A Tabu Search Algorithm for Order Acceptance and Scheduling

Bahriye Cesaret, Ceyda Oğuz and F. Sibel Salman

Koç University, Turkey

bcesaret, coguz, ssalman@ku.edu.tr

Keywords: Order acceptance, scheduling, sequence dependent setup times, tabu search.

1 Order Acceptance and Scheduling Problem

Order acceptance and scheduling decisions arise when the limited production capacity entails a manufacturer to select among incoming orders. In this paper, we consider an order acceptance and scheduling problem in a single machine environment where the orders are processed without preemption and a sequence dependent setup time is incurred between processing of every consecutive order. The problem is defined formally as follows. A set of incoming orders is available at time zero. Each incoming order $i$ is identified with a release date, $r_i$, a processing time, $p_i$, a due date, $d_i$, a deadline, $\bar{d}_i$, the maximum revenue, $e_i$, the weight (tardiness penalty), $w_i$, and a sequence dependent setup time where $s_{ji}$ denotes the setup time when order $j$ immediately precedes order $i$ in the processing sequence.

In this problem setting, $e_i$ shows the maximum revenue that the manufacturer can gain if this order is accepted and the tardiness of the order is zero. However, if the order is tardy, the revenue that the manufacturer can gain from this order, $R_i$, decreases linearly with its tardiness, $T_i$. The manufacturer may complete order $i$ until its deadline $\bar{d}_i$, but for each time unit beyond its due date, a tardiness penalty cost of $w_i$ is incurred. No revenue can be gained from order $i$ if its completion time exceeds its deadline, $\bar{d}_i$, and the revenue of order $i$ equals to zero at its deadline. To ensure this, we set $w_i$ to $e_i/(\bar{d}_i-d_i)$ for each order $i$. Tardiness of order $i$ is $T_i = \max(0, C_i - d_i)$, where $C_i$ is the completion time of order $i$. Then revenue from order $i$ is $R_i = e_i I_i - w_i T_i$. Here, $I_i$ is an indicator that equals to 1 if order $i$ is accepted, and 0 otherwise. Our aim is to find an optimal set of accepted orders and an optimal schedule for them that maximizes the manufacturer's total revenue, expressed as $\sum_{i=1}^{n} R_i$. We refer to this problem as the Order Acceptance and Scheduling (OAS) problem.

A well-known special case of the OAS problem that does not address acceptance/rejection decisions, but only determines the scheduling of a given set of orders, is the single machine total tardiness problem with sequence dependent setup times, which is known to be strongly NP-hard (Lawler and Lenstra 1982). This implies that the OAS problem is also strongly NP-hard. ? provided a mixed integer linear programming (MILP) formulation for OAS problem and tested its limits on generated instances. Since exact solutions could be obtained for only instances with a limited number of orders, a simple constructive heuristic (m-ATCS) and an iterative improvement heuristic (ISFAN) which uses simulated annealing concepts to handle the sequencing were proposed. In this study, we develop a TS algorithm with a probabilistic local search after each iteration that performs better compared to both of the constructive and the improvement heuristics given by ?. We provide comparisons with the m-ATCS heuristic only, since the ISFAN algorithm is not competitive with m-ATCS in instances with 50 or higher number of orders. For instances with less than 50 orders, our proposed heuristic provides near optimal solutions in less than two seconds, hence a comparison with ISFAN for this group of instances is also excluded.
2 A Tabu Search Algorithm

In the TS algorithm, we represent a solution by a vector of size $n$, in which $i^{th}$ entry indicates the position of order $i$ in the sequence. If order $i$ is not accepted, the corresponding entry is set to zero. The initial solution is constructed by using a greedy rule. The rule requires the calculation of the following Revenue-load ratio for each order:

$$RLR_i = e_i / (p_i + s_{average,i}),$$

where $s_{average,i} = (s_{0,i} + s_{1,i} + ... + s_{n,i}) / (n + 1)$. We sort the orders with respect to the defined ratio, starting from the highest, to give priority to orders that potentially generate a higher revenue and take a small amount of time to process. After sorting the orders, the ones that generate zero revenue are rejected. We consider swapping two entries of the solution vector, i.e. pairwise exchanges, as the move operator. This move operator allows us to exchange the positions of accepted orders as well as changing the accepted set of orders, while keeping the number of accepted orders the same. The tabu list is formed with the $k$ most recently performed swaps, where $k$ is the tabu tenure. In our implementation, we set the tabu tenure to 3 for 10 orders, 5 for 15 orders and 7 for 20, 25 and 50 orders after preliminary tests. We also employed the usual aspiration criterion, that is, if the revenue of a tabu solution is better than the revenue of the best-known solution so far, this solution is accepted even though it is in the tabu list. We set the termination criteria as 50 iterations without an improvement of revenue.

Due to the move operator, a TS iteration may cause the number of accepted orders to decrease, which may prevent reaching the global optimum. To remedy this, after each TS iteration, we perform a local search starting from the best solution obtained in the neighborhood by applying drop-insert operations. To decide which order to drop, we use a modified version of the revenue-load ratio, in which we replace the average setup time with the current required setup since we know the predecessor of each order. We drop the order which has the minimum value of this ratio from the sequence since dropping it may allow us to insert an order that may have a larger revenue. To select the order to be inserted we use the original revenue-load ratio to generate a probability distribution for the rejected orders. This allows us to bring in some randomness in the algorithm, providing a diversification mechanism. Once the order to be inserted is selected randomly, the position to be inserted is decided by a comparison of its revenue when it is inserted to the current position with the revenue of the best-known solution. We further use an improvement threshold, that is set to 0.996 after preliminary tests, to generate another diversification in the search. We repeat this drop and insert procedure $n/5$ times based on our preliminary tests. After the local search, the algorithm returns back to the next TS iteration.

3 Computational Study

Data Generation: In order to test the performance of the TS algorithm, we generated new test instances in varying parameter values and problem sizes. In the new data set the tardiness factor $\tau$ and the due date range $R$ are used with five different values: $\tau = 0.1, 0.3, 0.5, 0.7, 0.9$ and $R = 0.1, 0.3, 0.5, 0.7, 0.9$ as given by Potts et al. (1984). For each possible combination of $\tau$ and $R$, 10 problem instances are generated. Hence, the number of test instances for each problem size, $n$, is $10 \times 25 = 250$, which sum up to 1250 instances for five different $n$ values; $n = 10, 15, 20, 25, 50$.

Computational platform: The TS algorithm was coded in Matlab and run on a Workstation with an Intel Xeon processor, 3.00 GHz speed, and 4GB of RAM. For MILP runs and LP Relaxation runs, we solved test instances with ILOG CPLEX 11.2 and set a time limit of 3600 seconds for all MILP runs.

Performance measures: For the problem sizes of 10, 15 and 20 orders, the best upper bound obtained within time limit of 3600 seconds in CPLEX ($UB_{CPLEX}$) is used
to measure the quality of the solutions. We report the best feasible solution found by the CPLEX solver as the MILP solution. The solutions obtained by the MILP, m-ATSC and the TS algorithm are compared in terms of the maximum, average and the minimum percentage deviation from $UB_{CPLEX}$. We also report the number of optimal solutions out of 10 instances of each type.

The upper bound obtained by relaxing the MILP model is referred to as $UB_{LP}$ and used while measuring the quality of the solutions for problem sizes of 25 and 50 orders. The solutions obtained by m-ATSC and the TS algorithms are compared in terms of the maximum, average and the minimum percentage deviation from the $UB_{LP}$ together with run time.

**Analysis of the Results:** The results of our computational studies suggest that the TS algorithm is both efficient and effective for this type of problem sizes and types tested. The detailed analysis is as follows.

*Comparison of the TS algorithm with MILP.* The results indicate that the TS algorithm is competitive with the MILP for $n=10$, and outperforms the MILP results when $n$ equals 15 and 20. The TS algorithm finds the optimal solution in 200 out of 250 instances for $n=10$ and gives 1% deviation on the average. This performance is achieved with a run time of less than a second in all instances, whereas MILP runs in 228 seconds on the average and reaches the limit of one hour in one of the instances. The TS algorithm finds the optimal solution in 39 out of 250 instances for $n=15$ and gives 4% deviation on the average. In contrast, the MILP finds the optimal solution in 102 out of 250 instances when $n=15$ but gives 6% deviation on the average. When $\tau$ equals 0.1, 0.3 and 0.5, representing the more difficult instances, MILP finds the optimal solution in only 3 out of 150 instances. The average run time of MILP increases to 2189 seconds while the TS algorithm runs in still less than a second on the average. When the problem size increases to 20 orders, the TS algorithm still outperforms MILP. The TS algorithm finds the optimal solution in 23 out of 250 instances and gives 6% deviation on the average. In contrast, the MILP finds the optimal solution in 57 out of 250 instances but gives 11% deviation on the average. The average run time of MILP increases to 2838 seconds while the TS algorithm runs in less than two seconds on the average. We observe that MILP has difficulty in solving instances with $\tau$ equals 0.7 in addition to 0.1, 0.3 and 0.5 when $n=20$.

*Comparison of the TS algorithm with m-ATCS.* We see that the TS algorithm outperforms the m-ATCS heuristic in terms of solution quality in all instances tested. Since the run time of the TS algorithm remains practical when $n=50$, we can conclude that the TS algorithm scales well and is a good alternative as a solution procedure. In the worst case, the m-ATCS heuristic has a 61% deviation, whereas the TS algorithm has 31%. The average performance of m-ATCS ranges from 14% to 25% with the change in problem size, while that of the TS algorithm ranges from 1% to 10%.

*Difficulty of test instances with respect to $\tau$ and $R$.* We observe that although the optimal solutions of MILP could not be found within time limit for $\tau=0.1, 0.3, 0.5$, problem instances for $\tau=0.7, 0.9$ could be solved optimally within few seconds. These results indicate that the problem gets fairly easier as $\tau$ increases. This can be explained as follows. As $\tau$ gets larger the slack times of the orders get smaller. Hence, it may be possible for the MILP solver to reject a considerable number of orders; the problem size reduces and the MILP solver can find the optimal solution more easily. On the other hand, when orders have larger slack times, the number of feasible sequences increases. As a result, the hardness of the problem increases when $\tau$ gets smaller.

*Analysis of the upper bounds.* The percentage deviations of both heuristics from the upper bound are larger in instances with larger $\tau$ and smaller $R$ values. When we analyze the number of rejected orders in these instances, we see that significantly more orders are rejected by the heuristics compared to other instances. Note that the LP relaxation solution
accepts most of the orders since it relaxes the capacity constraint. As a result, the upper bound gets weaker when the number of orders to be rejected is high due to the overload at certain time intervals.

In order to reinforce this observation, we solved the instances with $n = 25$ and $\tau = 0.9$ using MILP with a time limit of one hour and we kept the best upper bound found. The results show that for this case, the average percentage deviation of the TS algorithm drops to 3% from 10% which indicates that the upper bound is weak. This also shows that the TS algorithm performs well because the average percentage deviation of m-ATCS drops to 23% from 25% in comparison.

Comparison of the TS algorithm with m-ATCS in terms of the number of accepted and tardy orders. In order to understand the structure of the solutions generated by the two heuristics, we analyzed the number of rejected and tardy orders. We notice that m-ATCS rejects more orders and accepts more tardy orders than the TS algorithm over all instances. This can be attributed to the constructive nature of m-ATCS. The TS algorithm considers acceptance and scheduling decisions simultaneously, hence rejects the tardy orders if there are more profitable orders.

4 Conclusions

We provide a competitive improvement heuristic for the order acceptance and scheduling problem in a single machine environment. The proposed improvement heuristic is a tabu search (TS) algorithm that is supported with probabilistic local search after each iteration. We compare the performance of the TS algorithm to a greedy constructive heuristic m-ATCS, which is known to perform well in large instances. Our computational study shows that the TS algorithm gives significantly better solutions than the m-ATCS solutions in terms of revenue in all instances tested. Furthermore, the TS algorithm achieves this improvement with a small increase in run time.

The success of the TS algorithm may be attributed to the following factors. Firstly, the compact representation of the solution that combines the acceptance and sequencing decisions enables the generation of a rich neighborhood by a simple swap operator. The neighborhood spans solutions varying in both the set of accepted orders and their sequence. Secondly, in the algorithm each tabu search iteration is complemented with probabilistic local search operations that provide effective diversification mechanisms. As a result, the solution space can be searched extensively without heavy computational effort.

References

