A Hybrid Simulated Annealing and Threshold Accepting 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, México

[2] Tecnológico de Monterrey, Campus Cuernavaca
Paseo de la Reforma 182-A, Col. Lomas de Cuernavaca, 62584, Morelos, México

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 usually uses the same Markov chain length for each temperature cycle but they spend a lot of time. In this paper a method based on the neighborhood structure to get the Markov chain length in a dynamical way for each temperature cycle is proposed. Three cooling schemes are also presented in the paper. The experimentation presented in the paper shows that the proposed method is more efficient than the classical one.

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

1 Introduction

Satisfiability Problem (SAT) is a NP-complete problem (NP in short) 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 a NP can be transformed to a SAT instance [4,5]. Therefore, if SAT can be solved efficiently with a particular algorithm A, then A will have a similar performance for any other NP. Since the seminal papers of Simulated Annealing algorithm (SA) [6,7], this algorithm has shown to be very efficient for solving combinatorial optimization. Due to this, new algorithms based on SA have been proposed; one of them is Threshold Accepting algorithm (TA) [8] which is similar to SA except for a small modification; this is done for reducing the execution time with similar quality of the final solution.

In this paper an analytical adaptive method to establish the initial and final temperatures and the length of each Markov chain in a dynamic way for the TA algorithm is presented. Experimentation showing a set of SAT instances using three cooling schemes is also presented.

2 Classic TA algorithm

Threshold Accepting algorithm (TA) [8], is very similar to SA; it has been applied to many areas, such as Data Bases [9], Bin Packing [10] 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 bat solution but a deterministic parameter named threshold. This parameter is usually the current temperature 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 (lines 2–9) that controls the threshold value (the current temperature c) and the inner cycle (lines 3–7) which makes a stochastic walk for each temperature cycle. In step 8 a new temperature is generated using a

cooling function [11]. A new solution is always accepted (line 5) if the cost of a new solution ($Z(S_{new})$) is lower that 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: c_i, c_f, and the cooling function, which the most used are [11]:

$$T_{k+1} = \alpha T_k \text{ (geometric)} \quad (1)$$

$$T_{k+1} = \exp(-\alpha) T_k \text{ (exponential)} \quad (2)$$

$$T_{k+1} = T_k / \ln(\alpha) \text{ (logarithmic)} \quad (3)$$

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. If $Z(S_{new})$ - $Z(S_j)$ < c
6. $S_i = S_{new}$
7. Until the equilibrium
8. c = New(α,c)
9. Until (c=c_f)

Fig. 1. Pseudo–code of TA algorithm.

2.1 Initial and final temperatures

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 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 local optima is high.

On the other hand, if c_f is set too high TA probably does not explore the desired area 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 [12]. In this sense, the neighborhood structure can be defined as follow:

Definition 1: 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.

From this definition on 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 [12]:

$$\Delta Z_{V_{\max}} = Max\{Z(S_j) - Z(S_i)\} \quad \forall S_j \in V_{S_i}, \forall S_i \in S \quad (4)$$

$$\Delta Z_{V_{\min}} = Min\{Z(S_j) - Z(S_i)\} \quad \forall S_j \in V_{S_i}, \forall S_i \in S \quad (5)$$

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

$$c_i = \Delta Z_{V_{\max}} \quad (6)$$

$$c_f \le \Delta Z_{V_{\min}} \quad (7)$$

Notice that SA, produces different c_i and c_f parameters because of Boltzmann distribution [12]:

$$c_i = \frac{-\Delta Z_{V\max}}{\ln(P_A(\Delta Z_{V\max}))} \quad (8)$$

$$c_f = \frac{-\Delta Z_{V\min}}{\ln(P_A(\Delta Z_{V\min}))} \quad (9)$$

In both cases, at the beginning of the process, the probability to accept any proposed new solution is too high, but it is reduced as the temperature is decreased. Thus, the acceptance probability for very high temperatures $P_A(\Delta Z_{V\max})$ is near to 1 but for low temperatures $P_A(\Delta Z_{V\min})$ is close to zero [13].

2.2 Markov Chains Length

TA makes a stochastic walk on the solution space which can be modeled as a sequence of homogeneous Markov chains using descending values of the control parameter c > 0. Let $L_k > 0$ be the length of each Markov chain for any temperature cycle c_k (k represents the sequence index), which must satisfy:

$$\lim_{k\to\infty} c_k = 0, c_k \geq c_{k+1}, \forall k \geq 1 \quad (10)$$

c_k and L_k have a strong relation, because when $c_k \to \infty, L_k \to 0$ and when $c_k \to 0, L_k \to \infty$.

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 using the SA results for the inner loop the next tuning parameters are obtained for TA: The maximum Markov chain L_{max} occurs at the final temperature and it is:

$$L_{\max} = -Ln(P_R(S_i))\left|V_{S_i}\right| \quad (11)$$

Where $\left|V_{S_i}\right|$ is 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 $L_k = g\left(\left|V_{S_i}\right|\right)$; 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 have the same acceptance probability. Therefore, the first Markov chain length must be very small ($L_1 \approx 1$) because its stationary probabilistic state is reached in only one iteration. 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 one at c_i until L_{max} at c_f An incremental Markov chain function can be proposed as: $L_{k+1} = \beta L_k$ where $\beta > 1$ and L_k is the length of the Markov chain at c_k, L_{k+1} represents the length of the Markov chain at c_{k+1} and β is the increment coefficient.

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 $L_{\max} = \beta^n L_1$ and:

$$\beta = \exp\left(\frac{Ln(L_{\max}) - Ln(L_1)}{n}\right) \quad (12)$$

For each cooling scheme (Equations 1,2,3) the n number of steps performed from c_i to c_f can be calculated from the next formulae derived from their cooling function:

$$n = \frac{Ln(c_f) - Ln(c_i)}{Ln(\alpha)} \quad (13)$$

$$n = \frac{Ln(c_i) - Ln(c_f)}{\alpha} \quad (14)$$

$$n = \frac{Ln\left(\frac{c_i}{c_f}\right)}{Ln(Ln(\alpha))} \quad (15)$$

3 Experiment proposal

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

1. TA pure: c_i and c_f are calculated using Equation 6 and 7 respectively.
2. TA with final temperature of SA: c_i obtained by TA pure (Equation 7) but c_f calculated as SA using Equation 9.
3. TA with initial temperature of SA: c_f obtained by TA pure but (Equation 6) c_i calculated as SA, using Equation 8.

4. TA hybridized with SA: c_i and c_f are calculated as TA (Equation 6 and 7) pure when $c=c_f$ then running SA with the last solution obtained by TA, and c_f is calculated as SA using Equation 7.

Table 1 shows several SAT instances with different σ relations of clauses/variables [14]; they were taken from SATLIB or generated with Hories algorithm [15]. 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 the average execution time and the average quality solution were obtained. The quality solution was calculated by:

$$Q = \frac{clauses \quad true}{total \quad clauses} \times 100 \qquad (16)$$

The Alpha's value used for each cooling scheme (Equations 1,2,3) was obtained experimentally as 0.99, 0.01 and 2.745 respectively

Table 1. SAT instances tested.

Sat Instance	σ	Sat Instance	σ
aim-50-1_6-yes1-3	1.60	par8-5-c	3.97
aim-50-1_6-no-2	1.60	G2_V100_C400_P4_I1	4.00
aim-100-1_6-yes1-1	1.60	hole8	4.13
aim-200-1_6-no-1	1.60	uuf225-045	4.27
aim-50-2_0-no-4	2.00	RTI_k3_n100_m429_150	4.29
aim-50-2_0-yes1-1	2.00	uuf100-0789	4.30
aim-50-2_0-no-3	2.00	Uf175-023	4.30
G2_V100_C200_P2_I1	2.00	Uf50-01	4.36
dubois21	2.67	uuf50-01	4.36
dubois26	2.67	Ii8a2	4.44
dubois27	2.67	G2_V50_C250_P5_I1	5.00
BMS_k3_n100_m429_161	2.83	hole10	5.10
G2_V50_C150_P3_I1	3.00	Ii32e1	5.34
G2_V300_C900_P3_I1	3.00	Anomaly	5.44
BMS_k3_n100_m429_368	3.08	aim-50-6_0-yes1-1	6.00
hole6	3.17	G2_V50_C300_P6_I1	6.00
par8-1	3.28	Jnh201	8.00
aim-50-3_4-yes1-2	3.40	Jnh215	8.00
hole7	3.64	Medium	8.22
par8-3-c	3.97	Jnh301	9.00

4 Experiment results

In Figure 2 we can observe the quality solution Q obtained by using Equation 16; to obtain these results the quality solution was calculated for each instance and the average was obtained for each cooling scheme.

For each cooling scheme its correspondent number of steps in the Metropolis cycle was calculated; the initial and final temperature were calculated with the model presented in section 3.
As we can notice in Figure 2 the best results for the quality solution measure were obtained with the hybrid algorithm which the final temperature used is just c_f of SA.

Figure 3 shows that the better execution time is obtained when the temperatures were calculated with TA equations (Equations 6 and 7). The execution times obtained by using schemes 3 and 4 described in section 3 are also very good.

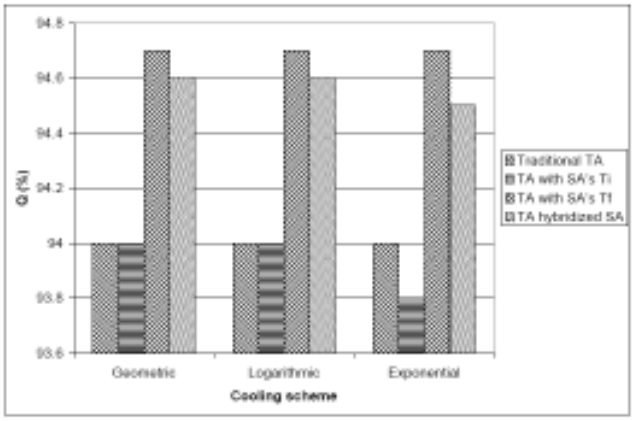

Fig. 2. Quality solutions using Equation 16.

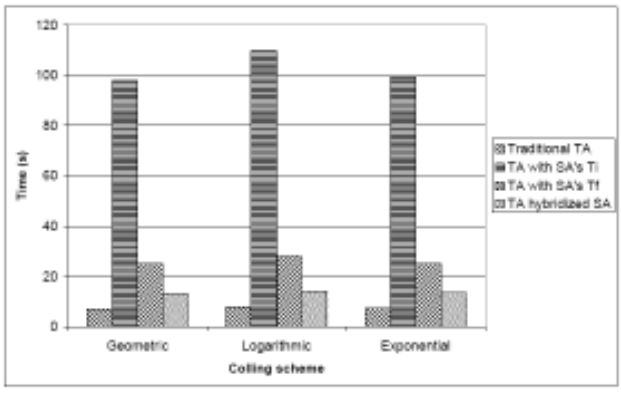

Fig. 3. Execution times.

Based on the fact that a good criterion for comparing random algorithms is very good quality with reasonable 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.

5 Conclusion

In this paper a method based on the neighborhood structure to get the Markov chain length in a dynamical way for each temperature cycle is proposed.

Experiments showed that TA hybridized with SA has 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 $c=c_f$ (c_f calculated for TA) then is executed a SA algorithm with the better solution obtained in TA.

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

[5] 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

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

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

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

[9] 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

[10] 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

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

[12] 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)

[13] 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

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

[15] 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)