Annals of Operations Research, 21 (1989) 85–108

A CONTROLLED SEARCH SIMULATED ANNEALING METHOD FOR THE SINGLE MACHINE WEIGHTED TARDINESS PROBLEM

Hirofumi MATSUO, Chang Juck SUH and Robert S. SULLIVAN

Department of Management, Graduate School of Business, The University of Texas at Austin, Austin, Texas 78712, U.S.A.

Abstract

In this paper, a new controlled search simulated annealing method is developed for addressing the single machine weighted tardiness problem. The proposed method is experimentally shown to solve optimally 99% of fifteen job problems with less than 0.2 CPU seconds, and to solve one hundred job problems as accurately as any existing methods, but with far less computational effort. This superior performance is achieved by using controlled search strategies that employ a good initial solution, a small neighborhood for local search, and acceptance probabilities of inferior solutions that are independent of the change in the objective function value.

1. Introduction

Simulated annealing is a randomized local search method that approximately solves optimization problems. Deterministic local search methods iteratively replace a seed solution by a superior one that might exist in the prespecified neighborhood of the seed solution. Otherwise, it terminates with a local optimum that may be far from a global optimum. Unlike deterministic local search methods, simulated annealing can replace a seed solution with an inferior one with some positive probability. Therefore, a global optimum may be attained, perhaps after a large number of iterations.

Simulated annealing was first developed as a simulation model to describe a physical annealing process of condensed matter (see Metropolis et al. [20]). Thereafter, it was applied to combinatorial problems, and was experimentally shown to provide near optimal solutions for some problem classes (Kirkpatrick et al. [16] and Cerny [7]). Many papers have subsequently reported successful applications of the technique in the areas of computer aided circuit design, image processing, code design and neural network theory. The reader is referred to van Laarhoven and Aarts [31] for a comprehensive discussion of the theory of simulated annealing, and for applications to a variety of problems.

The process of simulated annealing requires the specification of an acceptance probability. That is, the probability of replacing the seed solution with an inferior solution within a prespecified neighborhood. Let i denote the seed solution and j

denote some other solution within a prespecified neighborhood. Also, let C_i and C_j denote the respective objective function values of the solutions. For minimization problems, Metropolis et al. [20] used the acceptance probability

$$AP_{ij}(k) = \min\{1, \exp(-(C_j - C_i)/\beta_k)\},$$

where k is the stage of the search and $\beta_1 > \beta_2 > \ldots$ The stage is a level in which the same acceptance probability is used, and β_k is a control parameter. Note that for a given β the process of simulated annealing can be described as a Markov chain, where under certain conditions there exists a stationary (equilibrium) distribution (see van Laarhoven and Aarts [31]).

An annealing schedule specifies the number of iterations at each stage, the β values to use and the point at which to terminate the search. For the case where each stage includes one iteration, Gidas [13] and Hajek [15] derived necessary and sufficient conditions for the simulated annealing algorithm to converge with probability one to a global optimum. Several sufficient conditions for convergence also have been derived by Geman and Geman [12], Gelfand and Mitter [11], Anily and Federgruen [2,3], and Mitra et al. [21]. Even where convergence to an optimum with probability one is guaranteed after an infinite number of iterations, experimental results indicate that commonly used annealing schedules require a large number iterations to obtain a near optimal solution.

Considerable research has been directed at methods for accelerating the convergence of simulated annealing. In practical situations, it is not possible to attain equilibrium due to excess computational effort. Therefore, Aarts and van Laarhoven [1] introduce the concept of quasi-equilibrium, and they derive formula for determining β_k's which linearly reduce a distance between two successive stationary distributions. Greene and Supowit [14] show that nearly all moves are rejected for small β. They propose a procedure for avoiding rejections, and demonstrate that their procedure reduces computational effort for small β. Bohachevsky et al. [5] use simulated annealing for addressing nonlinear optimization problems. Here, they apply acceptance probabilies that depend on the current value of the objective function.

In this paper, a simulated annealing method is developed to address the single machine weighted tardiness problem. The proposed method, referred to as controlled search simulated annealing (CSSA), provides good approximate solutions with much less computational effort than with any other previously reported methods. This superior performance is achieved by using new controlled search strategies that employ a good initial (seed) solution, a small neighborhood for local search, and acceptance probabilities that are independent of the change in the objective function value. The approach taken here should be contrasted with conventional applications of simulated annealing where an initial solution is selected arbitrarily. This paper demonstrates that, if the quality of an initial solution is good, then the computation time to attain a near optimal solution is reduced considerably.

The single machine weighted tardiness problem has a structure much simpler than that found in most actual manufacturing environments. Despite its simple structure, this problem is unary NP-complete (Lawler [17] and Lenstra et al. [18]). Consequently, no computationally practicable exact solution methods have been developed, nor have there been any sophisticated heuristic methods that are efficient and provide good solutions. Improved methods for solving the single machine weighted tardiness problem can provide valuable insights for addressing more complex real-world manufacturing problems.

In this paper, CSSA is applied to the single machine weighted tardiness problem, and provides good solutions with practicable computation effort. CSSA is experimentally shown to solve optimally 99% of fifteen job problems with less than 0.2 seconds of CPU time. For one hundred job problems, it yields solutions as accurate as those given by the simulated annealing method of Aarts and van Laarhoven [1], but with far less computational effort. The search strategies used in CSSA considerably reduce the computational effort required for using simulated annealing. As a consequence, simulated annealing can use any good heuristic solution to the weighted tardiness problem as an initial seed, and improve upon it with a small increase in computation effort.

This paper is organized as follows. Section 2 provides a description of the single machine weighted tardiness problem, and reviews the state of the art in solution methodologies. In section 3, simulated annealing is described so as to provide a foundation for the proposed new search strategies used with CSSA. Section 4 explains the significance of the initial (seed) solution, while section 5 presents the results of comprehensive experiments that compare the quality of solutions and computational effort with those of the best methods reported in the literature. Section 6 summarizes the results and indicates the potential significance for more complex manufacturing environments.

2. The single machine weighted tardiness problem

The single machine weighted tardiness problem is characterized by having n jobs available for processing on a single machine, with each job i having an associated weight w_i and due date d_i, for $i = 1, 2, \ldots, n$. The objective is to find the nonpreemptive sequence of the n jobs on the single machine that minimizes the total weighted tardiness cost; that is the sequence that minimizes

$$\sum_{i=1}^{n} w_i \{\max(0, f_i - d_i)\},$$

where f_i is the scheduled completion time of job i for $i = 1, 2, \ldots, n$.

To optimally solve the weighted total tardiness problem, branch and bound algorithms using dominance relations and lower bounds have been developed by Elmaghraby [8], Emmons [9], Shwimer [30], Rinnooy Kan et al. [27], Fisher [10],

Picard and Queyranne [24] and Potts and Van Wassenhove [25,26]. Among these, Potts and Van Wassenhove's [26] algorithm is fastest. However, it sometimes fails to solve problems involving 30 jobs within 60 CPU seconds. Schrage and Baker [29] developed a dynamic programming algorithm that incorporates an efficient labeling scheme. This algorithm has a prohibitively large computer storage requirement for problems involving more than 20 jobs.

To avoid excessive computation time and storage requirements, heuristic dispatching rules have been extensively studied. However, there is no one best heuristic method for all problem environments. For example, it is well-known that if there is no more than one tardy job, then the earliest due data (EDD) sequence is optimal. Generally, EDD works well for a lightly loaded machine (Schild and Fredman [28]). But when all jobs are necessarily tardy, then the shortest weighted processing time (SWPT) sequence minimizes the weighted tardiness (see Baker and Martin [4]). Therefore, SWPT generally performs well in a heavily loaded shop. More sophisticated dispatching rules also have been developed by Carroll [6], Montagne [22] and Morton et al. [23]. Several of the best of these dispatching rules will be described in section 5, and will be used to develop initial seed solutions for simulated annealing. In addition, more complex deterministic local search heuristics have been studied by Wilkerson and Irwin [32] and Matsuo and Tang [19].

3. Simulated annealing and iterative improvements

In this section, we describe the proposed CSSA approach for using simulated annealing to address the single machine weighted tardiness problem. The following notation shall be used:

NOTATION

i = a sequence of jobs, which is a permutation of $\{1, 2, \ldots, n\}$
j = a sequence of jobs, which is a permutation of $\{1, 2, \ldots, n\}$
N = set of permutations $\{1, 2, \ldots, n\}$
N_i = neighborhood of sequence i
C_i = total weighted tardiness of sequence i,
ΔC_{ij} = $C_j - C_i$,
k = stage in simulated annealing that uses the same functional form of acceptance probability for $k = 1, 2, \ldots, K$.
M = number of iterations at each stage
$Ap_{ij}(k)$ = acceptance probability of new sequence j from sequence i at stage k.

Using the above notation, we can describe the simulated annealing process by Pidgin Algol:

THE SIMULATED ANNEALING PROCESS

```
begin
  i := an initial starting sequence generated by a heuristic
  for k = 1, 2,..., K do
    for m = 1, 2,..., M do
    begin
      j := a sequence selected from N_i (selection methods are discussed later);
      if ΔC_ij ≤ 0 then
        begin
          set i := j;
          if C_i is less than the current minimum total weighted tardiness then
          find a local optimum by a deterministic search from i and keep the
          best solution so far;
        end
      else
        begin
          a := a random number generated from the uniform distribution
          between 0 and 1;
          if a ≤ AP_ij(k) then i := j
        end
    end
end
```

In section 5, we shall discuss several heuristic rules to be used for deriving an initial seed solution used in the simulated annealing algorithm. Note that the algorithm checks for a local optimum whenever C_i becomes less than the current minimum weighted total tardiness, and that it always keeps the best solution found thus far. The purpose of this procedure is to attain a near optimal solution as rapidly as possible, as opposed to ensuring convergence to an optimal solution.

The following new features have been introduced into CSSA for addressing the single machine weighted tardiness problem.

Adjacent pairwise interchange

The neighborhood, N_i, is composed of the set of permutations of $\{1, 2, \ldots, n\}$ that can be obtained by interchanging a pair of adjacent jobs. The cardinality of N_i is $n-1$. The worst case and empirical performance of deterministic local search methods using several other neighborhoods with $O(N^2)$ elements is discussed in Matsuo and Tang [19]. The choice of the smaller neighborhood is related to the acceptance probability function applied.

Acceptance probability

We use an acceptance probability that does not depend on the change, ΔC_{ij}; that is, $AP_{ij}(k) = AP(k)$ for any i and j with $\Delta C_{ij} > 0$. The benefit of using an

acceptance probability that does depend on ΔC_{ij} by contrast with the approach taken here is that it avoids a large change in solution structure in the latter stages of the simulated annealing process. Note that the exponential form does depend on ΔC_{ij}. Since we use a relatively small neighborhood and a good initial solution, a constant acceptance probability for each stage is expected to perform as well as the exponential form. The benefit of using constant acceptance probabilities is that it facilitates designing experiments to empirically identify the best annealing schedule. Here, each stage is explicitly characterized by the probability of accepting an inferior solution. The range and the form of reducing acceptance probabilities shall be discussed in section 5.

Neighborhood search

In most published research on simulated annealing, a solution is randomly selected from the neighborhood of a seed solution. In this paper, we search a neighborhood sequentially. For example, assume that (1, 2, 3, 4, 5) is a seed solution and that (2, 1, 3, 4, 5) and (1, 3, 2, 4, 5) are rejected at an iteration. Also, assume that (1, 2, 4, 3, 5) is the first solution found with a smaller objective function value. Then, (1, 2, 4, 3, 5) becomes the new seed with neighborhood consisting of (2, 1, 4, 3, 5), (1, 4, 2, 3, 5), (1, 2, 3, 4, 5) and (1, 2, 4, 5, 3). Here, note that the tardiness value associated with (1, 2, 3, 4, 5) is less than or equal to that of (2, 1, 3, 4, 5). This implies that the tardiness of (1, 2, 4, 3, 5) is less than or equal to that of (2, 1, 4, 3, 5). Consequently, we search the neighborhood of (1, 2, 4, 3, 5) in the order of (1, 4, 2, 3, 5), (1, 2, 3, 4, 5), (1, 2, 4, 5, 3) and (2, 1, 4, 3, 5) so as to increase the chance of improvement.

4. Initial acceptance probability

Simulated annealing tends to yield a near optimal solution independent of the initial seed solution. Initially, simulated annealing accepts inferior solutions with high acceptance probabilities, and thereafter the acceptance probability of inferior solutions is gradually reduced until it becomes zero. For example, consider using the Metropolis acceptance probability function. Then, the acceptance probability AP_{ij} is close to 1 for any i and j when β is large, and it is close to 0 when β is small. Therefore, the method generally takes extensive computational effort when the initial β value is set high.

Assume that $|N_i|$ is a constant G for any solution i and that it is possible to reach any solution from any other solution in some finite number of steps. Then, if we use the Metropolis acceptance probability function with a given β, a Markov process results with transition probabilities:

$$P_{ij}(\beta) = \begin{cases} G^{-1} \max\{0, \exp(-\Delta C_{ij}/\beta)\} & \text{for } j \in N_i \\ 1 - \sum_{k \in N_i} G^{-1} \max\{0, \exp(-\Delta C_{ij}/\beta)\} & \text{for } j = i \\ 0 & \text{otherwise} \end{cases}$$

Let $\pi_i(\beta)$ denote the stationary probability of being in solution i in the Markov process defined by β. Then, $\pi_i(\beta)$ can be expressed as follows:

$$\pi_i(\beta) = \exp(-(C(i) - C(i_*))/\beta) / \sum_{i \in N} \exp(-(C(i) - C(i_*))/\beta),$$

where $C(i_*)$ is the global optimal value, and N denotes the set of all feasible solutions.

When the system reaches equilibrium state perhaps after many iterations, then for a given β the expected value of the objective function is defined as

$$\mathrm{E}[C_\beta] = \sum_{i \in N} \pi_i(\beta) C(i).$$

This $\mathrm{E}[C_\beta]$ can be approximated by

$$\bar{C}(\beta) = \left(\sum_{k=1}^{M_\beta} C(y_k) \right) / M_\beta,$$

where M_β denotes the number of iterations at stage β and y_k is the solution of

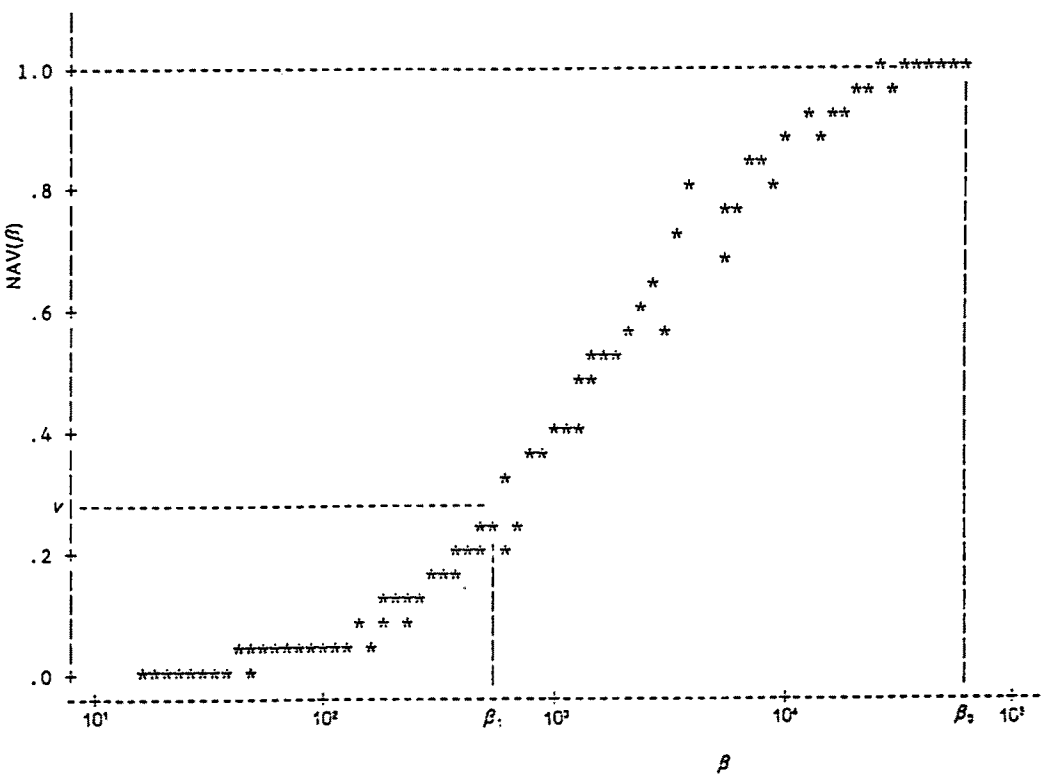

Fig. 1. Normalized average value of the objective function as a function of control parameter β for an example 15-job total weighted tardiness problem with $r = 0.6$ and $R = 0.4$.

the kth iteration at stage β. Consider the normalized average value of the objective function (NAV(β)) given by

$$\mathrm{NAV}(\beta) = \left(\bar{C}(\beta) - C_{\mathrm{opt}}\right)/\left(\bar{C}_{\infty} - C_{\mathrm{opt}}\right),$$

where C_{opt} denotes a global optimal value of the objective function, and $\bar{C}_{\infty}$ denotes the average value of the objective function for $\beta = \infty$. Figure 1 shows the NAV(β) associated with various values of β for an example 15-job total weighted tardiness problem when an annealing schedule is applied. The curve describes a typical transition of the average objective function values in response to the decrease in β, and shows that simulated annealing yields a near optimal solution for this particular problem.

In the scheduling literature, several simple heuristic procedures have been developed for addressing the total weighted tardiness problem. Assume that a heuristic procedure almost always provides a solution satisfying $(v(j) - C_{\mathrm{opt}})/(\bar{C}_{\infty} - C_{\mathrm{opt}}) \leqslant v$, where $v(j)$ is the objective function value of heuristic solution. Let β_1 denote the control parameter corresponding to v in fig. 1. If simulated annealing is initiated with the heuristic seed solution and the control parameter β_0, then a near optimal solution likely will be obtained by following the entire path depicted in fig. 1. However, we expect that there exists an initial control parameter β between β_1 and β_0 for which simulated annealing will be able to find a near optimal solution from the initial heuristic solution. If this β value is very small as compared with β_0, then using a heuristic solution as an initial point along with the small β value significantly reduces the computational time. In the next section, we shall empirically derive the appropriate initial parameter setting for each of several heuristic seed solutions, and show that the proposed approach leads to fast convergence of CSSA to a near optimal solution.

5. Computational results

5.1. EXPERIMENTAL DESIGN

CSSA was applied to test problems involving 15, 50 and 100 jobs, respectively. An experiment was designed to help determine the appropriate parameter settings, and to determine the performance of the method as compared with other simulated annealing methods reported in the literature. The test problems were generated as follows:

(i) Processing times: The processing time p_j for job j is generated from the normal distribution such that $p_j \sim N(100, 25)$. This confirms to the experimental study in Baker and Martin [4].

(ii) Due dates: The generation of due dates is controlled by the due date range, R, and the tardiness factor, r. The due date d_j for job j is drawn from a uniform distribution on

$$\left[\sum_{j=1}^{n} p_j\left(1-r-\frac{R}{2}\right), \sum_{j=1}^{n} p_j\left(1-r+\frac{R}{2}\right)\right].$$

Note that its range is

$$\left(\sum_{j=1}^{n} p_j\right)R$$

and the mean is

$$\left(\sum_{j=1}^{n} p_j\right)(1-r).$$

In this experiment, we set R equal to 0.4 and 0.8, and r equal to 0.2, 0.4, 0.6 and 0.8.

(iii) Tardiness weights: Tardiness weight w_j for job j is generated from the discrete uniform distribution on [1,10].

For each combination of parameters, twenty problems are randomly generated, and four measures of performance are used. These are:

(i) Average Relative Error (ARE): This is the average deviation of the heuristic value from the optimal value divided by the optimal value. That is, for the twenty test problems

$$\text{ARE} = \text{average of } (\text{T(SA)} - \text{T(OPT)})/\text{T(OPT)},$$

where T(SA) and T(OPT) denote the total weighted tardiness of simulated annealing and the optimal total weighted tardiness, respectively.

(ii) PSO: This is the percentage of problems solved optimally from the 20 problems.

(iii) Average Relative Improvement (ARI): For problems with 50 jobs and 100 jobs, no computationally practical algorithms have been developed for determining optimal solutions. Furthermore, a lower bound developed by Potts and Van Wassenhove [26] is not good enough to replace the T(OPT) value in ARE. Therefore, the relative improvement over the AU heuristic, which shall be introduced in section 5.2, is measured. That is,

$$\text{ARI} = \text{average of } \text{T(SA)}/\text{T(AU)},$$

where T(AU) is the total weighted tardiness of the AU heuristic.

(iv) Normalized Absolute Improvement (NAI): This is used to guage the performance of 50 and 100 job problems. The normalized weighted tardiness was used in Morton et al. [23] and Ow (1984) to compare results with different parameter values. For the conventional tardiness problem, NAI is obtained by dividing the

improvement of the absolute tardiness by the number of jobs, the mean processing time and the mean weight. NAI is defined as follows:

$$\text{NAI} = \text{average of } (T(\text{heu}) - T(\text{SA}))/\left(n\left(\sum_{j=1}^{n} p_j/n\right)\left(\sum_{j=1}^{n} w_j/n\right)\right),$$

where T(heu) is the objective value of an initial seed solution.

Altogether, twenty four combinations of parameter settings are used (i.e. $2(\text{R}) \times 4(r) \times 3(\text{n}) = 24$ classes). For each combination twenty problems are randomly generated and solved, and the averages of the performance measures are calculated. Computation time is measured by the average CPU seconds. The algorithms are coded in FORTRAN, compiled using the MNF compiler, and run on the CDC Dual Cyber 170/750.

5.2. INITIAL SEQUENCE

In section 4, we conjectured that CSSA, using a good initial seed solution, can start with a low initial acceptance probability, thereby reducing computation effort. To generate different initial seed solutions, we use four dispatching rules. These are:

(a) SWPT (Shortest Weighted Processing Time): sequence the jobs in decreasing order of w_j/p_j.

(b) EDD (Earliest Due Date): sequence the jobs in increasing order of d_j.

(c) COVERT (Cost Over Time): the COVERT rule (Carroll 1965) adapted from Fisher [10] determines a priority index for each unscheduled job as follows:

$$\pi_i = \begin{cases} w_i/p_i & \text{for} \quad d_i \leqslant p_i + t \\ w_i(T - d_i)/(p_i(T - t - p_i)) & \text{for} \quad t + p_i < d_i < T \\ 0 & \text{for} \quad T \leqslant d_i, \end{cases}$$

where

$$T = \sum_{i=1}^{n} p_i.$$

The COVERT rule schedules the next job with the largest priority index among the unscheduled jobs.

(d) AU (Apparent Urgency Rule): the AU heuristic (Morton et al. [23]) assigns the next job with the highest apparent urgency priority. This priority is determined by

$$\pi_i = (w_i/p_i)\exp\left[-(d_i - t - p_i)^+/k\bar{p}\right],$$

where π_i denotes the priority index for job i, t the start time for job i, k a look-ahead parameter, $\bar{p}$ the average processing time which is equal to

$$\sum_{i=1}^{n} p_i/n,$$

Table 1
Results of initial heuristic solutions

w	R	r	ARE		NAI	ARE		NAI
			SWPT	CSSA	SWPT	AU	CSSA	AU
1–10	0.4	0.2	22.2562 (0)	0.0574 (18)	0.1415	0.6139 (12)	0.0018 (19)	0.0071
		0.4	1.0955 (0)	0.0196 (13)	0.3141	0.1456 (6)	0.0079 (17)	0.0511
		0.6	0.4244 (0)	0.0027 (18)	0.4557	0.0469 (3)	0.0007 (17)	0.0501
		0.8	0.1160 (0)	0.0000 (20)	0.3251	0.0139 (4)	0.0000 (20)	0.0397
	0.8	0.2	2535.0839 (0)	0.0000 (20)	0.3239	0.0000 (20)	0.0000 (20)	0.0000
		0.4	25.6462 (0)	0.0393 (18)	0.8010	0.7190 (4)	0.0336 (18)	0.0330
		0.6	0.9995 (0)	0.0012 (14)	0.9617	0.0945 (1)	0.0005 (18)	0.0806
		0.8	0.1889 (0)	0.0000 (19)	0.5653	0.0206 (5)	0.0000 (20)	0.0581
PSO (%)			0.00	87.50		34.38	93.13	
Time (sec.)			0.001	0.092		0.003	0.081	

( ): the number of problems solved optimally.

and $(x)^+ = \max(0, x)$. Our preliminary investigations show that, for 15 job problems, $k = 0.5$, 0.9, 2.0 and 2.0 provide good performance for $r = 0.2$, 0.4, 0.6 and 0.8, respectively, and that, for 50 and 100 job problems, $k = 0.5$, 1.8, 4.0 and 2.0 provide better performance for $r = 0.2$, 0.4, 0.6 and 0.8, respectively.

The initial acceptance probability is set at 0.5, and is decreased linearly and discretely at the rate 0.02. The number of the searches at each stage is set to 120. Table 1 presents the results of controlled search simulated annealing when the initial seed sequence was developed using the two heuristic methods, SWPT and AU. The performance of both EDD and COVERT is superior to SWPT and inferior to AU. Therefore, their results are not shown in table 1. As seen in table 1, CSSA significantly improves each heuristic seed solution. The AU heuristic appears to provide the better seed sequence, and the ARE after using CSSA is less than 1% for nearly every combination of parameters. The only exception is an ARE of 3.36% for $r = 0.4$ and R = 0.8. Note that the ARE for solutions derived using the AU method alone average 20.68%. When using the AU solutions as the initial seed solutions for CSSA, the average ARE is reduced to 0.56%, or to less than 3% of the original average ARE given by the AU method. This dramatic improvement in accuracy is achieved with an average of 0.081 CPU seconds. This is an increase in computation time of approximately 0.078 CPU seconds over using the AU method alone. Therefore, we conclude that CSSA yields sequences very close to the optimal with only a modest increase in computation effort. This conclusion is also supported by the fact that 87.5%, 92.5%, 90.6%, and 93.1% of 160 15-job problems are solved optimally when CSSA uses SWPT, EDD, COVERT, and AU rule as a seed sequence, respectively.

Figure 2 shows the accuracy associated with various initial acceptance probabilities when the results obtained by the AU rule are used as the initial solution.

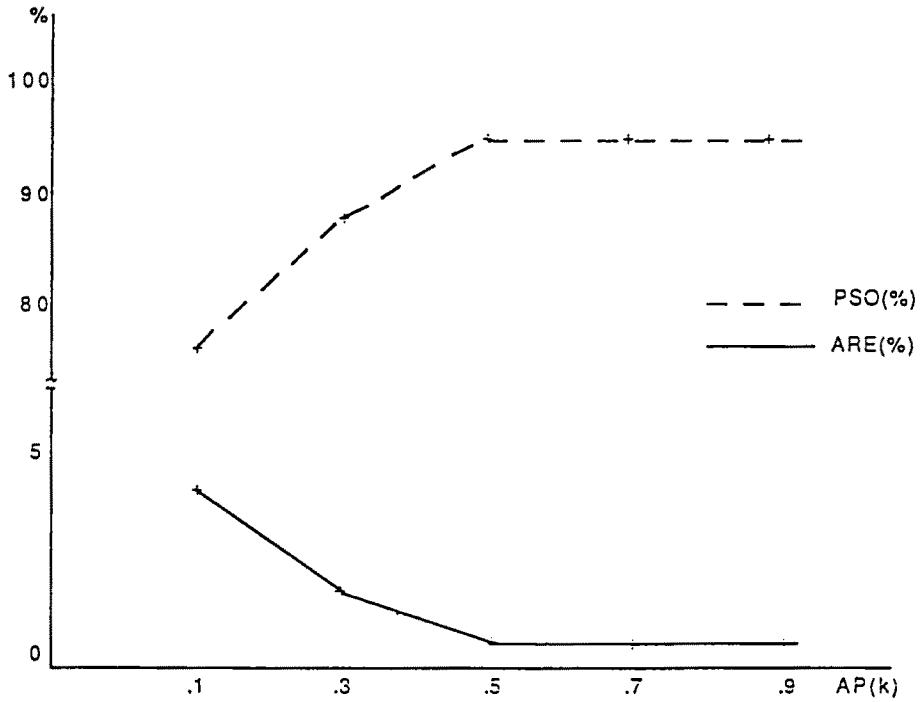

Fig. 2. The performance of CSSA associated with various range of initial acceptance probabilities.

The accuracy of the final solutions are of the same magnitude for any initial acceptance probabilities greater than or equal to 0.5. However, when the initial acceptance probability is lowered below 0.5, the accuracy deteriorates signifi-

Table 2
CSSA using SWPT solution as an initial seed
ARE, PSO(%) and computational time

w	R	r	$AP(k)$	
			0.6–0.0 *	0.5–0.0
1–10	0.4	0.2	0.0000 (20)	0.0574 (18)
		0.4	0.0056 (15)	0.0196 (13)
		0.6	0.0022 (19)	0.0027 (18)
		0.8	0.0007 (18)	0.0000 (20)
	0.8	0.2	0.0000 (20)	0.0000 (20)
		0.4	0.0005 (19)	0.0393 (18)
		0.6	0.0004 (18)	0.0012 (14)
		0.8	0.0000 (20)	0.0000 (19)
PSO (%)			93.13	87.50
TIME (sec.)			0.111	0.092

*: range of acceptance probabilities
( ): the number of problems solved optimally

Table 3
Performance of CSSA

Range of $AP(k)$	0.9–0.0			0.5–0.0
Initial seed	ARB		AU	AU
No. of stages	120	240	120	120
PSO (%)	90.00	95.00	93.13	93.13
Time (sec.)	0.147	0.296	0.142	0.081

ARB: an arbitrary initial seed solution.

cantly. This supports the contention that if a good initial seed is given, CSSA can start with low initial acceptance probabilities. As shown in table 1, SWPT yields an initial solution inferior to the other methods, and consequently the final solution obtained by CSSA using SWPT also is inferior. Table 2 shows that an increase in the initial acceptance probability from 0.5 to 0.6 improves the final solution using SWPT. In this case, its accuracy improves up to that obtained by CSSA using the AU rule with the initial acceptance probability equal to 0.5. Clearly, a better initial solution requires a lower initial acceptance probability, and consequently less computation time is required for convergence to a good solution. Note that, although the AU solution is reasonably good, it sometimes requires rather extensive change to attain an optimum. This is realized by maintaining relatively high acceptance probabilities.

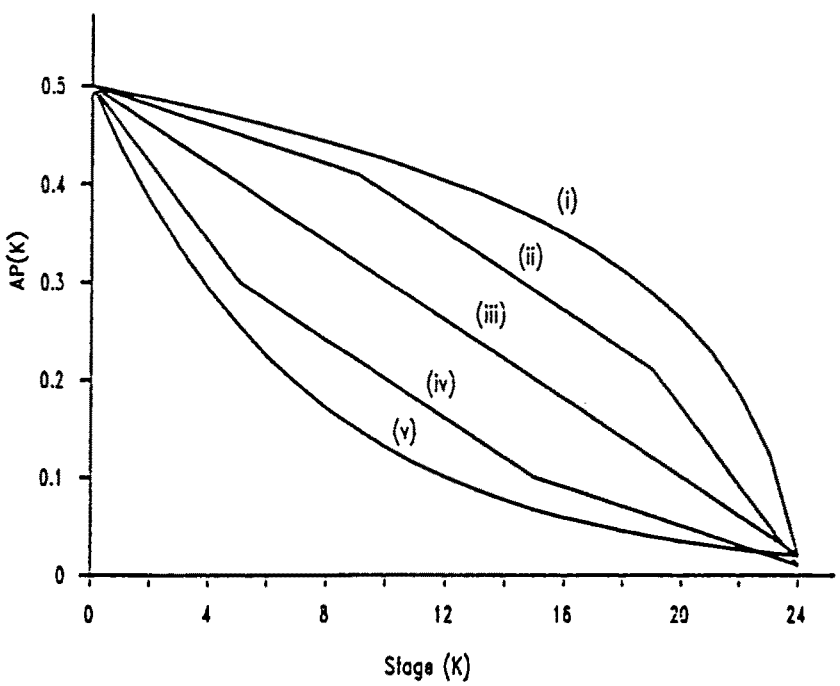

Fig. 3. Functional forms of acceptance probability.

Table 3 displays the saving in computation effort when a good initial solution is used with CSSA. By using the AU rule to drive an initial seed solution, the computation time is decreased to approximately one-fourth of that required when using a random initial seed solution. Note that when an arbitrary initial seed solution (ARB) is used, then the number of searches at each stage is doubled in order to obtain the same accuracy as that using the AU seed solution. Consequently, it takes longer to reach equilibrium at each stage.

5.3. FUNCTIONAL FORM OF ACCEPTANCE PROBABILITY ($AP_{ij}(k)$)

The success of simulated annealing largely depends on the functional form of the acceptance probability, AP_{ij}. To analyze this dependency, we lower AP_{ij} discretely and linearly, and use several other different forms of acceptance probability. Figure 3 shows the various forms of $AP_{ij}(k)$ used. These are:

(i) Logarithmic Concave

$$AP_{ij}(k) = 0.6034 + 0.1491 \ln c(k),$$

where $c(k) = 0.5 - 0.02 \times k$ for $k = 0, 1, \ldots, 24$.

(ii) Piecewise Concave

$$AP_{ij}(k) = \begin{cases} 0.50 - (0.01 \times k) & \text{for} \quad k = 0, 1, \ldots, 9 \\ 0.59 - (0.02 \times k) & \text{for} \quad k = 10, 11, \ldots, 19 \\ 0.97 - (0.04 \times k) & \text{for} \quad k = 20, 21, \ldots, 24 \end{cases}$$

(iii) Linear: $AP_{ij}(k) = 0.5 - (0.02 \times k)$ for $k = 0, \ldots, 24$.

(iv) Piecewise Convex

$$AP_{ij}(k) = \begin{cases} 0.50 - (0.04 \times k) & \text{for} \quad k = 0, 1, \ldots, 5 \\ 0.40 - (0.02 \times k) & \text{for} \quad k = 6, 7, \ldots, 15 \\ 0.25 - (0.01 \times k) & \text{for} \quad k = 16, 17, \ldots, 24 \end{cases}$$

(v) Exponential Convex

$$AP_{ij}(k) = \exp(c(k)),$$

where $c(k) = -0.6931 - (0.1341 \times k)$ for $k = 0, 1, \ldots, 24$.

When $AP_{ij}(k)$ is concave, then it decreases slowly at the early stages, and thereafter it decreases rapidly. The initial acceptance probability is set at 0.5, and is decreased in 25 discrete stages. At each stage, 120 searches are made.

Table 4 gives the ARE, PSO(%) and computational time for various patterns in decreasing $AP_{ij}(k)$. The results indicate that linear and piecewise forms yield greater improvement in the initial solutions than do the logarithmic concave and the exponential convex forms. In terms of the number of the problems optimally solved, the exponential convex form of $AP_{ij}(k)$ yields significantly poorer results.

Table 4
The performance of the forms of $AP(k)$
ARE, PSO(%) and computational time

w	R	r	AU	CONVEX		LINEAR	CONCAVE	
				EXP	PIE	LIN	PIE	LOG
1–10	0.4	0.2	0.6139 (12)	0.2761 (14)	0.0273 (19)	0.0018 (19)	0.0024 (19)	0.0000 (20)
		0.4	0.1456 (6)	0.1452 (6)	0.0278 (16)	0.0079 (17)	0.0187 (17)	0.0088 (17)
		0.6	0.0469 (3)	0.0460 (3)	0.0029 (17)	0.0007 (17)	0.0007 (18)	0.0047 (15)
		0.8	0.0139 (4)	0.0070 (9)	0.0000 (19)	0.0000 (20)	0.0006 (18)	0.0007 (17)
	0.8	0.2	0.0000 (20)	0.0000 (20)	0.0000 (20)	0.0000 (20)	0.0000 (20)	0.0000 (20)
		0.4	0.7190 (4)	0.2467 (6)	0.0106 (17)	0.0336 (18)	0.0000 (20)	0.0369 (18)
		0.6	0.0945 (1)	0.0945 (1)	0.0002 (18)	0.0005 (18)	0.0007 (16)	0.0017 (17)
		0.8	0.0206 (5)	0.0178 (6)	0.0000 (20)	0.0000 (20)	0.0000 (20)	0.0001 (17)
PSO (%)			34.38	40.63	91.25	93.13	92.50	88.13
Time (sec.)			0.003	0.098	0.080	0.081	0.082	0.097

( ): the number of problems solved optimally.

However, the performance of the linear and piecewise forms is approximately the same. Table 4 also shows that linear and piecewise forms are slightly more efficient than the exponential and the logarithmic in terms of computation time. From among the forms tested, the linear discrete form of $AP_{ij}(k)$ performs best.

5.4. THE NUMBER OF SEARCHES AT EACH STAGE

At each stage, simulated annealing theoretically continues to search until the system reaches an equilibrium state. To test the effect on performance of the number of searches at each stage, we use a constant number of searches at all

Table 5
The number of searches at each stage
ARE, PSO(%) and computational time

w	R	r	AU	CSSA				
				15	30	60	120	240
1–10	0.4	0.2	0.6139 (12)	0.0696 (17)	0.0302 (18)	0.0273 (19)	0.0018 (19)	0.0000 (20)
		0.4	0.1456 (6)	0.1060 (8)	0.0622 (13)	0.0167 (12)	0.0079 (17)	0.0100 (17)
		0.6	0.0469 (3)	0.0049 (16)	0.0039 (15)	0.0064 (16)	0.0007 (17)	0.0018 (18)
		0.8	0.0139 (4)	0.0025 (13)	0.0012 (15)	0.0006 (18)	0.0000 (20)	0.0002 (19)
	0.8	0.2	0.0000 (20)	0.0000 (20)	0.0000 (20)	0.0000 (20)	0.0000 (20)	0.0000 (20)
		0.4	0.7190 (4)	0.0792 (14)	0.0626 (14)	0.0520 (15)	0.0336 (18)	0.0000 (20)
		0.6	0.0945 (1)	0.0202 (12)	0.0012 (16)	0.0097 (15)	0.0005 (18)	0.0007 (18)
		0.8	0.0206 (5)	0.0004 (14)	0.0011 (14)	0.0003 (16)	0.0000 (20)	0.0000 (20)
PSO (%)			34.38	71.25	78.13	81.88	93.13	95.00
Time (sec.)			0.003	0.012	0.023	0.042	0.081	0.172

( ): the number of problems solved optimally.

stages. This number is labeled M. The initial acceptance probability is set at 0.5, and is decreased linearly and discretely by 0.02 until it reaches 0.02. Experiments are conducted for $M = 15$, 30, 60, 120 and 240. Here, 15 searches at each stage are necessary to cover the entire neighborhood of a seed sequence at least once.

Table 5 displays the results using the various values of M. As expected, there is a significant improvement in accuracy as the number of searches increases. However, for $M \geqslant 120$ further improvement is minimal except for $r = 0.4$ and $R = 0.8$. Consequently, the system approaches the equilibrium state within 120 searches at each level. As can be seen in table 5, the number of problems solved optimally also increases as the number of the searches increases. In particular, significant improvement is realized as the number of searches increases from 60 to 120. However, there is no significant improvement for $M \geqslant 120$. These results clearly indicate that the system reaches an equilibrium state for $M \leqslant 120$. With all these experiments, the computation time increases almost linearly with the number of searches.

5.5. SEARCH METHOD

Two methods for searching the neighborhood of a seed sequence are examined. The first is a random search method that randomly draws a sequence from the neighborhood of a seed sequence. The solution value associated with the sequence is then compared with that of the seed. With this search method, an equal selection probability is assigned to each of the neighborhood solutions; that is, each solution in the neighborhood of a seed is chosen with probability $1/(n-1)$ for an n-job problem. The second search method examined is the sequential search method previously described in section 2. Note that in this case, each

Table 6
Search methods
ARE, PSO (%) and computational time

w	R	r	AU	CSSA	
				RANDOM	SEQUENCE
1–10	0.4	0.2	0.6139 (12)	0.0406 (18)	0.0018 (19)
		0.4	0.1456 (6)	0.0200 (14)	0.0079 (17)
		0.6	0.0469 (3)	0.0031 (16)	0.0007 (17)
		0.8	0.0139 (4)	0.0003 (18)	0.0000 (20)
	0.8	0.2	0.0000 (20)	0.0000 (20)	0.0000 (20)
		0.4	0.7190 (4)	0.0343 (17)	0.0336 (18)
		0.6	0.0945 (1)	0.0014 (15)	0.0005 (18)
		0.8	0.0206 (5)	0.0004 (17)	0.0000 (20)
PSO (%)			34.38	84.38	93.13
Time (sec.)			0.003	0.090	0.081

( ): the number of problems solved optimally.

neighborhood solution has a different probability of selection. As before, the initial value of AP_{ij} is set to 0.5, and it is linearly decreased to 0.02 at the rate 0.02. The number of searches at each stage is set to 120.

Table 6 shows that the sequential search is superior to the random search in terms of both accuracy and number of problems solved optimally. The random search requires more computational time to select the next move than does sequential search.

5.6. DEPENDENCE ON ΔC_{ij}

Most applications of simulated annealing relate $AP_{ij}(k)$ to the change in the objective function value, ΔC_{ij}. Here, we investigate the benefit of this approach. We use the acceptance probability $AP_{ij}(k) = \min\{\exp(-\Delta C_{ij}/\beta_k), 1\}$, and determine β_k so that the proportion of accepted moves at stage k is approximately equal to $AP(k) = 0.5 - 0.02(k-1)$ for $k = 1, 2, \ldots, 25$. Let $\Delta\bar{C}(k)$ denote the average of the positive changes ΔC_{ij} at stage k. We estimate $\Delta\bar{C}(k)$ using exponential smoothing with smoothing constant equal to 0.2. That is,

$$\Delta\bar{C}_m = 0.8\Delta\bar{C}_{m-1} + 0.2\Delta C_{ij}$$

for the mth positive change in the objective function. The initial average value $\Delta\bar{C}(0)$ is computed as the average of the positive changes when simulated annealing with $AP_{ij}(k) = 0.5$ is iterated 120 times. In order for solution j with the average amount of positive change to be accepted with probability $AP_{ij}(k)$, we set

$$\exp\left(-\Delta\bar{C}(k-1)/\beta_k\right) = AP(k).$$

This implies

$$\beta_k = -\Delta\bar{C}(k-1)/\ln(AP(k)).$$

Table 7
Acceptance probability dependent on the change of the objective function value
ARE, PSO (%) and computational time

w	R	r	LINEAR	EXPON
1–10	0.4	0.2	0.0018 (19)	0.0024 (19)
		0.4	0.0079 (17)	0.0028 (18)
		0.6	0.0007 (17)	0.0002 (19)
		0.8	0.0000 (20)	0.0000 (20)
	0.8	0.2	0.0000 (20)	0.0000 (20)
		0.4	0.0336 (18)	0.0084 (18)
		0.6	0.0005 (18)	0.0000 (20)
		0.8	0.0000 (20)	0.0000 (20)
PSO (%)			93.13	96.25
TIME (sec.)			0.081	0.127

EXPON: exponential form of acceptance probability dependent on ΔC_{ij}
( ): the number of problems solved optimally.

Table 7 shows that the performance using acceptance probabilities independent of ΔC_{ij} is nearly as good as that obtained using acceptance probabilities dependent on ΔC_{ij}. In this study, we use AP_{ij} that are independent of ΔC_{ij}. This facilitates designing experiments to empirically determine the best annealing schedule.

5.7. COMBINED APPROACH USING TWO HEURISTIC SEED SOLUTIONS

Controlled search simulated annealing is so computationally efficient that it might be practical to apply it twice using different initial seed solutions. In this case, the best final solution would be the one selected. To test this combined approach, COVERT and AU rules are used to generate initial seed sequences. The initial value of AP_{ij} is set equal to 0.5, and is decreased linearly and discretely at the rate 0.02. The number of searches at each stage equal to 120. As shown in table 8, the combined method yields improved accuracy, but with twice the computational effort. It is of interest to compare these results with those obtained by CSSA in which only the AU rule is used to develop an initial seed, and 240 searches at each stage are made. Although computation time is almost same, the combined method outperforms single simulated annealing in terms of accuracy. The combined method solves optimally nearly 99% of al 15-job test problems. We of course expect this result since the solution approximately approaches steady state within 120 searches.

5.8. PERFORMANCE OF CSSA FOR LARGE SIZE PROBLEMS

We have empirically shown that CSSA improves accuracy when solving 15-job problems, and does so in approximately 0.1• cpu seconds. We now examine its

Table 8
Combined approach using two heuristic seed solutions
ARE, PSO (%) and computational time

w	R	r	CSSA(CAR+AU)	CSSA(240)
1–10	0.4	0.2	0.0000 (20)	0.0000 (20)
		0.4	0.0000 (20)	0.0100 (17)
		0.6	0.0002 (19)	0.0018 (18)
		0.8	0.0000 (20)	0.0002 (19)
	0.8	0.2	0.0000 (20)	0.0000 (20)
		0.4	0.0027 (19)	0.0000 (20)
		0.6	0.0000 (20)	0.0007 (18)
		0.8	0.0000 (20)	0.0000 (20)
PSO (%)			98.75	95.00
TIME (sec.)			0.170	0.173

(240): 240 searches at each stage
( ): the number of problems solved optimally.

performance for problems involving 50 jobs and 100 jobs. The initial value of AP_{ij} is set at 0.5, and is decreased linearly and discretely at the rate 0.02. There are 25 stages, and the number of searches at each stage is 800 for 50-job problems and 1600 for 100-job problems. The performance of CSSA is compared with simulated annealing method developed by Aarts and Laarhoven [1]. Aarts and Laarhoven use the following set of formulas to determine an annealing schedule:

(i) The control parameter β_k at stage k is determined by

$$\beta_k = \beta_{k-1}(1 + \ln(1+\delta)\beta_{k-1}/3\sigma(\beta_{k-1}))^{-1},$$

where δ is a control constant nd $\sigma^2(\beta_{k-1})$ denotes the variance of the objective function values at stage $k-1$.

(ii) The termination criterion is given by

$$Z = \frac{\beta_k}{\bar{C}(\beta_0)} \frac{d\bar{C}_s(\beta_k)}{d\beta_k} < \epsilon,$$

where $\bar{C}_s(\beta_k)$ denotes the average value of the objective function at stage k and $\bar{C}(\beta_0)$ denotes the average value of the objective function at the initial stage.

Based on preliminary tests, δ is set equal to 0.5, the number of searches at each stage equal to 1600, and the termination criterion ϵ equal to 0.01. Because the algorithm sometimes did not terminate within a reasonable time, another termination criterion is added that limits the number of stages to 300.

Table 9 and Table 10 show that CSSA method using only 25 stages performs well for 50-job and 100-job problems in terms of accuracy and computation time. The exception is for problems involving the combination of parameters: ($r = 0.4$ and $R = 0.4$), (0.6 and 0.4), and (0.6 and 0.8). The inferior performance of the algorithm in these three cases is attributed to the inferior results of the AU rule used to develop the initial seed solution. For 100-job problems, the AU rule provides initial seed solutions that have relative errors in excess of 36%, 8%, and

Table 9
ARI for 50-job problems

w	R	r	AARTS		CSSA	
			ARI	TIME	ARI	TIME
1–10	0.4	0.2	0.9473	8.28	0.9473	0.631
		0.4	0.8206	8.70	0.9443 (0.8132)	0.649 (5.78)
		0.6	0.9777	9.21	0.9749	0.639
		0.8	0.9967	9.84	0.9963	0.627
	0.8	0.2	1.0000	7.86	1.0000	0.027
		0.4	0.8054	8.66	0.7790	0.628
		0.6	0.9286	9.62	0.9588 (0.9355)	0.627 (5.73)
		0.8	0.9952	9.85	0.9955	0.627

TIME: CPU seconds
( ): results obtained by using the adjusted annealing schedule.

Table 10
ARI for 100-job problems

w	R	r	AARTS		CSSA	
			ARI	TIME	ARI	TIME
1–10	0.4	0.2	0.8705	15.82	0.8587	1.35
		0.4	0.7673	17.41	0.9496 (0.7602)	1.34 (11.64)
		0.6	0.9263	17.50	0.9871 (0.9288)	1.31 (11.59)
		0.8	0.9976	12.40	0.9977	1.28
	0.8	0.2	1.0000	10.70	1.0000	0.10
		0.4	1.1106	17.13	0.9307	1.33
		0.6	0.9403	19.13	0.9711 (0.9436)	1.29 (11.70)
		0.8	0.9980	13.99	0.9973	1.28

TIME: CPU seconds
( ): results obtained by using the adjusted annealing schedule.

8% for ($r = 0.4$ and $R = 0.4$), (0.6 and 0.4), and (0.6 and 0.8), respectively. The values in parentheses in tables 9 and 10 are the ARI obtained from using an adjusted parameter set for the annealing schedule. Here, the initial value of AP_{ij} is set to 0.95, and is decreased linearly and discretely at the rate 0.0035 until reaching 0.15. This gives 229 stages, with 800 searches for 50-job problems and 1600 searches for 100-job problems at each stage. This adjusted annealing schedule requires the same computational effort as that developed by Aarts and Van Laarhoven [1]. This implies that if we have a fast algorithm that provides a good heuristic seed solution, then we can significantly speed up convergence of simulated annealing by applying a smaller initial acceptance probability. Since the AU rule does not provide a good seed solution for the three cases identified above, then a full implementation of simulated annealing is necessary.

Comparing tables 1, 9 and 10, the computation time required for CSSA is observed to be approximately proportional to KM.

Table 11 shows the normalized average improvement (NAI) starting from AU solutions based on problem size. For these test problems, CSSA improves AU

Table 11
NAI associated with problem sizes

w	R	r	15-JOB	50-JOB	100-JOB
1–10	0.4	0.2	0.0071	0.0002	0.0007
		0.4	0.0511	0.1634	0.3888
		0.6	0.0501	0.0589	0.4258
		0.8	0.0397	0.0284	0.0350
	0.8	0.2	0.0000	0.0000	0.0000
		0.4	0.0330	0.0076	0.0062
		0.6	0.0806	0.1171	0.1646
		0.8	0.0581	0.0375	0.0435

solutions for 50-job and 100-job problems at least as much as for 15-job problems with respect to the decrease of tardiness per job. Here, the annealing schedule for 15-job problems is that used to derive table 1, and those for 50-job and 100-job problems are the adjusted annealing schedules used to derive tables 9 and 10, respectively.

6. Conclusions and further research

This paper presented an efficient new approach for the application of simulated annealing to the single machine total weighted tardiness problem. Controlled search simulated annealing uses a good heuristic solution as an initial seed along with a low initial acceptance probability. This approach on the surface may appear to be conflicting with a basic tenet of simulated annealing. Convergence theorems developed in the context of simulated annealing suggest that the choice of an initial solution does not affect the probability of attaining a global optimal solution after an infinite number of iterations. Therefore, conventional applications of simulated annealing use an arbitrary initial solution along with a high acceptance probability. This approach often requires a large amount of computer time to eliminate the effect of the initial solution. That is, the conventional approach can be computationally impractical for realtime implementation. In this paper, we show that exploiting a good initial solution accelerates the process of attaining a near optimal solution, and attains much more rapidly a near optimal solution. Extensive computational experiments indicate that CSSA obtains optimal solutions for 99% of all 15-job problems tested. The experiments also indicate that a good seed solution obtained by a fast heuristic method can be used to significantly reduce the computation time without reducing accuracy. However, many simple heuristic dispatching rules do not provide good solutions for particular problem structures. For these cases, a full implementation of simulated annealing is necessary to obtain a near optimal solution.

An alternative not considered in depth in this research is the use of multiple initial solutions with simulated annealing. Such an approach may in fact speed up identification of a near optimal solution. As table 8 indicates, two good solutions that are derived from different heuristic rules serve as complementary starting points. Here, generating such a set of initial solutions is key for success. Also, heuristics for improving the set of initial solutions need to be identified so that a near optimal solution can be obtained. At this point, it is speculative whether or not such multiple starting points used in conjunction with heuristic methods of improvement will work efficiently.

In this study, we use acceptance probabilities that are independent of the change in the objective function value. This was done without deteriorating the performance of CSSA by restricting the search to a small neighborhood. The constant acceptance probability at each stage facilitates investigating the effect of annealing schedules on performance. Consequently, it was possible to show that

annealing schedules with a linear functional form outperforms the rest, and that using constant acceptance probabilities at each stage yields results nearly as good as using the Metropolis acceptance probability.

We have also developed a generic framework for designing simulated annealing algorithms, and have applied it to the single machine weighted tardiness problem. As demonstrated, CSSA works very well for this NP-complete problem. We expect that the superior performance of CSSA can be extended to nearly all single machine sequencing problems studied in the literature. This expectation is based on the following observations:

(1) The CSSA algorithm is not affected by the functional form of the objective function. That is, several scheduling criteria that are important in practice, but that are computationally intractable, can be used without significant alternation from the approach proposed in this paper. These criteria include quadratic penalty costs, combined earliness and tardiness costs, and other combinations of multiple criteria.

(2) Inclusion of release time and deadline constraints into single machine scheduling problems may require further study of the composition of the neighborhood. Other than the definition of neighborhood, CSSA should work well.

(3) Restricting our attention to permutation schedules for flowshops, then again CSSA can be applied almost without alternation. Here, the permutation schedule means that only a loading sequence of jobs is determined, and that the same sequence is applied to all machines in the flowshop. Therefore, a feasible solution is chosen from permutations of $\{1, 2, \ldots, n\}$ as in the single machine problem.

It will be important to verify that CSSA works well for the problems mentioned above. If our speculation proves correct, then we can expect extensive use of simulated annealing on shop floors. Also, from an academic perspective, solutions derived using CSSA may be used as bench marks for these problems. For example, CSSA was used to show that the AU rule does not work well for large problems with particular parameter settings. When researchers construct efficient heuristics for the various problems, then they should be compared with results using CSSA as described in this paper. As a consequence, CSSA may accelerate the development of other faster heuristics through serving as a bench mark method. It may serve to establish bench mark solutions for large problems.

CSSA perhaps can be applied to much more complex scheduling problems. However, application to such problems may require modifications due to differences in the solution structure. The types of modifications necessary for the successful application of CSSA to other scheduling environments warrant further investigation.

Acknowledgement

This research was supported in part by the Advanced Manufacturing Program in the IC^2 institute, the University of Texas at Austin, and by a University

Research Institute Summer Research Award, the University of Texas at Austin. The authors are grateful to the referees for their helpful comments.

References

[1] E.H.L. Aarts and P.J.M. van Laarhoven, Statistical cooling: a general approach to combinatorial optimization problems, Philips J. Res. 40 (1985) 193–226.
[2] S. Anily and A. Federgruen, Simulated annealing methods with general acceptance probabilities, J. Applied Probability 24 (1987) 657–667.
[3] S. Anily and A. Federgruen, Ergodicity in parametric non-stationary Markov chains: an application to simulated annealing methods, Operations Research 35 (1987) 867–874.
[4] K.R. Baker and J.B. Martin, An experimental comparison of solution algorithms for the single machine tardiness problem, Naval Research Logistics Quarterly 21 (1974) 187–199.
[5] I.O. Bohachevsky, M.E. Johnson and M.L. Stein, Generalized simulated annealing for function optimization, Technometrics 28 (1986) 209–217.
[6] D.C. Carroll, Heuristic sequencing of single and multiple components, Ph.D. Dissertation, Massachusetts Institute of Technology, MA, 1965.
[7] V. Cerny, Thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm, Journal of Optimization Theory and Applications 45 (1985) 41–51.
[8] S.E. Elmaghraby, The one-machine sequencing problem with delay costs, Journal of Industrial Engineering 19 (1968) 105–108.
[9] H. Emmons, One-machine sequencing to minimize certain functions of job tardiness, Operations Research 17 (1969) 701–705.
[10] M.L. Fisher, A dual algorithm for the one-machine scheduling problem, Mathematical Programming 11 (1976) 229–252.
[11] S.B. Gelfand and S.K. Mitter, Analysis of simulated annealing for optimization, *Proc. 24th Conf. on Decision and Control*, Ft. Lauderdale, December 1985, 779–786.
[12] S. Geman and D. Geman, Stochastic relaxation, Gibbs Distributions, and the Bayesian restoration of images, IEEE Proc. Pattern Analysis and Machine Intelligence 6 (1984) 721–741.
[13] B. Gidas, Nonstationary Markov chains and convergence of the annealing algorithm, Journal of Statistical Physics 39 (1985) 73–131.
[14] J.W. Greene and K.J. Supowit, Simulated annealing without rejected moves, IEEE Trans. on Computer-Aided Design, CAD-5 (1986) 221–228.
[15] B. Hajek, Cooling schedules for optimal annealing, Mathematics of Operations Research 13 (1988) 311–329.
[16] S. Kirkpatrick, C.D. Gelatt, Jr. and M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[17] E.L. Lawler, A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness, Annals of Discrete Mathematics 1 (1977) 331–342.
[18] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, Complexity of machine scheduling problem, Annals of Disrete Mathematics 1 (1977) 343–362.
[19] H. Matsuo and G. Tang, Worst case and empirical analysis of local search heuristics for the one-machine total tardiness problem, Graduate School of Business, University of Texas at Austin, Preprint, 1986.
[20] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, Equation of state calculations by fast computing machines, Journal of Chemical Physics 21 (1953) 1087–1092.
[21] D. Mitra, F. Romeo and A.L. Sangiovanni-Vincentelli, Convergence and finite-time behavior of simulated annealing, Adv. Appl. Prob. 18 (1986) 747–771.

[22] E.R. Montague, Jr., Sequencing with time delay costs, Arizona State University Industrial Engineering Research Bulletin (January 1969) 20–31.
[23] T.E. Morton, R.M. Rachamadugu and A.P.J. Vepsalainen, Accurate myopic heuristics for tardiness scheduling, GSIA Working Paper #36-83-84, Carnegie-Mellon University, PA, 1984.
[24] J.C. Picard and M. Queyranne, The time-dependent traveling salesman problem and applications to the tardiness problem in one-machine scheduling, Operations Research 26 (1978) 86–110.
[25] C.N. Potts and L.N. Van Wassenhove, A decomposition algorithm for the single machine total tardiness problem, Operations Research Letters 1 (1982) 177–181.
[26] C.N. Potts and L.N. Van Wassenhove, A branch and bound algorithm for the total weighted tardiness problem, Operations Research 33 (1985) 363–377.
[27] A.H.G. Rinnooy Kan, B.J. Lageweg and J.K. Lenstra, Minimizing total costs in one-machine scheduling, Operations Research 23 (1975) 908–927.
[28] A. Schild and I.J. Fredman, On scheduling tasks with associated linear loss functions, Management Science 7 (1961) 280–285.
[29] L. Schrage and K.R. Baker, Dynamic programming solution of sequencing problem with precedence constraints, Operations Research 26 (1978) 444–449.
[30] J. Shwimer, On the N-job, one-machine sequence-independent scheduling problem with tardiness penalties: a branch and bound solution, Management Science 18 (1972) B301–B313.
[31] P.J.M. Van Laarhoven and E.H.L. Aarts, *Simulated Annealing: Theory and Applications* (D. Reidel, Dordrecht, 1987).
[32] L.J. Wilkerson and J.D. Irwin, An improved algorithm for scheduling independent tasks, AIIE Transactions 3 (1971) 239–245.