ignore the input/output waiting times since they can be made small by choosing $\bar{t}$ properly (§ 2.3.3).

Therefore, if $\bar{r}_i$ is the mean value of waiting time durations, we have

$$\lambda_i = \frac{\lambda_{i0}}{1+\lambda_{i0}\bar{r}_i} \tag{25}$$

assuming that λ_i and λ_{i0} are parameters for the same program. Parameter λ of the resultant flow in time-sharing is

$$\lambda = \sum_{i=1}^{n} \lambda_i$$

$$= \sum_{i=1}^{n} \frac{\lambda_{i0}}{1+\lambda_{i0}\bar{r}_i}.$$

Therefore, if we assume that for a given program j, λ_{i0} and $\bar{r}_i$ are equal for all i, and denoting them by λ_{j0} and $\bar{r}_j$, we have

$$\lambda = \frac{n\lambda_{j0}}{1+\lambda_{j0}\bar{r}_j}.$$

Improvement factor in this case, is given by the ratio η_j of λ to λ_{i0}

$$\eta_j = \frac{n}{1+\lambda_{j0}\bar{r}_j}. \tag{26}$$

We will call η_j "the improvement factor by time-sharing for program j".

For a small number of component flows, the maximum $\bar{r}_j$ as in (22) and (24) is given by

$$\bar{r}_{j\max} = \alpha_j(n-1)\bar{s}_j$$

where

$$\alpha_j = (n-1)\lambda_j\bar{s}_j. \tag{27}$$

Denoting the η_j corresponding to this $\bar{r}_{j\max}$ by $\eta_{j\min}$, it is clear that

$$\eta_{j\min} \leqq \eta_j \leqq n, \tag{28}$$

$$\eta_{j\min} = \frac{n}{1+\alpha_j(n-1)\lambda_{j0}\bar{s}_j}. \tag{29}$$

And from (25) and (26)

$$\lambda_j = \frac{\eta_j}{n}\lambda_{j0}.$$

Now letting

$$\eta_{j\min} = \eta_j(1-\delta), \qquad \lambda_{j0}\bar{s}_j = \alpha_{j0}$$

and using (27) and (29) we get

$$\eta_j=\frac{2n}{1+\sqrt{1+4(n-1)^2\alpha_{j0}^2(1-\delta)}}.$$

Therefore, if we let $\delta=0$ here, we again have $\eta_{j\min}$, where

$$\eta_{j\min}=\frac{2n}{1+\sqrt{1+4(n-1)^2\alpha_{j0}^2}}, \tag{30}$$

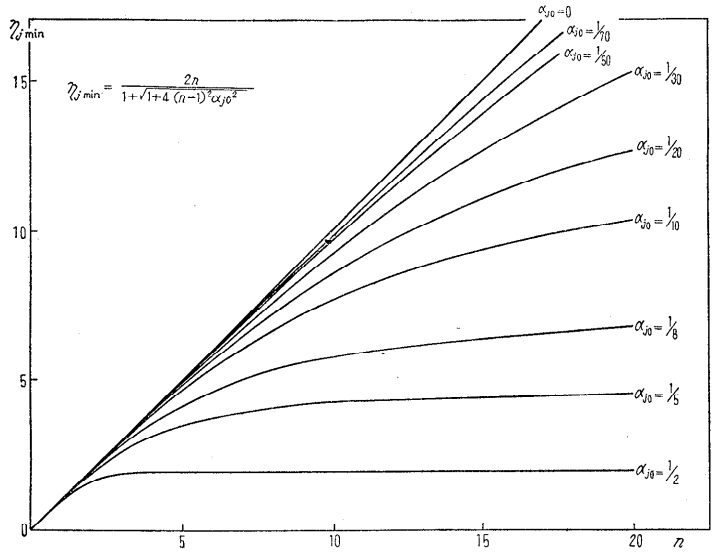

Fig. 4. Improvement factors for a given program

which also conforms to (28). Fig. 4 shows $\eta_{j\min}$ as a function of n with α_{j0} as parameter for the program j.

If $(n-1)^2\alpha_{j0}^2$ is small, (30) becomes

$$\eta_{j\min}\fallingdotseq n[1-(n-1)^2\alpha_{j0}^2]. \tag{31}$$

Therefore, the decreasing rate from n (n being the maximum η_j) is

$$\frac{n-\eta_{j\min}}{n}\fallingdotseq(n-1)^2\alpha_{j0}^2. \tag{32}$$

In the range where the decreasing rate is small, the allowable values for n can be determined by obtaining the decreasing value from (32) using α_{j0} from the given program.

4.2. *Improvement factor by time-sharing for a general case*

If the i-th component flow corresponds to program i and if the weight attached to it is w_i, then the amount of data processed for the component flow i in a given unit of time is proportional to

$$w_i\lambda_i=\frac{w_i\lambda_{i0}}{1+\lambda_{i0}\bar{r}_i}.$$

Therefore, for the resultant of n component flows it becomes proportional to

$$\sum_{i=1}^{n} w_1\lambda_i=\sum_{i=1}^{n}\frac{w_i\lambda_{i0}}{1+\lambda_{i0}\bar{r}_i}.$$

Without time sharing, the amount of data processed in n unit of time is proportional to

$$\sum_{i=1}^{n} w_i\lambda_{i0}.$$

Hence the improvement factor η by time-sharing for a general case is

$$\eta=\frac{n\sum_{i=1}^{n} w_i\lambda_i}{\sum_{i=1}^{n} w_i\lambda_{i0}}=\sum_{i=1}^{n}\left(\frac{w_i\lambda_{i0}}{\sum_{i=1}^{n} w_i\lambda_{i0}}\right)\left(\frac{n}{1+\lambda_{i0}\bar{r}_i}\right)=\sum_{i=1}^{n} p_i\eta_i \tag{33}$$

where

$$p_i=\frac{w_i\lambda_{i0}}{\sum_{i=1}^{n} w_i\lambda_{i0}}, \tag{34}$$

$$\eta_i=\frac{n}{1+\lambda_{i0}\bar{r}_i}. \tag{35}$$

p_i in this case is the relative weight corresponding to the i-th component flow. η_i is the same form as η_j in (26), and we refer to it as the improvement factor for the i-th program. w_i can be broadly thought as the amount of data to be processed by the i-th program.

To attach higher priority on a program is equivalent to assigning larger w_i and p_i and smaller $\bar{r}_i$ for that program.

Here again as in 3.2 we consider batches of computations to be executed with time-sharing. In dealing with multi-programs we assume no priority. Since p_i can be obtained from w_i and λ_{i0}, in order to get η we need only to obtain η_i. By substituting i for n in (26) we get $\bar{r}_{i\,\max}$, which when substituted in (33) for η_i gives $\eta_{i\,\min}$ the lower limit of η_i:

$$\eta_{i\,\min}=\frac{n}{1+(n-1)\alpha_i\alpha_{i0}} \tag{36}$$

where

$$\left.\begin{aligned}\alpha_i&=\sum_{j=1}^{n}{}' \lambda_j\bar{s}_j,\\ \alpha_{i0}&=\lambda_{i0}\bar{s}_i.\end{aligned}\right\} \tag{37}$$

It should be noted that $\sum_{j=1}^{n}{}'$ is a summation over $j=1, 2, \cdots, n$ with the exception of $j=i$. In (37) if we let α_i and λ_j be substituted by α_i' and λ_{i0} respectively, then $\alpha_i'\geqq\alpha_i$.

If we substitute α_i' for α_i in (36) and denote this $\eta_{i\,\min}$ by $\eta_{i\,\min}'$, $\eta_{i\,\min}'$ can still be considered as the lower limit of η_i; and $\eta_{\min}'$ obtained from

it can also be considered as the lower limit of η. Since α_i' actually never exceeds 1, we have

$$\eta_{i\,\min}' = \frac{n}{1+(n-1)[\alpha_i']\alpha_{i0}}, \tag{38}$$

$$\eta_{\min}' = \sum_{i=1}^{n} p_i \eta_{i\,\min}' \tag{39}$$

where

$$[\alpha_i'] = \sum_{j=1}^{n}{}' \lambda_{j0}\bar{s}_j \quad \text{when} \quad \sum_{j=1}^{n}{}' \lambda_{j0}\bar{s}_j \leqq 1,$$
$$= 1 \quad \text{when} \quad \text{〃} \quad > 1.$$

Since λ_{j0}, $\bar{s}_j$ are determined by each component flow program, $\eta_{i\,\min}'$ and, therefore, $\eta_{\min}'$ can be obtained.