Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 176 (2007) 1423–1435
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

A tabu search algorithm for the single machine total weighted tardiness problem

Ümit Bilge *, Müjde Kurtulan, Furkan Kıraç

Department of Industrial Engineering, Boğaziçi University, Bebek, 80815 İzstanbul, Turkey

Received 5 March 2003; accepted 7 October 2005
Available online 18 January 2006

Abstract

In this study, a tabu search (TS) approach to the single machine total weighted tardiness problem (SMTWT) is presented. The problem consists of a set of independent jobs with distinct processing times, weights and due dates to be scheduled on a single machine to minimize total weighted tardiness. The theoretical foundation of single machine scheduling with due date related objectives reveal that the problem is NP-hard, rendering it a challenging area for meta-heuristic approaches. This paper presents a totally deterministic TS algorithm with a hybrid neighborhood and dynamic tenure structure, and investigates the strength of several candidate list strategies based on problem specific characteristics in increasing the efficiency of the search. The proposed TS approach yields very high quality results for a set of benchmark problems obtained from the literature.

Keywords: Tabu search; Single machine scheduling; Total weighted tardiness minimization

1. Introduction and literature survey

One of the scheduling problems, simple to state but not easy to solve, is the problem of sequencing n independent jobs on a single machine with the objective of minimizing total weighted tardiness. The machine processes only one job at a time and without interruption. Each job i has an integer processing time p_i, a positive weight w_i and a distinct due date d_i. For a given processing order of the jobs, the earliest completion time C_i and tardiness $T_i = \max\{0, C_i - d_i\}$ can be computed for each job. The problem is represented as $n/1/\sum w_i T_i$ and referred as single machine total weighted tardiness problem (SMTWT).

The SMTWT problem is NP-hard (see Lawler, 1977; Lenstra et al., 1977; Du and Leung, 1990) and solution approaches like Dynamic Programming and Branch-and-Bound are computationally

* Corresponding author. Tel.: +90 212 359 6407; fax: +90 212 265 1800.
E-mail address: bilge@boun.edu.tr (Ü. Bilge).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2005.10.030

inefficient, especially when the number of jobs is beyond 50, as the results presented in a comparative study by Abdul-Razaq et al. (1990) demonstrate. Special sub-models are investigated to identify polynomially solvable cases; some recent works to this end are by Cheng et al. (2005) and Tian et al. (2005). It is also well known that there is no simple dispatching rule that works best for all problem environments. If there is no more than one tardy job, then the earliest due date (EDD) sequence is optimal, whereas weighted shortest processing time (WSPT) order gives the optimal sequence when all jobs are necessarily tardy. Therefore, EDD generally performs well for lightly loaded machines while WSPT should be preferred under heavy loading. Several heuristic dispatching rules like those developed by Carroll (1965), Montagne Jr. (1969), Rachamadugu and Morton (1982), Morton et al. (1984), and Panwalkar et al. (1993) have been based on this idea. As the reviews by Koulamas (1994) and Chen et al. (1998) indicate developing approximation algorithms with good performance guarantees for this problem is difficult.

The quest for good and robust heuristics, therefore, continued with sophisticated approaches like meta-heuristics which extend neighborhood search beyond local optima. Matsuo et al. (1989) address the SMTWT by a simulated annealing algorithm which starts with a good initial solution and low acceptance probability to accelerate the search for a near optimal solution. Potts and Van Wassenhove (1991) propose a descent heuristic and a simulated annealing method for SMTWT. Crauwels et al. (1998) present single and multi-start versions of descent, simulated annealing, tabu search (TS) and genetic algorithm implementations for the same problem and show that while simulated annealing is outperformed, tabu search dominates the other methods. Congram et al. (2002) treat SMTWT with an 'iterated dynasearch' algorithm, which is a local search technique that uses dynamic programming to find the best move which is composed of a set of independent interchange moves and searches an exponential size neighborhood in polynomial time. They obtain results that are superior to other local search procedures. Laguna et al. (1991) consider a single machine scheduling problem for minimizing the sum of setup costs and linear delay penalties, and propose a TS algorithm that uses hybrid neighborhood consisting of both swap and insertion moves. Bilge et al. (2003) develop a totally deterministic TS algorithm for parallel machine total tardiness problem (PMTT), and report results that are much superior on the benchmarks available in the literature. They use a hybrid neighborhood generation technique with different context-dependent candidate list strategies and report that the time performance and the quality of the results obtained under candidate list strategies are superior to the case when no candidate list strategy is employed.

TS algorithm proposed in this paper also employs a hybrid neighborhood along with a dynamic tenure structure, however both the hybrid neighborhood and the dynamic tenure structures are different from the PMTT case; and the theoretical foundation of single machine scheduling is exploited to develop new effective candidate list strategies. The next section presents the details of the tabu search algorithm implemented. The performance of the proposed algorithm is tested over the problem set used by Crauwels et al. (1998) and presented in Section 3. Finally, conclusions are discussed in Section 4.

2. Tabu search approach to SMTWT

Tabu search (Glover and Laguna, 1997) is a meta-heuristic that guides a local search procedure to explore the solution space beyond local optimality. TS allows intelligent problem solving by the incorporation of *adaptive memory* and *responsive exploration*. Key elements of the search path are selectively remembered and strategic choices are made to guide the search out of local optima and into diverse regions. The adaptive memory usage is a clever compromise between the rigid memory structure of exact techniques like Branch-and-Bound and the memoryless heuristics like local search procedures.

The basic procedure of TS can be summarized as follows. Starting from an initial solution, TS iteratively moves from the current solution to its

best neighbor, even if this new solution is worse than the one available, until a pre-specified stopping criterion becomes true. However, considering every possible move from the current solution may be extremely time consuming and computationally expensive and devising a *candidate list strategy* to isolate good candidates from the neighborhood may be helpful. In order to avoid cycling and becoming trapped in local optima, certain moves (or solution attributes) that lead to previously explored regions are forbidden or classified as *tabu*, forming the short-term memory of TS. The tabu status of a move may be cancelled making it an allowable move if an *aspiration criterion* is satisfied (if, for instance, the tabu move leads to a new best solution). The length of time during which a certain move is classified as tabu, *tabu tenure*, is an important parameter for tabu search. Tabu tenure, can be kept constant or varied dynamically throughout the search. TS cannot always succeed to direct the search into the regions where the best solutions are found by solely employing short-term memory strategies. Hence, long-term memory strategies to lead the search into unexplored regions of the solution space (*diversification*) or to perform a more thorough examination in some good or promising regions (*intensification*) may be needed. These key aspects of the totally deterministic TS algorithm tailored to SMTWT in this paper are described below.

2.1. Solution representation and initial solution generation

A solution for the SMTWT is represented as a permutation of integers $1,\ldots,n$, which defines the processing order of jobs on the single machine. Four methods are used to generate starting solutions. These are:

1. EDD based list scheduling, where jobs are ordered with respect to their earliest due dates.
2. WSPT, where the jobs are sequenced such that $(w_1/p_1) \geqslant (w_2/p_2) \geqslant \cdots \geqslant (w_n/p_n)$.
3. R&M heuristic developed by Rachamadugu and Morton (1982) (also known as Apparent Urgency Rule), which is based on sorting the jobs in order of non-increasing priorities, π_i, where

   $$\pi_i = (w_i/p_i)[\exp\{-(S_i)^+/kp_{\mathrm{av}}\}].$$

   Here, $S_i = d_i - p_i - t$ is the slack time of job i at time t. If the slack is negative, the job is sure to be tardy and receives full WSPT priority. Typical value of the factor k is suggested as $k = 2$, and p_{av} is the average processing time of all the jobs to be scheduled.
4. R&M+ heuristic, which is obtained by searching for the k value in R&M yielding the lowest $\sum w_i T_i$. Hence, a schedule is generated using each of the k values in the range $[0.5, 4.0]$ with increments of 0.1, and the best schedule thus obtained is used as the initial solution.

2.2. Neighborhood generation

Insert moves and pairwise exchanges (swaps) are frequently used move types for neighborhood generation in permutation problems. An insert move identifies two jobs i and j in the current solution and moves job i to be between job j and its direct predecessor. A swap move exchanges the locations of two jobs so that each job is placed in the location previously occupied by the other. A swap move can be considered as a move that combines two insert moves. The neighborhood used in this study is a hybrid one that consists of the complete "insert neighborhood" and the complete "swap neighborhood".

2.3. Tabu classification

Attribute-based tabu classifications contain more information about an already visited solution and generally considered more effective in preventing cycling as opposed to move-based (i.e. prohibit moving a job directly involved in a recent move) tabu classifications (Glover and Laguna, 1997). This is the case for SMTWT as well, as verified by our preliminary experiments. Therefore, the tabu classification used is based on the arcs that appear in a solution. If in a given permutation job i is directly preceded by job $i-1$ then this is referred as arc $((i-1), i)$ as shown in Fig. 1. The

Fig. 1. Arc-based solution representation.

tabu classification aims to prohibit some of the arcs broken by recent moves to be constructed again during tabu duration. The swap and insert moves are handled separately and the arcs to be declared as tabu active after each type of move are determined through some experimentation. The resulting tabu classification strategy works as follows: If the move just performed is to insert job i between job $j-1$ and job j, all permutations in which arc $((j-1),j)$ or arc $((i-1),i)$ appear are classified as tabu. When jobs i and j are swapped, three of the arcs that are broken, namely arcs $((j-1),j)$, $((i-1),i)$, and $(j,(j+1))$ become tabu active. This requires all the new arcs formed by a candidate insert or swap move to be checked and the move is tabu if any of these arcs are tabu active. This arc-based tabu classification is stricter in its classification of tabu compared to the one used in Bilge et al. (2003), and therefore called Arc Check Strict Tabu Classification (ARC+). The tabu status of a move is to be overridden if it yields a solution better than the best obtained so far.

2.4. Candidate list strategies

The candidate list strategies (CLS) developed are based on problem specific characteristics. In this study, a CLS can use two basic types of constraints in screening the neighborhood. The first is based on the nature of the job to be selected for an insert or swap move. This is called the *job heuristic*. If only tardy jobs are allowed to initiate a move, the job heuristic is called "TARDY" and the CLS works as follows: Take a tardy job i and consider all inserts and swaps with jobs that currently precede it (not necessarily directly). If on the other hand tardiness restriction does not apply, the job heuristic is "NONE".

The second type of constraint applies to the set of allowable moves for a given job. This is called the *move heuristic*. The move heuristics are based on theoretical results on SMTWT, namely on the following two corollaries by Rinnooy Kan (1976):

Corollary 1. *For the SMTWT problem, $n/1/\sum w_iT_i$, if for any two jobs i and j, the three conditions stating that $\{d_j \leqslant d_i, w_j \geqslant w_i, p_j \leqslant p_i\}$ are all satisfied, then only those schedules in which "job j precedes job i" need to be considered.*

Corollary 2. *For the SMTWT problem, $n/1/\sum w_iT_i$, if for any two jobs i and j, the three conditions stating that $\{d_j \leqslant C_i, w_j \geqslant w_i, p_j \leqslant p_i\}$ are all satisfied, then only those schedules in which "job j precedes job i" need to be considered.*

The move heuristic is categorized as type A or type B depending on whether it is based on Corollaries 1 or 2. The candidate list strategy A-OR works as follows: Take a job i (as in Fig. 1) and consider all inserts and swaps with job j which currently precedes it (not necessarily directly) as long as at least one of the conditions in Corollary 1 does not hold. Hence, if ($d_i < d_j$ OR $w_i > w_j$ OR $p_i < p_j$) is true then job j does not *have to* precede job i and job i can be inserted before job j or swapped with job j. The candidate list strategy B-OR is the counterpart of this for Corollary 2.

Another type of move heuristic is devised for bringing in some diversifying effect against the aggressive nature of TS which usually leads the search into a local optimum in the most direct way. The Candidate List Strategy A-Diverse (A-DIV) works as follows: Take a job i (as in Fig. 1) and consider all inserts and swaps with job j which currently precedes it (not necessarily directly) as long as the conditions $\{d_i \leqslant d_j, w_i \geqslant w_j, p_i \leqslant p_j\}$ are *not all* satisfied simultaneously. Here, if job i *does* need to precede job j according to Corollary A, this move is not allowed to be performed explicitly (other moves may lead into it indirectly). Although this may seem somewhat unintuitive at first glance, it does create a different search path that turns out to perform well in many cases. Again, the counterpart for Corollary 2 is called Candidate List Strategy B-Diverse (B-DIV).

It is possible to exploit any combination of the job/move heuristics, since they can be superimposed on one another, e.g. TARDY-A-OR. If only a job heuristic is applied, and no move heuristic is used, then the CLS imposed becomes "Tardy Candidates List Strategy" and is denoted as TARDY-NONE. The choice of not using any of these candidate list strategies leads to the consideration of the entire neighborhood. This case is denoted as "No CLS" or "NONE" in short and is used mainly for benchmarking the other CLSs developed in terms of computational speed and solution quality.

2.5. Tabu tenure

In this study, both the single and dynamic tenure approaches are tested. The dynamic tenure strategy used creates a sequence of small (S), medium (M) and large (L) tabu tenure values repeated in a sequence called cycle string, throughout the search. Systematically varying the tabu tenure in this way results in a balance between intensification and diversification, since short tabu tenures allow fine-tuning of neighborhood search and close examination of regions around a local optimum, while long tenures tend to direct the search to different parts of the solution space (Glover and Laguna, 1997). Due to the dynamic tabu tenure and nature of the CLSs, no other diversification strategies are developed. However, an intensification strategy is employed after the short-term memory TS is completed.

2.6. Intensification strategy

The intensification strategy consists of recording a fixed length sequential list of elite solutions during the short-term-memory search phase, then, after clearing the memory, restarting and carrying out a search of given duration from each of those elite solutions, with a given CLS and a set of tenure values that depend on the CLS. An elite solution is defined as a solution that can survive as the best-so-far for a given number of iterations (Bilge et al., 2003). All of the CLS combinations described in Section 2.5 can be employed in the intensification phase; however, complementing CLSs should be employed in the short-term memory and intensification phases since it is not desirable to have too much screening in both phases.

3. Computational studies

The performance of the various TS strategies described above are compared through extensive experimentation on a set of benchmark problems obtained from literature, to come up with a robust TS algorithm for SMTWT. The experimentation is performed using a new version of "WinMeta", an interactive software program developed previously in C++ to solve parallel machine scheduling problems with tardiness objectives via meta-heuristic approaches (Bilge et al., 2003). As the generic structure of WinMeta allows extensions, the single-machine TS strategies designed in this study are implemented and incorporated in the software. The particular solution strategy to be employed for each problem instance can be set by the user by selecting the proper combination from the menus and specifying the necessary parameters. The user friendly graphical interface of WinMeta allows ease of experimentation and flexibility in the strategies to employ as well as ease of analysis of results via detailed output report files. The experiments are conducted on a Pentium 4–1.6 GHz CPU, Host Bus 133 MHz with 512 MB RAM.

3.1. Test problems and performance measures

The problem set used for experimentation consists of single machine scheduling problems of 40, 50 and 100 jobs, developed and tested by Crauwels et al. (1998), who adopted the problem generation scheme proposed by Potts and Van Wassenhove (1985). There are 125 instances in each set and these instances are available in the OR library run by Beasley (2001), which is a collection of test data sets for a variety of Operations Research problems. As presented in the electronic OR library, the optimal values of solutions are available for 124 of the 40 job problem instances and for 115 of the 50 job problem instances, with the unsolved problem being instance number 19 in

the 40 job problem set and instances 11, 12, 14, 19, 36, 44, 66, 87, 88, and 111, in the 50 job problem set. As for the 100 job problem instances, some of the best known solution values reported by Crauwels et al. (1998) were improved by Congram et al. (2002) using the 'iterated dynasearch' algorithm, hence these modified values are used for benchmarking.

In each experiment, the performance evaluation is based on two criteria: the average relative percent deviation from the optimal (or best known) value over all the 125 instances in the set (ARPD), and the total number of instances out of 125 for which the optimal (or best known) solution could not be obtained, namely the number of non-optimal (NO) solutions.

3.2. Parameter and strategy selection

The TS algorithm presented in this study employs various strategies and parameters. The strategy selection and parameter calibration process is performed in a number of stages. The first step consists of exploratory testing to assess the best set of strategies and good ranges of parameters; and this step is conducted only on 50-job set of problems. Fixing the strategies and relatively insensitive parameters obtained as such, the next step involves systematic testing over the more sensitive parameters, namely the tenure parameters, to arrive at an empirical pattern based on problem size. At this step, the 40-job problem set is used together with the 50-job set. Finally, the empirical parameter pattern thus obtained, which is a function of problem size, is directly applied over the 100-job set. Thus the parameter calibration task is overcome without further tedious experimentation over the large problems. As a result, a robust TS algorithm complete with parameter setting strategies is obtained. The basic results of these algorithmic design and parameter calibration steps are summarized in this section.

3.2.1. Strategy selection

This first stage of experimentation is performed only on the 50-job set of problems and aims to select the best TS strategies for SMTWT. The stopping criterion for the tabu search method is set to be 5000 non-improving iterations for each of the instances.

As the first step, four initial solution generation heuristics (EDD, WSPT, R&M, R&M+) and ten candidate list strategies (two job type vs. five move type CLS) are tested by taking all combinations. In each of these experiments the tenure is set equal to the problem size ($n = 50$). After eliminating those strategies that are dominated, the most promising five candidate list strategy-initial solution heuristic combinations are retained for further analysis. These are presented in Table 1. It can be seen that, R&M and R&M+ perform best among the initial solution heuristics.

The next step is to determine the best fixed tabu tenure values for the specific strategies chosen above. The tenure search is performed over different ranges depending on the specific strategy employed, i.e. over $[40, 100]$ for no CLS and $[20, 50]$ for a CLS. A CLS that reduces the search neighborhood considerably necessitates the selection of a smaller tenure value. On the contrary, selecting a large search neighborhood requires the selection of larger tabu tenure. The outcome of this phase of experimentation is summarized in the last column of Table 1.

Next, experimentation for devising a systematic dynamic tenure strategy is performed. Several sets of small (S), medium (M) and large (L) tenure values are applied for a duration of [2 × medium tenure] iterations. The outcome of this step is depicted in Table 2. The value of M turns out to be close to the best performing tenures in the previous step, and cycle string is selected to be SMLM. As clearly seen in Table 2, applying a dynamic tenure structure rather than a single tenure value improves the quality of the solutions remarkably.

Table 1
Best performing strategies revealed by the preliminary experimentation over 50-job problem set

Initial solution heuristic	CLS		Best performing fixed tenures
	Job	Move	
R&M	NONE	NONE	50, 60
R&M+	TARDY	NONE	25, 30
R&M+	TARDY	A-OR	25, 30
R&M+	TARDY	A-DIV	25, 30
R&M+	TARDY	B-DIV	25, 30

Table 2
The effects of dynamic tenure and intensification strategies over the best CLS combinations (before fine-tuning tenure parameters)

Initial solution heuristic	Short term						Intensification			ARPD	% Impr. in ARPD[a]	NO	% Impr. in NO[a]
	CLS[b]		Tenure				CLS[b]		Tenure multiplier				
	Job	Move	S	M	L	Cycle string	Job	Move					
R&M	NONE	NONE	–	50	–	–	–	–	–	0.099	–	22	–
R&M	NONE	NONE	40	50	100	SMLM	–	–	–	0.074	25.25	15	31.82
R&M	NONE	NONE	40	50	100	SMLM	TARDY	NONE	0.5	0.001	98.65	3	80.00
R&M+	TARDY	NONE	–	25	–	–	–	–	–	0.035	–	28	–
R&M+	TARDY	NONE	20	30	40	SMLM	–	–	–	0.026	25.71	24	14.29
R&M+	TARDY	NONE	20	30	40	SMLM	NONE	NONE	1.7	0.004	84.61	7	70.83
R&M+	TARDY	A-OR	–	25	–	–	–	–	–	0.072	–	50	–
R&M+	TARDY	A-OR	20	30	40	SMLM	–	–	–	0.059	18.06	44	12.00
R&M+	TARDY	A-OR	20	30	40	SMLM	NONE	NONE	1.7	0.008	86.44	13	70.45
R&M+	TARDY	A-DIV	–	25	–	–	–	–	–	0.054	–	37	–
R&M+	TARDY	A-DIV	20	30	40	SMLM	–	–	–	0.033	38.88	30	18.92
R&M+	TARDY	A-DIV	20	30	40	SMLM	NONE	NONE	1.5	0.006	81.82	10	66.67
R&M+	TARDY	B-DIV	–	25	–	–	–	–	–	0.055	–	42	–
R&M+	TARDY	B-DIV	20	30	40	SMLM	–	–	–	0.041	25.45	36	14.29
R&M+	TARDY	B-DIV	20	30	40	SMLM	NONE	NONE	1.5	0.007	82.93	10	72.22

[a] With respect to the previous row.
[b] Strategies selected as best combinations as a result of experimentation.

Before attempting to further fine-tune the dynamic tenure structure, appropriateness of the intensification strategy is tested. The intensification phase is applied over short-term memory TS of 5000 non-improving iterations. The nature of the elite solutions is defined by "elite window", which is the number of iterations that a solution should survive as the best solution found so far in order to identify it as an elite solution. Elite windows of 200 or 300 iterations and recording two or three elites worked well in general. The duration of the intensification phase around each elite is set to be equal to 1500 non-improving iterations. Finally, in order to regulate the tenure in the intensification phase, a tenure factor is incorporated in the intensification strategy. This factor is a multiplier by which the tenures (S, M, L) for the short-term TS phase are multiplied before being used in the intensification phase. This tenure multiplier is regulated according to the nature of the CLSs employed both in the short-term TS phase and the intensification phase. The experiments revealed that the CLS during intensification must be selected in accord with the strategy used in the short-term memory TS phase. The best strategy combinations resulting from these experiments are also summarized in Table 2 along with the overall effect of applying intensification.

A direct observation from Table 2 is that intensification has a major effect on all of the strategies employed, if the appropriate complementary strategy is applied. The highest response is observed when "No CLS" in the short-term memory TS strategy is followed by an intensification phase employing the TARDY-NONE combination of candidate list strategies. The quality of the solutions obtained in this manner is also relatively higher than the other cases, both in terms of ARPD and the resulting number of non-optimal solutions. This is a reasonable match in that after the detailed search with no screening of candidates, the search procedure goes to the elites and performs a more directed and smarter search, which is faster than the "No CLS" case. Hence, the search re-originates at those local optima with a completely different attitude and goes in unvisited

yet good regions of the search space. On the other hand, when the short-term memory TS uses a candidate list strategy, it is reasonable to drop screening and explore all moves around the elites with "No CLS".

3.2.2. Parameter calibration

In the second stage of experimentation, the 40-job set is used as well, and the promising strategies are further fine-tuned for both sets with the aim of establishing some empirical rules to set the parameter values with respect to problem size. The resulting empirical dynamic tenure pattern is as follows: When "No CLS" is used in the short-term memory TS phase, the dynamic tenure attains a form of (15)–(n)–(2.25n) for small–medium–large tenures, where n is the number of jobs. If a CLS is used, then the dynamic tenure pattern becomes (15)–(0.5n)–(1.5n). The tenure values obtained in this manner are quite robust, i.e. varying the S or L tenure values by ±5 around these values does not cause a change in the solutions for 40-job problems, and for 50-job problems the number of non-optimal solutions may rise by up to three, the ARPD may increase by up to 0.003. The tenure values computed from these empirical patterns are shown in Table 3, and for the 100-job problem set they are directly used without further fine-tuning. The tenure multiplier of the intensification phase is another parameter to fine-tune, the ranges searched are $[0.3, 0.6]$ for no CLS and $[1.0, 1.7]$ for a CLS during intensification. As the problem

Table 3
Dynamic tenure pattern with respect to problem size

Problem size	CLS		Dynamic tenure			
	Job	Move	S	M	L	Cycle string
40-Job set	NONE	NONE	15	40	90	SMLM
	TARDY	*	15	20	60	SMLM
50-Job set	NONE	NONE	15	50	110	SMLM
	TARDY	*	15	25	75	SMLM
100-Job set	NONE	NONE	15	100	225	SMLM
	TARDY	*	15	50	150	SMLM

* NONE, A-DIV or B-DIV.

Table 4
Final results for 50-job problem set for selected TS strategies and fine-tuned parameters

50-Job problem set				
Method	TS1	TS2	TS3	TS4
Initial solution heuristic	R&M	R&M+	R&M+	R&M+
Candidate list strategy				
Job	NONE	TARDY	TARDY	TARDY
Move	NONE	NONE	A-DIV	B-DIV
Tabu classification	ARC+	ARC+	ARC+	ARC+
Dynamic tenure				
S	15	15	15	15
M	50	25	25	25
L	110	75	75	75
Cycle string	SMLM	SMLM	SMLM	SMLM
Intensification				
Maximum elite count	3	3	3	3
Elite window	300	300	300	300
Stopping condition	1500	1500	1500	1500
Tenure multiplier	0.5	1.4	1.4	1.4
Job	TARDY	NONE	NONE	NONE
Move	NONE	NONE	NONE	NONE
ARPD	0.001	0.002	0.004	0.003
NO	1	3	4	3
Average CPU (entire search)	170.712	42.712	44.576	41.536
Average CPU (time elapsed up to best)	16.91	4.61	8.42	8.68

size gets larger, selecting a slightly lower value of tenure multiplier seems more suitable to achieve the intensifying effect. A further decision made at this stage is to eliminate TARDY-A-OR candidate list strategy since it is outperformed by the others.

The results for the entire problem set that are presented in the next sub-section are obtained with these best performing strategy and parameter settings.

3.3. Results

The final results for the 50-job problem set are presented in Table 4. "No CLS" in the short-term memory TS phase and intensifying with the TARDY candidates list strategy provides the best results, optimally solving 124 of the 125 instances with the percent deviation of the single instance from optimal being only 0.08%. TARDY, TARDY-A-DIV and TARDY-B-DIV prove to be fast and efficient in that they succeed in solving all the problem instances with only three, four and three non-optimal solutions, respectively as seen in Table 4. When the search behaviors are observed via the graphical display of WinMeta, it is seen that the "DIVERSE" strategies follow different search paths resulting in a diverse set of elite solutions as intended.

The average CPU times required by each strategy are presented in two different formats. The first is the duration of the entire search. The second measure is the time elapsed from the beginning of the search until the best solution is found in that search, denoted as "Time elapsed up to best". All CPU measurements are in seconds. The results presented in Table 4 show that although "No CLS" yields higher quality solutions, the total computational time it requires is three times the total computational time required by the alternative methods. The TARDY-A-DIV and TARDY-B-DIV strategies have almost equal CPU, and their actual CPU requirements is half that required by "No CLS". The best CPU performance is given by the TARDY candidate list strategy, and it

Table 5
Final results for 40-job problem set for selected TS strategies and fine-tuned parameters

40-Job problem set				
Method	TS1	TS2	TS3	TS4
Initial solution heuristic	R&M	R&M+	R&M+	R&M+
Candidate list strategy				
Job	NONE	TARDY	TARDY	TARDY
Move	NONE	NONE	A-DIV	B-DIV
Tabu classification	ARC+	ARC+	ARC+	ARC+
Dynamic tenure				
S	15	15	15	15
M	40	20	20	20
L	90	60	60	60
Cycle string	SMLM	SMLM	SMLM	SMLM
Intensification				
Maximum elite count	3	3	3	3
Elite window	300	300	300	300
Stopping condition	1500	1500	1500	1500
Tenure multiplier	0.4	1.6	1.5	1.5
Job	TARDY	NONE	NONE	NONE
Move	NONE	NONE	NONE	NONE
ARPD	0	0	0	0.034
NO	0	0	0	1
Average CPU (entire search)	98.46	28.46	81.01	27.12
Average CPU (time elapsed up to best)	8.32	2.74	10.86	3.54

succeeds in optimally solving 122 out of 125 instances in 4.61 seconds per instance on the average. These strategies are named as TS1, TS2, TS3 and TS4, as seen in the first row of Table 4, for ease of reference.

It is seen in Table 5 that, all of the problem instances in the 40-job problem set can be solved to optimality by TS1, TS2 and TS3, while TS4 can solve 124 of the 125 instances. The best time performance is obtained with TS2, where the best solution is reached in an average time of 2.74 seconds.

For the 100-job set (Table 6), R&M+ becomes the preferred the initial solution heuristic also for TS1. This is expected, since the search space for the 100-job set is much larger compared to the 40-job set and it is better to initiate the search at the best possible starting point.

In the 100-job problem set, the best performance is achieved by TS4 which succeeds in obtaining 108 of the best-known solutions with an ARPD of only 0.007. This is achieved in an average computation time of 2 minutes. However, even the worst performing strategy, TS1 in this case, yields an ARPD of only 0.014 and this is accomplished at an average time requirement of 76 seconds. These robust results justify the empirical parameter selection rules based on problem size.

Crauwels et al. (1998) solved 123 of the 40-job and 118 of the 50-job problem instances optimally with their TS algorithm, as opposed to 125 and 124, respectively in our case. For the 100 job problem set they obtained 103 of the best solutions known then, but an improved set of results are used here (we obtained 108 of the best solutions in the updated benchmark). The ARPD for the 100-job instances is reported as 0.04 in the aforementioned study, and keeping in mind that this is calculated relative to an inferior set of best solutions, it is significantly worse than the 0.007 in our case. Furthermore, the multi-start TS algorithm they present is not deterministic and the statistics reported is based on the best solution obtained at the end of a single run of five random restarts.

Table 6
Final results for 100-job problem set for selected TS strategies and fine-tuned parameters

100-Job problem set				
Method	TS1	TS2	TS3	TS4
Initial solution heuristic	R&M+	R&M+	R&M+	R&M+
Candidate list strategy				
Job	NONE	TARDY	TARDY	TARDY
Move	NONE	NONE	A-DIV	B-DIV
Tabu classification	ARC+	ARC+	ARC+	ARC+
Dynamic tenure				
S	15	15	15	15
M	100	50	50	50
L	225	150	150	150
Cycle string	SMLM	SMLM	SMLM	SMLM
Intensification				
Maximum elite count	3	3	3	3
Elite window	300	300	300	300
Stopping condition	1500	1500	1500	1500
Tenure multiplier	0.5	1.2	1.3	1.4
Job	TARDY	NONE	NONE	NONE
Move	NONE	NONE	NONE	NONE
ARPD	0.014	0.010	0.007	0.007
NO	18	19	21	17
Average CPU (entire search)	341.26	271.39	277.48	283.46
Average CPU (time elapsed up to best)	76.18	82.88	125.47	127.59

Although it is not possible to make an exact comparison on the basis of time performance (they used an HP 9000-G50 computer), the TS algorithm presented here dominates this previous TS algorithm on the basis of solution quality.

The iterated dynasearch algorithm of Congram et al. (2002) succeeded to solve the 40-job and 50-job sets optimally and yielded an updated best solution set for the 100-job set. Both their time performance and ARPD are reported in a corrected form relative to the findings of Crauwels et al. (1998); for instance, whenever they found a solution which is better than the best solution known to Crauwels et al. its relative percent deviation is negative and actually reduces their ARPD. For this reason, it is difficult to make an exact comparison of the average performance of the iterated dynasearch algorithm with respect to our TS for the 100-job case. Even if we assume the extreme case of all best solutions being found in all ten replications they performed, the average relative percent deviation of our TS from this method would be only 0.007.

4. Conclusions

This study aims to develop a robust TS algorithm tailored for the single machine total weighted tardiness (SMTWT) problem. The TS method devised is completely deterministic and attains the best performance by employing a candidate list strategy based on problem context, a dynamic tenure structure and an intensification phase around elite solutions after the short-term memory TS is completed.

The results indicate that rather than employing a single tabu tenure value for the entire duration of TS, it is better to systematically vary the tenure to overcome some difficulties presented by the topology of the search space and to induce a balance between intensification and diversification. The initial solution is also a dominant effect in TS, and if the problem structure is not considered for appropriate selection of initial solution, TS deteriorates in its quality, as reflected in the preliminary experimentation phase. R&M+ heuristic performs well in this respect.

Candidate list strategies are critical in determining the efficiency of the TS strategy, and this study makes use of various candidate list strategies that incorporate the problem specific information in the screening mechanism. The tabu classification used is compatible with the candidate list strategies in that it functions in accord with what the CLS aims to accomplish in terms of broken arcs in a move. The neighborhood used contains all insert and swap moves and the candidate list strategies bring clever and efficient screening that enable a fast search over this large neighborhood. This however, like any screening mechanism, poses the risk of skipping some good neighbors. Therefore, intensification is performed around elite solutions that are identified during the short-term memory TS phase. This intensification phase employs a different neighborhood screening approach than the one in the initial phase so that different search paths are followed around the elite solutions when the search restarts. Hence, the candidate list strategy employed in the short-term memory TS phase is complemented with the candidate list strategy employed in the intensification phase. As such, the overall strategy becomes to employ one of two methods: The first method is to search intensely in the short-term memory TS phase with "No CLS" and then to intensify around the selected elites with TARDY Candidates List Strategy in a fast way. The second approach is to first perform a fast and efficient search in the short-term memory TS phase with TARDY, TARDY-A-DIV or TARDY-B-DIV candidate list strategy and then to restart the search around the elites with an intense search attitude, namely, with no candidate list strategy. The CPU time required for the first approach (TS1) is high as compared to the cases employing the second approach while quality of results is similar. Therefore, TS2, TS3 and TS4 which employ CLSs come forth as effective solution techniques for SMTWT.

As a result of these observations, the algorithm presented in Fig. 2 is suggested as a robust TS algorithm for the SMTWT Problem.

The proposed TS approach yields very high quality results for the set of benchmark problems obtained from the literature. The results show that

1. Generate an initial solution by means of a heuristic.
   Suggested strategy:
   - Use R&D+ heuristic.

2. Employ short-term TS search.
   Suggested strategy:
   - Apply an arc-based tabu classification (i.e. ARC+).
   - Apply TARDY candidate list strategy to screen the neighborhood of all insert and swap moves.
   - For larger size problems apply also a move type CLS with diversification effect (i.e. B-DIV) together with TARDY.
   - Use dynamic tenure with small (S), medium (M) and large (L) tenure values along a repeated pattern of SMLS, each for a duration of (2x M) iterations.
   - Select tenure values as S=15, M=0.5n, and L=1.5n (for a CLS strategy as described above) where n is the number of jobs.

3. Apply intensification.
   Suggested strategy:
   - Use three elite solutions identified during the short-term memory TS phase to restart the search.
   - Drop CLS at intensification phase (for a short-term CLS strategy as described above).
   - Adapt the short-term tenure values with a multiplier around [1.4,1.6] for a CLS strategy as described above.

Fig. 2. A robust TS algorithm for SMWTP.

the set of 40-job SMTWT problems is solved to optimality via the TS approach. As for the 50-job and 100-job problem sets, 99.2% and 86.4% of the best-known values reported in the literature are reached, respectively, with very insignificant deviations for the rest. These results are better than those for other state-of-the-art TS algorithms for SMTWT. The performance of proposed TS algorithm is also quite competitive with respect to the iterated dynasearch algorithm of Congram et al. (2002). The look-ahead capability of the latter approach resulting from allowing several moves in a single iteration draws the attention to incorporation of compound moves into TS neighborhood as well. Compound moves, an often neglected issue in TS literature (Glover and Laguna, 1997), may result into very large neighborhoods, but then again good candidate list strategies may help speeding up the search.

References

Abdul-Razaq, T.S., Potts, C.N., Van Wassenhove, L.N., 1990. A survey of algorithms for the single machine total weighted tardiness scheduling problem. Discrete Applied Mathematics 26, 235–253.

Beasley, J.E., 2001. OR-Library. Available from: <http://mscmga.ms.ic.ac.uk/jeb/orlib/wtinfo.html>.

Bilge, Ü., Kıraç, F., Kurtulan, M., Pekgün, P., 2003. A tabu search algorithm for parallel machine total tardiness problem. Computers and Operations Research 31, 397–414.

Carroll, D.C., 1965. Heuristic Sequencing of Single and Multiple Components, Ph.D. Dissertation, Massachusetts Institute of Technology, MA.

Chen, B., Potts, C.N., Woeginger, G., 1998. A review of machine scheduling: Complexity, algorithms and approximability. In: Handbook of Combinatorial Optimization. Kluwer Academic Publishers, Boston, pp. 21–169.

Cheng, T.C.E., Ng, C.T., Yuan, J.J., Liu, Z.H., 2005. Single machine scheduling to minimize total weighted tardiness. European Journal of Operational Research 165, 423–443.

Congram, R.K., Potts, C.N., Van de Velde, S.L., 2002. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. Informs Journal on Computing 14 (1), 52–67.

Crauwels, H.A.J., Potts, C.N., Van Wassenhove, L.N., 1998. Local search heuristics for single machine total weighted tardiness scheduling problem. Informs Journal on Computing 10, 341–350.

Du, J., Leung, J.Y.T., 1990. Minimizing total tardiness on one machine is NP-hard. Mathematics of Operations Research 15, 483–495.

Glover, F., Laguna, M., 1997. Tabu Search. Kluwer Academic Publishers, London.

Koulamas, C., 1994. The total tardiness problem: Review and extensions. Operations Research 42, 1025–1041.

Laguna, M., Barnes, J.W., Glover, F., 1991. Tabu search methods for a single machine scheduling problem. Journal of Intelligent Manufacturing 2, 63–74.

Lawler, E.L., 1977. A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness. Annals of Discrete Mathematics 1, 331–342.

Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P., 1977. Complexity of machine scheduling problems. Annals of Discrete Mathematics 1, 343–362.

Matsuo, H., Suh, C.J., Sullivan, R.S., 1989. A controlled search simulated annealing method for the single machine weighted tardiness problem. Annals of Operations Research 21, 85–108.

Montagne Jr., E.R., 1969. Sequencing with time delay costs. Industrial Engineering Research Bulletin 5.

Morton, T. E., Rachamadugu, R.V., Vepsalainen, A., 1984. Accurate Myopic Heuristics for Tardiness Scheduling,

Working Paper 36-83-84, GSIA, Carnegie Mellon University, Pittsburgh, PA.

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

Potts, C.N., Van Wassenhove, L.N., 1985. A branch and bound algorithm for the total weighted tardiness problem. Operations Research 33, 363–377.

Potts, C.N., Van Wassenhove, L.N., 1991. Single machine tardiness sequencing heuristics. IIE Transactions 23, 346–354.

Rachamadugu, R.V., Morton, T.E., 1982. Myopic Heuristics for the Single Machine Weighted Tardiness Problem, Working Paper 30-82-83, GSIA, Carnegie Mellon University, Pittsburgh, PA.

Rinnooy Kan, A.H.G., 1976. Machine Scheduling Problems Classification, Complexity and Computations. Martinus Nijhoff, The Hague.

Tian, Z.J., Ng, C.T., Cheng, T.C.E., 2005. On the single machine total tardiness problem. European Journal of Operational Research 165, 843–846.