Memoirs of the Faculty of Engineering, Okayama University, Vol.29, No. 2, pp.89-94, March 1995

Backward Scheduling to Minimize the Actual Mean Flow Time with Dependent and Independent Setup Times

Kenji SEKOGUCHI* and Shigeji MIYAZAKI**

(Received January 27 , 1995)

The present paper deals with a new performance measure, the actual mean flow time, defined as a mean of the elapsed time of each job counted from the start time on a schedule to the corresponding due date. For the one machine backward scheduling model with a common due date and independent setup times, LPT schedule is shown as the optimal solution for the proposed measure. An optimal algorithm is presented for the case with dependent setup times on the basis of the algorithm by Arcelus and Chandra for a $n / 1 / \bar{F}$ forward scheduling problem. The proposed algorithm is coded in C-language and a computational experience is reported through a 16-bit computer.

1. INTRODUCTION

Consider a set of n jobs J_i $(i = 1,2,\cdots,n)$ to be processed on one machine, where the processing time P_i $(i = 1,2,\cdots,n)$ for job J_i is given and fixed. The due date of each job is assumed as a common date (d) for all jobs. The ready time R_i (the preparation) of each job is assumed to be able to correspond with the start time B_i on a schedule to be made. Every job should be completed before or just on the due date (no tardy jobs allowed), and leave the shop simultaneously with its due date. Other conditions are the same as the ordinary scheduling model [1].

According to the conditions above, the flow time of each job in the shop inevitably coincides with the elapsed time of the job counted from the start time of each job on a schedule to the corresponding due date. In what follows the flow time defined above is called the actual flow time F_i^a which is formulated as :

$$F_i^a = d - B_i \ , \ (i = 1,2,\cdots,n), \tag{1}$$

An average of actual flow times over n jobs called actual mean flow time $\bar{F}^a$:

$$\bar{F}^a = (1/n)\sum_{i=1}^{n} F_i^a , \tag{2}$$

In this paper actual mean flow time is used as a performance measure and optimal scheduling algorithms for the setup dependent and independent problems are proposed to minimize $\bar{F}^a$ under a constraint of no tardy jobs.

The backward scheduling is adopted here for a solution procedure, in which the job is scheduled in reverse order as from the last position to the earlier position. The method has a merit of easy scheduling with no tardy jobs. For the one machine model with a common due date and independent setup times, LPT schedule is shown as the optimum for the proposed measure. A Branch and Bound algorithm is presented for the case with dependent setup times on the basis of an algorithm for a conventional forward $n / 1 / \bar{F}$ setup dependent problem with the simultaneously arriving jobs. The proposed algorithm is coded in C-language and a computational experience is reported through a 16-bit micro computer.

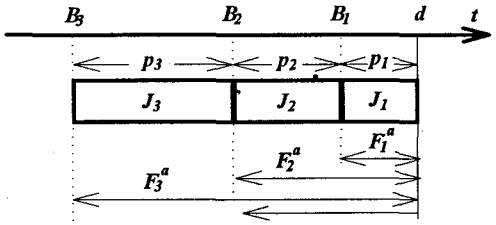
---

* The Graduate School of Natural Sience and Technology

** Engineering Mathematics

2. SEQUENCE INDEPENDENT SETUP TIMES

The model treated here can be illustrated in Fig. 1, in which the setup time of J_i is included in the processing time p_i and the sequence of jobs is being determined along with J_1 through J_3. The completion time of the last job (J_1) should coincide with the common due date (d) according to the constraint of no tardy job and for minimizing $\overline{F}^a$.



$$\overline{F}^a = (1/3)\sum_{i=1}^{3} F_i^a \rightarrow min.$$

Fig. 1. Sequence independent case.

[Theorem 1]

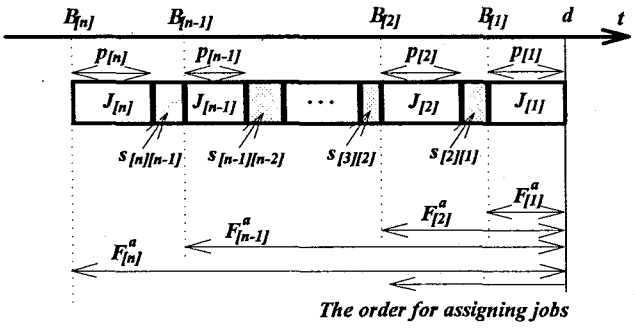
For the $n/1/\overline{F}^a$ backward model with sequence independent setup times, LPT schedule (the shorter job comes later position) gives the optimal solution.

Proof: $n/1/\overline{F}^a$ backward model can be transformed into the equivalent $n/1/\overline{F}$ forward model where SPT schedule is the optimal [2]. The reverse of SPT yields the optimal LPT schedule.

3. SEQUENCE DEPENDENT SETUP TIMES

Transformation into a forward model

In the case that the setup time for next job J_j is dependent upon the job J_i that was immediately processed on the machine, the setup time can be expressed by s_{ij}. The Gantt's chart for the one machine backward model with sequence dependent setup times for minimizing $\overline{F}^a$ (the model is expressed by $n/1/\overline{F}^a, s_{ij}/b$) is illustrated in Fig. 2. In the model the completion time of the last job should coincide with the due date (d) like the previous model.



$$\overline{F}^a = (1/n)\sum_{i=1}^{n} F^a_{[i]}$$

$$= (1/n)\sum_{i=1}^{n} (d - B_{[i]}) \rightarrow min.$$

Fig. 2. $n/1/\overline{F}^a, s_{ij}/b$ model.

If $J_{[i]}$ denotes the job sequence at the ith position counted from the end position on a time scale as shown in Fig. 2, and if $F^a_{[i]}$ and $p_{[i]}$ denote the actual flow time and the processing time of $J_{[i]}$, there exists a relation :

$$F^a_{[i]} = \sum_{j=1}^{i} p_{[j][j-1]} \ (i = 1,2,\cdots,n), \tag{3}$$

where

$$p_{[j][j-1]} = p_{[j]} + s_{[j][j-1]} \ \ (j = 2,3,\cdots,n),$$
$$p_{[1][0]} = p_{[1]} \tag{4}$$

and $s_{[j][j-1]}$ is the setup time attached for $J_{[j]}$ followed by $J_{[j-1]}$.

The actual mean flow time is given by :

$$\overline{F}^a = (1/n) \sum_{i=1}^{n} \sum_{j=1}^{i} p_{[j][j-1]} = (1/n) \sum_{i=1}^{n} (n-i+1)\, p_{[i][i-1]} \tag{5}$$

Arcelus and Chandra [3] have treated a one machine forward scheduling model with sequence dependent setup times for minimizing $\overline{F}$ (the model is expressed by $n/1/\overline{F}, s_{ij}/f$). $\overline{F}$ is defined as :

$$\overline{F} = (1/n) \sum_{i=1}^{n} F_i = (1/n) \sum_{i=1}^{n} (C_i - 0), \tag{6}$$

in which all the jobs are ready at time 0, and C_i denotes the completion time of J_i. The model is illustrated in Fig. 3.

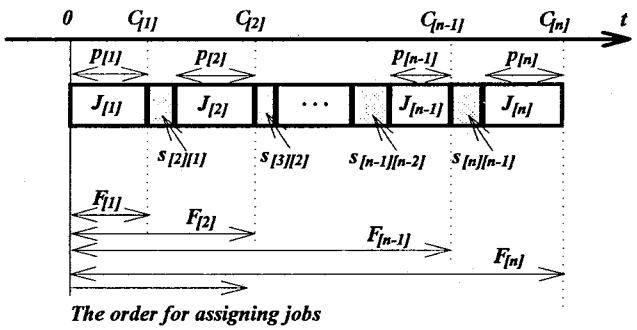


Fig. 3. $n/1/\overline{F}, s_{ij}/f$ model.

If $J_{[i]}$ denotes the job sequenced at the ith position counted from the beginning on a time scale in this case, there exists a relation :

$$\overline{F} = (1/n) \sum_{i=1}^{n} (n-i+1)\, p_{[i-1][i]}, \tag{7}$$

where

$$p_{[i-1][i]} = p_{[i]} + s_{[i-1][i]}, \ (i = 2,3,\cdots,n)$$
$$p_{[0][1]} = p_{[1]} \tag{8}$$

The difference between $n/1/\overline{F}, s_{ij}/f$ and $n/1/\overline{F}^a, s_{ij}/b$ is that $s_{[i-1][i]}$ is substituted by $s_{[i][i-1]}$. The arguments above show that $n/1/\overline{F}^a, s_{ij}/b$ problem can be solved by the following procedures. First, the data of setup time matrix s_{ij} should be transformed into s_{ji} in a new matrix (the data in ith row and jth column in the original matrix is exchanged to jth row and ith column in a revised matrix). Secondly solve the $n/1/\overline{F}, s_{ij}/f$ problem based on the revised matrix, and the obtained sequence should completely be reversed for the solution of $n/1/\overline{F}^a, s_{ij}/b$ problem.

Optimal algorithm

For $n/1/\overline{F}, s_{ij}/f$ problem, an optimal algorithm employing a Branch and Bound method was presented by Arcelus and Chandra. (AC algorithm) [3]. AC algorithm is adopting the frontier node search procedure which tends to expand explosively the memory spaces. The algorithm must provide several complicated steps for updating the list of nodes to discard the inactive nodes and renumber the active nodes consecutively. Although these steps are for saving the memory spaces, they cost much computational time.

This paper uses the active new node search procedure for saving the memory spaces beside the simple steps of the algorithm. The another advantage of the procedure is that it can rapidly reach trial solutions, even if a sufficient computational time for obtaining the optimal solution could not be provided.

The proposed algorithm for $n/1/\overline{F}^a, s_{ij}/b$ problem can be listed as follows :

[Step 1] The given matrix of setup times (s_{ij}) is transformed into s_{ji}. The element of the new matrix s_{ji} is rewritten as s_{ij} again. Make a matrix as shown in Tab. 1 in which the first row shows the processing time p_j of J_j and the other rows show p_{ij} defined by $p_{ij} = p_j + s_{ij}$. The minimum of each row i is denoted by $r(i)$ $(i = 0,1,\cdots,n)$ and the minimum of each column j is denoted by $c(j)$ $(j = 1,2,\cdots,n)$. Rearrange $r(i)$ $(i = 1,2,\cdots,n)$ and $c(j)$ $(j = 1,2,\cdots,n)$ in the nondecreasing order and denote them as $r[1],r[2],\cdots,r[n]$, and $c[1],c[2],\cdots,c[n]$.

Compute $RBND(k)$,$CBND(k)$ and $BND(k)$ by :

$$RBND(k) = \sum_{j=1}^{k}(k-j+1)r[j],\ (k = 1,2,\cdots,n-1), \tag{9}$$

$$RBND(n) = n \cdot r(0) + \sum_{j=1}^{n-1}(n-j)r[j], \tag{10}$$

$$CBND(k) = \sum_{j=1}^{k}(k-j+1)c[j],\ (k = 1,2,\cdots,n), \tag{11}$$

$$BND(k) = max\{RBND(k),CBND(k)\},\ (k = 1,2,\cdots,n), \tag{12}$$

where $RBND(k)$,$CBND(k)$ and $BND(k)$ stand for lower bounds based on the rows, the columns and the both of them in the p_{ij} matrix at the stage of k remaining (unassigned) jobs. $BND(n)$ is used as the lower bound obtained in this stage. Go to Step 2.

Tab. 1. Matrix showing p_{ij} (the sum of proccesing time p_j and setup time s_{ij}).

		To job					
		J_1	J_2	$\cdots$	J_j	$\cdots$	J_n	*min*
From job	J_1	p_1	p_2	$\cdots$	p_j	$\cdots$	p_n	$r(0)$
	J_1	--	p_{12}	$\cdots$	p_{1j}	$\cdots$	p_{1n}	$r(1)$
	J_2	p_{21}	--	$\cdots$	p_{2j}	$\cdots$	p_{2n}	$r(2)$
	:	:	:		:		:	:
	J_i	p_{i1}	p_{i2}	$\cdots$	p_{ij}	$\cdots$	p_{in}	$r(i)$
	:	:	:		:		:	:
	J_n	p_{n1}	p_{n2}	$\cdots$	p_{nj}	$\cdots$	--	$r(n)$
	min	$c(1)$	$c(2)$	$\cdots$	$c(j)$	$\cdots$	$c(n)$	

[Step 2] Obtain a feasible solution by "closest unvisited city" algorithm for the traveling salesman problem, and set the performance measure of the solution as $SOLN$. If $BND(n) = SOLN$,the feasible solution by "closest unvisited city" is the optimal solution, and terminate the algorithm. Otherwise, set the branching level (the number of already assigned jobs) $r=0$, the branching node $N_r^0 = \varnothing$, the upper bound $UB(N_r^0) = SOLN$, the performance measure of a best known solution $f^* = SOLN$, and go to Step 3. Note that $n - k = r$ holds, because k is defined as the remaining (unassigned) jobs.

[Step 3] Perform the branching procedure for N_r^0 and create $(n - r + 1)$ kinds of new nodes . Set $r = r + 1$ and $k = k - 1$.

[Step 4] For each node newly created, compute the total actual flow time of the partial sequence of already assigned jobs $PAAF$ by :

$$PAAF = \sum_{i=1}^{r}(n-i+1)\,p_{[i-1][i]}. \tag{13}$$

If the sum of $PAAF$ and $BND(k)$ by eq. (12) is greater than or equal to f^*, delete this node from the active node list and apply Step 4 to another node newly created. If no other nodes are remaining in this step, go to Step 8. If $PAAF + BND(k) < f^*$, go to Step 5.

[Step 5] Compute lower bound $LB(N_r)$ as follows : Revise the current p_{ij} matrix shown in Tab. 1 by the following procedure. Delete all the elements in the first row of the current p_{ij} matrix; delete all the elements in the column corresponding to the job (J_i^*) newly assigned for creating N_r; remove the row for J_i^* to the top of rows and shift the other rows accordingly. Compute $BND(k)$ by eq. (12) based on the revised p_{ij} matrix. Set lower bound as $LB(N_r) = PAAF + BND(k)$.

Compute $SOLN$ based on the revised p_{ij} matrix through the closest unvisited city algorithm. If the new $SOLN$ is less than f^*, set $f^* = SOLN$. Otherwise, maintain the previous f^*. If Step 5 has applied to all the nodes newly created, go to Step 6. Otherwise, repeat Step 4 for the next node remaining at the same stage.

[Step 6] Choose the node having the minimum lower bound among newly created nodes as the objective node (N_r^0) for a new branching. If $r = n$, then go to Step 7. If $r < n$ and $LB(N_r^0) < f^*$, then go to Step 3. If $r < n$ and $LB(N_r^0) \geqq f^*$, then go to Step 8.

[Step 7] If $LB(N_r^0) < f^*$, then the current N_r^0 should be a new trial solution. Set $f^* = LB(N_r^0)$, and go to Step 8. If $LB(N_r^0) \geqq f^*$, then delete N_r^0 and go to Step 8.

[Step 8] If any active nodes having the lower bounds smaller than f^* exist, then choose the node having the minimum lower bound among the node nearest to the current node N_r^0. Set the node as the new objective node N_r^0 for a branching. Reset r at the same level as the choosen N_r^0. Delete all the newest nodes, and return to Step 3. If there exists no objective node for branching, then go to Step 9.

[Step 9] Reverse the sequence of the current trial solution to obtain the optimal solution for the $n/1/\overline{F}^a, sij/b$ problem. (END)

4. COMPUTATIONAL EXPERIENCE

The proposed algorithm has been coded in C-language and numerical examples were solved through 16-bit micro computer, NEC PC-9801 VX, to obtain a computational experience. The average computational time to reach the optimal solution over ten example problems is shown in Fig. 4, of which horizontal axis is the number of jobs included in each problem. The average computational times through the complete enumeration mothod are estimated as 1.3 years for 14-job problem and 20.1 years for 15-job problem. Although the computational time through the proposed algorithm increases exponentially, it is less than 2 hrs. for 21-job problem, and less than 3.5 hrs. for 22-job problem. This time is rather better than the computational time of other implicit enumeration methods for general scheduling problems.

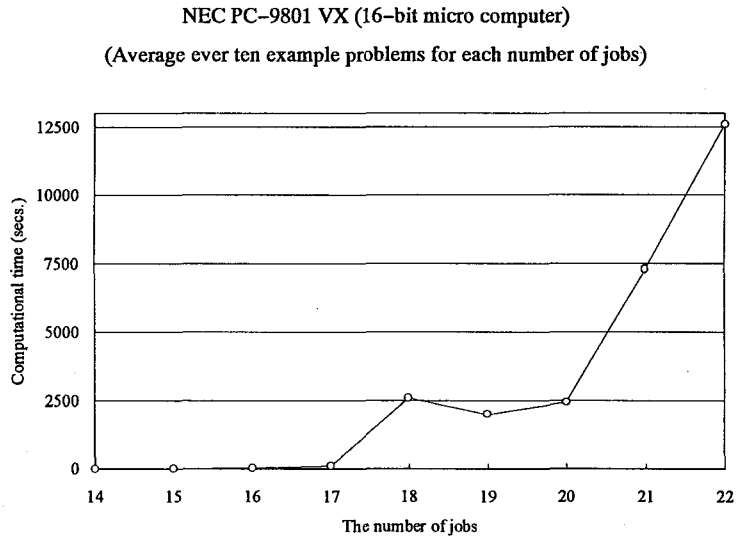


Fig. 4. Average computational time through the proposed algorithm

By use of the active new node search procedure, the proposed algorithm can yield trial solutions before obtaining the optimal solution. Each trial solution through the pass to the optimal solution was evaluated by the approximation ratio η :

$$\eta = \frac{\omega - t}{\omega - o} \times 100 \quad (\%) \qquad (14)$$

where ω, t and o are the performance measures of the worst (maximum), the trial and the optimal (minimum) solutions, respectively. The approximation ratio η defined by eq. (14) reaches 100% in the case that the trial solution coincides with the optimal one and 0% in the case with the worst one. The average of η calculated by ten example problems vs. the ordinal number of trial solutions is shown in Fig. 5, and the average of η vs. the required computational time is in Fig. 6. The average of η could attain 95% before obtaining the 10th trial solution and within the average computational time of 100 secs. for the example problems including less than 23 jobs.

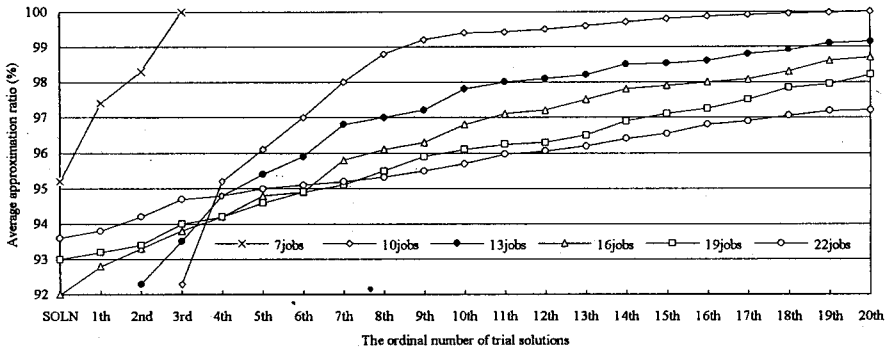


Fig. 5.Average approximation ratio vs. the number of trial solutions.

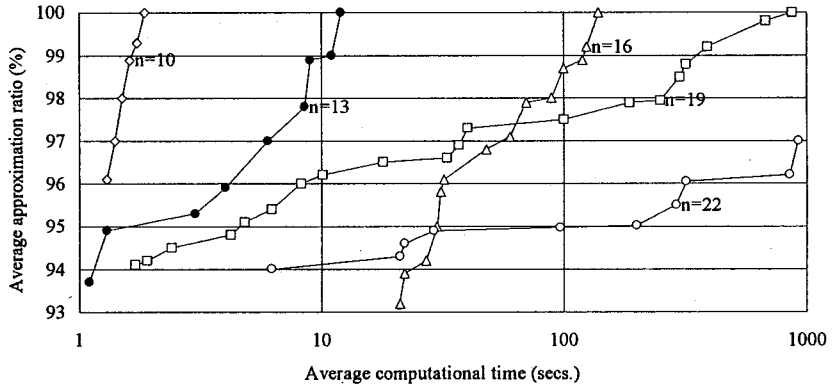


Fig. 6. Average approximation ratio vs. average computational time.

5. CONCLUDING REMARKS

The present paper proposed a new performance neasure, actual mean flow time $\overline{F}^a$, which reflects the recent popular production circumstance of which typical example can be seen in just-in-time production. The optimal algorithm for $n/1/\overline{F}^a$ backward scheduling with sequence dependent setup times was also presented. The algorithm employs a Branch and Bound method for $n/1/\overline{F}$ forward scheduling problem with a modification into the active new node search procedure.

The algorithm was coded in C-language and a computational experience was reported through a NEC PC-9801 VX micro computer. Although the computational time increases exponentially, it succeeded to obtain the optimal solution of 22-job problem within one shift duration (eight hours). The trial solutions on the way to the optimal solution can be used as a practical solution, because they revealed more than 95% approximation retio (performance measure attained among the range between the worst and the optimal solution) and the required computational time is within 100 secs. for the problem of which size is less than 23 jobs.

REFERENCES

[1]. K.R.Baker, Introduction to Sequencing and Scheduling, John Wiley, New York, (1974).

[2]. R.W.Conway, W.L.Maxwell and L.W.Miller, Theory of Scheduling, Addison-Wesley, Reading, (1967), pp.27-28.

[3]. F.J.Arcelus and R.Chandra, On $n/1/\overline{F}$ setup dependent problems, Engineering Optimization, 7 (1983), pp 58-67.