Journal of Applied Sciences

ISSN 1812-5654

Journal of Applied Sciences 13 (21): 4616-4619, 2013
ISSN 1812-5654 / DOI: 10.3923/jas.2013.4616.4619

Mixed Dispatch Rule for Single Machine Total Weighted Tardiness Problem

Aihua Yin and Jing Wang
Software and Communication School, Jiangxi University of Finance and Economics,
330032, Nanchang, China

Abstract: The single machine total weighted tardiness scheduling problem has been discussed for many years and several effective constructive algorithms have been presented in literatures. The methods for solving this problem are applied among manufacture and logistics fields. This study proposes a quite new constructive algorithm, the Mixed Dispatch Rule (MDR) for solving the problem effectively and efficiently. What the mixed dispatch rule differs from the other dispatch rules is that it takes advantage of not only the jobs' characters values, such as, process time, due date and the weight but also the values of the objective function for different choices of some job. In fact, in according with the process order of the jobs, at any moment, the status of the unprocessed jobs may be different, i.e., some of them are delayed but others aren't. So, the characters of these two sorts of jobs are quite different and combining those characters with the objective function's value can obtain effective dispatch rule. The computing experiment is based on those instances in the classic OR-Library and the computational results show that the algorithm, MDR, is effective and efficient.

Key words: Single machine scheduling, dispatch rule, heuristic, constructive algorithm

INTRODUCTION

The single machine total weighted tardiness problem discussed in this study which was researched by Smith (1956) is stated as following. There are n jobs $J_1,\dots, J_n$ to be processed on exact one machine. p_i, d_i, w_i are processing time, due date and weight of job J_i. At any time, there is at most one job is processed by machine. Procession of each job can not be interrupted. The objective is to generate start times of each job that minimizes the Total Weighted Tardiness (TWT) which is defined as following:

$$TWT = \sum w_i \bullet \max\{C_i-d_i, 0\} \ (i = 1, 2, 3,\dots, n) \qquad (1)$$

where, C_i denotes the finish time of job J_i.

The single machine total weighted tardiness problem which was proved to be NP-hard (Lenstra *et al*., 1977) is of both highly theoretical and practical value. In real world production system such as semiconductor wafer fabrication facility and logistics engineering (Mason *et al*., 2002), the single machine total weighted tardiness problem is important. Algorithms for solving the single machine total weighted tardiness problem can be divided into two categories: Exact algorithms and approximation algorithms which are also named as heuristics.

Exact algorithms guarantee to generate optimal solution to any problem instance but they consume too much time. Hence, exact algorithms can not be accepted by real world workers. They can only be used to solve small-sized problem instances. The branch and bound algorithm proposed by Potts and Van Wassenhove (1985) can solve problem instances with up to 50 jobs in reasonable computation time.

On the other hand, some kinds of heuristic algorithms can often produce optimal schedules or near-optimal schedules in reasonable computation time. Constructive heuristics are among the first heuristic algorithms proposed for the single machine total weighted tardiness problem. Constructive heuristics generate a schedule after n iterations. Constructive heuristics are very fast but the gaps between the schedules by them and the optimal schedules are wide. Constructive heuristics are often adopted as first part of an algorithm to generate initial schedules. WSPT (Smith, 1956), ATC (Vepsalainen and Morton, 1987), WMDD (Kanet and Li, 2004) and H2, H3 (Yoon and Lee, 2011) are constructive heuristics.

Local search algorithms, such as presented seperately by Avci *et al.* (2003), Maheswaran and Ponnambalan (2003) and Ergun and Orlin (2006), can produce much better schedules than constructive algorithms by consuming more computation time. Local search algorithms start from an initial schedule which is

---

Corresponding Author: Aihua Yin, Software and Communication School, Jiangxi University of Finance and Economics, 330032, Nanchang, China

usually generated by a construction algorithm and then iteratively generate a sequence of improved schedules. Some meta-heuristics such as tabu search discussed by Bilge *et al.* (2007) and simulated annealing presented by Nearchou (2004) are improved local search algorithms.

This study is organized as following. Section 2 describes an effective constructive heuristic algorithm, the mixed dispatch rule. Section 3 tests our algorithm on benchmark instances and gives a brief analysis. Section 4 draws the conclusions.

MIXED DISPATCH RULE

To finish all these jobs, for a constructive algorithm, it is a natural and quite good method to process the jobs one by one. At initial time, all the jobs aren't processed and then the first one of the jobs is selected out to process. After a sequence of steps, some of the jobs are processed while the others not, at last all the jobs are processed. In fact, these four situations are called the initial situation, the first situation, some situation and the last situation. Figure 1 shows the second situation of an instance with 7 jobs, after job 3 and 6 are processed, the current time $t = p_3+p_6$.

At initial situation, start times of all jobs are not assigned. When old situation is developing into new situation, at the first beginning, a job with highest priority is chosen. Then the earliest possible start time of this job is computed out. Finally this job is assigned to start at its earliest possible start time. In this way, the start times of all jobs are assigned one by one so as to generate a schedule.

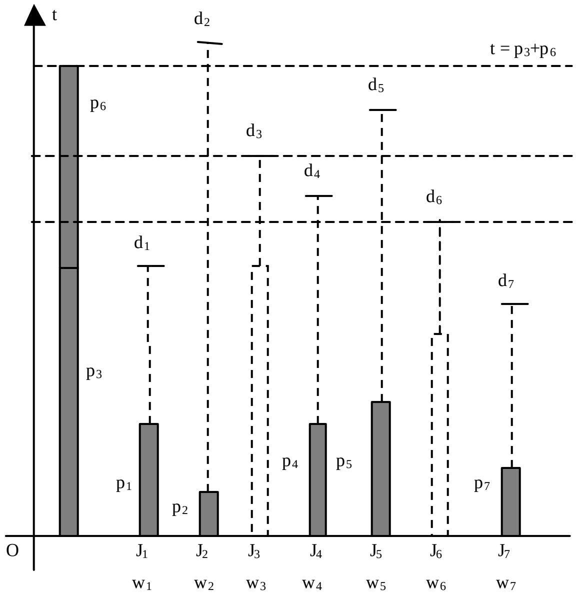

Fig. 1: Second situation of an instance with 7 jobs, after job 3 and job 6 are processed, the current time $t = p_3+p_6$

As the jobs are processed one by one, the situations are divided into three classes, i.e., class-1 without any unprocessed job delaying, class 2 with all unprocessed jobs delaying and class 3 with both delaying unprocessed jobs and un-delaying unprocessed jobs. It is easy to know a situation in class 1 or in 3 may be followed by a situation in any class but any situation in class 2 is followed only by another situation in class 2.

In the literatures on constructive algorithms (Kanet and Li, 2004; Lenstra *et al.*, 1977; Mason *et al.*, 2002), the authors adapted different effective rules to direct one situation to next one. All these rules take advantage of the job's characters, such as d, p and w. In this study, MDR takes into account both job's characters and the objective function's value.

In detail, if a situation is in class 1, the job with the smallest due date is selected to be processed and if a situation is in class 2, the Weighted Shortest Processing Time rule is certainly adapted for the theorem of Smith (1956) and if a situation is in class-2, two jobs are selected and then compare their incremental to the objective function's value, the smaller is selected to be processed right now. And this is the reason why our dispatch rule is mixed.

To describe our algorithm precisely and in detail, Let $J=\{J_1, J_2, \ldots, J_n\}$, t be the current time of schedule. So, t=0 at the initial time. MDR is as following:

Step 1: Set $JUD=\{J_k | \geq t+p_k,\ d_k\ J_k \in J\}$, $JD = \{J_k | d_k < t+p_k, J_k \in J\}$

Step 2: If $JUD \neq \varnothing$, $JD = \varnothing$, select $J_i \in J$ with the earliest due date. Go to step 5

Step 3: If $JUD = \varnothing$, $JD \neq \varnothing$, select $J_i \in J$ with the smallest p_i/w_i. Go to step 5

Step 4: If $JUD \neq \varnothing$, $JD \neq \varnothing$, $J_i \in JUD$ with the earliest due date, $J_j \in JD$ with the smallest p_j/w_j. Calculate $w_j \bullet (t+p_i+p_j-d_j)$ and $w_j(t+p_j-d_j)+w_i \bullet \max\{t+p_j+p_i-d_i, 0\}$:

- If $w_i \bullet (t+p_i+p_j-d_j) \geq w_j(t+p_j-d_j)+w_i \bullet \max\{t+p_j+p_i-d_i, 0\}$, select $J_j \in JD$, go to step 5
- If $w_i \bullet (t+p_i+p_j-d_j) < w_j(t+p_j-d_j)+w_i \bullet \max\{t+p_j+p_i-d_i, 0\}$, select $J_i \in JUD$, Go to step 5

Step 5: Let the selected job be J_l, set $t = t+p_l$

Step 6: Set $J = J \setminus \{J_l\}$, If $J = \varnothing$, stop. Else go to step 1

The complexity of MDR is $O(n^2)$, because the complexities of each of the steps of step 2, 3 and 4 is O(n).

Table1: MDR is compared with other three algorithms WMDD, H2, H3

		WMDD		MDR		H2		H3	
TF	Job num	Aver. gap (%)	NP	Aver. gap (%)	NP	Aver. gap (%)	NP	Aver. gap (%)	NP
0.2	40	309.80	7	37.74^*	15^*	48.82	14	311.92	9
	50	2938.74	10	41.83^*	10	457.60	15^*	3040.82	9
	100	1581.72	13	44.85^*	11	61.72	17^*	1368.11	13
0.4	40	104.72	0	49.17^*	5^*	86.48	1	102.48	0
	50	48.31	0	24.58^*	7^*	44.86	1	47.48	0
	100	147.11	1	24.44^*	4^*	133.59	1	180.37	0
0.6	40	3.45	0	4.20	2^*	4.95	0	3.19^*	2^*
	50	5.74	0	5.59^*	1^*	6.21	0	5.59^*	0
	100	7.21	0	6.58^*	1^*	7.25	0	6.83	0

*The black number with a star means the best value among the four values in the same line

However, both the job's characters and the objective function's value are taken into consideration, the following computing test shows that MDR is quite effective.

COMPUTING AND ANALYSIS

Our algorithm MDR is tested on benchmark instances presented in OR-Library (http://people. brunel.ac. uk/~mastjjb/jeb/info.html). The tested bench-mark instances are in the following three sets 1-3.

Set 1: 75 benchmark instances with exactly 40 jobs for each of them

Set 2: 75 benchmark instances with exactly 50 jobs for each of them

Set 3: 75 benchmark instances with exactly 100 jobs for each of them

These instances are randomly generated as follows. For each job J_i (i = 1, 2,..., n), an integer processing time p_i is generated from the uniform distribution [1, 100] and an integer processing weight w_i is generated from the uniform distribution [1, 10]. Instance classes of varying hardness are generated by using different uniform distributions for generating the due dates. For a given relative range of due dates} RDD (RDD = 0.2, 0.4, 0.6, 0.8, 1.0) and a given average tardiness factor} TF (TF = 0.2, 0.4, 0.6,0.8,1.0), an integer due date d_i for each job J_i is randomly generated from the uniform distribution

$$[P(1\text{-}TF\text{-}RDD/2), P(1\text{-}TF+RDD/2)] \qquad (2)$$

where, $P = \sum p(i)$ (i = 1,..., n). Five instances are generated for each of the 25 pairs of values of RDD and TF, yielding 125 instances for each value of n.

The algorithm MDR is compared with other three effective constructive heuristic algorithms named as WMDD (Kanet and Li, 2004), H2 and H3 (Yoon and Lee, 2011) which are among the best constructive algorithms up to now. Results are reported in Table 1. The gap (%) is expressed as $100.(TC_{Heu}\text{-}TC_{OPT})TC_{OPT}$, where TC_{Heu} denotes the value of the feasible solution that is generated by either one of the heuristics and TC_{OPT} denotes either the value of the optimal solution(if applicable) or the value of the best-known solution in the OR library. NP denotes the number of problems (out of 25) where the associated heuristic finds an optimal solution. If the denominator, TC_{OPT}, is 0, the gap (%) cannot be calculated. Hence, this study does not use problem instances with the value of the optimal solution is zero.

For each set of benchmark instances, H2, H3 or MDR, respectively got the best performance among the four algorithms. However, MDR gets the most times both in Aver. Gap(%) and in NP, even when TF = 0.2 it overwhelms the other three algorithms. Especially, MDR has obtained the optimal solution(the TWT value is 123893) of the 44 instance in set 2 for the first time as a constructive algorithm for this problem.

CONCLUSION

In this study, we presented an effective constructive heuristic algorithm for the single machine total weighted tardiness problem. Our algorithm has been tested on 225 benchmark instances which are available in OR-Library. Comparison with other constructive heuristic algorithm shows that this constructive heuristic algorithm is effective. For further research work, some other priority rules can be used when all the jobs are not delay, for example, select job from the two jobs, one with the smallest due date and another with the smallest weight due date.

ACKNOWLEDGMENTS

This research is supported by the National Natural Science Foundation of China No. 61262011 and the Natural Science Foundation of Jiangxi Province No. 2010GZS0065 and No. 20122BAB211023.

REFERENCES

Avci, S., M.S. Akturk and R.H. Storer, 2003. A problem space algorithm for single machine weighted tardiness problems. IIE Trans., 35: 479-486.

Bilge, U., M. Kurtulan and F. Kirac, 2007. A tabu search algorithm for the single machine total weighted tardiness problem. Eur. J. Oper. Res., 176: 1423-1435.

Ergun, O. and J.B. Orlin, 2006. Fast neighborhood search for the single machine total weighted tardiness problem. Oper. Res. Lett., 34: 41-45.

Kanet, J.J. and X. Li, 2004. A weighted modified due date rule for sequencing to minimize weighted tardiness. J. Scheduling, 7: 261-276.

Lenstra, J.K., A.H.G. Rinnooy Kan and P. Brucker, 1977. Complexity of machine scheduling problems. Ann. Discrete Math., 1: 343-362.

Maheswaran, R. and S.G. Ponnambalan, 2003. An investigation on single machine total weighted tardiness scheduling problems. Int. J. Adv. Manuf. Technol., 22: 243-248.

Mason, S.J., J.W. Fowler and W.M. Carlyle, 2002. A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops. J. Scheduling, 5: 247-262.

Nearchou, A.C., 2004. Solving the single machine total weighted tardiness scheduling problem using a hybrid simulated annealing algorithm. Proceedings of the 2nd IEEE International Conference on Industrial Informatics, June 26, 2004, Berlin, Germany, pp: 513-516.

Potts, C.N. and L.N. Van Wassenhove, 1985. A branch and bound algorithm for the total weighted tardiness problem. Oper. Res., 33: 363-377.

Smith, W.E., 1956. Various optimizers for single-stage production. Nav. Res. Logist. Q., 3: 59-66.

Vepsalainen, A.P.J. and T.E. Morton, 1987. Priority rules for job shops with weighted tardiness costs. Manage. Sci., 33: 1035-1047.

Yoon, S.H. and I.S. Lee, 2011. New constructive heuristics for the total weighted tardiness problem. J. Oper. Res. Soc., 62: 232-237.