Chapter 2

A Simulated Annealing Algorithm for the Satisfiability Problem Using Dynamic Markov Chains with Linear Regression Equilibrium

Felix Martinez-Rios and Juan Frausto-Solis

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46175

1. Introduction

Since the appearance of Simulated Annealing algorithm it has shown to be an efficient method to solve combinatorial optimization problems such as Boolean Satisfiability problem. New algorithms based on two cycles: one external for temperatures and other internal, named Metropolis, have emerged. These algorithms usually use the same Markov chain length in the Metropolis cycle for each temperature. In this paper we propose a method based on linear regression to find the Metropolis equilibrium. Experimentation shows that the proposed method is more efficient than the classical one, since it obtains the same quality of the final solution with less processing time.

Today we have a considerable interest for developing new and efficient algorithms to solve hard problems, mainly those considered in the complexity theory (NP-complete or NP-hard) [8]. The Simulated Annealing algorithm proposed by Kirkpatrick et al. [18] and Cerny [5, 6] is an extension of the Metropolis algorithm [23] used for the simulation of the physical annealing process and is specially applied to solve NP-hard problems where it is very difficult to find the optimal solution or even near-to-optimum solutions.

Efficiency and efficacy are given to Simulated Annealing algorithm by the cooling scheme which consists of initial (c_i) and final (c_f) temperatures, the cooling function $(f(c_k))$ and the length of the Markov chain (L_k) established by the Metropolis algorithm. For each value of the control parameter (c_k) (temperature), Simulated Annealing algorithm accomplishes a certain number of Metropolis decisions. In this regard, in order to get a better performance of the Simulated Annealing algorithm a relation between the temperature and Metropolis cycles may be enacted [13].

The Simulated Annealing algorithm can get optimal solutions in an efficient way only if its cooling scheme parameters are correctly tuned. Due this, experimental and analytical parameters tuning strategies are currently being studied; one of them known as ANDYMARK [13] is an analytical method that has been shown to be more efficient. The objective of these methods is to find better ways to reduce the required computational resources and to increment the quality of the final solution. This is executed applying different accelerating techniques such as: variations of the cooling scheme [3, 27], variations of the neighborhood scheme [26] and with parallelization techniques [12, 26].

In this chapter an analytic adaptive method to establish the length of each Markov chain in a dynamic way for Simulated Annealing algorithm is presented; the method determines the equilibrium in the Metropolis cycle using Linear Regression Method (LRM). LRM is applied to solve the satisfiability problems instances and is compared versus a classical ANDYMARK tune method.

2. Background

In complexity theory, the satisfiability problem is a decision problem. The question is: given the expression, is there some assignment of TRUE and FALSE values to the variables that will make the entire expression true? A formula of propositional logic is said to be satisfiable if logical values can be assigned to its variables in a way that makes the formula true.

The propositional satisfiability problem, which decides whether a given propositional formula is satisfiable, is of critical importance in various areas of computer science, including theoretical computer science, algorithmics, artificial intelligence, hardware design, electronic design automation, and verification. The satisfiability problem was the first problem refered to be as NP complete [7] and is fundamental to the analysis of the computational complexity of many problems [28].

2.1. Boolean satisfiability problem (SAT)

An instance of SAT is a boolean formula which consists on the next components:

- A set S of n variables $x_1, x_2, x_3, ..., x_n$.
- A set L of literals; a literal l_i, is a variable x_i or its negation $\widetilde{x_i}$.
- A set of m clauses: $C_1, C_2, C_3, ..., C_m$ where each clause consists of literals l_i linked by the logical connective OR ($\vee$).

This is:

$$\Phi = C_1 \wedge C_2 \wedge C_3 \wedge ... \wedge C_m \tag{1}$$

where Φ, in Equation 1, is the SAT instance. Then we can enunciate the SAT problem as follows:

Definition 1. *Given a finite set* $\{C_1, C_2, C_3, \ldots, C_m\}$ *of clauses, determine whether there is an assignment of truth-values to the literals appearing in the clauses which makes all the clauses true.*

NP-completeness in SAT problem, only refers to the run-time of the worst case instances. Many of the instances that occur in practical applications can be solved much faster, for example, SAT is easier if the formulas are restricted to those in disjunctive normal form, that is, they are disjunction (OR) of terms, where each term is a conjunction (AND) of literals. Such a formula is indeed satisfiable if and only if at least one of its terms is satisfiable, and a term is satisfiable if and only if it does not contain both x and $\widetilde{x}$ for some variable x, this can be checked in polynomial time.

SAT is also easier if the number of literals in a clause is limited to 2, in which case the problem is called $2 - SAT$, this problem can also be solved in polynomial time [2, 10]. One of the most important restrictions of SAT is HORN-SAT where the formula is a conjunction of Horn clauses (a Horn clause is a clause with at most one positive literal). This problem is solved by the polynomial-time Horn-satisfiability algorithm [9].

The 3-satisfiability (3-SAT) is a special case of k-satisfiability (k-SAT), when each clause contains exactly $k = 3$ literals. 3-SAT is NP-complete and it is used as a starting point for proving that other problems are also NP-hard [31]. This is done by polynomial-time reduction from 3-SAT to the other problem [28].

3. Simulated Annealing algorithm

Simulated Annealing improves this strategy through the introduction of two elements. The first is the Metropolis algorithm [23], in which some states that do not improve energy are accepted when they serve to allow the solver to explore more of the possible space of solutions. Such "bad" states are allowed using the Boltzman criterion: $e^{-\Delta J/T} > rnd(0,1)$, where ΔJ is the change of energy, T is a temperature, and $rnd(0,1)$ is a random number in the interval $[0,1)$. J is called a cost function and corresponds to the free energy in the case of annealing a metal. If T is large, many "bad" states are accepted, and a large part of solution space is accessed.

The second is, again by analogy with annealing of a metal, to lower the temperature. After visiting many states and observing that the cost function declines only slowly, one lowers the temperature, and thus limits the size of allowed "bad" states. After lowering the temperature several times to a low value, one may then "quench" the process by accepting only "good" states in order to find the local minimum of the cost function.

The elements of Simulated Annealing are:

- A finite set S.
- A cost function J defined on S. Let $S^* \subset S$ be the set of global minima of J.
- For each $i \in S$, a set $S(i) \subset S - i$ is called the set of neighbors of i.
- For every i, a collection of positive coefficients $q_{ij}, j \in S(i)$, such that $\sum_{j \in S(i)} q_{ij} = 1$. It is assumed that $j \in S(i)$ if and only if $i \in S(j)$.
- A nonincreasing function $T : N \to (0, \infty)$, called the cooling schedule. Here N is the set of positive integers, and $T(t)$ is called the temperature al time t.
- An initial state $x(0) \in S$.

The Simulated Annealing algorithms consists of a discrete time inhomogeneus Markov chain $x(t)$ [4]. If the current state $x(t)$ is equal to i, chose a neighbor j of i at random; the probability that any particular $j \in S(i)$ is selectec is equal to q_{ij}. Once j is chosen, the next state $x(t+1)$ is determined as follows:

$$\begin{array}{ll} \text{if } J(j) \leq J(i) \text{ then } x(t+1) = j & \\ \text{if } J(j) \leq J(i) \text{ then} & \\ \quad x(t+1) = j \text{ with probability } e^{-(J(j)-J(i))/T(t)} & \\ \text{else} & \\ \quad x(t+1) = i & \end{array} \tag{2}$$

In a formal way:

$$P\left[x(t+1) = j | x(t) = i\right] = \begin{cases} q_{ij} e^{\left[-\frac{1}{T(t)} max\{0, J(j)-J(i)\}\right]} & j \neq i, j \in S(i) \\ 0 & j \neq i, j \notin S(i) \end{cases} \tag{3}$$

In Simulated Annealing algorithm we are considering a homogeneus Markov chain $x_T(t)$ wich temperature $T(t)$ is held at a constant value T. Let us assume that the Markov chain $x_{T(t)}$ is irreducible and aperiodic and that $q_{ij} = x_{ji} \forall i, j$, then $x_{T(t)}$ is a reversible Markov chain, and its invariant probability distribution is given by:

$$\pi_T(i) = \frac{1}{Z_T} e^{\left[-\frac{J(i)}{T}\right]} \tag{4}$$

In Equation 4 Z_T is a normalized constant and is evident that as $T \to 0$ the probability π_T is concentrate on the set S^* of global minima of J, this property remains valid if the condition $q_{ij} = q_{ji}$ is relaxed [11].

In the optimization context we can generate an optimal element with high probability if we produce a random sample according to the distribution π_T, known as the Gibbs distribution. When is generated an element of S accomplished by simulating Markov chain $x_T(t)$ until it reaches equilibrium we have a Metropolis algorithm [23].

The Simulated Annealing algorithm can also be viewed as a local search method occasionally moves to higher values of the cost function J, this moves will help to Simulated Annealing escape from local minima. Proof of convergence of Simulated Annealing algorithm can be revised [4].

3.1. Traditional Simulated Annealing algorithms

Figure 1 shows the classic algorithm simulated annealing. In the algorithm, we can see the cycle of temperatures between steps 2 and 5. Within this temperature cycle, are the steps 3 and 4 which correspond to the Metropolis algorithm.

As described in the simulated annealing algorithm, Metropolis cycle is repeated until thermal equilibrium is reached, now we use the formalism of Markov chains to estimate how many times it is necessary to repeat the cycle metropolis of so that we ensure (with some probability) that all solutions of the search space are explored.

Similarly we can estimate a very good value for the initial and final temperature of the temperature cycle. All these estimates were made prior to running the simulated annealing algorithm, using data information SAT problem is solved.

```
1  Initializing:
   Initial solution $S_i$
   Initial and final temperature: $T_i$ and $T_f$
   $T = T_i$
2  Temperatures cycle:
3  Metropolis cycle:
   Generating $S_j$ from $S_i$
   $dif = J(S_j) - J(S_i)$
   If $dif < 0$ then
          $S_i = S_j$
   else if $e^{\frac{-dif}{T}} >$ rnd(0,1)
          $S_i = S_j$
4  Metropolis condition:
   If thermal equilibrium is reached
          goto 5
   Else
          goto 3
5  Stop criterion:
   If the final temperature $T_f$ is reached
          End
   Else
          Update $T$
          goto 2
```

Figure 1. Simulated Annealing algorithm

It is well known that Simulated Annealing requires a well defined neighborhood structure and other parameters as initial and final temperatures T_i and T_f. In order to determine these paratmeters we follow the next method proposed by [30]. So following the analysis made in [30] we give the basis of this method.

Let $P_A(S_j)$ be the accepting probability of one proposed solution S_j generated from a current solution S_i, and $P_R(S_j)$ the rejecting probability. The probability of rejecting S_j can be established in terms of $P_A(S_j)$ as follows:

$$P_R(S_j) = 1 - P_A(S_j) \tag{5}$$

Accepting or rejecting S_j only depends on the cost deterioration size that this change will produce to the current solution, that means:

$$P_A(S_j) = g[J(S_i) - J(S_j)] = g(\Delta J_{ij}) \tag{6}$$

In Equation 6, $J(S_i)$ and $J(S_j)$ are the cost associated to S_i and S_j respectively, and $g(\Delta J_{ij})$ is the probability to accept the cost difference $\Delta J_{ij} = J(S_i) - J(S_j)$.

The solution selected from S_i may be any solution S_j defined by the next neighborhood scheme:

Definition 2. *Let* $\{\forall S_i \in S, \exists$ *a set* $V_{S_i} \subset S | V_{S_i} = V : S \longrightarrow S\}$ *be the neighborhood of a solution* S_i*, where* V_{S_i} *is the neighborhood set of* S_i*,* $V : S \longrightarrow S$ *is a mapping and* S *is the solution space of the problem being solved.*

It can be seen from the Definition 2 that neighbors of a solution S_i only depends on the neighborhood structure V established for a specific problem. Once V is defined, the maximum and minimum cost deteriorations can be written as:

$$\Delta J_{V_{\max}} = \max[J(S_i) - J(S_j)], \forall S_j \in V_{S_i}, \forall S_i \in S \tag{7}$$

$$\Delta J_{V_{\min}} = \min[J(S_i) - J(S_j)], \forall S_j \in V_{S_i}, \forall S_i \in S \tag{8}$$

where $\Delta J_{V_{\max}}$ and $\Delta J_{V_{\min}}$ are the maximum and minimum cost deteriorations of the objective function through J respectively.

3.2. Markov Chains and Cooling Function

The Simulated Annealing algorithm can be seen like a sequence of homogeneous Markov chains, where each Markov chain is constructed for descending values of the control parameter $T > 0$ [1]. The control parameter is set by a cooling function like:

$$T_{k+1} = f(T_k) \tag{9}$$

and T_k must satisfy the next property:

$$\begin{array}{l} \lim_{k \to \infty} T_k = 0 \\ T_k \geq T_{k+1} \quad \forall k \geq 1 \end{array} \tag{10}$$

At the beginning of the process T_k has a high value and the probability to accept one proposed solution is high. When T_k decreases this probability also decreases and only good solutions are accepted at the end of the process. In this regard every Markov chain makes a stochastic walk in the solution space until the stationary distribution is reached. Then a strong relation between the Markov chain length (L_k) and the cooling speed of Simulated Annealing exists: when $T_k \to \infty, L_k \to 0$ and when $T_k \to 0, L_k \to \infty$.

Because the Markov chains are built through a neighborhood sampling method, the maximum number of different solutions rejected at T_f when the current solution S_i is the optimal one, is the neighborhood size $|V_{S_i}|$. In this regard the maximum Markov chain length is a function of $|V_{S_i}|$. In general L_k can be established as:

$$L_k \leq L_{\max} = g(|V_{S_i}|) \tag{11}$$

In Equation 11, $L_{\max}$ is the Markov chain length when $T_k = T_f$, and $g(|V_{S_i}|)$ is a function that gives the maximum number of samples that must be taken from the neighborhood V_{S_i} in order to evaluate an expected fraction of different solutions at T_f . The value of $L_{\max}$ only depends on the number of elements of V_{S_i} that will be explored at T_f.

Usually a Simulated Annealing algorithm uses a uniform probability distribution function $G(T_k)$given by a random replacement sampling method to explore V_{S_i} at any temperature T_k, where $G(T_k)$ is established as follows:

$$G(T_k) = \begin{cases} \frac{1}{|V_{S_i}|} & \forall S_j \in V_{S_i} \\ 0 & \forall S_j \notin V_{S_i} \end{cases} \tag{12}$$

In this regard, the probability to get the solution S_j in N samples is:

$$P_A(S_j) = 1 - e^{-\frac{N}{|V_{S_i}|}} \tag{13}$$

Notice in Equation 13 that $P_A(S_j)$ may be understood as the expected fraction of different solutions obtained when N samples are taken. From Equation 13, N can be obtained as:

$$N = -\ln(1 - P_A(S_j)) \left|V_{S_i}\right| \tag{14}$$

In Equation 14, we define:

$$= -\ln(1 - P_A(S_j)) = -\ln(P_R(S_j)) \tag{15}$$

You can see that $P_R(S_j) = 1 - P_A(S_j)$, $P_R(S_j)$ is the rejection probability. Constant C establishes the level of exploration to be done In this way different levels of exploration can be applied. For example: if a 99% of the solution space is going to be explored, the rejection probability will be $P_R(S_j) = 0.01$, so, from Equation 15 we obtain $C = 4.60$.

Definition 3. *The exploration set of the search space, Φ_C, is defined as follows:*

- *Given the set of probability of acceptance* $\Phi_{P_A} = \{70, 75, 80, 85, 90, 95, 99, 99.9, 99.99, 99.999, ...\}$
- *Using Equation 15:* $\Phi_C = \{1.20, 1.39, 1.61, 1.90, 2.30, 3.00, 4.61, 6.91, 9.21, 11.51, ...\}$

Then in any Simulated Annealing algorithm the maximum Markov chain length (when $T_k = T_f$) may be set as:

$$L_{max} = N = C|V_{S_i}| \tag{16}$$

Because a high percentage of the solution space should be explored, C varies from $1 \leq C \leq 4.6$ which guarantees a good level of exploration of the neighborhood at T_f.

When the process is at the beginning the temperature T_i is very high. This is because in the Boltzman distribution the acceptance probability is directly related with the cost increment $P_A = e^{-(\Delta J/T_k)}$; where T_k is the temperature parameter, therefore:

$$T_k = -\frac{\Delta J}{\ln(P_A)} \tag{17}$$

At the beginning of the process, P_A is close to one (normally 0.99, [21]) and the temperature is extremely high. Almost any solution is accepted at this temperature; as a consequence the stochastic equilibrium of a Markov cycle is reached with the first guess solution. Similarly, when the process is ending the acceptance probability (tipically 0.01) and the temperature closer to zero but the Metropolis cicle is very long.

For instance SAT values $\Delta J_{V_{\max}}$ and $\Delta J_{V_{\min}}$ in the energy of different states can be estimated at the beginning of the execution on the simulated annealing algorithm. To estimate these values, we can count the maximum number of Clauses containing any of the variables of the problem, the largest number of clauses that can change when we change the value of a variable, is an upper bound to change maximum of Energy and:

$$T_i = -\frac{\Delta J_{V_{\max}}}{\ln(P_A)} = -\frac{\text{max number of clauses}}{\ln(0.99)} \tag{18}$$

Similarly, the minimum of change Energy can be estimated by counting the clauses that are changed when creating a new neighbor and obtain the lowest of these values:

$$T_f = -\frac{\Delta J_{V_{\min}}}{\ln(P_A)} = -\frac{\text{min number of clauses}}{\ln(0.01)} \tag{19}$$

Some criticisms about Simulated Annealing are about the long time of execution of standard Boltzmann-type Simulated Annealing, has many times driven these projects to utilize a temperature schedule too fast to satisfy the sufficiency conditions required to establish a true ergodic search. In this chapter we use a logarithmic an exponential temperature schedule that is consistent with the Boltzmann algorithm follow:

$$T_k = T_0 e^{[(\alpha-1)k]}, 0 < \alpha < 1 \tag{20}$$

From Equation 20 we can obtain:

$$\frac{\Delta T}{\Delta k} = T_k(\alpha - 1), k \gg 1 \tag{21}$$

and

$$\Delta T = T_k(\alpha - 1)\Delta k, k \gg 1 \tag{22}$$

if in the previous expression Δk is equal to 1 then obtain the equation for two successive values of the temperature

$$T_{k+1} = \alpha T_k, 0 < \alpha < 1, k \gg 1 \tag{23}$$

where T_k is the "temperature," k is the "time" index of annealing [16, 17].

3.3. Simulated Annealing algorithm with the Markov chain Lenght dynamically

In [13, 20, 21] authors shown a strong relation between the cooling function and the length of the Markov chain exists. For the Simulated Annealing algorithm, the stationary distribution for each Markov chain is given by the Boltzmann probability distribution, which is a family of curves that vary from a uniform distribution to a pulse function.

At the very beginning of the process (with $T_k = T_i$), Simulated Annealing has a uniform distribution, henceforth any guess would be accepted as a solution. Besides any neighbor of the current solution is also accepted as a new solution. In this way when Simulated Annealing is just at the beginning the Markov chain length is really small, $L_k = Li \approx 1$. When running the temperature cycle of simulated annealing, for values of k greater than 1, the value of T_k is decremented by the cooling function [16], until the final temperature is reached ($T_k = T_f$):

$$T_{k+1} = \alpha T_k \tag{24}$$

In Equation 24 α is normally in the range of $[0.7, 0.99]$[1].

In this regard the length of each Markov chain must be incremented at any temperature cycle in a similar but in inverse way that T_k is decremented. This means that L_k must be incremented until L_{max} is reached at T_f by applying an increment Markov chain factor (β). The cooling function given by Equation 24 is applied many times until the final temperature T_f is reached. Because Metropolis cycle is finished when the stochastic equilibrium is reached, it can be also modeled as a Markov chain as follows:

$$L_{k+1} = \beta L_k \tag{25}$$

In previous Equation 25, L_k represents the length of the current Markov chain at a given temperature, that means the number of iterations of the Metropolis cycle for a T_k temperature. So L_{k+1} represents the length of the next Markov chain. In this Markov Model, β represents an increment of the number of iterations in the next Metropolis cycle.

If the cooling function given by Equation 24 is applied over and over, n times, until $T_k = T_f$, the next geometrical function is easily gotten:

$$T_f = \alpha^n T_i \tag{26}$$

Knowing the initial (T_i) and the final (T_f) temperature and the cooling coefficient (α), the number of times that the Metropolis cycle is executed can be calculated as:

$$n = \frac{\ln T_f - \ln T_i}{\ln \alpha} \tag{27}$$

If we make a similar process for increasing the equation of the Markov chain length, another geometrical function is obtained:

$$L_{\max} = \beta^n L_1 \tag{28}$$

Once n is known by Equation 27, the value of the increment coefficient (β) is calculated as:

$$\beta = e^{\left(\frac{\ln L_{\max} - \ln L_1}{n}\right)} \tag{29}$$

Once L_{max} (calculated form Equation 16), L_1 and β are known, the length of each Markov chain for each temperature cycle can be calculated using Equation 27. In this way L_k is computed dynamically from $L_1 = 1$ for T_i until L_{max} at T_f. First we can obtain T_i from Equation 18 and T_f from Equation 27, with both values and Equation 29 algorithm can calculate β [30].

In Figure 2 we can see the simulated annealing algorithm modifications using Markov chains described above. Below we will explain how we will use the linear regression for the simulated annealing algorithm run more efficiently without losing quality in the solution.

4. Linear Regresion Method (LRM)

We explain, in Section 3.2, how to estimate the initial and final temperature for SAT instances that will be provided to the simulated annealing algorithm to determine if it is satisfiable or not.

As shown in the Figure 3, the algorithm found Metropolis various configurations with different energy at a given temperature.

The typical behavior of the energy for a given temperature can be observed in Figure 3. We set out to determine when the cycle of Metropolis reaches the equilibrium although not all

```
1  Initializing:
   Initial solution $S_i$
   Initial and final temperature: $T_i$ and $T_f$
   Calculate $n, \beta, L_{\max}$
   $T = T_i$
2  Temperatures cycle:
   $L = L_1$
3  Metropolis cycle:
   Generating $S_j$ from $S_i$
   $dif = J(S_j) - J(S_i)$
   If $dif < 0$ then
       $S_i = S_j$
   else if $e^{\frac{-dif}{T}} >$ rnd(0,1)
       $S_i = S_j$
4  Metropolis condition:
   $L = \beta L$
   if $L = L_{max}$
       goto 5
   Else
       goto 3
5  Stop criterion:
   If $T_k = T_f$
       End
   Else
       $T_k = \alpha T_k$
       goto 2
```

Figure 2. Simulated Annealing algorithm with dinamically Markov chain

of the iterations required by Markov have been executed. In order to determine this zone in adaptive way, we will fit by least squares a straight line and will stop the Metropolis cycle if the slope of this line is equal or smaller than zero. This Linear Regression Method LRM is a well known method but never was applied to detect Metropolis equilibrium in Simulated Annealing.

Suppose that the data set consists of the points:

$$(x_i, y_i), i = 1, 2, 3, ..., n \tag{30}$$

We want to find a function f such that $f(x_i) \approx y_i$. To attain this goal, we suppose that the function f is of a particular form containing some parameters $(a_1, a_2, a_3, ..., a_m)$ which need to be determined.

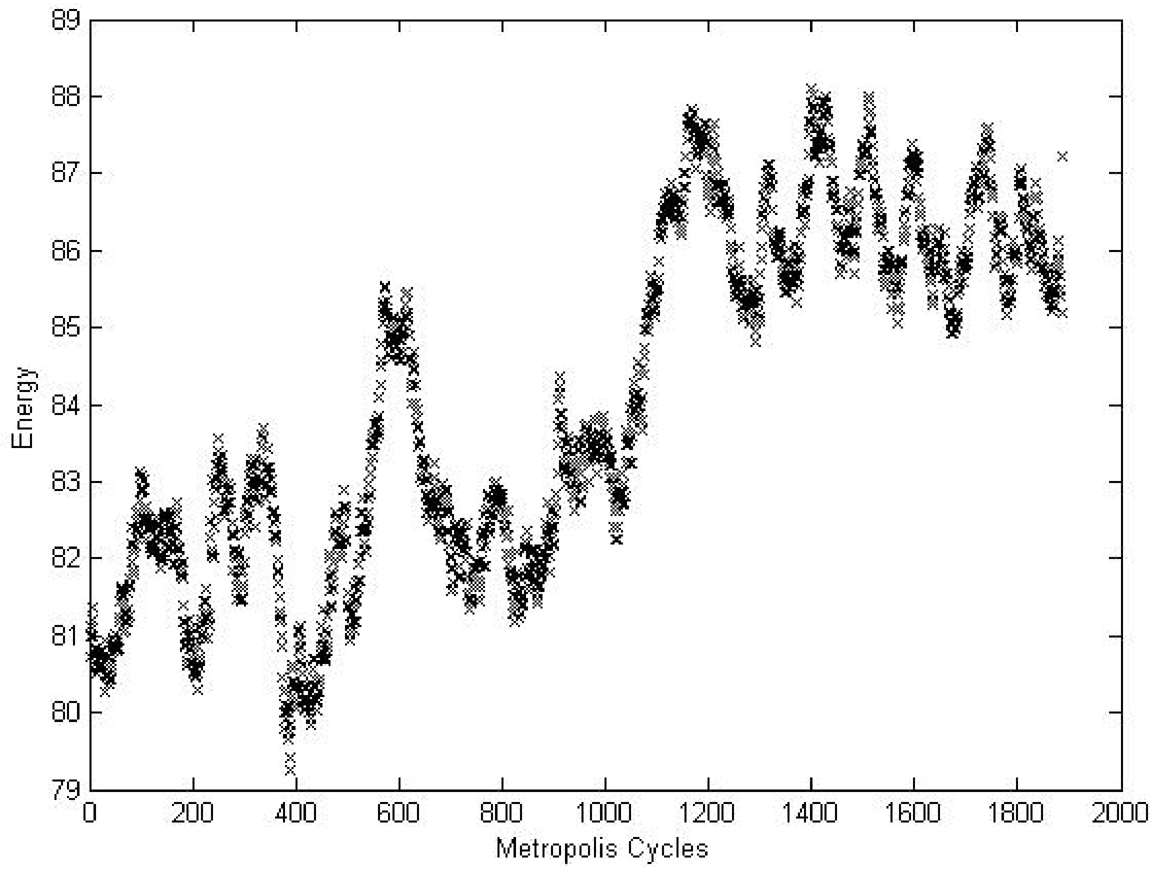

Figure 3. Energy of different states, explored in the Metropolis cycle, for a fixed temperature

In our problem:

$$y_i \approx f(x_i, a, b) = ax_i + b \tag{31}$$

In Equation 31 a and b are not yet known. In our problem $f(x_i, a, b) = f(i, a, b) = J_i$.

As usual, we now seek the values of a and b, that minimize the sum of the squares of the residuals as follows:

$$S = \sum_{i=1}^{n} [y_i - (ax_i + b)]^2 \tag{32}$$

As it is well known regression equations are obtained by differentiating S in Equation 32 with respect to each parameter a and b, and we obtain this system of linear equations:

$$a \sum_{i=1}^{n} x_i^2 + b \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i \tag{33}$$

$$a \sum_{i=1}^{n} x_i + b \sum_{i=1}^{n} 1 = \sum_{i=1}^{n} y_i \tag{34}$$

In Equation 33 and Equation 34 we can define the following constants:

$$A = \sum_{i=1}^{n} x_i^2 \, , \, B = \sum_{i=1}^{n} x_i \, , \, C = \sum_{i=1}^{n} x_i y_i \, , \, D = \sum_{i=1}^{n} y_i \tag{35}$$

Then the system of equations (Equation 33 and 34) can be rewritten as:

$$aA + bB = C \tag{36}$$

$$aB + bn = D \tag{37}$$

We recall that parameter a, the slope of Equation 31 is:

$$a = \frac{Cn - BD}{An - B^2} \tag{38}$$

In our data $x_i = 1, 2, 3, ..., n$, then we can write:

$$A = \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \tag{39}$$

and

$$B = \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \tag{40}$$

in the same way:

$$C = \sum_{i=1}^{n} iJ_i \tag{41}$$

and

$$D = \sum_{i=1}^{n} J_i \tag{42}$$

By substitution of equations: 39, 40, 41 and 42; in Equation 38 finally we get the equation

$$a = \frac{Cn - BD}{n^3 - n} \tag{43}$$

In order to apply LRM to traditional Simulated Annealing, we apply the following strategy:

1. Metropolis cycle is running as usual, just as explained in the Section 3.3, using the maximum value of Markov chain length calculated by the Equation 28 L_{max}^{C}, C is calculated $P_A^i \in \Phi_{P_A}$
2. When the repeats of metropolis, L, are equal to L_{max}^{C-1} (L_{max}^{C-1} is calculated by Equation 28 with $P_A^{i-1} \in \Phi_{P_A}$).
3. If the value of the slope a, in the equation of the line (Equation 31), found by Equation 43 is close to zero, then stop the cycle of Metropolis although this has not reached the value L_{max}.

Notice from Equation 43 and for Figure 4, that the computation of LRM is $O(n)$ where n is the number of points taken to compute the slope. So the complexity of Simulated Annealing with LRM is not affected [19, 22].

5. Experimental results

In order to prove LRM algorithm we used the SAT instances in Table 1 and Table 2. Some of these instances were generated using the programs proposed by Horie et al. in 1997 [15] and

```
1  Initializing:
   Initial solution $S_i$
   Initial and final temperature: $T_i$ and $T_f$
   Calculate $n, \beta, L_{\max}$
   $T = T_i$
2  Temperatures cycle:
   $L = L_1$
3  Metropolis cycle:
   Generating $S_j$ from $S_i$
   $dif = J(S_j) - J(S_i)$
   If $dif < 0$ then
         $S_i = S_j$
   else if $e^{\frac{-dif}{T}} >$ rnd(0,1)
         $S_i = S_j$
   If $L \geq L_{max}^{C-1}$ then
         calculate $a$
   ..........If $a \approx 0$ then
               $L = L_{max}$
   ..................goto 4
4  Metropolis condition:
   $L = \beta L$
   if $L = L_{max}$
         goto 5
   Else
         goto 3
5  Stop criterion:
   If $T_k = T_f$
          End
   Else
         $T_k = \alpha T_k$
         goto 2
```

Figure 4. Simulated Annealing algorithm with dinamically Markov chain and LRM

other are in SATLIB [14]. We generated several instances that had the same relation of clauses and variables σ [24, 25].

The measurement of efficiency of this algorithm was based on the execution time and we also obtained a solution quality measure (SQM), SQM is taken as the number of "true" clauses in an instance at the end of the program execution.

SAT problem	Id	Variables	Clauses	σ	SAT?
aim-100-1_6-yes1-1	a1	100	160	1.60	Yes
aim-50-1_6-yes1-3	a2	50	80	1.60	Yes
aim-200-1_6-no-1	a7	200	320	1.60	No
aim-50-1_6-no-2	a8	50	80	1.60	No
g2_V100_C200_P2_I1	g1	100	200	2.00	Yes
aim-50-2_0-no-4	a10	50	100	2.00	No
aim-50-2_0-yes1-1	a3	50	100	2.00	Yes
aim-50-2_0-no-3	a9	50	100	2.00	No
dubois21	d2	63	168	2.67	No
dubois26	d1	78	208	2.67	No
dubois27	d3	81	216	2.67	No
BMS_k3_n100_m429_161	b1	100	283	2.83	Yes
g2_V300_C900_P3_I1	g15	300	900	3.00	Yes
g2_V50_C150_P3_I1	g17	50	150	3.00	Yes
BMS_k3_n100_m429_368	b2	100	308	3.08	Yes
hole6	h2	42	133	3.17	No
par8-1	p1	350	1149	3.28	Yes
aim-50-3_4-yes1-2	a4	50	170	3.40	Yes
hole7	h3	56	204	3.64	No
par8-3-c	p2	75	298	3.97	Yes
par8-5-c	p3	75	298	3.97	Yes

Table 1. SAT instances for testing algorithms

Both algorithms: Simulated Annealing Algorithm with the Markov Chain Lenght dynamically (*SA_C*) and Simulated Annealing with Linear Regresion Method (*SA_LRM*), were implemented in Dell Lattitude with 1 Gb of Ram memory and Pentium 4 processor running at 2.13 GHz.

5.1. Experiment design

The experiments with these algorithms require a considerable run-time, because each instance SAT, is solved several times to take average values of performance.

Another important element to consider is to guarantee that the conditions of execution on the various algorithms are similar (because we measure time of execution on). In this regard, the first thing we did was on the Evaluation of a set of computers with similar hardware and software conditions.

To run this task, there were used programs available from the Internet, that perform different computers test and give us the result of this evaluation [29, 32, 33].

In Table 3, MOS means: million operations per second, MSS million strings per second and MBTS represent millions of bytes transferred per second. As we can see Table 3 the differences

SAT problem	Id	Variables	Clauses	σ	SAT?
g2_V100_C400_P4_I1	g3	100	400	4.00	Yes
hole8	h4	72	297	4.13	No
uuf225-045	u4	225	960	4.27	No
RTI_k3_n100_m429_150	r1	100	429	4.29	Yes
uf175-023	u1	175	753	4.30	Yes
uuf100-0789	u8	100	430	4.30	No
uf50-01	u7	50	218	4.36	Yes
uuf50-01	u9	50	218	4.36	No
ii8a2	i3	180	800	4.44	Yes
g2_V50_C250_P5_I1	g19	50	250	5.00	Yes
hole10	h1	110	561	5.10	No
ii32e1	i2	222	1186	5.34	Yes
anomaly	a6	48	261	5.44	Yes
aim-50-6_0-yes1-1	a5	50	300	6.00	Yes
g2_V50_C300_P6_I1s	g20	50	800	6.00	Yes
jnh201	j1	100	800	8.00	Yes
jnh215	j3	100	800	8.00	No
medium	m1	116	953	8.22	Yes
jnh301	j2	100	900	9.00	Yes

Table 2. SAT instances for testing algorithms, continuation

Computer	1	2	3	4	5	6	7	8	9
Floating Point Math (MOS)	168.1	168.3	168.0	168.0	168.2	168.1	167.9	168.2	168.2
Integer Maths (MOS)	33.61	33.56	33.60	33.59	33.65	33.62	33.56	33.65	33.62
Search for prime numbers (MOS)	126.6	126.6	126.4	126.5	126.7	126.7	126.3	126.6	126.6
String Sorting (MSS)	415.6	416.0	415.3	408.3	415.9	415.8	415.7	415.6	415.7
Memory blocks transfer (MbTS)	470.9	476.6	475.9	465.7	466.4	467.4	468.7	469.0	459.7
Cache memory read (MbTS)	1141	1142	1141	1141	1142	1141	1141	1142.	1088
Non cache memory read (MbTS)	831.8	831.7	832.0	831.2	830.7	831.1	830.6	831.6	831.2
Memory write (MbTS)	377.3	379.7	378.2	378.3	378.0	379.1	378.1	378.1	377.0
Overall calculation speed	232.5	232.6	232.1	232.0	232.4	232.5	232.1	232.5	232.6
Overall memory speed	209.9	210.8	210.4	209.6	209.5	210.0	209.8	209.8	209.9

Table 3. Performance of computers used for experiments

between computers are at most equal to 1.6 percent, so we can infer that we obtain similar results in one or another computer.

Each SAT instance was executed 100 times with a slow cooling function (0.99 in Equation 20 and we obtained the average time of the executions and the average quality of the solution.

The SQM is established by the next expression:

$$SQM = \frac{\text{clauses true}}{\text{total clauses}} \times 100 \tag{44}$$

Both results, SA_C and SA_LRM, were compared using two quotients which we denominated time improvement Q_{time} and quality improvement $Q_{quality}$ defined by:

$$Q_{time} = \frac{AverageTime_{SA\_LRM}}{AverageTime_{SA\_C}} \times 100 \tag{45}$$

$$Q_{quality} = \frac{SQM_{SA\_LRM}}{SQM_{SA\_C}} \times 100 \tag{46}$$

If $Q_{quality}$ is close to 100% this means that both algorithm found good solutions, however $Q_{quality}$ factor must decrease, which implies that the new algorithm SA_LRM is faster than SA_C.

	$L^{C}_{max} = 4.61$ $L^{C-1}_{max} = 3.00$		$L^{C}_{max} = 3.00$ $L^{C-1}_{max} = 2.30$		$L^{C}_{max} = 2.30$ $L^{C-1}_{max} = 1.96$	
Instance	$Q_{quality}$	Q_{time}	$Q_{quality}$	Q_{time}	$Q_{quality}$	Q_{time}
a1	99.4	13.0	99.3	12.9	99.2	11.7
a10	99.0	11.0	99.2	11.6	99.6	9.7
a2	99.5	11.4	98.7	11.2	99.0	9.8
a3	99.9	11.3	99.4	11.6	99.2	9.6
a4	99.2	9.8	99.6	10.4	98.9	8.8
a5	99.5	10.1	99.1	10.6	99.5	9.0
a6	99.0	34.5	99.3	32.1	99.2	27.8
a7	98.7	13.3	99.1	13.9	98.8	11.8
a8	99.6	11.3	99.2	12.2	99.4	9.8
a9	99.2	10.9	98.7	11.6	99.4	9.5
b1	99.6	54.0	99.7	51.0	99.9	46.0
b2	100.2	57.7	99.8	53.5	99.9	49.2
d1	99.2	11.8	99.4	11.6	99.2	10.1
d2	99.5	11.6	99.5	11.5	99.4	10.1
d3	100.0	11.5	99.1	11.3	99.1	9.5
g1	99.8	71.7	99.9	69.7	99.8	64.7
g15	99.7	28.7	99.7	28.3	99.9	29.0
g17	99.3	11.7	99.2	12.7	99.5	10.5
g19	99.6	11.5	98.6	11.5	99.3	9.8.0
g20	99.5	10.5	99.5	11.1	99.4	9.5.0

Table 4. Experimentals results

	$L^{C}_{max} = 4.61$ $L^{C-1}_{max} = 3.00$		$L^{C}_{max} = 3.00$ $L^{C-1}_{max} = 2.30$		$L^{C}_{max} = 2.30$ $L^{C-1}_{max} = 1.96$	
Instance	$Q_{quality}$	Q_{time}	$Q_{quality}$	Q_{time}	$Q_{quality}$	Q_{time}
g3	99.8	20.8	99.6	21.2	99.6	18.4
h1	100.1	54.5	99.6	53.5	99.9	47.1
h2	99.7	22.0	99.3	23.2	99.5	19.1
h3	100.1	30.5	99.7	29.7	99.5	24.9
h4	98.9	41.0	99.3	38.5	99.0	32.7
i2	99.1	55.0	98.9	54.4	99.7	45.7
i3	100.0	65.0	100.2	65.4	100.3	58.7
j1	99.7	11.6	99.7	12.8	99.6	11.1
j2	99.7	11.6	99.8	12.3	99.8	10.4
j3	99.5	11.3	99.7	12.3	99.8	10.2
m1	100.5	74.5	99.9	70.9	99.7	66.8
p1	99.9	66.0	99.9	65.4	99.9	59.7
p2	99.4	13.8	99.0	13.3	99.3	11.8
p3	99.3	14.0	99.5	12.4	99.5	11.0
r1	99.4	24.0	99.7	22.6	99.5	21.1
u1	99.6	49.6	100.0	49.8	100.0	44.5
u4	99.7	53.3	100.0	50.9	99.3	48.1
u7	99.9	14.7	100.0	15.2	99.5	13.1
u8	99.7	26.0	100.0	23.9	99.6	20.0
u9	99.1	13.8	99.5	14.0	99.2	11.9

Table 5. Experimentals results, continuation

From Table 4 and Table 5 experimental results we can obtain the average values for the magnitudes Q_{time} and $Q_{quality}$, as shown in the following Table 6.

As you can see, in Table 6, the quality factor of the solutions, $Q_{quality}$ is very close to 100%, which implies that the *SA_LRM* algorithm finds solutions as good as the *SA_C* algorithm, it is important to note that 37% of SAT instances used for the experiments, are not-SAT, which implies that their respective *SQM* can not be equal to 100% and therefore the quality factor must be less than 100%.

Also in Table 6, we see that the factor Q_{time} diminishes values less than 30%, showing that our algorithm, *SA_LRM*, is 70% faster than the *SA_C* algorithm but maintains the same quality of the solution.

As shown in Figure 5 are some instances in which the reduction of run time is only 25% while other reducing runtime up to 90%.

L^{C}_{max}	L^{C-1}_{max}	$Q_{time}(\%)$	$Q_{quality}(\%)$
4.61	3.00	99.6	27.3
3.00	2.30	99.5	26.8
2.30	1.96	99.5	23.8

Table 6. Q_{time} and $Q_{quality}$ averages for all instances tested

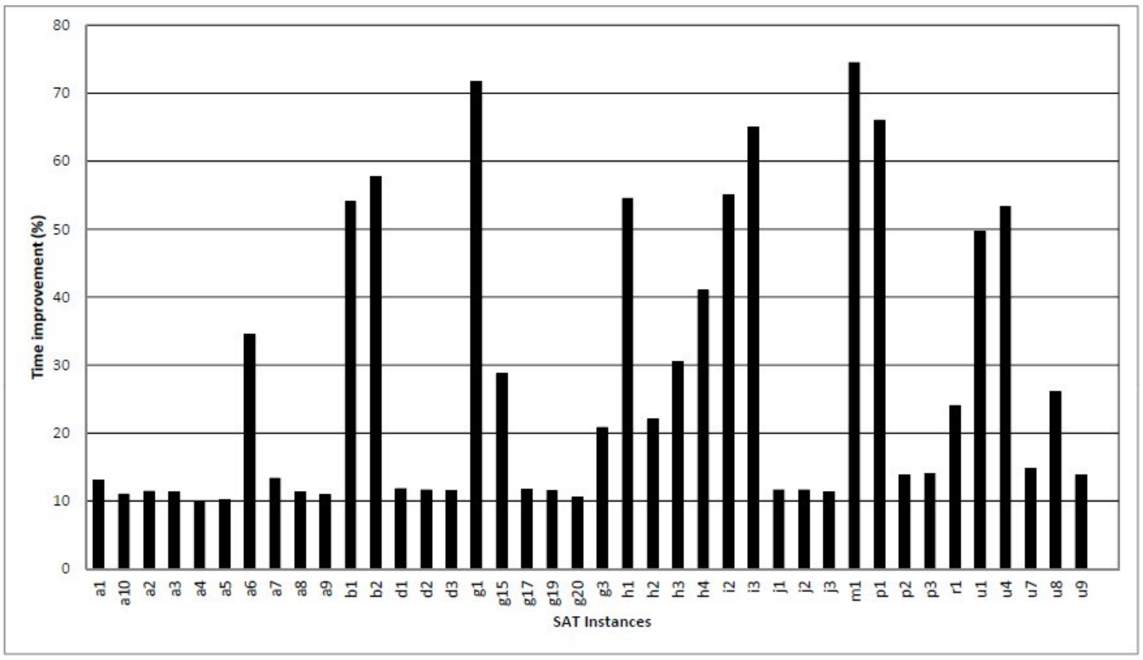

Figure 5. Q_{time} for SAT instances

6. Conclusions

In this paper a new adaptive Simulated Annealing algorithm named SA_LRM that uses least squares method as a way to find the equilibrium zone in Metropolis cycle is presented. When this zone is found our algorithm abort the Metropolis cycle, although the iterations calculated with the dynamic chains of Markov have not been completed. After experimentation we show that SA_LRM is more efficient than those tuned using only an analytical method.

Author details

Felix Martinez-Rios
Universidad Panamericana, México

Juan Frausto-Solis
UPMOR, México

7. References

[1] Aarts, E. & Korst, J. [1989]. *Simulated annealing and Boltzman machines: An stochastic approach to combinatorial optimization and neural computing*, John Wiley and Sons.

[2] Aspvall, B. & Tarjan, M. F. P. R. E. [1979]. Alinear-time algorithm for testing the truth of certain quantified boolean formulas, *Information Processing Letters* 8(3).

[3] Atiqullah, M. [2004]. An efficient simple cooling schedule for simulated annealing, 3045: 396–404.

[4] Bertsimas, D. & Tsitsiklis, J. [1993]. Simulated annealing, *Statistical Science* 8: 10–15.

[5] Cerny, V. [1982]. A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm, *Comenius University* .

[6] Cerny, V. [1985]. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm, *Journal of Optimazation Theory and Applications* 45(1).

[7] Cook, S. A. [1971]. *The complexity of theorem proving procedures*, Proceedings on the third annual ACM symposium on theory of computing.

[8] Crescenzi, P. & Kann, V. [1998]. How to find the best approximation results - a follow-up to garey and johnson, *ACM SIGACT News* 29(4): 90–97.

[9] Dowling, W. F. & Gallier, J. H. [1984]. Linear-time algorithms for testing the satisfiability of propositional horn formulae, *Journal of Logic Programming* 1(3): 267–284.

[10] Even, S., Itai, A. & Shamir, A. [1976]. On the complexity of timetable and multicommodity flow problems, *SIAM Journal on Computing* 5(4): 691–703.
URL: *http://link.aip.org/link/?SMJ/5/691/1*

[11] Faigle, U. & Kern, W. [1991]. Note on the convergence of simulated annealing algorithms, *SIAM journal on control and optimization* 29(1): 153–159.
URL: *http://e-archive.informatik.uni-koeln.de/67/*

[12] Fleischer, M. A. [1996]. Cybernetic optimization by simulated annealing: Accelerating convergence by parallel processing and probabilistic feedback control, *Journal of Heuristics* 1: 225–246.

[13] Frausto-Solis, J., Sanvicente, H. & Imperial, F. [2006]. Andymark: An analytical method to establish dynamically the length of the markov chain in simulated annealing for the satisfiablity problem, *Springer Verlag* .

[14] Hoos, H. H. & Stutzle, T. [2000]. Satlib: An online resource for research on sat, *SAT 2000* pp. 283–292. Disponible en http://www.satlib.org/.

[15] Horie, S. & Watanabe, O. [1997]. Hard instance generation for sat, *ISAAC '97: Proceedings of the 8th International Symposium on Algorithms and Computation*, Springer-Verlag, pp. 22–31.

[16] Ingber, L. [1993]. Simulated annealing: Practice versus theory, *Mathematical and Computer Modelling* 18(11): 29 – 57.

[17] Ingber, L. [1996]. Adaptive simulated annealing (asa): Lessons learned, *Control and Cybernetics* 25: 33–54.

[18] Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. [1983]. Optimization by simulated annealing, *Science* (4598)(220): 671–680.

[19] Martinez-Rios, F. & Frausto-Solis, J. [2007]. A hybrid simulated annealing threshold accepting algorithm for satisfiability problems using dynamically cooling schemes, *Electrical and Computer Engineering Series WSEAS* pp. 282–286.

[20] Martinez-Rios, F. & Frausto-Solis, J. [2008a]. Golden annealing method for job shop scheduling problem, *Mathematics and Computers in Science and Engineering, ISSN 1790-2769* .

[21] Martinez-Rios, F. & Frausto-Solis, J. [2008b]. Golden ratio annealing for satisfiability problems using dynamically cooling schemes, *Lecture Notes in Computer Science* 4994: 215–224.

[22] Martinez-Rios, F. & Frausto-Solis, J. [2008c]. Simulated annealing for sat problems using dynamic markov chains with linear regression equilibrium, *MICAI 2008, IEEE* pp. 182–187.

[23] Metropolis, N., Rosenbluth, A. W. R. M. N. & Teller, A. H. [1953]. Equation of state calculations by fast computing machines, *The journal of Chemicla Physics* 21: 1087–1092.

[24] Mezard, M., Parisi, G. & Zecchina, R. [2002]. Analytic and algorithmic solution of random satisfiability problems, *Science* 297(5582): 812–815.

[25] Mezard, M. & Zecchina, R. [2002]. The random k-satisfiability problem: from an analytic solution to an efficient algorithm, *Phys. Rev.* E66(056126).

[26] Miki, M., Hiroyasu, T. & Ono, K. [2002]. Simulated annealing with advanced adaptive neighborhood, *Second international workshop on Intelligent systems design and application*, Dynamic Publishers, Inc., pp. 113–118.

[27] Munakata, T. & Nakamura, Y. [2001]. Temperature control for simulated annealing, *PHYSICAL REVIEW E* 64.

[28] Papadimitriou, C. H. [1994]. *Computational Complexity*, Addison-Wesley.

[29] PassMark [2007]. Cpu burnintest. http://www.passmark.com/download/index.htm. URL: *http://www.passmark.com/download/index.htm*

[30] Sanvicente-Sanchez, H. & Frausto-Solis, J. [2004]. A method to establish the cooling scheme in simulated annealing like algorithms, *Lecture Notes in Computer Science* .

[31] Schaefer, T. J. [1978]. The complexity of satisfiability problems, *Proceedings of the tenth annual ACM symposium on Theory of computing*, STOC '78, ACM, pp. 216–226. URL: *http://doi.acm.org/10.1145/800133.804350*

[32] van Wandelen, C. J. [2007]. Cpubench. http://cpubench.softonic.com/eie/21660. URL: *http://cpubench.softonic.com/eie/21660*

[33] Vorte, B. [2007]. Cpumathmark. http://files.aoaforums.com/code.php?file=931. URL: *http://files.aoaforums.com/code.php?file=931*