Available online at www.sciencedirect.com
ScienceDirect
Computers & Industrial Engineering 52 (2007) 404–413
computers & industrial engineering
www.elsevier.com/locate/dsw

A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date ☆

Ching-Jong Liao *, Che-Ching Cheng

Department of Industrial Management, National Taiwan University of Science and Technology, Taipei, Taiwan

Received 23 April 2006; received in revised form 16 January 2007; accepted 22 January 2007
Available online 27 January 2007

Abstract

Although the concept of just-in-time (JIT) production systems has been proposed for over two decades, it is still important in real-world production systems. In this paper, we consider minimizing the total weighted earliness and tardiness with a restrictive common due date in a single machine environment, which has been proved as an NP-hard problem. Due to the complexity of the problem, metaheuristics, including simulated annealing, genetic algorithm, tabu search, among others, have been proposed for searching good solutions in reasonable computation times. In this paper, we propose a hybrid metaheuristic that uses tabu search within variable neighborhood search (VNS/TS). There are several distinctive features in the VNS/TS algorithm, including different ratio of the two neighborhoods, generating five points simultaneously in a neighborhood, implementation of the B/F local search, and combination of TS with VNS. By examining the 280 benchmark problem instances, the algorithm shows an excellent performance in not only the solution quality but also the computation time. The results obtained are better than those reported previously in the literature.

Keywords: Scheduling; Variable neighborhood search; Tabu search; Common due date; Weighted earliness/tardiness

1. Introduction

Just-in-time (JIT) production emphasizes that earliness, as well as tardiness, should be discouraged. This is due to the fact that an early job may result in inventory carrying cost, such as opportunity cost of the money invested in inventory, storage and insurance costs and deterioration. Contrarily, a tardy job may result in customer dissatisfaction, contract penalties, loss of sale and loss of reputation. Therefore, the criterion involving both earliness and tardiness costs has received significant attention recently.

In this paper, we consider the earliness and tardiness problem with a common due date. The field of common due date scheduling can be distinguished into two cases, unrestrictive and restrictive ones. In the unrestrictive case, which was first introduced by Kanet (1981), the common due date has to be determined, or its value is greater than or equal to the sum of processing times of all jobs, i.e., it has no influence on the optimal

☆ This manuscript was processed by Area Editor John W. Fowler.

* Corresponding author. Tel.: +886 2 2737 6337; fax: +886 2 2737 6344.
E-mail address: cjl@im.ntust.edu.tw (C.-J. Liao).

0360-8352/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2007.01.004

schedule. On the other hand, it is called restrictive if the common due date is known and may affect the optimal schedule. A comprehensive survey on the common due date scheduling can be found in Gordon, Proth, and Chu (2002).

For the restrictive common due date problem on single machine, Biskup and Feldmann (2001) generated a total of 280 benchmark problem instances with 10, 20, 50, 100, 200, 500 and 1000 jobs and restricted the common due date with 0.2, 0.4, 0.6, and 0.8 of the sum of all processing times. This problem is well known in the scheduling literature as NP-hard (Hall, Kubiak, & Sethi, 1991) such that exact solution methods are not suitable for solving large-size instances of the problem. Thus, they developed two heuristics for finding good solutions in reasonable computation times. Later, Feldmann and Biskup (2003) applied different metaheuristics, namely evolutionary search (ES), evolutionary search with a destabilization phase (ESD), simulated annealing (SA), threshold accepting (TA) and threshold accepting with a back step (TAR), to solve the 0.2 and 0.4 problems. They compared their solution quality and found that the proposed TAR is superior to the tabu search techniques of James (1997), which dealt with the same problem. Recently, Hino, Ronconi, and Mendes (2005) applied genetic algorithm (GA) and tabu search (TS) and demonstrated that their algorithms performs better than those of Feldmann and Biskup (2003) for the 0.2 and 0.4 problems. A hybrid algorithm (HGT) applying GA and TS in sequential form presents similar results with a shorter computation time.

This paper considers the same problem as studied by Feldmann and Biskup (2003) and Hino et al. (2005). We propose a hybrid approach to further improve the solution quality of the problem. Our approach applies variable neighborhood search (VNS) in combination with TS. VNS, proposed by Mladenović and Hansen (1997), is a recent metaheuristic based on the principle of systematic change of neighborhood during the search. It has been proved to be a simple and effective method for solving optimization problems, including traveling salesman problem (Mladenović & Hansen, 1997), p-median problem (Hansen & Mladenović, 1997), graph coloring problem (Avanthay, Hertz, & Zufferey, 2003), and minimum spanning tree problem (Ribeiro & Souza, 2002). Recently, several authors combine it with other metaheuristics, such as greedy randomized adaptive search procedure (GRASP-VNS; Drummond, Vianna, Silva, & Ochi, 2002) and tabu search (TS-VNS; Gagné, Gravel, & Price, 2005). On the other hand, tabu search is an improvement on the decent algorithm that avoids the trap of local optimum by allowing a non-improving move (Glover, 1986). TS has been successfully applied to a large number of combinatorial optimization problems, including network design (Crainic, Gendreau, & Farvolden, 2000), vehicle routing (Gendreau, Guertin, Potvin, & Taillard, 1999), quadratic assignment (Skorin-Kapov, 1990) and production scheduling (Al-Turki, Fedjki, & Andijani, 2001; Liaw, 2003), which have shown to be competitive with other metaheuristics.

The remainder of this paper is organized as follows. In Section 2, we give a formal definition and some useful properties of the problem. Section 3 provides a detailed description of the VNS/TS algorithm. Results of computational experiments to evaluate the performance of the proposed algorithm are reported in Section 4. Finally, Section 5 describes the main results and discusses some directions for future research.

2. Problem formulation

In this section we formally define the considered problem and give some useful properties. The following notation is used throughout the paper:

n	number of jobs
d	common due date
p_j	processing time of job j
α_j	weight of the earliness for job j
β_j	weight of the tardiness for job j
S_E	set of jobs completed at or before the due date, $S_E = \{j \mid C_j \leqslant d\}$
S_T	set of jobs started at or after the due date, $S_T = \{j \mid C_j - p_j \geqslant d\}$
I	idle time between time zero and the first job in S_E
S	a feasible schedule of jobs, $S = (I, S_E, S_T)$
C_j	completion time of job j

E_j	earliness of job j, $E_j = \max(0, d - C_j)$
T_j	tardiness of job j, $T_j = \max(0, C_j - d)$
h	due-date factor, $d = [h \times \sum_{j=1}^{n} p_j]$
η	indicator of the neighborhoods
TL_i	size of the tabu list, TL_1 for insertion move and TL_2 for swap move

The problem considered in this paper can be stated as follows. There is a common due date d for a set of n jobs which is to be processed on a continuously available single machine. The machine can process only one job at a time. Each of the n jobs is available for processing at time zero and has a known processing time p_j, earliness weight α_j, and tardiness weight β_j. The objective is to determine a feasible schedule S with minimum total weighted earliness and tardiness, i.e.,

$$f(S) = \sum_{j=1}^{n} (\alpha_j E_j + \beta_j T_j)$$

The earliness and the tardiness of job j are calculated as $E_j = \max(0, d - C_j)$ and $T_j = \max(0, C_j - d)$. Using the standard scheduling notation (Pinedo, 2002), the problem can be written as $1/d_j = d/\sum \alpha_j E_j + \sum \beta_j T_j$. In this problem, an optimal schedule has a number of useful properties.

Property 1. *In an optimal schedule there are no idle times between the processing of any two consecutive jobs.*

Proof. See Cheng and Kahlbacher (1991). □

The second property is the so-called V-shaped property. Under any given schedule, the jobs can be partitioned into two disjoint sets. One set, denoted by S_E, contains the jobs that are completed at or before the due date, i.e., $S_E = \{j | C_j \leqslant d\}$. The other set, denoted by S_T, contains the jobs that are started at or after the due date, i.e., $S_T = \{j | C_j - p_j \geqslant d\}$.

Property 2. *In an optimal schedule, the jobs $j \in S_E$ are scheduled by their non-increasing ratios p_j/α_j and the jobs $j \in S_T$ by non-decreasing ratios p_j/β_j.*

Proof. See Baker and Scudder (1990). □

Note that even with Property 2 an optimal schedule may contain a so-called straddling job not belonging to the sets S_E and S_T, which is started before and completed after the due date and is placed between S_E and S_T. If such a straddling job exists in an optimal schedule, its ratios p_j/α_j and p_j/β_j can be greater than that of the remaining jobs (Hoogeveen & van de Velde, 1991).

Property 3. *There exists an optimal schedule in which either the first job is started at time zero or one job is completed exactly at the due date.*

Proof. See Hoogeveen and van de Velde (1991). □

3. Proposed VNS/TS algorithm

Due to the complexity of the problem, metaheuristics are recommended for searching good solutions in reasonable computation times. As stated in the Introduction section, the metaheuristics that have been used for this problem include SA, TAR, GA, TS, GA + TS, among others. In this section, we propose a hybrid algorithm that uses tabu search within variable neighborhood search (VNS/TS). The components of the proposed VNS/TS algorithm are described below.

3.1. Variable neighborhood search

VNS is different from most local search heuristics in that it uses two or more neighborhoods, instead of one, in its structure. In particular, it is based on the principle of systematic change of neighborhood during the search. In addition, to avoid costing too much computational time, the best number of neighborhoods is often

two (Glover & Kochenberger, 2003; p. 147), which is followed by our algorithm. The two neighborhoods employed in our algorithm are defined below:

1. Insertion moves ($\eta = 1$): Randomly identifies two particular jobs, one for each set S_E and S_T, and places one job in the position that directly precedes the other job.
2. Swap moves ($\eta = 2$): Randomly identifies two particular jobs, one for each set S_E and S_T, and places each job in the position previously occupied by the other.

The proposed VNS has a slightly different procedure in dealing with problems with "tight" or "loose" due date. The benchmark problem instances of Biskup and Feldmann (2001) used four restrictive due-date factors ($h = 0.2, 0.4, 0.6, 0.8$), which were used in determining the due date as $d = [h \times \sum_{j=1}^{n} p_j]$. A due date is called loose if $h = 0.2$ or 0.4; it is called tight if $h = 0.6$ or 0.8. The steps of our VNS structure are described below:

Step 1. Find an initial solution x. Set $\eta = 1$ and $i = 0$.
Step 2. Repeat the following steps until N accumulative no-improving iterations:
(i) Generate five schedules x' at random from the ηth neighborhood of x.
(ii) If the due date is loose, apply a local search method to each schedule.
(iii) Choose the best schedule x''. If x'' is better than the incumbent, update $x = x''$ and return to (i).
(iv) If $\eta = 1$, set $\eta = 2$ and return to (i); otherwise $i = i + 1$.
(v) If $i = 10$, set $\eta = 1$ and $i = 0$. Return to (i).

We now elaborate on the steps in more detail. In Step 1, we use the best initial solution method for the respective case. That is, we employ the initial solution method of Hino et al. (2005) for the tight due date case (i.e., $h = 0.2$ and 0.4) and the method of Feldmann and Biskup (2003) for the loose due date case (i.e., $h = 0.6$ and 0.8).

In Step 2(i) we generate five, instead of one, schedules x' to extend the probability of searching for a better solution. We will demonstrate by experiments that this approach can improve the performance of our algorithm. For the loose due date problem (i.e., $h = 0.6$ and 0.8), a local search method, to be discussed later in this section, is further applied in Step 2(ii) to each generated schedule. Step 2(iii) and (iv) specify that the insertion move is executed once when the swap move results in no improvement for 10 accumulative moves. This is because the swap move is more effective than the insertion move (Feldmann & Biskup, 2003) for the problem, and hence we give a larger weight to the effective one. The computational experiments conducted in the next section will verify our reasoning.

It should be mentioned that whenever a new sequence is generated (in Step 2(i) and (ii)), all the three essential properties stated in Section 2 have to be satisfied in constructing the new schedule. That is, the new schedule will be arranged such that once started there are no inserted idle times, jobs are put in V-shaped order, and the schedule either starts its processing at time zero or completes a job exactly at the due date.

3.2. B/F local search

As stated earlier, a local search is performed in Step 2(ii) for the loose common due date case (i.e., $h = 0.6$ and 0.8). The local search implements either a backward move or a forward move, and is hereinafter called B/F local search. In the following steps, the jobs $j \in S_E$ are scheduled by their non-increasing ratios p_j/α_j and the jobs $j \in S_T$ by non-decreasing ratios p_j/β_j.

Step 0. Set $i = 1$. Compute the objective function value $f(S)$ for schedule $S = (I, S_E, S_T)$.
Step 1. Consider the first job in S_T, say job k. If $p_k > I$, go to Step 3; otherwise, move job k to S_E and update $I = I - p_k$, $S' = (I, S_E + \{k\}, S_T - \{k\})$.
Step 2. Rearrange the jobs in S_E in V shape and compute $f(S')$. If $f(S') < f(S)$, set $S = S'$ and $i = i + 1$; return to Step 1. Otherwise, go to Step 3 if $i = 1$, or stop the algorithm if $i \neq 1$ (S is the resulting schedule).
Step 3. Move the last job in S_E, say job k, to S_T, and update $I = I + p_k$, $S' = (I, S_E - \{k\}, S_T + \{k\})$.
Step 4. Rearrange the jobs in S_T in V shape and compute $f(S')$. If $f(S') < f(S)$, set $S = S'$ and return to Step 3; otherwise, stop the algorithm (S is the resulting schedule).

We now briefly explain the above steps. In Steps 1 and 2, we try to move jobs backward from S_T to S_E. If no backward step succeeds, then we try to move jobs forward from S_E to S_T in Steps 3 and 4. We illustrate the B/F local search by means of the following example where only the backward moves are performed:

Example 1. Consider a seven-job problem with $d = 64$, $S = (I = 40, S_E = (1,2,3), S_T = (4,5,6,7))$, p_j, α_j, β_j and $f(S)$ as follows:

j	1	2	3	4	5	6	7
p_j	9	10	5	3	7	16	20
α_j	1	2	5	2	1	10	7
β_j	6	9	4	7	3	4	4

$f(S) = 364$.

Initially, we set $i = 1$ and start to try job 4 (the first job in S_T). Since $p_4 = 3 \leqslant 40 = I$, we move job 4 from S_T to S_E and update $I = I - p_4 = 37$. To obey the V-Shaped property, we rearrange the jobs in S_E and obtain $S' = (37,(1,2,4,3),(5,6,7))$ with $f(S') = 329$. Because $f(S') = 329 < 364 = f(S)$, we set $S = S'$, $i = i + 1 = 2$ and try the next job. Since $p_5 = 7 \leqslant 37 = I$, we move job 5 from S_T to S_E and update $I = I - p_5 = 30$. We rearrange the jobs in S_E in V shape and obtain $S' = (30,(1,5,2,4,3),(6,7))$ with $f(S') = 277$. Because $f(S') = 277 < 329 = f(S)$, we set $S = S'$, $i = i + 1 = 3$ and try the next job. Since $p_6 = 16 \leqslant 30 = I$, we set $I = I - p_6 = 14$ and move job 6 from S_T to S_E. We rearrange the jobs in S_E in V shape and obtain $S' = (14,(1,5,6,2,4,3),(7))$ with $f(S') = 293$. Now $f(S') = 293 > 277 = f(S)$, we stop the algorithm and $S = (30,(1,5,2,4,3),(6,7))$ is the resulting schedule.

3.3. Tabu search

Tabu search (TS) is an improvement on the decent algorithm that avoids the trap of local optimum by allowing a non-improving move (Glover, 1986). To prevent cycling back to previously visited solutions, a tabu list is used to record the recent moves. Because there are two neighborhoods in our VNS structure, we use two tabu lists. The first one is dedicated to the insertion move and its size (TL_1) is fixed at 7. The second tabu list is dedicated to the swap move and three different sizes (TL_2) 7, 14 and 21 are used depending on the number of jobs. In order to allow one to revoke a tabu list when there is an attractive tabu move, aspiration criteria are introduced in our algorithm (Glover, 1993). When there is no improvement for 50 accumulative iterations, we allow overriding the tabu state of a solution to include it in the allowed set if the solution is better than the current best-known solution.

4. Computational results

To verify the performance of the VNS/TS algorithm, we tested the same set of problems as Biskup and Feldmann (2001), Feldmann and Biskup (2003) and Hino et al. (2005). The algorithm was code in C++ and implemented on a Pentium 4 3.2 GHz with 512MB memory. Each instance was run 10 times and the best solution was selected.

The benchmark problem instances were provided by Biskup and Feldmann (2001), which can be obtained at http://people.brunel.ac.uk/~mastjjb/jeb/orlib/schinfo.html (Beasley, 2005). There are seven different numbers of jobs ($n = 10, 20, 50, 100, 200, 500, 1000$) with four restrictive due-date factors ($h = 0.2, 0.4, 0.6, 0.8$), where the common due date was determined by the expression $d = [h \times \sum_{j=1}^{n} p_j]$. For each combination, 10 different problem instances were generated, resulting in a total of 280 problem instances.

Before conducting a formal experiment for comparing with existing algorithms, the following four preliminary experiments will be carried out:

1. Determine if the insertion move can improve the algorithm. If yes, determine the best ratio of the two neighborhoods.

Table 1
Comparative evaluation of algorithms with different ratios of two neighborhoods

n	Insertion: Swap	h				
		0.2	0.4	0.6	0.8	Mean
200	0:1	−5.72	−3.71	−0.13	−0.13	−2.42
	1:1	−5.73	−3.72	−0.14	−0.14	−2.43
	1:5	−5.77	−3.74	−0.15	−0.15	−2.45
	1:10	−5.78	−3.75	−0.15	−0.15	−2.46
	1:20	−5.76	−3.74	−0.14	−0.14	−2.45
500	0:1	−6.33	−3.51	−0.08	−0.08	−2.50
	1:1	−6.35	−3.53	−0.08	−0.08	−2.51
	1:5	−6.41	−3.54	−0.10	−0.10	−2.54
	1:10	−6.42	−3.56	−0.11	−0.11	−2.55
	1:20	−6.39	−3.53	−0.09	−0.09	−2.53

2. Determine the best number of generated schedules in a neighborhood.
3. Evaluate the algorithm with and without the use of the B/F local search.
4. Evaluate the algorithm with and without the combination of TS.

All of the four experiments were conducted for two job sizes $n = 200$ and 500. To have a fair comparison, 1 second of computation time is set as the stopping criterion for all the instances with $n = 200$, and 7 s for all the instances with $n = 500$. To measure the effectiveness of algorithms, we compute the percentage improvement (PI) of the solution value obtained (F_a) with respect to the benchmark value provided by Biskup and Feldmann (2001; F_{BF}) as follows:

$$\mathrm{PI} = 100 \times \frac{F_a - F_{BF}}{F_{BF}}$$

The benchmark values from Biskup and Feldmann (2001) are upper bounds on the optimal objective function values, except for the instances with 10 jobs which were solved optimally with LINDO software.

Table 2
Comparative evaluation of algorithms with different numbers of generated schedules in a neighborhood

n	Number of schedules	h				
		0.2	0.4	0.6	0.8	Mean
200	1	−5.77	−3.73	−0.15	−0.15	−2.45
	5	−5.78	−3.75	−0.15	−0.15	−2.46
	10	−5.78	−3.75	−0.15	−0.15	−2.46
500	1	−6.41	−3.53	−0.10	−0.09	−2.53
	5	−6.42	−3.56	−0.11	−0.11	−2.55
	10	−6.41	−3.54	−0.11	−0.10	−2.54

Table 3
Comparative evaluation of algorithms with and without the B/F local search

n	B/F	h		
		0.6	0.8	Mean
200	With	−0.15	−0.15	−0.15
	Without	−0.13	−0.13	−0.13
500	With	−0.11	−0.11	−0.11
	Without	−0.06	−0.05	−0.06

Table 4
Comparative evaluation between VNS and VNS/TS

n	Algorithm	h				
		0.2	0.4	0.6	0.8	Mean
200	VNS	−5.78	−3.74	−0.15	−0.15	−2.46
	VNS/TS	−5.78	−3.75	−0.15	−0.15	−2.46
500	VNS	−6.41	−3.53	−0.10	−0.10	−2.54
	VNS/TS	−6.42	−3.56	−0.11	−0.11	−2.55

The result of the first experiment is given in Table 1, where the second column indicates the ratio of insertion move to swap move. The ratio 0:1 means no insertion moves, and ratio 1:10 denotes that an insertion move is performed for every 10 swap moves. It is observed from Table 1 that the insertion move can result in an improvement. Because the proposed algorithm is quite effective, the solutions with different ratios are all very close to the optimal solution, resulting in similar PI values. However, for all the eight combinations

Table 5
Comparison with HGT and GA of Hino et al. (2005)

n	h	HGT	GA	VNS/TS	
10	0.2	0.12	0.12	0.00	+
	0.4	0.19	0.19	0.00	+
	0.6	0.01	0.03	0.00	+
	0.8	0.00	0.00	0.00	
20	0.2	−3.84	−3.84	−3.84	
	0.4	−1.62	−1.62	−1.63	+
	0.6	−0.71	−0.68	−0.72	+
	0.8	−0.41	−0.28	−0.41	
50	0.2	−5.70	−5.68	−5.70	
	0.4	−4.66	−4.60	−4.66	
	0.6	−0.31	−0.31	−0.34	+
	0.8	−0.23	−0.19	−0.24	+
100	0.2	−6.19	−6.17	−6.19	
	0.4	−4.93	−4.91	−4.94	+
	0.6	0.04	−0.12	−0.15	+
	0.8	−0.11	−0.12	−0.18	+
200	0.2	−5.76	−5.74	−5.78	+
	0.4	−3.75	−3.75	−3.75	
	0.6	0.07	−0.13	−0.15	+
	0.8	0.07	−0.14	−0.15	+
500	0.2	−6.41	−6.41	−6.42	+
	0.4	−3.58	−3.58	−3.56	−
	0.6	0.15	−0.11	−0.11	
	0.8	0.13	−0.11	−0.11	
1000	0.2	−6.74	−6.75	−6.75	
	0.4	−4.39	−4.40	−4.37	−
	0.6	0.42	−0.05	−0.05	
	0.8	0.40	−0.05	−0.05	
Mean		−2.06	−2.12	−2.15	

+ VNS/TS is the best one. − Either HGT or GA is the best one.

of n and h, ratio 1:10 always gives the best solution, and hence it is chosen as the insertion/swap ratio in our algorithm.

The result of the second experiment is given in Table 2, which compares the algorithms with different numbers of generated schedules in a neighborhood. It is observed that generating 5 points in a neighborhood is superior to generating 1 point or 10 points.

The third experiment evaluates the algorithm with and without the use of the B/F local search, which is implemented only for the loose due date (i.e., $h = 0.6, 0.8$). It can be observed from Table 3 that the B/F local search always results in an obvious improvement.

The fourth experiment evaluates the algorithm with and without the combination of TS, i.e., to compare the hybrid VNS/TS with the pure VNS. It is observed from Table 4 that the hybrid VNS/TS indeed has a better performance.

Table 6
The solution values of VNS/TS for the 280 instances

n	k	h				n	k	h			
		0.2	0.4	0.6	0.8			0.2	0.4	0.6	0.8
10	1	1936	1025	841	818	20	1	4394	3066	2986	2986
	2	1042	615	615	615		2	8430	4847	3206	2980
	3	1586	917	793	793		3	6210	3838	3583	3583
	4	2139	1230	815	803		4	9188	5118	3317	3040
	5	1187	630	521	521		5	4215	2495	2173	2173
	6	1521	908	755	755		6	6527	3582	3010	3010
	7	2170	1374	1101	1083		7	10,455	6238	4126	3878
	8	1720	1020	610	540		8	3920	2145	1638	1638
	9	1574	876	582	554		9	3465	2096	1965	1965
	10	1869	1136	710	671		10	4979	2925	2110	1995
50	1	40,697	23,792	17,969	17,934	100	1	145,516	85,884	72,017	72,017
	2	30,613	17,907	14,050	14,040		2	124,916	72,981	59,230	59,230
	3	34,425	20,500	16,497	16,497		3	129,800	79,598	68,537	68,537
	4	27,755	16,657	14,080	14,080		4	129,584	79,405	68,759	68,759
	5	32,307	18,007	14,605	14,605		5	124,351	71,275	55,286	55,103
	6	34,969	20,385	14,251	14,066		6	139,188	77,778	62,398	62,398
	7	43,134	23,038	17,616	17,616		7	135,026	78,244	62,197	62,197
	8	43,839	24,888	21,329	21,329		8	160,147	94,365	80,708	80,708
	9	34,228	19,984	14,202	13,942		9	116,522	69,457	58,727	58,727
	10	32,958	19,167	14,366	14,363		10	118,911	71,850	61,361	61,361
200	1	498,654	295,697	254,259	254,259	500	1	2,955,335	1,787,979	1,579,140	1,579,109
	2	541,181	319,205	266,002	266,002		2	3,365,952	1,995,161	1,712,429	1,712,466
	3	488,665	293,888	254,488	254,476		3	3,102,930	1,864,852	1,641,706	1,641,718
	4	586,257	353,034	297,109	297,109		4	3,221,423	1,887,902	1,640,785	1,640,784
	5	513,240	304,678	260,278	260,278		5	3,114,982	1,807,378	1,468,256	1,468,263
	6	478,022	279,940	235,702	235,702		6	2,792,362	1,610,445	1,411,867	1,411,841
	7	454,757	275,030	246,330	246,313		7	3,173,069	1,902,767	1,634,330	1,634,330
	8	494,286	279,177	225,215	225,215		8	3,122,442	1,819,891	1,540,458	1,540,470
	9	529,292	310,418	254,659	254,637		9	3,364,709	1,973,920	1,680,486	1,680,647
	10	538,354	323,109	268,353	268,354		10	3,120,925	1,837,705	1,519,215	1,519,205
1000	1	14,057,673	8,112,258	6,411,260	6,411,352						
	2	12,299,505	7,273,716	6,110,369	6,110,400						
	3	11,971,501	6,989,567	5,983,589	5,983,430						
	4	11,799,103	7,025,955	6,088,472	6,089,268						
	5	12,452,386	7,366,950	6,342,433	6,342,525						
	6	11,646,125	6,929,479	6,079,207	6,079,243						
	7	13,279,165	7,863,861	6,574,569	6,574,465						
	8	12,278,210	7,224,251	6,067,688	6,067,727						
	9	11,760,010	7,060,904	6,185,834	6,185,813						
	10	12,430,428	7,278,460	6,146,054	6,145,999						

In the formal experiment, we compare our VNS/TS algorithm with the best algorithms for the problem in the literature. The following parameters, determined by a series of pilot experiments, are used in our algorithm: $TL_1 = 7$, $TL_2 = 7$ (for $n = 10, 20, 50$), 14 (for $n = 100, 200$), 21 (for $n = 500, 1000$); $MaxIter = 600$ for $h = 0.2, 0.4$ and $MaxIter = 150$ for $h = 0.6, 0.8$, where $MaxIter$ is the maximum number of iterations.

We compare our results with current best methods presented by Hino et al. (2005). They developed two metaheuristics and two hybrid algorithms: GA, TS, HTG (TS + GA), and HGT (GA + TS). The hybrid algorithms simply apply the metaheuristics in a sequential form. Among the four algorithms, GA and HGT are the two best ones, where GA obtained an average percentage improvement (API) of −2.12% against those benchmark values with a mean processing time of 21.5 s; HGT obtained a −2.06% API with only 7.8 s.

The comparative results of our VNS/TS with the better one of GA and HGT is given in Table 5. It can be observed that VNS/TS achieves better results in 14 and worst in 2 of 28 suites of problem instances and gains average 0.09% and 0.03% against HGT and GA, respectively. Note that the results of VNS/TS for all 10-job problems are equal to the benchmark values which had been proved by Biskup and Feldmann (2001) to be optimal. As for the computation time, our VNS/TS requires only 4.7 s for each run (on a Pentium 4 3.2 GHz with 512MB memory), HGT takes 7.8 s and GA takes 21.5 s (both on a Pentium 4 1.7 GHz with 512MB memory). By considering the different computer environments, VNS/TS and HGT have relatively the same efficiency, but VNS/TS is much more efficient than GA.

The solution values for all the 280 instances yielded by our VNS/TS are provided in Table 6. Because we cannot obtain the results from Hino et al. (2005), the current best known solutions for the benchmark problems are unknown except for the instances with 10 jobs. Nevertheless, based on the above comparative results, we believe that most of the solutions in Table 6 constitute the best solutions known for the benchmark problems as to date.

5. Conclusions

Although the concept of JIT production systems has been proposed for over two decades, it is still important in many real-world production systems. This paper presents a metaheuristic algorithm for the special case, the total weighted earliness and tardiness with the common due date on a single machine, of JIT problems. There are several distinctive features in the developed VNS/TS algorithm, including different ratio of the two neighborhoods, generating five points simultaneously in a neighborhood, implementation of the B/F local search, and combination of TS with VNS. These features make the algorithm very efficient and effective. By examining the 280 benchmark problem instances, the algorithm shows an excellent performance in not only the solution quality but also the computation time. The results obtained are better than those reported previously in the literature and constitute the best solutions known for the benchmark problems as to date.

Extension of the VNS/TS algorithm with its simple, efficient and effective characteristics to other machine environment problems involving earliness and tardiness costs is a possible direction of more research. Consideration of other classes of objective functions is also interesting. Finally, development of an exact solution method (e.g., branch and bound method) for solving relatively larger sized problem instances, which could be used to better evaluate the effectiveness of metaheuristic, is a challenging area for further research.

References

Al-Turki, U., Fedjki, C., & Andijani, A. (2001). Tabu search for a class of single-machine scheduling problems. *Computers and Operations Research, 28*, 1223–1230.

Avanthay, C., Hertz, A., & Zufferey, N. (2003). A variable neighborhood search for graph coloring. *European Journal of Operational Research, 151*, 379–388.

Baker, K. R., & Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: A review. *Operations Research, 38*, 22–36.

Beasley, J. E. (2005). Available from OR-Library. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/schinfo.html.

Biskup, D., & Feldmann, M. (2001). Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. *Computers and Operations Research, 28*, 787–801.

Cheng, T. C. E., & Kahlbacher, H. G. (1991). A proof for the longest-job-first policy in one-machine scheduling. *Naval Research Logistics, 38*, 715–720.

Crainic, T. G., Gendreau, M., & Farvolden, J. M. (2000). Simplex-based tabu search for the multicommodity capacitated fixed charge network design problem. *INFORMS Journal on Computing, 12*, 223–236.

Drummond, L. M. A., Vianna, L. S., Silva, M. B., & Ochi, L. S. (2002). Distribution parallel metaheuristics based on GRASP and VNS for solving the traveling purchaser problem. In *Proceedings of the 9th International Conference on Parallel and Distributed System*, pp. 257–263.

Feldmann, M., & Biskup, D. (2003). Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. *Computers and Industrial Engineering, 44*, 307–323.

Gagné, C., Gravel, M., & Price, W. L. (2005). Using metaheuristic compromise programming for the solution of multiple-objective scheduling problems. *Journal of the Operational Research Society, 56*, 687–698.

Gendreau, M., Guertin, F., Potvin, J.-Y., & Taillard, E. D. (1999). Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science, 33*, 381–390.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research, 13*, 533–549.

Glover, F. (1993). A user's guide tabu search. *Annals of Operations Research, 41*, 3–28.

Glover, F., & Kochenberger, G. A. (2003). *Handbook of metaheuristics*. Boston, MA: Kluwer Academic Publisher.

Gordon, V., Proth, J.-M., & Chu, C. (2002). A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research, 139*, 1–25.

Hall, N. G., Kubiak, W., & Sethi, S. P. (1991). Earliness-tardiness scheduling problems, II: Deviation of completion times about a restrictive common due date. *Operations Research, 39*, 847–856.

Hansen, P., & Mladenović, N. (1997). Variable neighborhood search for the p-Median. *Location Science, 5*, 207–226.

Hino, C. M., Ronconi, D. P., & Mendes, A. B. (2005). Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. *European Journal of Operational Research, 160*, 190–201.

Hoogeveen, J. A., & van de Velde, S. L. (1991). Scheduling around a small common due date. *European Journal of Operational Research, 55*, 237–242.

James, R. J. W. (1997). Using tabu search to solve the common due date early/tardy machine scheduling problem. *Computers and Operations Research, 24*, 199–208.

Kanet, J. J. (1981). Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly, 28*, 643–651.

Liaw, C. F. (2003). An efficient tabu search approach for the two-machine preemptive open shop scheduling problem. *Computers and Operations Research, 30*, 2081–2095.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research, 24*, 1097–1100.

Pinedo, M. (2002). *Scheduling: Theory, algorithms, and systems*. Englewood Cliffs, NJ: Prentice-Hall.

Ribeiro, C. C., & Souza, M. C. (2002). Variable neighborhood search for the degree-constrained minimum spanning tree problem. *Discrete Applied Mathematics, 118*, 43–54.

Skorin-Kapov, J. (1990). Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing, 2*, 33–45.