A dynamic programming algorithm for the single machine total completion time scheduling problem with availability constraints

Chérif Sadfi and Bernard Penz and Christophe Rapine
GILCO, 46 avenue Félix-Viallet, 38031 Grenoble, France.
sadfi@gilco.inpg.fr, penz@gilco.inpg.fr, rapine@gilco.inpg.fr

Abstract

In this paper, we study the single machine total completion scheduling problem subject to a period of maintenance. The problem is known to be *NP*-hard. We propose a pseudo-polynomial dynamic programming algorithm for the problem with time complexity *O(*nR*)*, where n refers to the number of jobs and R to the date of the beginning of the maintenance. The efficiency of the algorithm is improved using a fixation of variables technique.

1. Introduction

In this paper, we consider the problem of scheduling a set of n tasks $\{J_1, ..., J_n\}$ on a single machine. Each job requires a processing time p_i (integer) on the machine. The objective is to minimize the total completion time ΣC_i where C_i represents the completion time of job J_i. The machine is subject to a period of maintenance during the time horizon of the schedule. The maintenance is planned between the known dates R and D, and its duration is denoted by L. The preemption is not allowed, *i.e.* a task cannot be interrupted by the maintenance.

The problem is denoted in standard notations as $1,h_1| |\Sigma C_i$. Adiri et al. [1] and Lee and Liman [2] established that this problem is NP-hard. They also studied the SPT (shortest processing time) algorithm to solve the problem. Lee and Liman [2] showed that the SPT sequencing is not worse than a factor of 9/7 from an optimal schedule. Sadfi et al. [3] proposed an improvement of the SPT algorithm (MSPT), and proved a performance ratio of 20/17. For a survey of scheduling problems with limited machine availability see the paper from Schmidt [5].

This paper is organized as follows. The dynamic programming approach is presented in the next section. The following section describes the fixation of variables technique used to improve the dynamic programming algorithm.

2. A dynamic programming approach

Adiri et al. [1] and Lee and Liman [2] proved that the solutions in which jobs before and after the maintenance are sequenced in SPT order are dominant. The dynamic programming algorithm we propose now is based on this result. The problem is then to find a partition of jobs into two sets B and A such that jobs in B are scheduled before the maintenance and jobs in A are scheduled after. In the following, we assume the jobs indexed in the non-decreasing order of their processing times.

We introduce now some notations. For a set of jobs J, the quantity p(J) will represent the sum of the processing times of the jobs. We define s(J) as the sum of completion times of the jobs in set J sequenced in SPT order from time 0. Then, for a given partition $B \cup A$, the objective value is:

$$s(B \cup A)=s(B)+ |A|D + s(A)$$

The partition $B \cup A$ leads to a feasible solution if all jobs in B can be scheduled before the maintenance, and then $p(B) \leq R$. The problem is then to find an optimal solution $B \cup A$ minimizing $s(B \cup A)$, such that $p(B) \leq R$.

Let $\sigma(k,r)$ be the smallest total completion time of a solution for the set of jobs $\{J_1, ...,J_k\}$, and the total processing time of jobs scheduled before R is r. If such a solution does not exist, we set $\sigma(k,r)=+\infty$. Then the optimal objective value for all the jobs is clearly $\min\{\sigma(n,r) \mid r=0,...,R\}$.

Consider an optimal partition $B\cup A$ of jobs $\{J_1, ...,J_k\}$ with an objective value $\sigma(k,r)$. It defines a partition $B'\cup A'$ of jobs $\{J_1, ...,J_{k-1}\}$ ($B'=B \setminus \{J_k\}$ and $A'=B \setminus \{J_k\}$).

Assume that J_k is scheduled before the maintenance in the optimal solution. The job J_k is put in set B ($B=B'\cup\{J_k\}$). Clearly, due to the dominance of SPT ordering, it must be the last job of the sequence. Then we obtain:

$$\sigma(k,r) = r + s(B') + |A'|D + s(A') = r + s(B'\cup A')$$

The sub-optimality principle shows that we have to find an optimal solution $B'\cup A'$ for the problem of scheduling the jobs $J_1,...,J_{k-1}$, under the constraints that $p(B') \le r-p_k$. Hence we have:

$$\sigma(k,r) = \sigma(k-1,r-p_k) + r$$

Assume now that the job J_k is scheduled after the maintenance period in the optimal schedule. In the same way, it is the last job scheduled among all the jobs from set $A=A'\cup\{J_k\}$. Then:

$$\sigma(k,r) = s(B') + |A'|D + s(A') + C_k = s(B'\cup A') + D + P_k - r \quad \text{where} \quad P_k = \sum_{i=1}^{k} p_i$$

Hence we have to find an optimal solution $B'\cup A'$, subject to $p(B) \le r$. Then:

$$\sigma(k,r) = \sigma(k-1,r) + D + P_k - r$$

Finally, the dynamic formulation of our problem is:

$$\sigma(k,r) = \min\{\sigma(k-1,r-p_k) + r \, , \, \sigma(k-1,r) + D + P_k - r\}$$

A description of the dynamic programming algorithm is given in figure 1. The time complexity of our algorithm is in $O(nR)$.

Input: A set $J_1,...,J_n$ of jobs sorted in a non-decreasing order of their processing times.
Set $\sigma[0,0] := 0$
For r=1..R
    set $\sigma[0,r] := +\infty$
EndFor ;
For k=1..n
    For r=0..R
        If ($r-p_k < 0$) **Then** set *before* := $+\infty$
        Else set *before* := $r + \sigma[k-1, r-p_k]$
        set *after* := $D + P_k - r + \sigma[k-1, r]$
        set $\sigma[k,r] = \min\{before, after\}$
    EndFor ;
EndFor ;
Output: Optimum value := $\min_{\{r=0,..,R\}} \sigma(n,r)$.

Figure 1. Dynamic programming algorithm.

3. A dynamic programming algorithm with fixation of variables

The principle of the fixation of variables is to test if a job is necessarily before (resp. after) the maintenance in an optimal solution. Let consider a job J_x. We determine if J_x is scheduled before

the maintenance in any optimal schedule by computing a lower bound LBb(x) of the objective value of solutions scheduling J_x before the maintenance. We compare the value of this lower bound to a known solution obtained by an approximation algorithm (MSPT in our study [3]). If the lower bound is greater than the known feasible solution, we conclude that J_x should be after the maintenance in an optimal solution. Using the same technique, we test if J_x should be before the maintenance. In this case, we need a lower bound LBa(x) that takes into account that J_x is after the maintenance.

For the lower bound LBa(x), we consider the solution $B_{SPT} \cup A_{SPT}$ given by the SPT sequencing of the jobs. Let J_x be a job scheduled in this solution before the maintenance, *i.e.* $J_x \in B_{SPT}$. We take for LBa(x) the objective value of the solution scheduling job J_x immediately after the maintenance, and sequencing the other jobs according to the SPT ordering, allowing preemption. For LBb(x), we consider a job $J_x \in A_{SPT}$ and we schedule it just before the maintenance. We proved that LBa(x) and LBb(x) are lower bounds [4].

The fixation of variables phase allows to reduce the size of the problem by fixing some jobs before or after the maintenance. Let assume that the fixation phase has determined a set B_f of jobs scheduled before the maintenance in any optimal solution, and a set A_f of jobs scheduled after. Let $n'=n-|B_f|-|A_f|$ be the number of remaining jobs. For sake of simplicity, we reindex these jobs $J_1,...,J_{n'}$ in the non-decreasing order of their processing times. The problem is now to find a partition $B \cup A$ of $J_1,...,J_{n'}$. In order to calculate an optimal partition using a dynamic programming technique, we have to compute the following quantities.

Let us consider a set T of jobs together with a job J_x not belonging to T; we define:

$$p(T, J_x) = p(\{ y \in T \mid p_y \leq p_x\})$$
$$v(T, J_x) = |\{y \in T \mid p_y > p_x\}|$$

To get the meaning of these quantities, consider a SPT ordering of the jobs $T \cup \{J_x\}$. Then $p(T,J_x)$ represents the sum of the processing times of the jobs scheduled before J_x in the sequence, while $v(T, J_x)$ is the number of jobs scheduled after J_x.

Now consider again the dynamic programming scheme presented in the previous section. Given the sets B_f and A_f, we compute dynamically $\sigma'(k,r)$, the best objective value for a partition $B \cup A$ of the jobs $\{J_1,...,J_k\} \cup B_f \cup A_f$, under the constraint that $p(B) \leq r$ and $B_f \subseteq B$ and $A_f \subseteq A$. We have two possibilities for the value of $\sigma'(k,r)$:

Assume that J_k is scheduled before the maintenance. Job J_k is not necessarily the last job of the sequence, since some jobs of B_f can have a larger processing time. But due to the dominance of SPT, its completion time is exactly $p(B_f,J_k) + r$, and exactly $v(B_f,J_k)$ jobs of B_f are scheduled after it. Thus we obtain:

$$\sigma'(k,r) = \sigma'(k-1,r-p_k) + p(B_f,J_k) + r + v(B_f,J_k)p_k$$

Conversely assume that J_k is scheduled after the maintenance. In the same way we obtain:

$$\sigma'(k,r) = \sigma'(k-1,r) + D + p(A_f,J_k) + P_k - r + v(A_f,J_k)p_k$$

The algorithm consists in computing the quantities $p(B_f, J_k)$, $v(B_f,J_k)$, $p(A_f,J_k)$ and $v(A_f,J_k)$ for all the unfixed jobs J_k. We then call the dynamic programming procedure using the two previous equalities on the instance constituted of $n' = n - |B_f \cup A_f|$ unfixed jobs, with $R' = R - p(B_f)$.

4. Computational experiences

A computational experiment was conducted to evaluate the performance of the proposed methods. The algorithm has been implemented in C language on a PC/Pentium 3 (500MHz). The processing times of jobs were randomly generated as integer numbers in the interval [1,100]. We studied the improvement of the fixation of variables method on the running time of the dynamic

programming algorithm. The total number of jobs ranges from 1000 to 25000 jobs, and 100 tests were run for each set of conditions on n, L and R. When R= 25% of the sum of processing times of the jobs, the dynamic programming algorithm solves optimally instances of 25000 jobs in about 11 minutes. The figure 2 compares the evolution with the number of jobs of the running time of the dynamic programming algorithm without (DYN) and with the fixation of variables method (DYN_FIX). The figure 3 shows the evolution with the number of jobs of the ratios: n_free=100.n'/n, R_free=100.R'/R and the percentage of improvement of the running time of the

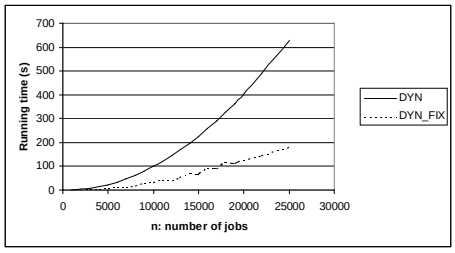

Figure 2

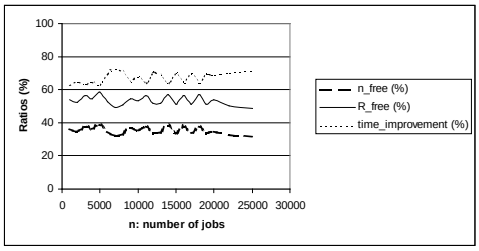

Figure 3

dynamic programming algorithm when using the fixation of variables method (time_improvement). Using the fixation of variables method, the running time of the dynamic programming algorithm is reduced, and solving instances of 25000 jobs takes time in average less than 3 minutes. The fixation of variables method can fix about 60% of the jobs and R is reduced with an average of 45%. The running time is reduced with an average of 68%.

5. Conclusion

In this paper, we proposed a pseudo-polynomial dynamic programming algorithm to solve the single machine total completion time scheduling problem subject to a single period of maintenance. We improved this algorithm by a fixation of variables technique, which allows us to fix some jobs to be scheduled before the period of maintenance and some other ones after the period of maintenance. Using the fixation of variable method, the dynamic programming algorithm is improved, and the running time is reduced with an average of 68%.

Acknowledgments

The authors wish to thank the referees whose suggestions led to a significant improvement of an earlier version of this paper.

References

[1] Adiri, J., Bruno, J., Frostig, E. and Rinnooy Kan, A.H.G. (1989). Single machine flow-time scheduling with a single breakdown, *Acta Informatica*, 26, 679-696.

[2] Lee, C.-Y. and Liman, S.D. (1992). Single machine flow-time scheduling with scheduled maintenance. *Acta Informatica*, 29, 375-382.

[3] Sadfi, C., Błażewicz, J., Formanowicz, P., Penz, B. and Rapine, C. (2001). A better approximation algorithm for the single machine total completion time scheduling problem with availability constraints. *International Conference on Industrial Engineering and Production Management*, Québec, Canada.

[4] Sadfi, C., Penz, B. and Rapine C. (2001). Single machine total completion time scheduling problem with availability constraints. *Research Report GILCO*.

[5] Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, 121(1), 1-15.