Some New Results on Simulated Annealing Applied to the Job Shop Scheduling Problem

M. Kolonko
Institut für Mathematik, Universität Hildesheim
Marienburger Platz 22, D-31141 Hildesheim
Germany [1]
13.12.96

Abstract: We present two results about heuristic solutions to the job shop scheduling problem. First, we show that the well-known analytical results on convergence of simulated annealing do not hold in the application to the job shop scheduling problem. We give a simple counterexample where the simulated annealing process converges against a suboptimal schedule. To overcome this problem at least heuristically, we present a new approach that uses a small population of simulated annealing runs in a genetic algorithm framework. The novel features are an adaptive temperature control that allows 'reheating' of the simulated annealing and a new type of time-oriented crossover of schedules. Though the procedure uses only standard properties of the job shop scheduling problem it yields excellent results on the classical test examples and improves some of their best known solutions.

Keywords: scheduling theory, simulated annealing, adaptive temperature schedule, genetic algorithms, hybrid optimization

[1] now at Institut für Mathematik, Techn. University Clausthal, Postfach 1253, D-38670 Clausthal-Zellerfeld, Germany, email : kolonko@math.tu-clausthal.de

1 Introduction

The job shop scheduling problem (JSP) is a well-known NP-hard problem of combinatorial optimization. It is also known to be a very difficult problem such that some test problems of moderate size are still unsolved. Modern heuristic methods as simulated annealing (SA), genetic algorithms (GA) and tabu search take up a growing space in the literature on the JSP, see e.g. [4] for a recent comprehensive overview.

In this paper we present a negative and a positive result. The negative result is that SA, when applied to the JSP, is no longer a convergent stochastic process, though this is generally assumed in the literature (see [21] and [22], references to [22] are made in many papers on JSP). The reason for this is that the standard neighbourhoods used for the JSP are not symmetric. It turns out that if suboptimal solutions are easy to access but difficult to leave by the SA process, then the process may converge to these non-optimal states. We present a simple counterexample for which the limiting distribution, that can be calculated explicitely, concentrates on a suboptimal schedule for decreasing temperature values. Usually, SA implementations return the best solution seen during the run as their result. The lack of convergence means that the sequence of solutions produced by SA need not approach the optimal solution even after an excessive amount of computations. Hence, also the best solution seen after a finite number of steps will tend to be worse than under a convergent procedure. To make this point precise one must have detailed information on the rate of convergence which is a difficult problem with SA (see e.g. [18]).

As a positive result we present a combination of simulated annealing and genetic algorithms with some new ingredients. The population of the genetic algorithm consists of independent simulated annealing runs. They use an adaptive temperature control that allows to leave local minima faster than the standard cooling schedules and still preserves the classical convergence properties. As the asymmetric neighbourhoods of the JSP may distroy this convergence too, we allow the solutions of the SA runs to crossover in each generation using a new time-oriented crossover operation. This improves the empirical behaviour of the procedure considerably when compared with simple SA.

We start with a short formal description of the JSP in section 2. In Section 3 we briefly collect the main results about the convergence of simulated annealing, show where the JSP fails and present the counterexample. The genetic algorithm framework and the new crossover operation for schedules are defined in Section 5. Also, the main ideas of the adaptive temperature control introduced in [14] are sketched there. Finally in Section 6 we give some implementational details and our computational results for the hard problems from the standard test library JSPlib. Though the procedure does not use any problem specific knowledge except the standard methods to produce neighbour solutions, the computational results are quite excellent.

2 The Job Shop Scheduling Problem

In a JSP there are given n *jobs* , each consisting of m *operations* and there are m *machines* each of which can handle exactly one of the m operations of each job. The order in which the operations of one job have to be performed is fixed. Each operation needs a certain amount of *processing time*. The

aim is to find an order x of the operations on each machine such that the finishing time $c(x)$ of the last job (the overall processing time for all jobs) is minimized. An overview about different scheduling problems can be found in [5].

A problem instance can be given as a directed graph where the vertices are the operations (plus a start and a stop node) and the arcs between operations of one job define their required order, see Fig. 1. A solution x can then be represented by additional arcs between the operations on each machine in the order determined by x, see Fig. 2. A solution x is feasible, if no deadlocks occur, i.e. if there are no cycles in the graph. Let S denote the set of feasible solutions.

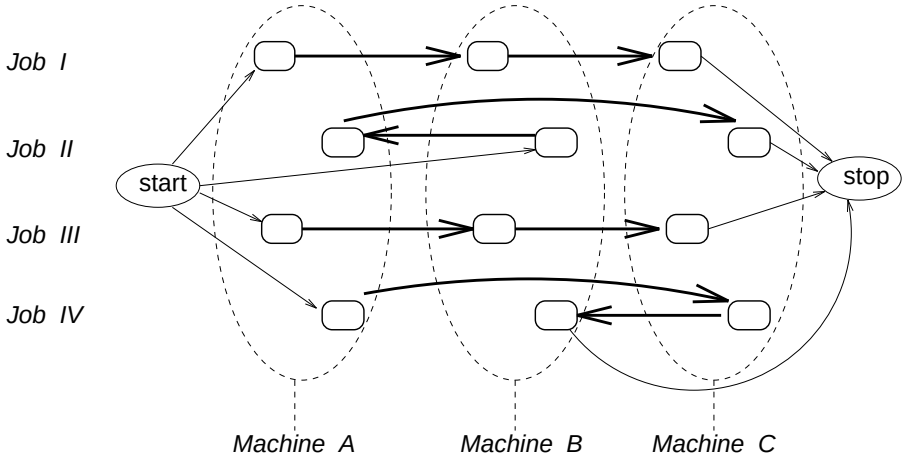

Figure 1: A job shop scheduling problem with four jobs, each consisting of three operations (indicated by rounded boxes) which have to be performed on the machines A,B and C in the order of the arcs.

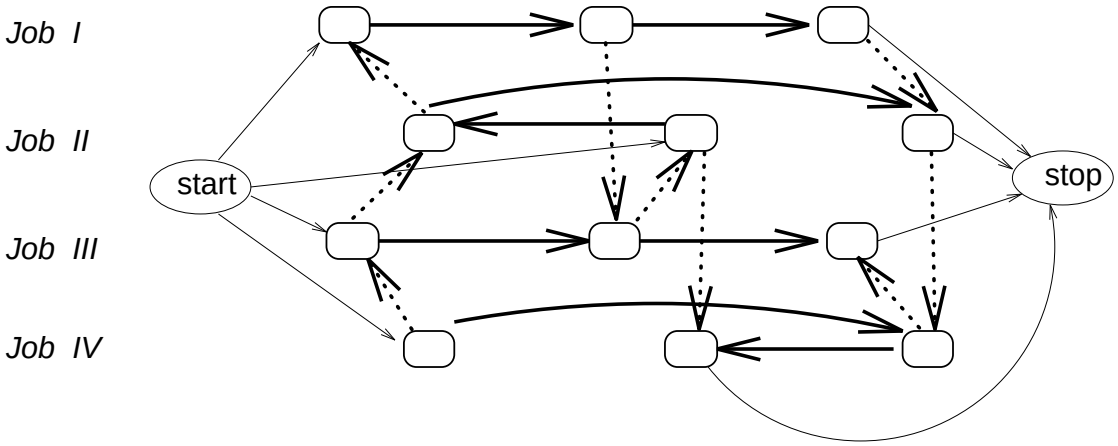

Figure 2: For the problem instance from Fig. 1 the dotted arcs represent a feasible solution x, i.e. a feasible order of the operations on each machine

Let $\mathcal{J}, \mathcal{O}$ and $\mathcal{M}$ denote the set of jobs, operations and machines. We write $o' \prec o$ if operations $o', o \in \mathcal{O}$ belong to the same job and o' has to be finished before operation o can start. We write $o' \prec_x o$ if o, o' have to be performed on the same machine and if o' precedes o in the precedence given by a solution x. Let

$$o' \preceq_x o \ :\Longleftrightarrow \ \Big((o' \prec o) \ \vee (o' \prec_x o)\Big).$$

If x is a feasible solution we can visit the nodes (operations) in the so-called topological order, i.e.

we can enumerate the operations in a (not necessarily unique) way as $(o_1, \ldots, o_{mn})$ such that each operation o appears after all its predecessors with respect to $\preceq_x$. Let $p(o)$ denote the processing time of operation o. Then we can define the earliest starting time $t(o)$ of each operation $o \in \mathcal{O}$ under a feasible solution x recursively by

$$t(o) := \max\{t(o') + p(o') \mid o' \preceq_x o,\ o' \in \mathcal{O}\}$$

with $\max \emptyset = 0$.

The costs $c(x)$ of a solution x are then obtained as the latest finishing time

$$c(x) \ := \ \max\{t(o) + p(o) \mid o \in \mathcal{O}\}.$$

In the representation as a graph, $c(x)$ is the length of a longest path from start to stop where the length (duration) of a path is determined by the processing times of its nodes. Edges between operations on the same machine lying on a longest path are called *critical edges*.

There is a huge literature on heuristics for the JSP, see e.g. [5], [4], [17]. For an iterative improvement, most of the authors recommend the so-called N1 operation ([5], p. 210) which produces a new feasible solution from a given one by reversing one critical edge and adapting the adjacent edges, see Fig. 3. Other operators that are in use also reverse edges on longest paths.

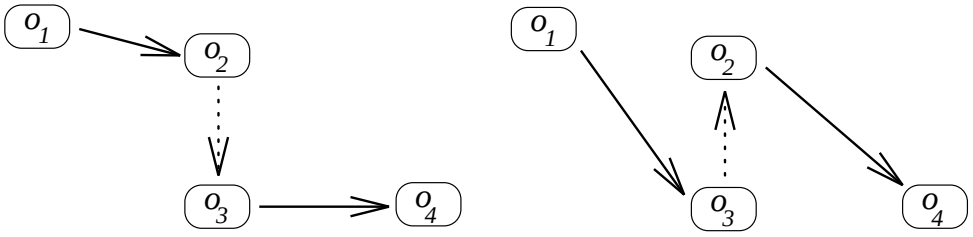

Figure 3: The figure shows part of a longest path. The critical (dotted) arc is reversed and the adjacent edges are adapted

3 Convergence of Simulated Annealing and the JSP

Simulated annealing (SA) is a stochastically relaxed local search method allowing to climb 'uphill' in the cost function landscape. It was introduced and studied e.g. in [11], [6],[15], [8], [18], [2], [13]. See [20] for a survey of the early literature. The first papers studying SA for the JSP seem to be [21] and [22].

We first sketch the main principles of SA and the conditions for its convergence for the general case. Let S be a finite state space and $c : S \to \mathbb{R}_+$ a cost function to be minimized. Given a present solution $x \in S$, SA chooses at random a candidate y from a predefined *neighbourhood* $N(x) \subset S$ of x using a *generating probability* $G(x, \cdot)$. If y is an improvement, i.e. $c(y) < c(x)$, then y replaces x and becomes the new solution. If $c(y) \geq c(x)$ then y is still accepted with the *acceptance probability* $\alpha(x, t, y)$, where t, the so-called *temperature*, is a control parameter. With probability $1 - \alpha(x, t, y)$, the candidate y is rejected. This procedure is repeated until some stopping criterion is fulfilled.

Usually, $G(x,\cdot)$ is the uniform distribution on the neighbourhood $N(x)$ or some other distribution with $G(x,y) > 0 \iff y \in N(x)$. α is chosen as

$$\alpha(x,t,y) := \begin{cases} 1 & \text{if } c(y) \le c(x) \\ \exp\left(-\frac{c(y)-c(x)}{t}\right) & \text{if } c(y) > c(x) \end{cases} \qquad \text{with} \quad t > 0 \tag{1}$$

(the so-called Metropolis acceptance probability). Hence the probability to accept a candidate y with $c(y) > c(x)$ decreases with decreasing temperature t, forcing the process to a stable behaviour in the long run.

More precisely, the process of solutions is modeled as a Markov chain $(X_n)_{n\ge 0}$ with state space S and transition probability

$$P_t(x,y) := \begin{cases} G(x,y)\alpha(x,t,y) & \text{if } x \ne y \\ 1 - \sum_{y' \ne x} G(x,y')\alpha(x,t,y') & \text{if } x = y \end{cases} \tag{2}$$

where we assume (1). For the so-called homogeneouos analysis with *fixed* temperature $t > 0$ we have the following convergence theorem.

Theorem 1

Let $S^* = \{x \in S \mid c(x) = \min_{x' \in S} c(x')\}$ be the set of optimal states and let $S_1, \ldots, S_k$ be the recurrent classes of the Markov matrix $G(\cdot,\cdot)$. If, for all $1 \le i \le k$, we have

(i) $S_i \cap S^* \ne \emptyset$ and

(ii) $S_i \cap (S - S^*) \ne \emptyset$

then the following holds.

a) For any $1 \le i \le k$, let Q_i be the matrix P_t restricted to the set $S_i \times S_i$. Then Q_i has a unique stationary distribution π_t^i which is also the limiting distribution of Q_i.

b) For any $x_0 \in S$ we have (for $t > 0$ fixed)

$$\lim_{n\to\infty} \mathbf{P}(X_n = x \mid X_0 = x_0) \le \pi_t^i(x) \quad \text{for } x \in S_i,\ 1 \le i \le k$$

and

$$\lim_{n\to\infty} \mathbf{P}(X_n = x \mid X_0 = x_0) = 0 \quad \text{for } x \notin \bigcup_{i=1}^{k} S_i.$$

Thus if each recurrent class S_i contains optimal as well as non-optimal (i.e. rejectable) states then the local stationary distributions π_t^i yield an upper bound for the limiting distribution of P_t with fixed temperature $t > 0$. The limiting behaviour of π_t^i for $t \to \infty$ will be examined below.

As Theorem 1 follows from standard Markov theory, we only sketch the **proof**. First, we have for the Metropolis acceptance probability

$$\alpha(x,t,y) \ge \exp\left(\frac{-\max\{c(x') - c(y') | x', y' \in S\}}{t}\right) > 0$$

hence it follows from (2) that the recurrent classes of P_t are those of G for $t > 0$. Moreover, by (i) and (ii), a recurrent class S_i contains an optimal state x^* which has a non-optimal neighbouring state $y \in S_i$. Hence S_i is aperiodic. This follows from

$$\begin{aligned}
\mathbf{P}(X_1 = x^* | X_0 = x^*) &= 1 - \sum_{y' \neq x} G(x^*, y')\alpha(x^*, t, y') \\
&\geq 1 - \sum_{y' \neq x, y} G(x^*, y') - G(x^*, y)\alpha(x^*, t, y) \\
&> 1 - \sum_{y' \neq x} G(x^*, y') \\
&= 0.
\end{aligned}$$

as $0 < \alpha(x^*, t, y) = \exp(-(c(y) - c(x^*))/t) < 1$ and $G(x^*, y) > 0$. Hence, each S_i is a recurrent, aperiodic class of P_t and (a) follows.

Now (b) follows from (a) as in [22]. For $x \in S_i$, we have (cp [10], p. 91)

$$\begin{aligned}
&\lim_{n \to \infty} \mathbf{P}(X_n = x \mid X_0 = x_0) \\
&\quad = \mathbf{P}(X_m \in S_i \text{ for some } m \mid X_0 = x_0) \cdot \pi_t^i(x) \\
&\quad \leq \pi_t^i(x),
\end{aligned}$$

whereas for transient states x the limiting probability is 0. ■

We call $\omega = (x_0, \ldots, x_l)$ a *path of length* l from x to y if $x_0 = x, x_l = y$ and $G(x_i, x_{i+1}) > 0$ for all $0 \leq i \leq l - 1$. Let $\hat{c}(\omega) := \max\{c(x_i) \mid 1 \leq i \leq l\}$ be the maximal costs on the path $\omega = (x_0, \ldots, x_l)$. Then

$$\eta(x, y) := \min\{\hat{c}(\omega) \mid \omega \text{ is a path of arbitrary length from } x \text{ to } y\}$$

denotes the minimal cost height that has to be climbed on a path from x to y.

Theorem 2

Assume that G is an irreducible Markov matrix and that for all $x, y \in S$

$$\eta(x, y) = \eta(y, x) \tag{3}$$

then for any $x \in S - S^*$,

$$\lim_{t \to 0} \pi_t(x) = 0,$$

i.e. the steady state distributions concentrate on the optimal states for $t \to 0$.

For a **proof** see [7]. The so-called *weak reversibility* (3) includes the *symmetric case* where $G(x, y) = G(y, x)$ and the *uniform case* where $G(x, \cdot)$ is the uniform distribution on $N(x)$ for each $x \in S$ and where in addition $N(\cdot)$ is symmetric in the sense that

$$y \in N(x) \Rightarrow x \in N(y) \tag{4}$$

for all $x, y \in S$. Note that symmetry of the neighbourhood does not imply symmetry of G as can be seen from Fig. 4. Of course the neighbourhoods in Fig. 4 can be made symmetric by adding edges between pairs of the outer vertices turning the planar structure into a torus. But then also vertices which are 'far apart' in the original structure become close neighbours which may be undesirable.

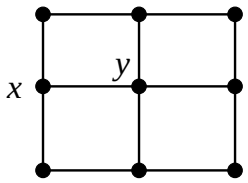

Figure 4: Let the neighbourhood $N(x)$ of a vertex x be the adjacent vertices. Then N is symmetric as the graph is undirected, but $G(x, y) = 1/3, G(y, x) = 1/4$, i.e. G is not symmetric.

In the symmetric case π_t can be given explicitely as

$$\pi_t(x) \;=\; \frac{\exp\Big((c(x^*) - c(x))/t\Big)}{\sum_{y \in S} \exp\Big((c(x^*) - c(y))/t\Big)},$$

where $x^* \in S^*$ is an arbitrary optimal state. Also in the the uniform case

$$\pi_t(x) \;=\; \frac{|N(x)| \exp\Big((c(x^*) - c(x))/t\Big)}{\sum_{y \in S} |N(y)| \exp\Big((c(x^*) - c(y))/t\Big)},$$

see e.g. [15]. Unfortunately, in [15] condition (4) for the uniform case is used only implicitely and not stated explicitely. Other authors seem to have overseen this.

If G has more then one recurrent class, then Theorem 2 may be applied to each recurrent class separately leading to the following general result.

Theorem 3

If each recurrent class of G contains optimal as well as suboptimal solutions and if G is weakly reversible then for fixed $t > 0$ the limiting distributions

$$\lim_{n \to \infty} \mathbf{P}(X_n = x \mid X_0 = x_0) \;=:\; p_t(x_0, x)$$

exist for all $x_0, x \in S$ and have the property

$$\lim_{t \to 0} p_t(x_0, x) \;=\; 0$$

for $x \notin S^*$.

Examining the steady state distributions for fixed $t > 0$ is only a very rough approximation to the real solution process. In most applications, the temperature is changed constantly during the optimization leading to an inhomogeneous Markov chain or even non-Markovian processes. Convergence results for this case are more involved, see e.g. [2], [8], [13], [14] and [18], all of which require symmetry or weak reversibility of G.

To our knowledge, these general convergence results have not been applied correctly in the literature on the JSP. In [1], condition (i) of Theorem 1 was falsely stated to be necessary and sufficient for the convergence of the SA process ([1], 1. (N_1) (iii)). [21] resp. [22] used results as in Theorem 3 but did not check conditions (ii) of Theorem 1 and condition (4) which both need not hold in the application to the JSP. If condition (ii) of Theorem 1 is violated for S_i then this class consists of optimal solutions only and the process cycles periodically through these states. There is no formal convergence of the process in this case, but as it will be in some optimal state in the long run this is of no practical importance.

To see that condition (4) need not hold for the JSP, assume that a solution y has been obtained by reversing a critical edge in a solution x. This edge need no longer be on a longest path in y, hence x need not be a neighbour of y. Even if there is a path from y back to x it need not be at the same cost level as is shown in the counterexample below. Without such symmetry, Theorem 2 or similar results need not hold. The counterexample in the next section is irreducible and has steady state distributions that converge to a non-optimal solution for $t \to 0$.

4 A Counter Example

Consider the extremely simple JSP with two jobs each consisting of three operations as given in Fig. 5.

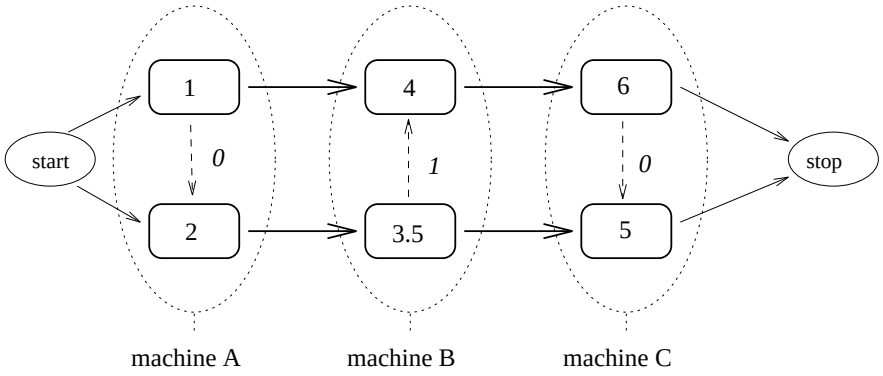

Figure 5: A JSP with two jobs, three machines. The values in the rounded boxes are the processing times. A solution x defines the direction of the dashed arcs. If down-arrows are coded as 0, up-arrows as 1, solutions are 3-bit digits. The picture shows the solution 2 = (010) in binary code.

There are eight possible solutions which are all feasible. Coding the machine precedences for each solution in a binary fashion as indicated in Fig.5 we can calculate the longest paths and obtain the cost function as in Fig.6.

solution no.	0	1	2	3	4	5	6	7
machine precedences	000	001	010	011	100	101	110	111
costs	16	19.5	21.5	17.5	18	21.5	20.5	16.5

Figure 6: The costs, i.e. the length of the longest path, for each of the possible solutions.

We use the Metropolis acceptance probability and the uniform distribution on the N1-neighbours

as generating probability. In the Markov transition graph of P_t in Fig.7 the solutions are the nodes, their costs being represented by their vertical position. To each solution, all adjacent nodes are N1-neighbours, i.e. they can be reached by reversing one critical edge. The asymmetry is obvious. Also it can be seen that G is irreducible. All downward arcs are transitions that are accepted with probability one as they represent an improvement. The bold arcs are transitions which lead to higher costs; their transition probabilities are calculated from (2). We obtain $a = b = \exp(-3.5/t)$, $c = d = \exp(-4/t)$. Transitions without arcs have probability 0.

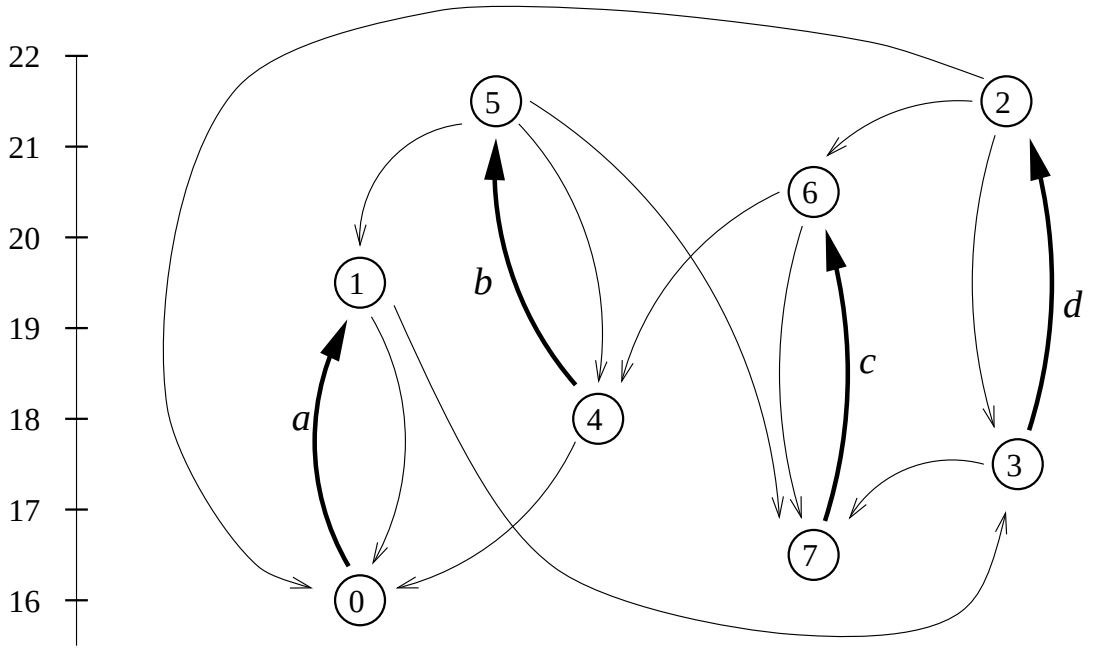

Figure 7: The Markov chain transition graph for the example JSP. The transition probabilities a, b, c and d for the upward arcs are given in the text.

We can calculate the steady state probabilities $\pi_t(0), \ldots, \pi_t(7)$ explicitely from the steady state equations $\pi_t' P_t = \pi_t$ and $\pi_t' \mathbf{1} = 1$. We obtain $\pi_t(x) = \tilde{\pi}_t(x)/C_t$ where

$$\begin{aligned}\big(\tilde{\pi}_t(0), \ldots, \tilde{\pi}_t(7)\big) \;=\; \Big(\;& 3c(6+b+4d+bd),\; 2ac(3+b)(3+2d),\; 3acd(3+b),\; 6ac(3+b),\\ & 6ac(3+d),\; 3abc(3+d),\; 2ac(3+2b)(3+d),\; 3a(6+4b+d+bd)\Big)\end{aligned}$$

and

$$C_t \;=\; 18a + 12ab + 18c + 72ac + 3bc + 33abc + 3ad + 3abd + 12cd + 33acd + 3bcd + 14abcd.$$

As $\lim_{t\to 0} a \;=\; \lim_{t\to 0} b \;=\; \lim_{t\to 0} c \;=\; \lim_{t\to 0} d \;=\; 0$ and $\lim_{t\to 0} c/a \;=\; \lim_{t\to 0} d/a \;=\; 0$ we have

$$\lim_{t\to 0} \pi_t(x) \;=\; \lim_{t\to 0} \frac{\tilde{\pi}_t/a}{C_t/a}$$

with

$$\lim_{t\to 0} \frac{C_t}{a} \;=\; 18 \qquad \text{and} \qquad \lim_{t\to 0} \frac{\tilde{\pi}_t(x)}{a} \;=\; \begin{cases} 18 & \text{for } x = 7 \\ 0 & \text{for } x < 7 \end{cases}.$$

Hence

$$\lim_{t\to 0} \pi_t \;=\; (0, 0, 0, 0, 0, 0, 0, 1)$$

i.e. the steady state probability converges against the one-point mass on $x = 7$ whereas the optimal state $x^* = 0$ has limiting probability 0. This result is quite plausibel as Fig.7 shows that it is easy to get into state 7 but one has to climb 4 units to leave it. State 0 can be left climbing only 3.5 units.

Note that this result also shows that the N1-neighbourhood is not weakly reversible as e.g. 7 can be reached from state 3 at maximal costs $c(3) = 17.5$ whereas any path leading back from 7 to 3 has to pass state 6 at costs $c(6) = 20.5$.

5 Genetic Combinations of SA

If a situation as in the counterexample occurs then increasing the number of search steps in SA will only lead closer to the suboptimal schedule. As a heuristic countermeasure we started several SA runs which then have the chance to get absorbed into different recurrent classes S_i or at least can visit a greater part of the solution space in the case of irreducibility of S. In particular, the combination with genetic algorithms (GA) turned out to be very efficient in our experiments. GA break the local search character of SA by the more 'chaotic' crossover operation and by selection. Also it adds the possibility for parallel computations. Our general approach here is very similar to [16]. We shall first describe the GA framework and the crossover operator and then give some details on the SA runs.

A population in our algorithm consists of K individuals, each of which is an independent SA run with a starting solution and the best schedule found during the run. The 'fitness' of such an individual is the length of its best schedule. For the starting population, the starting values of the SA runs are chosen at random. For the production of a new generation, randomly selected pairs of individuals are taken from the present population and their schedules are crossed in a fashion described below. The resulting schedule is taken as starting solution to a SA run, the best solution found during the SA run is the new 'offspring'. After enough offspring individuals have been produced the population is reduced to its original size by selecting K individuals. We used a random selection where the selection probability of an individuum x is proportional to $\max\{c(x')|x' \in \text{ present population }\} - c(x)$. Note that here fitness is the length $c(x)$ of a solution and has to be minimized. This scheme may either be regarded as parallel simulated annealing with crossover or as a genetic algorithm where simulated annealing serves as a local improvement routine.

The most important feature in this context is the crossover operator, as it must operate on the internal representation of x and should be able to preserve good partial solution from the 'parent' individuals. The crossover we use takes the representation of schedules as Gantt charts (cp. Fig.8), i.e. the schedule is a matrix with a row for each machine. In each row the operations for that machine are given in their order of increasing starting times. For the crossover we first select a random time T between 0 and the best schedule length $c(x_1)$ of the first parent individual. The starting solution x for the offspring individual is formed by taking all operations that start in x_1 not later than T in their order from x_1 into x. Then for each machine all operations from x_2 not yet in x are taken in their order from x_2, skipping all that are already in x.

For a formal definition of the crossover operator, let $t(o)$ be the earliest starting time of operation o under parent schedule x_1. For two operations o, o' to be processed on the same machine we define

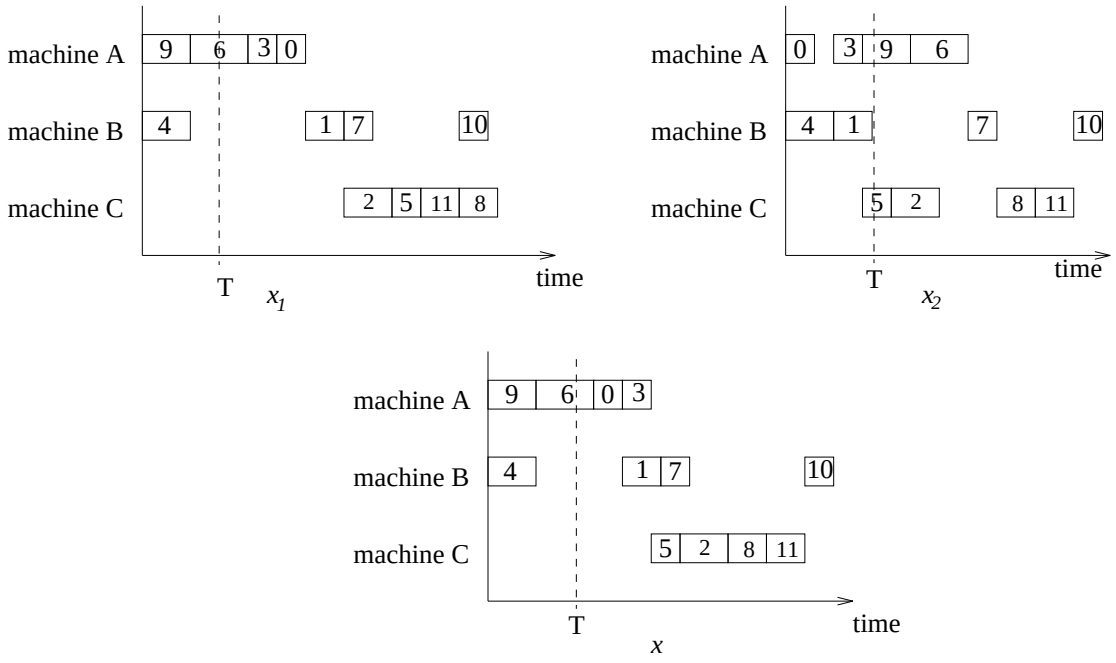

Figure 8: The crossover operates on two parents schedules x_1 and x_2 in Gantt-chart representation. It produces a new feasible schedule x.

the order in the resulting schedule x by

$$o \prec_x o' \ :\Longleftrightarrow \ \begin{cases} \Big(t(o) \le T \ \wedge \ t(o') \le T \ \wedge \ o \prec_{x_1} o'\Big) \ \vee \\ \Big(t(o) > T \ \wedge \ t(o') > T \ \wedge \ o \prec_{x_2} o'\Big) \ \vee \\ \Big(t(o) \le T < t(o')\Big) \end{cases} . \tag{5}$$

The following Theorem asserts that the new schedule x is a feasible solution.

Theorem 4

If $x_1, x_2 \in S$ are feasible solutions then x produced by the crossover (5) is a feasible solution.

Proof : We only have to check that there can be no cycles with respect to $\preceq_x$ (notation from Section 2). Now assume that $o_1, \ldots, o_l$ is a cycle of operations such that $o_1 = o_l$ and $o_i \preceq_x o_{i+1}$ for all $1 \le i \le l-1$. Note that the pairs (o_i, o_{i+1}) are arcs with $o_i \prec o_{i+1}$ or $o_i \prec_x o_{i+1}$.

Let $\mathcal{J}_T := \{o \in \mathcal{O} | t(o) \le T\}$ be the set of operations which start under x_1 not later than T. There are four cases to be checked. If $t(o_j) \le T$ for all $1 \le j \le l$ then $o_1, \ldots, o_l$ is a cycle under the schedule x_1, contradicting the feasibility of x_1. Similarly, $t(o_j) > T$ for all $1 \le j \le l$ contradicts the feasibility of x_2. Now assume that we have for some index $k \ge 1$, $t(o_j) \le T$ for all $1 \le j \le k$. If $t(o_j) > T$ for all $k < j \le l$ this would contradict $o_1 = o_l$. Hence there must be at least one index $j > k$ with $t(o_j) > T$ and $t(o_{j+1}) \le T$. From (5) we see that then $o_j \prec o_{j+1}$, i.e. the arc (o_j, o_{j+1}) must be a precedence given by the job. But then o_j must start before o_{j+1} under x_1, i.e. $t(o_j) \le t(o_{j+1})$, a contradiction. Hence there can be no cycles. ■

There is a second variant of this operation which first takes all operations starting after T from the second parent and then inserts the missing operations in front in the order of the first parent. Note, that these crossover operations are not symmetric or self-inverse and can therefore not analysed that easily (cp. [12]).

In [14], a new type of control for the temperature parameter t in the acceptance probability of the SA was introduced (to avoid misunderstanding we use here the term 'temperature control' instead of the more common 'temperature schedule'). It allows to rise the temperature depending on the history of the solution process thus enabling the process to leave local minima faster. The process will then no longer be a Markov chain. It is shown in [14] that if the temperature changes are reasonable bounded and have an overall tendency towards a given small temperature $\vartheta_0 > 0$ then the process of solutions converges in distribution to the steady state distribution π_{ϑ_0} (or the corresponding matrix if there is more than one recurrent class). If the conditions of Theorem 3 hold then π_{ϑ_0} will be concentrated near the optimal states, for small values of ϑ_0 .

We shall use a particular variant of this scheme which adapts the future temperature to the cost difference $d(x,y) := c(x) - c(y)$ for a candidate solution y and the present solution x. Note that if y is an improvement then $d(x,y) > 0$. The central part of the acceptance mechanism in (1) is the expression $d(x,y)/t$ which relates cost differences to temperatures. To make the two more compatible we 'norm' the cost differences and the temperature changes. First we observe a fixed number of sample solutions with acceptance probability $\alpha = 1$ (i.e. each solution is accepted). From this sequence we calculate an estimate d_{97} for the 97%-quantile of the absolute values $|d(x,y)|$. Assume that d_{97} is positive. Then one can expect that $c(x) - c(y)$ will move in the intervall $(-d_{97}, d_{97})$ for a large portion of steps. Let

$$\hat{d}(x,y) \;:=\; \frac{d(x,y)}{d_{97}} \;=\; \frac{c(x) - c(y)}{d_{97}}$$

be the 'normed' cost difference, which will mainly take on values in $(-1,1)$, depending on the quality of the estimate d_{97}. Let $\vartheta_0 > 0$ be a small given temperature and $t_0 > \vartheta_0$ the starting temperature. We use

$$\Delta_n \;:=\; \frac{t_0 - \vartheta_0}{n^\gamma} \tag{6}$$

as a bound for the $n-$th temperature change with a constant $0.5 < \gamma < 1$. The temperature control in the next Theorem roughly uses temperature steps Δ_n multiplied by $\hat{d}(x,y)$.

Theorem 5

Let x be the present solution in the $n-$th step, y a candidate solution and t the present temperatur. Then we define the temperature t' for the next step by

$$t' := \max\left\{\vartheta_0,\; t \;-\; d_{n+1} \cdot \Delta_n\right\}$$

where d_{n+1} is given by

$$d_{n+1} := \begin{cases} \hat{d}(x,y) & \text{if } c(y) \le c(x), \\ \alpha(x,t,y)^{-1} - 1 - \hat{d}(x,y) & \text{if } c(y) > c(x) \text{ and } y \text{ was accepted}, \\ -\left(1 + \hat{d}(x,y)\right)^+ & \text{if } c(y) > c(x) \text{ and } y \text{ was rejected}. \end{cases} \tag{7}$$

Then

$$\lim_{n\to\infty} \mathbf{P}(X_n = x) \;=\; \pi_{\vartheta_0}(x) \quad \text{for all } x \in S.$$

The **proof** of this Theorem is a direct consequence of Theorems 1 and 4 in [14].

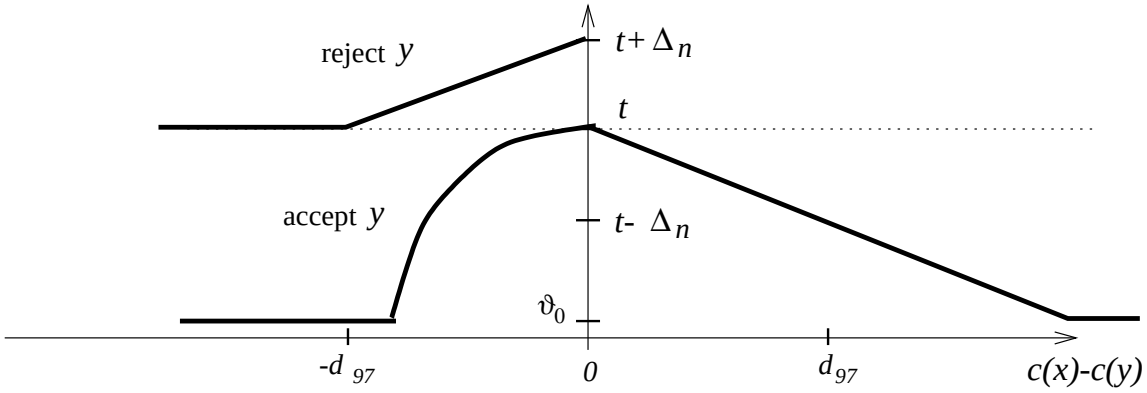

Figure 9: The new temperature value t' as function of the value of $d(x, y)$.

The heuristical motivation of the scheme (7) is the following (see also Fig. 9): the actual temperature change consists of a temperature step $\pm\Delta_n$ scaled by d_{n+1} which itself is a simple function of the normed cost difference $\hat{d}(x, y)$. If y is at least as good as x, this is considered as a success and the temperature is decreased by

$$\hat{d}(x, y)\Delta_n \;=\; \frac{c(x) - c(y)}{n^\gamma} \cdot \frac{t_0 - \vartheta_0}{d_{97}}.$$

If a candidate y with higher costs is accepted then the temperature is lowered by the (possibly huge) amount

$$\Delta_n \left(\frac{1}{\alpha(x, t, y)} - 1 - \frac{c(x) - c(y)}{d_{97}} \right)$$

to damp the uphill-climbing. Finally, if a candidate y with higher costs is rejected, the temperature is increased by the amount $(1+\hat{d}(x, y))^+\Delta_n \geq 0$ (note that in this case $\hat{d}$ is negative). Here, $(1+\hat{d}(x, y))^+$ takes on values near 1 for $c(y)$ only a little larger than $c(x)$, it decreases linearly and is equal to zero for $c(y) > c(x) + d_{97}$. This enables the process to leave local minima more easily.

Fig. 9 shows the behaviour of t' as a function of the cost difference $c(x) - c(y)$ for fixed t. If d_{97} is chosen such that a large portion of all (x, y)-pairs has $|\hat{d}(x, y)| \leq 1$ then the control will move mainly in the central part of the figure.

This temperature control applied to the JSP showed much better empirical results compared with standard SA with fixed temperature sequences, see the computational results in [14]. Nevertheless, as Theorem 3 does not hold for the JSP, we cannot assume that for small values of ϑ_0 the steady state distribution π_{ϑ_0} in Theorem 5 will concentrate near the optimal states. The computational results in the next section show that the combination of GA and SA finds the best values in most cases and even improves some of them.

6 Computational Results

In the implementation of our system 'SAGen' we used the disjunctive graph of the problem to generate and evaluate neighbours of a solution and a Gantt-like list representation of schedules for the crossover operation. The parameter d_{97} for the temperature control as in Theorem 5 and the mean cost difference are estimated from a sample whose size depends linearly on l, the amount of unsuccessful SA steps allowed, see below. These parameters are estimated for each new individuum after crossover and thus are part of the object 'individuum' in an object-oriented sense. The starting temperature t_0 and the minimal temperature ϑ_0 are chosen such that given fractions of the mean cost difference seen in the sample would be accepted, see [14] for details.

			Results with SAGen				Aarts et al.	
problem instance	jobs × mach.	optimal value resp. lower/ upper bound	best value found	average value	mean rel. error	average CPU secs	best value found	average CPU secs
law16	10×10	945	945	945.2	0.000	38.8	969	88.2
law19	10×10	842	842	844.0	0.002	34.6	854.6	93.8
law21	15×10	1046	1047	1051.0	0.005	549.2	1078	243.4
law22	15×10	927	931	932.4	0.006	452.8	944	254.2
law24	15×10	935	938	940.4	0.006	569.6	960	234.8
law25	15×10	977	977	979.0	0.002	644.4	1003	254.8
law27	20×10	1235	1236	1244.8	0.008	3650.6	1275	492.0
law29	20×10	1142 / 1153	1167	1169.2	0.024	4494.0	1225	471.0
law36	15×15	1268	1268	1269.8	0.001	4655.2	1307	602.2
law37	15×15	1397	1401	1412.8	0.011	4144.4	1440	636.2
law38	15×15	1196	1201	1202.4	0.005	5049.4	1235	635.0
law40	15×15	1222	1226	1228.6	0.005	4544.0	1254	596.8
abz07	20×15	656	658	659.6	0.005	28487.4	667	-
abz08	20×15	645 / 669	670	670.4	0.039	28194.6	670	-
abz09	20×15	661 / 679	683	685.6	0.037	26202.0	691	-

Table 1: Results for some of the hard problems from the JSPlib. For our method SAGen averages are from 5 runs. The results 'Aarts et al.' are from [1]

For the starting solutions we examined two different methods. The first simply chooses an order of the jobs and then processes the operations at each machine in the order of the corresponding jobs. This produces very bad starting solutions, still the overall results are only slightly worse than from the second method taken from [3]. Here, the operations are given a random order. Then, on the places occupied by operations belonging to one job, these operations are ordered according to the required precedence of the job. For the time-consuming reevaluation of longest paths after a critical edge has been reversed we used an algorithm that visits only those nodes that are affected, i.e. those that lie 'behind' the new edge in topological order.

As a stopping criterion for the local SA improvement runs we used the number l of neighbour evaluations without improvement of the best value found so far in that run. The whole algorithm stops if for a given number g of generations (i.e. production-reduction cycles of the GA) there has been no improvement of the best solution found so far.

We applied the procedure SAGen to the set of difficult problems from the test library JSPlib [2].

[2] obtainable via `ftp` from `mscmga.ms.ic.ac.uk`

The results are given in Table 1. These are average results from five runs with a population size 10. The algorithm stopped after $g = 2$ consecutive generations without improvement. The local SA improvement stopped after a number of trials without improvement ranging from $l = 10,000$ (for law16) to $l = 2,000,000$ (for abz0x). Due to the stopping criterion, the best solution were found after about 2/3 of the overall time given in the table. The times are for a Pentium 120 for the law-problems and a Pentium 166 for the abz-problems. The last two columns in Table 1 give the best average results from [1]. They compared different multi-start, SA and genetic algorithms. In contrast to their study we adapted the stopping criterion to the complexity of the problem which led to CPU times which are much larger in our case. On the other hand, in [1] older machines (VAX 875) were used making direct comparisons difficult. For the last three 'abz'-problems the results in [1] were obtained with run times up to 15h (compared with 8h in our case). Their results of these instances are the best from a comparison between SA, GA, tabu search and JSP-specific heuristics.

problem instance	jobs × mach.	optimal value resp. lower / upper bound	best value found	average best value	mean rel. error	average CPU secs	average no. of trials
swv01	20×10	1392 / 1418	1427	1428.0	0.026	47828	312320976
swv02	20×10	1475	1487	1489.7	0.010	43089	282676997
swv03	20×10	1369 / 1398	1422	1427.7	0.043	40684	250332641
swv04	20×10	1450 / 1483	1487	1490.3	0.028	44257	276227664
swv05	20×10	1421 / 1434	1449	1453.2	0.023	40045	254417756
swv06	20×15	1591 / 1696	1697	1703.0	0.070	112647	377949381
swv07	20×15	1446 / 1620	1627	1630.0	0.127	97504	337848245
swv08	20×15	1640 / 1770	1773	1776.5	0.083	56781	244833220
swv09	20×15	1604 / 1663	1665	1682.5	0.049	24474	109808770
swv10	20×15	1631 / 1773	1791	1794.5	0.100	44467	148570911
swv11	50×10	2983 / 3005	3075	3081.5	0.033	117454	247525101
swv12	50×10	2972 / 3038	3108	3115.0	0.048	124549	258701571
swv13	50×10	3104 / 3146	3177	3178.5	0.024	92756	222799699
swv14	50×10	2968	3010	3013.5	0.015	104088	247336666
swv15	50×10	2885 / 2940	3004	3004.0	0.041	161365	375564078

Table 2: Results from three runs for the difficult swv-problems from the JSPlib

Table 2 shows the best solutions found for the swv - problems from the JSPlib. These are best results from 2 - 3 runs. The algorithm used population size 20 and stopped after two generations without improvement. The local SA was allowed to spend 2 million trials without improvement before stopping. This led to CPU times of up to 44 h on Pentium 120/166.

The upper and lower bounds in Table 1 and 2 are taken from the JSPlib. They are obtained by different authors mostly with exact, highly JSP-specific methods. Only part of these results are published in detail, so that the CPU times are not all available.

The results show that if one is willing to spend enough time then our method 'SAGen' constantly finds very good solutions. When comparing its results to other methods, one has to keep in mind that SAGen is a very general method. It needs less problem specific knowledge than many other, in particular exact methods. Therefore, it is easier to set up and it can be applied to completely different problems with little adaptation. The price for this generality is a larger running time than is needed by many of the JSP-specific algorithms.

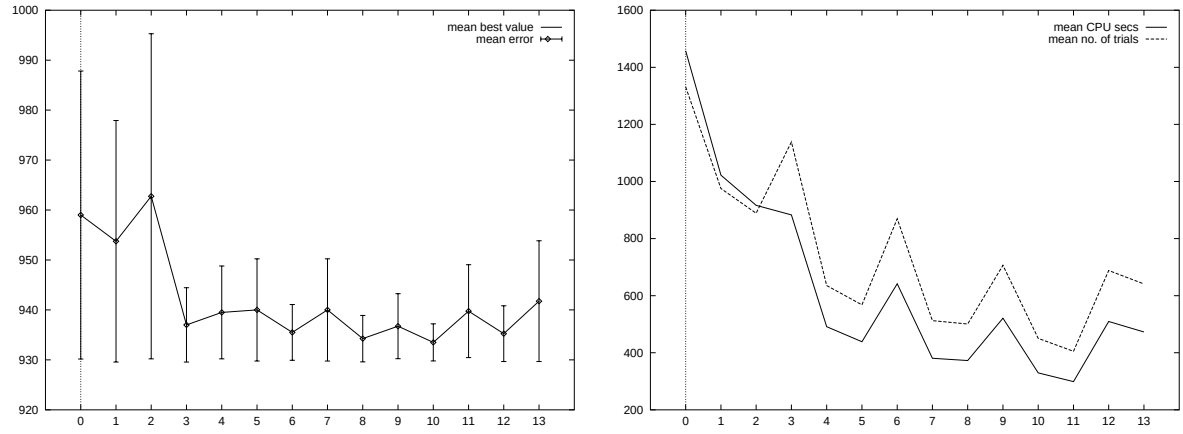

Figure 10: Mean best values, mean CPU times and total number of trials for the different parameter settings given in Table 3.

No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13
population size	10000	5000	3333	1000	500	333	100	50	33	10	5	3	10	1
no. of generations without improvement	1	2	3	1	2	3	1	2	3	1	2	3	0	0
no. of SA trials without improvement	100	100	100	1000	1000	1000	10000	10000	10000	100000	100000	100000	200000	2000000

Table 3: The number of the different parameter settings is used as x-value in Fig. 10.

Finally we compared the results of mt10, a 10×10 - problem for different population sizes. We chose the stopping criterion such that the amount of local improvement steps after the best solution was found was about constant for all settings. That means that the product $p * g * l$ is constant where p is the population size. In the following two figures x-values correspond to different parameter settings as explained in Table 3. The runs for $x = 12$ are the best results from 10 independent SA-runs that were stopped after $l = 200000$ trials without improvement (i.e. multi-start without crossover). $x = 13$ is from a single SA run that was allowed to spend $l = 2$ million unsuccessful trials before it was stopped. Fig. 7 shows that the best and most stable behaviour was observed for a small population with large local improvement (e.g. $x = 8, x = 10$). Also these settings were the fastest. The results are averages from four runs.

Though these values will differ with the size of the problem instances, it underlines our viewpoint that our algorithm is essentially a parallelized Simulated Annealing procedure that is allowed to exchange information via crossover now and then.

7 Conclusions

Convergence to an optimal solution is an important property for iterative search methods. Even if in practical implementations only a relatively small amount of solutions is visited and the best solution is

returned, an overall convergence of the method indicates that the results will improve when additional effort (computation time) is spent. If there is no convergence or convergence to suboptimal solutions as in the SA applied to the JSP, additional trials may lead to worse solutions.

As in such situations one cannot be sure to encounter the optimal solution during one run, a 'multistart' approach where SA starts several times from different initial solutions could improve the performance. A more sophisticated way to do this in parallel is the genetic algorithm. Here, several SA runs try to 'share' their findings by crossover. GA (in particular crossover operations) add a more global look to the local character of SA.

This method of combinig SA and GA could be refined by a more detailed statistical analysis of the cost landscape before and during the optimization. This would allow to adapt the temperature control and the GA operations more closely to the course of the optimization (see [23] where these ideas are applied to a multicriterial GA). Also one could add elements from tabu search to avoid cycling (though this does not seem to be a major problem here due to the asymmetry of the neighbourhoods). Parallelizing the implementation would allow larger populations to search a larger part of the solution space.

Presently, we are applying SAGen to the quadratic assignment problem, which in its standard form has symmetric neighbourhoods, but does not allow a natural crossover.

References

[1] Aarts, E. H. L., P. J. M. van Laarhoven, J. K. Lenstra and N. L. J. Ulder (1994), A Computational Study of Local Search Algorithms for Job Shop Scheduling. *ORSA Journal on Computing* **6**, 118–125

[2] Anily, S. and A. Federgruen (1987), Simulated Annealing Methods with General Acceptance Probabilities. *J. Appl. Prob.* **24**, 657–667 .

[3] Bierwirth, C.(1995), A generalized permutation approach to job shop scheduling with genetic algorithms. *OR Spektrum* **17**, 87-92.

[4] Blazewicz, J., W. Domschke and E. Pesch(1996), The job shop scheduling problem : Conventional and new solution techniques. *Europ. J. of Operations Research* **93** 1 -33.

[5] Brucker, P. (1995), *Scheduling Algorithms.* Springer Verlag, Berlin.

[6] Cerny, V. (1985), Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm. *JOTA* **45**, 41–51 .

[7] Faigle, U. and W. Kern(1989), Note on the Convergence of Simulated Annealing Algorithms. *SIAM J. of Control and Optimization* **29**, 153-159.

[8] Hajek, B. (1988), Cooling Schedules for Optimal Annealing. *Math. Op. Res.* **13**, 311–329.

[9] Holland, J. H.(1975), *Adaptation in Natural and Artificial Systems*, MIT Press, Ann Arbor.

[10] Karlin, S. and H. M. Taylor (1975), *A First Course in Stochastic Processes.* 2end edition, Academic Press.

[11] Kirkpatrick, S. and C.D. Jr. Gelatt (1983), Optimization by Simulated Annealing. *Science* **220**, No. 4598, 671–680.

[12] Kolonko, M. (1995), A Generalized Crossover Operation for Genetic Algorithms. *Complex Systems* **9**, 177-191.

[13] Kolonko, M. (1995), A Piecewise Markovian Model for Simulated Annealing with Stochastic Cooling Schedules. *J. Appl. Prob.***32**, 649-658.

[14] Kolonko, M. and M.T. Tran (1997), Convergence of Simulated Annealing with Feedback Temperature Schedules. *Probability in the Engin. and Informational Sciences* **11**, 279-304.

[15] Lundy, M. and A. Mees (1986), Convergence of an Annealing Algorithm. *Math. Progr.* **34**, 111–124 .

[16] Mahfoud, S. W. and D.E. Goldberg (1992), . A genetic algorithm for parallel simulated annealing. *Parallel Problem Solving from Nature 2*, R. Männer and B. Manderick (Eds.), Elsevier Science Publishers B.V., 301 - 310.

[17] Mattfeld, D. (1996), *Evolutionary Search and the Job Shop* Physica Verlag, Heidelberg.

[18] Mitra, D., F. Romeo and A. Sangiovanni-Vincentelli (1986), Convergence and Finite-Time Behaviour of Simulated Annealing. *Adv. Appl. Prob.* **18**, 747 – 771.

[19] Vaessens, R. J. M., E. H. L Aarts and J. K. Lenstra (1994), Job Shop Scheduling by Local Search. *Memorandum COSOR 94-05, Eindhoven University of Technology, February 1994.*

[20] van Laarhoven, P.J.M. and E.H.L. Aarts (1987), *Simulated Annealing: Theory and Applications.* D. Reidel Publ. Comp., Dordrecht NL.

[21] van Laarhoven, P.J.M. (1988), *Theoretical and computational aspects of simulated annealing* CWI Tract 51, Stichting Mathematisch Centrum, Amsterdam.

[22] van Laarhoven, P. J. M., E. H. L. Aarts and J. K. Lenstra (1992), Job Shop Scheduling by Simulated Annealing. *Operations Research* **40**, 113 – 125.

[23] Voget, S. and M. Kolonko (1997): Multidimensional Optimization Using Fuzzy Genetic Algorithms To appear in *Journal of Heuristics.*