International Journal of Computational Intelligence Systems, Vol. 4, No. 4 (June, 2011), 456-466

A Novel Ant Colony Algorithm for the Single-Machine Total Weighted Tardiness Problem with Sequence Dependent Setup Times

Fardin Ahmadizar*
Department of Industrial Engineering, University of Kurdistan, Pasdaran Boulvard, Sanandaj, Iran

Leila Hosseini
Department of Industrial Engineering, University of Kurdistan, Pasdaran Boulvard, Sanandaj, Iran

E-mail: leila.hosseini@ uok.ac.ir

Received 31 October 2010
Accepted 2 May 2011

Abstract

This paper deals with the NP-hard single-machine total weighted tardiness problem with sequence dependent setup times. Incorporating fuzzy sets and genetic operators, a novel ant colony optimization algorithm is developed for the problem. In the proposed algorithm, artificial ants construct solutions as orders of jobs based on the heuristic information as well as pheromone trails. To calculate the heuristic information, three well-known priority rules are adopted as fuzzy sets and then aggregated. When all artificial ants have terminated their constructions, genetic operators such as crossover and mutation are applied to generate new regions of the solution space. A local search is then performed to improve the performance quality of some of the solutions found. Moreover, at run-time the pheromone trails are locally as well as globally updated, and limited between lower and upper bounds. The proposed algorithm is experimented on a set of benchmark problems from the literature and compared with other metaheuristics.

Keywords: Scheduling; Ant colony optimization; Genetic operators; Total weighted tardiness; Sequence dependent setups.

1. Introduction

Scheduling problems have been considered for over five decades. In this context, some research efforts are concerned with due date related objectives such as the maximum/total tardiness, the total weighted tardiness and the number of tardy jobs. However, the total weighted tardiness as the performance criterion has attracted a large amount of literature on scheduling. Many researchers have studied the single-machine total weighted tardiness scheduling problem — denoted as $1//\sum w_jT_j$ by the three-field notation — and examined with different approaches. Lawler *et al*. have proved[1] that the $1//\sum w_jT_j$ problem is strongly NP-hard. The single-machine total weighted tardiness scheduling problem with sequence dependent setup times (STWTSDS) — denoted in the literature as $1/s_{ij}/\sum w_jT_j$ — is strongly NP-hard too, because the STWTSDS is clearly more complicated than the problem with sequence independent setup times.

To solve the STWTSDS, some solution methods have been developed which may generally be classified into two categories, heuristic and metaheuristic algorithms. Lee *et al*. have proposed[2] one of the best known constructive heuristics employing a priority rule, called the apparent tardiness cost with setups (ATCS) rule. Although the ATCS can quickly derive a feasible solution to the STWTSDS, the performance quality of

* Corresponding author. Tel./fax: +98-871-6660073; *E-mail: f.ahmadizar@uok.ac.ir*

its solution is not superior, particularly for large size problems. Cicirello and Smith have proposed[3] four improvement heuristics, namely, limited discrepancy search, heuristic-biased stochastic sampling, value-biased stochastic sampling and value-biased stochastic sampling seeded hill climber, as well as a simulated annealing (SA) approach. A beam search has been presented[4] by Valente and Alves. Lin and Ying have developed[5] some metaheuristics including a genetic algorithm (GA), a SA by means of a random swap and insertion search, and a tabu search by adopting a swap and an insertion tabu list. Liao and Juan have developed[6] an ant colony optimization (ACO) algorithm by introducing a new parameter for initializing pheromone trails and adjusting the timing of applying local search. Anghinolfi and Paolucci have proposed[7] another ACO algorithm in which a new scheme is applied to update pheromone trails. Anghinolfi and Paolucci have also developed[8] a discrete particle swarm optimization algorithm for the STWTSDS. Moreover, Tasgetiren *et al*. have proposed[9] a discrete differential evolution algorithm with excellent results by employing some constructive heuristics to generate the initial population.

In this paper, a new ACO algorithm is developed for the STWTSDS by incorporating fuzzy sets and genetic operators. Three well-known priority rules are adopted as fuzzy sets and then aggregated in order to calculate the heuristic information. In recent years, a number of hybrid algorithms combining ACO and GA have been developed for different problems (e.g., see Refs. 10–13). However, we combine ACO and GA in a manner somewhat similar to that in Ref. 12. That is, when all artificial ants have constructed their solutions, genetic operators such as crossover and mutation are applied to generate new solutions. Moreover, a local search is performed to improve some solutions found. The proposed ACO algorithm is then compared with other metaheuristics developed for the STWTSDS. Computational experiments on 120 benchmark problem instances from the literature show the effectiveness of the algorithm, especially in comparison with the existing ACO algorithms. We would like to emphasize that the main goal of this study is to develop a new ACO algorithm that can outperform other ACO algorithms available in the literature for the STWTSDS.

The remainder parts of the paper are organized as follows. Section 2 provides the problem statement. In Sec. 3, the ACO algorithm developed for the STWTSDS is presented. Section 4 gives the computational results. Finally, Sec. 5 is the conclusions.

2. Problem Definition

The STWTSDS can be expressed as follows. There are n independent jobs that have to be processed on a single machine without interruption. All jobs are available at time zero. The machine can process one job at a time. Each job j has a processing time p_j, a due date d_j, a weight w_j, and a setup time s_{ij} if it immediately follows job i in the job sequence and s_{0j} otherwise, i.e., if it is first in the job sequence.

Let π be a sequence of jobs such that $\pi = \{\pi_{(0)}, \pi_{(1)}, \ldots, \pi_{(n)}\}$, where $\pi_{(k)}$ is the index of the job scheduled at the kth position and $\pi_{(0)} = 0$ is a dummy job. The completion time of the job scheduled at the kth position can be calculate as

$$C_{\pi_{(k)}} = \sum_{l=1}^{k} s_{\pi_{(l-1)}\pi_{(l)}} + p_{\pi_{(l)}}, \tag{1}$$

and its tardiness as

$$T_{\pi_{(k)}} = \max\left(0, C_{\pi_{(k)}} - d_{\pi_{(k)}}\right). \tag{2}$$

The total weighted tardiness of job sequence π is finally defined as $F^{\pi} = \sum_{k=1}^{n} w_{\pi_{(k)}} T_{\pi_{(k)}}$. Then, the objective is to find a sequence such that the total weighted tardiness is minimized.

3. Proposed ACO Algorithm

ACO algorithms belonging to the class of constructive metaheuristics have been applied successfully to hard combinatorial optimization problems (e.g., see Refs. 14 and 15). These algorithms are inspired by the foraging behavior of real ants in finding shortest paths from their nest to food sources. Real ants are social insects which live in colonies. They have not visual cues but use a chemical substance, called pheromone, deposited on their paths for communicating among each other. Ants that select longer paths will get to the food and back more slowly than ants that selects shorter paths. As a greater amount of pheromone is deposited on shorter paths, such paths will be chosen by following ants with higher probability.

In the proposed ACO algorithm, each artificial ant probabilistically constructs step by step a solution as an order of jobs by completing at each step a partial

solution. The construction of solutions by artificial ants is guided by both the heuristic information and pheromone trails. The heuristic information is determined by a new approach combining three priority rules. When every ant in the colony has built its solution, crossover and mutation operators are implemented on the population of the constructed solutions. A local search is then performed to improve the best solution found in the iteration. Moreover, the pheromone trails are updated at run-time through local and global updating rules, and limited between lower and upper bounds. The flow chart of the proposed ACO algorithm is shown in Fig. 1.

3.1. *Pheromone trails*

In the proposed algorithm, a pheromone trail is associated with the assignment of a job to a position. Let $\tau_h(k, j)$ be the pheromone trail denoting the desire of placing job j in the kth position of a sequence at iteration h of the algorithm. The pheromone trails form a kind of adaptive memory of previously found solutions. Like most applications of ACO, at the beginning of the proposed algorithm (and also at each pheromone trails resetting), a fixed value τ_0 is assigned to all pheromone trails. Then, at run-time, the trail intensities are updated by applying local and global updating rules. Moreover, to prevent the algorithm from convergence to local optima, like the max–min ant system[16], the pheromone trails are always limited between a lower bound $\tau_{\min}$ and an upper bound $\tau_{\max}$ calculated as follows:

$$\tau_{\max} = 1/(G_1 \rho F^*), \qquad \tau_{\min} = G_2 \tau_{\max}, \tag{3}$$

where G_1 and G_2 are two positive parameters, ρ is the global pheromone trail evaporation rate, and F^* is the total weighted tardiness of the global best solution, i.e., the best solution found since the start of the algorithm.

It should be noted that, whenever a solution better than the current global best solution is found, $\tau_{\max}$ as well as $\tau_{\min}$ stated in Eq. (3) is modified and then, the pheromone trails are adjusted, that is, if a pheromone trail is smaller than the latest $\tau_{\min}$ or greater than the latest $\tau_{\max}$, it is set equal to $\tau_{\min}$ or $\tau_{\max}$, respectively. Obviously, the interval $[\tau_{\min}, \tau_{\max}]$ shifts to the right through modifying. This enforces that trails having a little amount of pheromone are increased to the new

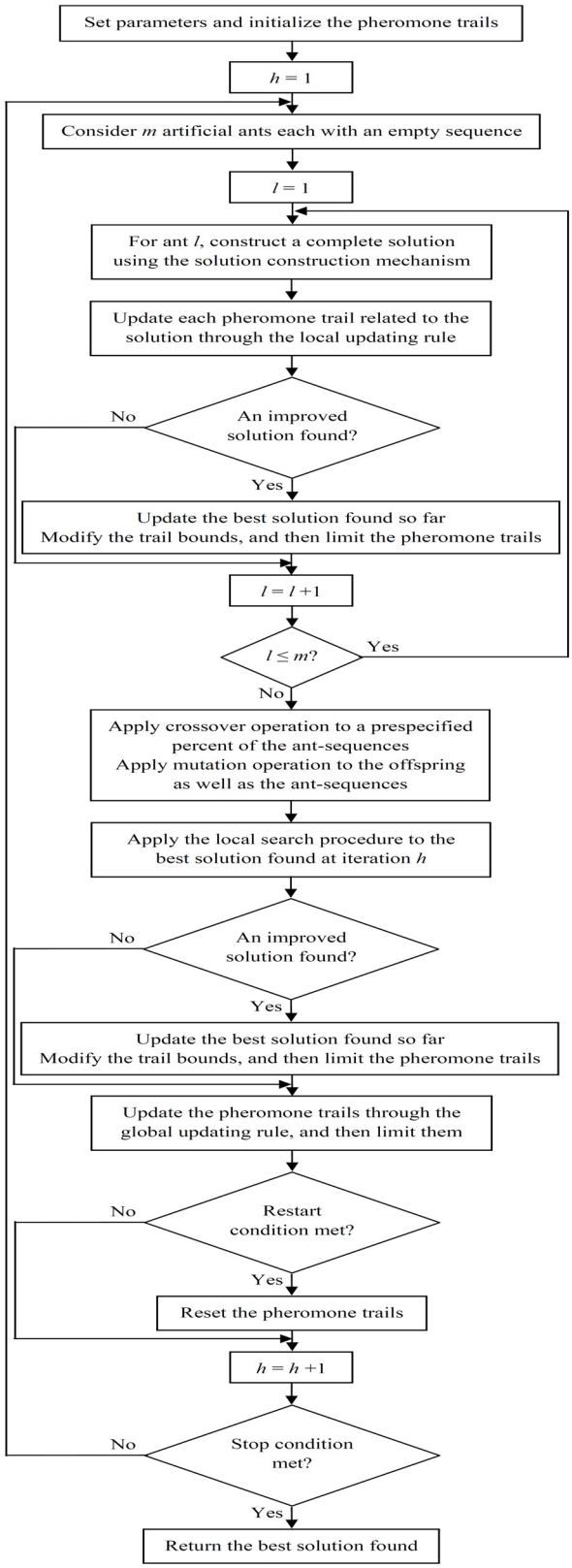

Fig. 1. Flow chart of the proposed ACO algorithm.

$\tau_{\min}$. In addition, the amounts of the pheromone trails related to the new global best solution are allowed to be increased to the new $\tau_{\max}$ — by means of the global updating rule (see Eq. (13).

3.2. *Heuristic information*

Let $\eta(k, j)$ be the heuristic information denoting the desire of placing job j in the kth position of a sequence. The heuristic information represents *a priori* information about the problem or run-time information provided by a source different from the artificial ants.

In the ACO algorithms proposed in Refs. 6 and 7, the dispatching rule of ATCS has been used as the heuristic information. The ATCS rule combines the following three well-known priority rules in a single ranking index[17]:

- WSPT (weighted shortest processing time) rule, which orders the jobs in non-increasing order of w_j/p_j;
- MS (minimum slack) rule, which schedules at time t the job with minimum slack where the slack of job j is calculated as $\max(d_j - p_j - t, 0)$;
- SST (shortest setup time) rule, which orders the jobs in non-decreasing order of their setup times.

According to the ATCS rule, at time t the job with maximum index is scheduled in which the index of job j at time t when job i has completed its processing on the machine is calculated as

$$I_j(t,i) = \frac{w_j}{p_j}\exp\left(-\frac{\max\left(d_j - p_j - t, 0\right)}{K_1\bar{p}}\right)\exp\left(-\frac{s_{ij}}{K_2\bar{s}}\right), \quad (4)$$

where $\bar{p}$ is the average of the processing times of the unscheduled jobs, $\bar{s}$ the average of the setup times of the unscheduled jobs, K_1 the due date related scaling parameter and K_2 the setup time related scaling parameter.

In this paper, to combine the three priority rules mentioned above, a new approach is proposed as follows. Each of the rules is firstly adopted as a fuzzy set. To give an example, the WSPT rule may be viewed as: if the w_j/p_j of job j is large (where the concept "large" is fuzzy), it must be scheduled with a high desirability (representing the heuristic information). Accordingly, the WSPT rule is adopted as the fuzzy set of "jobs with large w_j/p_j". Similarly, the MS and SST rules are, respectively, adopted as the fuzzy sets of "jobs with small slack" and "jobs with small setup time". Let $R_j^{(W)}$, $R_j^{(M)}$ and $R_j^{(S)}$ be the grade of membership of unscheduled job j (to be placed in the kth position of a sequence) in the fuzzy sets associated with the WSPT, MS and SST rules, respectively. These grades may be calculated in several manners. In this study, however, they are given by

$$R_j^{(W)} = \frac{w_j/p_j}{\sum_{u\in U} w_u/p_u}, \quad (5)$$

$$R_j^{(M)} = \frac{1/(1+\max(d_j - p_j - t, 0))}{\sum_{u\in U} 1/(1+\max(d_u - p_u - t, 0))}, \quad (6)$$

$$R_j^{(S)} = \frac{1/(1+s_{ij})}{\sum_{u\in U} 1/(1+s_{iu})}. \quad (7)$$

where U is the set of all unscheduled jobs. Note that, to avoid division by zero, 1 is added to the slack in Eq. (6) as well as to the setup time in Eq. (7).

$R_j^{(w)}$, $R_j^{(M)}$ and $R_j^{(s)}$ can then be aggregated in order to calculate the heuristic information. As an aggregation operator, four operators have been considered, including the maximum, average, product and minimum operators. Our experimental analyses show that the product operator can obtain better results than the others. Consequently, the desire of placing job j in the kth position — given that job i has been scheduled in the (k-1)th position and completed at time t — is calculated as

$$\eta(k,j) = R_j^{(W)} R_j^{(M)} R_j^{(S)}. \quad (8)$$

Clearly, the proposed approach stated in Eqs. (5)–(8) guarantees that the larger w_j/p_j, the smaller $\max(d_j - p_j - t, 0)$ and s_{ij}, the higher $\eta(k, j)$. In addition, $\eta(k, j)$ is always between 0 and 1.

3.3. *Solution construction*

Each ant constructs a sequence of jobs by starting with an empty sequence and then iteratively appending an unscheduled job to the partial solution until all jobs are scheduled. At each step, an unscheduled job is chosen by applying a transition rule so-called pseudo-random proportional rule[18].

The pseudo-random proportional rule is based upon a parameter q_0 between 0 and 1 which determines the relative importance of exploitation versus exploration.

A random number q uniformly distributed in [0, 1] is then generated. If $q \le q_0$, an ant at position k selects the unscheduled job j for which (exploitation)

$$j = \arg\max\left[\left(\tau_h(k,j)\right)^{\alpha}\left(\eta(k,j)\right)^{\beta}\right], \qquad (9)$$

where α and β are two positive parameters determining the relative importance of the pheromone trail versus the heuristic information. Otherwise, the ant selects an unscheduled job j according to the following probability distribution (exploration):

$$p(k,j) = \frac{\left(\tau_h(k,j)\right)^{\alpha}\left(\eta(k,j)\right)^{\beta}}{\sum_{u\in U}\left(\tau_h(k,u)\right)^{\alpha}\left(\eta(k,u)\right)^{\beta}}, \qquad (10)$$

3.4. *Local updating rule*

While constructing a solution by an ant, the local updating rule is applied to the pheromone trails related to the selected jobs. So, if job j is placed at the kth position of the sequence (at iteration h), the amount of the associated pheromone trail is updated as follows:

$$\tau_h(k,j) = (1-\rho')\tau_h(k,j) + \rho'\tau_{\min}, \qquad (11)$$

where ρ' is the local pheromone trail evaporation rate.

According to Eq. (11), $\tau_h(k, j)$ is decreased from its earlier value (when $\tau_h(k, j) = \tau_{\min}$, it of course remains unchanged). Hence, the effect of the local updating rule is to make choice of putting job j at the kth position less desirable for the next ants at iteration h in order to achieve diversification.

3.5. *Genetic operators*

As mentioned earlier, when all artificial ants in the colony have terminated their constructions, crossover and mutation genetic operators are implemented on the population of the constructed solutions to generate new regions of the solution space. These operators have a role in the diversification of the search permitting a better exploration of the solution space. In the other hand, since they perform a search in the neighborhood of the ant-sequences, they also have a role in the intensification of the search.

In this study, the crossover and mutation operators developed[19] by Chou for the problem with sequence independent setup times are adopted. The crossover operator combines information from two solutions as parents by the two point exchange method to produce four offspring, i.e., to generate four new solutions so that each of which has some characteristics of each parent. Parents are selected to undergo the crossover according to the bias roulette wheel method; as a result, the constructed solutions with lower objective function value are expected to have a higher chance of being selected. The number of parents is set equal to a percentage of the number of artificial ants. The percentage is a parameter defined as the crossover rate. Moreover, the mutation operator selects two genes (i.e., two jobs in a given sequence) at random and exchanges them. The swapping mutation mechanism is applied to the offspring as well as the solutions constructed by artificial ants in order to produce random variation.

3.6. *Local search*

To improve the performance of the algorithm, it is hybridized with a local search procedure having a role in the intensification of the search. We have tested various methods for the intensification phase, and found that the local search procedure proposed[7] by Anghinolfi and Paolucci is superior. The procedure performs a series of random insert moves until no improvement is found, and then executes a series of swap moves. Whenever a swap move is not able to find an improved solution, a new series of random insert moves is started. Moreover, after completing every $n \times (n\text{-}1)$ series of insert and swap moves, the procedure performs a random restart as in the iterated local search. The maximum number of random restarts allowed is limited by $n/5$. Since the local search procedure is clearly time-consuming, to achieve a good trade off, it is applied only to the best solution generated at each iteration.

3.7. *Global updating rule*

Once all ants in the colony have completed their solutions, the global updating rule is applied to the pheromone trails in a way similar to that in Ref. 20. First, each of the pheromone trails is evaporated (at the end of iteration h) as follows:

$$\tau_{h+1}(k,j) = (1-\rho)\tau_h(k,j). \qquad (12)$$

Then, the amount of each pheromone trail related to the global best solution is increased. If job j is placed at the kth position of the solution, the associated pheromone trail is updated as follows:

$$\tau_{h+1}(k,j) = \tau_{h+1}(k,j) + \rho\Omega/F^{*}, \qquad (13)$$

where Ω is a parameter determining the importance of the global best solution. The global updating rule allows the intensification of the search during the next iteration.

4. Computational Results

The proposed ACO algorithm has been coded in Visual C^{++} and run on a Pentium 4, 2 GHz PC with 2 GB memory. To evaluate the performance of the algorithm, it has been tested on a set of 120 benchmark problems, each with 60 jobs, from Cicirello and Smith[3] (available at http://www.ozone.ri.cmu.edu/benchmarks.html). Each problem instance is characterized by three factors: the due date tightness factor specified by {0.3, 0.6, 0.9}, the due date range factor specified by {0.25, 0.75}, and the setup time severity factor specified by {0.25, 0.75}. For each of the 12 combinations of factor values, 10 problem instances with 60 jobs were generated.

4.1. *Parameter settings*

For setting the numeric parameters of the algorithm, in the preliminary experiment various combinations of the parameter values have been tested, where the following values have been superior and used for all further studies: 20 artificial ants in the colony, $G_1 = 100$, $G_2 = 0.001$, $q_0 = 0.99$, $\alpha = 5$, $\beta = 0.9$, $\rho = 0.03$, $\rho' = 0.1$ and $\Omega = 4$. In addition, τ_0 has been set to 1 and the crossover rate to 0.9. The pheromone trails are reset if no improvement can be made for 30 successive iterations, and the algorithm terminates when either the total number of iterations reaches 150 or no improvement can be made for 50 successive iterations.

4.2. *Contribution of the genetic operators*

To show the effect of incorporating the genetic operators, an experimental test has been conducted by temporarily removing them from the algorithm. For convenience, we denote the proposed ACO algorithm, which includes the genetic operators, as ACO_{GO}, and the algorithm without the genetic operators as ACO_{-GO}. Ten problem instances randomly chosen have been solved by ACO_{GO} as well as ACO_{-GO}. Each test problem has been tested for five trails, and the average total weighted tardiness achieved has then been chosen. In addition, to make a fair comparison, the maximum CPU time has been set to 45 seconds. Fig. 2 gives the percentages of improvement of ACO_{GO} over ACO_{-GO}. As seen, incorporating the genetic operators can improve the performance of the ACO algorithm.

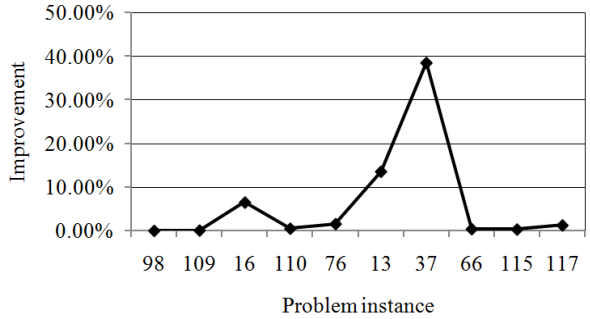

Fig. 2. Improvement of ACO_{GO} over ACO_{-GO}.

4.3. *Contribution of the local search*

To show the effect of the local search, another experimental test has been conducted by temporarily removing it from the algorithm. The algorithm with (ACO_{GO}) as well as without (ACO_{GO}-LS) local search has been tested on ten problem instances randomly chosen. Again, the maximum CPU time has been set to 45 seconds, each test problem has been tested for five trails, and the average total weighted tardiness achieved has then been chosen. Fig. 3 gives the percentages of improvement of ACO_{GO} over ACO_{GO}-LS. It is observed that incorporating the local search procedure can improve the performance of the ACO algorithm.

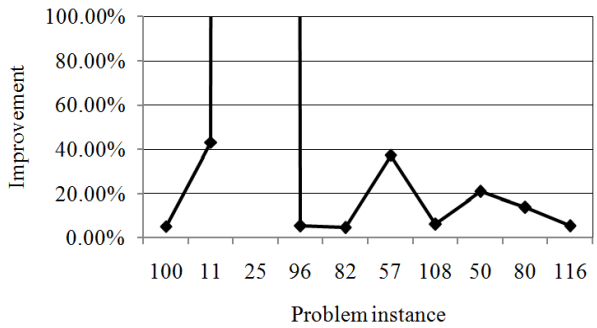

Fig. 3. Improvement of ACO_{GO} over ACO_{GO}-LS.

4.4. *Performance analysis of the proposed algorithm*

To assess the performance of the ACO algorithm proposed to solve the STWTSDS, each of the problem instances has been tested for ten trails. The numerical results are summarized in Table 1, which gives, for each

test problem, the best and average total weighted tardiness achieved. In the table, the BKS is the best known solution (the BKSs include our algorithm). ACO_{GO} has improved the BKSs for two problem instances, test problems 52 and 98, and found solutions equal to the BKSs for 39 test problems. In addition, the average CPU time for each run has been less than 44 seconds (with a minimum of 0.004 and a maximum of 73.843). From Table 1, as the best and average objective function values are relatively close to each other, it can be concluded that ACO_{GO} is robust.

Furthermore, Table 2 gives a comparison between the proposed algorithm and other ACO algorithms available in the literature for the STWTSDS, including:

- ACO_{LJ}: the ACO algorithm proposed[6] by Liao and Juan,
- ACO_{AP}: the ACO algorithm proposed[7] by Anghinolfi and Paolucci,

and also other well-known heuristics and metaheuristics from the literature, including:

- RBS: the recovering beam search proposed[4] by Valente and Alves,
- SA: the simulated annealing approach proposed[5] by Lin and Ying,
- GA: the genetic algorithm proposed[5] by Lin and

Table 1. Performance of the proposed ACO algorithm on the benchmark problems.

		ACO_{GO}				ACO_{GO}				ACO_{GO}	
No.	BKS	Best	Average	No.	BKS	Best	Average	No.	BKS	Best	Average
1	474	516	603.3	41	69102	69627	70529.8	81	383485	383485	384507.8
2	4902	5046	5153.4	42	57487	57679	58336.0	82	409544	409544	411965.5
3	1465	1593	1659.8	43	145310	146068	147097.3	83	458752	458863	459849.0
4	5946	6046	6241.4	44	35289	35289	35905.0	84	329670	329670	330623.0
5	4084	4224	4325.2	45	58935	59281	59868.9	85	554766	555328	556646.8
6	5788	6757	7020.0	46	34764	34887	35431.5	86	361417	361417	364029.6
7	3350	3458	3596.2	47	73005	73621	74204.7	87	398551	398670	399780.7
8	114	121	134.4	48	64612	65138	65644.9	88	433186	433939	434764.2
9	5803	5914	6117.2	49	77641	78424	78968.1	89	410092	410092	410597.1
10	1799	1871	1991.4	50	31565	31705	32131.4	90	401653	401959	402400.1
11	3294	3811	4242.0	51	49907	51139	51798.0	91	340030	348056	351601.0
12	0	0	0	52	93973	93973	97709.8	92	361152	361514	365948.2
13	4194	4379	5114.4	53	84841	86364	88276.3	93	404548	408201	409795.6
14	2268	2761	2930.7	54	118809	118862	120947.3	94	332983	333535	334628.4
15	964	1216	1319.4	55	66006	67911	69176.0	95	517011	522717	526292.5
16	3876	4178	4970.4	56	75367	75589	76208.6	96	457631	458971	464550.0
17	61	125	178.9	57	64552	65815	69434.6	97	409263	410755	413900.7
18	857	1195	1385.0	58	45322	47159	47687.5	98	522093	522093	528477.6
19	0	0	70.4	59	52001	54553	55029.3	99	364442	368603	370459.5
20	2111	2485	2765.1	60	60765	64093	65711.7	100	431736	434118	443820.8
21	0	0	0	61	75916	75916	76400.9	101	352990	352990	353454.5
22	0	0	0	62	44769	44781	44944.4	102	492748	493846	494738.1
23	0	0	0	63	75317	75317	75811.0	103	378602	378602	380198.0
24	1033	1042	1056.5	64	92572	92572	92622.4	104	357963	358017	358372.7
25	0	0	0	65	126696	126696	127627.0	105	450806	450806	451346.5
26	0	0	0	66	59685	59685	59988.3	106	454379	454983	455955.0
27	0	0	0	67	29390	29390	29454.1	107	352766	352766	354426.5
28	0	0	0	68	22120	22120	22263.7	108	460793	461452	463608.1
29	0	0	0	69	64632	71118	71285.9	109	413004	413019	414462.3
30	0	0	73.6	70	75102	75102	75102.0	110	418769	418769	428679.2
31	0	0	0	71	145007	146957	148147.4	111	342752	343953	347353.0
32	0	0	0	72	43904	45348	47427.8	112	367110	370822	376086.2
33	0	0	0	73	28785	28824	29551.7	113	260176	260288	269393.7
34	0	0	0	74	30313	31952	32482.3	114	464136	464495	468019.1
35	0	0	0	75	21602	21991	22990.9	115	457289	458385	461126.3
36	0	0	0	76	53555	54758	57667.1	116	527459	533160	539378.1
37	107	354	596.8	77	31937	32239	33942.8	117	502840	504161	511437.3
38	0	0	0	78	19462	20452	20868.5	118	349749	353897	358553.3
39	0	0	0	79	114999	117190	120458.8	119	573046	578827	579858.1
40	0	0	0	80	18157	18157	20166.9	120	396183	399756	401077.8

Ying,

- TS: the tabu search proposed[5] by Lin and Ying,
- DPSO: the discrete particle swarm optimization algorithm proposed[8] by Anghinolfi and Paolucci,
- DDE: the discrete differential evolution algorithm proposed[9] by Tasgetiren *et al*.

In Table 2, to be consistent with the other algorithms, the best results achieved by ACO_{GO} are used in the comparison. Moreover, for each problem instance, if the total weighted tardiness of a given solution (obtained by any algorithm) is F, the solution quality is then measured by the mean percentage difference from the BKS as (F - BKS) / BKS × 100.

In comparison with ACO_{LJ}, ACO_{GO} obtains better solutions for 98 problems and equal solutions for 16 problems. While in comparison with ACO_{AP}, ACO_{GO} obtains better solutions for 65 problems and equal solutions for 38 problems. The results concerning average of the deviations of all 120 test problems (shown in the last row of Table 2) confirm that the proposed ACO algorithm outperforms the other ACO algorithms, ACO_{LJ} and ACO_{AP}, as is the main goal of the paper. It is noted that, even if those instances with zero weighted tardiness for which ACO_{LJ} has not obtained an optimal solution (e.g., test problem 30) are not considered, ACO_{LJ} gives an average deviation equal to 34.72%. In such a situation, ACO_{AP} gives an average deviation equal to 6.24%.

In comparison with RBS, ACO_{GO} finds better solutions for 105 problems and equal solutions for 12 problems. In comparison with SA as well as GA, ACO_{GO} obtains better solutions for 98 problems and equal solutions for 20 problems. In comparison with TS, ACO_{GO} finds better solutions for 99 problems and equal solutions for 18 problems. However, in comparison with DPSO and DDE, ACO_{GO} obtains better solutions for 25 and 5 problems and equal solutions for 38 and 42 problems, respectively. The results concerning average of the deviations of all 120 test problems confirm that the proposed ACO algorithm outperforms RBS, SA, GA and TS. Even if those instances with zero weighted tardiness for which RBS has not obtained an optimal solution are not considered, RBS then gives an average deviation equal to 85.59%. In such a situation, SA, GA and TS give an average deviation equal to 14.98%, 14.65% and 18.04%, respectively. Although DPSO and DDE outperform the proposed algorithm in terms of average deviation, it should be highlighted here that the results of DPSO (reported in Ref. 8) as well as DDE (reported in Ref. 9) shown in Table 2 are the best results amongst several variants, whereas those of ACO_{GO} are the best results achieved using only one configuration.

Finally, the average CPU times (in seconds) of the other algorithms (for each run) provided by the literature have been as follows:

- ACO_{LJ} (run on a Pentium 4, 2.8 GHz PC): 4.99,
- ACO_{AP} (run on a Pentium 4, 2.8 GHz PC): 65,
- RBS (run on a Pentium 4, 2.8 GHz PC): 0.18,
- SA, GA as well as TS (run on a Pentium 4, 1.4 GHz PC): 27,
- DPSO (run on a Pentium 4, 2.8 GHz PC): 22.6,
- DDE (run on a Pentium 4, 3.2 GHz PC): 9.

Because of the differences in the computer platforms used, a direct comparison of the computation times, of course, is difficult. However, the average CPU time of the proposed ACO algorithm is quite low and reasonable.

5. Conclusions

In this paper, a novel ant colony algorithm is developed for the single-machine total weighted tardiness scheduling problem with sequence dependent setup times. The main features of the proposed algorithm are that the heuristic information is calculated by adopting three well-known priority rules as fuzzy sets and then aggregating them, and that genetic operators are employed to search new regions of the solution space. Based on the heuristic information and pheromone trails, artificial ants construct solutions to the problem. Once all artificial ants have constructed their solutions, crossover and mutation genetic operators are implemented on the population of the solutions. Moreover, a local search is performed to improve the best solution found at the iteration. Undoubtedly, another feature of the proposed algorithm is that, to make the search more directed, at run-time the pheromone trails are purposefully updated (locally and globally) and limited (between lower and upper bounds). The proposed ant colony algorithm has been tested on a set of benchmark problems from the literature, and compared with other algorithms. The computational results demonstrate the superiority of the algorithm, in particular when comparing with the existing ant colony algorithms. The proposed algorithm has also found new best solutions for two instances.

Table 2. Comparison of the proposed ACO algorithm with the other algorithms.

No.	ACO_{LJ}	ACO_{AP}	RBS	SA	GA	TS	DPSO	DDE	ACO_{GO}
1	88.61	8.23	271.3	54.43	44.30	55.70	12.03	0.00	8.86
2	28.66	3.67	64.52	11.93	4.90	3.67	3.79	0.00	2.94
3	36.72	20.75	91.81	22.32	22.73	22.73	9.83	0.00	8.74
4	34.59	5.72	104.0	11.74	9.75	13.12	3.36	0.00	1.68
5	27.69	4.38	67.58	20.45	14.15	19.76	6.24	0.00	3.43
6	0.00	21.41	94.92	24.62	26.95	33.52	18.04	14.93	16.74
7	23.88	7.40	61.76	14.54	10.24	13.40	4.90	0.00	3.22
8	39.47	13.16	450.9	45.61	24.56	50.00	15.79	0.00	6.14
9	29.07	5.01	38.86	14.65	9.41	15.96	6.03	0.00	1.91
10	30.35	7.34	52.58	12.34	13.29	20.34	5.34	0.00	4.00
11	54.61	16.97	182.5	41.47	17.40	49.91	10.78	0.00	15.70
12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
13	42.16	9.61	103.7	48.38	35.55	40.58	5.63	0.00	4.41
14	77.91	27.91	163.1	42.55	34.26	47.75	21.21	0.00	21.74
15	192.8	29.15	353.8	95.54	51.24	91.18	29.67	0.00	26.14
16	58.75	15.63	131.2	44.84	27.45	45.51	6.48	0.00	7.79
17	626.2	109.8	1283.6	234.4	262.3	323.0	22.95	0.00	104.9
18	140.3	44.34	260.2	98.13	87.86	126.9	13.30	0.00	39.44
19	26500	0.00	62400	22800	20800	23100	0.00	0.00	0.00
20	99.15	20.56	190.1	53.96	40.55	53.62	26.72	0.00	17.72
21	0.00	0.00	40500	0.00	0.00	0.00	0.00	0.00	0.00
22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
24	50.15	1.36	454.9	6.29	2.90	6.49	0.97	0.00	0.87
25	0.00	0.00	55800	0.00	0.00	0.00	0.00	0.00	0.00
26	0.00	0.00	76300	0.00	0.00	0.00	0.00	0.00	0.00
27	13700	0.00	134000	0.00	0.00	0.00	0.00	0.00	0.00
28	1900	0.00	180400	0.00	0.00	0.00	0.00	0.00	0.00
29	0.00	0.00	22400	0.00	0.00	0.00	0.00	0.00	0.00
30	37200	13000	137500	16500	18000	26900	0.00	0.00	0.00
31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
34	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
35	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
36	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
37	1842.1	273.8	3788.8	605.6	705.6	843.0	73.83	0.00	230.8
38	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
39	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
41	6.48	1.67	11.55	3.60	3.98	3.02	0.00	0.20	0.76
42	5.96	0.63	12.79	2.29	3.53	1.24	0.00	0.04	0.33
43	3.00	0.95	6.01	1.31	2.05	1.37	0.39	0.00	0.52
44	5.95	0.12	18.37	1.02	1.35	1.65	0.12	0.00	0.00
45	6.11	0.00	10.46	1.18	0.63	1.52	0.41	0.00	0.59
46	11.55	1.59	16.51	3.41	2.69	1.60	0.12	0.00	0.35
47	4.12	1.07	14.59	1.34	2.73	1.74	0.51	0.00	0.84
48	6.56	1.00	16.45	0.85	1.51	1.86	0.00	0.00	0.81
49	5.01	1.01	12.41	2.18	2.57	1.82	0.17	0.00	1.01
50	12.49	0.83	16.80	3.94	4.80	3.90	0.78	0.00	0.44
51	11.80	1.73	23.76	7.10	5.47	8.79	0.00	0.04	2.47
52	11.95	2.10	11.16	6.51	5.56	8.28	0.21	0.67	0.00
53	13.41	2.92	25.94	7.62	8.20	12.10	2.42	0.00	1.80
54	4.48	1.66	5.08	8.51	8.95	7.16	0.00	0.35	0.04
55	12.82	4.30	34.57	9.16	5.71	10.32	4.00	0.00	2.89
56	16.06	1.51	24.78	7.90	4.77	6.52	0.16	0.00	0.29
57	4.48	3.07	15.77	7.39	10.04	10.80	0.04	0.00	1.96
58	16.39	3.79	29.40	8.06	8.49	6.09	0.79	0.00	4.05
59	9.42	3.92	30.65	6.53	7.46	8.06	0.00	0.40	4.91
60	19.48	3.40	37.53	13.55	13.31	13.72	4.24	0.00	5.48

Table 2. (Continued).

No.	ACO_{LJ}	ACO_{AP}	RBS	SA	GA	TS	DPSO	DDE	ACO_{GO}
61	5.83	0.00	33.43	0.63	1.28	0.71	0.00	0.00	0.00
62	3.79	0.22	18.37	0.88	0.46	0.00	0.00	0.00	0.03
63	3.67	0.00	10.94	0.00	1.04	1.69	0.00	0.00	0.00
64	2.74	0.00	21.58	0.00	0.87	0.14	0.00	0.00	0.00
65	4.25	0.00	16.52	0.96	2.01	1.63	0.00	0.00	0.00
66	6.02	0.00	19.94	0.25	2.73	0.66	0.00	0.00	0.00
67	9.95	0.00	32.51	0.01	0.00	0.01	0.00	0.00	0.00
68	19.20	0.00	41.27	1.79	3.35	0.13	0.00	0.00	0.00
69	0.00	10.04	26.62	10.86	11.88	10.30	10.04	10.04	10.04
70	8.33	0.00	20.21	0.72	0.00	0.60	0.00	0.00	0.00
71	7.77	0.56	13.21	5.44	3.93	5.64	0.53	0.00	1.34
72	24.93	4.34	61.73	13.13	11.00	6.83	0.20	0.00	3.29
73	18.40	0.43	68.92	2.16	5.22	2.44	0.00	0.00	0.14
74	11.26	6.90	84.74	10.60	18.70	10.10	1.39	0.00	5.41
75	26.14	5.21	123.2	6.06	6.24	1.21	0.00	0.00	1.80
76	24.82	3.25	35.72	2.80	8.87	9.72	0.64	0.00	2.25
77	16.66	2.52	59.91	8.75	9.37	9.71	0.00	0.94	0.95
78	27.40	5.44	89.54	11.48	13.83	10.44	1.02	0.00	5.09
79	6.13	2.53	35.82	7.43	5.32	6.38	0.00	0.00	1.91
80	45.78	3.68	124.6	23.06	12.00	13.30	0.00	0.00	0.00
81	1.14	0.00	1.60	0.52	0.39	0.73	0.06	0.00	0.00
82	0.89	0.11	0.65	0.35	0.77	0.35	0.00	0.00	0.00
83	1.24	0.03	0.48	0.54	0.86	0.49	0.01	0.00	0.02
84	0.32	0.00	0.15	0.22	0.50	0.55	0.00	0.00	0.00
85	1.32	0.00	1.10	0.06	0.12	0.37	0.07	0.04	0.10
86	1.05	0.07	1.07	0.82	0.97	1.01	0.00	0.00	0.00
87	0.75	0.03	2.37	0.33	0.22	0.54	0.00	0.03	0.03
88	0.86	0.28	1.11	0.73	0.41	0.71	0.08	0.00	0.17
89	0.55	0.00	1.78	0.21	1.14	0.22	0.00	0.00	0.00
90	0.61	0.08	0.94	0.51	0.14	0.30	0.00	0.00	0.08
91	1.59	0.00	0.34	1.83	1.94	1.46	0.88	0.14	2.36
92	1.13	0.07	1.00	1.26	1.10	1.29	0.00	0.00	0.10
93	2.09	0.99	1.48	3.07	2.82	2.92	0.54	0.00	0.90
94	0.77	0.02	0.94	2.55	3.08	1.92	0.00	0.01	0.17
95	1.92	0.03	3.27	2.93	0.87	1.95	0.81	0.00	1.10
96	0.84	0.84	1.83	2.08	3.18	1.42	0.37	0.00	0.29
97	2.47	0.50	3.32	2.44	2.58	0.94	0.40	0.00	0.36
98	2.11	0.91	1.04	2.38	3.18	2.14	0.10	0.27	0.00
99	1.55	1.09	1.09	2.97	4.58	2.86	0.19	0.00	1.14
100	2.33	1.20	1.57	2.52	1.90	3.38	0.23	0.00	0.55
101	0.67	0.00	0.46	0.27	0.33	0.12	0.00	0.00	0.00
102	0.66	0.24	0.91	0.72	0.64	0.23	0.07	0.00	0.22
103	0.35	0.00	2.36	0.58	0.76	0.56	0.00	0.00	0.00
104	0.78	0.02	0.30	0.10	0.47	0.07	0.00	0.00	0.02
105	0.91	0.00	1.71	0.00	0.94	0.32	0.00	0.00	0.00
106	1.15	0.16	0.90	0.34	0.32	0.61	0.17	0.00	0.13
107	0.38	0.17	1.22	0.17	1.22	0.44	0.03	0.00	0.00
108	1.14	0.14	1.37	0.57	0.87	0.42	0.00	0.00	0.14
109	0.46	0.10	1.51	0.05	0.66	0.26	0.00	0.00	0.00
110	0.55	0.00	1.51	0.54	0.78	0.17	0.00	0.00	0.00
111	1.31	1.17	2.56	2.49	2.13	1.88	0.00	0.00	0.35
112	1.67	1.64	1.46	1.75	3.13	2.75	0.58	0.00	1.01
113	0.84	0.09	4.28	1.10	0.41	0.60	0.00	0.27	0.04
114	1.33	0.13	3.21	2.06	2.00	1.83	0.00	0.29	0.08
115	0.42	0.11	0.54	2.36	1.50	2.87	0.13	0.00	0.24
116	0.00	1.02	2.24	3.63	2.84	2.92	0.95	0.63	1.08
117	1.88	0.77	2.43	3.03	4.61	2.05	0.07	0.00	0.26
118	0.68	1.76	4.71	1.96	3.18	2.46	0.28	0.00	1.19
119	1.92	0.15	0.48	1.48	1.16	1.12	0.00	0.00	1.01
120	0.61	0.34	1.28	1.93	2.98	2.71	0.00	0.00	0.90
Average	694.4	114.5	5990.7	342.2	337.7	434.4	2.76	0.24	5.03

References

1. E. L. Lawler, A 'pseudo polynomial' algorithm for sequencing jobs to minimize total tardiness, *Ann. Discrete Math.* **1** (1977) 331–342.
2. Y. H. Lee, K. Bhaskaram and M. Pinedo, A heuristic to minimize the total weighted tardiness with sequence-dependent setups, *IIE Trans.* **29** (1997) 45–52.
3. V. A. Cicirello and S. F. Smith, Enhancing stochastic search performance by value-based randomization of heuristics, *J. Heuristics* **11** (2005) 5–34.
4. J. M. S. Valente and R. A. F. S. Alves, Beam search algorithms for the single machine total weighted tardiness scheduling problem with sequence dependent setups, *Comput. Oper. Res.* **35** (2008) 2388–2405.
5. S. W. Lin and K. C. Ying, Solving single-machine total weighted tardiness problems with sequence-dependent setup times by meta-heuristics, *Int. J. Adv. Manuf. Tech.* **34** (2007) 1183–1190.
6. C. J. Liao and H. C. Juan, An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups, *Comput. Oper. Res.* **34** (2007) 1899–1909.
7. D. Anghinolfi and M. Paolucci, A new ant colony optimization approach for the single machine total weighted tardiness scheduling problem, *Int. J. Oper. Res.* **5** (2008) 44–60.
8. D. Anghinolfi and M. Paolucci, A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times, *Eur. J. Oper. Res.* **193** (2009) 73–85.
9. M. F. Tasgetiren, Q. K. Pan and Y. C. Liang, A discrete differential evolution algorithm for the single machine total weighted tardiness problem with sequence dependent setup times, *Comput. Oper. Res.* **36** (2009) 1900–1915.
10. L. Y. Tseng and S. C. Chen, A hybrid metaheuristic for the resource-constrained project scheduling problem, *Eur. J. Oper. Res.* **175** (2006) 707–721.
11. Z. J. Lee, S. F. Su, C. C. Chuang and K. H. Liu, Genetic algorithm with ant colony optimization (GA–ACO) for multiple sequence alignment, *Appl. Soft Comput.* **8** (2008) 55–78.
12. G. H. Hajimirsadeghi, M. Nabaee and B. N. Araabi, Ant colony optimization with a genetic restart approach toward global optimization, in *Communications in Computer and Information Science* 6, eds. H. Sarbazi-Azad *et al*. (Springer-Verlag, Berlin Heidelberg, 2008), pp. 9–16.
13. T. Kolasa and D. Król, ACO–GA approach to paper-reviewer assignment problem in CMS, in *Lecture Notes in Computer Science* 6071, eds. P. Jedrzejowicz *et al*. (Springer-Verlag, Berlin Heidelberg, 2010), pp. 360–369.
14. X. M. You, S. Liu and Y. M. Wang, Quantum dynamic mechanism-based parallel ant colony optimization algorithm, *Int. J. Comput. Int. Sys.* **3** (2010) 101–113.
15. F. Ahmadizar and F. Barzinpour, A hybrid algorithm to minimize makespan for the permutation flow shop scheduling problem, *Int. J. Comput. Int. Sys.* **3**(6) (2010) 853–861.
16. T. Stützle and H. H. Hoos, Max–Min ant system, *Future Gener. Comp. Sy.* **16** (2000) 889–914.
17. M. Pinedo, *Scheduling*: *Theory, Algorithms, and Systems* (Prentice-Hall, Englewood Cliffs, NJ, 1995).
18. M. Dorigo and L. M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE T. Evolut. Comput.* **1** (1997) 53–66.
19. F. D. Chou, An experienced learning genetic algorithm to solve the single machine total weighted tardiness scheduling problem, *Expert. Syst. Appl.* **36** (2009) 3857–3865.
20. F. Ahmadizar, M. Ghazanfari and S.M.T. Fatemi Ghomi, Group shops scheduling with makespan criterion subject to random release dates and processing times, *Comput. Oper. Res.* **37** (2010) 152–162.