USING SIMULATED ANNEALING TO SOLVE NP-COMPLETE PROBLEMS

التقطيع الصوري باستعمال الاحماء

Dr. Abul-Rahman H. Al-Husaini M. Abdullah Dr. S. K. Majeed
د. عبدالرحمن حامد الحسيني محمد عبدالله مدياني د. سعد كاظم مجيد

Ministry of Higher Education
وزارة التعليم العالي والبحث العلمي

ABSTRACT

Simulated annealing algorithm is a recent powerful technique for solving hard problems. Furthermore, If has a very interesting features: general applicability. In fact, it has been used successfully for solving many NP-Complete Problems.

This paper describes elementary theoretical concepts and Principles of this technique and presents. As well, some examples with results in witch simulated annealing algorithm is used to solve hard problems.

***Key words** : Simulated annealing, NP-Complete, heuristic, algorithms.*

1. Introduction;

In technological systems, the combinatorial Optimization problem (minimization or maximization) is often posed, Unfortunately, many of those problems are NP-Complete [1] [2] whose complexity in terms of computation time or memory size maybe intended (exponential complexity). This problem stands even there exist several algorithms that solve the same problem using different time and space complexity consequence of the trade off between time space [14]. Besides. If a problem is to be solved, one of two classes of algorithms, implied by two different approaches, may be exclusively involved [3]:

a. Optimization algorithms: this class of algorithms means to get the exact best solutions (s), at the risk of very large computation time depending on the problem size. The enumeration or exhaustive algorithms, which pick the best solution by visiting all possible solutions, are an example of this class.
b. Approximation (heuristic) algorithms: these algorithms go for quickly obtainable solutions but not necessarily the global optimum. It may be sub-optimal. Local search (hill climbing) algorithm [13], which accepts only solutions that improve the object function. Nevertheless, it may return a non-global optimum solution if the objective function is non-convex..

Simulated annealing (SA) technique [4] is a very general method inspired from statistical physics and successfully used in optimization problems. In general, it behaves as an approximation algorithm but well choosing its parameters can give in most cases promising solutions.

The main interest of this paper is to present the SA algorithm and to give some examples of its application with particular emphasis on parameter choosing. This paper is organized in two parts. The second section is theoretical presentation of the SA algorithm in which statistical basics that guarantee the convergence of the algorithm are discussed without encountering long calculations. Instead only results that interest the programmers are given. In the third section, programming techniques are presented as well as some examples with results. Finally, a conclusion ends the paper.

11 SA ALGORITHM

There exists an analogy between a physical looking for lowest energetic states and a combinatorial optimization process [5]. In this section a brief description of the physical process is presented firstly. Further. The algorithm's principles and basics are discussed.

1. ***Thermal Annealing***

   The SA algorithm originated from a physical thermal process called annealing ; used to obtain minimum energy crystalline structure of a metal. It is a two-step process [4]:

   - The temperature is increased until the solid is softened and particles are arranged randomly, the temperature should be sufficiently high.
   - The temperature is decreased carefully until particles arrange themselves in the ground state. The cooling should be sufficiently slow.

   Indeed, a double dynamic is contemplated : first, seeking for minima at fixed temperature; second, decreasing the temperature..

   The process of temperature decreasing allows the system to search attraction pools relatively large at the beginning, as well as to avoid being attracted by local minima.

2. ***Simulated Annealing Technique***

   The aforementioned analogy between the way in which a metal cools and the search for a minimum[1]

---

[1]Minimum is used by default with out loosing generality, because a maximization problem can be easily translated into a minimization problem.

In a combinatorial problem can be summarized in the following equivalences [3]:

- Solutions in the optimization problem are equivalent to states (different configurations) of a physical system.
- The cost of a solution is equivalent to the state energy.

  In addition, a parameter called *control parameter* is introduced in order to represent the temperature.

2.1 Theoretical Concepts of SA

In spite of the similarity between the SA algorithm and the hill climbing heuristic [13]. The former is somewhat more advantageous. It has the ability to escape from local minima. Indeed, because a random search is employed, not only changes that principal is referred to as *Metropolis criterion* [6]. Furthermore. Convergence conditions and results from quantitative analysis derived for the SA are based on the *ergodicity hypothesis.*

a. *Metropolis Criterion*

Let f denotes an objective function of a combinatorial optimization problem.

Given two solutions $i. j$ with costs $f(i). f(i)$ respectively. The Metropolis criterion determines whether j is accepted from i by applying the following acceptance probability [3]:

$$P\{\text{accept } j\} = \begin{cases} 1 & \text{if } f(j) \le f(i) \\ e^{\frac{f(j)-f(i)}{c}} & \text{if } f(j) > f(i) \end{cases} \qquad (1)$$

Where *c*, positive non-null real. Denotes the control parameter.

b. Ergodicity Hypothesis

A physical many-particle system is compatible with a statistical ensemble. And the corresponding ensemble averages determine the averages of observable of the physical system then a number of useful quantities can be derived for the system at the equilibrium [7] [3].

This hypothesis, in statistical physics, is used to derive some interesting results and allow an analytic evaluation of statistical quantities (average, spreading..) .

By analogy, all results of equilibrium statistics of a physical system can be applied to the SA algorithm.

c. Acceptance Ratio

Some generated solutions are not accepted, because of the acceptance probability. Then a ratio of acceptance is defind in function of the control parameter as follows [3]:

$$\lambda(c) = \left. \frac{number\ of accepted\ transitions}{number\ of\ proposed\ transitions} \right|_{c} \qquad (2)$$

As result : $O \le \%(C) \le 1, \forall C.$

d. Average Cost and Spreading Cost

For each value of the control parameter `c` a number of iterations `L` is evoked Numerical values of the average cost *f (c) and of the spreading of the cost* σ, *are calculated from the following expressions [3]:*

$$\ddot{f}(c) = \frac{1}{L}\sum_{i=1}^{L} f.(c) \qquad (3)$$

and

$$\sigma(c) = \sqrt{\frac{1}{L}\sum_{i=j}^{L}(f_1(c) - \ddot{f}(c))^2} \qquad (4)$$

2.2 Implementation Techniques

The SA algorithm can be implemented in a general manner. Although, a set of specific elements must be provided with each implementation [8][9]:

- Representation of possible solutions.
- Generator of random changes in solutions.
- An acceptance criterion.
- An annealing schedule.

A. General Structure of the SA Algorithm

The process consists of three parts, initialization and two nested loops, as shown in the basic structure of the algorithm in Figure 1.

The initialization phase is achieved in three non-ordered steps:

- Generating an initial solution (randomly in general) .
- Estimating an initial value of the control parameter (temperature).
- Estimating a number of loops to attend equilibrium at a fixed temperature.

The two nested loops are consequence of the double dynamic used in the thermal annealing process.

The inner loop applies the Metropolis criterion (decision N° 8 in Figure 1) on a generated solution. It holds for a fixed value of temperature determined in the outer loop.

The outer loop is iterated while a stop criterion (decision N°7 in Figure 1) is not true.

In the algorithm, solutions are assessed via the objective function (in procedure N° 4 in figure 1).

Each of operations used will be discussed in the following.

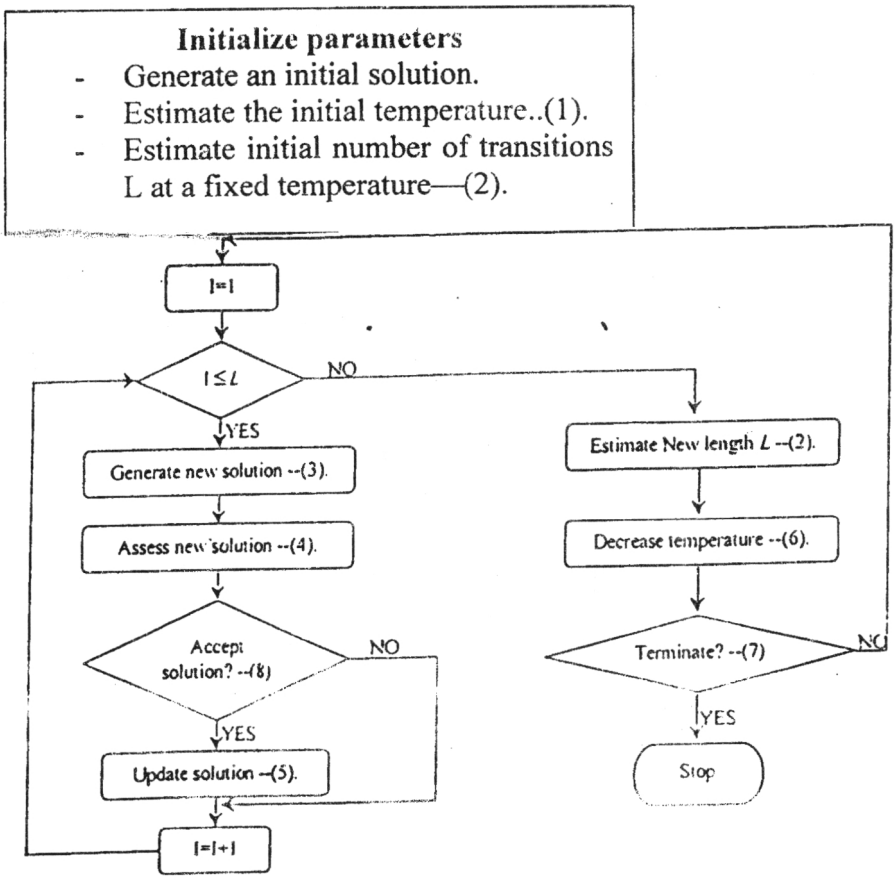


Figure 1 Structure of SA algorithm

B. Solution Representation

The way in which solutions are represented in problem-specific, i.e. for a given optimization problem a solution structure is adopted. In general, the most obvious representation of the solution is usually appropriate.
The solution representation imposes a neighborhood structure. It defines for each solution a set of close (neighbor) solutions; i.e. it determines a topology for solutions.

C. Generation of New Solutions

The mechanism of generating a new solution from a given solution (procedure N° 3 in figure 1) should satisfy the following requirements:

- Generated solution should be a neighbor of the given solution.
- New solution is obtained by introducing small random changes in the given solution [9].
- All possible solutions should be reachable [9].

The generation mechanism is problem-specific. Because it should be, obviously, compatible with the chosen representation.

D. Acceptance Criterion

A rule, which determines whether to accept a generated solution, based on the Metropolis criterion is provided (in procedure N° 4 in Figure 1). It can be defined as follows:
Given two solutions i and j, and let f denotes the objective function, then

If $f(j) \leq f(i)$ then accept j

else

if $e^{\frac{f(i)-f(j)}{c}} > r$ then accept j

Where 'c' is the value of the control parameter and 'r' is a random number generated from a uniform distribution on the interval [0,1].

Note that at large values of c. large number of non-improving solutions is accepted, but as the value of c approaches 0 non-improvement transitions will not be accepted at all. This feature explains why the SA compared to local search algorithms. Can avoid being trapped by local minima.

E. Cooling Schedule

Search in SA algorithm is guided by a mechanism called the cooling schedule. It is extremely important because it specifies the set of parameters that govern the convergence of the algorithm. In fact, the convergence will be guaranteed if those parameters are well chosen.

Determination of values-of parameters is a result of the analogy between thermal annealing and SA and of the quant-itative analysis based on ergodicity hypothesis.

The cooling schedule should specify the following [3]:

- A finite sequence of values of the control parameter:
  - An initial value (procedure 1).
  - A decrement function for decreasing the value of the control parameter (represented by procedure N° 6 in Figure 1).
  - A final value of the control parameter specified by a stop criterion (represented by procedure N°7 in figure 1) .

- A finite number of transitions at each value of the control parameter (represented by procedure N° 2 Figure 1).

E.1 *Initial Value of the Control Parameter*

The acceptance ratio (equation 2) is, generally, the means of finding the initial temperature [3] [8] [9] [10].

An efficient and simple procedure ia as follows[3]:

- The acceptance ratio is set to a fixed initial value λ close to 1.
- Generate a number m_o of trials. Let m_1 the number of cost-decreasing trials and $m_2 = m_0 - m_1$.
- Calculate $\Delta\bar{f}$ the average difference in cost-increasing trials (i.e.in m_2 *trial).*
- Calculate the initial value of the control parameter using the following expression ;

$$c_0 = \frac{\max(0, \overline{\Delta f})}{In\left(\frac{m_2}{m_2\lambda - m_1(1-\lambda)}\right)}$$

E.2 *Decreasing the Temperature*

Usually, small changes in the values of the control parameter are preferred. Many techniques were successfully used for this purpose . Some important methods are [3] [11] [12]:

- *Geometric annealing (* called as well Boltzman annealing):

$c_{k+1} = ac_k \; 0 < a < 1.$

- *Logarithmic annealing*

$$c_k = \frac{c_o}{\ln(k)}$$

- *Fast annealing*

$$c_k = c_o \, e^{(\alpha-1)k} \qquad 0 < \alpha < 1$$

- *Distance annealing*

$$c_{k+1} = \frac{c_o}{1 + \dfrac{c_k \ln(1+\delta)}{3\sigma_{c_j}}}$$

Where
δ is the distance parameter (small δ – values lead to small decrements and vice versa [3]).
σ_{cl} is the spreading of the cost function.

E.3 *Final Value of the Control Parameter*

The termination condition is related to the final value of the control parameter.

Here follows the description of two well-known stop criterion:

1. Let a_1 be the average of the cost on N landings of temperature and a_2 be the average on N next landings. Then the algorithm is terminated if [8]:

$$\frac{a_2 - a_1}{a_2} < \varepsilon_N \qquad (11)$$

2. The algorithm is terminated if for some k the following expression holds [3]:

$$\frac{c_k}{\langle f \rangle_o} \frac{\partial \langle f \rangle_c}{\partial c}\bigg|_{c=cl} < \varepsilon_N \qquad (12)$$

Where

$$\frac{\partial \langle f \rangle_c}{\partial c} \approx \frac{\sigma_c^2}{c^2} \quad < \varepsilon_N \qquad (13)$$

ε_N Is small positive real number. It is referred to as the stop parameter.

E.4 *Number of Transitions of Each Value of the Control Parameter [3] [9]*

Quasi-equilibrium is to restored at each value of the control parameter. To allow this, two different approaches are available:

1. To restore quasi-equilibrium a minimum number of transitions $\eta_{\min}$ should be accepted at each temperature, with an upper bound for the total number of transitions to avoid extremely long loops for small values of the temperature.
2. The number of transitions L is constant and is chosen equal to Θ the size of neighborhoods (number of solutions that can be generated from a solution).

$$L_k = L = \Theta \quad k = 1,2,..... \qquad (14)$$

III Practical Results

Two examples that apply SA to NP-Complete problems are given in this section: the Traveling Salesman Problem (TSP) and a real function optimization. The effect of different parameters is illustrated by curves issued from an application designed for this purpose.

1. *Traveling Salesman Problem*

The TSP is one of the most famous combinatorial optimization problems[15]. An instance of this problem consists of n cities and a matrix $D = (d_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le n}}$ of distances between cities.

The salesman's objective is to make a shortest tour visiting all the cities once and only once before returning to his starting point.

1.1 Problem Complexity

As previously mentioned, this problem is NP-complete; so that any algorithm that attempts to solve it will be of exponential complexity. The most obvious is an exhaustive search through all possible permutations between cities. Obviously, such an algorithm is $O(n!)$ time complexity. Better algorithms may be found; but still all exponential. For example, by a dynamic programming approach an algorithm whose time complexity is $O(n^2 2^n)$ had been designed [16]; but on an other hand its space complexity goes to $O(n2^n)$

1.2 TSP Using SA

In this paragraph, the elements provided with this specific implementation are given.

a. Solution Representation

Several representations can be adopted without reflecting the remaining elements. An example is to represent cities in the order as a string.

In this application, solutions are represented as any array whose I^{th} one

b. **The Cost Function**

Assume *Ar* is a solution array. Cost of this solution can be calculated by referring to distance matrix and using the following expression:

$$\text{cost(Ar)}=\sum_{i=1}^{n} d_{i,Ar[i]}$$

c. ***Generation of New Solutions***

A new solution can be obtained by simply inverting a section of the tour including certain number of cities. In this application, a 2-change strategy [3] is adopted as shown in figure 2.

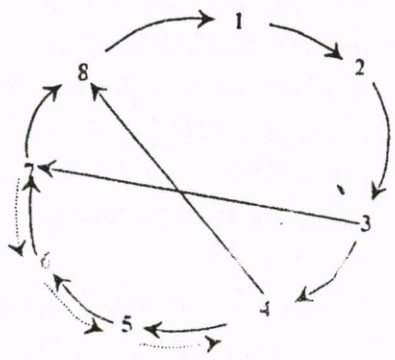


Figure 2 2-change strategy in TSP

d. Cooling Schedule

There is no general way to determine the best values of the parameters that contribute in different equations. Nevertheless, the well-known method for this is tuning (make a large number of trials).

In calculation of the final value of the control parameter, instead of the equation 13 the following expression gave good results (usage of mathematical definition of the derivative) :

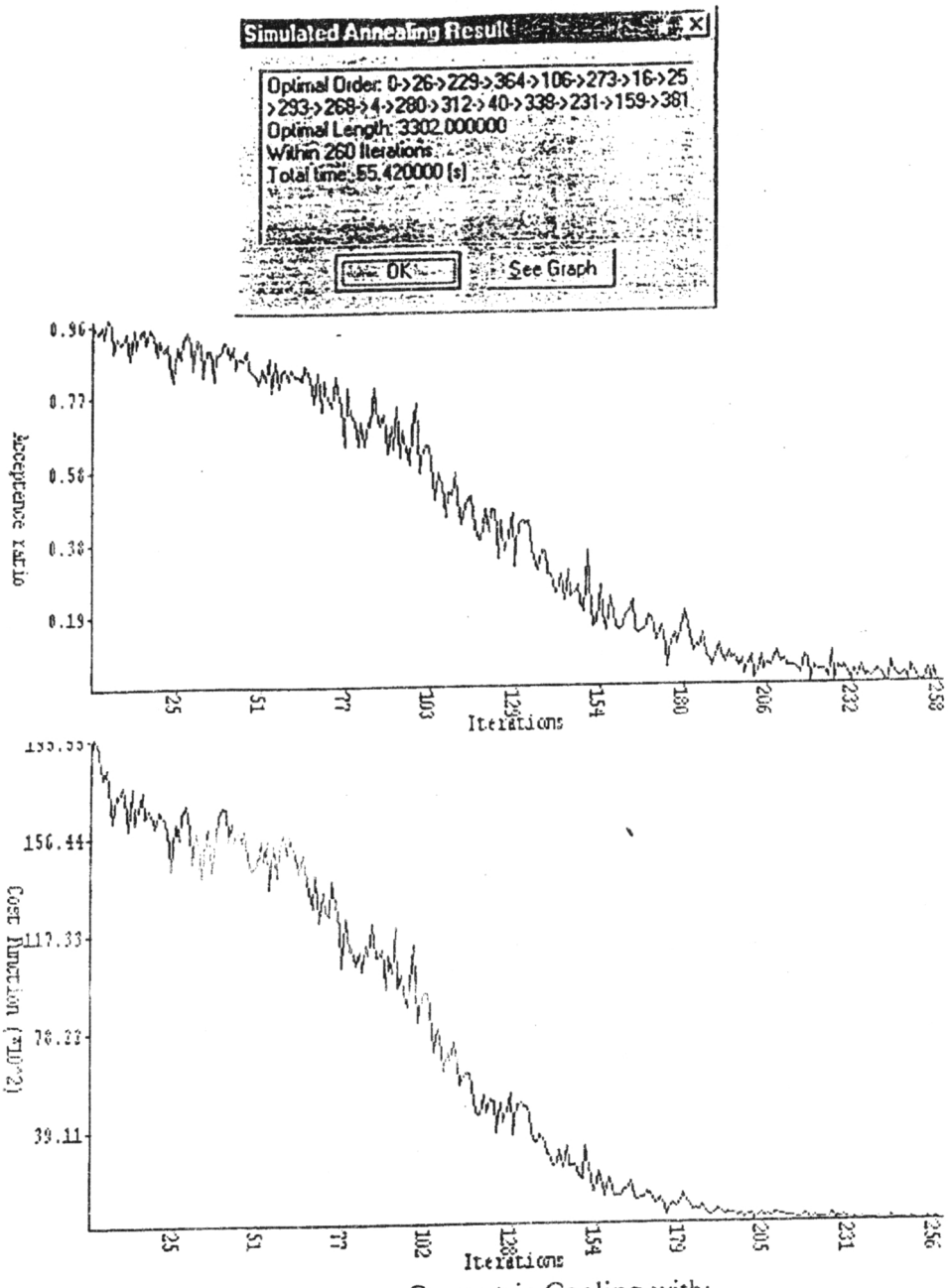
$$\frac{\partial \langle f \rangle_c}{\partial c} = \frac{\langle f \rangle_{c_k} - \langle f \rangle_{c_{k-1}}}{c_k - c_{k-1}} \qquad .....(16)$$

The number of transitions at each value of the control parameter is taken equal to the number of neighborhoods. Adopting a 2-change strategy, it will be the number of possibilities to choose two different non-adjacent cities among *n*. In other word:

$$\Theta = n(n-3) \qquad ....(17)$$

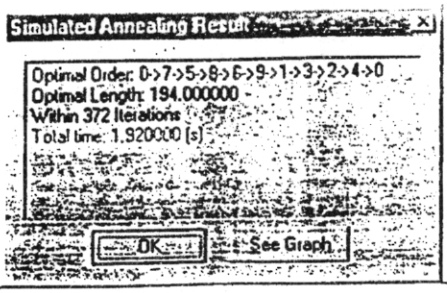
Figure 3 shows results of a 400-city-example (distances are set randomly).

To compare with an exhaustive search, figure 4 gives results of an instance with ten cities. The path is not necessarily the same but the optimal length is.



Geometric Cooling with:
α=0.98, χ_0=0.95, ε_s=10^{-6}.
n=400.

Figure 3 TSP Using SA Results



Paths

Order: 0->1->2->3->4->5->6->7->8->9->0 (Length= 581.000000)
Order: 0->1->2->3->4->5->6->7->9->8->0 (Length= 648.000000)
Order: 0->1->2->3->4->5->6->8->7->9->0 (Length= 490.000000)
...

Final result

Optimal order: 0->4->2->3->1->9->6->8->5->7->0
Optimal path length: 194.000000
Optimal path found in: 173.510000 (s)

Figure 4 TSP Using SA and Exhaustive Search

Effect of the distance parameter's value on the cooling can be seen in figure 5 (n=50.) Small δ-values lead to small decrements, but the inverse leads to a metastable state. In figure 6 effect of the geometric coefficient is shown, where a metastable state is obtained for lower values, and the initial acceptance ratio in figure 7.

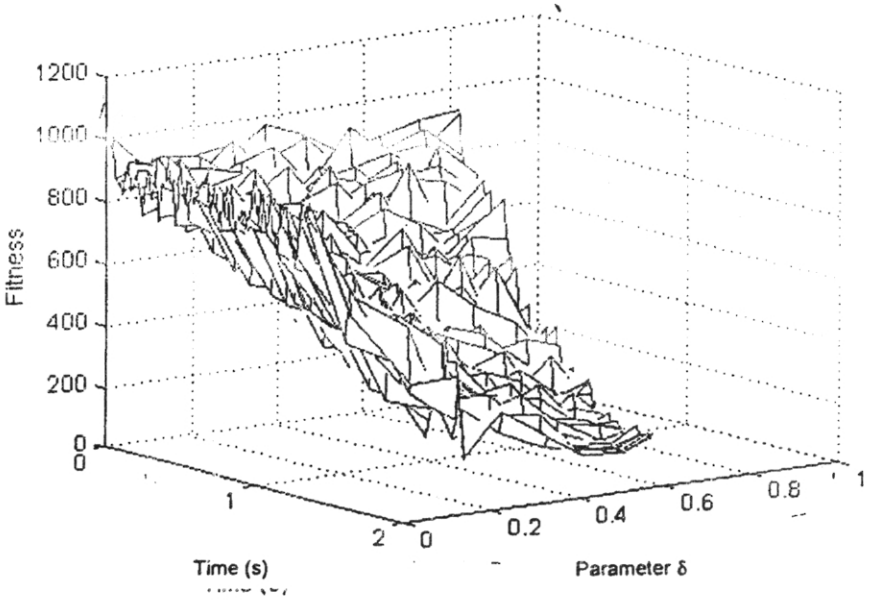


Figure 5 Effect of distance parameter

Different cooling schedules do not have the same rate of convergence. It can be seen in figure 8 that exponential, fast, and geometric cooling are faster than distance and logarithmic cooling; but this is not always useful because it may lead to a meta-stable state.

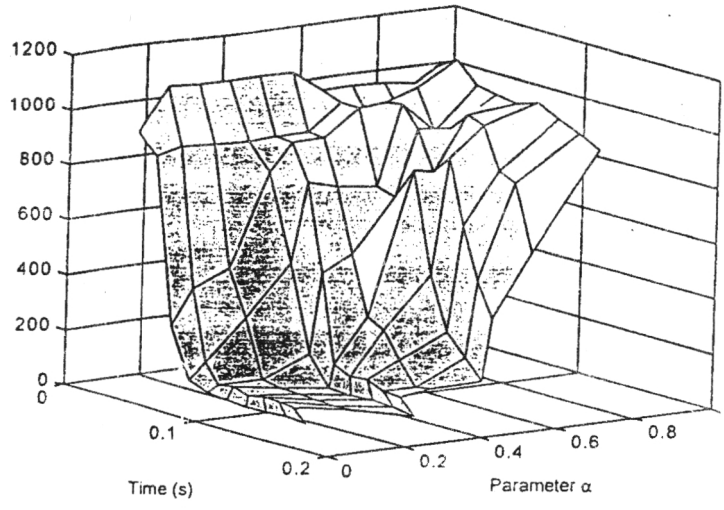


Figure 6 Effect of the geometric parameter

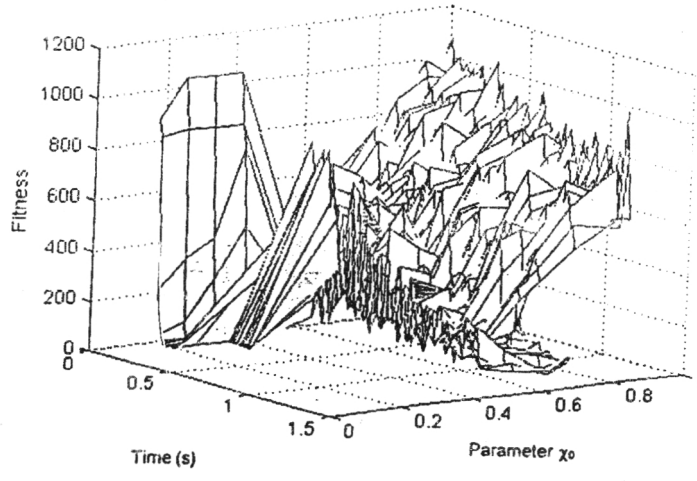


Figure 7 Effect of the initial acceptance ratio

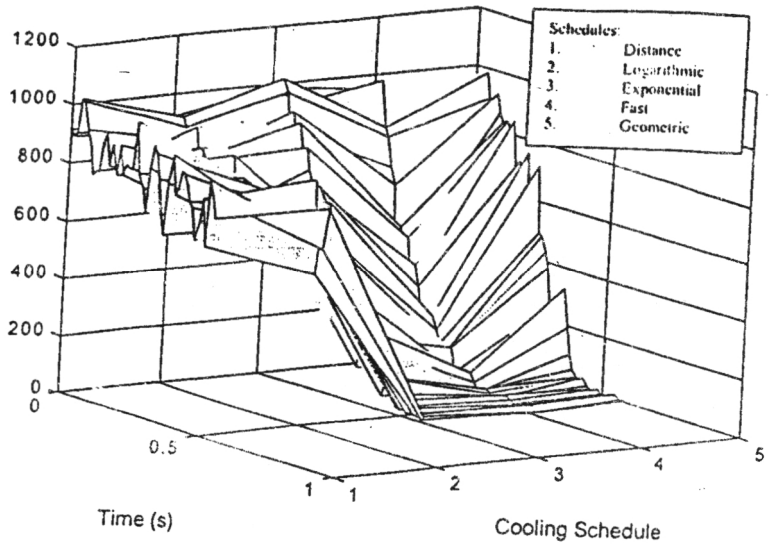


Figure 8 Effect of different schedules

2. ***Real Function Optimization***

Another Np-complete problem is to find the optimal value of a real function in a specific domain. A function that has lots of local minima is to optimize so that efficiency of SA can be noted; it can, really, escape from local minima. The function is given in the equation 17. The graph of this function is shown in figure 9. The optimum value is sought for in the real interval [7,13].

$$F(x)=(x/2-5)^2+\sin(5x+60)+2 \qquad (17)$$

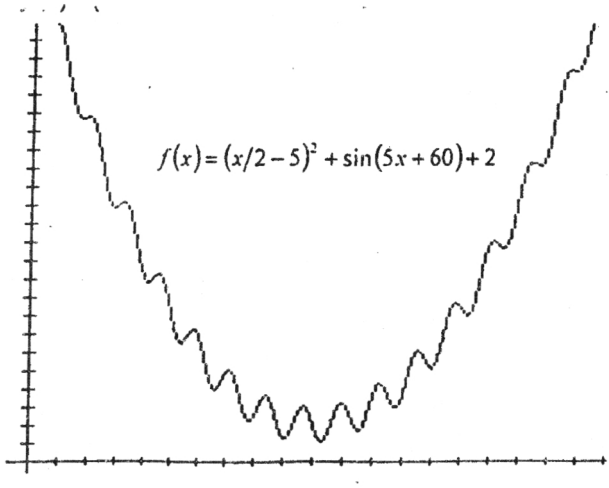


Figure 9. Function to be optimized

2.1 Problem Complexity

Assume that a real number is coded on n digits. Each digit takes one of ten values (0..9). Then the optimal value is chosen among 10^n number. By consequence, the time complexity of an exhaustive algorithm is $0(10^n)$.

2.2 Function Optimization Using SA

a. Solution Representation

An obvious representation is to consider a real as a string of digits issued from the decimal representation of that real. In this example, n the number of digits is taken equal to 7; two of

which are before the decimal point and five precision digits.

b. **The Cost Function**

The cost of a solution x is, simply, the value of the function for this solution. In order word:

Cost $(x)=f(x)$ (17)

In this implementation, x is taken among the interval [7,13].

c. **Generation of New Solutions**

A new solution is obtained by making small random change in the given solution. Changes in the decimal part are preferred. This process consists of randomly adding ± 1 to a randomly selected position in the precision part.

$$\begin{cases} x_k = 12.65356 \\ Mask = -0.001 \end{cases} \Rightarrow x_{k-1} = 12.65256$$

Figure10 Making small random changes to generate new solution

d. **Cooling Schedule**

The same as the previous example, 16 was used rather then equation 13. The number of transitions equation evoked at each value of the control parameter is $\Theta = 2*5 = 10$. Five precision digits and two possibilities to adjust a digit: add or subtract.

The effect of the distance parameter can be seen in figure 12.

IV CONCLUSION

In this paper, the SA algorithm was presented and its principles were discussed. However, this algorithm presents a very interesting feature: it can escape from minima, while it still exhibits the favorable features of local search algorithms: simplicity and general applicability. In addition, it does not mind the degree of non-linearity, discontinuity, or stochasticity of the cost function. Furthermore, speed and convergence of the algorithm are guaranteed by well choosing its parameters, but in general it is statistically guaranteed to find an optimal solution; i.e. the algorithm converges .

Although SA is an approximation algorithm, a very promising method, which can be used as well with other approximation algorithms, is to consider its result as an initial solution for an optimization algorithm.

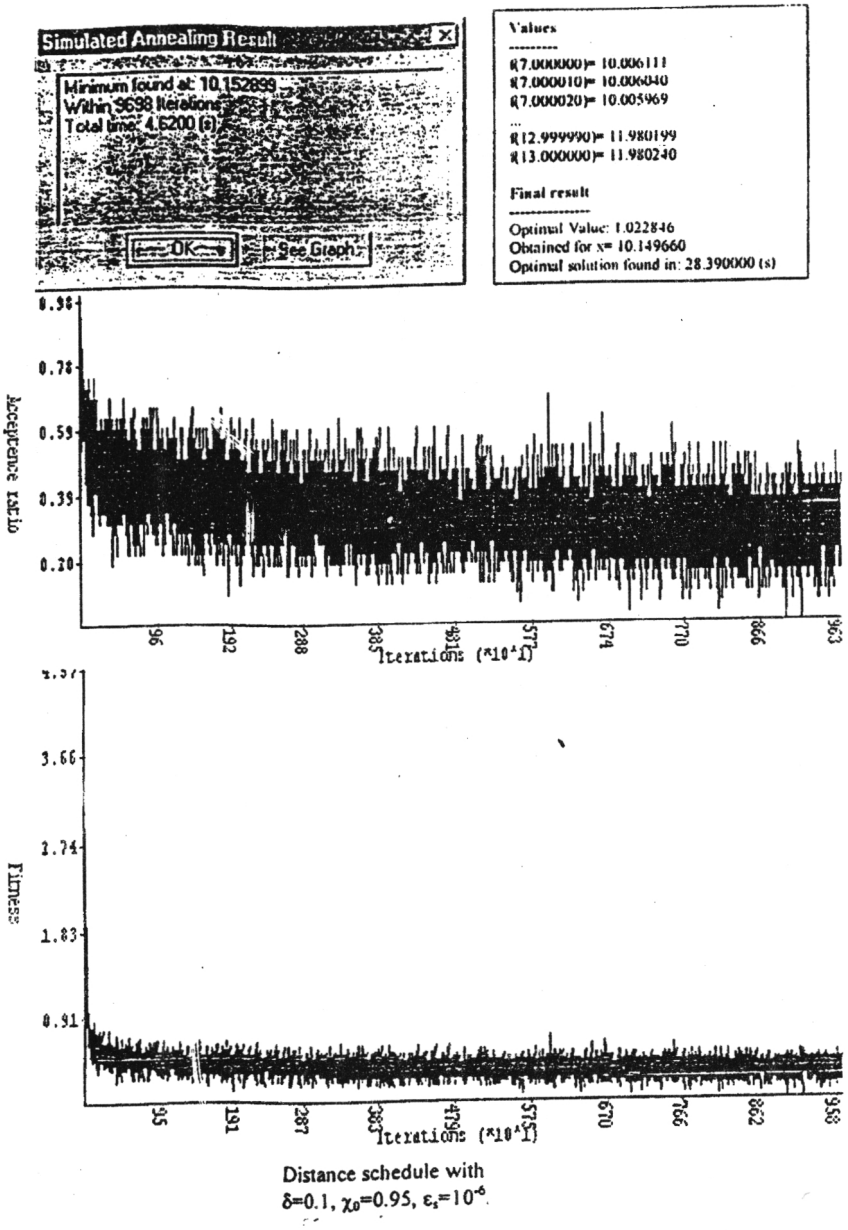


Figure 11 Function optimization using SA

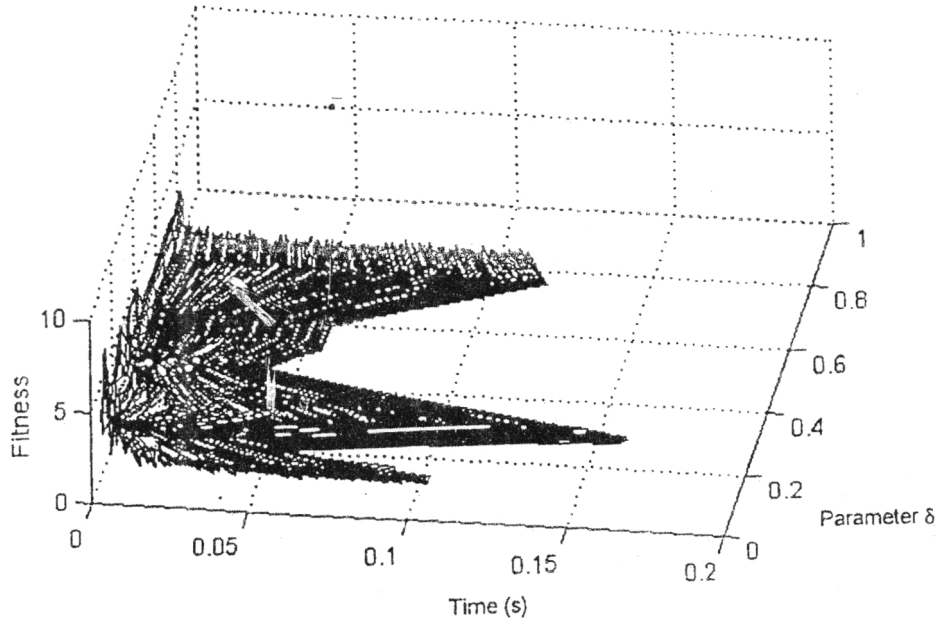


Figure 12 Effect of the distance parameter

REFERENCES

1. S.A. COOK, " The complexity of theorem procedures, " Proc. 3rd ACM
   Symposium of the theory of computing, pp.151-158, 1971.
2. S.A. Cook, "An overview of the computational complexity," *communications of the ACM 26, pp. 400-408,1972.*
3. E. A arts, J. Korst, *Simulated annealing and Boltzman machines,* John Wiley & sons, 1989.
4. S. Kirkpatrick, Jr. Gelatt, M. P. Vecchi, " Optimization by Simulated annealing, " *Science* 220, pp. 671-680, 1983.
5. "Le recuit simule, " chapitre 9. Web site.
   http://www.lps.ens.fr/-weisbuch/simulan.html
6. N. Metropolis, A. Rosenbleth, M. Rosenbleth, A. Teller, E. Teller, " Equation of state calculations by fast computing machines, " *Journal of chemical physics 21,*Springer Verlag , Berlin, pp.1087-1092, 1953.
7. M. Toda, R. Kubo, N. Saito, *Statistical,* Springer Verlag, Berlin, 1983.
8. F. Sultani, "Inversion de la Transformation de Poisson par Recuit simule," Rapport se DEA, Universite de Nice.
9. G. Robinson, " Simulated annealing,"
   http://www.npac.syr.edu/copywrite/pew/node252html
   1995
10. E. Sundermann, I. Lemahieu, P. Desmedt, " PET Image Reconstruction Using Simulated Annealing, " Proceedings of the ProRISC, Workshop on Circuits, Systems and Signal Processing. Papendal, Arnhem (NL), pp. 239-245,1994.

11.I.. Ingber, " Adaptive simulated annealing (ASA): lessons learned," *Control and Cybernetics, special issue on* " Simulated Annealing Applied to Combinatorial Optimization," 1996. .

12.L. Ingber, "Simulated annealing: Practice versus theory", *Mathematical computer modeling* 18(11),pp.29-57,1993.

13.G. F. Luger, W. A. Stubblefield. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* Addition-Wesley, 1998.

14.B. W. Wah, C. V. Ramamoorthy. " Theory of Algorithms and computation complexity with applications to software design," *in Handbook of Software Engineering* .CBS publishers and distributors, India, 1986.

15.O. Martin, S. W. Otto, "Large-Step Markov Chains for the Traveling Salesman Problem," *ComplexSysyems,*5 (3), pp.299-328, 1991

16.R. Bellman, "Dynamic programming treatment of the traveling salesman problem," *Journal ACM,* 9, pp. 61-63,1962.