Randomization in Discrete Optimization: Annealing Algorithms

Sanguthevar Rajasekaran
Dept. of Computer and Information Science and Engineering
University of Florida, Gainesville, FL 32611
raj@cise.ufl.edu

Abstract

Annealing algorithms have been employed extensively in the past decade to solve myriads of optimization problems. Several intractable problems such as the traveling salesperson problem, graph partitioning, circuit layout, etc. have been solved to get satisfactory results. In this article we survey convergence results known for annealing algorithms. In particular we deal with Simulated Annealing and Nested Annealing.

Simulated Annealing (SA) is a randomized heuristic that can be used to solve any combinatorial optimization problem. SA is typically used to produce quasi-optimal results. In practice SA has been applied to solve some presumably hard (e.g., NP-hard) problems. The level of performance obtained has been promising (Golden and Skiscim 1986, ElGamal and Shperling April 1984, Johnson et al. 1987, Vecchi and Kirkpatrick 1982). The success of this heuristic technique has motivated the study of convergence of this technique. One of the early results in this direction is due to Mitra et al. (Sept. 1986)) who proved that SA converges in the limit to a globally optimal solution with probability 1. Later results proved certain time bounds within which SA is guaranteed to converge with high probability. In this article we provide one such proof (due to Rajasekaran (2000)).

Nested Annealing (NA) (Rajasekaran and Reif 1992) is a variation of SA that has been proven to perform better for optimization problems for which the cost function has some special properties (Rajasekaran and Reif 1992). In this article we provide a summary of NA and its convergence properties.

1 Introduction

SA is a heuristic technique proposed by Kirkpatrick et al. (May 1983) for solving combinatorial optimization problems such as the traveling salesman problem, graph partitioning, etc. This randomized algorithm was obtained in analogy with a physical system, say, a fluid. A fluid can be brought to a low energy state using
annealing. SA uses a procedure similar to annealing to compute the minimum value of a given objective function. In particular, SA employs the Metropolis algorithm for the computer simulation of annealing.

Annealing a substance involves melting the substance at a very high temperature and then cooling it slowly. At each temperature the system should reach a steady state and hence enough time should be given. The lower the temperature, the higher the time given should be. SA imitates this process using the Metropolis’ algorithm.

Each basic step of Metropolis’ simulation perturbs the value of one of the variables by a small amount. The resultant configuration will be accepted if it has a lower cost. If the new configuration has a higher cost, even then it will be accepted with certain probability. This enables the system to hill-climb from a locally optimal state.

SA is the repeated application of the above basic step until no more improvement in the cost function is possible.

SA can be modeled as a time inhomogeneous Markov chain. Mitra et al. (Sept. 1986) have employed such a model to prove the convergence of SA. This convergence proof 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.

The result of (Rajasekaran 2000) shows that SA converges in time $M(n, d, D)$ with probability $\geq (1 - n^{-\Omega(1)})$ for some appropriate function $M(\cdot)$. Here $n$ is the number of states, $D$ is the diameter, and $d$ is the degree of the underlying Markov chain. Convergence in this context means that the Markov chain had been in a globally optimal state at least once. Perhaps this notion of convergence is more relevant than the earlier notions, from a computing point of view. By keeping track of the state with the minimum cost visited so far by the Markov chain, the globally optimal solution can be found.

It can be shown that the performance of SA can be improved when the cost functions have some special properties. For cost functions which are ‘small separable’ faster convergence has already been proved by Rajasekaran and Reif (see (Rajasekaran and Reif 1992)). They call their algorithm ‘Nested Annealing’ (abbreviated as NA). A simpler convergence proof for NA has been given in (Rajasekaran 2000). We describe this proof in this article.

NA employs the following idea. For any optimization problem, we can define a graph. If this graph is known to be ‘small separable’, then NA leads to ‘faster’ convergence. Several other algorithms have also been
designed which exploit the ‘small separability’ of the underlying graphs (see e.g., (Lipton and Tarjan 1980)). For a given arbitrary problem, the ‘separability’ of the underlying graph may not be know. In this article we provide tight bounds on the ‘separability’ of a random graph (as given in (Rajasekaran 2000)). These ‘separability’ bounds are then used to analyze the expected behavior of NA on an arbitrary optimization problem.

2 Preliminaries

SA is a family of randomized algorithms. Classical approaches to introducing randomness in algorithms were based on certain assumptions on the input distribution. Based on these assumptions the average run times of algorithms will be computed. When the assumptions made do not hold, the analysis fails. We could still achieve good performance without making assumptions on input distributions by introducing randomness in the algorithm itself (see e.g., (Rabin 1976, Solovay and Strassen 1977)). An algorithm employing this technique is called a ‘randomized algorithm’.

A randomized algorithm is one which makes coin flips to make certain decisions. Such an algorithm is typically 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 $cf(n)$ on any input of size $n$, with probability $\geq (1 - n^{-\alpha})$.

By an Optimization Problem we mean the problem of computing the minimum value of a given ‘objective function’ or ‘cost function’ $C(.)$ of $n$ parameters $p_1, p_2, \ldots, p_n$, subject to some given constraints.

For any non-negative random variable $X$ with mean $\mu$, Markov’s inequality (see e.g., (Feller 1966)) states that $\operatorname{Prob}(X \geq k\mu) \leq 1/k$, for any $k > 0$.

3 Modeling SA

Here we state the mathematical model to be employed for SA.

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.

\[
\text{procedure } \text{Anneal}(\text{start state}, \text{start temperature});
\]

\[
\text{while (not frozen) do}
\]

\[
\text{while (not steadystate) do}
\]

\[
\begin{align*}
\text{Generate a random neighbor of the current state;}
\end{align*}
\]

\[
\text{Let } \Delta = \text{cost of old state} - \text{cost of the new state;}
\]

\[
\text{Accept the new state with probability } \min\{1, \exp(-\Delta/T)\}, T \text{ being the current temperature;}
\]

\[
\text{Update the temperature; steadystate = false;}
\]

In the above algorithm ‘frozen’ is a boolean variable that takes the value \textit{true} when no improvement in the given cost function has been observed in a ‘long’ time (e.g., during the past few temperatures). The boolean variable ‘steadystate’ becomes \textit{true} when the system attains steady state at the given temperature.

Several schemes (also known as ‘annealing schedules’) have been employed to derive the sequence of temperatures the system goes through. For example, the temperature can be decreased by a constant factor each time. Other annealing schedules can be found, for example, in (Mitra et al. Sept. 1986).

For any given OP, we can define a directed graph \(G(V,E)\) as follows: Nodes in \(G\) are simply the states of the OP and the edges going out of any node (or state) are to the neighbors of this state. SA can be thought of as a random walk on this graph. State transition made at any step is dependent only on the current state. For a given temperature, the transition probabilities from out of any node are fixed. These facts suggest modeling SA as a Markov chain. SA can be modeled as a time homogeneous Markov chain at any given temperature. However, the transition probabilities might change with temperature. Thus the whole of SA can be modeled as a time inhomogeneous Markov chain (Mitra et al. Sept. 1986). 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 \(N(i)\) stand for the set of neighbors of \(i\). Let \(C(i)\) be the cost of state \(i\). Under the assumption that, when the Markov chain is in state \(i\), each one of its \(|N(i)|\) neighbors is equally likely to be generated next, 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 (Rajasekaran 2000).

4 Convergence of SA

SA is said to have converged if the underlying Markov chain had been in a globally optimal state at least once. Another way to define convergence is to require that the probability state vector reach an optimal vector (see e.g. (Mitra et al. Sept. 1986)). Since our goal is to find an optimal solution, our definition of convergence is more appropriate.

Now we describe the convergence proof of (Rajasekaran 2000). Given a cost function \( C(p_1, p_2, \ldots, p_n) \), let \( G(V, E) \) be its state graph. Let \( T \) denote 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.

For any \( i \in V \), \( P_{ij} \) will be at least \( \exp(-\Delta/T) \) if \( j \in N(i) \) (for any temperature). The following facts are needed.

**Fact 4.1** 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 4.1** Let \( X \) be any state in \( V \). 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.** If \( g \) is any globally 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. The 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 \( \left( \frac{1}{d} \exp(-\Delta/T) \right)^q \geq \left( \frac{1}{d} \exp(-\Delta/T) \right)^D \), (under the assumption that each neighbor of a state is equally likely to be generated next).

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

**Theorem 4.1** SA converges in time \( \leq 2k[d \exp(\Delta/T)]^D \), with probability \( \geq (1 - 2^{-k}) \), independent of the start state.

**Proof.** Let \( E = 2[d \exp(\Delta/T)]^D \). We can 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 \( A \) denote the event that \( g \) is not visited during the first \( E \) steps, and \( B \) denote the event that \( g \) is not visited during the next \( (r-1)E \) steps.

Probability that \( g \) is not visited in \( rE \) steps, \( P \), is given by

\[
P = \text{Prob.}[B/A] \times \text{Prob.}[A].\]

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

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

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

\[
P \leq \frac{1}{2} 2^{-(r-1)} = 2^{-r}.\]

\textbf{Corollary 4.1} If \( \Delta, T, \) and \( d \) are assumed to be constants and \( D = \Theta(\log |V|) \), then, \( \text{SA} \) converges in time polynomial in \( |V| \) with probability \( \geq (1 - 2^{-\Omega(|V|)}) \).

**Note.** The above analysis is oblivious to the annealing schedule used.

\( \text{SA} \) is typically used to obtain ‘quasi optimal’ solutions by running it for only a small amount of time. The solutions obtained have been very satisfactory.

The above proof assumes that each neighbor of a state is equally likely to be generated next. This proof can be extended to the case where this assumption is invalid as follows.

In some applications, it may be necessary to generate certain neighbors with higher probability. Mitra et al. (Sept. 1986) assume that the probability of generating state \( j \) from state \( i \) is \( \frac{g(i, j)}{g(i)} \), 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 4.2** SA algorithm converges in time \( \leq 2k \left[ \frac{1}{T} \exp(\Delta/T) \right]^D \) with probability \( \geq (1 - 2^{-k}) \), no matter what the start state is.

## 5 Nested Annealing

Rajasekaran and Reif (1992) have shown that if the cost function being optimized is 'small separable', then SA can be modified to get faster convergence. The modified algorithm is called 'Nested Annealing' (NA). A simple proof of convergence for NA (Rajasekaran 2000) is described below.

### 5.1 Some Definitions

A graph \( G(V, E) \) with \( n \) nodes is said to be '\( s(n) \)-separable' if there exist constants \( \alpha < 1, \beta > 0 \) such that \( V \) can be partitioned into three subsets \( V_1, S, V_2 \) such that no vertex in \( V_1 \) is adjacent to any vertex in \( V_2 \), both \( |V_1| \) and \( |V_2| \) are less than \( \alpha n \), and \( |S| \) is less than \( \beta s(n) \). Also, 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 can define the 'separability' of \( C \) as follows. Write \( C = C_1 + C_2 + \ldots + C_k \), 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.** Consider the problem of CNF-satisfiability where the input is a boolean formula in conjunctive normal form, \( F \), on \( n \) variables. The problem is to decide if there is an assignment to the variables that
makes $F$ 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.

5.2 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 explained above.

Assume that each parameter is binary. The minimum value of $C$ can be found 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.

If we employ 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.

NA can be specified as follows.
procedure Nested Annealing($G_C(V,E)$);

Find a separator set $S$ for $G_C$. Let $V_1, S, \text{ and } V_2$ 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 NA for finding a globally optimal solution to a given 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 the worst case run time of SA on an OP with $N$ parameters be $2^M(n)$. The number of states visited by NA at the top level 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 $\alpha n$ parameters each, accounting for a total expected cost of $\leq 2^T(\alpha n)$.

As a result we have,

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

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 5.1 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 5.1 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 (Rajasekaran and Reif 1992).

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 5.2 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 \( 2^{O(M(n))} \).

6 Separability of Random Graphs

Algorithms that exploit the separability of the underlying graphs presuppose that a separator is known for the given graph. Algorithms have been devised for finding separators of restricted classes of graphs. For instance efficient algorithms exist for computing the separator set of planar graphs (Lipton and Tarjan 1980). No such algorithms are known for general graphs. In fact, deciding if a given graph is \( s(n) \)-separable is NP-hard. In this section we state tight bounds that are known for the separability of random graphs. These bounds can be used to study the expected behavior of NA on arbitrary OPs.

6.1 A Modification of NA

The separability results of (Rajasekaran 2000) indicate 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 \( an \) 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. We can modify NA 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 this modified NA together with the procedure for finding a separator in a random graph (given in (Rajasekaran 2000)) can be used to solve the OP. In this section we state expected bounds on the convergence time of this modified NA on an arbitrary OP. The expected convergence of the original NA will be nearly the same since a random graph is not ‘small separable’.

### 6.2 The Separator Theorems

There are two popular models of random graphs (Bollobás 1985). 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 \( G_p \).

We assume the second model. 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 \( X \).

The following separability results are proven in (Rajasekaran 2000).

**Theorem 6.1** Let \( p > \frac{\ln n}{n} \). 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.

**Theorem 6.2** 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 \( G_p \).

**Theorem 6.3** Let \( \frac{1}{p} \to \infty \). For almost every \( G_p(V,E) \), the following holds. If \( X \) is any set of \( \delta n \) nodes (for any \( \delta \)), then the cardinality of \( T_X = \{ y \in V - X : \Gamma(y) \cap X = \Phi \} \) is at the most \( \left\lfloor \left( \frac{2-\delta}{\delta \ln \pi} \right) + \epsilon \right\rfloor \frac{1}{p} \), for any constant \( \epsilon > 0 \).

**Corollary 6.1** If \( X \) is any set of \( \delta n \) nodes (where \( \delta \) is a constant), then, \( T_X \) has cardinality \( O(1/p) \).

### 7 Conclusions

In this article we have described SA and NA which are randomized algorithms that can be used to solve any combinatorial optimization problem. NA is a variation of SA that exploits the separability of the underlying
graphs. We have summarized convergence results known for SA and NA. Separability results for random graphs have been stated and applied to study the expected behavior of SA and NA.

References


