

Nonequilibrium Simulated Annealing: A Faster Approach to Combinatorial Minimization

Margarida F. Cardoso, Romualdo L. Salcedo,* and Sebastião F. de Azevedo

Departamento de Engenharia Química, Faculdade de Engenharia da Universidade do Porto, Rua dos Bragas, 4099 Porto, Portugal

A nonequilibrium simulated annealing (NESA) algorithm is presented for solving combinatorial minimization problems. The original Metropolis algorithm and its variant due to Glauber were modified by enforcing the cooling schedule as soon as an improved solution is obtained, without the need to reach near-equilibrium conditions at each temperature level. The original and modified algorithms were applied to the classical traveling salesman problem and to the optimization of a pressure relief header network. Statistical evaluation of the performance of the algorithms revealed that the proposed modification resulted in a faster approach to the global optimum. A simple stopping criterion, based on an averaged gradient of the objective function with respect to the number of function evaluations, provides a convenient method to control the convergence of the modified algorithms.

Simulated Annealing: An Overview

Simulated annealing represents a class of solution methods for combinatorial optimization problems whose common basis is found on the analogies which can be drawn between those problems and the physical process of annealing.

Combinatorial minimization is the optimization of functions that may assume several distinct discrete configurations. This class of problems is well represented by the classical traveling salesman problem (Stanley and Sherman, 1965; Lin and Kernighan, 1973) and by the optimization of process equipment networks, namely heat-exchanger and pressure relief header networks (Dolan et al., 1989, 1990). These problems are characterized by the usefulness in achieving near-optimum solutions in an adequate processing time; viz., it may not be necessary to fully determine the global optimum, since the small incremental gain in the objective function would only be achieved at a great expense in central processing unit (CPU) time.

Annealing is the physical thermal process of melting a solid by heating it, followed by slow cooling and crystallization into a minimum free energy state, viz., a stable state. If the cooling rate is not carefully controlled or the initial temperature is not sufficiently high, the cooling "solid" does not attain thermal equilibrium at each temperature. In such circumstances, local optimal lattice structures may occur which translate into lattice imperfections, viz., a metastable state. Thermal equilibrium at a given temperature is characterized by a Boltzmann distribution function of the energy states. Under these conditions, even at a low temperature, a transition may occur from a low to a high energy level, albeit with a small probability. Such transitions are assumed to be responsible for the system reaching a minimum energy state instead of being trapped in a local metastable state.

The Metropolis algorithm (Metropolis et al., 1953; Kirkpatrick et al., 1983; Kirkpatrick, 1984) was the first proposed to simulate this process. Starting from a high energy state E_i , corresponding to a particular system temperature T_i , a series of new energy states E_j are stochastically generated. Each new system configuration

is accepted if $E_j \leq E_i$. Otherwise, and by analogy with the Boltzmann distribution for energy states at thermal equilibrium, E_j will be accepted as an improved state with a probability given by $P(\Delta E) = \exp[(E_i - E_j)/K_B T_i]$, where T_i is the current system temperature and K_B is Boltzmann's constant. At high temperatures this probability is close to 1; viz., most energy transitions are permissible. At each temperature it is assumed that thermal equilibrium is reached. As the system temperature decreases, the probability of accepting a higher energy state as being an improved energy state approaches 0.

At the core of the approach and from an optimization point of view, simulated annealing explores the key feature of algorithms for the physical annealing process of generating transitions to higher energy states, applying to the new states an acceptance/rejection probability criterion which should naturally become more and more stringent with the progress of the procedure. Those possible transitions correspond to "wrong-way" movements in optimization algorithms, employed to overcome local optima (Salcedo et al., 1990). For solving a particular problem with a simulated annealing algorithm, the following steps are thus necessary:

- (i) definition of an objective function to be minimized
- (ii) adoption of an annealing cooling schedule, whereby the initial temperature, the number of configurations generated at each temperature, and a method to decrease it are specified
- (iii) at each temperature in the cooling schedule, stochastic generation of the alternative combinations, centered on the currently accepted state
- (iv) adoption of criteria for the acceptance or rejection of the alternative combinations, against the currently accepted state at that temperature

It is self-evident that steps ii and iv constitute the algorithmic essence of the simulated annealing approach. Step i is eminently problem-dependent whereas step iii can be seen as a mathematical issue whose main argument for discussion is on the possible influence of the pseudo-random sequences on the results (since in practice pure random sequences cannot be generated).

Simulated annealing has been applied to a great variety of combinatorial minimization problems, such as the classical traveling salesman problem (Press et al., 1986), heat-exchanger and pressure relief header networks (Dolan et al., 1989, 1990), graph partitioning (Johnson et al., 1989),

* Author to whom correspondence should be addressed.
E-mail: rsalcedo@fe.up.pt.

optimization of physical data tables (Corey and Young, 1989), batch process scheduling (Das et al., 1990; Ku and Karimi, 1991; Patel et al., 1991), graph coloring and number partitioning (Johnson et al., 1991), and imaging applications (Silverman and Addler, 1992; Groisman and Parker, 1993). Apart from combinatorial minimization, simulated annealing has been applied to pure integer 0-1 problems (Connolly, 1992) and to the optimization of multimodal continuous spaces (Vanderbilt and Louie, 1984; Bohachevsky et al., 1986; Corana et al., 1987; Press and Teukolsky, 1991). Aarst and Korst (1989) give an excellent overview of simulated annealing, concerning both practical applications and theory.

The Metropolis and the Glauber algorithms (Glauber, 1963) are possibly the best known proposals for computing the probability for accepting a move (Das et al., 1990; Patel et al., 1991). Several theoretical studies are devoted to development of more effective rejection/acceptance criteria (Greene and Supowit, 1986; Anily and Federgruen, 1987) and to the convergence properties of various simulated annealing schemes (Faigle and Kern, 1991). Also, theoretical efforts have been concentrated on determining efficient general cooling schedules (White, 1984; Huang et al., 1986). In analogy to the physical process, a fast cooling schedule will most probably drive the algorithm toward a poor local optimum (a metastable state) whereas a slow cooling schedule will produce unacceptably large processing times. Finally, the basic requirement for reaching thermal equilibrium at every temperature along the cooling schedule is usually transported to the practical implementations on the form of a parameter representing the number of random configurations generated at each level. Algorithms published in the literature implement the view that if such number is made sufficiently high, equilibrium or near equilibrium will be attained.

This paper is limited in scope to exploring hypotheses which break equilibrium coupled with appropriate convergence tests, with respect to their influence on the convergence and overall performance of algorithms based on the original Metropolis and Glauber approaches. It is shown, for two types of problems, that simple modifications of the original algorithms, albeit leading to nonequilibrium trajectories, result in significant improvements in algorithm performance, namely a much faster approach to the global optimum.

Algorithms and Overall Test Conditions

Simulated annealing algorithms are usually recognized in the literature with respect to the proposal for acceptance/rejection procedure adopted. Thus, different algorithms derived by employing alternative schemes for the initial temperature and for the cooling schedule are seen as variants of the basic approaches. Accepting such naming, in this work we deal with the following four simulated annealing algorithms: (i) the original Metropolis algorithm, (ii) the original Glauber algorithm, (iii) the NESAs/Metropolis algorithm, and (iv) the NESAs/Glauber algorithm.

The Metropolis and the Glauber Algorithms. With the Metropolis algorithm, the probability of acceptance of new configurations (solutions) is as follows:

$$P = 1; \quad \Delta C < 0 \\ = \exp(-\Delta C/K_B T); \quad \Delta C \geq 0 \quad (1)$$

where ΔC is the difference in cost between a newer configuration and the current solution, K_B is Boltzmann's constant, and T is the annealing temperature.

```

a Estimate the initial temperature control parameter (Ti)
  Stochastically generate a starting solution vector (Vi)
  Start global iteration
    Start inner loop (j=1 to L)
      Generate new solution (Vj)
      d = FOB(j) - FOB(i)
      if Metropolis then
        if (d ≤ 0) or (d > 0 and exp(-d/Ti) > rnd(0,1))
          update solution (Vi = Vj)
        elseif Glauber then
          if exp(-d/Ti) / (1 + exp(-d/Ti)) > rnd(0,1)
            update solution (Vi = Vj)
          end if
        end if
      End inner loop
    Test convergence of current solution vector Vi
    If test passes
      stop
      Decrease Ti according to cooling schedule
    End global iteration

b Estimate the initial temperature control parameter (Ti)
  Stochastically generate a starting solution vector (Vi)
  Start global iteration
    Start inner loop (j=1 to L)
      Generate new solution (Vj)
      d = FOB(j) - FOB(i)
      if Metropolis then
        if (d ≤ 0) or (d > 0 and exp(-d/Ti) > rnd(0,1))
          update solution (Vi = Vj)
          if (d < 0) exit inner loop
        elseif Glauber then
          if exp(-d/Ti) / (1 + exp(-d/Ti)) > rnd(0,1)
            update solution (Vi = Vj)
            if (d < 0) exit inner loop
          end if
        end if
      End inner loop
    Test convergence of current solution vector Vi
    If test passes
      stop
      Decrease Ti according to cooling schedule
    End global iteration
  
```

Figure 1. (a) Pseudoprogram for the original Metropolis and Glauber algorithms. (b) Pseudoprogram for the modified Metropolis (NESAs/Metropolis) and Glauber (NESAs/Glauber) algorithms.

Glauber (1963) proposed a variant of the Metropolis algorithm, whereby both uphill and downhill moves are accepted with a probability given by

$$P = \frac{\exp(-\Delta C/K_B T)}{1 + \exp(-\Delta C/K_B T)} \quad (2)$$

In limit, at high temperatures, viz., far from the global optimum, the Metropolis algorithm will accept all moves with probability 1. The Glauber algorithm will accept all moves with probability 0.5. As the temperature decreases, the Metropolis algorithm will accept downhill moves (for a minimization problem) with a probability of 1 and uphill moves with a probability that approaches 0. On the other hand, the Glauber algorithm will accept downhill moves with a probability that approaches 1, and uphill moves with a probability that approaches 0.

The overall procedure is represented in Figure 1a. Details on the estimation of initial temperatures, the cooling schedule, and termination criteria will be discussed

later in this section. At this point, it is relevant to note that, in this procedure, equilibrium is assumed to be reached by promoting L iterations in the inner loop.

Nonequilibrium Simulated Annealing (NESA). The first modification proposed in this paper consists of enforcing the cooling schedule as soon as an improved (for the Metropolis procedure) and accepted (for the Glauber procedure) solution is obtained. The algorithm is represented in Figure 1b. This modification corresponds to a change in the termination criterion for the inner loop. The condition of "thermal" equilibrium will thus not be necessarily attained at each level of the "temperature" control parameter. At the beginning of the optimization, the "temperature" control parameter is estimated according to some criterion adopted (see below). At this level the solution is most probably far from the optimum and CPU time is not wasted in the search for a near equilibrium state. As the optimization proceeds, the "temperature" control parameter drops and solution acceptance is decided according to the algorithm being employed. It will eventually drop to a point where no more poorer acceptances are allowed, thus assuring convergence to a local optimum, as in the patterns observed for the original simulated annealing procedures.

The second modification proposed is the adoption of a convergence criterion based on an averaged gradient of the objective function with respect to the number of function evaluations. This is suggested by the trajectories observed (average path vs number of function evaluations) in the application of the NESA approach to the case studies employed in this work. It will be shown later that such modification provides an improved variant of the NESA algorithms, as measured against the original procedures.

Initial Annealing Temperature. The initial control temperature is an important parameter which can be obtained by specifying the fraction of generated solutions to be accepted initially (Aarst and Korst, 1989; Ku and Karimi, 1991; Patel et al., 1991). With a high initial temperature, the number of moves accepted is large and the parameter space is more thoroughly searched. This may result in long computational times, for the unmodified algorithms. Such large initial space is a good feature and will not be a problem for the implementation of the NESA modifications, since the cooling schedule is enforced at a faster pace.

It is important that the initial temperature control parameter be estimated such that the acceptance probabilities are close to 1 for the Metropolis algorithm and close to 0.5 for the Glauber algorithm. For this purpose, the initial temperature T_i was estimated by

$$X = \frac{m_1 + m_2 \exp(-\Delta f^+ / T_i)}{m_1 + m_2} \quad (3)$$

where the acceptance ratio X was set to 95% (Aarst and Korst, 1989). For a universe of m_0 combinations, and in a sequential analysis, m_1 represents the number of successful moves ($E_k < E_{k-1}$ for $k = 2$ to m_0), m_2 the number of unsuccessful moves ($E_k > E_{k-1}$) and Δf^+ the average increase in cost for the m_2 unsuccessful moves.

Cooling Schedule. In this work, the cooling schedule employed was the Aarst and Van Laarhoven (1985) scheme

$$T_{\text{new}} = \frac{T_{\text{old}}}{1 + \frac{T_{\text{old}} \ln(1+\delta)}{3\sigma}} \quad (4)$$

where δ and σ are trajectory parameters. The parameter δ controls the cooling rate and is a measure of the desired

closeness to equilibrium (Das et al., 1990; Patel et al., 1991). Small values of δ (<1) produce slow convergence and large values (>1) convergence to poor local optima. The cooling schedule and the acceptance criterion have a coupled effect in convergence. Consequently, the same value of δ will produce significantly different trajectories for the unmodified and modified algorithms. In this work, tests were performed for a range of δ values for the unmodified algorithms. Where required for comparative analysis of performance, corresponding values of δ for the modified algorithms were found by trial and error, in order to achieve a similar number of function evaluations. This allows a first comparison of the algorithms based on similar computational efforts.

The parameter σ is the standard deviation of all cost configurations at the current temperature T_{old} . With the NESA algorithms, situations arise where this parameter cannot be computed, since only a single cost may be evaluated before exiting the inner loop (Figure 1b). This occurs frequently at the beginning of the optimization, and sporadically afterward. Thus, to compute the initial value for σ , the first inner loop for the NESA variants was completely evaluated, such as with the original algorithms. During the course of the optimization, whenever the standard deviation of the cost function could not be computed for the NESA algorithms, the value corresponding to the previous temperature level was automatically assumed.

Alternative cooling schedules are usually of the exponential type (Kirkpatrick et al., 1983), where a constant multiplicative factor is employed to obtain the new temperature. Tests performed lead to verify that both the original and modified algorithms performed better with the Aarst and Van Laarhoven (1985) scheme, rather than with the exponential decrease in temperature, a finding which is in agreement with those of other authors (Aarst and Korst, 1989; Das et al., 1990; Dolan et al., 1990; Patel et al., 1991).

Generation of System Configurations. The generation of each system configuration on a N -dimensional space, specified by the vector components V_j ($j = 1, N$), from the previous solution V_i ($i = 1, N$), may be problem dependent. The generated configurations are options presented to the system and may be obtained by simple random changes or may include some heuristic guidance (Press et al., 1986). As seen below with the two examples given in this paper, each particular problem may exploit a different generating scheme.

Performance Evaluation: Numerical Implementation and Case Studies. The four candidate algorithms and their internal variants were written in Fortran 77. All runs were performed with double precision on a HP 730 Workstation, running compiler optimized Fortran 77 code.

Simulated annealing algorithms are stochastic, and as such can be sensitive to the sequence of pseudorandom numbers. Thus, the pseudorandom number generators employed within the computer programs that implement these methods should be uniform and independent, so that deficiencies in the optimization algorithms may not somehow be compensated by deficiencies in the pseudorandom number generators. In this work, a Lehmer linear congruential generator (Shedler, 1983) was used, which was tested for multidimensional uniformity and found to be satisfactory (Salcedo et al., 1990; Salcedo, 1992).

Performance evaluation was carried out with a 400-city traveling salesman problem (Kirkpatrick et al., 1983) and with the optimization of a pressure relief header network (Cheng, 1976; Cheng and Mah, 1976; Dolan et al., 1989).

These two examples were chosen since heuristic rules of reversing and interchanging segments are employed for solving the traveling salesman problem, while no heuristics are at all employed for optimizing the pressure relief header network. Thus, two completely different types of combinatorial minimization problems and approaches were considered.

For the traveling salesman problem, to avoid excessive computational burden, the performance of the algorithms was statistically evaluated by starting from 10 different configurations (energy levels or costs), obtained by random shuffling of permissible states (Press et al., 1986).

For the optimization of the pressure relief header network, a problem requiring considerably less CPU time, the algorithms were run for a single starting configuration corresponding to the maximum possible cost (Dolan et al., 1989), but with 100 different seeds, taken from Shedler (1983), for the pseudorandom number generator used to provide the different stochastic solutions.

Thus, 10 runs were performed for each algorithm with the traveling salesman problem and 100 runs for the optimization of the pressure relief header network. Comparison of the algorithms is provided on the basis of average solutions and corresponding standard deviations, best solutions attained, average number of function evaluations, and a direct measure of CPU time. Typical annealing schedules are also given, since these provide a better insight as to the progress of the algorithms.

Termination Criteria. The number of function evaluations in the inner loop, which is the maximum number of moves at each temperature level, was fixed at 100 times the number of cities (for the traveling salesman problem) and at both 2 and 100 times the number of pipe segments (for the pressure relief header network, as explained below). For the original Metropolis and Glauber algorithms, the number of moves at each temperature level should be sufficiently large to allow equilibrium to be reached.

Both the original and modified algorithms were initially implemented with the same stopping criteria for the outer loop (corresponding to "test convergence of current solution vector" in Figure 1). Patel et al. (1991) studied two different termination criteria: specifying a lower limit for the derivative of the smoothed average cost at a temperature level with respect to that temperature and specifying a fixed number of iterations occurring without acceptance of any moves. These authors found that both criteria gave satisfactory results. In this work, preliminary tests lead to employing the latter criterion, viz., failure to obtain an improved solution in the inner loop, since it gave essentially the same results as the more complex derivative criterion.

However, later in the work, a stopping criterion based on the derivative of an averaged cost with respect to the number of function evaluations was successfully implemented when using the NESAs variants. This will be presented and discussed after the pressure relief header network case study.

Results and Discussion

Traveling Salesman Problem. The objective of the traveling salesman problem (Stanley and Sherman, 1965; Lin and Kernighan, 1973) is to find the minimum closed path that visits n cities, given a matrix of their coordinates and the constraint that each city should only be visited once. This problem belongs to a class of minimization problems where the objective function has several local optima which are close to the global optimum, even for moderate values of n . Thus, an optimization algorithm

that obtains near global optimum solutions, which probably cannot be improved without undue computational effort, is certainly of practical significance. Such is the case of simulated annealing, which has been applied by a variety of authors to this type of problem (Kirkpatrick, 1984; Huang et al. 1986; Press et al., 1986; Aarst and Korst, 1989; Percy, 1992).

We have chosen the traveling salesman problem given by Kirkpatrick (1984) because it has a known analytical solution. It corresponds to a fictitious 400-city problem, where the cities are points on a regular 20×20 square grid in a square of unit dimension. The normalized global optimum is known to be 1 (Kirkpatrick, 1984).

Some heuristic rules related to the interchange of cities and segment reversal were included in all four algorithms (Lin and Kernighan, 1973; Press et al., 1986). Heuristics are simple rules that can limit the number of prospects that must be examined, while hopefully eliminating most of the poor solutions (Percy, 1992). Thus, heuristics tend to be problem-specific. For the traveling salesman problem, heuristics provide a substantial reduction in the computational effort required to attain near global optimum solutions, and as such were considered in the present work. These rules are available in a computer listing given by Press et al. (1986). Starting from an initial solution vector V_i ($i = 1, N$), subsequent solutions V_j ($j = 1, N$) are obtained as suggested by Lin and Kernighan (1973), either by replacing a random section of path by the same cities running in the opposite order or by moving a section of path to another place in between different cities stochastically chosen.

The first objective in the analysis was to assess the capacity of the NESAs algorithms to attain similar results (average and best path indexes) for a similar number of function evaluations.

Figure 2a shows a typical starting configuration, corresponding to a high energy state, while Figure 2b shows the best result obtained. Table 1 gives a summary of the statistical evaluation of the four algorithms, for a single seed and ten different starting configurations. Thus, only suboptimal solutions were achieved with all algorithms, albeit very close to the global optimum and corresponding to the presence of only a few local imperfections (see Figure 2b). All results are, on the average, within 1.6% of the global optimum. This figure compares well with a 2% reported figure for a similar-sized problem (Huang et al., 1986). It can be seen that all four candidate algorithms performed reasonably well and on the average performed better as the cooling parameter δ decreases. This is in agreement with the results obtained by Dolan et al. (1989), which used $\delta = 0.01$, Das et al. (1990), which used $\delta = 0.3$, and Patel et al. (1991), which found an optimum δ of 0.11, albeit for other types of problems. As stated before, this parameter was estimated for the NESAs algorithms by trial and error, in order to obtain a similar number of function evaluations as those corresponding to the unmodified algorithms. Table 1 also shows that the Metropolis algorithm, whether in its original or modified form, seems to perform better than the Glauber algorithm, in agreement with the findings of Patel et al. (1991) for other types of problems.

The second aspect for performance evaluation concerns the trajectories observed for each algorithm. Figure 3 shows the average paths obtained with the four algorithms as a function of the incremental number of function evaluations and the cooling parameter δ . It is clearly seen that the NESAs variants approach toward the global minimum is much faster than that observed with the

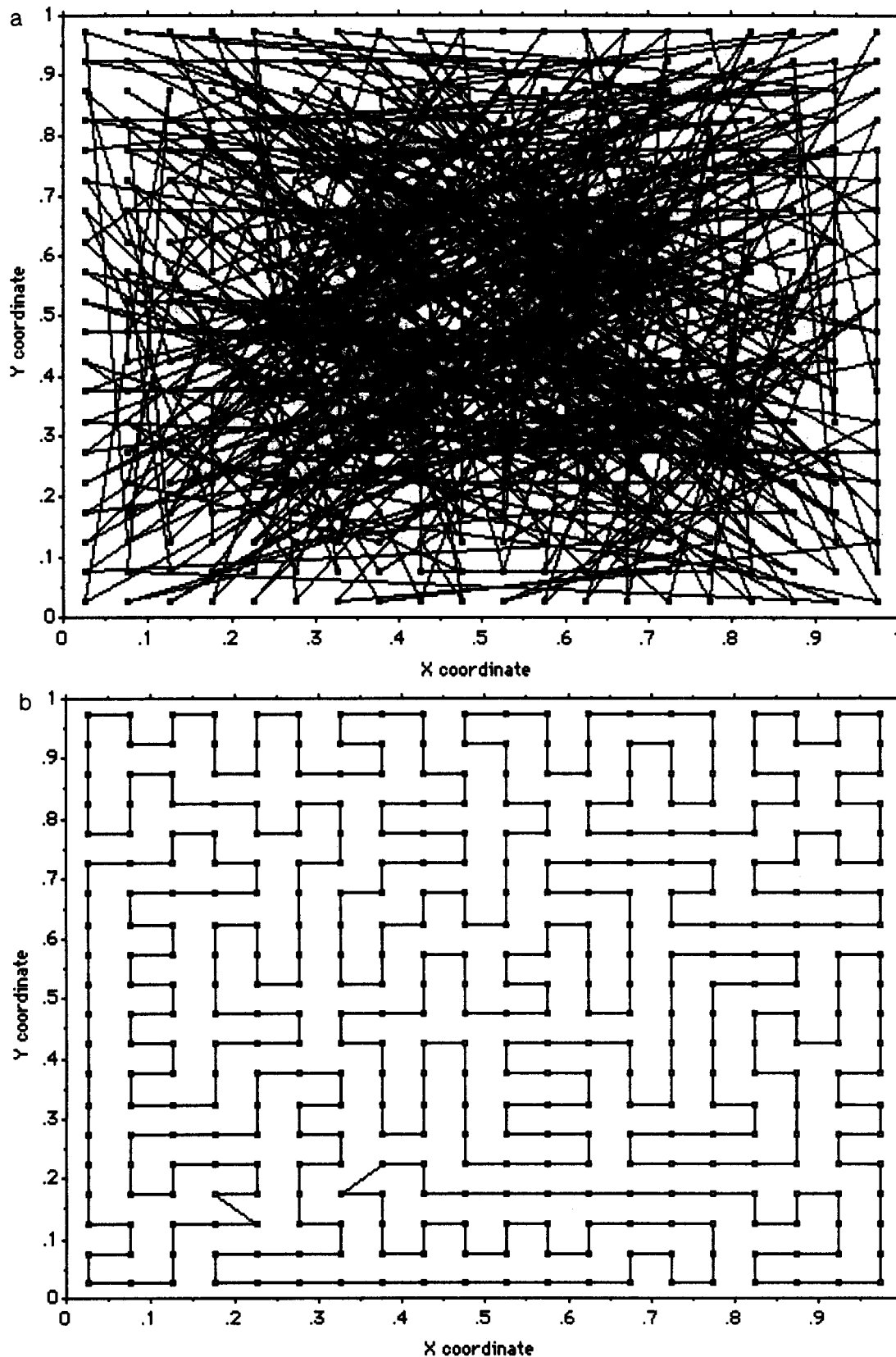


Figure 2. (a) Starting configuration for the 400-city traveling salesman problem (path = 10.731). (b) Best configuration obtained for the 400-city traveling salesman problem (path = 1.0021).

unmodified Metropolis or Glauber algorithms. Also, the NESAs are much less sensitive to the cooling parameter. A much simpler problem (36 cities) was also solved with the four algorithms (Cardoso et al., 1993; Salcedo et al., 1993), and the NESAs again showed a faster approach to the global optimum.

The profiles presented in Figure 3 also show that

improvements in the stopping criterion for the NESAs may substantially reduce the number of function evaluations necessary to obtain a near-optimum path, in contrast to the original algorithms. This point will be discussed later in the paper.

Figure 4 shows typical annealing curves for this 400-city problem. The NESAs are characterized by

Table 1. Statistical Summary of Results (400-City Problem) (N_{FOBJ} = Average Number of Function Evaluations)

algorithm	δ	best path	average path	std dev	N_{FOBJ}	CPU time (min)
Metropolis	10	1.0041	1.0106	0.0028	9 675 788	6.8
NESA/Metropolis	0.072	1.0062	1.0093	0.0023	8 613 348	5.3
Glauber	10	1.0103	1.0155	0.0037	7 311 630	6.1
NESA/Glauber	0.072	1.0062	1.0133	0.0038	7 295 739	5.0
Metropolis	1	1.0041	1.0083	0.0028	19 230 210	14.9
NESA/Metropolis	0.028	1.0041	1.0093	0.0038	22 092 332	12.3
Glauber	1	1.0062	1.0131	0.0031	18 052 114	15.6
NESA/Glauber	0.030	1.0014	1.0152	0.0032	20 257 499	13.0
Metropolis	0.1	1.0021	1.0060	0.0025	121 568 809	95.7
NESA/Metropolis	0.0028	1.0041	1.0084	0.0031	137 809 622	73.8
Glauber	0.1	1.0083	1.0132	0.0032	115 078 128	100.6
NESA/Glauber	0.0030	1.0018	1.0134	0.0019	134 239 437	83.1

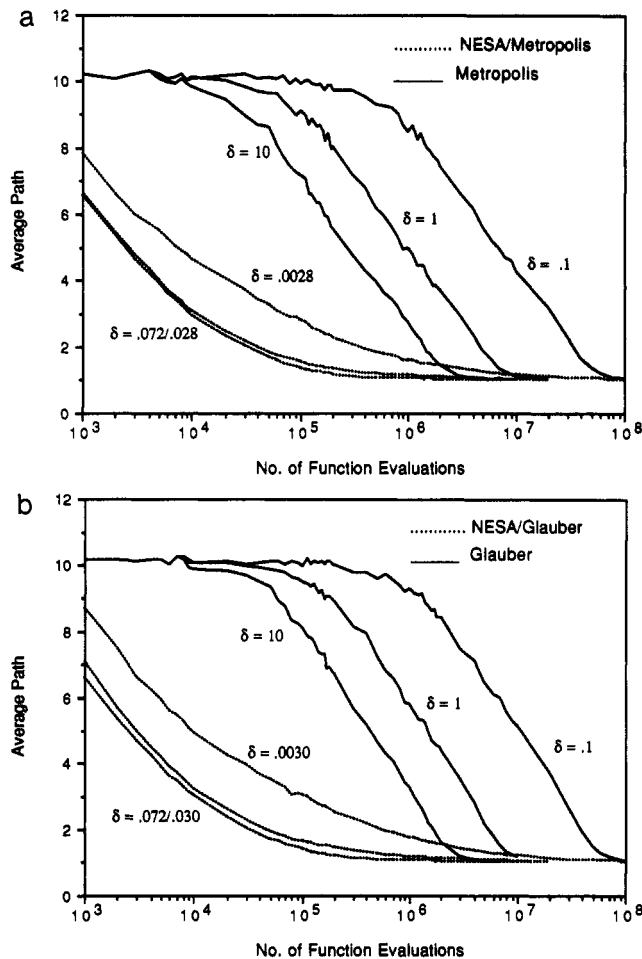


Figure 3. (a) Average paths for the 400-city traveling salesman problem with the Metropolis and NESA/Metropolis algorithms, as a function of the parameter δ . (b) Average paths for the 400-city traveling salesman problem with the Glauber and NESA/Glauber algorithms, as a function of the parameter δ .

smoother cooling schedules near the beginning of the optimization, a faster cooling schedule in a second phase, and, finally, cooling schedules approaching those of the unmodified algorithms. It is important to note that, for small values of δ , the cooling schedule of the NESA variants are similar to those of the unmodified algorithms. Thus, by choosing a sufficiently small value for δ , one can use the NESA variants with cooling schedules which are typical of the Metropolis or Glauber algorithms. The advantage of this will be apparent when discussing improvements in the stopping criterion.

Figure 5 shows how the number of function evaluations at each temperature level (internal iterations) increases for the NESA variants, with the progress of the optimization (global iterations or temperature reductions). For

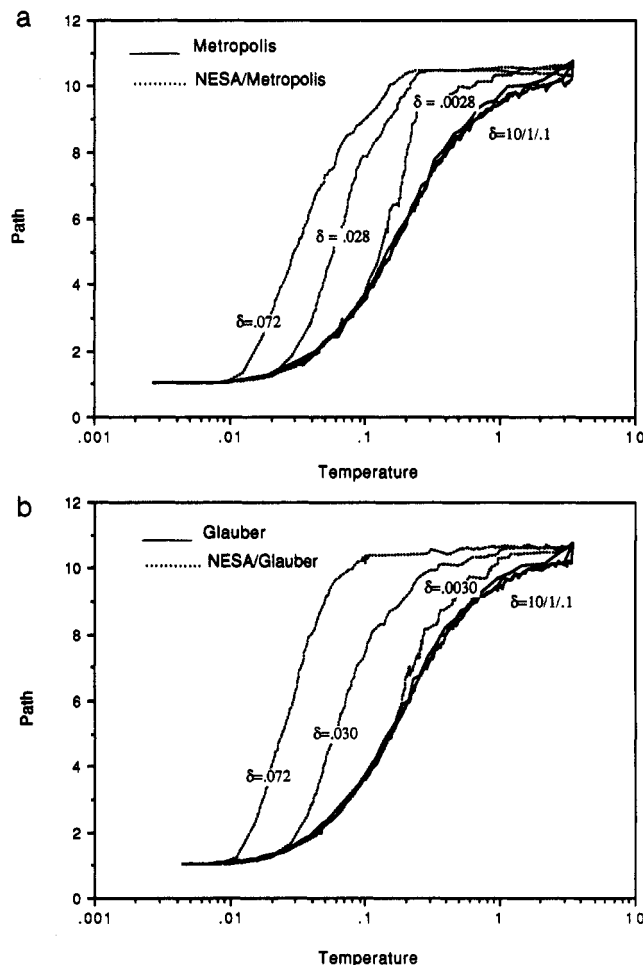


Figure 4. (a) Typical annealing schedules for the 400-city traveling salesman problem with the Metropolis and NESA/Metropolis algorithms, as a function of the parameter δ . (b) Typical annealing schedules for the 400-city traveling salesman problem with the Glauber and NESA/Glauber algorithms, as a function of the parameter δ .

the unmodified algorithms this is constant (100×400 cities) and independent of the temperature level. Thus, accepting that equilibrium may be reached at each temperature level with the unmodified algorithms, it is apparent that such does not occur with the NESA variants, at least during a significant part of the optimization.

Optimization of a Pressure Relief Header Network. Next, we examine the problem of optimizing a pressure relief header network, as shown in Figure 6, a problem with a greater significance to chemical engineering. The idea is to transport fluids most economically from process equipment to a flare head. In order to achieve sonic transport velocities during some transitory period, the pressure on the downstream side of the valves available

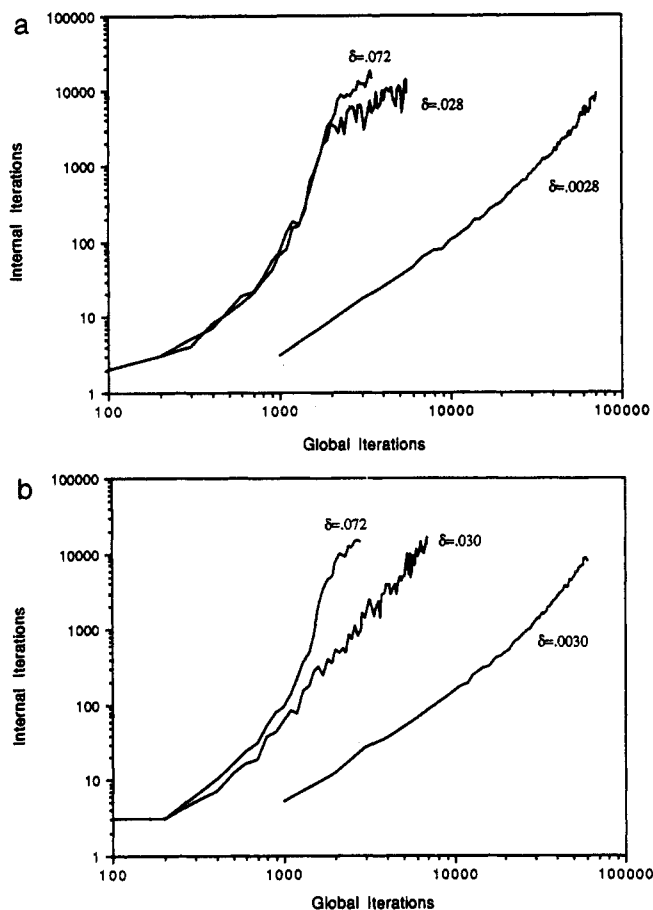


Figure 5. (a) Typical evolution of the number of function evaluations with the NESAs/Metropolis algorithm for the 400-city traveling salesman problem. (b) Typical evolution of the number of function evaluations with the NESAs/Glauber algorithm for the 400-city traveling salesman problem.

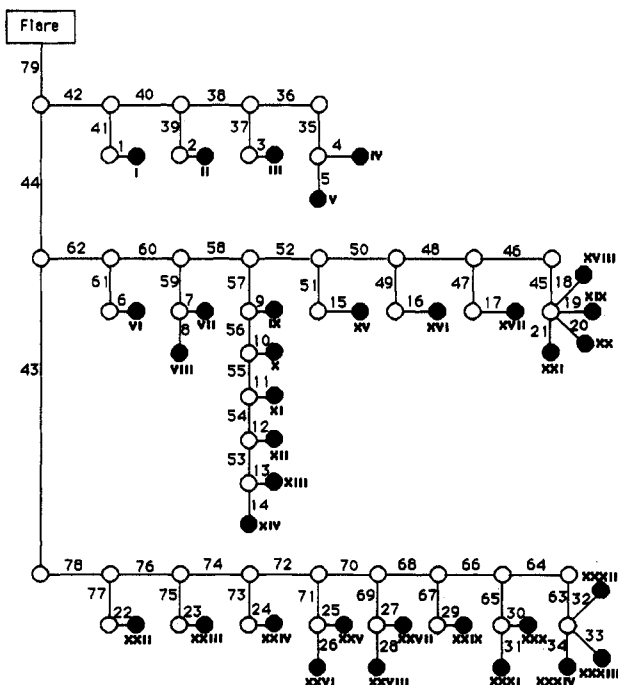


Figure 6. Pressure relief header network.

for flow control must be maintained below certain values (Dolan et al., 1989). This lower limit is determined by process conditions and valve specifications, which set maximum allowable pressure drops between downstream sides of valves and the flare. These constitute a set of

constraints that must be included in the optimization problem.

Cheng (1976), Cheng and Mah (1976), and Dolan et al. (1989) give a full description of the process. There are 79 fixed-length pipe segments and 34 fluid source nodes. The decision variables are the pipe diameters, which may take 1 of nine available discrete values, making the number of possible unconstrained designs 9^{79} , which is indeed a very large number. The pressures in the upstream (A) and downstream (B) end of a pipe segment are approximated by

$$P_A^2 - P_B^2 = \frac{0.3697f(L + L_e)W^2RT}{\pi^2Md^5} \quad (5)$$

where L is the physical length of the pipe section, L_e is the equivalent length, W is the mass flow rate, T is the temperature, M is the molecular weight, d is the pipe diameter, R is the gas constant, and f is a nondimensional friction factor given by

$$f = 0.0475(1.1744 \times 10^{-12}W/\mu d)^{-0.186} \quad (6)$$

where μ is fluid viscosity. The temperatures and molecular weights corresponding to the various pipe sections are determined by a simple mixing rule (Dolan et al., 1989). The objective of this optimization problem is to minimize the total cost

$$C = (\alpha + \beta d)L \quad (7)$$

where α and β are constants, subject to the pressure constraints

$$b_i \geq \sum (P_A^2 - P_B^2) \quad (8)$$

where the index refers to node i and the summation is carried out between all sets of pipes between node i and the flare.

Cheng (1976) has solved this problem with a discrete merging technic, where lists of pipe sections are generated by serial and parallel merging. Dolan et al. (1989) give the same expression for the pressure drop as eq 5 but with a different numerical coefficient in eq 6 (2.8186×10^6 instead of 1.1744×10^{-12}). Since Cheng (1976) has employed British units for this problem, we have rederived the equations within the International System (SI) of units, and came up with eq 6. Also, there is a difference between the physical length for branch 40 in the data reported by Cheng (1976) and by Dolan et al. (1989), namely 19.81 vs 20.73 m. We solved this problem with Cheng's original formulation and corresponding data and with our rederived equations in the SI system of units and could reproduce the cost values obtained by Cheng. However, the solution reported by Cheng (1976) is infeasible since it violates all but the pressure constraint corresponding to the first node. We could also reproduce the solution reported by Dolan et al. (1989) by considering the data used by these authors for the physical length of branch 40. Contrary to the solution reported by Cheng (1976), the solution reported by Dolan et al. (1989) is feasible, since there is only 1 pressure constraint corresponding to node 12 which is violated by a very small amount, namely 0.6%. This could be due either to round-off error or a slightly relaxed acceptance criteria. In any case, totally different results are obtained if the friction factor expression given by Dolan et al. (1989) is employed.

To solve this problem with simulated annealing, we have followed the approach of Dolan et al. (1989), but considering the physical length of branch 40 as given by Cheng (1976). Starting from a fixed network configuration, where

Table 2. Statistical Summary of Results (Pressure Relief Header Network) (N_{FOBJ} = Average Number of Function Evaluations)

algorithm	δ	best cost	average cost	std dev	N_{FOBJ}	CPU time (min)
Metropolis	1	165 281	165 946	407	762 903	0.37
NESA/Metropolis	0.002	165 250	167 135	4006	835 432	0.36
Glauber	1	165 376	166 272	456	723 008	0.39
NESA/Glauber	0.003	165 308	167 370	2681	739 845	0.39
Metropolis	0.1	165 018	165 353	177	5 024 163	2.36
NESA/Metropolis	0.00020	165 096	165 563	1634	5 694 154	2.42
Glauber	0.1	165 101	165 448	199	4 551 506	2.38
NESA/Glauber	0.00038	165 113	165 517	769	4 824 825	2.49
Metropolis	0.01	165 028	165 212	96	45 406 830	21.43
NESA/Metropolis	0.00002	165 081	165 212	61	41 291 403	17.71
Glauber	0.01	165 055	165 248	98	41 238 869	21.56
NESA/Glauber	0.00003	165 102	165 316	234	38 306 056	19.72

Table 3. Discrete Merge and Simulated Annealing Solutions to Pressure Relief Header Network (Cost = 165 018)

branch	1	2	3	4	5	6	7	8	9	10	11	12
pipe (discrete merge ^a)	1	2	2	1	2	1	1	1	1	1	1	1
pipe (simulated annealing)	1	2	2	1	2	1	1	1	1	1	1	1
branch	13	14	15	16	17	18	19	20	21	22	23	24
pipe (discrete merge ^a)	2	2	2	2	1	1	1	1	1	1	1	1
pipe (simulated annealing)	2	2	2	2	2	2	1	1	1	3	1	1
branch	25	26	27	28	29	30	31	32	33	34	35	36
pipe (discrete merge ^a)	1	1	1	1	2	1	2	1	1	1	2	2
pipe (simulated annealing)	1	1	1	1	2	2	2	1	1	1	2	2
branch	37	38	39	40	41	42	43	44	45	46	47	48
pipe (discrete merge ^a)	2	3	2	3	1	3	4	4	1	2	1	2
pipe (simulated annealing)	2	3	2	3	1	3	4	6	1	1	1	2
branch	49	50	51	52	53	54	55	56	57	58	59	60
pipe (discrete Merge ^a)	2	3	2	3	2	2	2	3	2	4	1	4
pipe (simulated annealing)	2	3	2	3	2	2	2	3	2	4	1	4
branch	61	62	63	64	65	66	67	68	69	70	71	72
pipe (discrete merge ^a)	1	4	2	2	2	3	2	3	1	4	1	3
pipe (simulated annealing)	1	4	2	2	2	3	2	3	1	3	1	3
branch	73	74	75	76	77	78	79					
pipe (discrete merge ^a)	1	3	1	4	2	3	7					
pipe (simulated annealing)	3	3	1	4	2	3	7					

^a Dolan et al., 1989.

all pipe diameters are set to the largest possible value, a random number is used to select a pipe segment. A second random number is used to determine whether the pipe segment will have its diameter increased or decreased by 1 unit (in the discretized table of pipe diameters). The Metropolis or Glauber algorithms in their original and modified (NESA) forms are used to decide upon the acceptance/rejection of the proposed configuration. The constraints are then checked for feasibility.

According to Dolan et al. (1989), the number of moves attempted at each temperature level need only be as large as the number of reachable states, which in this case is 2×79 segments, since the outcome for the random number may be one of two states: to increase or decrease its diameter from its current value. Dolan et al. (1989) used an exponentially decreasing cooling schedule, which we have found for other problems to be inferior to the cooling schedule given by eq 4 and employed in this work, a reasoning which is supported by other researchers (Das et al., 1990; Dolan et al., 1990; Patel et al., 1991). We have solved this problem with the conditions used by Dolan et al. (1989), namely 158 moves allowable in the inner loop (Figure 1) and the initial temperature set to 10 000, using the exponential cooling schedule and as stopping criteria the simultaneous failure to obtain an alternative solution in the inner loop and the temperature dropping below 1. However, we were not able to obtain the solutions reported, namely around \$165 000, and higher costs were obtained, of the order reported by Cheng (1976), namely around

\$200 000. The same applies with the Aarst and van Laarhoven (1985) cooling schedule given by eq 4, where costs below \$184 000 could not be obtained even by reducing δ to 10^{-5} . Thus, we have arbitrarily increased the maximum number of moves within each temperature level to 100 times the number of length segments, while keeping δ as a variable parameter.

For a first comparative evaluation of algorithm performance, once again the initial step was to compare attainable results for the same number of cost function evaluations. Table 2 gives a summary of the statistical evaluation of the four algorithms. It shows that very good solutions could be found by all four algorithms within a reasonable execution time. Table 3 gives the pipe diameters corresponding to the best solution obtained, together with the solution of Dolan et al. (1989). This solution does not give the exact cost of \$165 075 reported by these authors since the physical length of branch 40 does not correspond to the data given by Cheng (1976) which were employed in this work. Again, the Metropolis algorithm, whether in its original or modified NESA form, seems to perform better than the Glauber algorithm.

Considering now trajectory evaluation, Figure 7 shows the average costs obtained with the four algorithms as a function of the incremental number of function evaluations and of the cooling parameter δ . As happened with the traveling salesman problem, the NESA variants approach the global minimum much faster than the original Metropolis or Glauber algorithms.

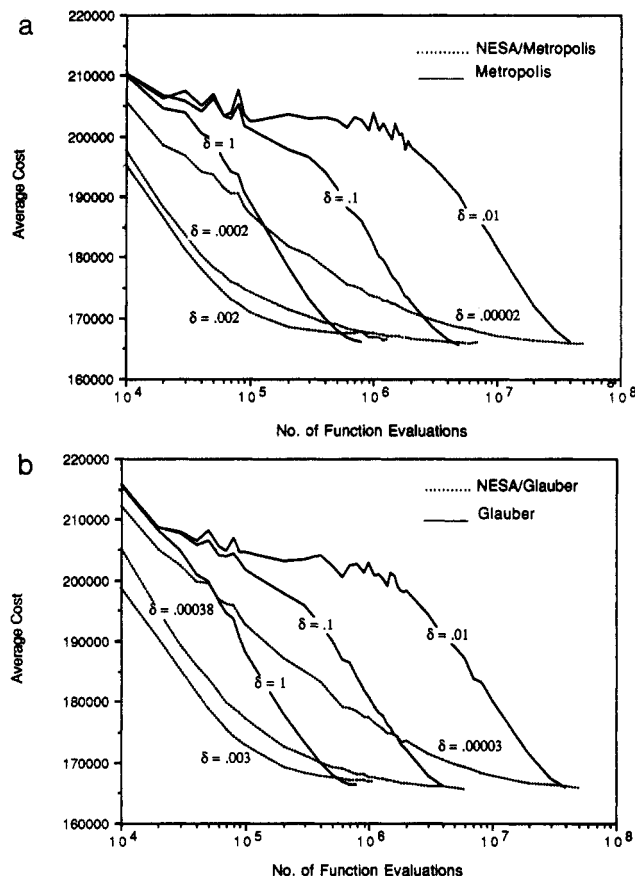


Figure 7. (a) Average paths for the pressure relief header network problem with the Metropolis and NESA/Metropolis algorithms, as a function of the parameter δ . (b) Average paths for the pressure relief header network problem with the Glauber and NESA/Glauber algorithms, as a function of the parameter δ .

Figure 8 shows typical cooling schedules. These curves are again smoother for the NESA variants near the beginning of the optimization and faster afterward, eventually collapsing into the unmodified curves. Again, it is clear that by choosing a small value for δ , one can use the NESA variants with cooling schedules which are typical of the Metropolis or Glauber algorithms. Figure 9 shows how the number of function evaluations at each temperature level increases for the NESA variants with the progress of the optimization, viz., an approximate power law. For the unmodified algorithms this is constant (100×79 pipes) and independent of the temperature level. Thus, it is again apparent that equilibrium is not probably reached with the NESA variants, at least during a significant part of the optimization.

Improving the Termination Criterion. For the NESA algorithms, Figures 3 and 7 suggest that a termination criterion based on the gradient of the objective function with respect to the number of function evaluations may substantially decrease the CPU time without greatly affecting the quality of the attained solutions. To implement such a criterion while smoothing out fluctuations in the objective function, groups of 1000 values were averaged. The normalized gradient was computed from these averages, as follows:

$$\frac{dC^*}{C^* dN} = \frac{C^*_i - C^*_{i-1}}{C^*_i(N_i - N_{i-1})} < \epsilon \quad (9)$$

where $0 < \epsilon \ll 1$, N_i is the cumulative number of function evaluations at iteration i , and C^*_i is the respective averaged cost. Since the factor $N_i - N_{i-1}$ is constant (in our case

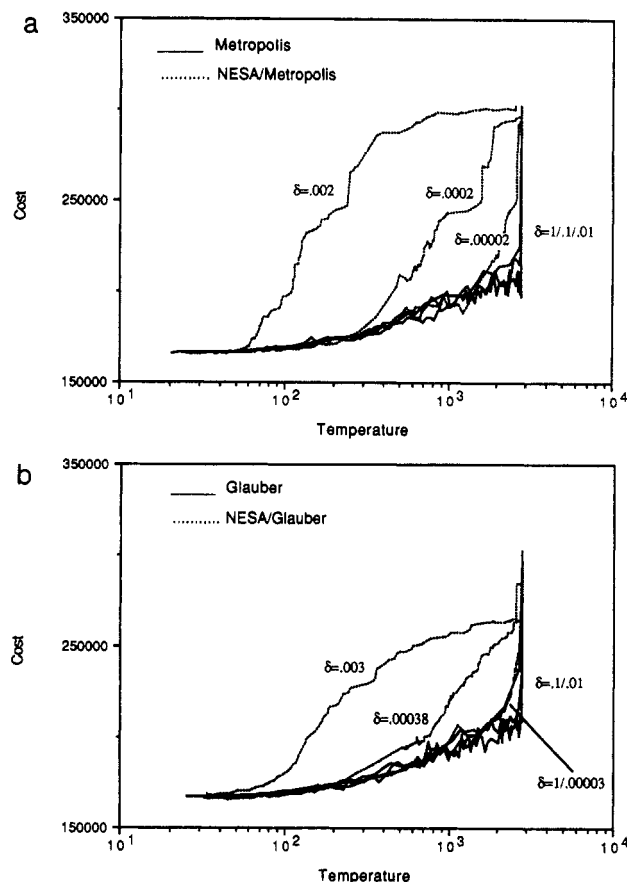


Figure 8. (a) Typical annealing schedules for the pressure relief header network problem with the Metropolis and NESA/Metropolis algorithms, as a function of the parameter δ . (b) Typical annealing schedules for the pressure relief network problem with the Glauber and NESA/Glauber algorithms, as a function of the parameter δ .

equal to 1000), the gradient given by eq 9 is simply an expression for the relative error in the objective function, averaged over a fixed number of function evaluations.

This termination criterion was applied to both problems for the NESA variant of the Metropolis algorithm. Three values for the error term were considered for each problem, and the results are given in Tables 4 and 5 for a particular value of the cooling parameter. For the more stringent values of the error term the number of function evaluations required by the NESA algorithms is high. For similar computational effort the original algorithm is able to produce the same kind of results. However, relaxing the error (see $\epsilon = 10^{-5}$ or $\epsilon = 10^{-6}$ for the traveling salesman study and $\epsilon = 10^{-4}$ or $\epsilon = 10^{-5}$ for the pressure relief study), the NESA algorithm still produces good results with a fraction of the effort whereas for the same level of function evaluations the original algorithm gives much poorer results. Similar conclusions are obtained with the other values of the parameter δ .

The decrease in computational burden may be achieved with the unmodified algorithms by specifying a larger value for the cooling parameter δ or an aggressive cooling schedule, such as an exponential one. However, it is known that large values for δ usually produce poor solutions and as such should be avoided (Dolan et al., 1989; Das et al., 1990; Patel et al., 1991). In the NESA variants, the decrease in computational burden may be achieved through the error criterion ϵ , while employing very small values for the cooling parameter δ . This will produce cooling schedules which are similar to those of the unmodified algorithms, as shown in Figures 4 and 8.

While the reduction in CPU time may not be very

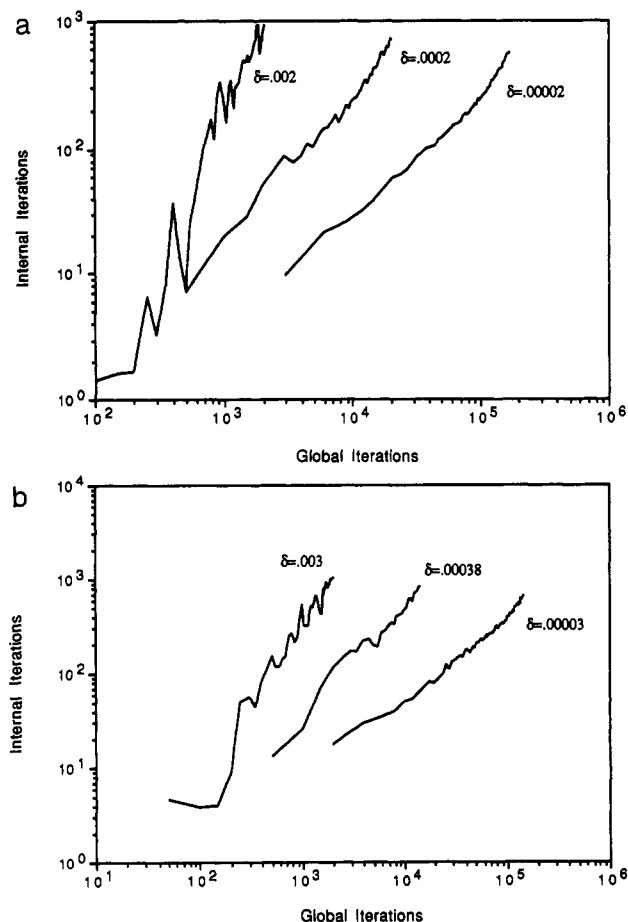


Figure 9. (a) Typical evolution of the number of evaluations with the NESAs/Metropolis algorithm for the pressure relief header network problem. (b) Typical evolution of the number of function evaluations with the NESAs/Glauber algorithm for the pressure relief header network problem.

Table 4. Performance of Algorithms with Stopping Criteria ϵ (400-City Problem) (N_{FOBJ} = Average Number of Function Evaluations)

algorithm	δ	best path	average path	std dev	N_{FOBJ}	ϵ
Metropolis	10	4.3047	4.9136	1.4455	296 480	
NESAs/Metropolis	0.072	1.0537	1.1581	0.0779	328 200	10^{-5}
Metropolis	10	1.0933	1.1488	0.0831	2 871 980	
NESAs/Metropolis	0.072	1.0103	1.0482	0.0441	3 056 400	10^{-6}
Metropolis	10	1.0083	1.0120	0.0022	7 511 536	
NESAs/Metropolis	0.072	1.0062	1.0198	0.0203	7 753 739	10^{-7}

Table 5. Performance of Algorithms with Stopping Criteria ϵ (Pressure Relief Header Network) (N_{FOBJ} = Average Number of Function Evaluations)

algorithm	δ	best cost	average cost	std dev	N_{FOBJ}	ϵ
Metropolis	1	185 037	190 987	4848	59 144	
NESAs/Metropolis	0.002	166 765	175 853	7988	63 280	10^{-4}
Metropolis	1	171 761	176 873	2991	189 457	
NESAs/Metropolis	0.002	165 624	169 950	5871	197 451	10^{-5}
Metropolis	1	166 565	169 220	1962	397 866	
NESAs/Metropolis	0.002	165 271	168 234	5142	402 491	10^{-6}

important for the type of problem dimension presented and employing a fast computer, the use of slower computational means, such as personal computers, or the optimization of much more demanding combinatorial minimization problems, such as the design of VLSI circuits, which may take several hours even on very fast computers (Huang et al., 1986), may constitute interesting grounds

for applying the modified NESAs algorithms, at least where a near global optimum solution is acceptable.

Conclusions and Significance

Nonequilibrium simulated annealing algorithms (NESAs) were tested based on a simple modification of the Metropolis and Glauber algorithms, whereby enforcement of the cooling schedule is promoted as soon as an improved or accepted solution is obtained. This modification conceptually destroys the condition of near thermal equilibrium on which simulated annealing is based. This is especially noticeable near the starting system configuration, which however is expected to be far from the global optimum. With the progress of the optimization, the number of function evaluations at each temperature shows an approximate power law increase with each temperature reduction, and the acceptance of a new solution eventually becomes controlled by the probability distribution assumed in the base algorithm.

The application of the algorithms to the solution of a 400-city traveling salesman problem and to the optimization of a pressure relief header network revealed that the Metropolis algorithm, whether in its original or modified (NESAs) form, always performed better than its Glauber variant, in agreement with the findings of Das et al. (1990) and Patel et al. (1991) for other types of combinatorial minimization problems. Also, the NESAs variants are roughly equivalent to the unmodified algorithms, for the same number of function evaluations, employing as stopping criterion either the derivative of the smoothed average cost at a temperature level with respect to that temperature or specifying that at least 1 global iteration should occur without acceptance of any moves.

To reduce the computational burden of the Metropolis or Glauber algorithms, aggressive cooling schedules may be enforced. This should however be avoided since it may produce poor results or convergence to poor local optima (Dolan et al., 1989; Das et al., 1990; Patel et al., 1991). With the NESAs algorithms, a substantial reduction (about 50%) in CPU time can be achieved without the adoption of aggressive cooling schedules. Such is possible through the imposition of a termination criterion based on the gradient of the relative error of the objective function with respect to the number of function evaluations. Since the annealing curves for the NESAs variants approach those of the unmodified algorithms for small values of the cooling parameter, it is expected that the adoption of a slow cooling schedule coupled with a reasonable error criterion may result in near global optimum solutions in a reduced CPU time. This should be especially important where the computational means are slow or for the optimization of computationally demanding problems.

Acknowledgment

This work was partially supported by JNICT, under Grant No. BD/1457/91-RM, employing the computational facilities of Instituto de Sistemas e Robotica (ISR)—Porto.

Nomenclature

- b_i = pressure constraint of node i (Pa^2)
- d = pipe segment inside diameter (m)
- C^* = cost averaged over a fixed number of function evaluations
- ΔC = difference in cost between new and current configurations
- Δf^+ = average increase in cost for the m_2 unsuccessful moves
- E_i = energy state of system configuration i
- f = friction factor

K_B = Boltzmann's constant
 L = physical length of pipe segment
 L_e = equivalent length of pipe segment
 M = molecular weight (kg/kg-mol)
 N_i = cumulative number of function evaluations at iteration i
 m_1 = number of successful moves
 m_2 = number of unsuccessful moves
 P = probability of acceptance/rejection of current solution
 p_A = upstream pressure of pipe segment (Pa)
 p_B = downstream pressure of pipe segment (Pa)
 R = universal gas constant (J/(kg-mol-K))
 T = absolute temperature of stream formed by mixing (K)
 T_i = system (annealing) temperature of configuration i (K)
 W = mass flow rate (kg/s)
 X = acceptance ratio

Greek Symbols

α = cost coefficient (\$/m)
 β = cost coefficient (\$/m²)
 δ = cooling rate control parameter
 ϵ = error criterion (relative)
 μ = gas viscosity (kg/(m-s))
 σ = standard deviation of all cost functions at current temperature

Literature Cited

- Aarst, E. H. L.; van Laarhoven, P. J. M. Statistical cooling: a general approach to combinatorial optimization problems. *Philips J. Res.* 1985, 40, 193-226.
- Aarst, E.; Korst, J. *Simulated Annealing and Boltzmann Machines—A Stochastic Approach to Combinatorial Optimization and Neural Computers*; Wiley: New York, 1989.
- Anily, S.; Federgruen, A. Simulated annealing methods with general acceptance probabilities. *J. Appl. Probab.* 1987, 24, 657-667.
- Bobachevsky, I. O.; Johnson, M. E.; Stein, M. L. Generalized simulated annealing for function optimization. *Technometrics* 1986, 28 (3), 209-217.
- Cardoso, M. F.; Salcedo, R. L.; Feyer de Azevedo, S. A fast simulated annealing algorithm for combinatorial minimization. *Chempor'93*, Oporto; International Chem. Eng. Conference; Rodrigues, A. E. Ed.; 1993; pp 373-380.
- Cheng, W. B. Computational techniques for simple pipeline networks design. M.Sc. Thesis, Northwestern University, 1976.
- Cheng, W. B.; Mah, R. S. H. Optimal design of pressure relieving piping networks by discrete merging. *AIChE J.* 1976, 22 (3), 471-476.
- Conoly, D. General purpose simulated annealing. *J. Opt. Res. Soc.* 1992, 43 (5), 495-505.
- Corana, A.; Marchesi, M.; Martini, C.; Ridella, S. Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. *ACM Trans. Math. Software* 1987, 13 (3), 263-280.
- Corey, E. M.; Young, D. A. Optimization of physical data tables by simulated annealing. *Comput. Phys.* 1989, 3 (3), 33-37.
- Das, H.; Cummings, P. T.; Levan, M. D. Scheduling of serial multiproduct batch processes via simulated annealing. *Comput. Chem. Eng.* 1990, 14 (12), 1351-1362.
- Dolan, W. B.; Cummings, P. T.; Levan, M. D. Process optimization via simulated annealing: application to network design. *AIChE J.* 1989, 35 (5), 725-736.
- Dolan, W. B.; Cummings, P. T.; Levan, M. D. Algorithmic efficiency of simulated annealing for heat exchanger network design. *Comput. Chem. Eng.* 1990, 14 (10), 1039-1050.
- Faigle, U.; Kern, W. Note on the convergence of simulated annealing algorithms. *SIAM J. Control Optim.* 1991, 29 (1), 153-159.
- Glauber, R. J. Time-dependent statistics of the Ising model. *J. Math. Phys.* 1963, 4, 294.
- Greene, J. W.; Supowit, K. J. Simulated annealing without rejected moves. *IEEE Trans. Comput. Aided Des.* 1986, CAD-5 (1), 221-228.
- Groisman, G.; Parker, J. R. Computer-assisted photometry using simulated annealing. *Comput. Phys.* 1993, 7 (1), 87-96.
- Huang, M. D.; Romeo, F.; Sangiovanni-Vicentelli, A. An efficient general cooling schedule for simulated annealing. *IEEE Trans. Comput. Aided Des.* 1986, 381-384.
- Johnson, D. S.; Aragon, C. R.; McGeoch, L. A.; Schevon, C. Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. *Oper. Res.* 1989, 37 (6), 865-892.
- Johnson, D. S.; Aragon, C. R.; McGeoch, L. A.; Schevon, C. Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. *Oper. Res.* 1991, 39 (3), 378-406.
- Kirkpatrick, S. Optimization by simulated annealing: quantitative studies. *J. Stat. Phys.* 1984, 34 (5/6), 975-986.
- Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. Optimization by simulated annealing. *Science* 1983, 220, 671-680.
- Ku, H.; Karimi, I. An evaluation of simulated annealing for batch process scheduling. *Ind. Eng. Chem. Res.* 1991, 30 (1), 163-169.
- Lin, S.; Kernighan, B. W. An effective heuristic algorithm for the traveling salesman problem. *Oper. Res.* 1973, 21 (2), 498-516.
- Metropolis, N.; Rosenbluth, A.; Rosenbluth, M.; Teller, A.; Teller, E. Equation of state calculations by fast computing machines. *J. Chem. Phys.* 1953, 21, 1087-1092.
- Patel, A. N.; Mah, R. S. H.; Karimi, I. A. Preliminary design of multiproduct noncontinuous plants using simulated annealing. *Comput. Chem. Eng.* 1991, 15 (7), 451-469.
- Percy, M. Simulated annealing for optimization problems. *C Users J.* 1992, July, 71-84.
- Press, W. H.; Teukolsky, S. A. Simulated annealing optimization over continuous spaces. *Comput. Phys.* 1991, 5 (4), 426-429.
- Press, W. H.; Flannery, B. P.; Teukolsky, S. A.; Vetterling, W. T. *Numerical Recipes: The Art of Scientific Computing*; Cambridge University Press: New York, 1986; pp 326-334.
- Salcedo, R. L. Solving nonconvex nonlinear programming and mixed-integer nonlinear programming problems with adaptive random search. *Ind. Eng. Chem. Res.* 1992, 31 (1), 262-273.
- Salcedo, R.; Gonçalves, M. J.; Feyer de Azevedo, S. An improved random-search algorithm for nonlinear optimization. *Comput. Chem. Eng.* 1990, 14 (10), 1111-1126.
- Salcedo, R. L.; Cardoso, M. F.; Feyer de Azevedo, S. A fast simulated annealing algorithm for combinatorial minimization. *Proceedings Escape 3*, Graz, Austria; Moser, F., Schnitzer, H., Bart, H. J., Eds.; 1993; Suppl. Vol., pp 12-17.
- Shedler, G. S. Generation methods for discrete event simulation. In *Computer Performance Modeling Handbook*; Academic Press: 1983; pp 227-251.
- Stanley, R.; Sherman, G. Discrete optimizing. *J. Soc. Ind. Appl. Math.* 1965, 13, 864-889.
- Vanderbilt, D.; Louie, S. G. A Monte Carlo simulated annealing approach to optimization over continuous variables. *J. Comput. Phys.* 1984, 56, 259-271.
- White, S. R. Concepts of scale in simulated annealing. *IEEE Trans. Comput. Aided Des.* ICCD 84, New York; 1984; pp 646-651.

Received for review December 2, 1993
 Revised manuscript received May 2, 1994
 Accepted May 9, 1994*

* Abstract published in *Advance ACS Abstracts*, July 1, 1994.