



Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

P.J.M. van Laarhoven, E.H.L. Aarts, J.K. Lenstra

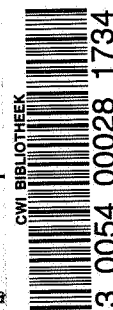
Job shop scheduling by simulated annealing

Department of Operations Research and System Theory

Report OS-R8809

July

Bibliotheek
Centrum voor Wiskunde en Informatica
Amsterdam



The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

Job Shop Scheduling by Simulated Annealing

Peter J.M. van Laarhoven
*Philips Research Laboratories,
P.O. Box 80.000, 5600 JA Eindhoven, The Netherlands*

Emile H.L. Aarts
*Philips Research Laboratories,
P.O. Box 80.000, 5600 JA Eindhoven, The Netherlands;
Department of Mathematics and Computing Science,
Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

Jan Karel Lenstra
*Centre for Mathematics and Computer Science,
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands;
Econometric Institute, Erasmus University,
P.O. Box 1738, 3000 DR Rotterdam, The Netherlands*

We describe an approximation algorithm for the problem of finding the minimum makespan in a job shop. The algorithm is based on simulated annealing, a generalization of the well-known iterative improvement approach to combinatorial optimization problems. The generalization implies the acceptance of cost-increasing transitions with a non-zero probability so as to avoid getting stuck in local minima.

We prove that our algorithm asymptotically converges in probability to a globally minimal solution, despite the fact that the Markov chains generated by the algorithm are generally not irreducible.

Computational experiments show that our algorithm can find shorter makespans than tailored heuristics for this problem, at the expense of larger computation times.

1980 Mathematics Subject Classification (1985 Revision): 90B35, 90C27.

Key Words & Phrases: job shop scheduling, probabilistic algorithms, simulated annealing.

Note: This paper has been submitted for publication.

In this paper we are concerned with a problem in machine scheduling, which is known as the *job shop scheduling problem* (Coffman 1976, French 1982). Informally, the problem can be described as follows. We are given a set of jobs and a set of machines. Each job consists of a chain of operations, each of which needs to be processed during an uninterrupted time period of a given length on a given machine. Each machine can process at most one operation at a time. The problem is to find a *schedule*, i.e. an allocation of

Report OS-R8809
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

the operations to time intervals on the machines, that has minimum length. A precise formulation of the problem is given in §1.

The job shop scheduling problem is among the hardest combinatorial optimization problems; not only is it NP -hard, but even among the members of the latter class it belongs to the more difficult ones (Lawler, Lenstra and Rinnooy Kan 1982). *Optimization algorithms* for job shop scheduling proceed by branch and bound, see for instance Lageweg, Lenstra and Rinnooy Kan (1977), Carlier and Pinson (1988). Most *approximation algorithms* use a *priority rule*, i.e. a rule for choosing an operation from a specified subset of as yet unscheduled operations. Adams, Balas and Zawack (1988) developed a shifting bottleneck procedure, which employs an ingenious combination of schedule construction and iterative improvement, guided by solutions to single-machine problems. In this paper we describe an alternative approach, known as *simulated annealing* (Kirkpatrick, Gelatt and Vecchi 1983, Černý 1985). Simulated annealing can be viewed as a generalization of iterative improvement and is described in more detail in §2.

In §3 we describe the application of simulated annealing to job shop scheduling. We prove asymptotic convergence of the algorithm to a globally minimal solution by showing that the neighbourhood structure is such that each ergodic set contains at least one global minimum. §4 contains the results of a computational study, in which simulated annealing is used to find approximate solutions to a large set of instances of the job shop scheduling problem. We show that simulated annealing performs slightly better than the shifting bottleneck procedure with respect to the length of the schedules returned by the algorithm, though computation times can be (very) long. We end this paper with some concluding remarks.

1 The Problem

We are given a set J of n jobs, a set M of m machines, and a set O of N operations. For each operation $v \in O$ there is a job $J_v \in J$ to which it belongs, a machine $M_v \in M$ on which it requires processing, and a processing time $t_v \in \mathbb{N}$. There is a binary relation \rightarrow on O that decomposes O into chains corresponding to the jobs; more specifically, if $v \rightarrow w$, then $J_v = J_w$ and there is no $x \notin \{v, w\}$ such that $v \rightarrow x$ or $x \rightarrow w$. The problem is to find a start time s_v for each operation $v \in O$ such that

$$\max_{v \in O} s_v + t_v, \tag{1}$$

is minimized subject to

$$s_v \geq 0, \forall v \in \mathcal{O} \quad (2)$$

$$s_w - s_v \geq t_v, \text{ if } v \rightarrow w, v, w \in \mathcal{O} \quad (3)$$

$$s_w - s_v \geq t_v \vee s_v - s_w \geq t_w, \text{ if } M_v = M_w, v, w \in \mathcal{O} \quad (4)$$

It is useful to represent the problem by the *disjunctive graph* model of Roy and Sussmann (1964). The disjunctive graph $G = (V, A, E)$ is defined as follows:

- $V = \mathcal{O} \cup \{0, N+1\}$, where 0 and $N+1$ are two fictitious operations; the weight of a vertex v is given by the processing time t_v ($t_0 = t_{N+1} = 0$).
- $A = \{(v, w) \mid v, w \in \mathcal{O}, v \rightarrow w\} \cup \{(0, w) \mid w \in \mathcal{O}, \nexists v \in \mathcal{O} : v \rightarrow w\} \cup \{(v, N+1) \mid v \in \mathcal{O}, \nexists w \in \mathcal{O} : v \rightarrow w\}$. Thus, A contains arcs connecting consecutive operations of the same job, as well as arcs from 0 to the first operation of each job and from the last operation of each job to $N+1$.
- $E = \{\{v, w\} \mid M_v = M_w\}$. Thus, edges in E connect operations to be processed by the same machine.

Figure 1 illustrates the disjunctive graph for a 3-job 3-machine instance, where each job consists of three operations.

For each pair of operations $v, w \in \mathcal{O}$ with $v \rightarrow w$, condition (3) is represented by an arc (v, w) in A . Similarly, for each pair of operations $v, w \in \mathcal{O}$ with $M_v = M_w$, the disjunctive constraint (4) is represented by an edge $\{v, w\}$ in E , and the two ways to settle the disjunction correspond to the two possible orientations of the edge. There is an obvious one-to-one correspondence between a set of choices in (4) that is overall feasible and an orientation of all the edges in E for which the resulting digraph is acyclic. The objective value (the makespan) of the corresponding solution is given by the length of a longest path in this digraph. Such a set of orientations decomposes \mathcal{O} into chains corresponding to the machines, i.e., it defines for each machine an ordering or permutation of the operations to be processed by that machine. Conversely, a set of machine permutations defines a set of orientations of the edges in E , though not necessarily one which results in an acyclic digraph. Since the longest path in a cyclic digraph has infinite length, we can now rephrase the problem as: find a set of machine permutations that minimizes the longest path in the resulting digraph. In §3 we use this formulation of the problem to find approximate solutions by simulated annealing.

2 Simulated Annealing

Ever since its introduction, independently by Kirkpatrick, Gelatt and Vecchi (1983) and Černý (1985), simulated annealing has been applied to many combinatorial optimization problems in such diverse areas as *computer-aided design of integrated circuits*, *image processing*, *code design* and *neural network theory*; for a review the reader is referred to Van Laarhoven and Aarts (1987). The algorithm is based on an intriguing combination of ideas from at first sight completely unrelated fields of science, viz. combinatorial optimization and statistical physics. On the one hand the algorithm can be considered as a generalization of the well-known iterative improvement approach to combinatorial optimization problems, on the other hand it can be viewed as an analogue of an algorithm used in statistical physics for computer simulation of the annealing of a solid to its *ground state*, i.e. the state with minimum energy. In this paper we mainly restrict ourselves to the first point of view; thus, we first briefly review iterative improvement.

Generally, a combinatorial optimization problem is a tuple (\mathcal{R}, C) , where \mathcal{R} is the *set of configurations* or *solutions* of the problem, and $C : \mathcal{R} \rightarrow \mathbb{R}$ the *cost function* (Papadimitriou and Steiglitz 1982). To be able to use iterative improvement we need a *neighbourhood structure* $\mathcal{N} : \mathcal{R} \rightarrow 2^{\mathcal{R}}$; thus, for each configuration i , $\mathcal{N}(i)$ is a subset of configurations, called the *neighbourhood* of i . Neighbourhoods are usually defined by first choosing a simple type of *transition* to obtain a new configuration from a given one and then defining the neighbourhood as the set of configurations that can be obtained from a given configuration in one transition.

Given the set of configurations, a cost function and a neighbourhood structure, we can define the iterative improvement algorithm as follows. The algorithm consists of a number of iterations. At the start of each iteration, a configuration i is given and a transition to a configuration $j \in \mathcal{N}(i)$ is generated. If $C(j) < C(i)$, the start configuration in the next iteration is j , otherwise it is i . If \mathcal{R} is finite and if the transitions are generated in some exhaustive enumerative way, then the algorithm terminates by definition in a local minimum. Unfortunately, a local minimum may differ considerably in cost from a global minimum. Simulated annealing can be viewed as an attempt to find *near-optimal* local minima by allowing the acceptance of cost-increasing transitions. More precisely, if i and $j \in \mathcal{N}(i)$ are the two configurations to choose from, then the algorithm continues with configuration j with a probability given by $\min\{1, \exp(-(C(j) - C(i))/c)\}$, where c is a positive *control parameter*, which is gradually decreased during the exe-

cution of the algorithm. c is the analogue of the temperature in the physical annealing process. Note that the aforementioned probability decreases for increasing values of $C(j) - C(i)$ and for decreasing values of c and that cost-decreasing transitions are always accepted.

For a fixed value of c , the configurations that are consecutively visited by the algorithm can be seen as a Markov chain with transition matrix $P = P(c)$ given by

$$P_{ij}(c) = \begin{cases} G_{ij}A_{ij}(c) & \text{if } j \neq i \\ 1 - \sum_{k=1}^{|\mathcal{R}|} G_{ik}A_{ik}(c) & \text{if } j = i, \end{cases} \quad (5)$$

where the generation probabilities G_{ij} are given by

$$G_{ij}(c) = \begin{cases} |\mathcal{N}(i)|^{-1} & \text{if } j \in \mathcal{N}(i) \\ 0 & \text{otherwise,} \end{cases} \quad (6)$$

and the acceptance probabilities A_{ij} by

$$A_{ij}(c) = \min \left\{ 1, \exp \left(\frac{-(C(j) - C(i))}{c} \right) \right\}. \quad (7)$$

The stationary distribution of this Markov chain exists and is given by [Folklore]:

$$q_i(c) = \frac{|\mathcal{N}(i)|A_{i_0i}(c)}{\sum_{j \in \mathcal{R}} |\mathcal{N}(j)|A_{i_0j}(c)}, \quad (8)$$

for some $i_0 \in \mathcal{R}_{opt}$, where \mathcal{R}_{opt} is the set of globally minimal configurations, provided the neighbourhoods are such that for each pair of configurations (i, j) there is a finite sequence of transitions leading from i to j . The latter condition is equivalent to the requirement that the matrix G be irreducible. It can readily be shown that

$$\lim_{c \downarrow 0} q_i(c) = \begin{cases} |\mathcal{R}_{opt}|^{-1} & \text{if } i \in \mathcal{R}_{opt} \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

Recalling that the stationary distribution of the Markov chain is defined as the probability distribution of the configurations after an infinite number of transitions, we conclude from (9) that the simulated annealing algorithm converges with probability 1 to a globally minimal configuration if the sequence of values of the control parameter converges to 0 and if the Markov chains generated at each value of c are of infinite length, provided the matrix

G is irreducible.

Unfortunately, the neighbourhood structure chosen for job shop scheduling in §3 is such that the corresponding matrix G is not irreducible. In that case, we can still prove asymptotic convergence provided the neighbourhoods are such that for each configuration i there is a finite sequence of transitions leading from i to some configuration $i_0 \in \mathcal{R}_{opt}$ (Van Laarhoven 1988). To do so, we use the fact that in every chain the recurrent configurations can be uniquely divided into irreducible ergodic sets S_1, S_2, \dots, S_T . In addition to the ergodic sets there is a set \mathcal{T} of transient configurations from which configurations in the ergodic sets can be reached (but not *vice versa*). Note that if the neighbourhoods satisfy the aforementioned condition, then each S_t contains at least one globally minimal configuration.

Now consider the sequence of configurations constituting the Markov chain associated with $P(c)$. There are two possibilities: either the Markov chain starts in a transient configuration or it does not. In the latter case, the configurations constituting the Markov chain all belong to the same irreducible ergodic set S_t and we can prove asymptotic convergence as before, with \mathcal{R} replaced by S_t . On the other hand, if the Markov chain starts in a transient configuration, it will eventually 'land' (Feller 1950) in an ergodic set S_t , $t \in \{1, \dots, T\}$, though it is not *a priori* known which one. The line of reasoning described above can then be applied again.

We can make the preceding arguments more precise by introducing the notion of a *stationary matrix* Q , whose elements q_{ij} are defined by

$$q_{ij} = \lim_{k \rightarrow \infty} \Pr\{X(k) = j | X(0) = i\}. \quad (10)$$

Using the results in chapter 15, sections 6-8 of Feller (1950), we obtain

$$q_{ij} = \begin{cases} 0 & \text{if } j \in \mathcal{T} \text{ or } i \in S_t, j \notin S_t, \text{ for some } t \in \{1, \dots, T\}, \\ \frac{A_{i_0j}(c)}{\sum_{l \in S_t} A_{i_0l}(c)} & \text{if } i, j \in S_t \text{ for some } t \in \{1, \dots, T\}, \\ x_{it} \frac{A_{i_0j}(c)}{\sum_{l \in S_t} A_{i_0l}(c)} & \text{if } i \in \mathcal{T}, j \in S_t \text{ for some } t \in \{1, \dots, T\}, \end{cases} \quad (11)$$

where x_{it} is the probability that the Markov chain, starting from the transient configuration i , eventually reaches the ergodic set S_t .

From (11) we obtain, for a recurrent configuration $j \in S_t$,

$$\begin{aligned}
0 &\leq \lim_{k \rightarrow \infty} \Pr\{X(k) = j\} = \sum_{i \in \mathcal{R}} \Pr\{X(0) = i\} \cdot q_{ij} \\
&= \left(\sum_{i \in T} \Pr\{X(0) = i\} \cdot x_{it} + \sum_{i \in S_t} \Pr\{X(0) = i\} \right) \cdot \frac{A_{i_0j}(c)}{\sum_{l \in S_t} A_{i_0l}(c)} \\
&\leq \frac{A_{i_0j}(c)}{\sum_{l \in S_t} A_{i_0l}(c)}. \tag{12}
\end{aligned}$$

Using (7) we find

$$\lim_{c \downarrow 0} \frac{A_{i_0j}(c)}{\sum_{l \in S_t} A_{i_0l}(c)} = 0, \tag{13}$$

if $j \in S_t$, $j \notin \mathcal{R}_{opt}$. Consequently, $\lim_{c \downarrow 0} (\lim_{k \rightarrow \infty} \Pr\{X(k) = j\}) = 0$ for any transient or non-globally minimal recurrent configuration j . In other words,

$$\lim_{c \downarrow 0} (\lim_{k \rightarrow \infty} \Pr\{X(k) \in \mathcal{R}'_{opt}\}) = 1, \tag{14}$$

where \mathcal{R}'_{opt} denotes the non-empty set of globally minimal recurrent configurations.

Some of the conditions for asymptotic convergence, as for instance the infinite length of Markov chains, cannot be met in practice. In any finite-time implementation, we therefore have to make a choice with respect to each of the following parameters:

- the length of the Markov chains,
- the initial value of the control parameter,
- the decrement rule of the control parameter,
- the final value of the control parameter.

Such a choice is usually referred to as a *cooling schedule* or *annealing schedule*. Our implementation uses the cooling schedule described in Aarts and Van Laarhoven (1985a, 1985b). This is a three-parameter schedule: the parameters χ_0 and ϵ_s determine the initial and final value of the control parameter, respectively, whereas the decrement rule depends on a parameter δ , hereinafter referred to as the distance parameter. The dependence is such that large values of δ correspond to a 'fast' decrement of c and small values

to a 'slow' decrement. Finally, for each instance the length of the Markov chains is set to the size of the largest neighbourhood.

Under some mild assumptions, it is possible to show that with the aforementioned cooling schedule the time-complexity of the simulated annealing algorithm is bounded by $O(\tau L \ln |\mathcal{R}|)$, where τ is the time involved in the generation and (possible) acceptance of a transition and L the size of the largest neighbourhood (the length of the Markov chains) (Aarts and Van Laarhoven 1985a). If one works out this bound for a particular combinatorial optimization problem, it is usually polynomial in the size of the problem. In those cases, we have a polynomial-time approximation algorithm. Such a result with respect to the *efficiency* of the algorithm is only worthwhile in combination with results on its *effectivity*, viz. on the difference in cost between solutions returned by the algorithm and globally minimal ones. From a theoretical point of view, very little is known about the effectivity of simulated annealing, but there are many empirical results; see for instance the extensive computational experiments of Johnson et al. (1987). For the job shop scheduling problem, we present an empirical analysis of the effectivity and efficiency of simulated annealing in §4, but first the application of simulated annealing to the job shop scheduling problem is discussed in more detail.

3 Simulated Annealing and Job Shop Scheduling

We recall from the previous section that in order to apply simulated annealing to any combinatorial optimization problem, we need a precise definition of configurations, a cost function and a neighbourhood structure. Furthermore, to prove asymptotic convergence we must show that the neighbourhood structure is such that for an arbitrary configuration i there exists at least one globally minimal configuration $i_0 \in \mathcal{R}_{opt}$ that can be reached from i in a finite number of transitions. Hereinafter, we discuss these items in more detail.

(i) Configurations

We recall from §1 that we can solve the job shop scheduling problem by considering sets of machine permutations and by determining, for such a set of permutations, the longest path in the digraph which results from giving the edges in the disjunctive graph the orientations determined by the permutations. We therefore define a configuration i of the problem as a set

$\Pi_i = \{\pi_{i1}, \dots, \pi_{im}\}$ of machine permutations. Consequently, the number of configurations is given by $\prod_{k=1}^m m_k!$, where m_k is the number of operations to be processed by machine k ($m_k = |\{v \in \mathcal{O} | M_v = k\}|$).

(ii) Cost function

For each configuration i we define the following two digraphs:

1. $D_i = (V, A \cup E_i)$, where

$$E_i = \{(v, w) | \{v, w\} \in E \text{ and } \pi_{ik}(v) = w \text{ for some } k \in \mathcal{M}\}. \quad (15)$$

2. $\bar{D}_i = (V, A \cup \bar{E}_i)$, where

$$\bar{E}_i = \{(v, w) | \{v, w\} \in E \text{ and } \pi_{ik}^l(v) = w \text{ for some } k \in \mathcal{M}, 1 \leq l \leq m\}. \quad (16)$$

In other words, \bar{D}_i is the digraph obtained from the disjunctive graph by giving the edges in E the orientations resulting from Π_i ; the digraph D_i can be obtained from \bar{D}_i by taking only those arcs from \bar{E}_i that connect successive operations on the same machine. It is well known that the longest paths in D_i and \bar{D}_i are identical; thus, the cost of a configuration i can be found by determining the length of a longest path from 0 to $N+1$ in D_i . To compute such a cost, we use a simple *labelling algorithm*, based on Bellman's equations (Bellman 1958), for solving the longest-path problem in a digraph. The time-complexity of this algorithm is proportional to the number of arcs in the graph. In our case, this number equals $|A| + |E_i| = (N+n) + (N-m)$; accordingly, the labelling algorithm takes $\mathcal{O}(N)$ time to compute the cost of a configuration.

(iii) Neighbourhood structure

A transition is generated by choosing vertices v and w , such that

1. v and w are successive operations on some machine k ,
2. $(v, w) \in E_i$ is a critical arc, i.e. (v, w) is on a longest path in D_i ,

and reversing the order in which v and w are processed on machine k . Thus, in the digraph D_i such a transition results in reversing the arc connecting v and w and replacing the arcs (u, v) and (w, x) by (u, w) and (v, x) , respectively, where $u = \pi_{ik}^{-1}(v)$ and $x = \pi_{ik}(w)$. Our choice is motivated by two facts:

- Reversing a critical arc in a digraph D_i can never lead to a cyclic digraph D_j (see Lemma 2).
- If the reversal of a non-critical arc in D_i leads to an acyclic graph D_j , a longest path q in D_j cannot be shorter than a longest path p in D_i (because D_j still contains the path p).

Thus, we exclude beforehand some non-cost-decreasing transitions and, in addition, all transitions that might result in a cyclic digraph. Consequently, the neighbourhood structure is such that the algorithm visits only digraphs corresponding to feasible solutions.

The neighbourhood of a configuration i is thus given by the set of acyclic digraphs that can be obtained by reversing a critical arc belonging to E_i in the graph D_i . Consequently, $|\mathcal{N}(i)| < \sum_{k=1}^m (m_k - 1) = N - m$.

(iv) Asymptotic convergence

It is not difficult to construct a problem instance containing pairs of configurations (i, j) for which there is no finite sequence of transitions leading from i to j (Van Laarhoven 1988). Thus, to prove asymptotic convergence, we must show that for each configuration i there is a finite sequence of transitions leading from i to some globally minimal configuration. In order to do so, we need two lemmas.

Lemma 1

Consider an arbitrary configuration i and an arbitrary global minimum $i_0 \in \mathcal{R}_{opt}$. If $i \notin \mathcal{R}_{opt}$, then the set $K_i(i_0)$ defined by

$$K_i(i_0) = \{e = (v, w) \in E_i \mid e \text{ is critical} \wedge (w, v) \in \bar{E}_{i_0}\} \quad (17)$$

is not empty.

Proof

The proof consists of two parts: first, we show that E_i always contains critical arcs, unless $i \in \mathcal{R}_{opt}$; next that there are always critical arcs in E_i that do not belong to \bar{E}_{i_0} unless again $i \in \mathcal{R}_{opt}$.

1. Suppose E_i contains no critical arcs, then all critical arcs belong to A . Consequently, a longest path consists of arcs connecting vertices corresponding to operations of the same job; accordingly, its length is given by the total processing time of that job. But this is a lower bound to the length of a longest path in any digraph D_j , hence $i \in \mathcal{R}_{opt}$.

2. Suppose that for all critical arcs e in E_i , we have that $e \in \overline{E}_{i_0}$. We then know that any longest path p in D_i is also a path q in \overline{D}_{i_0} . The length of a longest path r in \overline{D}_{i_0} is also the length of a longest path in D_{i_0} and because $i_0 \in \mathcal{R}_{opt}$, we have $length(r) \leq length(p)$. But by definition $length(r) \geq length(q) = length(p)$. Consequently, $length(p) = length(r)$ and $i \in \mathcal{R}_{opt}$. \square

Lemma 2

Suppose $e = (v, w) \in E_i$ is a critical arc of an acyclic digraph D_i . Let D_j be the digraph obtained from D_i by reversing the arc e in E_i . Then D_j is also acyclic.

Proof

Suppose D_j is cyclic. Because D_i is acyclic, the arc (w, v) is part of the cycle in D_j . Consequently, there is a path (v, x, y, \dots, w) in D_j . But this path can also be found in D_i and is clearly a longer path from v to w than the arc (v, w) . This contradicts the assumption that (v, w) is on a longest path in D_i . Hence, D_j is acyclic. \square

Given a configuration $i_0 \in \mathcal{R}_{opt}$, we define the following two sets for an arbitrary configuration i :

$$M_i(i_0) = \{e = (v, w) \in E_i \mid (w, v) \in \overline{E}_{i_0}\} \quad (18)$$

$$\overline{M}_i(i_0) = \{e = (v, w) \in \overline{E}_i \mid (w, v) \in \overline{E}_{i_0}\}. \quad (19)$$

In view of §2, the following theorem now ensures asymptotic convergence in probability to a globally minimal configuration.

Theorem 1

For each configuration $i \notin \mathcal{R}_{opt}$ it is possible to construct a finite sequence of transitions leading from i to a globally minimal configuration.

Proof

We choose an arbitrary configuration $i_0 \in \mathcal{R}_{opt}$ and construct a sequence of configurations $\{\lambda_0, \lambda_1, \dots\}$ as follows:

1. $\lambda_0 = i$
2. λ_{k+1} is obtained from λ_k by reversing an arc $e \in K_{\lambda_k}(i_0)$ in E_{λ_k} . According to Lemma 2, this can be done without creating a cycle in $D_{\lambda_{k+1}}$. Furthermore, this operation is of the aforementioned type of transition.

It can easily be seen that if $|\overline{M}_{\lambda_k}(i_0)| > 0$ then

$$|\overline{M}_{\lambda_{k+1}}(i_0)| = |\overline{M}_{\lambda_k}(i_0)| - 1. \quad (20)$$

Hence, for $k = |\overline{M}_i(i_0)|$, $|\overline{M}_{\lambda_k}(i_0)| = 0$. Using $K_i(i_0) \subseteq M_i(i_0) \subseteq \overline{M}_i(i_0)$, we find $K_{\lambda_k}(i_0) = \emptyset$ for $k = |\overline{M}_i(i_0)|$. According to Lemma 1, this implies $\lambda_k \in \mathcal{R}_{opt}$. \square

4 Computational Results

We have analysed the finite-time behaviour of the simulated annealing algorithm empirically by running the algorithm on a number of instances of the job shop scheduling problem, varying in size from six jobs on six machines to 30 jobs on ten machines. For all instances, the number of operations of each job equals the number of machines and each job has precisely one operation on each machine. In that case, the number of configurations of each instance is given by $(n!)^m$, the labelling algorithm takes $O(nm)$ time to compute the cost of a configuration, and the size of the neighbourhood of a configuration is bounded by $m(n-1)$.

FIS1, FIS2 and FIS3 are three problem instances due to Fisher and Thompson (1963), the forty instances in Table II are due to Lawrence (1984). FIS2 is a notorious 10-job 10-machine instance that has defied solution to optimality for more than 20 years. A couple of years ago, a solution with cost 930 was found after several hours of computation time (Lageweg 1988). This solution was only recently proved to be globally minimal by Carlier and Pinson (1988). For FIS1, FIS2 and FIS3, the processing times of the operations are randomly drawn and range from 1 to 10 (FIS1) or to 99 (FIS2 and FIS3) units of time. The sequence of machines for each job is such that lower-numbered machines tend to be used for earlier operations. For the Lawrence instances processing times are drawn from a uniform distribution on the interval $[5, 99]$; the sequence of machines for each job is random.

The performance of simulated annealing on these instances is reported in Table I for the Fisher-Thompson instances, and in Table II for the Lawrence instances. The averages in these tables are computed from five solutions, obtained by running the algorithm, controlled by the cooling schedule described in §2, five times on each instance and recording the best configuration encountered during each run (this need not necessarily be the final configuration). The probabilistic nature of the algorithm makes it necessary to carry out multiple runs on the same problem instance in order to get

meaningful results.

All results are obtained with the parameters χ_0 and ϵ_s set to 0.95 and 10^{-6} , respectively, and for different values of the distance parameter δ . Computation times are CPU times on a VAX-785.

From Tables I and II we can make the following observations:

- The quality of the average best solution returned by the algorithm improves considerably when δ is decreased. This is in accordance with the theory underlying the employed cooling schedule: smaller values of δ correspond to a better approximation of the asymptotic behaviour (Aarts and Van Laarhoven 1985a). Furthermore, the difference between the average best solution and a globally minimal one does not deteriorate significantly with increasing problem size. For the FIS2 instance, the five best solutions obtained with $\delta = 10^{-4}$ have cost values of 930 (twice), 934, 935 and 938, respectively. Thus, a globally minimal solution is found 2 out of 5 times, which is quite a remarkable result, considering the notoriety of this instance.
- As for computation times, we remark that the bound for the computation time given in §2 is $O((nm)^3 \ln n)$ ($L = O(nm)$, $|\mathcal{R}| = O((n!)^m)$ and $\tau = O(nm)$). Thus, for fixed m the bound is $O(n^3 \ln n)$, for fixed n it is $O(m^3)$. For the A, B and C instances in Table II, for which m is constant, the average computation time \bar{t} for $\delta = 0.01$ is approximately given by $\bar{t} = t_0 \cdot n^{2.215} \cdot \ln n$, for some constant t_0 ($\chi^2 = 1.00$); for the G, B and I instances, for which n is constant, the average computation time for $\delta = 0.01$ is approximately given by $\bar{t} = t_1 \cdot m^{2.406}$, for some constant t_1 ($\chi^2 = 1.00$). Thus, the observed computation times are in good accordance with the bound given in §2.

Table I also contains results obtained by repeated execution of the iterative improvement algorithm based on the same neighbourhood structure as simulated annealing. The initial configurations, to which the iterative improvement algorithm is applied, are randomly generated. The averages for iterative improvement are obtained from five macro-runs. Each macro-run consists of repeated execution of the iterative improvement algorithm for a large number of initial configurations and thus yields a large number of local minima. Execution of each macro-run is terminated as soon as the computation time exceeds the computation time of an average run of simulated annealing applied to the same problem instance with the distance parameter δ set to 10^{-3} (10^{-2} for FIS1); \bar{C}_{best} is the average of the best cost

value found during each macro-run.

We observe that repeated execution of iterative improvement is easily outperformed by simulated annealing for the two larger problems. The difference is significant: for FIS3, for instance, the average best solution obtained by simulated annealing is almost 11% better in cost than the one obtained by repeated execution of iterative improvement.

Table I and Table II also contain for each instance the cost value of the best solution obtained by Adams, Balas and Zawack (1988). Most values are obtained by a second heuristic, which embeds the aforementioned sliding bottleneck procedure and proceeds by partial enumeration of the solution space. The values for the instances F1, F5, G3 as well as for the D and H instances are obtained by the sliding bottleneck procedure only. The corresponding computation times are obtained by halving the CPU times in Adams, Balas and Zawack (1988), since these correspond to a VAX-780. Adams, Balas and Zawack show their approach to be superior to approaches based on priority dispatching rules: the typical improvement is reported to be between 4% and 10%.

Comparison of simulated annealing and the shifting bottleneck procedure leads to the following observations:

- For those instances for which Adams, Balas and Zawack do not find a globally minimal solution (mainly the A, B, C and I instances in Table II), the computation times of simulated annealing with $\delta = 0.1$ and of the heuristic of Adams, Balas and Zawack are of the same order of magnitude. In this case, the best solution found by Adams, Balas and Zawack is considerably better than the average best solution returned by simulated annealing and as good as the best solution found in five runs of simulated annealing.

Putting $\delta = 0.01$ makes simulated annealing of course much slower than the heuristic of Adams, Balas and Zawack, but now the best solution of Adams, Balas and Zawack is slightly worse than the average best solution of simulated annealing and considerably worse than the best solution in five runs of simulated annealing (the typical improvement is between 1 and 3%).

- For the instances for which the heuristic of Adams, Balas and Zawack finds a globally minimal solution, it outperforms simulated annealing: the latter algorithm also finds global minima, but takes much more computation time to do so.

Admittedly, the performance of the shifting bottleneck heuristic is liable to improve if it is allowed more computation time. Nevertheless, the results in Table I and Table II indicate that simulated annealing is a promising approach to job shop scheduling, as well as a robust one (cf. the small difference between \bar{C}_{best} and C_{best} in tables I and II for $\delta = 0.01$) and certainly superior to traditional approaches, such as procedures based on priority dispatching rules.

5 Conclusion

We have discussed a new approach to job shop scheduling based on a randomization version of iterative improvement. The probabilistic element of the algorithm (the acceptance of cost-increasing transitions with a non-zero probability) makes simulated annealing a significantly better approach than the iterative improvement approach on which it is based. The difference is especially pronounced for large problem instances. Furthermore, the algorithm has a potential for finding shorter makespans than other approximation algorithms discussed in the literature, but a sufficient amount of computation time to realize this potential is usually quite large. We consider this disadvantage to be compensated for by the simplicity of the algorithm, the fact that it is relatively easy to implement, requires no deep insight into the combinatorial structure of the problem instances, and, of course, by the high quality of the solutions it returns.

References

- AARTS, E.H.L AND P.J.M. VAN LAARHOVEN. 1985a. Statistical Cooling: A General Approach to Combinatorial Optimization Problems. *Philips J. of Research* 40, 193-226.
- AARTS, E.H.L AND P.J.M. VAN LAARHOVEN. 1985b. A New Polynomial Time Cooling Schedule. *Proc. IEEE Int. Conference on Computer-Aided Design*, Santa Clara, November 1985, 206-208.
- ADAMS, J., E. BALAS AND D. ZAWACK. 1988. The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science* 34, 391-401.
- BELLMAN, R.E. 1958. On a Routing Problem. *Quart. Appl. Math.* 16, 87-90.
- CARLIER, J. AND E. PINSON. 1988. An Algorithm for Solving the Job-Shop Problem. To appear in: *Management Science*.

- ČERNÝ, V. 1985. Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm. *J. Opt. Theory Appl.* **45**, 41-51.
- COFFMAN, E.G., ED. 1976. *Computer and Job-Shop Scheduling Theory*. Wiley, New York.
- FELLER, W. 1950. *An Introduction to Probability Theory and Applications*, vol. 1. Wiley, New York.
- FISHER, H. AND G.L. THOMPSON. 1963. Probabilistic Learning Combinations of Local Job-shop Scheduling Rules. In: J.F. MUTH AND G.L. THOMPSON, EDS. *Industrial Scheduling*. Prentice Hall, Englewood Cliffs, N.J., 225-251.
- FRENCH, S. 1982. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. Horwood, Chichester.
- JOHNSON, D.S., C.R. ARAGON, L.A. MCGEOCH AND C. SCHEVON. 1987. Optimization by Simulated Annealing: an Experimental Evaluation (Part I). Submitted to *Operations Research*.
- KIRKPATRICK, S., C.D. GELATT JR. AND M.P. VECCHI. 1983. Optimization by Simulated Annealing. *Science* **220**, 671-680.
- LAARHOVEN, P.J.M. VAN AND E.H.L. AARTS. 1987. *Simulated Annealing: Theory and Applications*. Reidel, Dordrecht.
- LAARHOVEN, P.J.M. VAN. 1988. Theoretical and Computational Aspects of Simulated Annealing. *Ph.D. Thesis*, Erasmus University, Rotterdam.
- LAGEWEG, B.J., J.K. LENSTRA AND A.H.G. RINNOOY KAN. 1977. Job-shop scheduling by implicit enumeration. *Management Science* **24**, 441-450.
- LAGEWEG, B.J. 1988. *private communication*.
- LAWLER, E.L., J.K. LENSTRA AND A.H.G. RINNOOY KAN. 1982. Recent Developments in Deterministic Sequencing and Scheduling: A Survey. In: M.A.H. DEMPSTER, J.K. LENSTRA AND A.H.G. RINNOOY KAN, EDS. *Deterministic and Stochastic Scheduling*. Reidel, Dordrecht, 35-73.
- LAWRENCE, S. 1984. Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques (supplement). Graduate School of Industrial Administration, Carnegie Mellon University.
- PAPADIMITRIOU, C.H. AND K. STEIGLITZ. 1982. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, N.J.

ROY, B. AND B. SUSSMANN. 1964. Les problèmes d'ordonnancement
avec constraints disjonctives. *Note DS No. 9 bis*, SEMA, Paris.

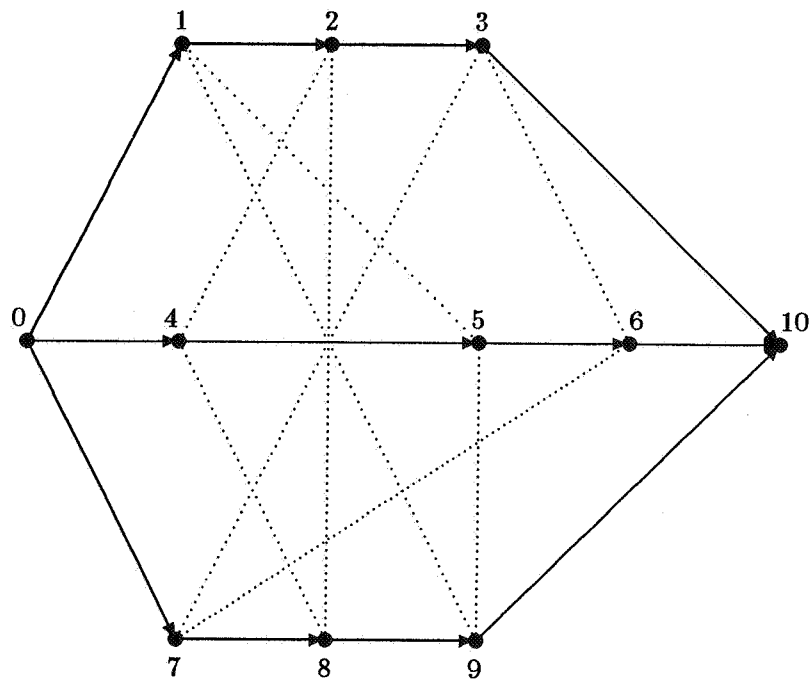


Figure 1. The disjunctive graph G of a 3-job 3-machine instance. Operations 1, 5 and 9 are processed by machine 1, operations 2, 4 and 8 by machine 2, and operations 3, 6 and 7 by machine 3. 0 and 10 are the fictitious initial and final operations, respectively. Thick arrows denote arcs in A , dotted lines edges in E .

Table I. Average cost of best solution (\bar{C}_{best}), average computation time in seconds (\bar{t}), standard deviations (σ_C and σ_t , respectively), cost of best solution (C_{best}), number of local minima per macro-run of iterative improvement (\bar{lm}) and % of average best cost value above globally minimal cost value. The results are obtained with the simulated annealing algorithm with different values of the distance parameter δ (upper part) and with repeated execution of the iterative improvement algorithm (lower part), respectively. The averages are obtained from five (macro-)runs. The table also includes for each instance the cost of the best solution found by Adams, Balas and Zawack (1988) (C_A) and the corresponding computation time (t_A). Provably globally minimal solutions are marked with an asterisk.

Problem	Simulated Annealing							ABZ	
	δ	\bar{C}_{best}	σ_C	%	\bar{t}	σ_t	C_{best}	C_A	t_A
6 machines, 6 jobs									
FIS1	10^{-1}	56.0	1.3	1.82	8	1	55*	55*	1
	10^{-2}	55.0*	0.0	0.00	52	8	55*		
10 machines, 10 jobs									
FIS2	10^{-1}	1039.6	15.1	11.78	113	13	1028	930*	476
	10^{-2}	985.8	22.1	6.00	779	61	951		
	10^{-3}	942.4	4.5	1.33	5945	180	937		
	10^{-4}	933.4	3.1	0.37	57772	2364	930*		
20 machines, 5 jobs									
FIS3	10^{-1}	1354.2	26.5	16.24	123	13	1325	1178	40
	10^{-2}	1229.0	33.6	5.49	848	93	1184		
	10^{-3}	1187.0	18.7	1.89	6840	389	1173		
	10^{-4}	1173.8	5.2	0.76	62759	7805	1165*		
Problem	Iterative Improvement								
		\bar{C}_{best}	σ_C	%	\bar{t}	σ_t	C_{best}	\bar{lm}	
FIS1		55.4	0.8	0.73	52	0	55*	803.2	
FIS2		1018.2	9.1	9.48	5945	0	1006	9441.2	
FIS3		1331.4	9.5	14.28	6841	0	1319	5221.0	

Table II. Average cost of best solution (\overline{C}_{best}), average computation time in seconds (t), standard deviations (σ_C and σ_t , respectively), and best cost of five runs (C_{best}). The averages are obtained from five runs of the simulated annealing algorithm with different values of the distance parameter δ . The table also includes for each instance the cost of the best solution found by Adams, Balas and Zawack (1988) (C_A) and the corresponding computation time (t_A). Provably globally minimal solutions are marked with an asterisk.

Table II

Problem	Simulated Annealing						ABZ	
	δ	\overline{C}_{best}	σ_C	\bar{t}	σ_t	C_{best}	C_A	t_A
10 machines, 10 jobs								
A1	1.0	1023.4	30.6	26	1.5	991	978	120
	0.1	981.0	17.3	110	17.1	956		
	0.01	966.2	10.1	686	83.3	956		
A2	1.0	861.0	41.2	23	3.7	797	787	96
	0.1	792.4	6.2	112	7.0	784		
	0.01	787.8	1.6	720	109.0	785		
A3	1.0	902.6	30.9	23	1.6	870	859	112
	0.1	872.2	12.4	112	22.1	861		
	0.01	861.2	0.4	673	69.0	861		
A4	1.0	950.0	54.5	24	5.3	904	860	120
	0.1	881.4	6.9	97	20.4	874		
	0.01	853.4	4.6	830	85.4	848		
A5	1.0	1021.6	26.2	30	1.9	994	914	144
	0.1	927.6	18.9	86	7.9	907		
	0.01	908.4	4.2	667	126.9	902		
10 machines, 15 jobs								
B1	1.0	1176.2	37.8	69	6.7	1133	1084	181
	0.1	1115.2	23.9	299	50.9	1085		
	0.01	1067.6	3.7	1991	341.1	1063		
B2	1.0	1125.6	35.6	65	3.6	1094	944	210
	0.1	977.4	19.5	307	36.5	963		
	0.01	944.2	4.7	2163	154.6	938		
B3	1.0	1155.8	64.2	63	5.6	1056	1032*	113
	0.1	1051.0	24.6	275	35.8	1032*		
	0.01	1032.0*	0.0	2093	89.7	1032*		
B4	1.0	1101.0	53.5	71	5.0	1032	976	217
	0.1	977.6	8.1	252	28.5	968		
	0.01	966.6	8.7	2098	406.0	952		
B5	1.0	1114.6	9.1	77	16.9	1103	1017	215
	0.1	1035.4	10.6	283	44.3	1017		
	0.01	1004.4	14.4	2133	374.5	992		
10 machines, 20 jobs								
C1	1.0	1397.0	69.1	139	16.0	1311	1224	372
	0.1	1268.0	9.7	555	81.7	1252		
	0.01	1219.0	2.0	4342	597.8	1218*		
C2	1.0	1434.2	40.0	139	6.4	1390	1291	419
	0.1	1311.6	12.7	651	82.9	1295		
	0.01	1273.6	5.2	4535	392.0	1269		
C3	1.0	1414.6	57.8	135	7.4	1335	1250	451
	0.1	1280.2	23.6	614	83.3	1246		
	0.01	1244.8	15.4	4354	349.8	1224		
C4	1.0	1387.4	47.0	138	14.1	1307	1239	446
	0.1	1260.4	35.4	581	24.0	1203		
	0.01	1226.4	6.5	4408	450.9	1218		
C5	1.0	1539.2	44.2	145	20.6	1492	1355*	276
	0.1	1393.6	9.6	605	84.4	1381		
	0.01	1355.0*	0.0	3956	428.2	1355*		
10 machines, 30 jobs								
D1	1.0	1882.2	39.3	442	79.3	1821	1784*	19
	0.1	1784.0*	0.0	1517	58.1	1784*		
D2	1.0	1921.4	35.3	492	66.2	1868	1850*	15
	0.1	1850.0*	0.0	1752	124.6	1850*		
D3	1.0	1761.8	12.2	433	40.4	1740	1719*	14
	0.1	1726.6	15.2	1880	130.8	1719*		
D4	1.0	1816.4	27.7	470	31.2	1788	1721*	11
	0.1	1775.6	38.4	1886	232.4	1721*		
D5	1.0	2011.2	81.3	434	34.6	1888*	1888*	11
	0.1	1890.0	4.0	1668	107.9	1888*		

Table II (cont'd)

Problem	Simulated Annealing						ABZ	
	δ	\overline{C}_{best}	σ_t	\bar{t}	σ_t	C_{best}	C_A	t_A
5 machines, 10 jobs								
F1	1.0	707.0	32.2	6	1.0	666*	666*	1
	0.1	666.0*	0.0	20	3.5	666*		
	0.01	666.0*	0.0	123	15.3	666*		
F2	1.0	719.0	20.0	6	1.0	685	669	6
	0.1	671.0	11.1	24	2.5	655*		
	0.01	663.0	4.9	117	19.0	655*		
F3	1.0	689.6	22.4	5	0.9	664	605	32
	0.1	635.6	9.5	24	3.8	626		
	0.01	617.6	8.5	129	12.6	606		
F4	1.0	665.4	56.9	6	0.9	608	593	23
	0.1	617.2	20.5	21	5.2	594		
	0.01	593.8	2.1	121	15.9	590		
F5	1.0	594.4	2.8	5	0.6	593*	593*	0
	0.1	593.0*	0.0	19	4.2	593*		
	0.01	593.0*	0.0	118	15.3	593*		
5 machines, 15 jobs								
G1	1.0	937.2	13.7	16	2.8	926*	926*	1
	0.1	926.0*	0.0	52	5.8	926*		
	0.01	926.0*	0.0	286	32.1	926*		
G2	1.0	948.6	44.1	15	1.6	911	890*	1
	0.1	900.6	8.5	66	15.2	890*		
	0.01	890.0*	0.0	376	48.3	890*		
G3	1.0	905.8	34.2	16	0.4	863*	863*	2
	0.1	863.0*	0.0	55	7.3	863*		
	0.01	863.0*	0.0	292	40.8	863*		
G4	1.0	965.2	20.0	13	1.0	951*	951*	0
	0.1	951.0*	0.0	47	5.9	951*		
	0.01	951.0*	0.0	283	25.9	951*		
G5	1.0	958.0*	0.0	14	1.6	958*	958*	0
	0.1	958.0*	0.0	45	2.0	958*		
	0.01	958.0*	0.0	243	42.3	958*		
5 machines, 20 jobs								
H1	1.0	1229.6	14.7	32	3.9	1222*	1222*	1
	0.1	1222.0*	0.0	108	17.2	1222*		
	0.01	1222.0*	0.0	627	18.4	1222*		
H2	1.0	1042.8	7.6	34	3.9	1039*	1039*	0
	0.1	1061.2	44.4	116	11.9	1039*		
	0.01	1039.0*	0.0	655	30.7	1039*		
H3	1.0	1154.6	9.2	32	2.5	1150*	1150*	1
	0.1	1150.0*	0.0	118	18.0	1150*		
	0.01	1150.0*	0.0	564	85.9	1150*		
H4	1.0	1292.0*	0.0	27	1.7	1292*	1292*	0
	0.1	1292.0*	0.0	93	20.6	1292*		
	0.01	1292.0*	0.0	462	21.8	1292*		
H5	1.0	1299.8	77.4	34	5.3	1207*	1207*	2
	0.1	1252.5	18.8	126	16.2	1233		
	0.01	1207.0*	0.0	736	26.3	1207*		
15 machines, 15 jobs								
I1	1.0	1487.6	40.4	152	6.4	1450	1305	268
	0.1	1343.2	30.2	785	80.6	1297		
	0.01	1300.0	7.8	5346	399.8	1293		
I2	1.0	1580.2	38.3	173	12.1	1523	1423	419
	0.1	1479.4	28.8	757	94.6	1457		
	0.01	1442.4	5.7	5287	688.5	1433		
I3	1.0	1422.2	28.3	173	25.3	1376	1255	540
	0.1	1303.4	30.5	713	90.8	1263		
	0.01	1227.2	8.2	5480	614.8	1215		
I4	1.0	1408.6	44.4	186	24.6	1348	1273	335
	0.1	1305.4	27.5	673	75.1	1264		
	0.01	1258.2	5.2	5766	800.3	1248		
I5	1.0	1399.8	60.2	162	8.8	1318	1269	450
	0.1	1282.2	15.5	745	68.4	1254		
	0.01	1247.4	9.9	5373	1066.4	1234		