Journal of the Operational Research Society (1997) 48, 751–755 0160-5682/97 $12.00

A tabu search approach for the single machine mean tardiness problem

A Islam and M Eksioglu
Wichita State University, USA

In this paper, a tabu search approach is proposed for solving the single machine mean tardiness scheduling problem. Simulation experiment results obtained from the tabu search approach and three other heuristics are compared. Although computation time is increased, the results indicate that the proposed approach provides a much better solution than the other three approaches.

Keywords: tabu search; heuristics; mean tardiness; single machine scheduling

Introduction

This paper presents a tabu search procedure for minimizing the mean tardiness on a single machine. Single-machine scheduling does not necessarily involve one machine. It can also involve a group of machines or a system that can be treated as one machine, such as serial production lines scheduled as a single unit. In addition, high-tech manufacturing facilities, such as computer controlled machining centres and robotic cells that can be treated as single machines for scheduling purposes are also often seen in industry. Minimizing the mean tardiness is one of the most important criteria in single-machine scheduling and it is NP-hard.[1]

Optimal solutions to the scheduling problem can be obtained through the enumeration methods, such as dynamic programming and branch-and-bound. These optimal techniques, however, may take a tremendous amount of computation time and require a powerful computer. Hence, efforts have been intensified in developing heuristic procedures.

Heuristics such as Fry *et al*,[2] Holsenback and Russell,[3] and simulated annealing have been the most successful for solving the one-machine scheduling problem so far. In this study, we propose and implement a tabu search approach for solving the one-machine scheduling problem where it is required to minimize mean tardiness. The results are compared with those of the three most successful heuristics.

Correspondence: Dr A Islam, Department of Industrial and Manufacturing Engineering, Wichita State University, Wichita, Kansas 67260-0035, U.S.A.
E-mail: maislam@imfge.twsu.edu

The description of the problem

The single-machine mean tardiness scheduling problem can be described as follows: at time zero, there exist n jobs simultaneously in a queue to be processed on a single machine. Each job has its processing time (p) and its due date (d). The goal is to find a processing sequence from the $n!$ possible sequences that minimizes the mean tardiness (T) defined as follows:

$$T = \left(\frac{1}{n}\right) \sum_{i=1}^{n} \max[0, C_i - d_i]$$

where

n = number of jobs
C_i = completion time; of ith job
d_i = due date of ith job

The following assumptions presented by Conway *et al*[4] and accepted by most researchers are adopted for the considered problem:

- p_i is an integer and is sequence independent,
- d_i is an integer measured from time zero when all jobs are available for processing,
- the machine is available continuously to process n jobs, and
- there are no setup times or time loss due to machine loading and unloading.

Algorithms for single machine scheduling

As stated earlier, the single-machine tardiness problem has been proven to be NP-hard and it causes exponential growth in computation time and storage requirements with increasing problem size. This is especially true when

the problem size is large ($n > 100$). Obtaining optimal solutions requires an enumerative approach such as branch-and-bound or dynamic programming to assure optimality. Due to the computational difficulties mentioned above, however, the optimal solutions are very difficult, if not impossible, to obtain. Therefore, the investigations are focused on developing good heuristic techniques to get a near optimal solution with a reasonable amount of computational effort. Some of these attempts are reviewed in the following paragraphs.

Wilkerson and Irwin[5] proposed one of the first heuristic procedures to solve the mean tardiness problem for the single-machine case by using an adjacent pair comparison strategy. Baker[6] suggested a dynamic programming-based algorithm for the jobs having precedence constraints.

Potts and Van Wassenhove[7] introduced a decomposition-based algorithm for a total tardiness problem. Lawler[8] developed a pseudo-polynomial algorithm to minimize the total tardiness. In this paper, among the enumerated approaches simulated annealing, Fry *et al* and Holsenback and Russell algorithms have been considered for comparison with the tabu search based heuristic.

Fry et al *heuristic*

The idea behind the Fry et al[2] algorithm was to explore a solution space by creating a new sequence with a hope that it may result in a local, although not necessarily global, optimum. They developed a very simple but effective algorithm based on the adjacent pairwise interchange (API) methodology to solve the single machine mean tardiness scheduling problem. The pairwise interchange strategy largely determines the quality of the new solution. The authors proposed three different API strategies that are used to transform three different initial sequences into local optimal sequences so as to select the best solution among the obtained sequences. If the interchange generates a favourable sequence, the switching is made and the process continues until there is no further favourable switching. Then a local optimum is found. The initial sequence also affects the solution quality. The three initial sequences that are evaluated by each of the API strategies are the earliest due date (EDD), the shortest processing time (SPT), and the smallest slack (SLK). Then the three best sequences obtained through the three initial sequences are compared. The one that results in the minimum mean tardiness is selected as the best final sequence. The probability of getting a better solution increases with the number of local optimums being evaluated.

Holsenback and Russell's heuristic

Holsenback and Russell[3] (HR) developed a heuristic method based on Emmons' corollary 2.2.[9] This principle stated that in EDD sequences a job should be assigned to the last position if it possesses a tardiness less than or equal to its processing time. The authors identified the jobs that have reducible tardiness. To select from the potential jobs for the last position, they developed an evaluation strategy based on net benefit of relocation (NBR). After relocation, the tardiness of the jobs following this particular job is reduced, whereas the tardiness of the job itself increased. This difference in tardiness is NBR. The algorithm selects a job with the largest positive NBR among the candidate jobs. This process starts from the end of the initial EDD sequence and continues until all the jobs of the sequence are considered. For the details of the algorithm, refer to Holsenback and Russell.[3]

Simulated annealing

The simulated annealing approach (SA), first applied to optimization problems by Kirkpatrick,[10] is a variant of the conventional descent methods for local optimization or neighbourhood search. It allows some uphill moves, in a controlled manner, to alleviate the problem of finding a local rather than global optimum.

In this approach, an initial sequence is generated using some heuristics or priority rules. The random API, that is, the random insertion of one job before another or interchange of two random jobs, is used to generate a new sequence. The new sequence is accepted if the mean tardiness is less than that of the previous sequence. Otherwise, the new sequence is accepted with a probability which decreases as the process continues. The algorithm is terminated when the new solution does not improve after some specified number of iterations or if it exceeds the maximum number of allowable iterations.

Potts and Van Wassenhove[11] also proposed a simulated annealing based algorithm for single machine mean tardiness problem. They generated the neighbourhood using all pair interchanges which makes their neighbourhood size $n(n-1)/2$. In our implementation, two randomly selected jobs are interchanged which makes our neighbourhood size 1. They have implemented a complicated procedure to calculate the probability of acceptance in comparison to our simple one shown below. Potts and Wassenhove[11] implemented an interweaving procedure. We did not.

In our implementation, the algorithm takes the following form when applied to the mean tardiness scheduling problem:

Initial solution: we have considered EDD, SPT, and SLK priority rules to generate initial sequences.

Neighbourhood generation: two random integer numbers between 1 and n are generated and jobs in this position are interchanged. For example, if the current sequence is 1-2-3-4-5-6 and the generated random integer numbers are 3 and

6, then new sequence is 1-2-6-4-5-3. The solution is accepted under the following condition:

$$P_a > r$$

where

$$P_a = e^{-\alpha \Delta t} = \text{acceptance probability}$$

and

α = control parameter which increases during the search

Δt = increase in mean tardiness

r = a random number from the uniform distribution [0, 1]

Several methods of varying the value of the parameter α have been investigated. The following model for changing α, however, gave the best results:

$$\alpha = 10 + \text{COUNT*STEP*2}$$

where COUNT is incremented in every iteration by 1 and STEP = 2.

Termination criterion: the algorithm is terminated at iteration = 2000.

Tabu search approach

Tabu search (TS), first introduced by Glover[12,13] as a technique for solving combinatorial optimization problems, is basically a strategy to overcome local optimality. Since then, TS has been successfully applied to a wide range of problems. Like simulated annealing, tabu search is also a 'steepest descent' approach for local optimization or neighbourhood search. Both SA and TS were developed to alleviate the shortcomings of hill-climbing. The TS algorithm starts with a feasible initial solution and chooses the best move according to the hill-climbing scheme. It, however, uses a *tabu list* (forbidden possible moves to discourage the reversal, and in some cases repetition, of selected moves) to force the search away from solutions selected for current iterations. This distinction tends to enable the search to escape local minima. Moves are rejected if they satisfy conditions given by the *tabu list* based on certain attributes of the most recent m moves. Solution vectors that are the result of moves having the attributes found on the *tabu list* are removed from consideration unless they meet an *aspiration criterion*. Therefore, the jobs are interchanged in a controlled manner so that the local optimum is avoided. If the new sequence improves the mean tardiness, it is accepted. The size of the *tabu list* is a major factor that greatly affects the solution. The algorithm terminates when a predetermined termination criterion is satisfied.

The elements of tabu search algorithm for the present implementation

For this study, the selection of the parameters is documented below:

Initial solution: modified EDD priority rule is used to generate the initial sequence.

Neighbourhood search: for problems with $n \leqslant 20$, API strategy is utilized. For problems with $n \geqslant 21$, random swapping is used as the switching strategy. In the implementation of this technique, neighbourhood size, N_s, varied from $3n$ to $n/4$ depending on the problem size (Table 1). For all the cases, selecting the best move was based on mean tardiness and this move was recorded as the new sequence.

Tabu list: if the tabu list size is too small, then algorithm may end up cycling and if it is too large, good solutions may more likely be skipped. Many researchers have suggested a tabu list size between 5 and 10. In our implementation, we have experimented with various tabu list sizes and 7, on average, produced better results in reasonable computation time. Therefore, a tabu list size of 7 was selected.

Aspiration criterion: any move which improves the performance measure is selected, even if the move is tabu.

Termination criterion: we experimented with different termination criteria and found that setting the number of allowable iterations equal to the number of jobs in the problem gave a reasonable compromise between solution quality and running time.

Figure 1 illustrates the flow chart of the tabu search algorithm utilized for this study.

Application details

In this research, the tabu search is compared with the heuristic approaches of Fry *et al*, Holsenback and Russell, and the simulated annealing algorithms. These algorithms are coded in FORTRAN 77 programming language and run on the same IBM 3090 mainframe computer. A total of 480 problems of size $n = 10$, 20, 30, 56, 70, 80 and 100 are generated as follows:

Table 1 Problem size and corresponding neighbourhood size

Problem size (n)	*Neighbourhood size (N_s)*
10	$3n$
20	$3n$
35	$2n$
56	n
70	n
80	n
100	$n/4$

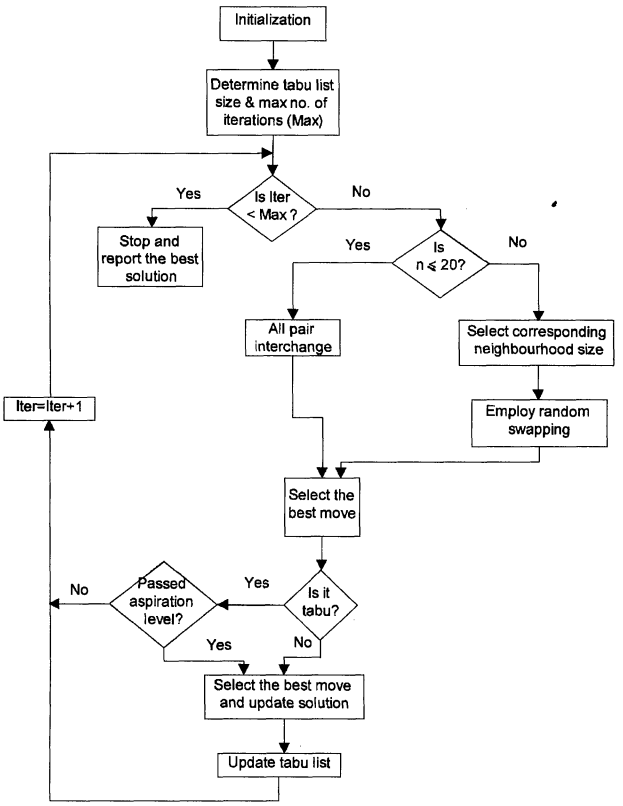

Figure 1 Flow chart for the tabu search algorithm.

Processing times (p_i): integer processing times for the n jobs are generated from a uniform distribution [1,100].

Due dates (d_i): to ensure the loose and tight due dates the following procedure developed by Potts and Wassenhove is adopted to generate random due dates. After the generation of n processing times the total processing time P is calculated as

$$P = \sum_{i=1}^{n} p_i$$

Then the due dates corresponding to n jobs are generated from the uniform distribution of

$$[P(1 - TF - RDD/2), P(1 - TF + RDD/2)]$$

where five replicates are taken for each setting of TF (mean tardiness factor) and RDD (range of due dates). The settings for TF and RDD are as follows:

TF	0.2	0.4	0.6	0.8
RDD	0.2	0.4	0.6	0.8

In this simulation study, all four algorithms described earlier are implemented to test the 480 problems. The implementation of the elements of the tabu search algorithm is done as follows:

1. Determine a feasible initial sequence using the modified EDD priority rule. If any job has a due date less than its processing time, its due date is changed to
$$d_{\text{mod}} = \max(p_i, d_i).$$
From this sequence the mean tardiness is determined.
2. Generate a new sequence by interchanging a pair of jobs in the initial sequence.
3. Check the tabu list to determine if this move is forbidden. If it is not tabu, update the best solution found so far and include the move in the tabu list. If the number of entries in the tabu list is greater than tabu list size, delete the oldest. If it is tabu, apply the aspiration criterion: if the current solution is better than the previous best solution then update this solution as the best solution so far and update the tabu list as before. If the aspiration criterion is not met than continue iteration.
4. If the number of iterations (iter) performed exceeds the maximum number of allowable iterations (max), stop and report the best solution. Otherwise continue iterations.

Computational results

The simulation results obtained from the tabu search and from the other algorithms under consideration are tabulated in Tables 2, 3 and 4. According to Table 2, the tabu search produces, for all number of jobs (n), a lower mean tardiness values than the other three algorithms. For instance, for problem size $n = 35$, the tabu search (TS) shows about 26%, 37%, and 9% decrease in mean tardiness when compared to Fry *et al* (F), Holsenback and Russell (HR), and the simulated annealing (SA), respectively. Simulated annealing results in lower mean tardiness values than the other two (F and HR) and overall F gives lower mean tardiness value than HR. Paired t-test results indicate that the differences in mean tardiness between the tabu search and the other three algorithms are significant at the 5% level.

From Table 3, it can be seen that the tabu search requires much more computation time and so is more expensive

Table 2 Comparisons of mean tardiness values for the algorithms

Problem size n	*Algorithms*				*Percentage improvement of tabu search (TS) over*		
	F	*HR*	*SA*	*TS*	*F*	*HR*	*SA*
10	52.56	58.81	54.97	50.08	4.71	14.84	8.89
20	92.23	124.69	92.67	79.70	13.59	36.08	14.00
35	175.22	193.60	142.82	130.06	25.77	36.82	8.93
56	289.00	298.78	233.62	184.84	36.04	38.16	20.88
70	339.37	359.31	275.78	228.91	32.55	36.29	17.00
86	470.24	466.78	393.55	314.82	33.05	32.55	20.00
100	485.82	511.83	375.33	293.56	39.57	42.65	21.79

Table 3 Average CPU times for algorithms with varying problem size

n	*F*	*HR*	*SA*	*TS*
10	0.70	0.60	1.20	1.11
20	0.84	0.69	1.52	6.72
35	1.20	0.76	2.15	7.20
56	2.03	1.03	3.46	13.04
70	2.78	1.48	3.60	7.27
86	3.85	1.57	4.71	11.38
100	4.95	2.38	4.92	18.27

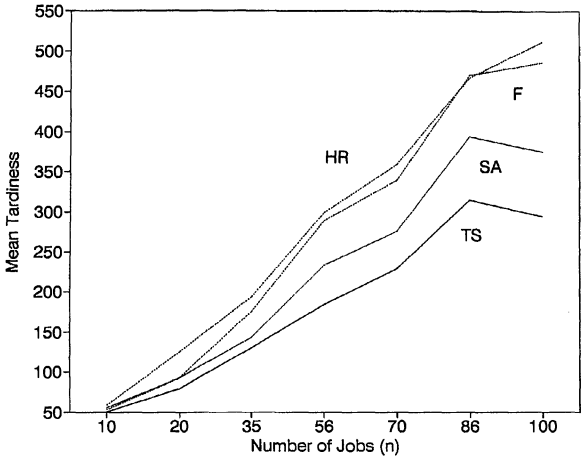

Figure 2 Comparison of mean tardiness.

when compared to the other three methods. The HR algorithm requires the least computation time.

Figures 2 and 3 graphically illustrate the mean tardiness and CPU times obtained from the four approaches through simulation.

From Table 4 it is clear that the tabu search performs much better than the other methods in comparison to the number of best solutions obtained.

Considering the simulation results, the tabu search should be preferred to the other three methods in minimizing the mean tardiness for the single machine case.

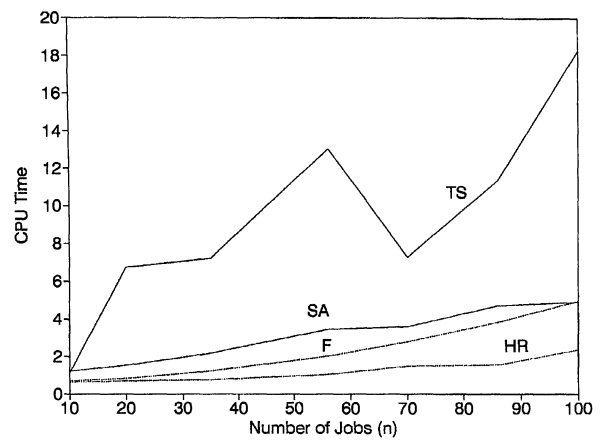

Figure 3 Average CPU times.

Table 4 Distribution of best solutions achieved by the algorithms

Problem size (n)	*F*	*HR*	*SA*	*TS*	*Total number of problems*
10	—	—	—	13	16
20	—	1	—	14	16
35	—	1	—	13	16
56	—	2	1	12	16
70	—	2	—	14	16
86	—	2	—	12	16
100	—	2	—	12	16

Conclusions

In this paper, a tabu search-based approach was proposed for solving the single machine mean tardiness scheduling problem. The results of a simulation experiment indicated that the proposed method provides much better solutions than the three best known heuristics for minimizing mean tardiness. The drawback of the tabu search was found to be a considerably higher computation time. Therefore, the tabu search algorithm appears to be a promising method to be utilized in sequencing jobs more efficiently.

References

1 Du J and Leung JY-T (1990). Minimizing total tardiness on one machine is NP-hard. *Math Opns Res* **15**: 483–495.
2 Fry TD, Vicens L, McLeod K and Fernandez S (1989). A heuristic solution procedure to minimize $\bar{T}$ on a single machine. *J Opl Res Soc* **40**: 293–297.
3 Holsenback JE and Russell RM (1992). A heuristic algorithm for sequencing on one machine to minimize $\bar{T}$ total tardiness. *J Opl Res Soc* **43**: 53–62.
4 Conway RW, Maxwell WL and Miller LW (1967). *Theory of Scheduling*. Addison-Wesley: Reading Mass.
5 Wilkerson J and Irwin JD (1971). An improved algorithm for scheduling independent tasks. *AIIE Trans* **3**: 239–345.
6 Baker KR (1974). *Introduction to Sequencing and Scheduling*. Wiley: New York.
7 Potts CN and Van Wassenhowe LN (1982). A decomposition algorithm for the single machine total tardiness problem. *Opns Res Lett* **32**: 177–181.
8 Lawler EL (1977). A pseudo time polynomial algorithm for sequencing jobs to minimize total tardiness. *Annals of Dis Math* **1**: 331–342.
9 Emmons H (1969). One machine sequencing to minimize certain functions of job tardiness. *Opns Res* **17**: 701–715.
10 Kirkpatrick S, Gelatt C and Vecchi M (1983). Optimization by simulated annealing. *Science* **220**: 671–680.
11 Potts CN and Van Wassenhove LN (1991). Single machine tardiness sequencing heuristics. *IIE Trans* **23**: 346–354.
12 Glover F (1989). Tabu search—Part I *ORSA J Computing* **1**: 190–206.
13 Glover F (1990). Tabu search—Part II. *ORSA J Computing* **2**: 4–32.

Received November 1995;
accepted February 1997 after two revisions