An ant colony optimization for single-machine weighted tardiness scheduling with sequence-dependent setups

FARHAD KOLAHAN, MAHDI ABACHIZADEH, SAEED SOHEILI
Department of Mechanical Engineering
Ferdowsi University of Mashhad
Mashhad
IRAN

Abstract: In this paper, an Ant Colony Optimization (ACO) method is applied to the problem of scheduling a single machine with sequence-dependent setup times. The objective is to minimize weighted tardiness of all jobs. Three sets of randomly generated problems of different sizes are solved with the proposed solution technique. To examine the performance of ACO algorithm, the same problems have been also solved using Tabu Search (TS) method. It is shown that a well-tuned ACO, with proper definition of heuristic function, can outperform TS in terms of solution quality while taking considerably longer CPU time.

Key-Words: scheduling, ant colony optimization, tabu search, setup time, weighted tardiness, single-machine

1 Introduction

Different variations of job shop scheduling (JSP) problems have extensively been studied in the last fifty years. Single machine scheduling with sequence dependent setups, is one of the most difficult one in the classical scheduling theory. Such problems are NP-hard in nature and can not be solved efficiently by traditional solution procedures such as branch and bound. Nevertheless, these problems can be good candidates for evaluating new solution procedures. Among different heuristics, genetic algorithm (GA), tabu search (TS) and simulated annealing (SA) have been successfully applied on JSP [1-3].

Single machine scheduling problem which is investigated in this paper is generally smaller than multi-machine problems in its size. However, its capability for modeling real life flexible production systems and also multi-machine environments with a single bottleneck, has kept it a fresh topic of research work.

2 Introducing Job Shop Problem

The JSP problem optimized here is as follows. There are n different jobs which can be all available for processing at time zero; nevertheless, the machine can only do a distinctive job at each certain time. For each job shown as j, four predefined parameters are involved which are processing time (p_j), due date (d_j), weight (w_j) - that can also be considered as cost per unit time paid due to tardiness - and finally, the setup time (s_{ij}). The last one is used when the job j immediately follows job i. It is notable that considering sequence-dependent setup times increases the complexity of the weighted tardiness as a NP-hard problem.

Among many performance criteria proposed in literature, weighted tardiness is more commonly used. Therefore, in this work, the objective is to implement the ACO technique in order to find the sequence of jobs that minimizes total weighted tardiness of all jobs.

3 Ant Colony Optimization and Its Application to JSP

This algorithm was first introduced by M. Dorigo et al. in 1997 [4] and up till now has been extensively applied on different types of problems such as traveling salesperson problem, vehicle routing problem [5], quadratic assignment problem [6], and JSP. Ant colony is increasingly used to solve a variety of job shop scheduling problems, both in the single-machine and multi-machine levels [7-9].

Real ants are capable of finding the shortest path from a food source to their nest without using any visual cues but by exchanging the information of pheromone, a chemical substance, with each other. It is very simple and not even deterministic but effective as it is cooperative. Each ant deposits pheromone while marching and the other ants follow in probability the trail of pheromone. More pheromone is accumulated if more ants follow a certain path and on the other hand, the trail intensity decreases as time passes due to evaporation. All this behavior is simulated with a little difference in ACO algorithm.

In this algorithm, Eq. (1) which is the combination of two distinctive operators, formulate the probability of moving to the next step in an evolutionary structure:

$$p_{ij} = \frac{[\tau(i,j)].[\eta(i,j)]^{\beta}}{\sum_{j \in allowed} [\tau(i,j)].[\eta(i,j)]^{\beta}} \quad (1)$$

The first function $\tau(i,j)$ shows the amount of pheromone interpreted as sequences in JSP. In this notation, j is the job which is decided to be assigned to position i. The amount of pheromone is directly related to the path's fitness function or its goodness. This function acts as a positive feedback, determining the probability of selecting each unselected job in permutation. It has a dynamic behavior due to the simultaneous evaporation and deposition of pheromone on sequences. The first effect lets the unselected sequences have still chances to be evaluated and the second one intensifies the best solutions obtained toward the optimum.

In this paper we employ two ACO approaches to the single-machine scheduling problem. In the first approach, we present this function in the form named "Sequence-based Pheromone Deposition" (SPD) in which the pheromone deposition is sequence based. Assuming that the optimized sequence is unique, there is another pheromone deposition based on the sequence succession, in addition to the normal deposition based on the job location. It should be mentioned that in SPD, no special heuristic is applied for $\eta(i,j)$.

The second approach used here, involves a heuristic named "Apparent Tardiness Cost with Setups" (ATCS) rule introduced by Lee et al. [10]. This heuristic is of constructive type and can easily be embedded in ACO algorithm due to the same nature of step-by-step construction of solution. The ATCS function is based on three sub-rules including shortest processing time (WSPT), minimum slack (MS), and the shortest setup time (SST). It is presented as follows:

$$\eta(i,j) = I_j(t,v) = \frac{w_j}{p_j} \exp\left[-\frac{\max(d_j - p_j - t, 0)}{k_1 \bar{p}}\right] \exp\left[-\frac{s_{vj}}{k_2 \bar{s}}\right] \quad (2)$$

where t is the current time and v is the index of the job just completed. Also, $\bar{p}$ is the average processing time of all jobs and $\bar{s}$ is the average setup time. The two parameters k_1 and k_2 are used for tuning the function and the other ones are as defined in section 2.

It should be noted that in ACO, the probability of moving to the next step obtained by Eq. (1) is not directly and simply applied as it seems. The mechanism is known as "state transition rule" that in addition to two other rules namely "local updating" and "global updating" form the gist of ACO. These three rules are described in the following sections.

3.1 State Transition Rule

Unlike algorithms such as tabu search, genetic algorithm and simulated annealing where the coded solution candidate is built altogether and then evaluated, the ants construct the solution in a step-by-step procedure in ACO. It means each ant should decide where to go for its next step by selecting among all unvisited candidate elements. The mechanism used in ACO is a combination of directed greedy behavior and Rolette wheel known as state transition rule:

$$l_{k+1} = \begin{cases} \arg\max_{l_i \in L} \left\{ [\tau(l_k, l_i)][\eta(l_k, l_i)]^{\beta} \right\} & \text{if } q \le q_0 \\ P\left(\frac{[\tau(l_k, l_i)][\eta(l_k, l_i)]^{\beta}}{\sum_{l_i \in L} [\tau(l_k, l_i)][\eta(l_k, l_i)]^{\beta}}\right) & \text{if } q > q_0 \end{cases} \quad (3)$$

The parameter l_k is the latest chosen element and l_i belongs to the list L which includes all possible candidates for the next step.

The state transition rule has two sub-rules, while q and q_0 determine which one to be used. The constant parameter q_0 demonstrates the relative importance of sub-rules; however, q is a randomly-generated number uniformly distributed in domain $[0,1]$.

If there comes $q \le q_0$ which is the case of exploitation, the element with largest combination of pheromone and heuristic is chosen. Otherwise, the algorithm does not decide deterministically but only gives chances to the elements in proportion to their values as it is in a Rolette wheel which is just the case explained in Eq. 1. Thus, exploration of candidates with smaller function values is made feasible. In general, m ants are employed acting in a parallel manner. Their first elements of solution are assigned randomly and then to the end of constructing the solution, state transition rule is repeated.

3.2 Local Updating Rule

To avoid premature convergence, a local pheromone trail updating is performed on the value of pheromone related to the pair of elements just chosen by state transition rule:

$$\tau(l_k, l_{k-1}) = (1-\rho)\cdot\tau(l_k, l_{k-1}) + \rho\tau_0 \quad (4)$$

where τ_0 is the initial amount of pheromone at the beginning of the first iteration and is assigned as a constant value. The other parameter is ρ which is also constant during the solution and is chosen from domain $[0,1]$.

3.3 Global Updating Rule

In ACO, the globally best ant which is the ant that has constructed the best solution from the beginning of the trial is allowed to deposit pheromone on its trail. This rule which acts as positive feedback makes the search for the real best solution more directed. The rule is given by:

$$\tau(l_k, l_{k-1}) = (1-\alpha)\cdot\tau(l_k, l_{k-1}) + \alpha\cdot\Delta\tau \quad (5)$$

$$\Delta\tau = \frac{1}{WT_{gb}} \quad (6)$$

where $0<\alpha<1$ in Eq. (5) is the pheromone decay parameter and WT_{gb} in Eq. (6), is the amount of objective function which is the weighted tardiness related to the best sequence.

4 Numerical Results

The two variants of ACO described above are employed to optimize three 8-, 20- and 50-job randomly generated problems. Ant algorithm is a probabilistic method that is highly sensitive to the parameter settings. Therefore, parameter tuning should be carefully applied. For the best possible efficiency of the algorithm, the following settings have been determined after several trials.

The algorithm employs 10 ants in each iteration to construct 10 different tours or solutions. The termination takes place after defined number of iterations which depends on the problem scale. In the state transition rule, $q_0 = 0.9$ and $\beta = 0.5$, which has proved to provide better convergence. In both local and global updating rules, α and ρ, the evaporation and decay factors, were both set to 0.1.

Since the initial amount of pheromone, τ_0, plays an important role in the quality of initial sequence and convergence towards the final best solution, it should be set with great care. After examination of different values for this parameter, the average quantity of $\tau_0 = 1/(nWT)$ which is proposed by Liao and Juan [8], produces satisfactory results. In this equation, n is the number of jobs, and WT is the approximate amount of weighted tardiness obtained by applying ATCS.

The convergence curve for the 50-job weighted tardiness problem is given in Figure 1. As illustrated in this Figure and Table 1, the algorithm converges to the best solution after more the 1300 iterations which is equivalent to 100 seconds of search time.

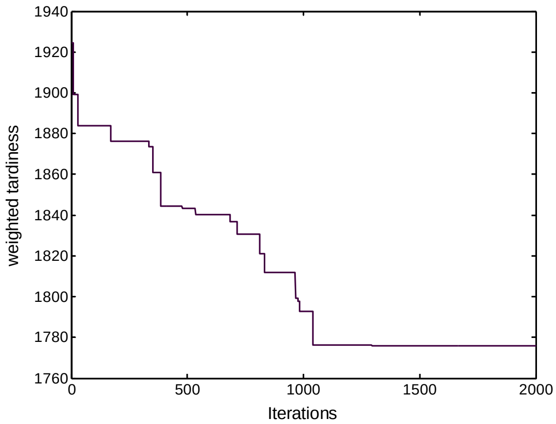

Fig.1. weighted tardiness of 50-job problem by ACO with ATCS

To evaluate the performance of ACO with probabilistic nature, the tabu search method with deterministic behavior is also applied to the same problems. A comprehensive description of TS method and its applications can be found in [11]. Among TS algorithm parameters, tabu list size and neighborhood generation mechanism are the most important ones. The tabu size is set to 20 moves to allow enhanced search of solutions space. For the neighborhood generation mechanism, pair wise interchange which is more suitable for large scale problems is employed in this paper [12]. Computational results of TS presented in Fig. 2 show no more improvement after about 85 iterations in just about 25 seconds.

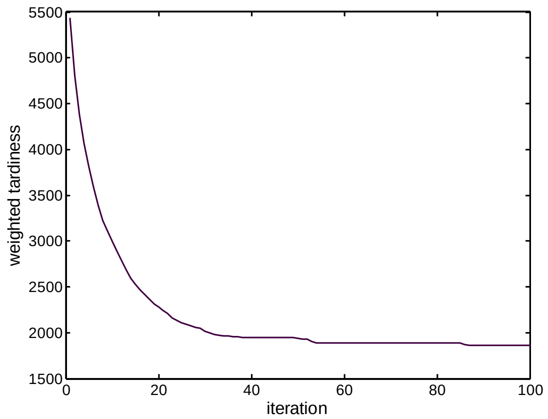

Fig.2. weighted tardiness of 50-job problem by TS

As indicated in Table 1, ACO and TS perform quite differently in terms of solution qualities and search times. The results clearly state that TS has a considerably lower convergence time. However, it stalls to improve in bigger problems and fails to approach the global optimum.

As it can be seen in Fig. 3, the ACO range of improvement is about 9% while the case is 65% for TS. This apparent improvement is mainly because of the inappropriate start point employed by the TS. In fact, the ACO creates its start point by applying the ATCS operator embedded in the state transition rule on the initial randomly generated sequence, while the TS start point is just the evaluation of that sequence.

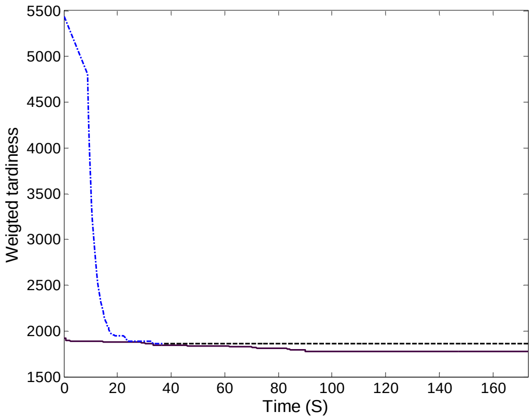

Fig.3. Time history of 50-job problem: The dash-dot and continuous lines show the TS and ACO with ATCS results, respectively. The dash line is the extension of TS line.

"Sequence-based pheromone deposition" method does not show satisfactory convergence which gives emphasis to the significance of heuristic operators in ACO. On the contrary, the ACO with ATCS heuristic outperforms the TS. In 8-job problem, due to the small size of sequences, both algorithms come to the global optimum. In 20- and 50-job problems the ACO results for weighted tardiness are 8% and 6% better than TS, respectively. However, the CPU time is lengthier due to the relative intricacy of operations needed to be done for each next-step decision.

5 Conclusions

An ant colony optimization was proposed to minimize the single-machine total weighted tardiness with sequence-dependent setups. To study the effect of problem size on the convergence and the quality of results, three randomly generated 8-, 20- and 50-job problems were investigated. Two heuristic functions, namely SPD and ATCS, were employed in ACO algorithm. The results obtained by this method are compared with tabu search method. It was shown that by using a proper heuristic rule and well-tuned parameters, ACO with its probabilistic nature outperforms TS with its deterministic behavior. However, the mechanisms employed for next-step evaluation in ACO are more time consuming in nature comparing to the ones employed by TS.

References:

[1] Kutub Uddina, M., Shankerb, K., Grouping of Parts and Machines in Presence of Alternative Process Routes by Genetic Algorithm, *Production Economics*, Vol. 76, 2002, 219-228.

[2] Bilge, U., Kıraç, F., Kurtulan, M., and Pekgün, P., A Tabu Search Algorithm for Parallel Machine Total Tardiness Problem, *Computers & Operations Research*, Vol. 31, 2004, pp. 397-414.

[3] VanLaarhoven, P.J.M., Aarts, E.H.L., and Lenstra, J.K., Job Shop Scheduling by Simulated Annealing, *Operations Research*, Vol. 40, 1992, pp. 113-125.

[4] Dorigo, M., and Gambardella, L. M., Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transactions on Evolutionary Computation*, Vol.1, No.1, 1997, pp. 53-66.

[5] Bell, J.E., and McMullen P.R., Ant Colony Optimization Technique for Vehicle Routing Problem, *Advanced Engineering Informatics*, Vol. 18, 2004, pp. 41-48.

[6] Gambardella, L.M., Taillard, D., and Dorigo, M., Ant Colonies for the Quadratic Assignment Problem, *Journal of Operational Research Society*, Vol. 50, 1999, pp.167–76.

[7] Boryczka, U., Ant Colony System for JSP, *ACRI 2004*, Springer-Verlag, 2004, pp. 296-305.

[8] Liao, C., Juan, H., An Ant Colony Optimization for Single-machine Tardiness Scheduling with Sequence-dependent Setups, *Computers and Operations Research*, August 2005.

[9] Bauer, A., Bullnheimer, B., Hartl R.F., and Strauss, C., An Ant Colony Optimization Approach for Single Machine Total Tardiness Problem. *Proceedings of 1999 Congress on Evolutionary Computation*, New York, pp. 1445-1450.

[10] Lee, Y.H., Bhaskaram, K., Pinedo, M., A Heuristic to Minimize the Total Weighted Tardiness with Sequence Dependent Setups, *IIE Transactions*, Vol. 29, 1997, pp. 45-52.

[11] Kovacevic-Vujcic, V.V., Cangalovic, M.M., Asic, M.D., Ivanovic, L., and Drazic, M., Tabu Search Methodology in Global Optimization,

Computers & Mathematics with Applications, Vol. 37, 1999, pp. 125-133.

[12] Wan, G., and Yen, B.P.-C., Tabu Search for Single Machine Scheduling with Distinct Due Windows and Weighted Earliness/Tardiness Penalties, *European Journal of Operational Research*, Vol. 142, Issue 2, 2002, pp. 271-281.

Table1. ACO and TS results for 8-, 20- and 50-job problems

Algorithms	**8-job problem**		**20-job problem**		**50-job problem**	
	Best solution obtained	**CPU time (s)**	**Best solution obtained**	**CPU time (s)**	**Best solution obtained**	**CPU time (s)**
ACO - SPD	**52.1**	**0.12**	**275**	**2.1**	**2015**	**95**
ACO - ATCS	**46.7**	**0.13**	**225.7**	**2.5**	**1775.9**	**100.2**
TS	**46.7**	**0.045**	**243.8**	**0.3**	**1883**	**24.1**