Pareto Archived Simulated Annealing for Single Machine Job Shop Scheduling with Multiple Objectives

Samer Hanoun
Centre for Intelligent Systems Research
Deakin University
Geelong, Australia
samer.hanoun@deakin.edu.au

Saeid Nahavandi
Centre for Intelligent Systems Research
Deakin University
Geelong, Australia
saeid.nahavandi@deakin.edu.au

Hans Kull
Informatic Technologies
Geelong, Australia
h.kull@inmatic.com

***Abstract*—In this paper, the single machine job shop scheduling problem is studied with the objectives of minimizing the tardiness and the material cost of jobs. The simultaneous consideration of these objectives is the multi-criteria optimization problem under study. A metaheuristic procedure based on simulated annealing is proposed to find the approximate Pareto optimal (non-dominated) solutions. The two objectives are combined in one composite utility function based on the decision maker's interest in having a schedule with weighted combination. In view of the unknown nature of the weights for the defined objectives, a priori approach is applied to search for the non-dominated set of solutions based on the Pareto dominance. The obtained solutions set is presented to the decision maker to choose the best solution according to his preferences. The performance of the algorithm is evaluated in terms of the number of non-dominated schedules generated and the proximity of the obtained non-dominated front to the true Pareto front. Results show that the produced solutions do not differ significantly from the optimal solutions.**

Keywords-Multi-criteria optimization; Simulated annealing; Metaheuristic procedures; Pareto optimal; Job shop scheduling.

I. Introduction

Real industry problems require simultaneous optimization of several incomparable and conflicting criteria. Often, there is no single optimal solution; rather there is a set of alternative solutions. In joinery manufacturing, the decision maker aims at simultaneously minimizing the tardiness and the material cost for the produced jobs. Jobs with similar materials have a savings factor when scheduled together. On the other hand, the customer requires fast delivery once the order is confirmed. Therein lies a dilemma: scheduling jobs with similar materials would help control the material cost, but this would definitely increase the tardiness. Minimizing the tardiness will meet the customer's requirements, but does not generate higher revenue. A proper balance would minimize the material cost while simultaneously finishing all the jobs in a timely manner. In other words, a trade-off must be made between the material cost and a timely completion of all the jobs. Hence, in most real industry scheduling problems, we encounter the multi-objective optimization.

A general multi-objective optimization problem can be formulated in the following way. Given an n-dimensional solution space S of decision variables vector $X = \{x_1, \ldots, x_n\}$, it is required to find a vector X^* that satisfies a given set of criteria depending on K objective functions $Z(X) = \{Z_1(X), \ldots, Z_K(X)\}$. Finding the ideal vector X^* that minimizes all objective functions simultaneously is usually unfeasible. The solution space S is generally restricted by a series of constraints, such as $g_j(X^*) = b_j$ for $j = 1, \ldots, m$, and bounds on the decision variables. Objectives under consideration always conflict with each other, hence, optimizing vector X with respect to a single objective often results in unacceptable results with respect to the other objectives. Therefore, a perfect multi-objective solution that simultaneously optimizes each objective function is almost impossible. A reasonable solution to a multi-objective problem is to investigate a set of solutions, each of, which satisfies the objectives at an acceptable level, and without being dominated by any other solution. Marler and Arora [1] summarize the multi-objective optimization area within the following definitions:

1) *Dominant solution*: If all objective functions are used for minimization, a feasible solution X is said to dominate another feasible solution Y ($X \succ Y$), if $Z_i(X) \geq Z_i(Y)$ for $i = 1, \ldots, K$ and $Z_i(X) < Zi(Y)$ for at least one objective function.
2) *Pareto optimal (Efficient) solution*: A solution is said to be Pareto optimal if it is not dominated by any other solution in the solution space. A Pareto optimal solution cannot be improved with respect to any objective without worsening at least one of the other objectives.
3) *Pareto optimal set*: The set of all feasible non-dominated solutions in S is referred to as the Pareto optimal set. For many problems, the number of Pareto optimal solutions is enormous (perhaps infinite). Therefore, the problem of reducing Pareto optimal sets by obtaining the additional information is very important.
4) *Pareto front*: For a given Pareto optimal set, the corresponding objective function vector values in the objec-

 ISBN: 978-1-61208-139-7

tive space are called the Pareto front.

Scheduling problems are combinatorial optimization problems. In most cases, they are NP hard for even a single criterion optimization and are therefore unlikely to be solvable in polynomial time. The approaches are classified, Nagar et al. [2], into two groups: (1) finding the exact optimal solution using implicit enumeration methods based on either branch-and-bound or dynamic programming techniques; (2) finding a near optimal solution using heuristic methods. Heuristics are either constructive (e.g., Panneerselvam [3]) or improvement derived from metaheuristic approaches, such as genetic algorithm (GA) and simulated annealing (SA) (e.g., Sridhar and Rajendran [4], Suman [5]).

SA has become very popular for solving multicriterion optimization problems [6][7][8]. The increasing acceptance of this technique is due to its ability to: (1) find multiple solutions in a single run; (2) work without derivatives; (3) converge speedily to Pareto-optimal solutions with a high degree of accuracy; and (4) handle both continuous function and combinatorial optimization problems with ease. There have been a few techniques that incorporate the concept of Pareto-dominance. Some such methods are proposed in [9][10][11] and [12], which use Pareto-domination based acceptance criterion.

In this paper, the concept of Pareto-dominance is incorporated into the SA procedure to find the non-dominated set of solutions required by the decision maker. We start by briefly discussing the problem and the methodology of combining the objectives into a single weighted composite function in Section II. In Section III, we describe the Pareto archived simulated annealing (PASA) algorithm, and in Section IV, the computational study carried out to show the performance of the algorithm. Finally, we draw conclusions in Section V.

II. The Problem under Study

In the joinery domain, the cost of products, such as kitchens, is largely determined by the number of material sheets used in manufacturing. Jobs with similar materials can be scheduled together to decrease the amount of material waste. This leads to minimizing the production cost and therefore increase in the profit; however, not without affecting the tardiness of the jobs. The goal is to find the proper balance between these objectives.

The deterministic job shop scheduling problem considered, in this paper, consists of a finite set J of n jobs to be processed on a single machine. It is desired to find the order (schedule) in, which these n jobs should be processed to maximize the total cost savings C and minimize the total tardiness time T.

Every two jobs, j and k, with the same material have a savings factor S_{jk}, which shows the reduction in material that can be achieved when producing the two jobs in sequence ($S_{jk} = S_{kj}$). Given the number of material sheets N and the cost of a material sheet M, the cost savings CS_{jk} is calculated as:

$$CS_{jk} = CS_{kj} = M_j * (N_j + N_k) * S_{jk} \tag{1}$$

where $j = 1, \ldots, n, \quad k = 1, \ldots, n$

The total cost savings C is defined by:

$$C = \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} CS_{jk} \tag{2}$$

Each job is to be processed for an uninterrupted processing period of p_j. The process time p_j is assumed to be known in advance, and necessary setup times are included in the processing times. The tardiness T of job j is determined by the completion time c_j. It is calculated as:

$$T = \sum_{j=1}^{n} \max(0, c_j - d_j) \tag{3}$$

where d_j is the due date and c_j is the completion time of job j.

It is worth to note that in general, we may have to minimize all the objective functions, maximize them all, or minimize some functions and maximize others. However, any objective function can always be converted from the minimization form to the maximization form, and vice versa since:

$$\max(f(a)) = -\min(f(a)) \text{ and}$$
$$\min(f(a)) = -\max(f(a))$$

This conversion is applied to the total cost savings C objective to transforming it to a minimization objective.

An attractive approach adopted by several investigators [13][14][15][16] is to combine the objectives into a weighted sum:

$$E(x) = \sum_{i=1}^{K} w_i f_i(x) \tag{4}$$

The composite objective is used as the energy to be minimized in a scalar form. Therefore, the two objectives (1) the cost saving C and (2) the tardiness T are combined in one energy function as:

$$E = w * T - (1 - w) * C \tag{5}$$

where w ($0 \leq w \leq 1$) is the weight assigned to each objective during the search process.

It is clear that SA with a composite energy as in (4) and (5) will converge to points on the Pareto optimal front where the objectives have ratios given by w_i^{-1}, if such points exist. However, it is unclear how to choose the weights in advance. Recognizing this, w is initialized to 0 and is increased by 0.1 at each search process in order to realize various search directions to uncover more non-dominated solutions in the solution space.

The notations used throughout this paper are given below.

n: Number of jobs;
p_j: The processing time of job j;
d_j: The due date of job j;
σ: The current schedule;
T: Tardiness of the schedule σ;
C: Total cost savings of the schedule σ;
σ': The candidate schedule;

 ISBN: 978-1-61208-139-7

T': Tardiness of the schedule σ';
C': Total cost savings of the schedule σ';
σ_b: The best solution obtained during the search;
T_b: The best tardiness obtained during the search;
C_b: The best total cost savings obtained during the search;
w: The non-negative weight of the objectives;
Z: The weighted sum of the objectives for the schedule according to (5);

III. The Pareto Archived Simulated Annealing Algorithm (PASA)

SA is a metaheuristic algorithm based on the basic idea of neighborhoods. It was derived from the analogy between the simulation of the annealing of solid and the strategy of solving combinatorial optimization problems [17]. A neighboring solution is derived from its originator solution by a random move, which results a new slightly different solution. This increases the chance of finding an improved solution within a neighborhood more than in less correlated areas of the search space. Also, SA overcomes the problem of getting stuck in local minima, by allowing worse solutions (lesser quality) to be taken some of the time (i.e., allowing some uphill steps). The simplicity of the approach and its substantial reduction in computation time [18][19] has made it a valuable tool for solving multi-objective optimization problems [13][15][16].

In this section the main components of the PASA algorithm are presented. The implementation of the algorithm is described in Figure 1. To preserve the non-dominated solutions obtained during the search process, an archive is maintained for storage. The Pareto search and archiving procedure, as well as the procedures followed for setting the parameters are explained below.

A. Pareto Search and Archiving

The PASA algorithm starts its search with a randomly generated solution σ. This solution is added to the Pareto archive and the objectives T and C and the weighted sum, based on w, of the two objectives are calculated. A neighbour solution σ' is generated from the current solution σ using the Randomly Pairwise Interchange mechanism. The candidate solution σ' is then compared to σ for non-domination. In case of the two objectives T and C, a solution σ' is said to dominate a solution σ, if the following condition is satisfied:

$$[((T' \leq T) \quad AND \quad (C' \geq C)) \\ AND \quad ((T' < T) \quad OR \quad (C' > C))] \tag{6}$$

If the candidate solution σ' dominates σ, then σ' becomes the current solution. Otherwise, the dominated candidate solution is accepted with the acceptance probability P_{accept} as given in (7).

$$P_{accept} = \exp^{-(\triangle Z/T)}, \quad \triangle Z = Z' - Z \tag{7}$$

Whenever a candidate solution σ' is accepted, it is compared with every member of the archive. Once any solution in the archive is identified as a dominated solution, it is removed from the archive. If σ' is dominated by any existing solution, then it is discarded and comparison is terminated. After all comparisons, non-dominated solutions will be left in the archive and σ' is added to the archive, if those within the archive and σ' are not dominating each other. Irrespective of whether the candidate solution is added into the archive or not, the search process is continued with the current solution.

B. Parameter Settings

The value of the initial temperature is chosen by experimentation. The range of change ΔZ in the value of the objective function with different moves is determined. The initial value of temperature t_o is calculated based on the initial acceptance ratio λ_o, and the average increase in the objective function, ΔZ_0:

$$t_o = -\frac{\Delta Z_0}{\ln(\lambda_o)} \tag{8}$$

The following steps describe the method used to calculate the value of t_o. Non-improver solutions are accepted with a probability of about 95 percent in the primary iterations (i.e., $\lambda_o = 0.95$).

Step 1:
/* Q represents the number of samples */
for $q = 1$ to Q **do**
  repeat
    Generate two solutions X_1 and X_2 at random
  until $Z(X_1) \neq Z(X_2)$
  $t_o^q = -\frac{|Z(X_1)-Z(X_2)|}{\ln(0.95)}$
end for

Step 2:
$t_o = \frac{1}{Q}\sum_{q=1}^{Q} t_o^q$

Enough number of iterations at each temperature are carried out to ensure that all represented states are searched and to enable reaching the global optimum. For our problem, a 150 non-improving iterations are used to terminate the current temperature level. The temperature is decremented in a proportional manner using the relationship $t_{i+1} = \alpha * t_i$, where α is the cooling factor constant and chosen to be 0.98. A final temperature value t_f equals to 5 percent of the initial temperature t_o is used for stopping the algorithm (i.e., $t_f = 0.05 * t_o$).

The re-annealing procedure restarts the SA process with the best solution obtained during the previous run as the seed solution. The search direction is changed by changing the weight coefficient w to uncover more non-dominated solutions. Initially, w is set to 0 and is changed with increments of 0.1 for every search process. During, the re-annealing, the temperature and other parameters are re-set to their initial values. The re-annealing process is carried until w reaches the value 1.0.

 ISBN: 978-1-61208-139-7

Algorithm PASA
Calculate the initial temperature t_0.
Initialize the Archive.
Initialize $w = 0$, non-improving iterations at each temperature ($nt = 150$), cooling factor $\alpha = 0.98$ and final temperature $t_f = 0.05 * t_0$.
Generate a random solution (σ_{seed}), add σ_{seed} to the Archive, and let $\sigma_b = \sigma_{seed}$.
while ($w \leq 1.0$)
  $t = t_0$.
  $\sigma = \sigma_{seed}$.
  Calculate T, C, and Z.
  Let $T_b = T$, $C_b = C$, and $Z_b = Z$.
  while ($t \geq t_f$)
    k = 1
    while (k $\geq nt$)
      Generate a neighbour solution σ' from σ.
      Calculate T', C', and Z'.
      if (σ' dominates σ) OR (σ' and σ are non-dominating to each other)
        $\sigma = \sigma'$, $T = T'$, $C = C'$, and $Z = Z'$.
        Check dominance of σ' w.r.t all solutions in the Archive and update the Archive.
        if (σ' dominates σ_b)
          $\sigma_b = \sigma'$, $T_b = T$, $C_b = C$, and $Z_b = Z$.
        End if
      else
        Generate a random number U.
        if ($U < e^{-\Delta Z/T}$)
          $\sigma = \sigma'$, $T = T'$, $C = C'$, and $Z = Z'$.
          Check dominance of σ' w.r.t all solutions in the Archive and update the Archive.
        End if
      End if
      k = k + 1
    End while
    $t = \alpha * t$
  End while
  $w = w + 0.1$
  $\sigma_{seed} = \sigma_b$.
End while
Return the Archive containing the generated non-dominated solutions.

Fig. 1. The PASA algorithm

IV. Computational Results

In this section, effectiveness of the proposed algorithm in obtaining the Pareto front is measured by considering the extreme solutions, i.e., the best tardiness and the best total cost savings, of the Pareto optimal solution set as the reference. The performance is verified using a number of numerical examples, inspired by the real data and generated randomly with pre-defined parameters. The problem sets used for testing consist of 5, 6, 7, 8, 9 and 10 jobs. Processing times for jobs are generated based on the job size, while the due dates are generated with different levels of tightness as proposed in [20]. The total processing time $P = \sum_{i=1}^{n} p_i$ is computed first, then the due date for each job is generated from the uniform distribution:

$$[P(1 - TF - \frac{RDD}{2})), P(1 - TF + \frac{RDD}{2}))] \qquad (9)$$

where TF is the average tardiness factor and RDD is the range of due dates. The settings of $TF = 0.6$ and $RDD = 0.4$ are used.

The relative percentage deviation (RPD), defined by (10), in the objective value of the obtained non-dominated front with respect to the objective value of the extreme solution is used

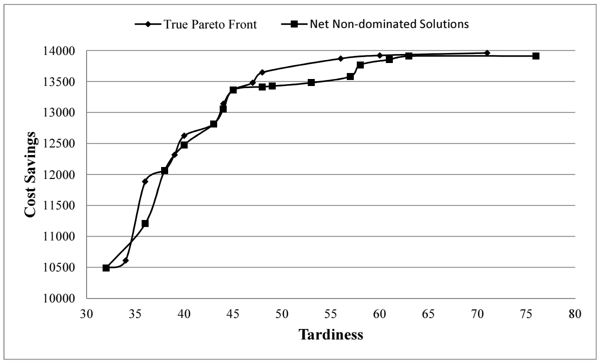

Fig. 2. True Pareto front and net non-dominated solutions for problem 22

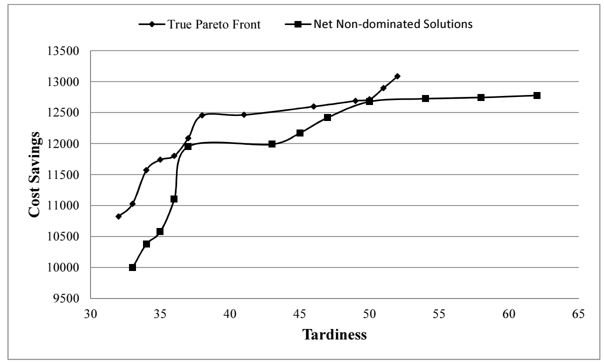

Fig. 3. True Pareto front and net non-dominated solutions for problem 27

as the main quality metric. Additionally, the mean relative percentage deviation (MRPD) is calculated for each problem set.

$$RPD = \frac{O_{obtained} - O_{extreme}}{O_{extreme}} * 100 \quad (10)$$

Table I shows the performance results of the algorithm for the generated problem sets. The true extreme solutions are obtained by enumerating all possible schedules to find the optimal values for T and C. The best values for T and C generated by the PASA are compared to the true extreme solutions. It is observed that the extreme solutions of the non-dominated front generated by PASA are very close to extreme solutions of the corresponding Pareto front. The non-dominated solutions generated are within 2.87% in T and 1.59% in C of the true extreme Pareto solutions on the average with a maximum deviation of 5.56% in T and 2.36% in C. Table II presents the net non-dominated solutions obtained for some problem instances. Figure 2 and Figure 3 show the net non-dominated front relative to the true Pareto front for sample of the problems (problem no. 22 and problem no. 27). Given the experimental results, the PASA produced very high quality solutions with low computational complexity based on the combinatorial nature of the problem.

V. Conclusion and Future Work

In this paper, a SA algorithm is presented to find Pareto solutions for the minimization of tardiness and the maximization of material cost savings for the single machine job shop scheduling problem in the joinery manufacturing domain. Different problem sets are solved with the proposed algorithm and the approximate Pareto optimal solutions are found. These solutions are compared with the true Pareto optimal front obtained by enumeration. Results show that the proposed method generates very close solutions to the optimal solutions for some problems and the true extreme solutions for other problems. Archiving the non-dominated solutions during the search process enables the decision maker to choose the best solution according to the conditions and constraints present at the time of decision making. For future work, more than two criteria will be considered as well as problems with dynamic and stochastic data.

Acknowledgment

This work is supported under the Australian Research Council (ARC) grant number LP0991175.

References

[1] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural Multidisciplinary Optimization*, vol. 26, pp. 369–395, 2004.

[2] A. Nagar, J. Haddock, and S. Heragu, "Multiple and bicriteria scheduling: A literature survey," *Eur J Oper Res*, vol. 81, p. 88104, 1995.

[3] R. Panneerselvam, "Simple heuristic to minimize total tardiness in a single machine scheduling problem," *Int J Adv Manuf Tech*, vol. 30, pp. 722–726, 2006.

[4] J. Sridhar and C. Rajendran, "Scheduling in flowshop and cellular manufacturing systems with multiple objectives a genetic algorithmic approach," *Prod Plan Control*, vol. 7, pp. 374–382, 1996.

[5] B. Suman, "Multiobjective simulated annealing a metaheuristic technique for multiobjective optimization of a constrained problem," *Found Comput Decis Soc*, vol. 27, pp. 171–191, 2002.

[6] R. W. Eglese, "Simulated annealing: A tool for operational research," *Eur J Oper Res*, vol. 46, pp. 271–281, 1990.

[7] P. Serafini, "Simulated annealing for multiobjective optimization problems," in *10th International Conference on Multiple Criteria Decision Making, Taipei Taiwan*, 1992, pp. 87–96.

[8] B. Suman and P. Kumar, "A survey of simulated annealing as a tool for single and multiobjective optimization," *J Oper Res Soc*, vol. 57, pp. 1143–1160, 2006.

[9] K. Smith, R. Everson, and J. Fieldsend, "Dominance measures for multi-objective simulated annealing," in *IEEE Congr Evol Comput*, 2004, pp. 23–30.

[10] R. K. Suresh and K. M. Mohanasundaram, "Pareto archived simulated annealing for job shop scheduling with multiple objectives," *Int J Adv Manuf Technol*, vol. 29, p. 184196, 2006.

[11] A. Haidine and R. Lehnert, "Multi-case multi-objective simulated annealing (mc-mosa): New approach to adapt simulated annealing to multi-objective optimization," *International Journal of Information and Communication Engineering*, vol. 4, no. 7, pp. 507–515, 2008.

[12] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: Amosa," *IEEE Tansactions on Evolutionary Computation*, vol. 12, pp. 269–238, 2008.

[13] P. Czyzak and A. Jaszkiewicz, "Pareto simulated annealing - a metaheuristic for multiple-objective combinatorial optimization," *Journal of Multi-Criteria Decision Analysis*, vol. 7, no. 1, pp. 34–37, 1998.

[14] A. Suppapitnarm, K. A. Seffen, G. T. Parks, and P. J. Clarkson, "A simulated annealing algorithm for multiobjective optimization," *Engineering Optimization*, vol. 33, pp. 59–85, 2000.

[15] P. Serafini, "Simulated annealing for multiobjective optimization problems," in *Multiple CriteriaDecision Making. Expand and Enrich the Domains of Thinking and Application*, G. H. Tzeng, H. F. Wang, V. P. Wen, and P. L. Yu, Eds. Springer-Verlag, Berlin, 1994, pp. 283–292.

TABLE I
PERFORMANCE OF THE PASA ALGORITHM, COMPARED TO THE OPTIMAL SOLUTION OBTAINED BY ENUMERATION, MEASURED IN TERMS OF THE RELATIVE PERCENTAGE DEVIATION (RPD) IN TARDINESS T AND TOTAL COST SAVINGS C

Problem no.	n	Optimal T	PASA T	RPD T	MRPD in T	Optimal C	PASA C	RPD C	MRPD in C
1	5	5	5	0.0		9523.04	9523.04	0.0	
2	5	4	4	0.0		4702.88	4702.88	0.0	
3	5	5	5	0.0	0.0	1850.37	1850.37	0.0	0.0
4	5	1	1	0.0		8106	8106	0.0	
5	5	2	2	0.0		8633.12	8633.12	0.0	
6	6	4	4	0.0		15691.5	15691.5	0.0	
7	6	15	15	0.0		5512.56	5512.56	0.0	
8	6	11	11	0.0	0.0	5687.64	5687.64	0.0	0.0
9	6	12	12	0.0		15420.24	15420.24	0.0	
10	6	13	13	0.0		14578.83	14578.83	0.0	
11	7	25	25	0.0		6203.12	6203.12	0.0	
12	7	18	18	0.0		10756.8	10756.8	0.0	
13	7	21	21	0.0	0.0	17698.68	17698.68	0.0	0.0
14	7	24	24	0.0		7284.69	7284.69	0.0	
15	7	22	22	0.0		18127.56	18127.56	0.0	
16	8	15	15	0.0		20439.54	20439.54	0.0	
17	8	18	18	0.0		23523.98	23523.98	0.0	
18	8	13	13	0.0	0.0	7512.15	7512.15	0.0	
19	8	49	49	0.0		10909.92	10909.92	0.0	
20	8	22	22	0.0		8081.01	8081.01	0.0	
21	9	33	33	0.0		28511.82	28511.82	0.0	
22	9	32	32	0.0		13960.54	13909.94	0.36	
23	9	36	36	0.0	0.0	24160.38	23958.06	0.84	0.24
24	9	25	25	0.0		23490	23490	0.0	
25	9	26	26	0.0		20104.14	20104.14	0.0	
26	10	62	62	0.0		17372.08	17330.28	0.24	
27	10	32	33	3.13		13083.84	12774.96	2.36	
28	10	33	34	3.03	2.87	10876.74	10681.86	1.79	1.59
28	10	36	38	5.56		5484.2	5365.2	2.17	
30	10	38	39	2.63		11345.43	11190.1	1.37	

TABLE II
THE NET NON-DOMINATED FRONT OBTAINED BY THE PASA ALGORITHM FOR INSTANCES OF THE PROBLEM SETS

	Prob. 1		Prob. 6		Prob. 11		Prob. 16		Prob. 22		Prob. 27	
	T	C	T	C	T	C	T	C	T	C	T	C
1	5	7979.68	4	13988.7	25	5430.04	15	16117.2	32	10489.38	33	9995.04
2	6	9336.48	5	14176.8	26	5436.2	16	17609.04	36	11202.84	34	10375.2
3	11	9523.04	7	14731.2	27	5855.08	18	18541.44	38	12068.1	35	10577.16
4			8	15691.5	28	5861.24	22	18714.6	40	12477.96	36	11099.88
5					29	6024.48	25	19174.14	43	12811.92	37	11951.28
6					32	6113.8	26	19467.18	44	13054.8	43	11986.92
7					42	6203.12	27	20439.54	45	13363.46	45	12169.08
8							34	20439.54	48	13414.06	47	12418.56
9									49	13424.18	50	12675.96
10									53	13479.84	54	12723.48
11									57	13581.04	58	12743.28
12									58	13768.26	62	12774.96
13									61	13854.28		
14									63	13909.94		
15									76	13909.94		

[16] E. L. Ulungu, J. Teghaem, P. Fortemps, and D. Tuyttens, "Mosa method: a tool for solving multiobjective combinatorial decision problems," *Journal of multi-criteria decision analysis*, vol. 8, pp. 221–236, 1999.

[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 20, pp. 671–680, 1983.

[18] S. Rajasekaran, "On the convergence time of simulated annealing," University of Pennsylvania Department of Computer and Information Science, Tech. Rep. MS-CIS-90-89, 1990.

[19] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statistical Science*, vol. 8, no. 1, pp. 10–15, 1993.

[20] C. N. Potts, "Single machine tardiness sequencing heuristics," *IIE Transactions*, vol. 23, no. 4, pp. 346–354., 1991.