Simulated annealing: An introduction

E.H.L. Aarts
Philips Research Laboratories
P.O. Box 80.000
5600 JA Eindhoven
The Netherlands
and
Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven
The Netherlands

P.J.M. van Laarhoven
Center for Quantitative Methods,
Nederlandse Philips Bedrijven B.V.,
P.O. Box 218,
5600 MD Eindhoven
The Netherlands

Simulated annealing is a general approach for approximately solving large combinatorial optimization problems. 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 viewed as an analogue of an algorithm used in statistical physics for computer simulation of the annealing of a solid to its minimum-energy state, on the other hand it can be considered as a generalization of the well-known iterative improvement approach to combinatorial optimization problems.

In this introductory paper we give a mathematical description of the simulated annealing algorithm and discuss its behaviour from both a theoretical and a practical point of view. The latter is illustrated by applying the algorithm to the travelling salesman problem.

This paper was written to familiarize the readers of *Statistica Neerlandica* with simulated annealing. It is a summary of papers written earlier by the authors and does not contain any new material.

Key Words and Phrases: combinatorial optimization, travelling salesman problem.

1. Introduction

In this paper we are concerned with *combinatorial optimization* (Lawler, 1976; Papadimitriou and Steiglitz, 1982), i.e. the search for optima of functions of discrete variables. Combinatorial optimization problems are nowadays ubiquitous in such diverse areas as, for example, computer science, engineering and operations research.

When dealing with a combinatorial optimization problem Π, one can try to construct either an *optimization algorithm*, i.e. an algorithm that returns a globally optimal solution for every instance of Π, or an *approximation algorithm*, i.e. an algorithm that "merely" returns a solution for each instance (GAREY and JOHNSON, 1979). Preferably, the latter algorithm should have the property that for all instances the returned solution is "close" to a globally optimal solution.

Unfortunately, many combinatorial optimization problems are *NP-hard* (GAREY and JOHNSON, 1979), i.e. they belong to the class of problems for which it is commonly believed that no algorithm can be constructed that solves each instance of the problem to optimality in a running time bounded by a polynomial function of the size of such an instance. Consequently, solving large problem instances to optimality is impracticable. For that reason many combinatorial optimization problems are tackled by constructing approximation rather than optimization algorithms. In that case the goal is to construct an approximation algorithm that runs in low-order polynomial time and has the property that final solutions are "close" to globally optimal ones. For many combinatorial optimization problems such algorithms are nowadays available, but usually they suffer from the fact that they are only applicable to the particular problem they are designed for (in this connection, the notion *tailored algorithm* is used). As soon as a new combinatorial optimization problem arises, a new algorithm has to be constructed. Generally applicable approximation algorithms, that are able to find near-optimal solutions for a wide variety of combinatorial optimization problems, are rare. The *simulated annealing algorithm*, the subject of this paper, is such an algorithm.

In the few years since its introduction, independently by KIRKPATRICK, GELATT and VECCHI (1983) and CERNY (1985), simulated annealing has attracted wide attention and many papers on the theory and applications of the algorithm have appeared. In this introductory paper we first give an informal description of the algorithm (Section 2). In Section 3 the algorithm is mathematically described in terms of Markov chains. Necessary and sufficient conditions are derived to ensure that asymptotically the algorithm finds a globally optimal solution with probability 1. However, these conditions cannot be satisfied in finite time, so that one has to find values for certain parameters of the algorithm (referred to as a *cooling schedule*) that ensure that in finite time near-optimal solutions are returned. Such cooling schedules are the subject of Section 4; in particular, we discuss a cooling schedule which achieves near-optimality of final solutions by imitating closely the aforementioned asymptotic behaviour.

In Section 5 we discuss computational experience by presenting results which are obtained by running simulated annealing on a large set of instances of the *travelling salesman* problem and pitting it against other approximation algorithms for this problem.

The paper is ended with some conclusions and remarks.

2. The simulated annealing algorithm

We recall that we are interested in solving combinatorial optimization problems. Such a problem can be formalized as a pair $(\mathfrak{R}, C)$, where $\mathfrak{R}$ is the finite *set of configurations* (also called *configuration space or solution space*) and C a *cost function,* $C:\mathfrak{R} \rightarrow \mathbb{R}$, which assigns a real number to each configuration. For convenience, only minimization problems are considered (which can be done without loss of generality). Thus, the problem is to find a configuration $i_0 \in \mathfrak{R}$, for which C takes its minimum value, i.e. such that

$$C_{opt} = C(i_0) = \min_{i \in \mathfrak{R}} C(i),$$

where C_{opt} denotes the minimum cost value.

The simulated annealing algorithm can be viewed as a generalization of the well-known *iterative improvement* approach to combinatorial optimization problems. The application of an iterative improvement algorithm presupposes the definition of configurations, a cost function and a *generation mechanism,* i.e. a simple prescription to generate a *transition* from one configuration to another. The generation mechanism defines a *neighbourhood* $\mathfrak{R}_i$ for each configuration i, consisting of all configurations that can be reached from i in a single transition. Iterative improvement is therefore also known as *neighbourhood search* or *local search* (Papadimitriou and Steiglitz, 1982). The iterative improvement algorithm can now be formulated as follows. We start at a given configuration and search for a configuration in its neighbourhood with lower cost. If such an improved configuration exists, we adopt it and repeat the neighbourhood search from the new configuration. The algorithm terminates when a configuration is obtained whose cost is no worse than any of its neighbours.

Iterative improvement algorithms possess the following disadvantages:

- By definition, they terminate in the first local minimum they encounter; generally, such a local minimum deviates substantially in cost from a global minimum.
- The returned local minimum depends on the initial configuration, for the choice of which generally no guidelines are available.
- In general, the worst-case time complexity of an iterative improvement algorithm is unknown.

It should be clear, however, that iterative improvement does have the advantage of being generally applicable: the main ingredients, viz. configurations, a cost function and a neighbourhood structure, are usually easy to define. Besides, though upper bounds for computation times are missing, a single run of an iterative improvement algorithm can **on the average** be executed in a small amount of computation time. Because of this, it is customary to execute the algorithm for a large number of initial configurations, drawn independently from the configuration space $\mathfrak{R}$. In this way, the first two of the disadvantages mentioned can be overcome.

We recall that the reason why an iterative improvement algorithm terminates in the first local minimum it encounters is that only transitions corresponding to a **decrease** in cost are accepted by the algorithm. Alternatively, we might think of an algorithm which also accepts, in some limited way, transitions corresponding to an **increase** in cost. Simulated annealing is an example of the latter approach: in addition to cost-decreasing transitions, cost-increasing transitions are accepted with a non-zero probability, which gradually decreases as the algorithm continues its execution. The precise way in which cost-increasing transitions are accepted is suggested by the analogy between the following two problems: that of finding the *ground state* of a solid and that of finding a globally minimal configuration in a combinatorial optimization problem. In condensed matter physics, *annealing* denotes a physical process by which the ground state of a solid can be found. The simulated annealing algorithm takes its name from the fact that it is based on an algorithm to simulate (parts of) the physical annealing process. Starting off at a given value of the temperature, the annealing process can be described as follows. At each temperature value T, the solid is allowed to reach *thermal equilibrium.* In thermal equilibrium the probability of occurrence of a state i with energy E_i is given by the *Boltzmann distribution:*

$$\mathbf{P}_T\{\text{state} = i\} = \frac{1}{Z(T)} \exp\left(-\frac{E_i}{k_B T}\right), \qquad (1)$$

where $Z(T)$ is the *partition function* and k_B the *Boltzmann constant.* As the temperature decreases, the Boltzmann distribution concentrates on the low-energy states and finally, when the temperature approaches zero, only the minimum-energy states have a non-zero probability of occurrence. However, if the cooling is too rapid, i.e. if the solid is not allowed to reach thermal equilibrium at each temperature value, defects can be 'frozen' into the solid resulting in metastable amorphous structures instead of the low-energy crystalline structure.

There is some similarity between a solid and a combinatorial optimization problem: in both cases there are many degrees of freedom (the states of the solid, the configurations in an optimization problem) and in both cases some global quantity has to be minimized (the energy of the solid, the cost function in combinatorial optimization). The observation of this analogy is the first step in the construction of the simulated annealing algorithm, the next step is to extend this analogy to the *Metropolis algorithm.*

To simulate the evolution to thermal equilibrium of a solid, METROPOLIS, ROSENBLUTH, ROSENBLUTH, TELLER and TELLER (1953) proposed a *Monte Carlo method,* which generates sequences of states of the solid in the following way. Given the current state of the solid, characterized by the positions of its particles, a small, randomly generated, perturbation is applied, i.e. a small displacement of a randomly chosen particle. If the perturbation results in a lower energy state of the solid, then the process is continued with the new state. If $\Delta E \geqslant 0$, then the probability of accepting the perturbed state is given by $\exp(-\Delta E / k_B T)$. This rule for accepting new states is referred to as the *Metropolis criterion.* Guided by this criterion, the solid eventually evolves into thermal equilibrium, i.e. after a large number of perturbations, using the aforementioned acceptance criterion, the probability distribution of the states approaches the Boltzmann distribution, given by (1).

The Metropolis algorithm can also be used to generate sequences of configurations of a combinatorial optimization problem. In that case, the configurations assume the role of the states of a solid while the cost function C and the *control parameter* c assume the roles of energy and temperature, respectively. The simulated annealing algorithm can be viewed as a sequence of Metropolis algorithms evaluated at decreasing values of the control parameter. It can thus be described as follows. Initially, the control parameter is given a large value and a sequence of trials is generated using the same generation mechanism as in the iterative improvement algorithm. Thus, in each trial, a configuration j is generated by choosing at random an element from the neighbourhood of the current configuration i. This corresponds to the small perturbation in the Metropolis algorithm. Let $\Delta C_{ij} = C(j) - C(i)$, then the probability of configuration j being the next configuration in the sequence equals 1, if $\Delta C_{ij} \leqslant 0$, and $\exp(-\Delta C_{ij}/c)$, if $\Delta C_{ij} > 0$ (the Metropolis criterion). Thus, there is a non-zero probability of accepting a configuration with higher cost than the current configuration. This sequence of trials is continued until equilibrium is reached, i.e. until the probability distribution of the configurations equals the Boltzmann distribution, now given by

$$\mathbb{P}_c\{\text{configuration} = i\} \overset{def}{=} q_i(c) = \frac{1}{Q(c)} \exp(-\frac{C(i)}{c}),$$

where $Q(c)$ is a normalization constant depending on the control parameter c, being the equivalent of the aforementioned partition function.

The control parameter is lowered in steps until it approaches 0, with the system being allowed to approach equilibrium for each step by generating a sequence of trials in the previously described way. After termination, the final 'frozen' configuration is taken as the solution of the problem at hand. Thus, as with iterative improvement, we have again a generally applicable approximation algorithm: configurations, a cost function and a neighbourhood structure are the only prerequisites to be able to apply simulated annealing.

Comparing iterative improvement and simulated annealing, it is apparent that the situation where the control parameter in the simulated annealing algorithm is set to 0 corresponds to a version of iterative improvement (it is not iterative improvement *per se,* because in an iterative improvement approach the neighbouring configurations are not necessarily examined in random order). Furthermore, simulated annealing is a generalization of iterative improvement in that it accepts, with non-zero but gradually decreasing probability, deteriorations in cost. In Section 5, when discussing computational experience with simulated annealing, we compare the two algorithms more extensively.

3. Mathematical model of the algorithm

Simulated annealing can mathematically be described by means of a *Markov chain:* a sequence of trials, where the outcome of each trial only depends on the outcome of the previous trial (Feller, 1950). In the case of simulated annealing, trials correspond to transitions and outcomes to configurations. A Markov chain is described by means of a set of *conditional probabilities* $P_{ij}(k)$ for each pair of outcomes (i,j); $P_{ij}(k)$ is the probability that the outcome of the k-th trial is j, given that the outcome of the $(k-1)$-th trial is i. let $X(k)$ denote the outcome of the k-th trial, then we have:

$$P_{ij}(k) = \mathbb{P}\{X(k) = j | X(k-1) = i\}.$$

If the conditional probabilities do not depend on k, we write P_{ij} instead of $P_{ij}(k)$. The corresponding Markov chain is then called *homogeneous,* otherwise it is called *inhomogeneous.*

Returning to simulated annealing, we note that $P_{ij}(k)$ denotes the probability that the k-th transition is a transition from configuration i to configuration j and that $X(k)$ is the configuration obtained after k transitions. In view of this, $P_{ij}(k)$ is called the *transition probability* and the $|\mathfrak{R}|\times|\mathfrak{R}|$- matrix $P(k)$ the *transition matrix.*

The transition probabilities depend on the value of the control parameter c, the analogue of the temperature in the physical annealing process. Thus, if c is kept constant, the corresponding Markov chain is homogeneous and the elements of the corresponding transition matrix $P=P(c)$ are given by:

$$P_{ij}(c) = \begin{cases} G_{ij}(c)A_{ij}(c) & \forall j \neq i \\ 1 - \sum_{l=1, l\neq i}^{|\mathfrak{R}|} G_{il}(c)A_{il}(c) & j=i, \end{cases}$$

where the *generation probability* $G_{ij}(c)$ denotes the conditional probability of generating configuration j, given that the current configuration is i, and the *acceptance probability* $A_{ij}(c)$ denotes the conditional probability of accepting the transition from configuration i to configuration j. The corresponding matrices $G(c)$ and $A(c)$ are called the *generation* and *acceptance matrices,* respectively.

A frequently used choice of the generation and acceptance matrices, which closely follows the analogy with the physical annealing process, is given by the following relations:

$$G_{ij}(c) = G_{ij} = \begin{cases} |\mathfrak{R}_i|^{-1} & \text{if } j \in \mathfrak{R}_i \\ 0 & \text{elsewhere,} \end{cases} \tag{2}$$

i.e. G is independent of c and corresponds to a uniform distribution on the neighbourhoods, and

$$A_{ij}(c) = \begin{cases} \exp\left[-\frac{C(j)-C(i)}{c}\right] & \text{if } C(j)>C(i) \\ 1 & \text{if } C(j)\leqslant C(i), \end{cases} \tag{3}$$

i.e. the acceptance probability $A_{ij}(c)$ is chosen as the Metropolis criterion. Hereinafter, we refer to these relations as the standard choice.

As pointed out before, the control parameter c is decreased during the course of the algorithm. We distinguish two algorithms, according to the way in which this decrement is carried out:

- the *inhomogeneous algorithm*, in which c is decreased after each transition, and which can therefore be described by a single inhomogeneous Markov chain;
- the *homogeneous algorithm*, in which c is decreased after a number of transitions, and which can therefore be described by a sequence of homogeneous Markov chains, each generated at a fixed value of c.

Both algorithms find a globally minimal configuration if, after a (possibly large) number of transitions, say K, the following relation holds:

$$\mathbf{P}\{X(K)\in\mathfrak{R}_{opt}\} = 1, \qquad (4)$$

where $\mathfrak{R}_{opt}$ is the set of globally minimal configurations. In the next subsections we briefly discuss the asymptotic convergence to (4) for both algorithms.

3.1. Asymptotic convergence of the homogeneous algorithm

The following theorem provides sufficient conditions on $G(c)$ and $A(c)$ to ensure that if the Markov chains are all of infinite length and if the limit $c\downarrow 0$ is taken, then the algorithm converges in probability to a globally minimal configuration.

THEOREM 1. (FOLKLORE) *Suppose the following conditions on the matrices $G(c)$ and $A(c)$ are satisfied:*

(1) $\forall c>0,\ \forall i,j\in\mathfrak{R}\ \exists p\geq 1, \exists\lambda_0,\lambda_1,\ldots,\lambda_p\in\mathfrak{R}\ (i=\lambda_0, j=\lambda_p)$:

$$G_{\lambda_k\lambda_{k+1}}(c)>0,\ k=0,1,\ldots,p-1;$$

(2) $$\forall c>0,\ \forall i,j\in\mathfrak{R}: G_{ji}(c) = G_{ij}(c);$$

(3) $$\forall c>0,\ \forall i,j,k\in\mathfrak{R}:$$

$$C(i)\leq C(j)\leq C(k) \Rightarrow A_{ik}(c) = A_{ij}(c)A_{jk}(c);$$

(4) $\forall c>0,\ \forall i,j\in\mathfrak{R}: C(i)\geq C(j) \Rightarrow A_{ij}(c) = 1;$

(5) $\forall c>0,\ \forall i,j\in\mathfrak{R}: C(i)<C(j) \Rightarrow 0<A_{ij}(c)<1;$

(6) $\forall i,j\in\mathfrak{R}: C(i)<C(j) \Rightarrow \lim_{c\downarrow 0} A_{ij}(c) = 0.$

Then

$$\lim_{k\to\infty}\mathbf{P}_c\{X(k) = i\} \overset{def}{=} q_i(c) = \frac{A_{i_0 i}(c)}{\sum_{j\in\mathfrak{R}} A_{i_0 j}(c)} \qquad (5)$$

and

$$\lim_{c\downarrow 0} q_i(c) = \begin{cases} |\mathfrak{R}_{opt}|^{-1} & \text{if } i\in\mathfrak{R}_{opt} \\ 0 & \text{elsewhere.} \end{cases} \tag{6}$$

Essential to the proof of Theorem 1 is the fact that the aforementioned conditions ensure that for $c>0$ the *stationary distribution* of the homogeneous Markov chain with transition matrix $P(c)$ exists. The stationary distribution of a homogeneous Markov chain is defined as the vector q whose i-th component is given by (FELLER, 1950)

$$q_i = \lim_{k\to\infty} \mathbb{P}\{X(k) = i | X(0) = j\}, \tag{7}$$

for an arbitrary j. If $\mathbf{q}$ exists, it is straightforward to show that (cf. (5))

$$q_i = \lim_{k\to\infty} \mathbb{P}\{X(k) = i\}.$$

Thus, the stationary distribution is the probability distribution of the configurations after an infinite number of transitions.

Using the conditions (1)-(5) of Theorem 1 and Theorem 2 - Corollary II in chapter 15 of (FELLER, 1950), it can be shown that $\mathbf{q}$ (c) $(c>0)$ is given by

$$\forall i\in\mathfrak{R} : q_i(c) = \frac{A_{i_0 i}(c)}{\Sigma_{j\in\mathfrak{R}} A_{i_0 j}(c)}, \tag{8}$$

for an arbitrary $i_0\in\mathfrak{R}_{opt}$. Using (8) and conditions (4) and (6), it is then straightforward to show that (6) holds.

A few remarks on this result are in order.

- Combination of (5) and (6) yields:

$$\lim_{c\downarrow 0}(\lim_{k\to\infty} \mathbb{P}_c\{X(k)\in\mathfrak{R}_{opt}\}) = 1. \tag{9}$$

  Thus, Theorem 1 states that, if the algorithm is allowed unlimited computation time ((7) implies that each individual Markov chain is of infinite length), it finds a global minimum with probability 1. In itself, this is not a very satisfactory result; convergence can be shown to hold for the simplest algorithm imaginable: a complete enumeration of all configurations. Nevertheless, as we show in Section 4, (9) provides guidelines for a proper finite-time implementation of the algorithm.
- Condition (1) of Theorem 1 implies that the Markov chains generated by the simulated annealing algorithm are *irreducible* for positive values of c (FELLER, 1950; VAN LAARHOVEN and AARTS, 1987).
- Condition (2) can be replaced by (LUNDY and MEES, 1986)

$$(2)' \quad \forall i\in\mathfrak{R} : G_{ij} = \begin{cases} |\mathfrak{R}_i|^{-1} & \text{if } j\in\mathfrak{R}_i \\ 0 & \text{elsewhere.} \end{cases} \tag{10}$$

  In this case, the stationary distribution is slightly different from (8), but (6) is still valid. In other words, it suffices to demand that G is either symmetric or given by the uniform distribution on the neighbourhoods.
- It is easily verified that for simulated annealing with the standard choice

for the generation and acceptance matrices conditions (3)-(6) are satisfied. The generation matrix is given by (10); thus, condition (2)′ is also satisfied. Consequently, one should only verify that condition (1) is satisfied. If this is not the case, i.e. if it is not possible for an arbitrary pair of configurations (i,j) to construct a finite sequence of transitions leading from i to j, it is still possible to prove asymptotic convergence to a globally minimal configuration if the following condition is satisfied (VAN LAARHOVEN, 1988):

$$(1)' \quad \forall i \in \mathfrak{R}\ \exists i_0 \in \mathfrak{R}_{opt},\ p \geqslant 1,\ \lambda_0, \lambda_1, \ldots, \lambda_p \in \mathfrak{R}\ (i = \lambda_0, i_0 = \lambda_p): \\ G_{\lambda_k \lambda_{k+1}}(c) > 0,\ k = 0, 1, \ldots, p-1,$$

i.e. if for an arbitrary configuration i it is possible to construct a finite sequence of transitions leading from i to a globally minimal configuration i_0.

3.2. *Asymptotic convergence of the inhomogeneous algorithm*

We now discuss briefly necessary and sufficient conditions for the inhomogeneous algorithm to converge to a global minimum. We recall that the inhomogeneous algorithm is described by an inhomogeneous Markov chain, whose transition matrix $P(k)$ $(k = 1,2,\ldots)$ is given by

$$P_{ij}(k) = \begin{cases} G_{ij}(c_k) A_{ij}(c_k) & \forall j \neq i \\ 1 - \sum_{l=1, l \neq i}^{|\mathfrak{R}|} G_{il}(c_k) A_{il}(c_k) & j = i, \end{cases} \tag{11}$$

where the sequence $\{c_k\}$, $k = 1,2,\ldots$ denotes the sequence of values of the control parameter. A number of authors derive sufficient conditions for asymptotic convergence of the inhomogeneous algorithm to a global minimum, notably Geman and GEMAN (1984), ANILY and FEDERGRUEN (1985), MITRA, ROMEO and SANGIOVANNI-VINCENTELLI (1986) and GELFAND and MITTER (1985). These derivations are all based on ergodicity theorems for inhomogeneous Markov chains (SENETA, 1981).

Necessary and sufficient conditions are derived by HAJEK (1988). Hajek's result is restricted to the case where the generation matrix is independent of c and the acceptance matrix is given by (3) (the Metropolis criterion). In order to formulate this result, we need two definitions:

DEFINITION 1. (HAJEK, 1988) A configuration j is called *reachable at height H* from a configuration i, if the following holds:

$$\exists p \geqslant 1,\ \lambda_0, \lambda_1, \ldots, \lambda_p \in \mathfrak{R}\ (i = \lambda_0, j = \lambda_p): \\ G_{\lambda_k \lambda_{k+1}} > 0,\ k = 0, 1, \ldots, p-1$$

and

$$C(\lambda_k) \leqslant H, \; k=0,1,...,p.$$

DEFINITION 2. (HAJEK, 1988) The *depth of a local minimum i* is defined as the smallest number $d(i)$ such that there is a configuration j with $C(j)<C(i)$ reachable at height $C(i)+d(i)$ from i. If i is a global minimum, then by definition $d(i)=+\infty$.

Hajek's result can be formulated as follows:

THEOREM 2. (HAJEK, 1988) *Suppose that the components of the transition matrix are given by* (11), *where* $A(c_k)$ *is given by* (3) *(the Metropolis criterion) and* $G(c_k)=G$ *satisfies the following two conditions:*

1. *condition* (1) *of Theorem* 1;
2. *for any real number H and any two configurations i and j, i is reachable at height H from j if and only if j is reachable at height H from i.*

Assume furthermore that the sequence $\{c_k\}$, $k=1,...$ *satisfies the following two conditions:*

$$1. \quad \lim_{k\to\infty} c_k = 0; \tag{12}$$

$$2. \quad c_k \geqslant c_{k+1}, \; k=1,2,.... \tag{13}$$

Then

$$\lim_{k\to\infty} \mathbf{P}_c\{X(k)\in\mathfrak{R}_{opt}\} = 1, \tag{14}$$

if and only if

$$\sum_{k=1}^{\infty} \exp(-\frac{D}{c_k}) = \infty, \tag{15}$$

where D is given by

$$D = \max\{d(i) | i \notin \mathfrak{R}_{opt}, \; i \text{ is a local minimum}\}.$$

Again, a few remarks are in order:

- The reader is referred to (KERN, 1986) for a detailed discussion of the notion depth. In particular, Kern considers the maximum depth D of all local minima (cf. Theorem 2). For several combinatorial optimization problems, upper bounds on D are derived and it is shown that the computation of D is at least as difficult as the solution of the problem itself.
- The conditions of Theorem 2 imply that the algorithm is allowed unlimited computation time: from (12) and (13) we conclude that all c_k are non-negative, whereas (15) implies that there can be no integer K, such that $c_k=0$ for $k \geqslant K$ (unless $D=0$, in which case there are no non-global local minima, so that even iterative improvement always finds the global minimum). Hence, $c_k>0$ for all k and the limit in (12) is attained only after an infinite number of transitions. Thus, the corresponding inhomogeneous Markov chain is of infinite length.

- The term $\exp(-\frac{D}{c_k})$ relates to the probability of accepting a transition corresponding to an increase in the cost function by an amount D. Thus, (15) states that for each value of the control parameter there should be a positive probability of accepting an increase in the cost function of value D.
- If c_k is of the form

$$c_k = \frac{\Gamma}{\log k},\ k = 1,2,..., \tag{16}$$

  for some constant Γ, Theorem 2 implies that (14) holds if and only if $\Gamma \geqslant D$. (16) is the expression for c_k which results from the necessary conditions derived in (GEMAN and GEMAN, 1984; ANILY and FEDERGRUEN, 1985; MITRA, ROMEO and SANGIOVANNI-VINCENTELLI, 1986; GELFAND and MITTER, 1985).
- Theorem 2 can also be applied to the homogeneous algorithm by considering the homogeneous algorithm as an inhomogeneous algorithm with a zero-decrement of c in between the transitions of the homogeneous Markov chains.

4. FINITE-TIME BEHAVIOUR OF SIMULATED ANNEALING

From the previous section we conclude that the asymptotic behaviour of both the homogeneous and the inhomogeneous algorithm is that of an optimization algorithm but that the conditions for asymptotic convergence imply that both algorithms must be allowed an infinite number of transitions. In this section we discuss the behaviour in finite time on the basis of a *cooling schedule.*

In any finite-time inplementation of the algorithm, the following parameters should be specified:

1. a **finite** number of transitions at each value of the control parameter or, in other words, a finite length of each homogeneous Markov chain;
2. a **finite** sequence of values of the control parameter, more specifically:
   (a) an initial value of the control parameter c_1;
   (b) a rule for changing the current value of the control parameter into the next one;
   (c) a final value of the control parameter (a stop criterion).

A choice for these parameters is referred to as a cooling schedule or an annealing schedule. The search for adequate cooling schedules has been addressed in many papers during the last few years. The literature provides a number of conceptually simple schedules that are similar to the original schedule proposed by KIRKPATRICK, GELATT and VECCHI (1983): the algorithm starts off at an experimentally determined value for c_1 for which the *acceptance ratio* $\chi(c_1)$ is smaller than but close to 1 ($\chi(c_k)$ is the ratio between the number of accepted transitions and the number of proposed transitions at a given value c_k). Next, a sequence of Markov chains is generated at descending values of c_k, where $c_{k+1} = \alpha \cdot c_k$, with α a constant close to 1 (typically between 0.9 and

0.99). Execution of the algorithm is terminated if the observed improvement in cost over a number of consecutive Markov chains is small.

More elaborate cooling schedules are given by a number of authors (VAN LAARHOVEN and AARTS, 1987). We briefly discuss the schedule described in (AARTS and VAN LAARHOVEN, 1985a). This is a three-parameter schedule: the parameters ξ and ϵ_s determine the initial and final values of the control parameter, respectively, whereas the decrement rule depends on a parameter δ. More precisely:

- The initial value of the control parameter, c_1, is chosen such that the acceptance ratio of the first Markov chain, $\chi(c_1)$, is approximately equal to ξ. ξ is chosen close to 1 (typically 0.95): in that case the first Markov chain is almost identical to a random walk through the configuration space, which is in accordance with the stationary distribution for $c \rightarrow \infty$ (the uniform distribution on $\mathfrak{R}$, which follows immediately from (3) and (8)).
- The decrement of the control parameter is given by

$$c_{k+1} = \frac{c_k}{1 + \frac{c_k \cdot \ln(1+\delta)}{3\sigma_k}}, \quad \delta > 0,$$

  where σ_k is the standard deviation of the cost values of the configurations obtained by generating the k-th Markov chain, and is such that the stationary distributions of two succeeding Markov chains approximately satisfy

$$\forall i \in \mathfrak{R}: \frac{1}{1+\delta} < \frac{q_i(c_k)}{q_i(c_{k+1})} < 1+\delta, \; k=1,2,...$$

  Thus, for small values of δ, the stationary distributions of succeeding Markov chains are "close" to each other and we may then expect that, after decreasing c_k to c_{k+1}, a small number of transitions suffices to let the probability distribution of the configurations approach the new stationary distribution $q(c_{k+1})$. Note that small values of δ correspond to a slow decrement of the control parameter.
- Small in 'a small number of transitions' (see the previous point) is specified as the size of the largest neighbourhood, i.e.

$$L_k = L = \max_{i \in \mathfrak{R}} |\mathfrak{R}_i|, \; k=1,2,.... \tag{17}$$

  primarily because we want the algorithm to have the possibility to visit at least a major part[1] of the neighbourhood of the current configuration before starting the generation of the next Markov chain.

1. It can be shown that the expected number of different elements found when sampling N times with replacement from a set with N elements is approximately $\frac{2}{3}N$ for $N > 5000$ (AARTS, BEENKER and KORST, 1985).

- Execution of the algorithm is terminated when

$$\frac{c_k}{\mu_1} \cdot \frac{\partial \bar{\mu}}{\partial c}(c_k) < \epsilon_s, \tag{18}$$

where μ_k is the average cost value of the k-th Markov chain and $\bar{\mu}(c_k)$ is obtained by smoothing the curve of μ_k. Thus, the stop criterion is satisfied when $\bar{\mu}(c) - C_{opt} \simeq c \cdot \frac{\partial \bar{\mu}(c)}{\partial c}$ is small compared to μ_1. In practice, this stop criterion turns out to be not much different from the previously mentioned 'simple' rule to terminate if the observed improvement in cost is small for a number of consecutive Markov chains.

It is possible to prove that the aforementioned decrement rule for the control parameter, together with the final value of the control parameter determined by (18), leads to a total number of steps in the control parameter bounded by $\mathcal{O}(\ln|\mathfrak{R}|)$. Thus, the computation time $\mathfrak{T}$ of the algorithm with the three-parameter schedule satisfies

$$\mathfrak{T} = \mathcal{O}(\tau \cdot L \cdot \ln|\mathfrak{R}|),$$

where the term L originates from the length of the Markov chains (17) and τ is the computation time of one transition. 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, ARAGON, MCGEOCH and SCHEVON (1987; 1988). For the travelling salesman problem, one of the best-known problems in combinatorial optimization, we present an empirical analysis of the effectivity and efficiency of simulated annealing in section 5.

To analyse the performance of an approximation algorithm, one should, besides efficiency and effectivity, also consider such criteria as *simplicity, ease of implementation* and *flexibility*. The simplicity of simulated annealing hardly calls for any further comment - it is part of its attraction. It is also an easy-to-implement algorithm - simulated annealing algorithms typically consist of a few hundred lines of computer code - though it is not always easy to formulate a problem in a way that lends itself to application of the algorithm; see for instance the application of simulated annealing to the *job shop scheduling* problem described in (VAN LAARHOVEN, AARTS and LENSTRA, 1988). Flexibility refers to the ability of an algorithm to handle problem variations and different problems. The rich variety of problems to which the algorithm is applied, see (VAN LAARHOVEN and AARTS, 1987), strongly supports the claim that simulated annealing is a very flexible approximation technique.

5. An example of the application of simulated annealing

Informally, the travelling salesman problem (TSP) can be stated as follows. A salesman starts from his home city, visits each city on a given list exactly once and returns to his home city. The problem is to select the order in which to visit the cities such that the total distance travelled in the tour is as small as possible.

More formally, the problem can be described as follows. If a permutation π of the set $\mathfrak{N}=\{1,...,n\}$ is interpreted such that $\pi(k)$, $k\in\mathfrak{N}$, denotes the successor of city k in the tour of a salesman who is to visit n cities, then each tour can be characterized by a *cyclic permutation* of $\mathfrak{N}$, i.e. a permutation π of $\mathfrak{N}$ such that for every $k\in\mathfrak{N}$ we have

$$\pi^l(k) \neq k,\ l = 1,2,...,n-1,\ \pi^n(k) = k.$$

The n-city TSP can now be stated as follows. Given an $n\times n$-matrix $D=(d_{ij})$, hereinafter referred to as the *distance matrix*, find a cyclic permutation π of $\mathfrak{N}$ minimizing

$$\sum_{k\in\mathfrak{N}} d_{k\pi(k)}.$$

To apply simulated annealing, we recall that we need to define configurations, a cost function and a neighbourhood structure. Furthermore, to prove asymptotic convergence we must show that the neighbourhood structure is such that condition (1) of Section 3 is satisfied. Hereinafter, we discuss these items in more detail.

(i) Configurations
Each configuration i of the problem is characterized by a cyclic permutation π_i of $\mathfrak{N}$; consequently, for a symmetric distance matrix, the total number of different configurations $|\mathfrak{R}|$ is $\frac{1}{2}(n-1)!$.

(ii) Cost function
The cost of a configuration i, characterized by a permutation π_i, is given by

$$C(i) = \sum_{k\in\mathfrak{N}} d_{k\pi_i(k)}.$$

(iii) Neighbourhood structure
A transition is generated by replacing two edges in the current tour by two non-tour edges such that again a tour is obtained (see Figure 1) (an edge in a tour, characterized by a permutation π, is a pair of cities $(k,\pi(k))$). This transition is called *2-change* and is a special case of *k-change* (Lin, 1965; Lin and Kernighan, 1973), in which k edges are replaced by k other edges. Iterative improvement algorithms based on this type of transition have been shown to be quite effective for the TSP; see for instance (Lin and Kernighan, 1973).

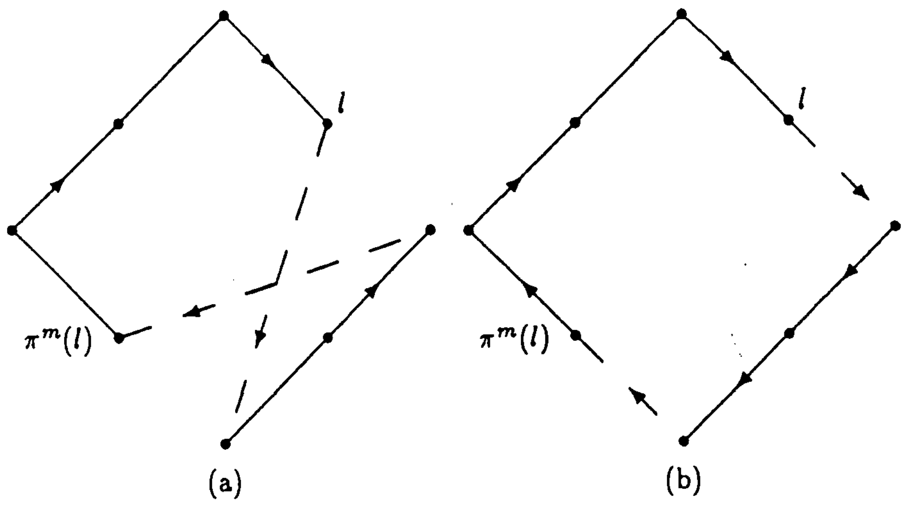

FIGURE 1: 2-change example; (a) current tour; (b) tour; after reversing the order between cities l and $\pi^m(l)$.

The neighbourhood $\mathfrak{R}_i$ of a configuration i, characterized by a permutation π_i, is thus given by

$$\mathfrak{R}_i = \{j \mid \pi_j \text{ is obtained from } \pi_i \text{ by a 2-change}\}$$

and $|\mathfrak{R}_i| = \begin{bmatrix} n \\ 2 \end{bmatrix} = \frac{1}{2}n(n-1)$ for all i.

(iv) Asymptotic convergence

For two arbitrary tours i and j, it is straightforward to construct a sequence of permutations $(\pi_{\lambda_0}, \pi_{\lambda_1}, \ldots, \pi_{\lambda_n})$, such that λ_0 is i and for $t \geq 1$, the first t cities in the tour λ_t are identical to the first t cities in the tour j. Consequently, $\pi_{\lambda_n} = \pi_j$, so that we have a sequence of n transitions leading from π_i to π_j. Thus, condition (1) of Theorem 1 is satisfied by the 2-change mechanism, since each configuration can be transformed into any other configuration by a finite number of 2-changes. The standard choices (2) and (3) therefore lead to asymptotic convergence.

The empirical analysis of the finite-time behaviour of the algorithm is carried out by running the algorithm on a number of instances of the travelling salesman problem, varying in size from 48 to 442 cities. The performance of simulated annealing for these instances is reported in Table 1. The averages in this table are computed from five final solutions obtained by running the algorithm, controlled by the cooling schedule described in Section 4, five times on each instance, each time with a different initial configuration. All results are obtained with the parameters ξ and ϵ_s set to 0.95 and 10^{-6}, respectively, and for different values of the distance parameter δ.

From Table 1 we observe firstly that the quality of the average solution (the average cost value) returned by the algorithm improves when δ is decreased. This is in accordance with the theory underlying the employed cooling schedule: a smaller value of δ leads to a better approximation of the asymptotic behaviour. Furthermore, we observe that the standard deviation in the final cost values decreases with decreasing δ, which indicates that the reliability of the results increases with decreasing δ. We also observe that the difference between the average final solution and a globally minimal one does not significantly deteriorate with increasing problem size, and that even for the largest problems the average deviation from the global minimum is less than 2%.

As for computation times, we mention that the average computation time $\bar{t}$ is approximately given by $\bar{t} = t_0 \cdot n^p \cdot \ln n$, for some constant t_0 where p is 2.53, 2.62 and 2.69 for δ is 10, 1 and 0.1, respectively ($\chi^2 = 1.000$). We remark that the bound for the computation time given by the worst-case bound mentioned in section 4 is $\Theta(n^3 \ln n)$ (since $L = \frac{1}{2}n(n-1)$ and $\ln|\mathfrak{R}| = \ln(\frac{1}{2}(n-1)!) = \Theta(n \ln n)$), which is only slightly worse than the observed computation times.

In Table 2, simulated annealing is compared with repeated execution of iterative improvement based on 2-changes. The initial configurations to which the iterative improvement algorithm is applied are randomly generated cyclic permutations. 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 0.1; C_{best} is the average of the best cost value found during each macro-run. The results for simulated annealing are taken from Table 1 ($\delta = 0.1$).

TABLE 1. Average cost of final solution ($\overline{C}_{final}$), average computation time in seconds ($\bar{t}$), standard deviations (σ_C and σ_t, respectively) and % of average final cost value above globally minimal cost value for different-sized instances of the travelling salesman problem (n denotes the number of cities). The results are obtained with the simulated annealing algorithm with different values of the distance parameter δ. The averages are obtained from 5 runs.

Problem	n	δ	$\overline{C}_{final}$	σ_C	%	$\bar{t}$	σ_t
GRO48	48	10.0	5203.6	65.5	3.12	6.3	0.3
		1.0	5203.0	111.4	3.11	15.9	1.5
		0.1	5094.8	23.9	0.97	93.8	5.6
TOM57	57	10.0	13340.8	241.4	2.98	10.2	0.5
		1.0	13218.6	153.2	2.03	26.9	2.3
		0.1	13068.0	50.5	0.87	158.2	3.9
EUR100	100	10.0	21852.6	329.7	3.40	45.8	1.7
		1.0	21711.8	200.9	2.73	121.2	2.5
		0.1	21339.4	171.9	0.97	801.6	8.3
GRO120	120	10.0	7269.6	81.1	4.72	72.2	3.2
		1.0	7174.8	29.8	3.35	205.6	1.5
		0.1	7057.2	72.0	1.66	1369.4	24.5
LIN318	318	10.0	43132.8	545.3	4.44	1126.8	29.7
		1.0	42262.6	183.4	2.33	3437.6	147.1
		0.1	41957.4	176.8	1.59	23941.6	967.8
GRO442	442	10.0	5287.0	28.2	4.30	2749.6	70.0
		1.0	5206.8	25.5	2.72	8458.4	106.9
		0.1	5147.0	20.9	1.54	58674.6	432.4

We observe that for the larger problems repeated execution of iterative improvement is easily outperformed by simulated annealing, even if we take into account the standard deviations (which are slightly smaller for iterative improvement). For GRO442 the difference is especially pronounced and the results suggest that the quality of the average best solution returned by (repeated execution of) the iterative improvement algorithm deteriorates significantly with increasing problem size, contrary to simulated annealing.

Table 2 also contains a similar comparison with repeated execution of the *Lin-Kernighan algorithm* (LIN and KERNIGHAN, 1973), which is known as one of the best approximation algorithms tailored to the TSP. The Lin-Kernighan algorithm is a sophisticated iterative improvement algorithm, based on k-

changes, where at each stage the algorithm chooses dynamically a 'good' value for k. For each instance, the number of (randomly generated) initial solutions for repeated execution of the Lin-Kernighan algorithm was predetermined such that the average computation time of a macro-run for that instance was approximately the same as an average run of simulated annealing with the distance parameter δ set to 0.1. The comparison is biased in favour of simulated annealing by the fact that the cost values in the right-hand part of Table 2 were obtained by simulated annealing, followed by iterative improvement based on 2-changes and a special type of 3-changes (those where two of the three cities are successors in the current tour).

TABLE 2. Average cost of best solution ($\overline{C}_{best}$) or final solution ($\overline{C}_{final}$), standard deviation (σ_C), average number of local minima per macro-run of the iterative improvement/Lin-Kernighan algorithm ($\overline{lm}$) and % of average final cost value (annealing) above average best cost value (iterative improvement/Lin-Kernighan) for different-sized instances of the travelling salesman problem. The results are obtained with repeated execution of the iterative improvement/Lin-Kernighan algorithm and with simulated annealing, respectively. The averages are obtained from five (macro-)runs.

	Iterative Improvement			*Simulated Annealing*		
Problem	$\overline{C}_{best}$	σ_C	$\overline{lm}$	$\overline{C}_{final}$	σ_C	%
GRO48	5056.0	4.6	399.6	5094.8	23.9	0.76
TOM57	12997.0	46.4	435.6	13068.0	50.5	0.55
EUR100	21596.6	77.3	528.2	21339.4	171.9	-1.19
GRO120	7146.6	32.0	606.8	7057.2	72.0	-1.25
LIN318	43313.8	159.8	851.4	41957.4	176.8	-3.13
GRO442	5389.6	13.7	1011.0	5147.0	20.9	-4.50
	Lin-Kernighan			*Simulated Annealing*		
GRO48	5046.0	0.0	66	5084.2	22.4	0.76
TOM57	12960.0	0.0	84	13063.4	50.9	0.84
EUR100	21135.0	0.0	87	21336.2	169.6	0.96
GRO120	6943.8	3.6	112	7037.6	56.5	1.35
LIN318	41433.2	13.9	214	41862.2	231.6	1.04
GRO442	5080.8	5.7	250	5136.4	13.7	1.09

The results of Table 2 clearly indicate that repeated execution of the Lin-

Kernighan algorithm is superior to the simulated annealing algorithm; in fact, even if the former is allowed only a limited fraction of the computation time taken by the latter, it still finds substantially better solutions (cf. the small values of the standard deviation σ_C).

Apart from the travelling salesman problem, simulated annealing has been widely applied to other well-known combinatorial optimization problems, see e.g. (AARTS and VAN LAARHOVEN, 1985b) (graph partitioning), (BONOMI and LUTTON, 1984) (travelling salesman), (BONOMI and LUTTON, 1986) (quadratic assignment), (BURKARD and RENDL, 1984) (quadratic assignment), (GOLDEN and SKISCIM, 1986) (travelling salesman), (JOHNSON, ARAGON, MCGEOCH and SCHEVON, 1987) (graph partitioning), JOHNSON, ARAGON, MCGEOCH and SCHEVON, 1988) (travelling salesman, number partitioning, graph colouring), (KIRKPATRICK, 1984) (travelling salesman), (VAN LAARHOVEN, AARTS and LENSTRA, 1988) (job shop scheduling), (LUTTON and BONOMI) (matching), (MORGENSTERN and SHAPIRO) (graph colouring). The results described in these papers and in this section are fairly consistent in the conclusions that can be drawn from them:

- The algorithm has a potential for finding high-quality solutions; the required amount of computation time to realize this potential is usually quite large.
- The probabilistic element of the algorithm (the acceptance of cost-increasing transitions with a non-zero probability) makes simulated annealing a significantly better technique than the iterative improvement algorithm on which it is based. The difference between these two algorithms is especially pronounced for large problem instances.
- Simulated annealing is not a panacea: if a sophisticated tailored algorithm is available, it is usually competitive with and often superior to simulated annealing.

6. CONCLUSION

Ever since its introduction in 1982, simulated annealing has been successfully and widely applied in such diverse areas as, for example, computer-aided circuit design, image processing, code design and neural network theory; for a review the reader is referred to (VAN LAARHOVEN and AARTS, 1987). In many of these areas, previous algorithms, if existing at all, performed quite poorly and the territory was ripe for an easy-to-implement approach with a modicum of sophistication.

In more competitive areas, as for example the problems mentioned at the end of section 5, the algorithm is usually able to produce near-optimal solutions (typically 1-2% in cost above a globally minimal solution), but the computational effort involved is usually much bigger than that of a tailored algorithm. However, we conjecture that the computation times taken by the algorithm can be considerably reduced through execution on a multiprocessor architecture, without affecting the quality of the solutions found by the algorithm. See for example (AARTS, DE BONT, HABERS and VAN LAARHOVEN, 1986), where it is shown that an almost linear speed-up can be achieved on an 8-processor architecture when applying simulated annealing to instances of the travelling salesman problem. Another promising area is that of the *Boltzmann machine* (AARTS and KORST, 1987), which can be seen as a model for massively parallel hardware implementation of the simulated annealing algorithm.

The success of simulated annealing has given rise to research on other approaches to the approximate solution of combinatorial optimization problems based on similar hill-climbing ideas. The *tabu search* method (GLOVER, 1985) is a recent example of such an approach. It has been reported that for graph colouring problems tabu search is more effective and efficient than simulated annealing (HERTZ and DE WERRA, 1987); for other problems such comparisons are not yet available.

Finally we mention that a theoretical analysis of the finite-time effectivity of the algorithm is still an open problem and considered to be an important area for future research.

REFERENCES

AARTS, E.H.L., G.F.M. BEENKER and J.H.M. KORST (1985), Asymptotic Probability Calculations for Aselect Sampling, Philips Research Report No. 6073.

AARTS, E.H.L., F.M.J. DE BONT, J.H.A. HABERS and P.J.M. VAN LAARHOVEN (1986), Parallel Implementations of the Statistical Cooling Algorithm, *Integration 4,* 209-238.

AARTS, E.H.L. and J.H.M. KORST (1987), Boltzmann Machines and their Applications, *Proc. PARLE, Springer Lecture Notes in Computer Science 258,* 34-50.

AARTS, E.H.L. and P.J.M. VAN LAARHOVEN (1985a), Statistical Cooling: A General Approach to Combinatorial Optimization problems, *Philips Journal 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 1987, pp. 206-208.

ANILY, S. and A. FEDERGRUEN (1987), Simulated Annealing Methods with General Acceptance Probabilities, *Journal of Applied Probability 24,* 657-667.

BONOMI, E. and J.-L. LUTTON (1984), The *N*-city Travelling Salesman problem: Statistical Mechanics and the Metropolis Algorithm, *SIAM Review 26,* 551-568.

BONOMI, E. and J.-L. LUTTON (1986), The Asymptotic Behaviour of Quadratic Sum Assignment Problems: A Statistical Mechanics Approach, *European Journal of Operational Research 26,* 295-300.

BURKARD, R.E. and F. RENDL (1984), A thermodynamically motivated simulation procedure for combinatorial optimization problems, *European Journal of Operational Research 17,* 169-174.

CERNY, V. (1985), Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm, *Journal Opt. Theory Appl. 45,* 41-51.

FELLER, W. (1950), *An Introduction to Probability Theory and Its Applications,* Vol. 1, Wiley, New York.

GAREY, M.R. and D.S. JOHNSON (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W.H. Freeman and Co., San Francisco.

GELFAND, S.B. and S.K. MITTER (1985), Analysis of Simulated Annealing for Optimization, *Proc. 24th Conf. on Decision and Control,* Ft. Lauderdale, December 1985, pp. 779-786.

GEMAN, S. and D. GEMAN (1984), Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, *IEEE Proc. Pattern Analysis and Machine Intelligence PAMI-6,* 721-741.

GLOVER, F., C. MCMILLAN and B. NOVICK (1985), Interactive Decision Software and Computer Graphics for Architectural and Space Planning, *Annals of Operations Research 5,* 557-573.

GOLDEN, B.L. and C.C. SKISCIM (1986), Using Simulated Annealing to Solve Routing and Location Problems, *Naval Logistics Research Quarterly 33,* 261-279.

HAJEK, B. (1988), Cooling Schedules for Optimal Annealing, *Mathematics of Operations Research 13,* 311-329.

HERTZ. A. and D. DE WERRA (1987), Using Tabu Search Techniques for Graph Coloring, *Computing 39,* 345-351.

JOHNSON, D.S., C.R. ARAGON., L.A. MCGEOCH and C. SCHEVON (1987), Optimization by Simulated Annealing: an Experimental Evaluation (Part I), to appear in *Operations Research.*

JOHNSON, D.S., C.R. ARAGON, L.A. MCGEOCH and C. SCHEVON (1988). Optimization by Simulated Annealing: an Experimental Evaluation (Part II), in preparation.

KERN, W. (1986b), On the Depth of Combinatorial Optimization Problems, Universität zu Köln, Report No. 86.33.

KIRKPATRICK, S., C.D. GELATT JR. and M.P. VECCHI (1983), Optimization by Simulated Annealing, *Science 220,* 671-680.

KIRKPATRICK, S. (1984), Optimization by Simulated Annealing: Quantitative Studies, *Journal of Statistical Physics 34,* 975-986.

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 Universiteit, Rotterdam; also *CWI Tract* 51, Centre for Mathematics and Computer Science, Amsterdam.

LAARHOVEN, P.J.M. VAN, E.H.L. AARTS and J.K. LENSTRA (1988), Job Shop Scheduling by Simulated Annealing, to appear in *Operations Research.*

LAWLER, E.L. (1976), *Combinatorial Optimization: Networks and Matroids,* Holt, Rinehart and Winston, New York.

LIN, S. (1965), Computer Solutions of the Traveling Salesman problem, *Bell System Tech. Journal 44,* 2245-2269.

LIN, S. and B.W. KERNIGHAN (1973), An Effective Heuristic Algorithm for the Traveling Salesman Problem, *Operations Research 21,* 498-516.

LUNDY, M. and A. MEES (1986), Convergence of an Annealing Algorithm, *Mathematical Programming 34,* 111-124.

LUTTON, J.-L. and E. BONOMI (1986), Simulated Annealing Algorithm for the Minimum Weighted Perfect Euclidean Matching Problem, *R.A.I.R.O. Recherche opérationnelle 20,* 177-197.

METROPOLIS, N., A. ROSENBLUTH, M. ROSENBLUTH, A. TELLER and E. TELLER (1953), Equation of State Calculations by Fast Computing Machines, *Journal of Chem. Physics 21,* 1087-1092.

MITRA, D., ROMEO, F. and A.L. SANGIOVANNI-VINCENTELLI (1986), Convergence and Finite-Time Behavior of Simulated Annealing, *Adv. Appl. Prob. 18,* 747-771.

MORGENSTERN, C.A. and H.D. SHAPIRO (1986), Chromatic Number Approximation Using Simulated Annealing, Department of Computer Science, The University of New Mexico, Albuquerque, Technical Report No. CS86-1.

PAPADIMITRIOU, C.H. and K. STEIGLITZ (1982), *Combinatorial Optimization: Algorithms and Complexity,* Prentice-Hall, New York.

SENETA, E. (1981), *Non-negative Matrices and Markov Chains,* (2nd ed.), Springer Verlag, New York.

Received March 1988, Revised August 1988.