Journal of Intelligent Manufacturing (2000) **11,** 453–460

Tabu search for scheduling on identical parallel machines to minimize mean tardiness

VINÍCIUS A. ARMENTANO and DENISE S. YAMASHITA

Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, C.P. 6101 – CEP 13083-970, Campinas – SP, Brazil

This paper presents a tabu search approach for scheduling jobs on identical parallel machines with the objective of minimizing the mean tardiness. Initially, we consider a basic tabu search that uses short term memory only. Local search is performed on a neighborhood defined by two types of moves. Insert moves consist of transferring each job from one machine to another and swap moves are those obtained by exchanging each pair of jobs between two machines. Next, we analyze the incorporation of two diversification strategies with the aim of exploring unvisited regions of the solution space. The first strategy uses long term memory to store the frequency of the moves executed throughout the search and the second makes use of influential moves. Computational tests are performed on problems with up to 10 machines and 150 jobs. The heuristic performance is evaluated through a lower bound given by Lagrangean relaxation. A comparison is also made with respect to the best constructive heuristic reported in the literature.

Keywords: Scheduling, parallel machines, due date, tardiness, heuristic, tabu search

1. Introduction

In a highly competitive market, the tardiness of jobs with respect to their due dates is a very important performance measure for various manufacturing environments. The complexity and the short term decision-making nature of scheduling problems has led to a growing interest for heuristic methods which are able to produce good quality solutions in a reasonable time. In this paper, we discuss the use of tabu search for scheduling jobs on identical parallel machines in order to minimize the mean tardiness of jobs with respect to their due dates. Tabu search (Glover, 1989, 1990; Glover and Laguna, 1997) is a metaheuristic that has proven to be very effective for solving scheduling problems and other types of combinatorial problems. For parallel machine scheduling, tabu search has been applied to problems of minimizing the following performance measures: weighted earliness with deadlines (Laguna and Velaverde, 1991), weighted flow time (Barnes and Laguna, 1993) and makespan (Hübscher and Glover, 1994; França *et al*., 1995).

Despite the importance of the tardiness criterion, the literature on the problem is rather limited, as pointed out by Koulamas (1994) in a recent review. Optimal methods either deal with special cases of the problem (Lawler, 1964; Root, 1965; Elmaghraby and Park, 1974; Barnes and Brennan, 1977) or can handle only small problems (Dogramaci, 1984) in the general case.

Constructive heuristics proposed by Wilkerson and Irwin (1970), Dogramaci and Surkis (1979), Ho and Chang (1991) and Koulamas (1994) are based on the following framework:

Step 1 Sort the jobs using a priority rule.
Step 2 Assign the jobs to the machines according to some criterion or method.
Step 3 (Optional) Minimize total tardiness on each machine by means of an optimal or a heuristic method.

As an example, Dogramaci and Surkis (1979) use the earliest due date (EDD) priority rule to sort the jobs which are then assigned to the earliest available machine. A dynamic programming algorithm is applied to minimize total tardiness on each machine. Computational tests reported by Koulamas (1994) show that his heuristic, denoted KPM, presents a much better performance than the other heuristics cited above. Luh *et al*. (1990) present an integer programming formulation and a Lagrangean relaxation of capacity constraints which correspond to the number of parallel machines available. Subgradient optimization is then applied to obtain a lower bound on the value of the optimal solution. At the end of this procedure, a greedy heuristic is applied to the solution of the Lagrangean problem in order to construct a feasible solution. Guinet (1995) applies simulated annealing to solve the problem with uniform and identical machines and a lower bound is presented in order to evaluate the performance of the proposed method. In the case of identical machines, computational results are presented for problems involving up to 100 jobs and three machines. The solutions obtained have good quality, although, as the author points out, computational times are high.

In this paper we start from a solution generated by the KPM heuristic and then apply tabu search to obtain new and better solutions. Diversification strategies are included in order to guide the search to unexplored regions of the solution space. Computational tests are performed on 900 problems involving up to 150 jobs and 10 machines. Due to the high computational cost we were able to compute a lower bound, provided by Lagrangean relaxation, for 540 problems. The results show that for nearly 62% of such problems the solution provided by tabu search is within 1% of the lower bound. The KPM heuristic solutions are also evaluated against the lower bound and are compared with the solutions provided by tabu search.

2. Problem description

The problem addressed here consists of assigning n jobs to m identical machines and scheduling the jobs on the machines so as to minimize the mean tardiness. The processing times of the jobs and their due dates are given. Assume that all jobs are ready for processing at time zero and the machines are continuously available. Each machine can handle one job at a time and preemption is not allowed.

Consider the following notation:
d_i, due date of job i; C_i, completion time of job i; T_i, tardiness of job i, $T_i = \max\{0, C_i - d_i\}$.

Let S be a schedule with the following form

$$S = \{S_1, S_2, \ldots, S_m\}$$

where

$$S_k = \{J_k(1), J_k(2), \ldots, J_k(n_k)\}$$

is the sequence of jobs for machine k, $J_k(i)$ is the index of the job in position i on machine k and n_k is the number of jobs assigned to machine k, for $k = 1, 2, \ldots, m$ and $n = \sum_{k=1}^{m} n_k$.

Formally, the problem consists of finding a schedule S in order to

$$\text{Minimize} \qquad T(S) = \frac{1}{n}\sum_{k=1}^{m} T(S_k)$$

where

$$T(S_k) = \sum_{i=1}^{n_k} \max(0, C_{J_k(i)} - d_{J_k(i)})$$

is the total tardiness on machine k.

3. Initial solution

The initial solution is obtained by the KPM heuristic which can be described as follows:

Step 1 List all unscheduled jobs using the shortest processing time (SPT) priority rule. Generate m copies of this list (one for each machine).

Step 2 For each machine, determine the next job to be scheduled using the PSK heuristic, which was developed by Panwalkar *et al*. (1993) for minimizing total tardiness on a single machine. Choose the job that can be scheduled on its respective machine without being late and as close as possible to its due date. Otherwise, schedule the job yielding minimum tardiness. Delete such a job from the m copies of the list and repeat Step 2.

Alidaee and Gopalan (1997) have shown that the PSK heuristic is an implementation of the modified due date (MDD) priority rule proposed by Baker and Bertrand (1982).

4. Tabu search

Tabu search is a metaheuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality. The local procedure is a search that uses an operation called move to define the neighborhood of any solution. Tabu search reduces the neighborhood by classifying certain moves as forbidden or tabu. This mechanism prevents cycling, i.e., the indefinite execution of the same sequence of moves, and directs the search to regions not explored. An aspiration criterion removes the tabu condition of a move if it is considered attractive at that moment of the search.

One of the main features of tabu search is the use of adaptive memory to store key elements of the history of the search. The short term memory keeps track of move attributes that have changed during the recent past. This is the kind of memory encountered in most applications of tabu search in the literature. However, in general, higher quality solutions are produced when using diversification strategies in order to reach unexplored regions. Diversification can be achieved by including long term memory in order to stimulate attributes of solutions which are not frequently selected. Another diversification strategy makes use of influential moves which seek to modify the solution structure in such a way as to cross barriers in the solution space topology. Influential moves were used in parallel machine scheduling problems by Barnes and Laguna (1993) and Hübscher and Glover (1994).

In the sequel we identify the components for the short term memory of the tabu search method applied to our problem.

Neighborhood structure

At each iteration of the search procedure we choose the best of two types of moves: insert moves and swap moves. Insert moves consist of transferring each job from one machine to all positions of the other machines. The swap moves consist of exchanging each pair of jobs between two machines and inserting them in all positions.

The tardiness evaluation of the solutions generated by removing jobs from a machine k and inserting them on a machine ℓ can be done iteratively as follows. The removal starts from the job placed in the last position of machine k and proceeds to the job placed in the first position. Analogously, the insertion begins by placing each job in the last position of machine ℓ and continues until the first position. The evaluation of solutions reached by swap moves is similar, since such moves are a combination of two insert moves. It can be verified in the Appendix that the time complexity to evaluate the solutions generated by insert moves and swap moves is $O(n^2m)$ and $O(n^3m)$, respectively.

Tabu activation rule

Rules involving the prohibition of the reversal of a move did not work well. Best results were achieved by the use of the following restrictive rule: when a job shifts from one machine to a certain position in another machine as a result of an insert move or swap move, it must remain there for a number of iterations, i.e., any move involving such a job is tabu.

Tabu tenure

The number of iterations that a move remains tabu is selected from an interval $[a, b]$ with a uniform distribution. A new value is chosen at each iteration. This type of tabu tenure is found in many applications of tabu search and is motivated by the fact that a fixed tabu tenure, in general, does not allow a balance between exploitation of a region and exploration of new regions. It was observed in the search that insert moves are much less frequent than swap moves. For this reason we used a smaller interval for insert moves in order to give them an incentive. After extensive tests, good results were obtained with the following intervals

$$\begin{aligned} &\text{Insert moves:} && a = \max\{1, \lfloor 0.5n(m-1)/m - n/m \rfloor\} \\ &&& b = \lfloor 0.5n(m-1)/m + n/m \rfloor, \\ &\text{Swap moves:} && a = \max\{1, \lfloor 0.8n(m-1)/m - n/m \rfloor\} \\ &&& b = \lfloor 0.8n(m-1)/m + n/m \rfloor, \end{aligned}$$

where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. Note that for a fixed number of jobs, the mean number of iterations that a move remains tabu increases with the number of machines. This is reasonable since the opportunity of moving distinct jobs should increase for a larger number of machines.

The data structure used for the short term memory is a unidimensional vector with n positions, where position i stores the last iteration at which job i is forbidden to be moved. Suppose that k is the current iteration and a move involving job i is performed. Then tabu tenure $[i] = k + t$, where $t \in [a, b]$.

Aspiration criterion

We used the most common form, which consists of performing a tabu move if it results in a better solution than all previously found.

Henceforth, tabu search with short term memory only is called basic tabu search.

5. Diversification

Next we discuss the diversification strategies that were implemented.

Diversification with insert moves

As observed above, swap moves are much more frequent than insert moves. Swap moves do not change the number of jobs assigned to each machine and the search seems to focus on such moves trying to obtain better solutions without great perturbations. On the other hand, insert moves can be considered influential since they alter the number of jobs assigned to the machines, as pointed out by Barnes and Laguna (1993). An insert move modifies the solution structure and as a result, in general, the solution quality deteriorates. However, insert moves are necessary to reach unexplored regions which may contain good solutions. This diversification strategy was implemented in the following way. If, after 10 iterations, the incumbent solution value is not updated, swap moves are forbidden and the best non-tabu insert move is executed.

Frequency-based diversification

This strategy uses long term memory, which stores the frequency of the moves throughout the search. This measure is used to avoid moves with high frequency and stimulate moves with low frequency. A lower triangular transition frequency matrix of dimension $n \times n$ stores in the position (i,j) the number of times job i was exchanged with job j. The frequency of insert moves is stored in the diagonal of the matrix.

When diversification is activated, the evaluation of the moves is altered. For example, regarding a swap move (i,j):

$$\text{move\_value}'(i,j) = \text{move\_value}(i,j) + \alpha \cdot \text{penalty}(i,j)$$

where α is a diversifying factor and penalty (i,j) is the element (i,j) of the transition frequency matrix divided by the largest element of the matrix. The factor α was taken proportional to the initial solution value I, i.e., $\alpha = K \cdot I$, where $K = 0.3$ for a swap move and $K = 0.05$ for an insert move. In this way, insert moves are less penalized due to their low frequency.

The diversification is started after 60 iterations of the basic tabu search without updating the incumbent solution. The search is restarted from the incumbent solution, the components of the tabu tenure vector are set to zero, and the evaluation of the moves is modified as described above. The diversification stops if the incumbent is updated, or after 50 iterations.

The restart from the incumbent solution showed itself to be better than the current solution. Note that the diversification period should be long enough to move away from the region of the incumbent solution but not so long, as this usually gives rise to a great deterioration in the quality of the solution.

6. Computational tests

Basic tabu search and tabu search with the diversification strategies described above were implemented and tested on a set of 900 problems with number of jobs $n = 20, 50, 100, 150,$ and number of machines $m = 2, 3, 5, 7, 10$. The stopping criterion for all tabu search versions is 300 iterations or 1200 s of running time. The algorithms were coded in C language and the tests were carried out on a SUN Classic workstation.

The parameters were generated in the following way. Processing times are integer values generated in the interval [1,100] with uniform distribution. Due dates are integer values obtained from a uniform distribution between $P(1 - \tau - R/2)$ and $P(1 - \tau + R/2)$, as suggested by Potts and Van Wassenhove (1982), where: P is the sum of processing times of all jobs divided by the number of machines; τ is the tardiness factor and the higher its value the smaller is the distribution mean; R is the due date range factor which controls the distribution dispersion. Three values were chosen for these two factors: $\tau = 0.4$, 0.6, 0.8 and $R = 0.4, 0.7, 1.0$. The combination of these factor values generates nine problem types as shown in Table 1. For each type, five problems were randomly generated.

The results provided by the tabu search versions were compared with a lower bound given by

Table 1. Types of problems generated

Type	τ	R	*Interval*
1	0.4	0.4	[0.40P, 0.80P]
2	0.4	0.7	[0.25P, 0.95P]
3	0.4	1.0	[0.10P, 1.10P]
4	0.6	0.4	[0.20P, 0.60P]
5	0.6	0.7	[0.05P, 0.75P]
6	0.6	1.0	[0.00P, 0.90P]
7	0.8	0.4	[0.00P, 0.40P]
8	0.8	0.7	[0.00P, 0.55P]
9	0.8	1.0	[0.00P, 0.70P]

Lagrangean relaxation and subgradient optimization (Luh *et al*., 1990). For problems with 100 and 150 jobs, the time required to obtain a good lower bound is about 3 h. Due to this high time requirement, we were able to compute only one lower bound for each problem type involving 100 and 150 jobs and thus 540 lower bounds were computed.

The three tabu search versions exhibit similar performance. In 46.8% of the problems the solution value is the same. Tabu search with insert diversification outperformed the other versions in 23.7% of the problems, followed by tabu search with frequency-based diversification which was best in 17.8% of the problems and basic tabu search, which was superior in 11.8% of the problems. The average solution value relative difference between the three versions is below 0.5%.

Tables 2–5 show the performance of tabu search with insert diversification (TSID) relative to the initial solution and the lower bound. Column 1 indicates the number of machines and column 2 shows the mean tardiness of the initial solution. Column 3 presents the mean percentage improvement of TSID with respect to the initial solution, which is given by

$$\text{Improvement} = \frac{T_i - T_{\text{TSID}}}{T_i} \cdot 100$$

where T_i and T_{TSID} are the mean tardiness of the initial solution and the solution yielded by TSID, respectively. Column 4 indicates the mean percentage gap between the mean tardiness obtained by TSID and the lower bound (L), where

$$\text{Gap} = \frac{T_{\text{TSID}} - L}{L} 100$$

Column 5 shows the largest gap. For 41 problems the initial solution is zero and they were discarded when computing the mean results in column 2. For six problems, TSID obtained zero mean tardiness starting from a non-zero initial solution. This results in 100% improvement, irrespective of the initial solution value. For this reason these problems were not considered in the calculation of the mean results in column 3. Problems with zero lower bound are obviously disregarded in the computation of the mean results in column 4.

Note that, in general, the gap is small and it decreases as the number of machines increases. However, some problems have very large gaps as shown, for example, in Table 3 for 50 jobs and three machines, where there is one problem with a gap of 246.94%. It is possible that TSID fails to produce good results in such problems, but it is more likely that these problems have a large duality gap or else that the subgradient optimization was not able to yield a good lower bound.

Table 6 presents the percentage of solutions obtained by TSID and KPM heuristic methods in certain gap ranges. We assigned a gap of 10% for problems for which the lower bound is zero and TSID was not able to find a solution with mean tardiness equal to zero. Notice that the solution values obtained by TSID and KPM have a gap within 1% for 61.76% and 15.35% of the problems, respectively.

The presentation of the results by problem type would require 20 tables for all problem sizes (n, m)

Table 2. Results for 20 jobs

m	*Initial solution value*	*Mean improvement*	*Mean gap*	*Largest gap*
2	62.77	3.92	1.38	8.80
3	47.09	6.23	0.86	13.16
5	37.03	5.20	0.33	2.48
7	33.69	4.48	0.11	1.73
10	32.53	3.15	0.01	0.24

Table 3. Results for 50 jobs

m	*Initial solution value*	*Mean improvement*	*Mean gap*	*Largest gap*
2	123.52	5.60	6.66	129.73
3	88.40	5.46	11.96	246.94
5	57.95	8.64	4.54	66.00
7	45.85	8.16	6.72	217.65
10	38.30	5.64	1.18	7.12

Table 4. Results for 100 jobs

m	*Initial solution value*	*Mean improvement*	*Mean gap*	*Largest gap*
2	247.99	3.38	26.24	157.37
3	171.77	3.92	2.23	5.05
5	110.16	4.04	7.96	26.21
7	76.38	3.39	2.42	12.00
10	57.46	10.69	1.86	6.58

Table 5. Results for 150 jobs

m	*Initial solution value*	*Mean improvement*	*Mean gap*	*Largest gap*
2	368.31	2.98	12.35	35.52
3	253.04	3.26	7.21	20.42
5	158.98	3.72	5.51	23.06
7	118.76	3.63	1.86	4.94
10	84.29	4.47	2.08	5.91

considered here. For this reason we show results in Table 7 only for $n = 50$, $m = 7$ in order to analyze the performance of TSID with respect to the problem types. The mean gap in Table 7 represents an average over five problems. Note that, in general, larger gaps are associated with problem types 1, 2 and 3, which are characterized by looser due dates. At the same time, such problem types experience larger improvements, since the initial solution value is smaller than the other problem types. We have found, in general, the same behavior for most problem sizes (n, m) considered in this work.

Finally, Table 8 shows the average computational time, in seconds, spent by the TSDI method for each problem size. Note that the time diminishes as the number of machines increases and the number of jobs is fixed. In this case, the number of jobs per machine gets smaller and consequently, the neighborhood size is also reduced, which explains the time reduction. Note that this contrasts with the time complexity

Table 7. Results for seven machines and 50 jobs

Type	*Initial solution value*	*Mean improvement*	*Mean gap*	*Largest gap*
1	17.56	5.70	0.98	1.69
2	6.78	12.37	5.91	10.29
3	9.65	34.46	49.38	217.65
4	38.26	2.25	0.98	1.93
5	46.02	4.20	0.83	1.22
6	35.16	8.40	1.67	2.45
7	84.68	1.32	0.28	0.43
8	88.88	1.58	0.16	0.25
9	85.68	3.12	0.27	0.44

Table 8. Computational times

m	*n*			
	20	50	100	150
2	4.5	60.5	515.7	1200.0
3	4.4	53.4	470.2	1200.0
5	3.2	33.2	216.3	888.8
7	2.8	22.3	148.1	428.4
10	2.7	16.3	97.9	229.5

$O(n^2m)$ and $O(n^3m)$ in order to evaluate the solutions generated by insert moves and swap moves, where we assume n jobs assigned to each machine, a case that obviously does not occur.

7. Conclusion

The use of tabu search to minimize the mean tardiness on parallel machines has proven to be very effective. Tabu search without diversification yielded good results for most problems. In addition, the inclusion of diversification strategies tends to produce better results for the same stopping criterion. In particular, influential diversification in parallel machine scheduling, by using insert moves, is very attractive since

Table 6. Gap of the heuristic solutions

	$gap = 0$	$0 < gap < 1$	$1 \leq gap < 2$	$2 \leq gap < 3$	$3 \leq gap < 5$	$5 \leq gap < 10$	$gap \geq 10$
TSID	24.76	37.04	17.66	4.80	5.95	4.22	5.57
KPM	3.26	12.09	15.35	9.40	18.62	18.62	22.66

it yields very good results and is easily implemented. This tabu version was tested on 900 problems and an average gain ranging from 2.98% to 10.69% was achieved over the best constructive heuristic reported in the literature. A lower bound provided by Lagrangean relaxation was computed for 540 problems. The results showed that for nearly 62% of such problems the solution provided by tabu search is within 1% of the lower bound. A very interesting extension of this work would consist of establishing a strategy to evaluate a reduced list of candidates in the neighborhood so as to obtain high quality solutions in a faster computational time.

Acknowledgments

This research was partially funded by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and the Fundação Coordenação de Aperfeiçoamento de Pessoal de Ensino Superior (CAPES).

Appendix

Procedure Insert_Move

i^*, j^*, k^* = store the best move : transfer job $J_i*(k^*)$ from machine M_i* to machine M_j*; best_value = best value move;
Best_Position$(J_i(k), M_j)$ = computes the best insert position of job $J_i(k)$ on machine M_j; value = value of a move (auxiliary variable).

Step 1 $i^* = j^* = k^* = 0$;
best_value $= \infty$;
Step 2 $i = 1$;
Step 3 If $i > m$, stop.
otherwise, set $k = 1$;
Step 4 If there is no job $J_i(k)$ at the position k of machine M_i, go to Step 8;
otherwise, set $j = 1$;
Step 5 If $j > m$, go to Step 7;
otherwise: value = Best_Position$(J_i(k), M_j)$;
if value < best_value
best_value = value;
$i^* = i$; $j^* = j$; $k^* = k$;
Step 6 Set $j = j + 1$, go to Step 5;
Step 7 Set $k = k + 1$, go to Step 4;
Step 8 Set $i = i + 1$, go to Step 3.

In Step 5 the procedure Best_Position selects the best insertion position of job $J_i(k)$ on machine M_j. For this, each position is evaluated and thus its complexity is $O(n)$. For each job $J_i(k)$ on machine M_i, Steps 5 and 6 are repeated m-1 times and hence their complexity is $O(nm)$. Since in the worst case n jobs are assigned to M_i Steps 4–7 are repeated n times. As i ranges from 1 to m-1 the complexity of the Insert_Procedure is $O(n^2m^2)$.

Procedure swap_move

i^*, j^*, k^*, l^* = store the best move : exchange job $J_{i^*}(k^*)$ on machine M_{i^*} with job $J_{j^*}(l^*)$ on machine M_{j^*}.
Best_Position$(J_i(k), J_j(l), M_i, M_j)$ = computes the best swap between job $J_i(k)$ on M_i and job $J_j(l)$ on M_j.

Step 1 $i^* = j^* = k^* = l^* = 0$; best_value $= \infty$;
Step 2 $i = 1$;
Step 3 If $i = m$, stop.
otherwise, set $j = i + 1$;
Step 4 If $j > m$, go to Step 10;
otherwise, set $k = 1$;
Step 5 If there is no job $J_i(k)$ at position k of machine M_i, go to Step 9;
otherwise, set $l = 1$;
Step 6 If there is no job $J_j(l)$ at position l of machine M_j, go to Step 8;
otherwise: value = Best_Move $(J_i(k), J_j(l), M_i, M_j)$;
if value < best_value
best_value = value;
$i^* = i$; $j^* = j$; $k^* = k$; $l^* = l$;
Step 7 Set $l = l + 1$, go to Step 6;
Step 8 Set $k = k + 1$, go to Step 5;
Step 9 Set $j = j + 1$, go to Step 4;
Step 10 Set $i = i + 1$, go to Step 3.

In Step 6 the procedure Best_Position has complexity $O(n)$ and is performed for each pair of jobs $J_i(k)$ and $J_j(l)$ and each pair of machines M_i and M_j. Therefore, the number of repetitions of this step is $O(n^2m^2)$ and the complexity of the Swap_Procedure is $O(n^3m^2)$.

In a given iteration, with the exception of the first one, the choice of the best swap and insert moves involves only two machines, say M_i and M_j. It follows that in the next iteration it is necessary to recalculate

only the moves associated with pairs of machines involving M_i or M_j. By using a suitable data structure to store the move values, the procedure Best_Position is called only $O(m)$ times for each job in the Insertion_Move procedure and also $O(m)$ times for each pair of jobs in the Swap_Move procedure. Therefore, the complexity of such procedures is reduced to $O(n^2m)$ and $O(n^3m)$, respectively.

References

Alidaee, B. and Gopalan, S. (1997) A note on the equivalence of two heuristics to minimize total tardiness. *European Journal of Operational Research*, **96**, 514–517.

Baker, K. R. and Bertrand, J. W. M. (1982) A dynamic priority rule for scheduling against due-dates. *Journal of Operations Management*, **3**, 37–42.

Barnes, J. W. and Brennan, J. J. (1977) An improved algorithm for scheduling jobs on identical machines. *AIIE Transactions*, **9**, 25–31.

Barnes, J. W. and Laguna, M. (1993) Solving the multiple-machine weighted flow time using tabu search. *IIE Transactions*, **25**, 121–128.

Dogramaci, A. and Surkis, J. (1979) Evaluation of a heuristic for scheduling independent jobs on parallel identical processors. *Management Science*, **25**, 1208–1216.

Dogramaci, A. (1984) Production scheduling of independent jobs on parallel identical processors. *International Journal of Production Research*, **22**, 535–548.

Elmaghraby, S. E. and Park, S. H. (1974) Scheduling jobs on a number of identical machines. *AIIE Transactions*, **6**, 1–13.

França, P. M., Gendreau, M., Laporte, G. and Müller, F. M. (1995) The *m*-traveling salesman problem with minimax objective. *Transportation Science*, **29**, 267–275.

Glover, F. (1989) Tabu Search: Part I. *ORSA Journal on Computing*, **1**, 190–206.

Glover, F. (1990) Tabu Search: Part II, *ORSA Journal on Computing*, **2**, 4–32.

Glover, F. and Laguna, M. (1997) *Tabu Search*, Kluwer Publishers, Boston.

Guinet, A. (1995) Scheduling independent jobs on uniform parallel machines to minimize tardiness criteria. *Journal of Intelligent Manufacturing*, **6**, 95–103.

Ho, C. J. and Chang, Y. L. (1991) Heuristics for minimizing mean tardiness for *m* parallel machines. *Naval Research Logistics*, **38**, 367–381.

Hübscher, R. and Glover, F. (1994) Applying tabu search with influential diversification to multiprocessor scheduling. *Computers and Operations Research*, **21**, 877–884.

Koulamas, C. P. (1994) The total tardiness problem: review and extensions. *Operations Research*, **42**, 1025–1041.

Laguna, M. and Velaverde, J. L. G. (1991) A search heuristic for just-in-time scheduling in parallel machines. *Journal of Intelligent Manufacturing*, **2**, 253–260.

Lawler, E. L. (1964) On scheduling problems with deferral costs. *Management Science*, **11**, 280–288.

Luh, P. B., Hoitomt, D. J., Max, E. and Pattiti, K. R. (1990) Schedule generation and reconfiguration for parallel machines. *IEEE Transactions on Robotics and Automation*, **6**, 687–696.

Panwalkar, S. S., Smith, M. L. and Koulamas, C. P. (1993) A heuristic for the single machine tardiness problem. *European Journal of Operational Research*, **70**, 304–310.

Potts, C. N. and Van Wassenhove, L. N. (1982) A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters*, **5**, 177–181.

Root, J. G. (1965) Scheduling with deadlines and loss functions on k parallel machines. *Management Science*, **11**, 460–475.

Wilkerson, L. J. and Irwin, J. D. (1970) An improved algorithm for scheduling independent tasks, Technical Report AU-T-15, Digital Systems Laboratory, Auburn University, Auburn, Alabama, in Baker, K. R. (1973) Procedures for sequencing tasks with one resource type. *International Journal of Production Research*, **11**, 125–138.