Analysis of Random Noise and Random Walk Algorithms for Satisfiability Testing

Bhaskar Krishnamachari[1], Xi Xie[1], Bart Selman[2], and Stephen Wicker[1]

[1] School of Electrical Engineering
Cornell University, Ithaca, NY 14853
{bhaskar,xie,wicker}@ee.cornell.edu
[2] Department of Computer Science
Cornell University, Ithaca, NY 14853
selman@cs.cornell.edu

Abstract. Random Noise and Random Walk algorithms are local search strategies that have been used for the problem of satisfiability testing (SAT). We present a Markov-chain based analysis of the performance of these algorithms. The performance measures we consider are the probability of finding a satisfying assignment and the distribution of the best solution observed on a given SAT instance. The analysis provides exact statistics, but is restricted to small problems as it requires the storage and use of knowledge about the entire search space. We examine the effect of p, the probability of making non-greedy moves, on these algorithms and provide a justification for the practice of choosing this value empirically.

1 Introduction

Local search algorithms such as GSAT, Random Walk and Random Noise search have been shown to be good at solving CNF satisfiability (SAT) problems [4,13]. Such methods perform better than systematic search algorithms on large satisfiability problems involving thousands of variables. They may be used for the problem of maximum satisfiability (finding a truth assignment that satisfies as many clauses as possible) as well as complete satisfiability (satisfying all clauses).

However, due to the complex interactions between the problem instance and algorithm implementation details, it is hard to predict the performance of these algorithms. Researchers have, therefore, mainly relied upon empirical studies for this purpose [5,14]. Although this approach provides very useful results, it is still desirable to have some theoretical understanding of algorithm performance.

A large portion of the literature on theoretical analysis of local search algorithms for other problems has been devoted to determining the convergence of search algorithms to the global optimum using Markov models [2,3,7,8,9,11]. The rates of convergence to the optimum have also been discussed assuming various properties of cost functions and search spaces [15,16]. Some work in the area of complexity theory has been focused on studying PLS (polynomial local search) problems regarding the time required to locate local optima [6,10].

R. Dechter (Ed.): CP 2000, LNCS 1894, pp. 278–290, 2000.

In this paper, we show how the Random Walk and Random Noise algorithms can be modeled using discrete Markov chains. We present a procedure to determine the probability of finding the global optimum as well as complete statistics of the best solution observed in a given number of iterations. The former measure of algorithm performance is most relevant to maximum satisfiability problems, while the latter is the statistic of interest when considering complete satisfiability. These measures are relevant because they tell us quantitatively how the algorithm will perform on a given problem in limited computational time. This in turn will help us to determine the best parameters for these search algorithms to use.

The procedure presented requires the storage and use of complete knowledge about the search space. Hence, it can only be carried out for small-scale satisfiability problems. Still, this analysis provides some insights regarding the performance of these algorithms on real world problems. Real world problems are characterized by the existence of local minima which hinder the performance of greedy local search. Both Random Noise and Random Walk algorithms provide ways of escaping local minima, using the parameter p, the probability of making random non-greedy moves. The value of p that provides optimum algorithm performance is of great interest. We find theoretical support for the practice of empirically choosing an optimal value for this parameter.

The rest of the paper is organized as follows: section 2 reviews the definitions of the Random Noise and Random Walk algorithms. Section 3 shows how these algorithms can be modeled as discrete Markov chains and presents a procedure for determining the performance statistics for these algorithms. The method of determining the one-step state transition matrix for these algorithms is described in section 4. Section 5 presents and discusses results obtained using this procedure. Concluding comments are presented in section 6.

2 Random Noise and Random Walk Algorithms for Satisfiability

The Random Noise and Random Walk algorithms are both based on GSAT, a greedy local search procedure for satisfiability which works as follows [12]:

Procedure GSAT

```
for i:= 1 to MAX-TRIES
    T:= a randomly generated truth assignment
    for j := 1 to MAX-FLIPS
        if T satisfies expression then return T
        Flip any variable in T that results in greatest decrease
            (could even be 0) in the number of unsatisfied clauses
    end for
end for
return "No satisfying assignment found"
```

The success of GSAT depends on its ability to make either strictly improving or "sideways" moves (moves to assignments with an equal number of unsatisfied clauses). When the algorithm finds itself at a non-optimal point in the search space where no further improvements are possible, it is essentially trapped in a region which is a local minimum and needs to be restarted with a random new assignment. Another mechanism for escaping such local minima that is widely used is to permit the search to make uphill moves occasionally. Random Noise and Random Walk algorithm are both closely related in the way they allow for the possibility of uphill moves [13]:

Random Noise

```
With probability p, pick any variable at random
    and flip its truth assignment.
With probability 1-p, follow the standard GSAT scheme,
        i.e., make the best possible local move
```

Random Walk

```
With probability p, pick a variable occurring in some
        unsatisfied clause and flip its truth assignment.
With probability 1-p, follow the standard GSAT scheme,
        i.e., make the best possible local move
```

Experimental results comparing the basic GSAT algorithm, Simulated Annealing, Random Walk and Random Noise strategies on a test suite including randomly-generated CNF problems and Boolean encodings of circuit synthesis, planning and circuit diagnosis problems can be found in [13]. The authors of this paper found that the Random Walk strategy significantly out-performed the other algorithms on these problems.

3 Modeling and Analysis

If we look at the search process as a sequence of decisions for moving from point to point in the search space, most local search algorithms can be called "memoryless" in the sense that the process of selecting the next point depends iteratively only on the current point. Therefore, the search process is a Markov process with finite *states* (e.g. the points in the search space). Furthermore, the search algorithms are performed at discrete steps/iterations and this allows us to model them as Markov chains. Such models for two widely used local search algorithms – Simulated Annealing and Genetic Algorithms, can be found in [1] and [9] respectively. In the context of satisfiability problems, each point in the search space corresponds to a unique truth assignment.

Theorem 1: The sequence of points visited by the Random Noise (also Random Walk) algorithm forms a Homogeneous Markov Chain.

Proof: To prove that this sequence forms a Markov chain, it suffices to show that the point visited at the $(k+1)^{st}$ iteration depends only upon which point was visited at the k^{th} iteration. This can be seen as follows: by the definition of both these algorithms, the truth assignment at the $(k+1)^{st}$ iteration differs from the truth assignment at the k^{th} iteration at exactly one variable. The set of variables that may be flipped at the k^{th} iteration depends only upon whether we are considering the Random Noise or Random Walk algorithm and not the points visited at any previous iteration. Finally, the probability of flipping each variable in this set also depends only on the value of p and not the points visited in the first $(k-1)$ iterations. The Markov chain will be homogeneous because the state transition probabilities will only be a function of p which is assumed to be constant for the duration of the search. **Q.E.D.**

In a SAT problem, if N is the number of variables, the search space X consists of a total of $|X| = 2^N$ possible truth assignments. Let $x^{[j]}$, $1 \leq j \leq |X|$ be a point in the search space. The cost $f(x^{[j]})$ is the number of unsatisfied clauses in the corresponding truth assignment. For simplicity of analysis and description, we assume that the points in the search space are sorted in non-decreasing order of costs, i.e. $j < k \Rightarrow f(x^{[j]}) \leq f(x^{[k]})$. The search space may contain two points with the exact same cost function value. We represent the sorted list of costs using a row vector $\overrightarrow{f}$ of size $|X|$ such that the j^{th} element $\overrightarrow{f}_{[j]} = f(x^{[j]})$.

Let x_i be the random variable describing which point the search is at during iteration i. The *probability mass function* (pmf) of x_i is represented by a row vector $\overrightarrow{\pi}_i$ of size $|X|$ such that the m^{th} element $\overrightarrow{\pi}_{i,[m]} = P\{x_i = x^{[m]}\}$. A homogeneous Markov chain based local search algorithm can then be described by a one-step state transition matrix $\mathbf{P}$ such that:

$$\overrightarrow{\pi}_i = \overrightarrow{\pi}_{i-1}\mathbf{P} \tag{1}$$

The performance statistics of interest (probability of finding the global optimum within a given number of iterations, best solution observed to date) require us to incorporate the search history as well. For this purpose, it is necessary to fully describe the state that the search algorithm is in at a given iteration. This description should include a) the pmf describing the probability that the search algorithm is at any given point in the search space, and b) the conditional pmf's of the best (lowest) cost seen up to the current iteration given that the search is currently at a certain point in the search space. Both these pmf's can be iteratively calculated at each step as they depend on only the pmf's of the previous iteration and on the search algorithm being analyzed.

Let x_i^* denote the point with the lowest cost function seen up to iteration i. We can use a matrix $\mathbf{D_i^*}$ to represent the conditional probability of the lowest cost seen to date given the current search point, i.e. $D^*_{i,[jk]} = P\{x_i^* = j | x_i = k\}$. Note that $\mathbf{D_0^*} = \mathbf{I}$. For entries representing equal value points, it does not matter how the weight is distributed among them as long as the total probability remains the same.

For the i^{th} iteration, the distribution $\overrightarrow{\pi}_i$ can be calculated from $\overrightarrow{\pi}_{i-1}$ using equation (1). The following formulae[1] can be used in sequence to calculate $\mathbf{D_i^*}$ from $\overrightarrow{\pi}_{i-1}$ and $\mathbf{D_{i-1}^*}$:

$$\mathbf{B_i^*} = \mathbf{D_{i-1}^*}\mathbf{diag}(\overrightarrow{\pi}_{i-1})\mathbf{P} \tag{2}$$

$$C^*_{i,[jk]} = \begin{cases} B^*_{i,[jk]} & j < k \\ \sum_{l=j}^{|X|} B^*_{i,[lk]} & j = k \\ 0 & j > k \end{cases} \tag{3}$$

$\mathbf{B_i^*}$ and $\mathbf{C_i^*}$ are temporary matrices. The best-to-date point cannot be worse than the current point at any time. An entry $B^*_{i,[jk]}$ in $\mathbf{B_i^*}$ represents the probability of having j as the best observed point and k as the current search point without this consideration. $\mathbf{C_i^*}$ contains the corresponding probabilities after considering this fact. Equation (4) normalizes $\mathbf{C_i^*}$ to the desired conditional pmf matrix [2]:

$$\mathbf{D_i^*} = \mathbf{C_i^*}(\mathbf{diag}(\overrightarrow{\pi}_i))^{-1} \tag{4}$$

Thus, given the initial state distribution $\overrightarrow{\pi}_0$ and the state transition matrix $\mathbf{P}$, we can derive $\mathbf{D_n^*}$ – the conditional pmf's of the lowest cost function value seen to date, and $\overrightarrow{\pi}_n$ – the distribution of costs at the n^{th} iteration. It is typically assumed that each point in the search space is equally likely to be picked as the starting point (uniform initial distribution). Once $\mathbf{D_n^*}$ and $\overrightarrow{\pi}_n$ are known, the expectation and variance of the best-to-date cost can then be readily calculated by definition:

$$E[f(x_n^*)] = \overrightarrow{f}\mathbf{D_n^*}\overrightarrow{\pi}_n^T \tag{5}$$

$$VAR[f(x_n^*)] = \overrightarrow{f}\,\mathbf{diag}(\overrightarrow{f})\mathbf{D_n^*}\overrightarrow{\pi}_n^T - (\overrightarrow{f}\mathbf{D_n^*}\overrightarrow{\pi}_n^T)^2 \tag{6}$$

The expectation and variance of best-to-date cost are useful measures if we are interested in the problem of maximum satisfiability. For complete satisfiability, we would like to know the probability $P[f(x_n^*) = f^*]$ of achieving the global optimum f^* within n iterations. This can be calculated as follows:

$$P[f(x_n^*) = f^*] = \overrightarrow{e}\mathbf{D_n^*}\overrightarrow{\pi}_n^T \tag{7}$$

where $\overrightarrow{e} = [1\ 0\ 0\ 0\ \ldots\ 0]$, consisting of a 1 followed by $(|X| - 1)$ zeros.

We note here that the above procedure for calculating these statistics up to iteration n, as outlined in equations (1) through (7), has a computational

[1] In these formulae, $\mathbf{diag}(\overrightarrow{v})$ represents the diagonal matrix derived from a vector $\overrightarrow{v}$; $\mathbf{B_i^*}$ and $\mathbf{C_i^*}$ are temporary matrices used during this updating process.

[2] Rigorously, the inverse of $\mathbf{diag}(\overrightarrow{\pi}_i))$ does not exist if any of the elements of $\overrightarrow{\pi}_i$ are 0 – although this only happens when using purely greedy search. However the notation used in equation 4 is convenient and the difficulty can be overcome by treating these 0 elements as arbitrarily small values ϵ.

complexity of $O(|X|^3 n)$, where $|X| = 2^N$ is the size of the search space. The exponential dependence on the number of variables renders this exact analysis infeasible for larger problems.

4 Determining the State Transition Matrix

Table 1. Sample 3-SAT instance with 3 variables, 15 clauses

$\{3, 1, -2\}$	$\{-1, 2, -3\}$	$\{-3, 2, 1\}$	$\{2, 1, -3\}$	$\{1, 2, 3\}$
$\{3, -2, -1\}$	$\{-2, 1, -3\}$	$\{1, -3, -2\}$	$\{3, 1, 2\}$	$\{1, 3, 2\}$
$\{-3, 2, -1\}$	$\{1, 3, 2\}$	$\{3, 1, 2\}$	$\{1, 3, -2\}$	$\{1, -2, 3\}$

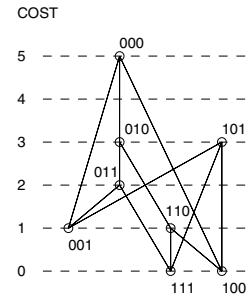

Fig. 1. Neighborhood definition and cost values for a randomly generated 3SAT instance with 3 variables and 15 clauses; 111 and 100 are satisfying assignments

We have shown how the performance statistics of interest may be obtained once the state transition matrix $\mathbf{P}$ is known. This matrix depends upon both the specific problem instance, as well as the algorithm used. We discuss here via an example how the matrix can be obtained for Random Noise algorithms if the satisfiability instance is known.

Table 1 shows a randomly generated 3-SAT instance with 3 variables and 15 clauses[3]. The cost for each truth assignment, i.e. the number of unsatisfied clauses as well as the neighboring truth assignments are shown in figure 1. From the figure, it is easy to see that the assignment 001 is a local minimum and that the global minima 111 and 100 are satisfying assignments.

Given a problem instance, the one-step transition matrix $\mathbf{P}$ can be determined for the Random Noise algorithm as follows:

[3] It may be seen that a number of these clauses are identical in this example, but this is to due to the small number of variables used for illustration.

– Determine the transition matrix $\mathbf{P_{greedy}}$ for the GSAT algorithm ($p = 0$).
– Determine the transition matrix $\mathbf{P_{random}}$ for the random noise algorithm with $p = 1$.
– $\mathbf{P} = (1 - p)\ \mathbf{P_{greedy}} + p\ \mathbf{P_{random}}$

For the SAT instance presented in figure 1, the corresponding transition matrices for the Random Noise algorithm: $\mathbf{P_{greedy}}$ and $\mathbf{P_{random}}$ are shown in figure 2. The $\mathbf{P_{random}}$ matrix is constructed by assigning equal transition probabilities to each neighbor of a given truth assignment (elements corresponding to non-Neighboring points are 0). The $\mathbf{P_{greedy}}$ is constructed by assigning equal transition probabilities from any given truth assignment to the neighbor(s) which have the greatest decrease in cost (0 or more). The procedure is nearly identical for obtaining the $\mathbf{P}$ for the Random Walk algorithm, with the only difference being in the construction of $\mathbf{P_{random}}$. To construct $\mathbf{P_{random}}$ for the Random Walk algorithm, assign equal state transition probabilities to each neighbor of a given truth assignment that can be obtained by flipping a variable involved in unsatisfied clauses.

$$P_{random} = \frac{1}{3}\begin{bmatrix} 0&0&0&1&0&0&1&1\\ 0&0&0&1&1&0&1&0\\ 0&0&0&0&1&0&1&1\\ 1&1&0&0&0&1&0&0\\ 0&1&1&0&0&1&0&0\\ 0&0&0&1&1&0&0&1\\ 1&1&1&0&0&0&0&0\\ 1&0&1&0&0&1&0&0 \end{bmatrix} \qquad P_{greedy} = \frac{1}{2}\begin{bmatrix} 2&0&0&0&0&0&0&0\\ 0&2&0&0&0&0&0&0\\ 0&0&2&0&0&0&0&0\\ 1&1&0&0&0&0&0&0\\ 0&2&0&0&0&0&0&0\\ 0&0&0&2&0&0&0&0\\ 1&1&0&0&0&0&0&0\\ 2&0&0&0&0&0&0&0 \end{bmatrix}$$

Fig. 2. Constructing State Transition Probability Matrices

5 Results

Using the method of analysis presented in this paper, it is possible to investigate the effect of p, the non-greedy move probability, on the performance of random noise algorithms. One significant result is the following:

Theorem 2: $E[f(x_n^*)]$, the expected best cost seen by a random noise or random walk algorithm after n iterations, is a polynomial in p of order at most n.

Proof: See appendix A.

Corollary 1: The variance of the best cost $VAR[f(x_n^*)]$ is a polynomial in p of order at most $2n$.

Corollary 2: $P[f(x_n^*) = f*]$, the probability of having found the global best assignment after n iterations, is a polynomial in p of order at most n.

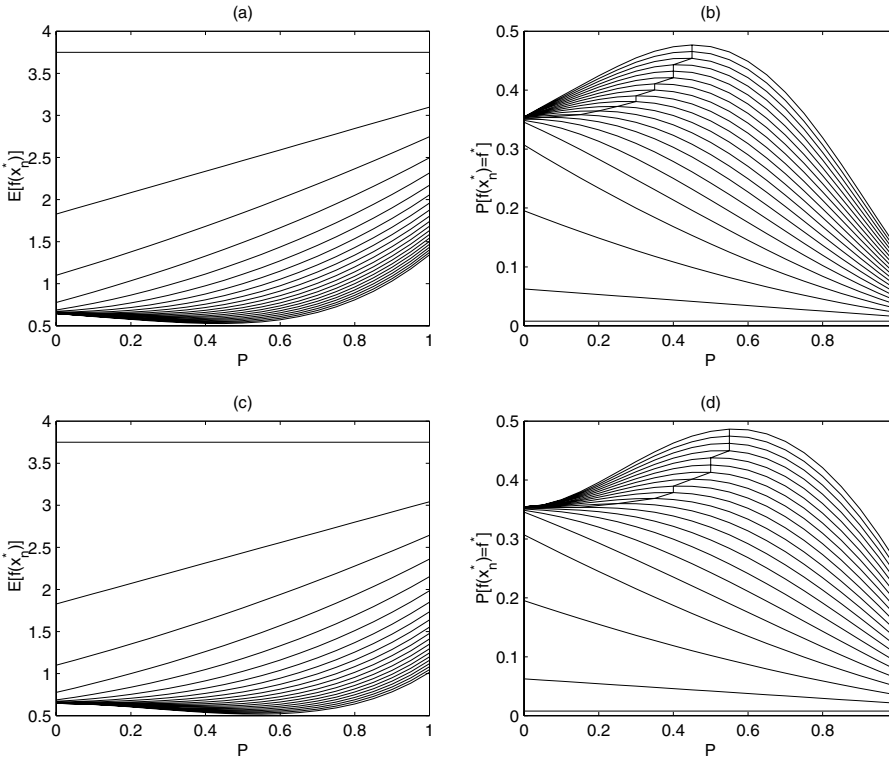

Fig. 3. Performance for Random Noise and Random Walk algorithms with respect to p for a randomly generated 3-SAT instance with 7 variables. Figures (a) and (c) show the curves for expectation of best-to-date costs for Random Noise and Random Walk algorithms respectively. Figures (b) and (d) show the curves for the probability of finding a satisfiable solution for Random Noise and Random Walk algorithms respectively. In (b) and (d), the locus connecting the peaks of the curves of the various iterations indicate the value of p for which this probability is maximized. For the expectation figures (a) and (c), each successively lower curve represents an increasing iteration number, while the opposite is true for the probability figures (b) and (d)

We applied the procedure described in equations (1) through (7) on randomly generated 3-SAT instances to determine the performance of Random Walk and Random Noise algorithms. Table 2, in appendix B, shows a typical instance with 7 variables and 30 clauses. Figure 3 shows the effect of p on Random Noise (3a,3b) and Random Walk (3c,3d) algorithms for this instance.

The data in figure 3 is for 21 values of p ranging from 0 to 1, with 0.05 increment, for the first 20 iterations of the algorithms. Figures 3a and 3c show the expected best cost. The first line on top corresponds to iteration 0, the starting point of the search. As the iteration number increases, the expected

cost goes down at each step and is indicated by the successively lower curves. Figure 3b and 3d show the probability of having found the global optimum (in this case, a satisfying assignment with 0 cost). This probability increases with iteration and is hence represented by successively higher curves. In all these graphs, for any given iteration, there is some $p = p_{best}$ for which the algorithm achieves the best value. In figures 3b and 3d, the p_{best}points for each iteration (subject to the resolution of the p values tested) are connected, forming a locus.

The performance statistics for the two algorithms are different, and this can be seen more clearly in figure 4, where the probability of having found the global minimum for both algorithms is compared for $p = 0, 0.5, and\ 1$. When $p = 0$, as noted earlier, both algorithms are identical to the GSAT algorithm and hence their performance is the same. For the other two values of p, it is seen that Random Walk algorithm out-performs the Random Noise algorithm. We have noticed this on other instances as well. This has also been observed empirically on large-scale problems [13,14].

From figure 3, especially from the loci in figure 3b and 3d, we can see that p_{best} can be different for each iteration. In practice, this implies that different p value might be needed depending on how many iterations the search algorithm is to run. However, as the iteration number increases, the change in p_{best} gets smaller.

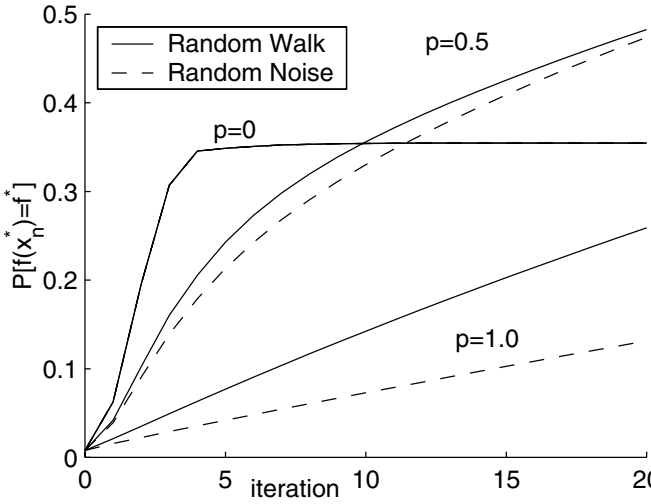

Fig. 4. Comparison of Random Noise and Random Walk algorithms

6 Conclusion

Local search algorithms can be modeled as Markov chains. We have shown how the Random Noise and Random Walk algorithms for satisfiability testing can

also be described using such a model. Based on this modeling, we are able to derive the probability of finding a satisfying assignment, and the statistics of the best solution observed within a given number of iterations. The former measure of algorithm performance is useful when considering problems of complete satisfiability, while the latter is more relevant to the problem of maximum satisfiability and related optimization problems.

For real world problems, it is almost always the case that a value of $p = p_{best} \in (0, 1)$ offers the best performance on a given problem. The results obtained for randomly generated 3-SAT instances using our analysis also show this behavior. The value of p_{best} depends upon the specific problem instance, as well as the iteration number. We observed that the performance measures vary slowly with respect to p. Further, we have proved that these performance measures are polynomial (hence continuous) functions of p, the probability of making non-greedy moves. Therefore, for real world problems, if the value of p chosen via experiments is close to p_{best}, it will result in near-optimal performance. In nearly all the 3-SAT instances we tested, the Random Walk algorithm out-performed the Random Noise algorithm. This merits further study.

The characteristics of a search space have a big impact on the algorithm performance. Only a limited number of SAT problem instances are tested in the experiments. Future research may include the study of effects of changes in parameters such as the ratio of constraints to variables. This may reveal more insights on how the structure of a problem and the search algorithms interact.

References

1. E. H. L. Aarts and J. H. M. Korst, *Simulated Annealing and Boltzmann Machines*, Wiley, 1989. 280
2. J. R. Cruz and C. C. Y. Dorea, "Simple conditions for the convergence of simulated annealing type algorithms," *Journal of Applied Probability*, vol. 35, no. 4, p. 885-92, December 1998. 278
3. A. E. Eiben, E. H. L. Aarts, and K. M. Van Hee, "Global convergence of genetic algorithms: a markov chain analysis," *Parallel Problem Solving from Nature, PPSN 1*, p. 4-12, October 1990. 278
4. J. Gu, "Efficient Local Search for Very Large Scale Satisfiability Problems," *Sigart Bulletin*, vol. 3, no. 1, p. 8-12, 1992. 278
5. J. Hansen and B. Jaumard, "Algorithms for the maximum satisfiability problem," *Computing*, vol. 44, pp. 279-303, 1990. 278
6. D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis, "How easy is local search?," *Journal of Computer and System Sciences*, vol. 37, no. 1, p. 79-100, August 1998. 278
7. C. Y. Mao and Y. H. Hu, "Analysis of Convergence Properties of a Stochastic Evolution Algorithm," *IEEE Transactions on Computer-Aided Design of integrated circuits and systems*, vol. 15, no. 7, July 1996. 278
8. D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli, "Convergence and finite-time behavior of simulated annealing," *Proceedings of the 24th IEEE Conference on Decision and Control*, vol. 2, p. 761-7, December 1985. 278

9. A. E. Nix and M. D. Vose, "Modeling genetic algorithms with Markov Chains," *Annals of Mathematics and Artificial Intelligence*, vol. 5, no. 1, p. 79-88. 278, 280
10. C. H. Papadimitriou, A. A. Schaffer, M. Yannakis, "On the complexity of local search," *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, p. 438-45, May 1990. 278
11. G. Rudolph, "Convergence Analysis of Canonical Genetic Algorithms," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, January 1994. 278
12. B. Selman, H. J. Levesque and D. G. Mitchell, "A new Method for Solving Hard Satisfiability Problems, *Proceedings AAAI-92*, San Jose, CA, pp. 440-446, 1992. 279
13. B. Selman, H. A. Kautz, and B. Cohen, "Local Search Strategies for Satisfiability Testing," *Second DIMACS Challenge on Cliques, Coloring, and Satisfiability*, October 1993. 278, 280, 286
14. B. Selman, H. A. Kautz, and B. Cohen, "Noise Strategies for Improving Local Search," *Proceedings of Twelfth National Conference on Artificial Intelligence, AAAI-94*, vol.1, pp. 337-43, July 1994. 278, 286
15. G. B. Sorkin, "Efficient simulated annealing on fractal energy landscapes," *Algorithmica*, vol. 6, no. 3, p. 367-418, 1991. 278
16. J. C. Spall, S. D. Hill, D. R. Stark, "Theoretical Comparisons of Evolutionary Computation and Other Optimization Approaches," *Congress on Evolutionary Computation*, vol. 2, p. 1398-405, July 1999. 278

Appendix A

Remark – Properties of the Transition Matrix: Since the sum of any row of the transition matrix $\mathbf{P}$ is 1, $\mathbf{P}\overrightarrow{1}^T = \overrightarrow{1}^T$, where $\overrightarrow{1}$ is a 1×2^N vector of 1's. The elements of $\mathbf{P}$ are polynomials in p of degree either 0 or 1.

Definition: If a matrix $\mathbf{A}$ has elements that are polynomials in p, $\overline{O}(\mathbf{A})$ is defined as the highest possible degree of these polynomials.

Theorem 2: For a given neighborhood, $E[f(x_n^*)]$, the expected best cost seen by a random noise algorithm after n iterations, is a polynomial in p of order at most n.

Proof: By substituting equation (4) into (5), we have:

$$E[f(x_i^*)] = \overrightarrow{f}\,\mathbf{C_i^*}(\mathbf{diag}(\overrightarrow{\pi}_i))^{-1}\overrightarrow{\pi}_i^T = \overrightarrow{f}\,\mathbf{C_i^*}\overrightarrow{1}^T \tag{8}$$

If the elements of $\mathbf{C_i^*}$ be polynomials in p, then $E[f(x_i^*)]$ is also a polynomial in p, and

$$\overline{O}(E[f(x_i^*)]) = \overline{O}(\mathbf{C_i^*}\overrightarrow{1}^T) \tag{9}$$

Hence it suffices to show that the elements of $\mathbf{C_i^*}$ are polynomials in p, and that $\overline{O}(\mathbf{C_i^*}) = i$. This can be done inductively:

Base Case (n = 1)

From the fact that $\mathbf{D_0^*} = \mathbf{I}$ and equation (2), we get $\mathbf{B_1^*} = \mathbf{diag}(\overrightarrow{\pi}_0)\mathbf{P}$. Hence the elements of $\mathbf{B_1^*}$ are polynomials in p and $\overline{O}(\mathbf{B_1^*}) = 1$. All the rows of $\mathbf{B_1^*}$ add up to one:

$$\overline{O}(\mathbf{B_1^*}\overrightarrow{1}^T) = \overline{O}(\mathbf{diag}(\overrightarrow{\pi}_0)\mathbf{P}\overrightarrow{1}^T) = \overline{O}(\mathbf{diag}(\overrightarrow{\pi}_0)\overrightarrow{1}^T) = 0 \tag{10}$$

The operations in equation (3) consist of adding all elements of $\mathbf{B_1^*}$ that are below the diagonal to the diagonal element in each column and then setting these below-diagonal elements to 0. Thus $\mathbf{C_1^*}$ is an upper-triangular matrix such that all its elements are also polynomials in p and $\overline{O}(\mathbf{C_1^*}) = \overline{O}(\mathbf{B_1^*}) = 1$. When each row of $\mathbf{C_1^*}$ is summed, the order 1 terms will not necessarily cancel out, and hence $\overline{O}(\mathbf{C_1^*}\,\overrightarrow{1}^T) = 1$.

Inductive Hypothesis

For any $k > 1$, the elements of $\mathbf{B_{k-1}^*}$ and $\mathbf{C_{k-1}^*}$ are polynomials in p. Further, $\overline{O}(\mathbf{C_{k-1}^*}) = \overline{O}(\mathbf{B_{k-1}^*}) = k-1$, $\overline{O}(\mathbf{B_{k-1}^*}\,\overrightarrow{1}^T) = k-2$, and $\overline{O}(\mathbf{C_{k-1}^*}\,\overrightarrow{1}^T) = k-1$.

Inductive Step

This is similar to the verification of the base case. From equations (2) and (4), we get:

$$\mathbf{B_i^*} = \mathbf{C_{i-1}^*}(\mathbf{diag}(\overrightarrow{\pi}_{i-1}))^{-1}\mathbf{diag}(\overrightarrow{\pi}_{i-1})\mathbf{P} = \mathbf{C_{i-1}^*P} \tag{11}$$

By this equation, the elements of $\mathbf{B_k^*}$ are polynomial in p, and $\overline{O}(\mathbf{B_k^*}) = \overline{O}(\mathbf{C_{k-1}^*}) + \overline{O}(\mathbf{P}) = k$. The operations in equation (3) ensure that $\mathbf{C_k^*}$ is an upper-triangular matrix and that all its elements are polynomials in p with $\overline{O}(\mathbf{C_k^*}) = \overline{O}(\mathbf{B_k^*}) = k$. Also from equation (11),

$$\overline{O}(\mathbf{B_k^*}\,\overrightarrow{1}^T) = \overline{O}(\mathbf{C_{k-1}^*P}\,\overrightarrow{1}^T) = \overline{O}(\mathbf{C_{k-1}^*}\,\overrightarrow{1}^T) = k-1 \tag{12}$$

This means that when each row of $\mathbf{B_k^*}$ is summed, any terms of degree k all cancel out. After the below-diagonal elements of $\mathbf{B_k^*}$ are moved to the diagonal terms in equation 3, when the rows of $\mathbf{C_k^*}$ are summed, the terms of degree k will not necessarily cancel. Hence $\overline{O}(\mathbf{C_k^*}\,\overrightarrow{1}^T) = k$.

Therefore by induction, we have that the elements of $\mathbf{C_n^*}$ are polynomials in p and that $\overline{O}(\mathbf{C_n^*}\,\overrightarrow{1}^T) = n,\ \forall n \geq 1$. **Q. E. D.**

Corollary 1: The variance of the best cost $VAR[f(x_n^*)]$ is a polynomial in p of order at most $2n$.

This follows immediately from the result of Theorem 1 and equation 6.

Corollary 2: $P[f(x_n^*) = f^*]$, the probability of having found the global best assignment after n iterations, is a polynomial in p of order at most n.

To see this, compare equations (5) and (7). The properties of $E[f(x_n^*)]$ with respect to p hold for $P[f(x_n^*) = f^*]$ as well.

Appendix B

Table 2. sample 3-SAT instance with 7 variables, 30 clauses

$\{-4,\ 7,\ 2\}$	$\{-3,-6,-4\}$	$\{-6,\ 4,-2\}$	$\{-4,-1,\ 7\}$	$\{\ 5,\ 6,-7\}$	$\{-6,-7,\ 4\}$
$\{-3,\ 7,\ 2\}$	$\{-5,\ 2,-1\}$	$\{-6,-5,\ 4\}$	$\{-2,-1,\ 3\}$	$\{-7,\ 1,-3\}$	$\{\ 4,\ 6,\ 3\}$
$\{-7,-4,\ 5\}$	$\{-7,-5,-3\}$	$\{-1,-7,-5\}$	$\{3,\ 5,\ 4\}$	$\{\ 7,-6,-5\}$	$\{-1,\ 4,-6\}$
$\{\ 1,\ 3,-2\}$	$\{\ 2,-4,\ 5\}$	$\{\ 4,-5,-3\}$	$\{-3,-2,-5\}$	$\{-1,\ 5,\ 2\}$	$\{-6,-1,\ 3\}$
$\{\ 5,-2,-7\}$	$\{\ 1,-5,6\}$	$\{\ 6,-1,-4\}$	$\{\ 1,\ 6,-3\}$	$\{-3,-6,\ 1\}$	$\{\ 5,\ 6,-2\}$