November 1990

On the Convergence Time of Simulated Annealing

Sanguthevar Rajasekaran
University of Pennsylvania

On the Convergence Time of Simulated Annealing

Abstract

Simulated Annealing is a family of randomized algorithms used to solve many combinatorial optimization problems. In practice they have been applied to solve some presumably hard (e.g., NP-complete) problems. The level of performance obtained has been promised [5, 2, 6, 14]. The success of its heuristic technique has motivated analysis of this algorithm from a theoretical point of view. In particularly, people have looked at the convergence of this algorithm. They have shown (see e.g., [10]) that this algorithm converges in the limit to a globally optimal solution with probability 1. However few of these convergence results specify a time limit within which the algorithm is guaranteed to converge(with some high probability, say). We present, for the first time, a simple analysis of SA that will provide a time bound for convergence with overwhelming probability. The analysis will hold no matter what annealing schedule is used. Convergence of Simulated Annealing in the limit will follow as a corollary to our time convergence proof.

In this paper we also look at optimization problems for which the cost function has some special properties. We prove that for these problems the convergence is much faster. In particular, we give a simpler and more general proof of convergence for Nested Annealing, a heuristic algorithm developed in [12]. Nested Annealing is based on defining a graph corresponding to the given optimization problem. If this graph is 'small separable', they [12] show that Nested Annealing will converge 'faster'.

For arbitrary optimization problem, we may not have any knowledge about the 'separability' of its graph. In this paper we give tight bounds for the 'separability' of a random graph. We then use these bounds to analyze the expected behavior of Nested Annealing on an arbitrary optimization problem. The 'separability' bounds we derive in this paper are of independent interest and have the potential of finding other applications.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-90-89.

On The Convergence Time Of Simulated Annealing

MS-CIS-90-89
GRASP LAB 242

Sanguthevar Rajasekaran

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

November 1990

On the Convergence Time of Simulated Annealing

Sanguthevar Rajasekaran
Dept. of Computer and Information Sciences
University of Pennsylvania
Philadelphia, PA 19104

ABSTRACT

Simulated Annealing is a family of randomized algorithms used to solve many combinatorial optimization problems. In practice they have been applied to solve some presumably hard (e.g., NP-complete) problems. The level of performance obtained has been promising [5, 2, 6, 14]. The success of this heuristic technique has motivated analysis of this algorithm from a theoretical point of view. In particular, people have looked at the convergence of this algorithm. They have shown (see e.g., [10]) that this algorithm converges in the limit to a globally optimal solution with probability 1. However few of these convergence results specify a time limit within which the algorithm is guaranteed to converge (with some high probability, say). We present, for the first time, a simple analysis of SA that will provide a time bound for convergence with overwhelming probability. The analysis will hold no matter what annealing schedule is used. Convergence of Simulated Annealing in the limit will follow as a corollary to our time convergence proof.

In this paper we also look at optimization problems for which the cost function has some special properties. We prove that for these problems the convergence is much faster. In particular, we give a simpler and more general proof of convergence for Nested Annealing, a heuristic algorithm developed in [12]. Nested Annealing is based on defining a graph corresponding to the given optimization problem. If this graph is 'small separable', they [12] show that Nested Annealing will converge 'faster'.

For an arbitrary optimization problem, we may not have any knowledge about the 'separability' of its graph. In this paper we give tight bounds for the 'separability' of a random graph. We then use these bounds to analyze the expected behavior of Nested Annealing on an arbitrary optimization problem. The 'separability' bounds we derive in this paper are of independent interest and have the potential of finding other applications.

1 Introduction

1.1 Basic Ideas

Simulated Annealing (abbreviated as SA) is a hueristic algorithm proposed by Kirkpatrick et. al. [7] for solving combinatorial optimization problems like the traveling salesman problem etc. This randomized algorithm was derived in analogy with a physical system, say, a fluid. Annealing is used to bring a fluid to a low energy state. The idea of SA is to use a procedure similar to annealing to find the minimum value of a given cost function. SA makes use of the Metropolis algorithm for computer simulation of annealing.

Annealing a substance involves melting the substance at a very high temperature and then cooling it slowly. At each temperature sufficient time should be given for the system to reach a steady state. The lower the temperature, the higher the time given should be. SA imitates this procedure by first identifying the correspondence between the fluid and the optimization problem, and then using the Metropolis' algorithm to simulate each step of annealing.

Each basic step of Metropolis' simulation amounts to perturbing the value of one of the variables by a small amount. If the new configuration has a lower cost, the configuration will be accepted. If the new configuration has a higher cost, even then it will be accepted with certain probability.

SA algorithm is repeated application of the above basic step until no more improvement in the cost function is possible. As evident, SA allows hill climbing from local optima.

1.2 Previous Work

Many researchers have attempted to analyze the performance of SA by assuming a mathematical model for it. The most popular model assumed is a (time inhomogeneous) Markov chain. Typical convergence proof [10] involved proving that the probability state vector of the Markov chain converges in the limit to an 'optimal stationary probability state vector' with probability 1. An optimal stationary probability state vector is one in which the only non-zero entries correspond to globally optimal states of the Markov chain. There are also results which compute the rate of convergence of the probability state vector to the optimal vector. Such results are extremely important from a theoretical point of view since they provide an explanation for why SA works in practice.

However none of these convergence proofs gives a time bound for convergence.

1.3 Contents of this paper

This paper analyzes the worst case convergence time of SA. In particular, we show that SA converges in time $M(n, d, D)$ with probability $\geq (1 - n^{-\Omega(1)})$. Here n is the number of states, D is the diameter, and d is the degree of the underlying Markov chain. $M(.)$ is a function to be specified later. By convergence we mean that the Markov chain had been in a globally optimal state at least once. We don't require the state probability vector to converge to an optimum vector.

Since the objective in combinatorial optimization is to find the optimum value of a given cost function, perhaps our definition of convergence is more appropriate from a computational point of view. If we can keep track of the state with the minimum cost visited so far by the Markov chain, our objective will be achieved.

We also show that faster convergence is possible in the case of cost functions with some special properties. For cost functions which are 'small separable' faster convergence has already been proved by Rajasekaran and Reif (see [12]). They call their algorithm 'Nested Annealing' (abbreviated as NA). In fact we also consider the same class of cost functions and give a simpler proof to the convergence of Nested Annealing. Also their proof applies only to problems for which SA algorithm converges in time polynomial in the number of states (of the corresponding Markov chain), whereas our proof applies to any problem.

NA employs the following idea. Given any optimization problem, define a corresponding graph. If this graph is known to be 'small separable', then NA leads to 'faster' convergence. Numerous other algorithms have also been designed which exploit the 'small separability' of the underlying graphs (see e.g., [8]). But, given an arbitrary problem, we may not know how 'separable' its graph is. In this paper we prove tight bounds on the 'separability' of a random graph. We make use of these 'separability' bounds to analyze the expected behavior of NA on an arbitrary optimization problem.

The 'separability' bounds derived here are of independent interest and have the potential of finding other applications. For example, we could study the expected behavior of the algorithms given in [8] on arbitrary graphs.

1.4 Some Definitions and Preliminaries

SA is a class of randomized algorithms in the sense of Rabin [11], and Solovay & Strassen [13]. Traditional approaches to introducing randomness in algorithms was done assuming certain distribution for possible inputs. Average case analysis of any algorithm will be performed based on this assumption on the inputs. This average case performance measure

can be totally misleading in the case of applications where the input distribution assumed does not hold. To rectify this problem Rabin [11], and Solovay & Strassen [13] proposed introducing randomness in the algorithm itself. An algorithm employing this technique is called a 'randomized algorithm'.

To be more precise, a randomized algorithm is one which makes coin flips to make certain decisions. A randomized algorithm will be shown to have a certain performance measure with 'high probability' (this probability will be over the space of all possible outcomes for coin flips made in the algorithm and not over the space of all possible inputs).

We say a randomized algorithm has a resource (like time, space, etc.) bound of $\tilde{O}(f(n))$ if there exists a constant c such that the resource used by the algorithm is no more than $c\alpha f(n)$ on any input of size n, with probability $\geq (1 - n^{-\alpha})$.

An *Optimization Problem* is to find the minimum value of a given 'cost function' $C(.)$ of n parameters $p_1, p_2, \ldots, p_n$, subject to some given constraints.

If X is any non-negative random variable with mean μ, Markov's inequality (see e.g., [3]) implies that Prob.$[X \geq k\mu] \leq 1/k$, for any $k > 0$.

2 A Model for SA

In this section we state the mathematical model to be used for SA. Before doing so, we give a description of the SA algorithm itself.

Given a cost function $C(p_1, p_2, \ldots, p_n)$, a *configuration* or *state* of the Optimization Problem (abbreviated as OP) is defined to be an assignment of values to its n parameters. 'Neighbors' of a given state are all those states which differ from the given state only in the value of one parameter by a 'small' amount. 'Temperature' in the case of an OP is simply a control parameter which has the same units as that of the cost function.

procedure Anneal(start_state, start_temperature);

while (not frozen) *do*

while (not steadystate) *do*

Generate a random neighbor of the current state;
Let Δ = cost of old state − cost of the new state;
Accept the new state with probability $\min\{1, \exp(-\Delta/T)\}$,
T being the current temperature;

Update the temperature; steadystate = *false*;

In the above algorithm 'frozen' is a boolean variable that becomes *true* when no improvement in the given cost function has been observed in a 'long' time (say during the past few temperatures). The boolean variable 'steadystate' becomes *true* when the system attains steady state at the given temperature.

Many schemes (also called 'annealing schedules') have been proposed to compute the sequence of temperatures the system should go through. For example, the temperature can be decreased by a constant factor each time. For other annealing schedules see [10].

One can construct a directed graph $G(V, E)$ corresponding to a given OP in the following way: Nodes in G are simply the states of the OP and the edges going out of any node (or state) will be the neighbors of this state. As the reader can easily see, SA algorithm performs a random walk on this graph. State transition made at any step is dependent only on the current state. Also for a given temperature, the transition probabilities from out of any node are fixed. These facts suggest modeling SA as a Markov chain. At any given temperature SA can be modeled as a time homogeneous Markov chain. But the transition probabilities can potentially change with temperature. Thus the whole of SA can be modeled as a time inhomogeneous Markov chain [10]. We assume G is strongly connected (i.e., there is a directed path from i to j for any two nodes i and j in V).

For any node i in G, let $N(i)$ be the set of neighbors of i, and let $C(i)$ be the cost of state i. If we assume that when the Markov chain is in state i, each one of its $|N(i)|$ neighbors is equally likely to be generated next, then, the transition probability from state i to state j at temperature T, $P_{ij}(T)$, is given by

$$P_{ij}(T) = \begin{cases} 0 & \text{if } j \notin N(i) \& j \neq i \\ \frac{1}{|N(i)|} \min(1, \exp\{(C(j) - C(i))/T\}) & \text{if } j \in N(i) \end{cases}$$

and

$$P_{ii}(T) = 1 - \sum_{j \in N(i)} P_{ij}(T)$$

We make use of this model to prove the convergence time of SA.

3 Convergence of SA

We say the SA algorithm has converged if the underlying Markov chain had been in a globally optimal state at least once. This definition of convergence is different from what other researchers have assumed. So far, only the convergence of the probability state vector to an optimal vector has been considered (see e.g. [10]). Our definition of convergence is

more appropriate from a computational point of view, since we are only interested in finding the minimum value of a given cost function. Convergence as defined here guarantees that we will find the global minimum of the given function.

In this section we give a time bound within which the SA algorithm will converge. Given a cost function $C(p_1, p_2, \ldots, p_n)$, let $G(V, E)$ be its state graph (as defined before). Let T be the minimum temperature that SA was ever in. Also let $\Delta = \max_{i \in V,\ j \in N(i)}\{C(i) - C(j)\}$. Denote the degree and diameter of $G(V, E)$ by d and D respectively.

Clearly, for any $i \in V$, P_{ij} (at any temperature) will be at least $\exp(-\Delta/T)$ if $j \in N(i)$. We state a few crucial facts before we present and prove the main theorem.

Fact 3.1 *Let X be the state of a Markov chain at time $t = t_0$. The probability that a global minimum state is visited during the next q steps (for any q) is dependent only on X and q and not on the states visited before.*

Lemma 3.1 *If X is any state in V, then the expected number of steps before a global optimal state is visited starting from X is $\leq \left(\frac{1}{d}\exp(-\Delta/T)\right)^{-D}$.*

Proof. Let g be any global optimal state. Then there exists a directed path from X to g in $G(V, E)$ of length $q \leq D$. Let $e_1, e_2, \ldots, e_q$ be the sequence of edges in the path.

Clearly, probability that g is visited starting from X is at least the probability that each one of the edges e_i, $1 \leq i \leq q$ is traversed in succession. The later probability is at least $\left[(\frac{1}{d})\exp(-\Delta/T)\right]^q \geq \left[(\frac{1}{d})\exp(-\Delta/T)\right]^D$, (assuming that each neighbor of a state is equally likely to be generated next).

Therefore, probability that g will ever be visited starting from X is $\geq [(1/d)\exp(-\Delta/T)]^D$. This implies that the expected number of steps before g is visited is $\leq [d\ \exp(\Delta/T)]^D$. □

Theorem 3.1 *SA converges in time $\leq 2k[d\ \exp(\Delta/T)]^D$, with probability $\geq (1 - 2^{-k})$, no matter what the start state is.*

Proof. Let $E = 2[d\ \exp(\Delta/T)]^D$. We prove that the probability of a global optimal state g not being visited in kE steps is $\leq 2^{-k}$, by induction on k.

Induction Hypothesis. Irrespective of the start state, probability that g is not visited in kE steps is $\leq 2^{-k}$.

Base case. When $k = 1$, for any start state X, expected number of steps before g is visited is $\leq E/2$ (using lemma 3.1). An application of Markov's inequality implies that the probability of g not being visited starting from X in E steps is $\leq 1/2$.

Induction step. Assume the hypothesis for all $k \leq (r-1)$. We'll prove the hypothesis for $k = r$.

Let $X_E, X_{2E}, \ldots, X_{(r-1)E}$ be the states of the Markov chain during time steps $E, 2E, \ldots, (r-1)E$ respectively. Let A be the event: g is not visited during the first E steps, and B be the event: g is not visited during the next $(r-1)E$ steps.

Now, probability that g is not visited in rE steps, P, is given by

$$P = Prob.[B/A] \times Prob.[A].$$

Using fact 3.1, $Prob.[B/A]$ depends only on what state the Markov chain is in at time step E and the time duration $(r-1)E$. And hence,

$$P = Prob.[A] \sum_{i \in V} Prob.[B/X_E = i] \times Prob.[X_E = i].$$

But, Prob.[A] is $\leq 1/2$ and Prob.[B/$X_E = i$] is $\leq 2^{-(r-1)}$ for each $i \in V$ (using the induction hypothesis). Therefore, we have,

$$P \leq \frac{1}{2} 2^{-(r-1)} = 2^{-r}.\square$$

Corollary 3.1 *If Δ, T, and d are assumed to be constants and $D = \theta(\log |V|)$, then, SA converges in time polynomial in $|V|$ with probability $\geq (1 - 2^{-\Omega(|V|)})$.*

Observation. The above analysis is oblivious to the annealing schedule used. Even if the system stays in the same temperature throughout, as long as enough time is given, the system will converge.

The success of any heuristic technique depends on how good the input is. If the heuristic is good on all possible inputs, then it will be termed an 'exact algorithm'. This fact is true in the case of SA also.

In practice, SA has been used to obtain 'quasi optimal' solutions by running it for only a small amount of time (in comparison with the time bound given in theorem 3.1). A plausible reason for this behavior is there are many quasi optimal states (for a large fraction of all possible inputs) in the Markov chain and these states are nearly uniformly distributed in $G(V, E)$. If we have some knowledge of how many quasi optimal states there are, and how they are distributed in $G(V, E)$, we can make use of the above analysis technique to get tighter bounds on the convergence time.

Also, the above proof assumes that each neighbor of a state is equally likely to be generated next. This is not a severe restriction. The analysis can be extended easily even to the case where this assumption is invalid as we show now.

In some applications, it may be necessary to generate certain neighbors with higher probability. Mitra et. al. [10] assume that the probability of generating state j from state i is $\frac{g(i)}{g(j)}$ where $g(i,j)$ is the 'weight' of j as a neighbor of i and $g(i)$ is a normalizing function such that $\sum_{j \in N(i)} g(i,j) = g(i)$. Under this assumption the state transition probabilities become:

$$P_{ij}(T) = \begin{cases} 0 & \text{if } j \notin N(i) \& j \neq i \\ \frac{g(i,j)}{g(i)} \min(1, \exp\{(C(j) - C(i))/T\}) & \text{if } j \in N(i) \end{cases}$$

and

$$P_{ii}(T) = 1 - \sum_{j \in N(i)} P_{ij}(T).$$

If $p = \min_{i \in V,\ j \in N(i)} \frac{g(i,j)}{g(i)}$, for the above general model we can prove a convergence result similar to the one given by theorem 3.1. We can prove the following

Theorem 3.2 *SA algorithm converges in time* $\leq 2k \left[\frac{1}{p} \exp(\Delta/T)\right]^D$ *with probability* $\geq (1 - 2^{-k})$, *no matter what the start state is.*

4 Cost Functions with Special Properties

Rajasekaran and Reif [12] have shown that if the cost function being optimized is 'small separable', then faster convergence can be obtained. They call their algorithm 'Nested Annealing'. In this section we give a simpler proof of convergence of Nested Annealing. Their convergence proof holds only for problems for which SA converges in time polynomial in the number of states of the OP. However, our proof generalizes it to any problem. We make a few definitions before presenting the proof.

4.1 Definitions

We say a graph $G(V,E)$ with n nodes is '$s(n)$– separable' if there exist constants $\alpha < 1$, $\beta > 0$ such that V can be partitioned into three subsets V_1, S, V_2. Also, no vertex in V_1 is adjacent to any vertex in V_2, both $|V_1|$ and $|V_2|$ are less than αn, and $|S|$ is less than $\beta s(n)$. Moreover, the induced subgraphs of G on V_1, and on V_2 are $s(|V_1|)$–separable and $s(V_2)$–separable, respectively. S will be referred to as 'the separator set' or simply 'the separator'. Intuitively, by eliminating (the nodes in) S from G we end up with two roughly equal disjoint subgraphs.

If C is a cost function on n parameters $p_1, p_2, \ldots, p_n$, we define 'separability' of C as follows. Write C as $C = C_1 + C_2 + \ldots + C_k$, where each C_i is a product of functions of the

parameters. Call each $C_i, 1 \leq i \leq k$ as a clause. Define a bipartite graph $G_C(V, E)$ whose nodes are the parameters and the clauses. There is an edge between a clause node and a parameter node if that parameter occurs in that clause. G_C is called the 'graph of C'. We say C is $s(n)$–separable if G_C is.

An Example. The problem of CNF-satisfiability is: given a boolean formula in conjunctive normal form, F, on n variables, we have to decide if there is an assignment to the variables that will make F *true*. The graph corresponding to this problem will consist of nodes one for each variable and each clause. There is an edge between a clause node and a variable node if and only if that variable occurs in that clause.

Small separability (i.e., small $s(n)$) of a cost function implies that by assigning values to a small number of parameters we can obtain two independent subproblems such that the parameters involved in one subproblem are disjoint from the parameters of the other subproblem.

4.2 The Algorithm

Given a cost function C on the n parameters $p_1, p_2, \ldots, p_n$, construct the graph of C, $G_C(V, E)$. If G_C is $s(n)$–separable, we can partition V into V_1, S, and V_2 as mentioned above.

In this section we assume each parameter is binary (i.e., can take on only two possible values). The analysis we perform is applicable with some minor changes to other cases as well. One way of computing the minimum value of C is as follows. For each possible assignment of values to parameters in S, find the minimum value of C, and pick the minimum of these minima. Finding the minimum of C under a particular assignment for S, is easy now. We need to find the minimum of two functions C_1 and C_2 where C_1 involves only parameters from V_1 and C_2 involves only parameters from V_2.

Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be the restrictions of G on V_1 and V_2 respectively. Finding the minimum of C_1 and C_2 can be done recursively by finding separators for G_1 and G_2 respectively.

At the top level, we are given a set S for which we need to find an 'optimum' assignment (an assignment that corresponds to a global minimum for C). We can think of this as an OP on $|S|$ parameters. There are thus $2^{|S|} \leq 2^{\beta s(n)}$ states of the OP. The cost of each state is the minimum of C under that particular assignment to S. Instead of considering each possible state of this OP, and computing the cost of each state, we can run a Simulated Annealing algorithm on this OP with $\leq \beta s(n)$ parameters.

SA algorithm, in practice, only visits a small fraction of all possible states of the OP to come up with a quasi-optimal solution. Therefore, if we use SA on the above OP with $|S|$ parameters, the number of states visited will be much less than $2^{\beta s(n)}$ and hence the run time of the overall recursive algorithm will be much less. This is the whole idea behind Nested Annealing.

Next we give a formal description of the algorithm followed by a simpler proof of convergence.

procedure Nested_Annealing($G_C(V, E)$);

> Find a separator set S for G_C. Let V_1, S, and V_2 be the partition of V.
> Also let G_1 and G_2 be restrictions of G on V_1 and V_2 respectively.
> Find an optimal assignment for S by running an SA algorithm on these parameters. For each state of the corresponding Markov chain visited by SA we need to compute the cost.
> To compute this cost, we need to find minimum of two other functions C_1 and C_2 (see the discussion above). Each of C_1 and C_2 involves $\leq \alpha n$ parameters.
> Find these two minima recursively by finding separators for G_1 and G_2 respectively.

Analysis. Let $T(n)$ be the expected run time of Nested_Annealing to find a global optimal solution on any OP with n parameters. To obtain an upper bound on $T(n)$, we need to know how many of the $2^{|S|}$ states at the top level will be visited (including repeated visits) by the corresponding SA algorithm, and on each state visited the time needed to compute the cost of the state. Let $2^{M(n)}$ stand for the worst case run time of SA on any OP with n parameters. (In section 3 we have given upper bounds on $2^{M(n)}$). Then, clearly, the number of states visited will be no more than $2^{M(\beta s(n))}$.

Computing the cost of each state involves computing the minimum of two other functions involving no more than αn parameters each, accounting for a total expected cost of $\leq$ 2 $T(\alpha n)$.

Thus we have,

$$T(n) \leq 2^{M(\beta s(n))}\, 2T(\alpha n),$$

which solves to

$$T(n) \leq 2^{\sum_{i=1}^{\log n} M(\beta s(\alpha^i n))}.$$

If $s(n)$ is assumed to be $O(n^\sigma)$ for some $\sigma < 1$, we have

$$T(n) \leq 2^{\gamma\ M(\beta s(n))} = 2^{O(M(\beta s(n)))}.$$

Here $\gamma \leq \frac{1}{1-\sqrt{\alpha}}$. Throughout we have used the fact that the expected value of the sum of any random variables is the sum of the expected values of the individual random variables.

Let $L = 2^{\gamma\ M(\beta s(n))}$. Probability that the run time of Nested_Annealing exceeds kL is less than $1/k$, using Markov's inequality. Thus we have the following

Theorem 4.1 *Nested_Annealing converges in time* $\leq n^\alpha\ L$ *with probability* $\geq (1 - n^{-\alpha})$.

For problems with $M(n) = O(n)$, we have the following

Corollary 4.1 *If* $M(n)$ *is* $O(n)$, *Nested_Annealing converges in time* $\leq n^\alpha 2^{O(s(n))}$ *with probability* $\geq (1 - n^{-\alpha})$.

The above corollary has already been proven in [12].

We can strengthen theorem 4.1 in the following way. Probability that the convergence time of Nested_Annealing exceeds $2L$ is $\leq 1/2$. Make $k \log n$ independent runs of the procedure and terminate each procedure after exactly $2L$ steps. Pick the minimum of the minima found in the $\log n$ runs. Probability that none of the runs finds the global minimum is $\leq 2^{-k}$. Thus we have the following

Theorem 4.2 *Nested_Annealing converges in time* $\leq 2kL$ *with probability* $\geq (1 - 2^{-k})$ *(for any* $k > 0$*).*

A number of important problems like planar traveling salesman, planar satisfiability, etc. (which have been proven to be NP-complete) have $s(n) = \sqrt{n}$. For all these problems Nested_Annealing converges in time $2^{O(M(\beta\sqrt{n}))}$, whereas SA has a convergence time of $2^{O(M(n))}$.

Even though the above analysis gives the time needed to find the global optimal value in the worst case, we can also make use of it to get estimates of time bounds to obtain quasi optimal solutions. If $2^{M(n)}$ is an estimate on the run time of SA to obtain a quasi optimal solution of any OP with n parameters, then theorem 4.2 can be interpreted as implying that for the same problem Nested_Annealing will run in time $2^{O(M(\beta s(n)))}$.

5 Separability of Random Graphs

All the algorithms that exploit the separability of the underlying graphs presuppose that a separator is known for the given graph. There are algorithms for finding separators of restricted classes of graphs. For example, if the graph is planar, efficient algorithms exist for computing the separator set [8]. But in practice we may know nothing about the graph being manipulated. In fact, deciding if a given graph is $s(n)$-separable is NP-hard [4]. In this section we prove tight bounds for the separability of random graphs. We use these bounds to study the expected behavior of NA on arbitrary OPs.

5.1 A Modification of NA

The separability results in this section assert that random graphs are not 'small separable'. Let $G_C(V, E)$ be the graph of a given OP. If G_C is $s(n)-$separable, then V can be partitioned into V_1, S, and V_2 such that $|S| = s(n)$ and there is no $V_1 - V_2$ edge. Moreover, $|V_1|$ and $|V_2|$ are less than αn for some $\alpha < 1$. If the cardinality of the separator set, S, itself is a constant fraction (or more) of $|V_1|$ and $|V_2|$, then there is no gain in running all the levels of recursion of the algorithm Nested_Annealing (given in section 4.2). It may suffice to stop the procedure at the top level. In more precise terms, we can modify Nested_Annealing procedure in the following way: Replace the instructions that call for computing the minimum of C_1 and C_2 recursively, with instructions to compute these two minima using SA.

This modified NA will have an expected convergence time of $2^{M(|S|)}\left[2^{M(|V_1|)} + 2^{M(|V_2|)}\right] = O\left(2^{M(|S|+\max[|V_1|,|V_2|])}\right)$. Here $2^{M(n)}$ stands for the convergence time of SA on an OP with n parameters.

If the separability of the given OP is not known we propose using this modified NA together with the procedure (given in this section) for finding a separator in a random graph. In this section we give expected bounds on the convergence time of this modified NA on an arbitrary OP. The expected convergence of the original Nested_Annealing will be nearly the same (upto a multiplicative constant) since (as we prove here) a random graph is not 'small separable'.

5.2 The Separator Theorems

There are two popular models of random graphs [1]. The first consists of all graphs with n vertices and $M(n)$ edges (for some specified $M(n)$), each such graph having equal probability. A member in this model is denoted as $G_{M(n)}$ or simply G_M. The second model consists of

all graphs with n nodes in which each of the $n^2/2$ possible edges is chosen independently with probability p. A member in this model is denoted as G_p.

There is a close connection between G_M and G_p (with $p = \theta(M/n^2)$) (read e.g., chapter II of [1]). In this paper we assume the second model and derive tight bounds on the separability of G_p. Even though the results proved are for a general graph, they are easily extendible to bipartite graphs. A random bipartite graph in the second model is denoted as $G_p(A, B, E)$ where both A and B have n nodes each and each possible edge is chosen independently with probability p. For any set of nodes X, we let $\Gamma(X)$ stand for the neighbors of X.

Next we state and prove the separability results.

Theorem 5.1 *Let p be $> \frac{100}{n}$. Then, almost every $G_p(V, E)$ has the following property. If X is* **any** *subset of nodes of G_p with $\frac{1}{p}$ nodes, then, the set $T_X = \{y \in V - X : \Gamma(y) \cap X = \Phi\}$ has at least $\frac{1}{p}$ nodes.*

Proof. Let $X \subset V$ be a set with $m = \frac{1}{p}$ nodes. If q is any node in $V - X$, then, Prob.$[\Gamma(q) \cap X \neq \Phi] = 1 - (1-p)^{|X|}$. This is the probability that X is a neighbor of q.

X will have a neighbor in each possible k-subset (for any k) of $V - X$ if and only if X is a neighbor of at least $n - |X| - (k-1)$ nodes in $V - X$. Thus the probability, P', that X has a neighbor in each possible m-subset of $V - X$, is given by

$$P' \leq \sum_{k=0}^{m-1} \binom{n}{k} \left[1 - (1-p)^{|X|}\right]^{n-|X|-k}.$$

But $(1-p)^{|X|} = (1-p)^{1/p} \geq 1/(2e)$. Therefore,

$$P' \leq \sum_{k=0}^{m-1} \binom{n}{k} \left(1 - \frac{1}{2e}\right)^{n-m-k} \leq m \binom{n}{m} \left(1 - \frac{1}{2e}\right)^{n-2m}.$$

The probability, P, that there is at least one X such that X is a neighbor of each m-subset of $V - X$, is given by

$$P \leq \binom{n}{m} m \binom{n}{m} \left(1 - \frac{1}{2e}\right)^{n-2m}.$$

Using the fact that m is $< .01n$ and the fact that $\binom{n}{\epsilon n}$ (for any small $\epsilon < 1$) is nearly equal to $2^{H(\epsilon)n}$ (where $H(.)$ is the binary entropy function defined by $H(\epsilon) = \epsilon \log \frac{1}{\epsilon} + (1 - \epsilon) \log \frac{1}{(1-\epsilon)}$), we get,

$$P \leq 2^{-0.1n}. \square$$

Corollary 5.1 *Almost every $G_p(V,E)$ is such that V can be partitioned into V_1, S, V_2 in such a way that there is no $V_1 - V_2$ edge and $|V_1| = \frac{1}{p}$. Also V_1 can be chosen to be* **any** *set of $\frac{1}{p}$ nodes.*

The above corollary suggests the following simple procedure for finding a separator set for $G_p(V,E)$. Take any set of $1/p$ nodes as V_1, $\Gamma(V_1) - V_1$ as S, and $V - S - V_1$ as V_2. The separator set so found will be such that $|S| + \max[|V_1|, |V_2|]$ is no more than $n - \frac{1}{p}$. A similar result can be proven for a bipartite graph (proof is along the same lines and hence omitted) which will imply the following

Theorem 5.2 *Modified NA converges in an expected $O(2^{M(n-\frac{1}{p})})$ time on an arbitrary OP with n parameters, given that the graph of the OP is a member of G_p.*

A good guess for p will be $\theta(M/n^2)$ where M is the number of edges in the graph of the OP. This modified Nested Annealing is being currently implemented. Results of the experiments will appear in a subsequent paper.

Next we show that the bound given in the above theorem is essentially tight.

Theorem 5.3 *Let $\frac{1}{p} \to \infty$. For almost every $G_p(V,E)$, the following holds. If X is any set of δn nodes (for any δ), then the cardinality of $T_X = \{y \in V - X : \Gamma(y) \cap X = \Phi\}$ is at the most $\left[\left(\frac{2-\delta}{\delta \log e}\right) + \epsilon\right] \frac{1}{p}$, for any constant $\epsilon > 0$.*

Proof. [A similar theorem has been proven in [1] (page 47, theorem 15).] Let X be a set of δn nodes and Y be any set of y nodes in $V - X$. Probability that X has no neighbor in Y is $(1-p)^{y\delta n}$. Thus, the probability, P, that there is at least one X whose T_X has cardinality y is given by

$$P \leq \binom{n}{\delta n}\binom{n-\delta n}{y}(1-p)^{y\delta n}$$
$$\leq 2^{(2-\delta)n - yp\delta n \log e}.$$

If y is $> \left[\left(\frac{2-\delta)}{\delta \log e}\right) + \epsilon\right] \frac{1}{p}$, then, $P \leq 2^{-\epsilon/p}$.□

Corollary 5.2 *If X is any set of δn nodes (where δ is a constant), then, T_X has cardinality $O(1/p)$.*

Similar results hold for bipartite graphs as well.

6 Conclusions

In this paper we have defined the convergence of SA to mean that an optimal global state has been visited at least once by the Markov chain of the OP. This definition of convergence is perhaps more appropriate from a computational point of view. We gave a proof of convergence of an SA algorithm (for a general annealing schedule). An important open problem will be to obtain tighter bounds for the convergence time.

Nested Annealing is a variation of SA proposed in [12]. Nested Annealing has been proven to be faster for small separable cost functions. We gave a simpler proof of convergence for Nested Annealing. We also generalized the convergence results of [12].

Further, we have analyzed the expected convergence time of NA on an arbitrary problem. This was possible as a result of bounds we derived for the separability of a random graph.

References

[1] Bollobás, B., *Random Graphs*, Published by Academic Press, 1985.

[2] ElGamal, A., and Shperling, I., 'Design of Good Codes via Simulated Annealing,' List of Abstracts, Workshop on Statistical Physics in Engineering and Biology, Yorktown Heights, NY, April 1984.

[3] Feller, W., *An Introduction to Probability Theory and its Applications*, Volumes I and II, Published by John Wiley and sons, 1966.

[4] Garey, M., and Johnson, D., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Published by W.H. Freeman & Co., New York, 1979.

[5] Golden, B.L., and Skiscim, C.C., 'Using Simulated Annealing to solve Routing and Location Problems,' Naval Research Logistics Quarterly, 33, 1986, pp. 261-279.

[6] Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C., 'Optimization by Simulated Annealing: An Experimental Evaluation (Part I),' Preliminary Draft, AT&T Bell Labs., Murray Hill, NJ, 1987.

[7] Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., 'Optimization by Simulated Annealing,' Science, May 1983.

[8] Lipton, R.J., and Tarjan, R.E., 'Applications of a Planar Separator Theorem,' SIAM Journal on Computing, vol.9, no.3, 1980.

[9] Lipton, R.J., and Tarjan, R.E., 'A Planar Separator Theorem,' SIAM Journal on Applied Mathematics, vol.36, no.2, 1979.

[10] Mitra, D.,Romeo, F., and Vincentelli, A.S., 'Convergence and Finite-Time Behavior of Simulated Annealing,' Advances in Applied Probability, Sept. 1986.

[11] Rabin, M.O., 'Probabilistic Algorithms,' in: Traub, J.F., ed., *Algorithms and Complexity*, Academic Press, New York, 1976. pp. 21-36.

[12] Rajasekaran, S., and Reif, J.H., 'Nested Annealing: A Provable Improvement to Simulated Annealing,' Proc. ICALP 1988. Also submitted to Theoretical Computer Science, 1988.

[13] Solovay, R., and Strassen, V., 'A Fast Monte-Carlo Test for Primality,' SIAM Journal of Computing, vol.6, 1977. pp. 84-85.

[14] Vecchi, M.P., and Kirkpatrick, S., 'Global Wiring by Simulated Annealing,' Technical Report, IBM Thomas J. Watson Research Center, New York 10598.