An Hybrid Simulated Annealing Threshold Accepting Algorithm for Satisfiability Problems using Dynamically Cooling Schemes

FELIX MARTINEZ-RIOS[1] and JUAN FRAUSTO-SOLIS[2]

[1]Universidad Panamericana, Campus Ciudad de México
Augusto Rodín 498, Col. Insurgentes Mixcoac, 03920, Distrito Federal
MEXICO
fmartin@up.edu.mx
[2]Tecnológico de Monterrey, Campus Cuernavaca
Autopista del Sol km 104, Colonia Real del Puente, 62790, Xochitepec, Morelos
MEXICO
juan.frausto@itesm.mx

Abstract: For Satisfiability (SAT) Problem there is not a deterministic algorithm able to solve it in a polynomial time. Simulated Annealing (SA) and similar algorithms like Threshold Accepting (TA) are able to find very good solutions of SAT instances only if their control parameters are correctly tuned. Classical TA's algorithms usually use the same Markov chain length for each temperature cycle but they spend a lot of time. In this paper a new hybrid algorithm is presented. This algorithm is in fact a TA algorithm which is hybridized with SA in a certain way. For this TA algorithm, the Markov chain length (L) is obtained in a dynamical way for each temperature. Besides, it is known that TA and SA obtain very good results whether their parameters are correctly tuned. Experimental tuning methods expend a lot of time before a TA algorithm can correctly be executed; in other hand, analytical tuning methods for TA were only completely developed for the geometrical cooling function. This paper also shows how TA can be tuned for three common cooling functions with an analytical model. Experimentation presented in the paper shows that the new TA algorithm is more efficient than the classical one.

Key-Words: Simulated Annealing, Threshold Accepting, Cooling function, Dynamic Markov Chains, SAT problem

1 Introduction

Satisfiability Problem (SAT) is an NP-complete (NP in short) problem which is fundamental to complexity theory [1, 2] and is widely studied in several areas such as: planning, circuit testing, temporal reasoning, scheduling and many others [3]. Besides, any instance of an *NP* problem can be transformed to a SAT instance by using a polynomial transformation [1, 4]. Therefore, if SAT can be solved efficiently with a particular algorithm, then similar results could be obtained for other *NP* problems using the same algorithm [5]. It other words, NP has the property found by Cook and others: "If and efficient algorithm for any problem in NP is found, it could be adapted to solve all the other NP problems as well" [5].

It is common to say that a polynomial algorithm π solve all the instances where π in an efficient way; in other words polynomial and efficient algorithms are considered as synonyms. A polynomial algorithm has a temporal function t(n) (where n is the instance size) measuring the execution time to solve all the instances of π; examples of t(n) can be n, n^2, 2^n and so on. Polynomial and exponential time algorithms are frequently referred as "good algorithms" (i.e. efficient) and "bad algorithms", (i.e. not efficient) respectively [6]; however this classification is not always correct, as can be noticed in the following examples [7]:

- A polynomial algorithm requires n^{1230} steps to find the answer of a problem which size is $3<n<10$. An exponential algorithm designed for the same problem, requires 2^n steps to find the solution. It is obvious that this polynomial algorithm cannot be classified as a good algorithm even though it is a polynomial one. Besides this polynomial algorithm has a worse performance that the

previous exponential one (and many other exponential algorithms).

- Cook [8] shows another example of a particular polynomial algorithm which is not a good one because it requires n^{100} steps (it is impractical even for n values around *1000*).
- It is well known than the old algorithm for linear programming known as Simplex [9] has an exponential complexity. However it is also well known that for many linear optimization problems Simplex is the best one [9, 10].

In the case of SAT, it is reported that by using complete methods (all of them being deterministic methods) it is very difficult to find the optimal solution or a solution close to the optimal one. Nowadays, most of the time, random methods have a better performance than complete methods for SAT instances. Therefore, in general random methods require less execution time to obtain good solutions than complete methods. Here, a "good solution" means the optimal one or a solution close to it.

Since the seminal papers of Simulated Annealing algorithm (SA) [11, 12], this algorithm has shown to be very efficient for solving combinatorial optimization problems. Due to this, new algorithms based on SA have been proposed; this kind of algorithms is referred as SALA algorithms (Simulated Annealing Like Algorithms). The classical SA and Threshold Accepting algorithm (TA) [13] are the most popular SALA algorithms. TA is similar to SA except for a small modification, which purpose is to reduce the execution time with similar quality of the final solution obtained by SA.

In this paper a new hybrid algorithm is presented. This algorithm is in fact a TA but hybridized with SA in a particular way. Besides, an analytical adaptive method to establish the initial and final temperatures and the length of each Markov chain in a dynamic way for TA algorithm is presented. Experimentation with a set of SAT instances shows that this new TA algorithm has a better performance than the classical one. This experimentation is done with three cooling functions. In addition, it is also presented analytical methods for TA tuning parameters for all of these cooling functions.

2 Simulated Annealing Like Algorithms (SALA)

A Simulated Annealing Like Algorithm (SALA) is an algorithm that works with a Simulated Annealing (SA) approach [14]. The classical SA of Kirkpatrick and Cerny [11, 12] and Threshold Accepting (TA) [13] among many others can be classified in this category.

As was mention before, SA is a simple and effective optimization method to find near optimal solutions to many instances of NP problems [2]. A SA algorithm may be seen like a Markov chain sequence [17] (a homogeneous one); L_k identifies the length of each Markov chain and obviously L_k must be greater than zero (where k is the sequence index). The states in a Markov chain are established by the solution space S of the optimization problem. The sequence of Markov chains is built on a descending sequence of a control parameter ($c_k>0$) commonly referred as the temperature. The output of a Markov chain is a solution $S_{eq}^{\ k} \in S$, where $S_{eq}^{\ k}$ is a solution when the dynamic equilibrium or the stationary distribution is reached. This control parameter must satisfy the following property:

$$\lim_{k\to\infty} c_k = 0 \quad (1)$$

$$c_k \geq c_{k+1}, \forall k \geq 1 \quad (2)$$

Consecutive temperatures c_k's are setting through a cooling function:

$$c_{k+1} = f(c_k) \quad (3)$$

SA does a stochastic walk on the solution space of the optimization problem. For each Markov chain, this stochastic walk is done until the stationary distribution is reached. During the stochastic walk, the accepted solutions depend on the temperature parameter; it should be remarked that during this stochastic walk, any solution with a worse cost (i.e. a cost deterioration) than the previous one is accepted with a Boltzmann distribution probability. It should be also taken into account that the acceptance probability decreases along the iterations (i.e. if the temperature decreases, then the acceptance probability is decremented).

TA [13] also does a stochastic walk on the solution space and it also uses a cooling function to control the transition probabilities among solutions in order to accept solutions with a cost deterioration (for a minimization problem, a cost deterioration means a greater cost in the new iteration). The distribution probability (usually Boltzmann in SA) is

now a hidden distribution probability which handles a parameter known as "threshold". It is common that in any iteration, the current temperature is the threshold parameter.

One of the main characteristics of SALA algorithms is the asymptotic convergence to the optimal solution. For this reason SALA are classified as approximation algorithms. Therefore, a good balance between efficiency (i.e. execution time) and efficacy (i.e. quality solution) need to be established.

Since the publication of the seed paper of SA algorithm [11, 12], several methods and procedures have been proposed to reduce the execution time of SA. These methods have mainly been focused on the cooling function parameters; although developed for SA, some of these methods have being extended to SALA algorithms since many years ago. However, most of these methods are based on experimentation and usually their tuning process requires a huge amount of experimentation, money and time. It will be advantageous to find a practical tuning method with an experimentation time as small as possible. Some alternatives tuning approaches have been proposed: a) Analytical Markov approaches [15] and b) Adaptive methods [16].

In fact both of these approaches uses some experimentation (but it is reduced) to define the SALA parameters. Trousset [15] presents a general framework to derive the SALA parameters using a Markov model; from them a general mathematical model is developed in [14]; these "analytical" methods use a set of formulas to derive the initial and final temperatures and the number of iterations in the metropolis cycle as well. Adaptive Methods adjust SALA parameters depending on the results obtained in the objective function; for instance the new temperature can be reduced in function of the improvement obtained in the previous metropolis cycle. The Hybrid SA presented here uses the advantages of both approaches.

2.1 SAT problem

SAT was the first problem referred to be as NP–complete [8] and is fundamental to the analysis of the computational complexity of many problems [2]. An instance of SAT is a boolean formula which consists on the next components:

1. A set of n variables $x_1, x_2, \ldots, x_n$
2. A set of literals; a literal is a variable x_i or its negation $\neg x_i$.
3. A set of m clauses: $C_1, C_2, \ldots, C_m$ linked by the logical connective AND ($\wedge$) where each clause consists of literals linked by the logical connective OR ($\vee$).

This is:

$$\Phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m \quad (4)$$

where Φ is the SAT instance. Then the SAT problem can be enunciated as follows:

Definition 1:

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

For instance, the following 4-variables SAT instance:

$$\Phi = (x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_3 \vee x_4)$$

is formed by four clauses. Φ is made true when S1= {x_1=false, x_2=true, x_3=true, x_4=true}. The same happens with S2= {x_1=false, x_2=false, x_3=true, x_4=true}. S1 and S2 are known as solutions of the SAT instance.

Any SAT instance can be represented as an optimization problem. Hence this problem is known as the Maximum Satisfiability problem or MAX-SAT problem.

The formulation of SAT to MAX-SAT is carried through by introducing the next objective function:

$$\max \quad Z = \sum_{j=1}^{m} C_j \quad (5)$$

where C_j is the j-th clause of m.

The goal in this problem is maximize (minimize) the number of true (false) clauses. In this sense, C_j=1 if the j-th clause is made true or C_j=0 if it is made false.

3 Classic TA algorithm

Threshold Accepting algorithm (TA) [13], is very similar to SA; it has been applied to many areas, such as Databases [18], Bin Packing [19] and many

others. SA and TA accept bad solutions in order to escape of local optima. However, TA does not use Boltzmann Distribution to accept bad solutions but the threshold deterministic parameter mention before. This parameter is usually the current temperature (c) as is shown in Fig. 1.

TA is similar to SA since it begins with a current solution S_i from which a new solution S_{new} is generated. In the algorithm c_i and c_f represent the initial and final temperatures respectively. Notice that TA has also two cycles:

- The outer cycle or temperature cycle which is between lines 2–13. This cycle controls the threshold value (the current temperature c)
- The inner cycle or Metropolis cycle which is between lines 3–11. In this cycle the stochastic walk discussed before is done for each temperature cycle.

As we can see in Fig. 1, once the algorithm executes the line twelve, a new temperature is generated using the cooling function [20]. A new solution is always accepted (line 7) whether the cost of a new solution ($Z(S_{new})$) is lower than the previous one ($Z(S_i)$) or if their difference is smaller than the threshold parameter c.

The outer loop parameters define the cooling scheme of TA:

- The initial temperature c_i,
- The final temperature c_f, and
- The cooling function (it is shown in line 12 of Figure 1).

In the experiments with each SAT instances three cooling functions were used [20], which are in the following equations:

$$c_{k+1} = \alpha c_k \quad \text{(geometric)} \tag{6}$$

$$c_{k+1} = \exp(-\alpha) c_k \quad \text{(exponential)} \tag{7}$$

$$c_{k+1} = \frac{c_k}{\ln(\alpha)} \quad \text{(logarithmic)} \tag{8}$$

The inner or Metropolis cycle is determined by the length of each Markov chain L (i.e. its iterations' number).

3.1 Markov Chains Length

As it was previously discussed, TA makes a stochastic walk on the solution space which can be modeled as a sequence of homogeneous Markov chains. In this sequence, every Markov chain has a length L, lower than the previous one; this length is calculated by using a function of the temperature control parameter c>0 which generates descending L values. Obviously any Markov chain length is always greater that zero. Therefore if $L_k>0$ is the length of a Markov chain for the temperature c_k, we have:

$$L_k > 0; \lim_{k\to\infty} c_k = 0, c_k \geq c_{k+1}, \forall k \geq 1 \tag{9}$$

1. Initialization (S_i, $c=c_i$)
2. Repeat
3. Repeat
4. S_{new} = Generate (S_i)
5. $\Delta Z = Z(S_{new}) - Z(S_j)$
6. If $Z(S_{new}) < Z(S_j)$ then
7. $S_i = S_{new}$
8. Else
9. If $\Delta Z < c$
10. $S_i = S_{new}$
11. Until the equilibrium
12. c = New(α,c)
13. Until ($c=c_f$)

Fig. 1. Pseudo–code of TA algorithm.

As can be observed, c_k and L_k have a strong relation. This can be easily explained for SA and then for TA as follows:

- First, when the process is at the beginning the temperature is very high. This is because in the Boltzman distribution the acceptance probability is directly related with the cost increment: $P_A = \exp(\Delta Z / T_k)$; where T_k is the temperature parameter c_k. Therefore $c_k = -\Delta Z / Ln(P_A)$ where $0<P_A<1$.
- At the beginning of the process, P_A is close to one 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 and the temperature

are closer to zero but the Metropolis Length is very long.

- The arguments described in the later paragraphs are represented by the next properties:

1. when $c_k \to \infty, L_k \to 0$ and
2. when $c_k \to 0, L_k \to \infty$

The control parameter is set by a cooling function like $c_{k+1} = f(c_k)$. At the beginning of the process the temperature has a high value and the probability to accept one proposed solution is high. When the temperature decreases this probability also decreases and only good solutions are accepted at the end of the process. In this sense every Markov chain makes a stochastic walk in the solution space until the stationary distribution is reached. Because Markov chains are built through a neighborhood sampling method, the maximum number of different solutions rejected at c_f when the current solution S_i is the optimal one. The Markov chain length is the neighborhood size $|V_{S_i}|$ (see Definition 2). In general L_k can be established as:

$$L_k \le L_{\max} = g(|V_{S_i}|) \quad (10)$$

where $g(|V_{S_i}|)$ 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 c_f.

Usually an SA algorithm uses a uniform probability distribution function given by a random replacement sampling method to explore V_{S_i} at any temperature c_k [21]. In this way, the probability to obtain the solution S_j in N samples is:

$$P(S_j) = 1 - \exp\left(\frac{N}{|V_{S_i}|}\right) \quad (11)$$

The length of the Markov Chain in TA can be taken in the same way that SA, assuming that an iterative use of threshold functions emulates the Boltzmann distribution. Therefore the maximum Markov chain L_{max} at the final temperature can be calculated with (11):

$$L_{\max} = -Ln(P_R(S_i))|V_{S_i}| \quad (12)$$

Where $|V_{S_i}|$ represents the neighborhood size and $P_R(S_i)$ is the rejection probability for a solution S_i (or a proportion of the solution space).

We can define $C = -Ln(P_R(S_i))$. C ranges from 1 to 4.6 which guarantee a good exploration level of the neighborhood at the final temperature; for instance if C=4.6 then P_R represents the exploration of 99% of the solution space.

Therefore $L_k = g(|V_{S_i}|)$; this function gives the maximum number of samples that must be taken from the neighborhood V_{Si} in order to evaluate an expected fraction of different solutions in a Markov chain. L_k depends only on the number of elements of V_{Si} that will be explored at c_k.

Because the strong relation between c_k and L_k, at the beginning of the process ($c_k = c_i$), any solution has the same acceptance probability. Therefore, as in SA, the first Markov chain length in TA must be as small as possible ($L_1 \approx 1$). When k is increased, c_k is decremented until it reaches c_f. Therefore, for consecutive values of c_k, TA is forced to increment its Markov chain length in order to reach its stationary probabilistic state.

Thus, L_k is incremented since its lower value (i.e. one) at c_i until it achieves is maximum value L_{max} at c_f. As a consequence, an incremental Markov chain function can be proposed for each cooling scheme (Equations 6, 7 and 8) as follows:

$$L_{k+1} = \beta L_k \quad \text{(geometric)} \quad (13)$$

$$L_{k+1} = \exp(-\beta) L_k \quad \text{(exponential)} \quad (14)$$

$$L_{k+1} = \frac{L_k}{\ln(\beta)} \quad \text{(logarithmic)} \quad (15)$$

Where $\beta > 1$ is called incremental coefficient and again L_k is the length of the Markov chain at c_k, and L_{k+1} represents the length of the Markov chain at c_{k+1}.

Because the Markov chain length is incremented from L_1 to L_{max} when c_k varies from c_1 to c_f in a Markov process we have:

$$L_{\max} = \beta^n L_1 \quad \text{(geometric)} \quad (16)$$

$$L_{\max} = \exp(-\beta)^n L_1 \quad \text{(exponential)} \quad (17)$$

$$L_{\max} = \frac{L_1}{\ln(\beta)^n} \quad \text{(logarithmic)} \quad (18)$$

Then β can be obtained from the previous equations as follows:

$$\beta = \exp\left(\frac{Ln(L_{\max}) - Ln(L_1)}{n}\right) \quad \text{(geometric)} \quad (19)$$

$$\beta = \frac{Ln(L_1) - Ln(L_{\max})}{n} \quad \text{(exponential)} \quad (20)$$

$$\beta = \exp\left(\exp\left(\frac{Ln\left(\frac{L_1}{L_{\max}}\right)}{n}\right)\right) \quad \text{(logarithmic)} \quad (21)$$

For each cooling scheme (Equations 6, 7 and 8) the *n* number of steps performed from c_i to c_f can be represented for:

$$c_f = \alpha^n c_i \quad \text{(geometric)} \quad (22)$$

$$c_f = \exp(-\alpha)^n c_i \quad \text{(exponential)} \quad (23)$$

$$c_f = \frac{c_i}{\ln(\alpha)^n} \quad \text{(logarithmic)} \quad (24)$$

Then n can be calculated from Equations 22, 23 and 24 as follows:

$$n = \frac{Ln(c_f) - Ln(c_i)}{Ln(\alpha)} \quad \text{(geometric)} \quad (25)$$

$$n = \frac{Ln(c_i) - Ln(c_f)}{\alpha} \quad \text{(exponential)} \quad (26)$$

$$n = \frac{Ln\left(\frac{c_i}{c_f}\right)}{Ln(Ln(\alpha))} \quad \text{(logarithmic)} \quad (27)$$

3.2 Initial and final temperatures in TA

In TA, c_i and c_f are explicit bounds since they determine the beginning and the end of the process of it. At the beginning c_i must be determined in such a way that almost all the transitions may be accepted. If c_i is too high TA will expend a lot of time, but if it is too small the probability to get stuck on a local minimum is high.

On the other hand, if c_f is set too high TA probably does not have a good exploration of the solution space. If c_f is set to a very low value a lot of time will be wasted at the final of the process. In this paper c_i and c_f was fixed with the method suggested in [21]. In this sense, the neighborhood structure can be defined as follows:

Definition 2:

Let

$$\{\forall S_i \in S, \exists a \quad set \quad V_{S_i} \subset S \mid V_{S_i} = V : S \to S\}$$

be the neighborhood of a solution S_i, where $\forall S_i$ is the neighborhood set of S_i, $V : S \to S$ is a mapping and S is the solution space of the problem being solved.

In this definition, it can be noticed that the neighbors of S_i depend only on the neighborhood structure V from every particular problem, and then the maximum and minimum cost increments produced from this neighborhood structure are [21]:

$$\Delta Z_{V_{\max}} = Max\{Z(S_j) - Z(S_i)\} \quad \forall S_j \in V_{S_i}, \forall S_i \in S \quad (28)$$

$$\Delta Z_{V_{\min}} = Min\{Z(S_j) - Z(S_i)\} \quad \forall S_j \in V_{S_i}, \forall S_i \in S \quad (29)$$

Then c_i and c_f are calculated as the minimum and maximum deterioration of the objective function:

$$c_i = \Delta Z_{V_{\max}} \quad (30)$$

$$c_f \leq \Delta Z_{V_{\min}} \quad (31)$$

3.3 Initial and final temperatures in SA

SA (see Fig 2) like TA begins with a current solution S_i from which a new solution S_{new} is generated. In the algorithm c_i and c_f represent the initial and final temperatures respectively. SA has two cycles: the outer cycle (lines 2–13) that controls the current temperature *c* and the inner cycle in lines 3 to 11. As can be seen in line 12 a new temperature is generated using a cooling function. A new solution is always accepted (line 7) if the cost of a new solution ($Z(S_{new})$) is lower than the previous one ($Z(S_i)$) or if expression: *rnd*<exp(-ΔZ) is true, where *rnd* is a uniform distributed random number into the interval (0,1).

The inner or Metropolis cycle is determined by the length of each Markov chain (i.e. its iterations' number.

1. Initialization (S_i, $c=c_i$)
2. Repeat
3. Repeat
4. S_{new} = Generate (S_i)
5. $\Delta Z = Z(S_{new}) - Z(S_j)$
6. If $Z(S_{new}) < Z(S_j)$ then
7. $S_i = S_{new}$
8. Else
9. If rnd<exp(-ΔZ)
10. $S_i = S_{new}$
11. Until the equilibrium
12. c = New(α,c)
13. Until ($c=c_f$)

Fig. 2. Pseudo–code of SA algorithm.

In SA the maximum Markov chain L_{max} occurs at the final temperature, and $L_{\max} = C|V_{S_i}|$, where $|V_{S_i}|$ is the neighborhood size.

$C = -Ln(P_R(S_i))$ and $P_R(S_i)$ is the rejection probability for a solution S_i. C ranges from 1 to 4.6 that guarantee a good exploration level of the neighborhood at the final temperature. Different exploration levels P_R can be applied, for instance if 99% of the solution space is going to be explored, then $C = 4.6$. Let $\Delta Z_{V\max}$ and $\Delta Z_{V\min}$ be the maximum and minimum cost deteriorations of the objective function through the neighborhood set V. Then the initial and final temperatures c_i and c_f are [14]:

$$c_i = \frac{-\Delta Z_{V\max}}{\ln(P_A(\Delta Z_{V\max}))} \quad (32)$$

$$c_f = \frac{-\Delta Z_{V\min}}{\ln(P_A(\Delta Z_{V\min}))} \quad (33)$$

Notice in line 9 in Fig. 2, that SA produces different c_i and c_f parameters because of Boltzmann distribution [21].

4 Experiments Executed

TA can be executed with its own parameters or with some of them taken from SA. Therefore the following cases were tested:

1. TA pure: c_i and c_f are calculated using Equations 30 and 31 respectively.
2. TA with final temperature of SA: c_i obtained by TA pure (Equation 30) but c_f calculated as SA using Equation 33.
3. TA with initial temperature of SA: c_f obtained by TA pure (Equation 31) but c_i calculated as SA (using Equation 32).
4. TA hybridized with SA: c_i and c_f are calculated as TA (Equations 30 and 31) pure when $c=c_f$ then running SA with the last solution obtained by TA, and c_f is calculated as SA (using Equation 33).

Table 1 shows several SAT instances with different σ relations of clauses/variables [22]; they were taken from SATLIB or generated with Hories algorithm [23].

The measurement of efficiency is based on the execution time; a quality solution measure is defined as the percentage of "true" clauses with respect to the total clauses in an instance at the end of the execution program.

Both algorithms were implemented in a Dell Intel Core Duo with 2 Gb of RAM memory and Pentium 4 processor running at 2.40 GHz.

Each instance was executed 40 times and then the average execution time and the average quality solution were obtained. A quality solution measure can be defined by the equation:

$$Q = \frac{clauses \quad true}{total \quad clauses} \times 100 \qquad (34)$$

The alpha's values used for each cooling function (Equations 6, 7 and 8) were obtained experimentally as 0.99, 0.01 and 2.745 respectively. Notice than once the alpha value is obtained for the geometrical cooling function (Equation 6), the other alpha's values can be obtained by analyzing when equivalent results are obtained with the other cooling functions.

Table 1. SAT instances tested.

Instance	Id	σ	Sat?
aim-100-1_6-yes1-1	a1	1.60	Yes
aim-50-1_6-yes1-3	a2	1.60	Yes
aim-200-1_6-no-1	a7	1.60	No
aim-50-1_6-no-2	a8	1.60	No
aim-50-2_0-no-4	a10	2.00	No
g2_V100_C200_P2_I1	g1	2.00	Yes
BMS_k3_n100_m429_161	b1	2.83	Yes
g2_V300_C900_P3_I1	g15	3.00	Yes
g2_V50_C150_P3_I1	g17	3.00	Yes
BMS_k3_n100_m429_368	b2	3.08	Yes
par8-1	p1	3.28	Yes
aim-50-3_4-yes1-2	a4	3.40	Yes
par8-3-c	p2	3.97	Yes
par8-5-c	p3	3.97	Yes
g2_V100_C400_P4_I1	g3	4.00	Yes
uuf225-045	u4	4.27	No
RTI_k3_n100_m429_150	r1	4.29	Yes
uf175-023	u1	4.30	Yes
uuf100-0789	u8	4.30	No
uf50-01	u7	4.36	Yes
uuf50-01	u9	4.36	No
ii8a2	i3	4.44	Yes
uf20-0235	u2	4.55	Yes
uf20-0531	u3	4.55	Yes
g2_V50_C250_P5_I1	g19	5.00	Yes
ii32e1	i2	5.34	Yes
anomaly	a6	5.44	Yes
aim-50-6_0-yes1-1	a5	6.00	Yes
g2_V50_C300_P6_I1	g20	6.00	Yes
jnh201	j1	8.00	Yes
jnh215	j3	8.00	No
medium	m1	8.22	Yes
jnh301	j2	9.00	Yes

5 Experimental results

Tables 2, 3, 4 and 5 show the results obtained for each cooling function; in these tables, the initial and final temperatures were calculated as is described in section 4. The quality of the solution was obtained from Equation 34 and the execution time was measured in seconds. The average of the execution time and quality of solution for every SAT instance tested here, is shown in the last row of each of the tables 2,3,4, and 5.

Table 2. Results for TA pure: c_i and c_f are calculated using Equations 30 and 31 respectively.

	Geometric		Logarithmic		Exponential	
Id	Q(%)	T(s)	Q(%)	T(s)	Q(%)	T(s)
a1	93.8	0.34	93.9	0.39	93.8	0.37
a10	94.6	0.01	94.5	0.01	94.6	0.01
a2	95.6	0.08	95.5	0.08	95.8	0.08
a4	93.1	0.02	93.0	0.02	92.7	0.02
a5	90.4	0.00	90.6	0.01	90.4	0.01
a6	93.4	0.80	93.1	0.89	93.2	0.79
a7	94.5	0.35	94.6	0.39	94.6	0.36
a8	96.4	0.08	96.8	0.08	96.4	0.08
b1	94.5	6.81	94.5	7.53	94.4	6.66
b2	94.1	6.30	94.1	7.19	94.2	6.42
g1	96.4	5.67	96.4	6.34	96.2	5.60
g15	90.6	32.54	90.7	35.54	90.6	33.51
g17	96.7	0.66	96.7	0.70	96.8	0.64
g19	95.0	1.09	95.0	1.17	94.9	1.06
g20	94.3	0.98	94.2	1.05	94.4	0.92
g3	93.7	4.15	93.7	4.74	93.7	4.24
i2	91.1	3.64	91.1	4.28	90.7	4.07
i3	89.1	3.18	89.0	3.45	89.5	3.23
j1	97.0	1.47	97.2	1.69	97.1	1.44
j2	96.9	2.08	96.9	2.32	96.9	2.03
j3	96.4	2.25	96.5	2.39	96.5	2.17
m1	90.0	9.93	90.3	11.37	90.3	9.78
p1	86.7	83.51	86.7	97.09	86.8	85.36
p2	93.0	5.19	92.8	5.95	93.0	5.27
p3	93.1	6.63	93.1	7.44	93.0	6.56
r1	93.7	5.08	93.7	5.74	93.6	4.81
u1	92.5	14.93	92.5	17.32	92.5	15.18
u2	98.9	0.14	99.0	0.14	99.0	0.13
u3	99.0	0.20	99.0	0.23	99.2	0.18
u4	91.9	27.95	92.0	31.08	91.8	28.11
u7	95.8	1.35	96.0	1.51	95.8	1.37
u8	93.9	5.29	93.9	6.00	93.6	5.16
u9	95.3	1.16	95.3	1.33	95.3	1.19
Avg.	94.0	7.10	94.0	8.00	94.0	7.20

As can be observed the results obtained for the quality of the solution (Equation 34) is very similar for each cooling function. In fact, the differences observed for the average of the quality of the solution are very small and always smaller to one percent.

For the execution times analysis (reported in seconds), run times averages were obtained in every one of the four tuning schemes.

Then the next observations are obtained:

- The geometric cooling function has the best execution time than both of the other two (i.e. the exponential and logarithmic cooling function).

- The logarithmic cooling function always obtains the worst execution times but again the difference is always very small (in fact smaller than one percent). Similar results were obtained when the solution quality is analyzed.

Figures 3 and 4 show the results obtained for the execution times as well as the quality solution measurements. In this figures the results were grouped by every cooling function, whereas figures 5 and 6 group the results obtained for every tuning scheme (using initial and final temperatures proposed in section 4).

Now Figures 3 and 5 are showing very important results regarding the hybrid algorithm. As can be notice it is in fact TA algorithm hybridized with SA, because the former is taken some features from the second one. Particularly, these figures show the quality solution obtained with the hybrid TA algorithm but its final temperature c_f is taken from SA model (i.e. the final temperature of SA).

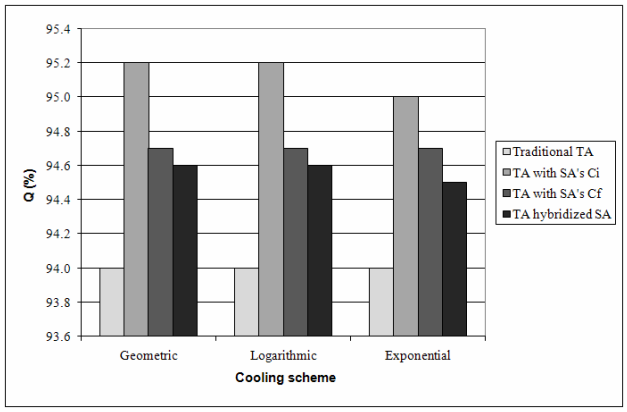

Fig. 3. Averages of quality solutions using Equation 34 grouped by each cooling function.

Table 3. Results for TA with initial temperature of SA: c_f obtained from TA pure (Equation 31) but c_i calculated as SA (using Equation 32).

	Geometric		Logarithmic		Exponential	
Id	**Q(%)**	**T(s)**	**Q(%)**	**T(s)**	**Q(%)**	**T(s)**
a1	95.0	19.86	95.3	21.13	94.5	20.13
a10	97.0	6.39	97.0	6.90	96.4	6.80
a2	97.5	6.17	97.8	6.58	97.0	6.38
a4	95.7	5.97	95.4	6.36	94.8	6.06
a5	93.8	17.77	93.9	19.84	93.1	18.84
a6	95.1	18.32	94.4	20.18	94.0	19.18
a7	95.8	20.01	95.8	21.26	95.4	20.26
a8	97.9	6.17	98.3	6.59	97.3	6.49
b1	94.8	59.64	94.7	65.97	94.8	60.25
b2	94.6	62.38	95.1	69.73	94.9	63.21
g1	96.9	44.39	96.9	47.49	97.1	44.96
g15	91.0	613.61	91.2	688.34	91.1	623.32
g17	97.3	14.46	97.6	15.36	97.9	14.66
g19	96.0	23.59	96.0	25.55	96.3	23.80
g20	95.8	26.60	95.8	30.03	95.3	26.93
g3	94.5	67.82	94.7	75.25	94.7	68.30
i2	94.2	181.40	94.1	215.29	93.9	184.13
i3	91.0	113.96	91.4	136.63	91.1	115.42
j1	97.8	101.17	97.8	112.86	97.8	101.55
j2	97.3	116.38	97.5	130.55	97.5	117.40
j3	97.0	106.70	97.2	118.02	97.1	106.63
m1	91.7	135.97	91.6	151.55	91.4	137.02
p1	87.2	638.81	87.3	717.23	87.0	650.70
p2	94.0	49.96	93.8	56.14	93.3	50.20
p3	93.5	53.33	93.6	60.25	93.6	54.54
r1	94.4	74.57	94.5	83.44	95.0	74.87
u1	93.1	200.24	93.1	223.50	93.0	202.64
u2	100.0	0.93	100.0	0.38	100.0	0.74
u3	100.0	0.43	100.0	0.55	100.0	0.95
u4	92.3	320.37	92.4	356.02	92.2	325.66
u7	96.9	22.51	97.1	24.06	96.8	22.71
u8	94.7	75.34	94.5	84.10	94.6	76.09
u9	96.2	21.60	96.1	23.47	96.2	21.96
Avg.	95.2	97.80	95.2	109.70	95.0	99.20

In Figures 3 and 5 the quality solution are reported for all the three cooling functions. In the case of Figure 3, the initial temperature is taken directly from its own model (i.e. the initial temperature for TA pure, or without hybridization.

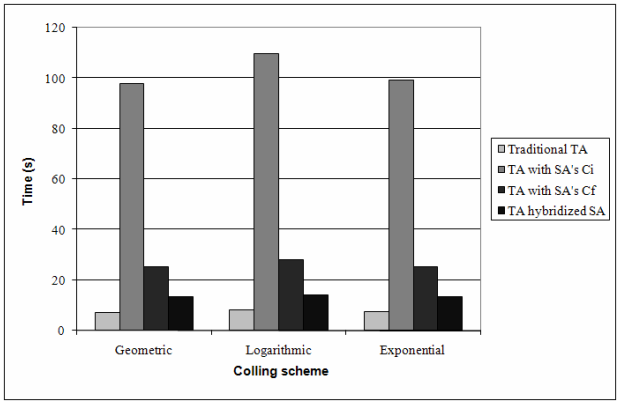

Fig. 4. Averages of execution times grouped by each cooling function..

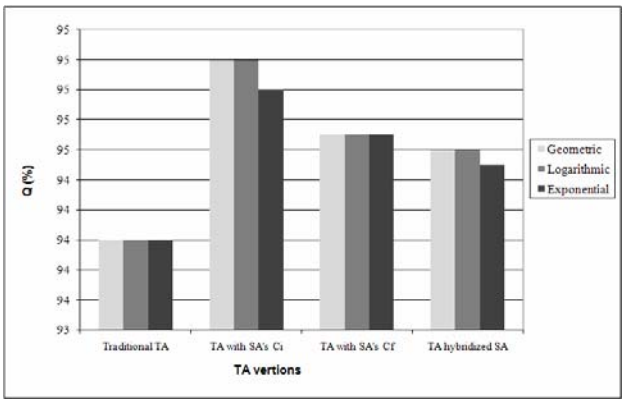

Fig. 5. Averages of quality of solutions grouped by TA algorithms versions.

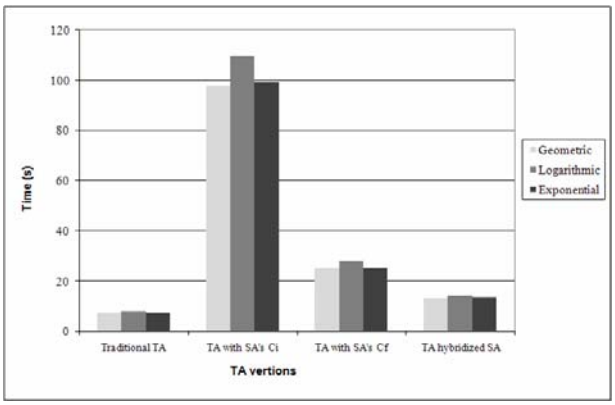

Fig. 6. Averages of execution times grouped by TA tuning schemes.

Table 4. Results for TA with final temperature of SA: c_i obtained from TA pure (Equation 30) but c_f calculated as SA (using Equation 33).

	Geometric		Logarithmic		Exponential	
Id	**Q(%)**	**T(s)**	**Q(%)**	**T(s)**	**Q(%)**	**T(s)**
a1	94.4	3.48	94.9	3.77	94.5	3.61
a10	97.3	0.84	96.6	0.87	96.4	0.83
a2	96.3	0.99	96.5	1.06	97.0	1.00
a4	95.1	1.40	94.7	1.46	94.8	1.36
a5	92.9	2.09	93.0	2.34	93.1	2.13
a6	94.4	3.96	94.0	4.33	94.0	3.97
a7	95.2	3.50	95.5	3.78	95.4	3.63
a8	97.5	0.98	97.8	1.06	97.3	1.00
b1	94.8	18.27	94.7	19.74	94.8	18.29
b2	94.5	17.86	94.4	20.17	94.6	18.29
g1	97.0	14.57	96.5	15.31	96.8	14.65
g15	91.2	143.54	90.7	160.95	90.9	144.47
g17	97.8	3.19	97.2	3.36	97.3	3.24
g19	95.3	5.21	95.5	5.56	95.6	5.33
g20	94.9	5.56	95.0	6.21	94.8	5.62
g3	94.0	16.57	94.1	18.72	94.0	16.42
i2	92.7	32.36	92.5	40.01	92.4	33.04
i3	89.9	22.25	90.9	26.13	90.3	21.99
j1	97.4	17.23	97.4	19.13	97.5	17.28
j2	97.2	20.59	97.3	23.12	97.2	20.84
j3	96.9	19.67	96.8	21.67	96.8	19.63
m1	91.2	35.27	90.9	39.32	91.1	35.86
p1	86.7	212.26	86.8	238.33	86.9	210.98
p2	93.3	14.62	93.2	16.30	93.2	14.92
p3	93.5	16.96	93.2	18.47	93.4	17.06
r1	94.5	19.05	94.1	21.17	94.1	19.02
u1	92.9	52.64	92.8	59.05	92.8	53.93
u2	99.2	0.73	99.6	0.60	99.1	0.80
u3	100.0	0.48	99.8	0.33	100.0	0.39
u4	92.2	89.71	92.1	100.17	92.2	89.03
u7	96.2	5.44	96.4	5.85	96.6	5.47
u8	94.1	19.35	94.5	21.47	94.0	19.27
u9	96.0	5.06	95.6	5.45	95.6	5.10
Avg.	94.7	25.00	94.7	28.00	94.7	25.10

Table 5. Results for TA hybridized with SA: c_i and c_f are calculated as TA (Equations 30 and 31) pure when $c=c_f$ then running SA with the last solution obtained with TA, and c_f is calculated as SA (using Equation 33).

	Geometric		Logarithmic		Exponential	
Id	**Q(%)**	**T(s)**	**Q(%)**	**T(s)**	**Q(%)**	**T(s)**
a1	94.8	2.00	94.6	2.15	94.5	2.07
a10	97.3	0.67	96.4	0.69	96.6	0.66
a2	96.6	0.61	97.3	0.63	96.3	0.61
a4	95.1	1.10	94.4	1.14	94.8	1.08
a5	93.1	1.93	93.1	2.03	93.1	1.96
a6	93.2	2.08	93.9	2.17	93.7	2.07
a7	95.6	2.01	95.1	2.16	95.1	2.07
a8	97.5	0.61	97.5	0.64	97.5	0.62
b1	94.7	9.85	94.4	10.00	94.6	9.61
b2	94.4	9.50	94.3	9.94	94.4	9.71
g1	96.4	7.83	96.4	8.04	96.4	7.72
g15	90.6	75.09	90.8	76.87	90.8	74.36
g17	96.9	1.66	97.2	1.71	97.5	1.66
g19	95.9	2.69	95.5	2.80	95.4	2.72
g20	94.4	2.98	94.6	3.06	94.8	2.95
g3	93.8	8.51	94.0	9.02	93.9	8.50
i2	91.9	18.61	93.4	21.15	92.3	18.85
i3	90.5	12.60	89.9	13.52	89.5	12.21
j1	97.4	10.45	97.5	10.73	97.6	10.37
j2	97.3	12.18	97.1	12.44	97.3	12.06
j3	96.7	11.21	96.7	11.31	96.7	11.09
m1	90.7	18.26	90.8	18.96	90.7	18.32
p1	86.6	112.64	86.7	121.91	86.8	114.77
p2	93.3	7.75	93.0	8.30	93.2	7.88
p3	93.0	9.22	92.8	9.60	93.2	9.16
r1	94.6	9.78	94.3	10.33	93.8	9.48
u1	92.7	26.72	92.6	28.98	92.7	27.39
u2	99.7	0.24	99.3	0.42	99.1	0.40
u3	99.7	0.33	99.6	0.32	99.6	0.32
u4	91.8	46.04	92.0	48.07	92.0	46.21
u7	96.3	2.78	96.0	2.93	95.9	2.77
u8	93.8	9.98	93.8	10.53	93.9	9.83
u9	95.4	2.58	95.5	2.72	95.6	2.63
Avg.	94.6	13.30	94.6	14.10	94.5	13.40

From Figures 4 and 6 can be noticed that the best execution times were obtained when the temperatures of the new algorithm were calculated directly from TA model (Equations 30 and 31). All the execution times were obtained by using tuning schemes three and fourth described in section 4. It can be noticed that all these execution times are really very good.

Figure 7 shows the averages of quality of solution for all cooling functions grouped by the tuning schemes. In Figure 7, not remarkable differences can be observed for the quality of solution for each instance. Nevertheless from Figure 8 (where the execution times are shown), remarkable differences in the values obtained for the instances: g15, p1, u1 and u4 can be observed.

Based on the fact that a good criterion for comparing random algorithms is very good quality with reasonable execution time, the algorithms presented here are listed from the best to the worst as follows:

1. TA hybridized with SA.
2. TA with final temperature of SA.
3. TA pure
4. TA with initial temperature of SA.

6 Conclusions

In this paper a new hybrid TA method based on the neighborhood structure to obtain the Markov chain length in a dynamical way is proposed.

Experiments showed that hybridizing TA with SA algorithm is an excellent option for SAT instances; it has an excellent performance based on its quality solution and its execution time. In this algorithm each SAT instance is executed with a TA algorithm but when its temperature parameter reaches the final temperature (i.e $c=c_f$,) then this temperature c_f is calculated as in TA); then once cf is reached an SA algorithm is executed with the better solution previously obtained with TA. However, the tuning method is adaptive dynamically only when the length of the Markov chain is modified in every iteration. In a future research an adaptive tuning method where the temperature be changed according with the objective function is currently been developed.

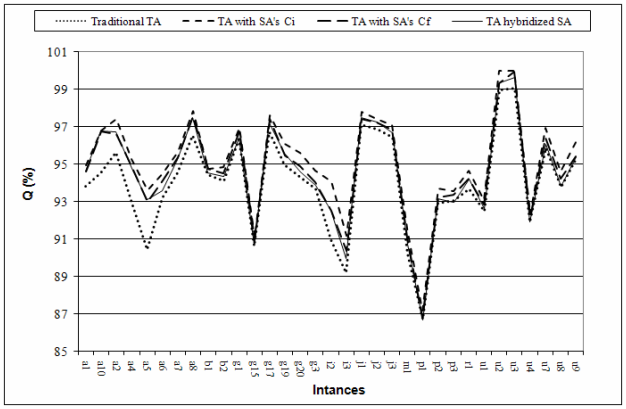

Fig. 7. Averages of quality of solutions for each SAT instance grouped by TA tuning schemes.

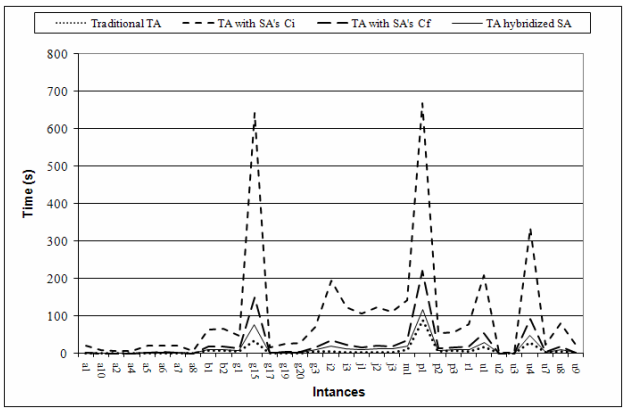

Fig. 8. Averages of execution times for each SAT instance grouped by by TA tuning schemes.

References:

[1] Cook, S.A., The complexity of theorem proving procedures. *Proceedings of 3rd Annual ACM symposium on the Theory of Computing*, ACM, (1971), pp.151–158

[2] Papadimitriou, C.H., *Computational Complexity*., Addison Wesley Longman, (1995)

[3] GU, J., Multispace search for satisfiability and np-hard problems, *DIMACS Series in Discrete Mathematics and Theoretical Computer Scienc,. Satisfiability Problem: Theory and Applications: Proceedings of a DIMACS Workshop 35,* (1996), pp. 407–517

[4] Creignou, N. The class of problems that are linearly equivalent to satisfiability or a uniform method for proving np-completeness, *Lecture Notes in Computer Science,* 702, (1993), pp.115–133

[5] Aaronson Scott, The limits of quantum computers, *Scientific American*, March, (2008), pp. 62-69.

[6] Edmons, J., Minimum partition of a matroid into independent subset. *J. Res. Nat. Bur. Standards* Sect. B, 69, pp 67-72, (1965)

[7] Frausto-Solis, J., Martinez-Rios, F.: Golden Ratio Annealing for Satisfiability Problems using Dynamically Cooling Schemes. 17th International Symposium on Methodologies for Intelligent Systems (ISMIS'08), Toronto, Canada, May 20, (2008), Accepted as a full paper for publication in ISMIS 2008 proceedings will appear in Springer's *Lecture Notes in Artificial Intelligence (LNAI)* (2008)

[8] Cook, S., Computational Complexity of Higher Type Functions. *Proc. International Congress of Mathematicians*, Kyoto, Japan, pp 51-69, Springer Verlag, (1991)

[9] Dantzig, G.B., Thapa, M.N., *Linear Programming: 1:* Introduction *(Springer Series in Operations Research and Financial Engineering)*, Springer; 1 edition, (1997)

[10] Bertzekas D. P., *Network Optimization: Continuos and Discrete Models*, Athena Scientific, (1998).

[11] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., Optimization by Simulated Annealing, *Science*, Number 4598, 13 May 1983, 220, 4598, (1983), pp. 671–680

[12] Cerny, V., Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm, *Journal of Optimization Theory and Applications,* 45, (1985), pp. 41–51

[13] Dueck, G., Scheuer, T., Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing., *Journal of Computational Physics,* (1990), pp.161–175

[14] Sanvicente-Sánchez, H., Frausto-Solís, J., Method to Establish the Cooling Scheme in Simulated Anneling Like Algorithms, *Computational Science and its Applications – ICCSA 2004*, Volume 3045, Springer Verlag, (2004)

[15] Trosset, M.W., What is Simulated Annealing?, *Optimization and Engineering*, 2, (2001), pp 201-213.

[16] Cicirello, V.A., On the Design of an Adaptive Simulated Annealing Algorithm, in First Workshop on Autonomous Search, In conjunction with CP'2007, September 23, (2007), Providence, Rhode Island, USA

[17] Aarts, E., Korst, J., *Simulated annealing and Boltzmann machines: A stochastic approach to*

combinatorial optimization and neural computing. John Wiley & Sons, Great Britain, (1989)

[18] Pérez Ortega, J., Pazos Rangel, R.A., Romero, D., Sataolaya, R., Rodríguez Ortiz, G. and Sosa, V. Adaptive and Scalable Allocation of Data-Objects in the Web, *Lecture Notes in Computer Science*, (LNCS 2667), Springer Verlag, ICCSA 2003, pp.134-143

[19] Pérez Ortega, J., Cruz, L., Landero Najera, R.V., Pazos Rangel, R., Pérez Rosas, V., Zarate Rivera, G. and Reyes Salgado, G.: Explaining Performance of the Threshold Accepting algorithm for the Bin Packing Problem, a causal approach, *Polish Journal of Environmental Studies*, Vol. 16, No. 5B, Hard, Poland, (2007), pp.72-76

[20] Ingber, L, Simulated Annealing; Practice Versus Theory, *J MATHL. Comput Modeling*, Vol 18, No. 11, 1993, pp.29-57

[21] Frausto, J., Sanvicente, H. and Imperial, F. ANDYMARK: An analytical Method to Establish Dynamically the Length of the Markov Chain in Simulated Annealing for the Satisfiablity Problem, Springer Verlag, (2006), ISSN: 0302-9743

[22] Mezard, M., Parisi, G. and Zecchina, R., Analytic and algorithmic solution of random satisfiability problems, *Science*, June 27, 2002

[23] Horie, S. and Watanabe, O., Hard instance generation for SAT, Technical Report TR97-0007, Dept. of Computer Science, Tokyo Inst. of Tech. (1997) (Available from CS Dept. TR Archive; the extended abstract appeared in Proc. ISAAC'97, *Lecture Notes in Computer Science* 1350)