An efficient algorithm to solve 1// $\sum C_i + \sum Y_i$ problem

By

Haeder Younis Ghawi

Dept. of math./college of computer science and math./ AL_Qadisiyia univ.

hadirkawi@yahoo.com

keywords: ***Deterministic, multi_criteria , single machine ,late work.***

Abstract:

In the problem of scheduling a single machine to minimize the sum of completion time and total late work, there are ***n*** jobs to be processed for which each has an integer processing time and a due date. The objective is to minimize the sum of total completion time and total late work ,where the late work for a job is the amount of processing of this job that is performed after its due date.

Although dominance rules are derived for the special cases in which all processing times are equal and all due dates are equal. Algorithm H is presented for the general non preemptive sum of completion time and the total late work simultaneous problems .

1. Introduction

The late work objective function, minimizing the total number of tardy units of particular activities executed in the system, takes into account mostly the mount of late work not the quantity of the delay. It combines the idea of two classical performance measures the total tardiness and the number of late jobs. Similarly as tardiness, late work increases with a job delay, but when the job becomes totally late, the penalty remains on the certain level (determined by the job processing time) as for the number of late jobs parameter.

The paper concerns a performance measure based on the amount of late work in the system [2, 6 9, 11]. This objective function was first proposed in the context of parallel machines [2, 4] and then applied to the one-machine scheduling problem [9, 10]. Based on this concept, a new branch of the research has also appeared which modifies the original formulation of the late work for the real-time applications by considering so called imprecise computations [1].

The scheduling model that we study is as follows. There are ***n*** independent jobs $J_1,\ldots,J_n$ that have to be scheduled on a single machine. Each job J_i (i = 1,… ,n) has a processing time p_j and a due date d_j . All jobs are available for processing at time 0. The goal is to schedule the jobs without preemption on the single machine such that the total late work and sum of completion time is minimized.

The field of late work scheduling has not been widely explored (see Table 1) which causes some problems in estimating the complexity of other cases, not analyzed yet[5].

PROBLEM	**COMPLEXITY**	**REFERENCE**
$P \mid r_i \mid Yw$	Unary *NP*-hard	[4]
$P \mid pmtn, r_i \mid Yw$	$O(n^7 \log n)$	[6]
$Qk \mid pmtn, r_i \mid Yw$	$O(k^3 n^7 \log kn)$	[6]
$1 \mid pmtn \mid Y$	$O(n \log n)$	[22]
$1 \mid\mid Y$	binary *NP*-hard	[22]
$1 \mid d_i = d \mid Y$	$O(n)$	[22]
$1 \mid p_i = p \mid Y$	$O(n \log n)$	[22]

Table 1. Results for the late work criteria

However, based on the gathered results, the late work criterion seemed to be settled in the difficulty rank between the maximum lateness and mean tardiness criteria [2].

Some basic assumptions taken from classical scheduling theory that are applied to each of the rules and algorithms presented within this research are as follows (Blazewicz, [3]; Gupta & Kypanrisis, [7]; Haupt, [8]).

1. All task parameters are deterministic and known a priori.
2. All jobs are available for processing at time zero.
3. Each job is independent of the other jobs.
4. Once a job is started on a machine it must continue to completion.
5. The only constraint is the machine and the machine is always available for job processing.
6. Job processing times include all ancillary processing times such as set-up, transportation, or operator time.

Another assumption is that computer resources are not a limiting factor. Our objective is to identify the scheduling rule, which results in a schedule that will provide the best total penalty performance, given various combinations of due date and deadline assignment.

This paper organized as follows: section 2 deals with the basic definitions for our criteria every one is independent to each other , in section 3 we propose general definition for the problem P ,some theorems ,some dominance rules ,present algorithm H and then numerical example, and in section 4 conclusions are presented.

2.Late work definition and completion time

In this paper, we consider a scheduling problem with the total late work criterion and sum of completion time .We'll use some simples to refers that:

p_j: processing time

d_j: due date time

C_j: completion time ($C_j=\sum_{i=1}^{j} p_i$)

L_j: lateness ($L_j=C_j-d_j$)

T_j: tardiness ($T_j=\max\{0,L_j\}$)

U_j: the number of tardy jobs($U_j=1$, if $C_j>d_j$, otherwise $U_j=0$).

We assume that the jobs are numbered in nondecreasing order of their due date (EDD order) so that $d_1 \le d_2 \le \ldots \le d_n$. We can easily compute the completion time C_j and its late work Y_j where Late work combines the features of two parameters: tardiness and the number of tardy jobs. Formally speaking, in the non-preemptive case

the late work parameter is defined as $Y_j=\min\{\max\{0, C_j-d_j\}, p_j\}=\min\{T_j, p_j\}$ or,in a more extensive way,as

$$Y_j = \begin{cases} 0 & \text{if } C_j \le d_j \\ C_j - d_j & \text{if } d_j < C_j < d_j+p_j \\ p_j & \text{if } d_j+p_j \le C_j \end{cases}$$

The late work Y_j is the amount of processing performed on job j after its due date . If $Y_j=0$, then job j is early; if $0 < Y_j < p_j$, then job j is partially early; if $Y_j=p_j$, then job j is late. The objective is to find a processing order of jobs which minimizes the sum of total late work and total completion time .

3.problem setting

Now we formulate our problem , where we have independent jobs and satisfy the conditions 1-6 from sec.1 above.

$$\text{Min } \{ \sum \mathbf{C_i} + \sum \mathbf{Y_i} \}$$

s.t.

$$\left.\begin{array}{ll} C_{\sigma(i)} \ge p_{\sigma(i)} & i=1,\ldots,n \\ C_{\sigma(i)} = C_{\sigma(i-1)} + P_{\sigma(i)} & i=2,\ldots,n \\ Y_i = \min\{ \max\{C_i - di, 0\}, p_i\} & i=1,\ldots,n \\ Y_i \ge 0 & i=1,\ldots,n \end{array}\right\} \quad \ldots\ldots\ldots \text{ P}$$

Before we solve this problem refers that we can write $Y=\sum Y_j$, and now gives some special cases ,theorems and dominance rule .

3.1 :Theorem 1 [5]

In the preemptive 1//Y problem ; $\alpha|\beta|L_{\max} \cong \alpha|\beta|Y$

3.2 Special cases

Case1: For problem P, if $T_{max}(EDD)=0$ then the optimal solution gives by "Smith backward algorithem[12]".

Proof: since $T_{max}=0$ then $Y=\min\{T_i,p_i\}=0$,but smith backward minimize $\sum C_i$ when $T_{max}=0$ hence we give minimum value for $\sum C_i+Y$.

Case2: For problem P, if $d_i=d$ for all i, then the SPT rule gives an optimal schedule for P.

Proof:

For SPT schedule $\sigma=\sigma_1,\sigma_2,\ldots,\sigma_n$

$Y_i=\min\{\max\{C_i-d,0\},p_i\}$

$Y_j=\min\{\max\{C_j-d,0\},p_j\}$

But $C_i \le C_j$ in σ then $Y_i \le Y_j$ and in the same time $\sum C_i \le \sum C_j$

3.3 Dominance rule:

D1: For problem P, if $p_i \le p_j$ for all j and the same time $d_i \le d_j$ for all j then the job i must be in the first postion in optimal schedule .

Proof:

Let we have σ_{ij} and σ_{ji} then

$p_i \le p_j$,then $C_i \le C_j$,but $d_i \le d_j$ then $T_i \le T_j$

$Y_i=\min\{\max\{C_i-d,0\},p_i\} \le Y_j=\min\{\max\{C_j-d,0\},p_j\}$

Therefore $p_{\sigma ij} \le p_{\sigma ji}$ for all j=2,…,n.

3.4 Algorithm H

step(0): Order the jobs by EDD rule and calculate T_{max} and $đ=d_j+p_j$.

step(1): If $T_{max}(EDD)=0$ use Smith backward method to find an optimal schedule and value for problem P.

Step(2): $U_e=U_T=U_l=\varphi$ where U_e set of early jobs,U_T set of tardy jobs,U_l set of late work,k=0,$N=\{J_1,\ldots,J_n\}$ un schedule jobs ,$C_0=0$.

Step(3): i=i+1, if $C_i \le d_i$ put J_i in U_e,$C_i=C_i+p_i$, else if $d_i<C_i<đ$ put J_i in U_T ,else put J_i in U_l.

Step(4):$N=N-\{J_i\}$, if i=n or $N=\varphi$ go to step 6,else go to step4.

Step(5): Order the jobs in U_e,U_T andU_l by SPT rule ; put $\sigma^H= U_e,U_T$,U_l and calculate $\sum C_i+Y$

3.5 Theorem 2: An algorithm H gives the best possible solution for problem P.

Proof:

Suppose we have another schedule σ in addition σ^H that satisfy algorithm H, we need to proof that $P(\sigma^H) \le P(\sigma)$.

For schedule σ_{ji}

$$Y(\sigma)= \sum_{k=1}^{j-1} Y_k + Y_j + Y_i + \sum_{k=i+1}^{n} Y_k \text{ and } \sum_{k=1}^{n} C_k = \sum_{k=1}^{j-1} C_k + C_j + C_i + \sum_{k=i+1}^{n} C_k$$

For schedule σ^{H}_{ij}

$Y(\sigma^{H})= \sum_{k=1}^{i-1} Y_k + Y_i + Y_j + \sum_{k=j+1}^{n} Y_k$ and $\sum_{k=1}^{n} C_k = \sum_{k=1}^{i-1} C_k + C_i + C_j + \sum_{k=j+1}^{n} C_k$

Now, $\sum C_k(\sigma^H) \le \sum_{k=1}^{n} C_k(\sigma)$.....1

since all jobs in σ^H ordered by SPT .

$\sum Y_k(\sigma) - \sum Y_k(\sigma^H) = \sum_{\sigma} \min\{\max\{C_i - d_i, 0\}, p_i\} - \sum_{\sigma^H} \min\{\max\{C_i - d_i, 0\}, p_i\}$

But $C_i(\sigma) \ge C_i(\sigma^H)$ then $\sum Y_k(\sigma) - \sum Y_k(\sigma^H) \ge 0, then \sum Y_k(\sigma) \ge \sum Y_k(\sigma^H)$.....2

From 1 and 2 we have σ^H gives best possible solution than σ □

3.5 numerical example :

let we have 6 jobs with p_i(3,6,7,7,8,9) and d_i(6,6,9,10,14,10) respectively .
the SPT is (1,2,3,4,5,6) with $\sum C_i$=122 and Y=34 then the sum is 156
the EDD is (1,2,3,4,6,5) with $\sum C_i$=123 and Y=34 then the sum is 157
now ,apply algorithm H,

K	N(EDD)	C_{k-1}	U_e	U_T	U_l
1	{1,2,3,4,6,5}	0	{1}	φ	φ
2	{2,3,4,6,5}	3	{1}	{2}	φ
3	{3,4,6,5}	3	{1}	{2,3}	φ
4	{4,6,5}	3	{1,4}	{2,3}	φ
5	{6,5}	10	{1,4}	{2,3}	{6}
6	{5}	10	{1,4}	{2,3,5}	{6}

U_e={1,4} ⟶ U_e(SPT)={1,4}
U_T={2,3,5} ⟶ U_T(SPT)={2,3,5}
U_l={6} ⟶ U_l(SPT) ={6}
Now, σ^H={ U_e,U_T,U_l } ={1,4,2,3,5,6} that have $\sum C_i$=123 and Y=30 then the sum is 153.

4. conclusions

In this paper we have developed algorithm to solve the classical single machine scheduling problem when the decision-maker evaluates alternative schedules using more than one criteria. If the user is interested only in aggregate performance measures (in this work: sum completion time with sum of late work), then, irrespective of the decision-maker's trade-offs between the criteria.

Questions related to the effective selection of a schedule from the

efficient set remain to be investigated. This work should include both decision theoretic approaches in which an attempt is made to identify the underlying preference function of the decision-maker and interactive approaches in which the decision-maker provides information sequentially that can be used to eliminate schedules as a part of the operation of the efficient set generation algorithm itself.

References:

[1] R. Bettati, D. Gillies, C.C. Han, K.J. Lin, C.L. Liu, J.W.S. Liu, W.K. Shih, Recent results in real-time scheduling, in: A.M. van Tilborg, G.M. Koob, eds., Foundations of Real-time Computing: Scheduling and Resource Management (Kluwer Academic Publishers, Boston, 1991) 129-156.

[2] J. Błażewicz, Scheduling preemptible tasks on parallel processors with information loss, Recherche Technique et Science Informatiques 3 (1984) 415-20.

[3] J.Blazewicz, Selected topics in scheduling theory. Annals of Discrete Mathematics, (1987) 31, 1-60.

[4] J. Błażewicz, G. Finke, Minimizing mean weighted execution time loss on identical and uniform processors, Information Processing Letters 24 (1987) 259-263.

[5] J. Błażewicz, E. Pesch, M. Sterna, F. Werner, open shop scheduling problems with late work criteria, discrete applied math. 134 (2004) ,1-24.

[6]. J. Błażewicz, E. Pesch, M. Sterna, F. Werner, Total late work criteria for shop scheduling problems, Operations Research Proceedings 1999 (Springer, Heidelberg, 2000) 354-359.

[7] S. K., Gupta, & Kyparisis, J. . Single machine scheduling research. OMEGA International Journal of Management Science, 15(3) (1987), 207-227.

[8] R. Haupt,. A survey of priority rule-based scheduling. OR Spektrum, 11(1989), 3-16.

[9] C.N. Potts, L.N. Van Wassenhove, Single machine scheduling to minimize total late work, Operations Research 40-3 (1991) 586-595.

[10] C.N. Potts, L.N. Van Wassenhove, Approximation algorithms for scheduling a single machine to minimize total late work, Operations Research Letters 11 (1992) 261-266.

[11] M. Sterna, Problems and Algorithms in Non-Classical Shop Scheduling (Scientific Publishers OWN, Polish Academy of Sciences, Poznań, 2000).

[12] L.N. Van Wassenhove, F.Gelders; Solving a bicriterion scheduling problem, European Journal of Operational Research 4(1980) 42-48.