SIMULATED ANNEALING: FROM STATISTICAL THERMODYNAMICS TO COMBINATORY PROBLEMS SOLVING

D. Thiel
ENITIAA, Nantes, France

Keywords: Combinatory Problem, Optimizing Problem, Global Search Method, Statistical Physics, Thermodynamics, Heuristics, Simulated Annealing

Contents

Summary

Faced to the complexity of the combinatory problem solving, different heuristic approaches which are based on physics and biology concepts, have been developed in the end of the last century. More particularly, the NP-complete problems and the archetypes of computationally intractable problems which are the *Satisfiability Problem* (SAT), the *Travelling Salesman Problem* (TSP) or the *Quadratic Assignment Problem* (QAP), need a calculation time which exponentially increases according to the problem size.

This chapter will particularly present the principle of one of these heuristic tools, the *Simulated Annealing* (SA) algorithm which has been particularly successful in solving various combinatorial decision and optimization problems.

Firstly, the thermal annealing process will be explained in the objective to show the analogies between this sequential thermodynamic transformation of the crystalline system states and the SA algorithm. Then, the order and disorder notions at the molecular level and the dynamics of the state changes and the systems stability, will be defined. These aspects correspond to the local optimality in the combinatory problems which is similar to attraction basins in thermodynamic systems. Contrarily to the

analytical methods which stop their research if a local optimum is attempt, the SA algorithm is able to go through metastable states and to jump towards other attraction basins. This discrete state change is activated by a random law which corresponds to the *Boltzmann* thermodynamic equation. From this introduction based on statistical physics and thermodynamics, the detailed principle and the conditions of convergence of the SA algorithm, will be described. Finally, some applications of SA will demonstrate the performance of this tool.

1. Complexities of Problems and Algorithms

Some combinatory problems can not be solved optimally (despite centuries of work of them) because the computer time to find the best solutions, increases exponentially with the size of these problems. The archetype of computationally intractable problems is the *Satisfiability Problem* or SAT. It is a decision problem which has a yes/no answer. For example, in a restaurant, a man gives his meal preferences in the following terms: he likes salads or meat. But he also likes salads or dessert, and finally, he hesitates between a meal without meat or a meal without desserts. The sole solution of this problem is that this man will only choose salads for his meal. If the propositions are salads or meat, salads or a meal without meat, a meal without salads or with meat, it is not possible to find a satisfying assignment.

To identify the complexity of combinatory problems, it is necessary to verify if the time to solve their, is a polynomial time or not (a polynomial P is an arithmetic expression composed by summing multiples of powers of some variables, the highest power qualifies the degree of the polynomial). Presently, there is no algorithm known that is guaranteed to solve any problem like SAT in a time polynomial in the number of variables. The remarkable discovery of Cook in the early 70s, was that many complex problems such as scheduling, can be polynomially reduced into a SAT problem. For the scheduling problems for instance, in this approach, the task of finding a plan in a given domain, is converted to the task of finding a model for a certain satisfiability problem. In practice, the SAT problem is fundamental in solving many application problems and the methods to solve this type of problems, play a central role in the development of efficient computing systems. There also has been a strong connection between the theory, the algorithms, and the applications of the SAT problem.

On these decision problems can be added other types of combinatory problems like search problems. For example, to solve a scheduling problem, it is not sufficient to prove the existence of a solution, it also needs to construct them. An optimization problem can be solved like a search problem by associating to a solution a value which consists in minimizing or maximizing an objective function. A typical and well-known example of archetypal problem, is the *Travelling Salesman Problem* (TSP) where a salesman must visit *n* cities in the shortest possible time. Mathematically, the TSP consists in finding the shortest Hamiltonian cycle in a graph with *n* vertices and with edge weights between the vertices, equal to the distances between the cities (see in Figure 1 three different elementary circuits C_1, C_2, C_3 among the great number of circuits possibilities. For example, if the number of cities is equal to ten, the number of cycles is equal to $(1\times2\times3\times4\times \ldots \times9\times10) = 3{,}628{,}800$ possibilities. This problem has a variety of solutions of varying complexity and efficiency. The running time to find the

best solution is exponential to the number of cities *n*.

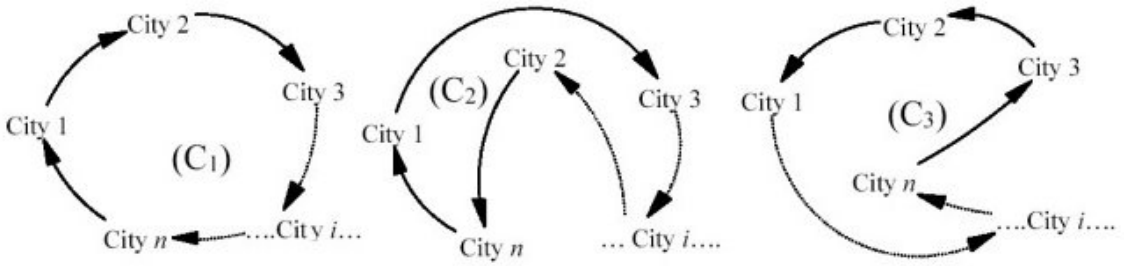

Figure 1. Some elementary circuits for a Traveling Salesman Problem.

To perceive the limits of this type of problems, it is necessary to evaluate the number of operations of the algorithms which can solve their, according to the data size. This corresponds to classify the problem according to the necessary time to solve it in polynomial time. Of course, if it is possible to demonstrate that the problem is polynomial, then it will be solved "classically" and it will not be useful to use heuristic techniques (in Greek, *heuristikein* means searching). By a more trivial approach, these problems have to be firstly presented in two classes of complexity, the P problems and the E problems. The P problems are "good problems" in the sense that the calculation of their solutions, is feasible in a reasonable time (i.e. polynomial problems). The E problems are exponential problems (their complexity is like *k* power *n*, where *k* is a constant and *n* the data size). But a lot of problems are excluded of this previous classification, there are NP-hard problems (the more "hardest" of this type of problems is called a NP-complete problem). By an intuitive point of view, NP-complete problems can be interpreted as a search of solutions in a tree. This tree contains all the possible solutions and each branch represents one possible solution. The height of such a tree, is polynomial but the number of branches is exponential, each node corresponds to a choice of a value of a variable. According to the three TSP circuits showed in the previous Figure 1, it is easy to recognize in the tree presented in the Figure 2, the beginning of the circuits C_1, C_2 and C_3.

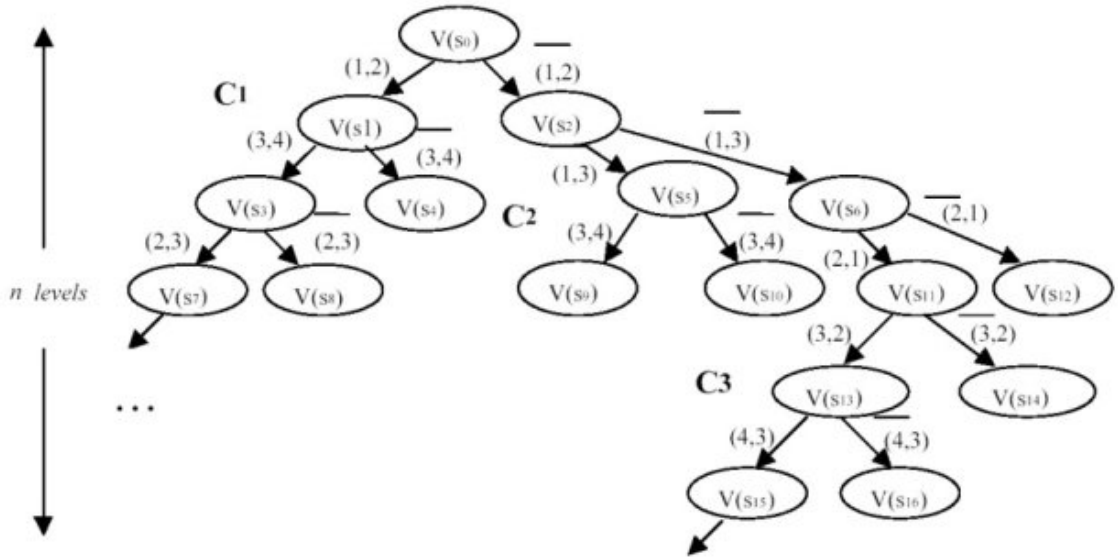

Figure 2. An example of a TSP solution tree.

The bottom of this tree corresponds to the initial solution s_0 and its "economic" value $v(s_0)$. The tree starts with this solution and then, in this example, has to choose to go through the arrow (1,2) or not. If the arrow (1,2) is taken, the next solution s_1 corresponding on a the beginning of the path (1,2), gives the "economic" value $v(s_1)$. New values are calculated and on each vertex *i*, the algorithm chooses the best value $v(s_i)$. The space of the possible solutions (s_0, s_1, s_2,…) with their respective values ($v(s_0)$, $v(s_1)$, $v(s_2)$,…) exponentially increases according to the depth of the tree. For example, the sequences of the arrows of the circuit C_1 is the following: (1,2) means from city 1 to city 2, and after that (3,4); (2,3) etc. A branch noted $\overline{i, j}$ blocks the way to the arrow (*i*, *j*). If the tree is binary and the depth is equal to *n*, then the number of branches is equal to 2^n. The only way to obtain an acceptable solution is to go all over the tree until finding a correct solution. In the less good case, this procedure can need to go all over the tree by testing all the different branches, the problem is therefore exponential.

The complexity of these different algorithms has to be evaluated according to the computer type and the chosen computer language. To do that, the Turing machine is used as a mathematical model which is universally known as representing the computer behavior. The principle of this machine is quite simple. It is composed by three parts: an infinitely long "tape" with symbols on which the instructions and the data are stored, a read/write head which reads or modifies the data of the tape and can be moved along that, and a finite *Control Unit* that defines how to operate the head and the tape. The tape is composed by integer numbers included into – infinite and + infinite, or by blanks (if no data). At each step, the machine reads the number at the current position on the tape. For each combination of current state and number read, a program specifies the new state and either a symbol to write to the tape or a direction to move the pointer (left or right) or to halt. If the machine starts at time $t = 0$, the first state s_0 corresponds to the position of the read/write head. If the word *w* is represented by *n* binary digits (w_1, w_2, w_3,…, w_n) for example the word (0, 1, 1, …,0), the head at time $t = 0$ is pointed on 0 (see Figure 3). The tested program P (cf. the algorithm included in the *Control Unit*) corresponds to a transition function which defines the rules depending on each state and on the value of the digit faced to the position of the head. For example, on state $s_{0,}$ if the value of the digit is equal to 0, then go one place to the left and remain in the same state $s_{0.}$ If it is equal to 1, then go one place to the right and move to the next state $s_{1.}$ If it is equal to *b* (blank), then stop the program (cf. the algorithm) and this is the final state s_f. These different rules are summarized in the Table 1.

	Digit Value = 0	**Digit Value = 1**	**Blank**
State s_0	go to s_0, move left	go to s_1, move right	go to s_2, move right
State s_1	go to s_1, move left	go to s_2, move right	go to s_2, move left
State s_2	go to s_f, STOP	go to s_f, STOP	go to s_2, move left

Table 1. An example of a transition function.

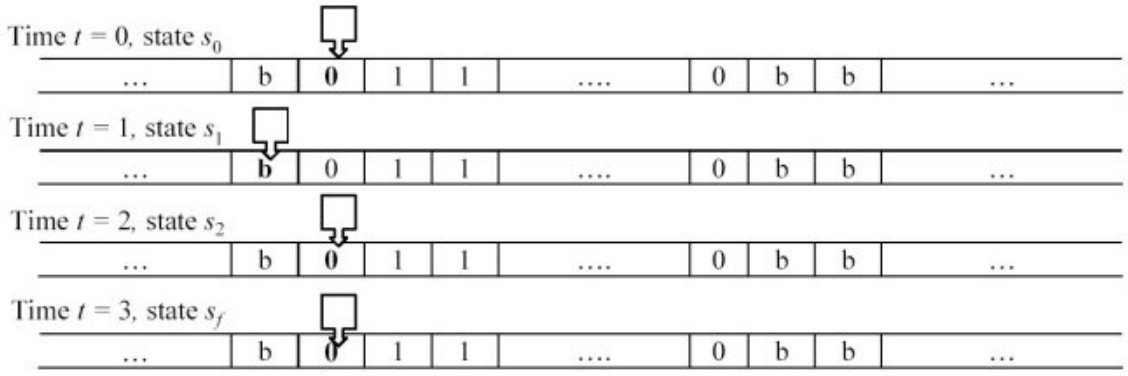

Figure 3. The Turing Machine.

If the Turing machine attempts the final state s_f in a finite time from an initial state s_0, it is possible to define the duration of the algorithm. This time is based on the number of necessary iterations the algorithm needs to move from s_0 to s_f (if the Turing machine does not attempt the final state s_f, the duration is infinite). In the previous example, the program P needs a time corresponding to three iterations to converge towards the final desired state (see Figure 3). Anything that can be solved by a Turing machine program, can be programmed in one of the thousand different models of computation. This machine is therefore used for computability theory proofs (Siegelmann reported in *Science* of 28 April 1995 that she has found a mathematically rigorous class of machines, based on ideas from chaos theory and neural networks, that are more powerful than Turing Machines).

This « measure » of complexity can imply that a given problem is presently of the complexity of the less complex known algorithm which is able to solve it. But if the less complex algorithm needs more calculation time that an usually ones, this last conclusion is to be moderated. To solve NP-complete problems, a non-deterministic algorithm can be compared with an algorithm realized by a non-deterministic Turing Machine. The difference is, when a deterministic algorithm produces only one calculation, a non-deterministic algorithm produces a set of calculations by introducing the choice notion (a set of solutions is accessible at each state of the algorithm).

The next section presents the principle of *Global Search Methods* for solving NP-complete optimization problems.

2. Introduction to Global Search Methods

In combinatory optimization problems solving, some iterative methods find solutions which are « sub-optimal » because there does not escape from certain optimums called local optimums and can not reach states where further improvements can be found. The Figure 4 gives an example of a three-dimensional space where the algorithm searches the lowest point of the « landscape » f(x, y) which is generated by the solutions s_i defined according to different possible values of x and y. For example, the function f(x,y) could represent the evolution of a cost function according to two variables x and y,

s_i are possible couples of solutions (x_i,y_i) which implies a corresponding cost $f(x_i,y_i)$. As it is possible to see in this figure, the algorithm stops its trajectory (s_0, s_1, s_2,…,s_p) in the lowest point s_p of one "valley" which is not the more depth of the global landscape. Therefore, the algorithm does not find the minimum value s_{min} of the objective function $f(x, y)$ of the optimization problem and is not able to continue its trajectory to the point s_{p+1}.

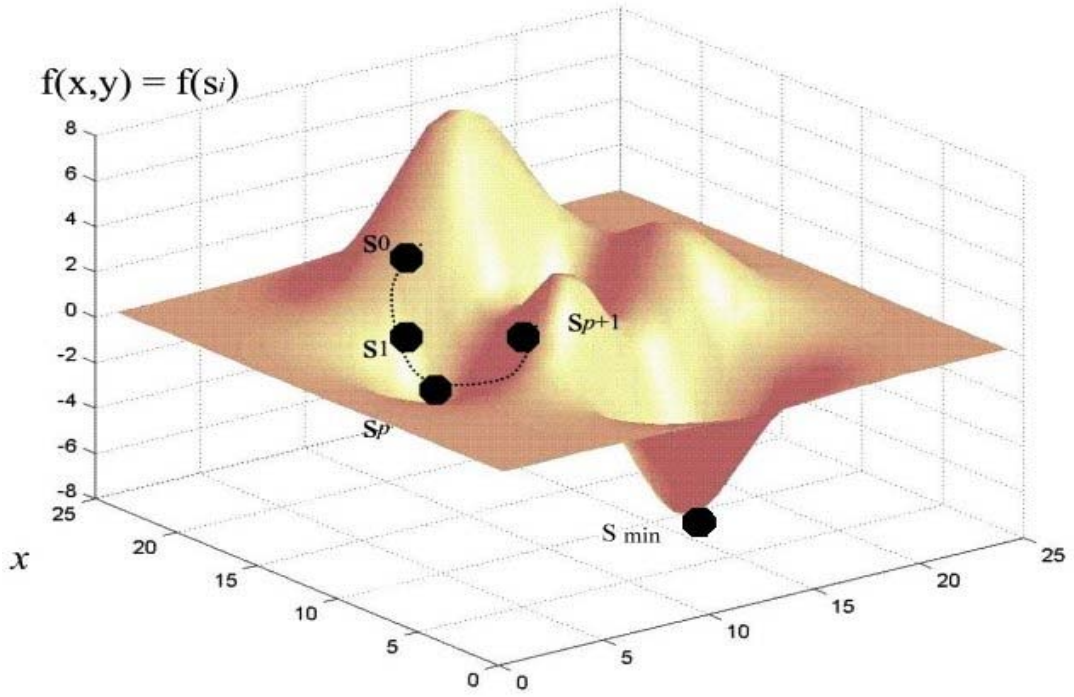

Figure 4. Global Search Methods (the algorithm stops its trajectory(S_0,S_1,S_2,……,S_p)in the lowest point S_p of one "Valley" which is not the more depth of the global landscape.

Figure 4 gives an example of a three-dimensional space where the algorithm searches the lowest point of the « landscape » $f(x, y)$.

Now, this procedure will be in detail, developed. An initial point is given (i.e. an initial solution s_0), for example in the TSP, an « usual » round or circuit which was empirically of randomly generated. This initial solution corresponds to a succession of arrows which are evaluated by an initial cost defined according to the objective function *f* (for example, the cost can be based on the distance or the time of the travel and is equal to the valuations sum of the arrows of the circuit). For the solution s_0, the cost is defined by $f(s_0)$. To choose a new circuit s_1 which is more « economic », such as $f(s_1)$ is lower than $f(s_0)$, an heuristic should be applied like for instance, switching two cities of the circuit (instead of the path …→ city i → city j → city k →…, do …→ city i → city k → city j →…). If the solution s_1 is more interesting as s_0, s_1 will be accepted and s_0 on while the new solutions are such as $f(s_i)$ is lower than $f(s_{i-1})$, for i greater than zero. The problem of this type of algorithm is, if after (p+1) iterations with p greater than zero, $f(s_{p+1})$ gets greater than $f(s_p)$, then the algorithm will stop in a local minimum. In the Figure 4, some solutions s_j (with s_j included in the interval [s_0, s_p]) give better results

$f(s_j)$ as $f(s_p)$ which corresponds to the best solutions s_p where the algorithm has previously stopped. But this type of algorithm will not be able to select their. Only a modification of the initial conditions (i.e. of the initial circuit) is possible to give new chances for the algorithm to scrutinize other "valleys". An problem is that the *f* function « profile » is *a priori* unknown, except in enumerating all the solutions but it is quite always impossible because of the extremely long computer time. For these different reasons, global optimization algorithms are used. Their objectives are to find a solution in the solution set for which the objective function obtains its smallest value, the global minimum. So, these algorithms avoid to remain only in one « valley » (or attraction basin which corresponds to a metastable state, in physical sense) and can find a final state which are more stable. Nevertheless, when the number of variables gets larger, the global optimization algorithms are to be more efficient. *Simulated Annealing* can solve global optimization problems with several hundred variables. However, this is a small number of variables when one considers that, in monocriteria operational research, integer programming can tackle problems with thousands of variables, and linear programming is able to solve problems with millions of variables (but these last algorithms can not solved NP-complete problems).

To summarize, global optimization is the task of finding the absolutely best set of admissible conditions to achieve a given objective, formulated in mathematical terms. Now, before developing the SA principles, it is useful to explain the scientific foundations of this global search algorithm by referring to statistical physics and thermodynamics.

-
-
-

Bibliography

Aarts E., Lenstra J.K. (eds.) (1997). *Local Search in Combinatorial Optimization*. Wiley, Chichester. [This book introduces classical and recent developments and positions local search within combinatorial optimization and basic exchange algorithms].

Kirkpatrick, S., Gelatt C.D., Vecchi M.D. (1983). Optimization by Simulated Annealing. *Science*, **220** (4598), 671-680. [After reviewing the central constructs in combinatory optimization and in statistical mechanics, this article presents different applications to partitioning, component placement, and wiring of electronic systems. To test the power of simulated annealing, the TSP is also tested with as several thousand cities].

Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H., Teller E., (1953). Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, **21** (6), 1087-1092. [The authors develop the SA algorithm which is a generalization of a *Monte Carlo* method for examining the equations of state and frozen states of *n*-body systems].

Garey M. R., Johnson D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman and Company. [The notion of approximation algorithms are presented

according to the theory of NP-completeness as a way of coping with the difficulty of solving NP- hard combinatorial optimization problems].

Aho, A.V., Hopcroft J.E., Ullman J.D. (1974). *The design and analysis if computer algorithms*, Reading, MA, Addison-Wesley. [This book tackles about analysis of algorithms, asymptotic notation, algorithmic techniques, data structures, sorting and order statistics, graph algorithms: and NP-completeness].

Prigogine I. Stengers I. (1984). *Order out of chaos: man's new dialogue with nature*, New York, Bantam. [The authors present a wide ranging and well documented discourse on the gradual emergence of philosophical and scientific thought in regard to conceptions of order and chaos. They chose to attempt investigation of a third and largely ignored class of systems, those which were far from equilibrium].

Geman, S., Geman D. (1984). Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**(6), 721-741. [On the problem of temperature decreasing law in SA, the authors proved that a logarithmic decrease of temperature with the proportionality constant roughly of the order of the system size, guarantees convergence to the optimal state for a wide class of combinatorial optimization problems].

Nulton J., Salamon P. (1988). Statistical mechanics of combinatorial optimization. *Physical Review A*, 37:1351-1356. [A theoretical criterion is offered for the design of a temperature schedule for SA. It is based on a measure of distance in probability space].

R. Heckman R., Lengauer Th. (1995). A simulated Annealing Approach to the Nesting Problem in the Textile Manufacturing Industry. In R. E. Burkard, P. L. Hammer, T. Ibakari and M. Queyranne, editors, *Annals of Operations Research*, Mathematics of industrial systems I, Vol. 57, 103-133. [The nesting problem in the textile industry is the problem of placing a set of irregularly shaped pieces (called stencils) on a rectangular surface, such that no stencils overlap and that the trim loss produced when cutting out the stencils is minimized. In this paper, an algorithmic approach using SA is presented covering a wide variety of constraints which may occur in the industrial manufacturing process].

Randelman R. E, Grest G.S. (1986). N-City Traveling Salesman Problem - Optimization by Simulated Annealing. *Journal of Statistic Physics*, **45**, 885-890. [In SA implementing, the authors propose the discrete function for the temperature decreasing function which is independent to the value of the previous state and only depends on the value of the gradient of temperature. This function can be used when the iterations number which is necessary to reach the equilibrium state, can be evaluated with a satisfactory precision].

Wilson R. A., Keil F. (eds.) (1999). *The MIT Encyclopedia of the Cognitive Sciences* (MITECS). MIT Press. [This Encyclopedia represents the methodological and theoretical diversity of the field of cognitive science and some chapters are devoted to Computational Intelligence].

Biographical Sketch

Daniel Thiel received a M.Sc. in Physics and an Engineer Graduate Level from the Ecole Centrale of Lyon. He also received a Ph.D. in Automatics, Computer Engineering and Signal Treatment from the University of Aix-Marseille III and presented his full professor dissertation in Management Science at the University of Nantes. Since 1995 he has been on the graduate engineering school ENITIAA at Nantes where he is now a Professor of decision-making sciences and methods. His is also researcher at the University of Nantes. Previously he had taught at Tours Graduate School of Business as well as he served on various boards for several companies during ten years. From 1989-1995, Pr. Thiel headed the Tours Graduate School of Business Research Center and from 1998, he is Scientific Director of ENITIAA. He has published in several academic Journals including International Journal of Production Research, Food Quality and Preference, Simulation Practice and Theory. In addition, he is a regular reviewer of the Journal of Operations Management. Pr. Thiel research interests include Logistics, Production Management, Psychosociology by using Operational Research tools and Simulation tools like Automata Networks and Continuous Simulation. He has published over hundred papers and communications in International Conferences in these areas. He is author of two books and editor of one book dealing with System Dynamics, Complexity and Chaos. He is a member of the editorial board of the European Journal of Economics and Social Sciences and a French Journal of Information Systems and reviewer of other International Journals.