Control and Cybernetics

vol. **25** (1996) No. 1

Simulated annealing applied to the total tardiness problem

by

Solomon R. Antony and **Christos Koulamas**

Department of Decision Sciences & Information Systems
College of Business, Florida International University
University Park, Miami, FL 33199, USA,
e-mail: koulamas@servms.fiu.edu

Abstract: In this paper the performance of Simulated Annealing (SA) as a heuristic on the well known total tardiness problem is tested. The experimental results indicate that the applicability of SA to the total tardiness problem is high. The authors found that the performance of the SA heuristic is not very sensitive to the starting solution and that slower cooling results in better solutions. Another interesting finding of this study is that a random generation of the neighborhood solutions results in better solutions more often than other methods.

1. Introduction

Simulated Annealing (SA) was first applied to combinatorial optimization problems by Kirkpatrick, Gelatt and Vecchi (1983). Many papers have subsequently reported successful applications. Van Laarhoven and Arts (1987) present a thorough treatment of the theory of SA and describe various applications of SA. In a recent survey Koulamas, Antony and Jaen (1994) present applications of SA to solve various Operations Research problems. It is clear from the above papers that the main use of SA is as a heuristic for combinatorial optimization problems. Necessary and sufficient conditions for SA to converge to a global optimum with probability 1 have been derived in Van Laarhoven and Aarts (1987). Experimental results indicate that this requires a very slow cooling scheme and a very large number of iterations; consequently this aspect of SA is primarily of theoretical interest.

The purpose of this paper is to experimentally test the performance of SA as a heuristic on a combinatorial optimization problem (the single machine total tardiness problem, $1//T$) which is known to be NP-hard. The $1//T$ problem can be stated as follows: There are n jobs to be processed without preemption

on a continuously available single machine which can handle only one job at a time. Job $J_i (i = 1, 2, \ldots, n)$ becomes available at time 0, requires a processing time p_i and has a due date d_i. The objective is to determine the processing sequence of the jobs, so that the total tardiness $\sum_{i=1}^{n} T_i = \sum_{i=1}^{n} \max(0, C_i - d_i)$ is minimized, where C_i is the completion time of job i. The above problem was selected for two reasons. First, one of the authors studied the problem in detail in a recent comprehensive survey (Koulamas, 1994) and proposed new heuristics for it. Second, there is a fast efficient polynomial heuristic for the $1//T$ problem (PSK heuristic by Panwalker, Smith and Koulamas 1993) which can be used to compare the performance of SA.

The main objective of the paper is to experimentally test the effect of various parameters on the performance of SA, namely

(i) initial solution (randomly generated versus provided by a fast heuristic)
(ii) selection of the neighborhood of the incumbent solution (switching two jobs in the sequence that were selected (a) randomly, (b) a randomly selected adjacent pair and (c) using complete enumeration)
(iii) cooling schemes (Given a temperature range, the number of distinct temperatures selected is 50, 100, 150 and 200 respectively)

It is anticipated that the experimental results will indicate the most effective combination of SA parameter which can lead to the appropriate decisions when computational time is viewed as a limited resource. In the next section we describe our implementation of the SA algorithm. In section 3 the experimental setup and the problem generation procedure are described. In section 4, the results of the experiment are discussed. In section 5, some concluding remarks are made.

2. SA algorithm

The Simulated Annealing algorithm was implemented using FORTRAN 77 on an Alpha series VAX mainframe computer. The pseudo-code of the algorithm is shown next. The objective is to find a solution $s \in S$, which minimizes $f(s)$. C is the number of temperatures to be used and I is the number of solutions to be tested at each temperature T.

Select an initial solution $s \in S$
Select an Initial (high) Temperature $T > 0$
Set temperature change counter $t = 1$
Repeat
    Set iteration counter $i = 1$
    Repeat
        Generate neighbor solution s_{new}
        Calculate $\delta = f(s_{new}) - f(s)$
        If $\delta < 0$ then $s = s_{new}$
        else if random$(0, 1) < e^{(-\delta/T)}$ then

$s = s_{new}$
$i = i + 1$
until $i = I$
$t = t + 1$
$T =$ Cooling function(T, t)
until $t = C$

At each temperature the algorithm attempts to find the minimum value for the objective function. It accepts worse solutions with a certain probability, which is computed as a function of the difference δ, and the current temperature T. The best solution attained so far is stored. As the temperature is lowered, the probability of accepting worse solution decreases and hence new minima are discovered.

3. Computational experiment

3.1. Problem generation

Individual test problems were generated as follows. For each job $i(1 \leq i \leq n)$, an integer processing time p_i is generated from the uniform distribution $(1, 100)$. Problem hardness is determined by two parameters: RDD, the relative range of due dates and TF, the tardiness factor. After $P = \sum_{i=1}^{n} p_i$ is computed, the RDD and TF values are selected from the $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ set. Then, for each job i, an integer due date d_i is generated from the uniform distribution $[P(1 - TF - RDD/2, P(1 - TF + RDD/2)]$. The values of RDD and TF provide a measure of problem hardness. Our preliminary experiments indicated that the hardest problems were generated when $TF = 0.6$ and $RDD = 0.6$. Consequently we conducted our main experiments using that TF, RDD value combination in order to sharpen the performance difference between the various alternatives. The number of jobs per problem is $n = 30$, 50 and 70 and fifty problem instances were generated for each n value.

3.2. Experimental design

We conducted three different experiments. Our objective is to determine how to utilize the computational time, when it is viewed as a limited resource. For example, should we invest in computing an initial solution by a heuristic or rather invest this computational time in a slower cooling scheme and use a randomly generated initial solution. In order to make these decisions, we evaluate the effect of three parameters on the performance of SA, namely the initial solution, the neighborhood selection method and the cooling scheme. This is accomplished by freezing two of them at predetermined levels and studying the effect of the various levels of the third one on the SA performance. It is anticipated that the performance of the SA algorithm is enhanced when the best level is se-

lected for each of the three parameters. However, the interdependencies among the three parameters are not explicitly considered.

The first experiment was to test the sensitivity of the SA algorithm to the initial solution, where we used (i) randomly generated initial solutions and (ii) solutions generated by a heuristic (PSK heuristic).

The second experiment was to compare the performance of different neighborhood generation schemes. A neighborhood solution is generated by exchanging the positions of two jobs in the sequence. The selection of the two jobs were done by (i) selecting two jobs randomly from the sequence, (ii) randomly selecting two consecutive jobs from the sequence and (iii) selecting all possible two job combinations from the sequence.

The third experiment was conducted to analyze the effect of different cooling schemes on the final solution. We adopted the system of performing a single iteration at each temperature. This has the advantage of reducing the number of parameters to be set. Intuitively, there is likely to be little difference between performing several iterations at the same temperature and performing these iterations at temperatures which do not vary significantly.

Our temperatures $T_1, T_2, \ldots, T_K$ where K is the total number of iterations (K sequences are generated and evaluated), followed the pattern $t_{k+1} = \frac{T_k}{1+\beta T_k}$ which is proposed by Lundy and Mees (1986). Using the results of preliminary experiments, we propose $T_1 = 100$ as the initial temperature and $T_K = 33$ as the final temperature.

The different cooling schemes we used in these experiments are characterized by the total number of iterations K (which equals the total number of temperatures since we perform a single iteration at each temperature). We used four cooling schemes with $K \in \{50, 100, 150, 200\}$ in our experiments, where a slower cooling scheme corresponds to a higher K value. Having fixed K, T_1, and T_K, the value of β is given by $\beta = \frac{T_1 - T_K}{(K-1)T_1 \cdot T_K}$.

For each of the above experiments, 50 problem instances were generated with $n = 30, 50, 70$ jobs. Each problem was solved by the PSK heuristic and various versions of the SA algorithm. The results of the experiments are presented next.

4. Results

The comparison of different methods is facilitated by a table which compares each method with all other methods. The square tabular representation enables us to compare the performance of a method on row i with the method in column j. For each comparison there are two square tables - one displaying the number of times method i outperformed method j (labeled as *Dominance*) and another displaying the number of times method i found the same solution as method j (labeled as *Equal.*)

4.1. Different initial solutions

The performances of the PSK heuristic (PSK), SA with random initial solution (SA_{Random}) and SA with PSK-given initial solution (SA_{PSK}) are compared in Table 1. In order to facilitate these comparisons, all SA experiments reported in table 1 utilize the same neighborhood generation scheme (random neighborhood generation) and the same cooling scheme (50 iterations). For problems involving 30 jobs, the SA_{Random} heuristic outperformed PSK 24 out of 50 times. The SA_{Random} heuristic outperformed the SA_{PSK} heuristic 11 times, whereas the SA_{PSK} heuristic outperformed the SA_{Random} only two times. On almost half the problem instances, all three heuristics yielded the same solutions. The performance of SA_{Random} and SA_{PSK} with respect to PSK, improves as the problem size increases. However, the relative improvement of SA_{PSK} over SA_{Random} increases as the problem size increases (SA_{PSK} outperformed SA_{Random} 14 times in the 50 jobs problems and 19 times in the 70 jobs problems). Hence, we have reason to believe that the use of a random initial solution is good enough for small problems; however for large problems the use of an initial solution supplied by a quick polynomial heuristic would result in better solutions. This, however, involves more computations. We may also note that the SA_{Random} heuristic does equally well or better than the PSK heuristic all the times.

4.2. Neighborhood selection

The three neighborhood selection methods are (i) random selection of two jobs (SA_{Random}), (ii) random selection of two adjacent jobs (SA_{Adj}) and (iii) consideration of all two-job combinations (SA_{Com}). In order to facilitate these comparisons and sharpen the effect of neighborhood selection methods on the final solutions, all SA experiments reported in Table 2 utilize the same initial solution (randomly generated) and the same cooling scheme (50 iterations). The results in Table 2 indicate that the performance of SA_{Random} is consistently better than the other heuristics (the zeros under the SA_{Random} column reveal this). As problem size increases (from 30 to 50 to 70 jobs) SA_{Random} increasingly outperforms SA_{Adj} (24, 35 and 41 times respectively) and SA_{Com} (21, 33 and 32 times respectively). When we compare SA_{Com} and SA_{Adj}, we find that SA_{Com} consistently outperformed SA_{Adj} (23 versus 1 in 30 job-problems, 35 versus 0 in 50-job problems and 41 versus 0 in 70-job problems). It can also be noted that the PSK heuristic did not do better than any other heuristic in any of the problems (zeros in the PSK rows reveal this). From the above findings, we can conclude that for the problem sizes considered, random neighborhood solution is more efficient in finding better solutions than other neighborhood selection methods. Although, intuitively we may expect SA_{Com} to do better than other two heuristics, we did not confirm it in our experiment. A possible reason for this is that the SA_{Com} heuristic selects the first 'better' solution it comes across as the next base solution, and proceeds afresh from that base solution.

30 Jobs

Dominance	PSK	SA_{Ran}	SA_{PSK}
PSK	0	0	0
SA_{Ran}	24	0	11
SA_{PSK}	24	2	0
Equal			
PSK	0	26	26
SA_{Ran}	26	0	37
SA_{PSK}	26	37	0

50 Jobs

Dominance	PSK	SA_{Ran}	SA_{PSK}
PSK	0	0	0
SA_{Ran}	35	0	14
SA_{PSK}	35	14	0
Equal			
PSK	0	15	15
SA_{Ran}	15	0	22
SA_{PSK}	15	22	0

70 Jobs

Dominance	PSK	SA_{Ran}	SA_{PSK}
PSK	0	0	0
SA_{Ran}	41	0	12
SA_{PSK}	41	19	0
Equal			
PSK	0	9	9
SA_{Ran}	9	0	19
SA_{PSK}	9	19	0

Table 1. Different initial solutions

30 Jobs

Dominance	PSK	SA_{Random}	SA_{Adj}	SA_{Com}
PSK	0	0	0	0
SA_{Random}	24	0	24	21
SA_{Adj}	1	0	0	1
SA_{Com}	24	0	23	0
Equal				
PSK	0	26	49	26
SA_{Random}	26	0	26	29
SA_{Adj}	49	26	0	26
SA_{Com}	26	29	26	0

50 Jobs

Dominance	PSK	SA_{Random}	SA_{Adj}	SA_{Com}
PSK	0	0	0	0
SA_{Random}	35	0	35	33
SA_{Adj}	0	0	0	0
SA_{Com}	35	1	35	0
Equal				
PSK	0	15	50	15
SA_{Random}	15	0	15	16
SA_{Adj}	50	15	0	15
SA_{Com}	15	16	15	0

70 Jobs

Dominance	PSK	SA_{Random}	SA_{Adj}	SA_{Com}
PSK	0	0	0	0
SA_{Random}	41	0	41	32
SA_{Adj}	0	0	0	0
SA_{Com}	41	3	41	0
Equal				
PSK	0	9	50	9
SA_{Random}	9	0	9	15
SA_{Adj}	50	9	0	9
SA_{Com}	9	15	9	0

Table 2. Neighborhood selection methods

Hence, it might have missed out on other solutions that are potentially better than the 'first better' solution.

4.3. Cooling schemes

The cooling rate was controlled by the number of iterations to be done between the starting temperature and the ending temperature. In order to facilitate these comparisons and sharpen the effect of cooling scheme on the final solutions, all SA experiments reported in Table 3 utilize the same initial solution method (randomly generated) and the same neighborhood selection method (randomly generated). We considered 50, 100, 150 and 200 iterations in this experiment according to the guidelines presented in Section 3.2. These methods are SA_{50}, SA_{100}, SA_{150} and SA_{200} respectively. The results in Table 3 indiacate that all four SA heuristics outperformed PSK in all the problems. The results confirm the well known principle that slower cooling improves the final solution. For the 30-job problems SA_{100} outperforms SA_{50} in 9 cases while SA_{150} outperforms SA_{100} in 12 cases and SA_{200} outperforms SA_{150} in 7 cases. In the 50-job problems SA_{100} outperforms SA_{50} in 30 cases, while SA_{150} outperforms SA_{100} in 19 cases and SA_{200} outperforms SA_{150} in 21 cases. In the 70-job problems SA_{100} outperforms SA_{50} in 17 cases, while SA_{150} outperforms SA_{100} in 29 cases and SA_{200} outperforms SA_{150} in 13 cases. This trend suggests that as the problem size increases, there is no corresponding improvement in the slower cooling algorithms. It may also be noted that the faster cooling schemes outperformed the slower ones in a number of cases. If we compare the performance a cooling scheme with the next slower scheme we find that their performance overlap to some extent. For example in the 30-job problems, SA_{50} outperforms SA_{100} in 9 cases and is outperformed by SA_{100} in 9 cases. The corresponding numbers for SA_{100} and SA_{150} are 3 and 12 respectively; for SA_{200} and SA_{150} are 6 and 7 respectively. However, this comparability of performance does not diminish as the problem size increases. For the 50-job problems, the numbers for SA_{50} and SA_{100} are 2 and 30 respectively; for SA_{100} and SA_{150} they are 12 and 19 respectively and for SA_{150} and SA_{200} they are 9 and 21 respectively. For the 70-job problems, the numbers for SA_{50} and SA_{100} are 18 and 17 respectively; for SA_{100} and SA_{150} they are 6 and 29 respectively and for SA_{150} and SA_{200} they are 12 and 13 respectively. Overall, slower cooling schemes improve the solution quality (as expected) but the improvement is not drastic.

5. Conclusions

Our findings with respect to the performance of SA on the single machine total tardiness problem can be summarized as follows: With respect to the initial solution, we found that for smaller problems, use of randomly selected initial solution yields good results, but for larger problems use of initial solutions provided by a quick heuristic yields better results than random initial solution.

30 Jobs

Dominance	PSK	SA_{050}	SA_{100}	SA_{150}	SA_{200}
PSK	0	0	0	0	0
SA_{050}	24	0	9	4	3
SA_{100}	24	9	0	3	1
SA_{150}	24	14	12	0	6
SA_{200}	24	16	14	7	0
Equal					
PSK	0	26	26	26	26
SA_{050}	26	0	32	32	31
SA_{100}	26	32	0	35	35
SA_{150}	26	32	35	0	37
SA_{200}	26	31	35	37	0

50 Jobs

Dominance	PSK	SA_{050}	SA_{100}	SA_{150}	SA_{200}
PSK	0	0	0	0	0
SA_{050}	37	0	2	5	3
SA_{100}	37	30	0	12	8
SA_{150}	37	29	19	0	9
SA_{200}	37	31	22	21	0
Equal					
PSK	0	13	13	13	13
SA_{050}	13	0	18	16	16
SA_{100}	13	18	0	19	20
SA_{150}	13	16	19	0	20
SA_{200}	13	16	20	20	0

70 Jobs

Dominance	PSK	SA_{050}	SA_{100}	SA_{150}	SA_{200}
PSK	0	0	0	0	0
SA_{050}	37	0	18	7	5
SA_{100}	37	17	0	6	5
SA_{150}	37	28	29	0	12
SA_{200}	37	31	22	13	0
Equal					
PSK	0	13	13	13	13
SA_{050}	13	0	15	15	14
SA_{100}	13	15	0	15	14
SA_{150}	13	15	15	0	16
SA_{200}	13	14	14	16	0

Table 3. Cooling schemes

With respect to the generation of neighborhood solutions, we found that random generation is superior to the other methods. With respect to the cooling schemes, we found that slower cooling results in better solutions more often, but the improvement is not drastic. The findings from this experimental study suggest that for single machine tardiness problems, SA heuristic is a strong alternative to other heuristics.

Acknowledgment: We would like to thank the anonymous referee for his useful comments which improved an earlier version of this paper.

References

KIRKPATRICK, S., GELATT, JR., C.D. and VECCHI, M.P. (1983) Optimization by simulated annealing. *Science*, **220**, 671-679.

KOULAMAS, C., ANTONY, S.R. and JAEN, R. (1994) A survey of simulated annealing applications to operations research problems. *Omega*, **22**, 41-56.

KOULAMAS, C. (1994) The total tardiness problem: Review and Extensions. *Operations Research*, **42**, 1025-1041.

LUNDY, M., MEES, A. (1986) Convergence of an annealing algorithm. *Mathematical Programming*, **34**, 111-124.

PANWALKER, S.S., SMITH, M.L. and KOULAMAS, C. (1993) A heuristic for the single machine tardiness problem. *European Journal of Operations Research*, **70**, 304-310.

VAN LAARHOVEN, P.J.M. and AARTS, E.H.L. (1987) *Simulated Annealing: Theory and Applications*, Reidel, Dordrecht.