PROBABILISTIC HILL CLIMBING ALGORITHMS: PROPERTIES AND APPLICATIONS

by

Fabio I. Romeo

Memorandum No. UCB/ERL M86/97

6 December 1986

PROBABILISTIC HILL CLIMBING ALGORITHMS: PROPERTIES AND APPLICATIONS

by

Fabio I. Romeo

Memorandum No. UCB/ERL M86/97

6 December 1986

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

PROBABILISTIC HILL CLIMBING ALGORITHMS: PROPERTIES AND APPLICATIONS

by

Fabio I. Romeo

Memorandum No. UCB/ERL M86/97

6 December 1986

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Abstract

Simulated annealing proposed by Kirckpatrick et al. has proven to be an effective technique to solve general combinatorial optimization problems. Its derivation was based heavily on the analogy between combinatorial optimization problems and the annealing process in physics.

A mathematical model of the operations of Simulated Annealing is needed to understand the essential features which guarantee the algorithm to perform efficiently and to improve the speed of execution. Markov chains are proposed as mathematical models of the simulated annealing algorithm. Using these models, it has been possible to prove that, under certain assumptions on the rules used by the algorithm to generate the configurations of the problem and on the time spent at each temperature, the simulated annealing algorithm generates a global optimum solution with probability one.

This result has made possible the definition of a general class of algorithms with the same statistical properties: the class of probabilistic hill-climbing methods. The mathematical properties of this class are presented and rules on the selection of annealing schedules are obtained from these properties.

Acknowledgements

My research advisor, Prof. Alberto Sangiovanni Vincentelli is greatly acknowledged for having introduced me to the study of the probabilistic optimization algorithms. Through the development of this work he was always an inexhaustible source of ideas and help. I wish to thank Carl Sechen for collaboration in obtaining the experimental results and for many lively arguments on the practical aspects of simulated annealing. Early discussions on simulated annealing and its analysis with Professor Karp are acknowledged. Professors Sastry and Walrand have been very helpful discussing various theoretical aspects of stochastic processes. The collaboration with Dr. Huang was of great help in developing the adaptive annealing schedule described in this thesis.

This research has been sponsored in part by SRC-82-11-008, the Italian National Research Council and Honeywell Information Systems Italy.

TABLE OF CONTENTS

CHAPTER 1

Introduction

The problem of finding the optimum of a function defined on a countable solution space, referred as a *combinatorial optimization problem*, is of crucial importance in the solution of many problems arising in a great variety of different applications ranging from engineering to economics. More formally a combinatorial optimization problem is defined as follows: Given a problem and the space of admissible solutions S, each of them with a given value of a cost function, find the best solution with respect to a given cost function. Well known examples of combinatorial optimization problem that arise quite often are *Traveling Salesman* [LLK85], *Bin Packing*, *Graph Coloring*, *Graph Partitioning*. [GaJ79].

Combinatorial optimization problems are classified according to the computational complexity of the best known algorithm used to determine an optimal solution. Such a classification partitions combinatorial optimization problems into two classes. The first one consists of problems for which a deterministic algorithm is known whose worst-case complexity is polynomial in the number of variables of the problem. The class is called P. Problems for which a deterministic algorithm with polynomial worst-case complexity is not known, are members of the second class called NP. While it is clear that $P \subseteq NP$ no formal proof of $P \subset NP$ has been found. There exists a subclass of NP called NP-complete class with the property that every element in the class can be reduced to any other member by means of a polynomial time transformation [Coo71]. The existence of the NP-complete class is of remarkable importance since the discovery of a deterministic algorithm with polynomial worst-case complexity that solves one of the problem

in the class will solve in polynomial time all the other problems in the class. Unfortunately the great majority of problems that are of interest in practical applications belongs to the NP-complete class [Kar72] [GaJ79].

The methods used to solve combinatorial optimization problems can be subdivided into exact and heuristic. Exact methods are those that are guaranteed to produce a solution with the best possible value of the cost function. Branch and bound, and integer programming techniques are examples of exact methods. Because of the NP nature of many of the interesting problems, the computation time necessary to solve them exactly grows fast and the size of problems of practical interest is usually large enough to require prohibitive computing times.

Heuristic methods are in general much faster than exact ones but are not guaranteed to obtain the best possible solution. In general only approximations of the optimal solution are obtained. The difference in cost between the obtained solution and the best one establishes the quality of the heuristic method. Heuristics are of two different types: Methods that compute the solution constructively starting from raw data and methods that iteratively improve an existing solution. In practice both the methods are present in a real heuristic algorithm.

While constructive methods are obviously closely related to characteristics of the problem to be solved, iterative improvement algorithms exhibit the following common structure: Starting from an initial configuration say j_0, $j_0 \in S$, a sequence of configurations is generated until a satisfactory one is found. The rules according to which a new configuration is generated and the algorithm terminates, specify the algorithm. Often the search terminates with a local minimum, i.e. with a configuration $\hat{j}$ such that if we denote by $c(j)$ the cost of j and by $S(j)$ the set of configurations that can be generated from j by the algorithm in one step, $c(\hat{j}) \leqslant c(j), \forall j \in S(\hat{j})$. This is often due to the fact that heuristic algo-

rithms have a local view of the problem. Often these algorithms are called "greedy", since the strategy they implement is to select moves which reduce "maximally" the cost are accepted.

To avoid this behavior, randomizing algorithms (e.g. [Sch80]) can be devised which generate the next configuration randomly. The configuration is recorded as a new temporary solution if its cost is lower than the present temporary solution. The algorithm terminates after a certain number of moves has been carried out. Randomizing algorithms perform well if the number of optimal solutions is fairly high, since the probability of stopping at an optimum is proportional to the ratio between the number of optimal configurations and the number of total configurations. Note that randomizing algorithms can "climb hills", i.e., moves that generate configurations of higher cost than the present one are accepted.

Simulated annealing as proposed by Kirckpatrick et al. [KGV83], allows "hill climbing" moves but these moves are accepted according to a certain criterion which takes the cost into consideration and not blindly as randomizing algorithms. The controlling mechanism is based on the observation that combinatorial optimization problems with a large configuration space exhibit properties similar to physical processes with many degrees of freedom.

In particular, bringing a fluid into a low energy state such as growing a crystal, has been considered in [KGV83] similar to the process of finding an optimum solution of a combinatorial optimization problem. Annealing is a well-known process to grow crystals. It consists in melting the fluid and then lowering the temperature slowly until the crystal is formed. The rate of decrease of temperature has to be very low, especially around the freezing temperature, to avoid the formation of glasses, i.e. areas in which the crystal lattice is not perfectly regular. The Metropolis Monte Carlo method [MRR53], [Bin78] can be used to simulate the

annealing process physics. The same procedure has been proposed as an effective method for finding global minima of combinatorial optimization problems. This method when applied to combinatorial optimization generates moves randomly and checks whether the cost of the new configuration satisfies an acceptance criterion based on a parameter, T, sometimes called "temperature" in analogy with the physical case. If the cost decreases, the move is accepted. If the cost increases, then a random number between zero and one is generated and compared with

$$f_T(\Delta c_{ij}) = \exp(\frac{-\Delta c_{ij}}{T})$$

where Δc_{ij} is the change in cost obtained by moving from configuration i to j and T is temperature, the controlling parameter. If the random number is larger than $f_T(\Delta c_{ij})$, the move is accepted, otherwise the move is discarded. Note that the higher the value of T is, the more likely it is that a "hill climbing" move is accepted. Note also that "hill climbing" moves are less and less probable as the T is decreased. A certain number of moves are generated and checked before a decrease in T is allowed. The initial value of T, the number of moves generated at each fixed value of T and the rate of decrease of T are all important parameters that affect the speed of the algorithm and the quality of the final configuration. Experimental results [KGV83], [VeK83], [SeS84], [AJM84] show that simulated annealing produces very good results when compared to other techniques for the solution of combinatorial optimization problems such as those arising from the layout of integrated circuits, at the expense of large computer time (a 1,500 standard cell placement problem can take as much as 24 hours of a VAX 11/780 [SeS84]).

A mathematical analysis of the algorithm is very important to understand the essential features which make the algorithm work well and to suggest techniques for controlling its operation. Markov chains [Kar73], [Fel70], [Doo51],

[Fre71] can be used as a mathematical model of simulated annealing. We proved that under certain assumptions on the mechanism used to generate the configurations and on the function f used to determine whether the new configuration is accepted, simulated annealing produces asymptotically the optimum solution of combinatorial optimization problems with probability 1. A similar proof was independently proposed by Lundy and Mees [LuM86]. The proof has underlined the essential properties of the algorithm, so that we have been able to derive a class of "probabilistic hill-climbing algorithms" that have the same asymptotic properties of simulated annealing.

The report is organized as follows. In Chapter 2, the class of Probabilistic Hill-Climbing (PHC) algorithms is formally defined and its mathematical representation in terms of Markov chains introduced. In Chapter 3, basic definitions and theorems specifying the properties of Markov chains relevant to the analysis of PHC algorithms are reviewed. In Chapter 4, the assumptions on the parameters of the PHC algorithms needed to guarantee the optimality properties are introduced and the convergence theorems are proved. In Chapter 5, additional results which give insight on how to select the the number of moves to be attempted at each value of T by the PHC algorithms are presented. Chapter 6 and 7 are devoted to the description of a new dynamical strategy to control the annealing process and to its applications to the solution of two combinatorial optimization problems respectively. In Chapter 8, new research directions and concluding remarks are given.

CHAPTER 2

Probabilistic Hill Climbing Algorithms

Given a combinatorial optimization problem specified by a finite set of configurations or states S and by a cost function $c : S \rightarrow \mathbb{R}+$ defined on all the states $j \in S$, Probabilistic Hill Climbing (PHC) algorithms are characterized by a rule to generate randomly a new state or configuration with a certain probability, and by a random acceptance rule according to which the new configuration is accepted or rejected. A parameter T controls the acceptance rule. We assume that $T \geqslant 0$ and that an updating rule generates a monotonically decreasing sequence $\{T_m\}, m = 1,2...$ with limit zero. PHC algorithms define a random variable X which takes values on the set of states S generated and accepted by the algorithm. Their structure is shown below.

PHC Algorithm Structure (j_0, T_0)

```
{
/* Given an initial state j0 and an initial value for the parameter T, T0 . */
T = T0;
X = j0;
while( "stopping criterion" is not satisfied )
     {
     while( "inner loop criterion" is not satisfied )
          {
          j = generate (X)
          if( accept( c(j), c(X), T )
          }
          T = update (T )
     }
}
```

The acceptance of a new state j is determined by **accept**, whose structure is shown below.

```
accept( c(j) , c(i) , T )

    {

    /*
     returns 1 if the cost variation passes a test.
     T is the control parameter.
    */

    Δc_ij = c(j) - c(i) ;
    y = f_T(Δc_ij ) ;
    r = random (0,1) ;

    /*
     random is a function which returns a pseudo random number uniformly
        distributed on the interval [ 0 , 1 ] .
    */

    if (r < y)
        return(1) ;
    else
        return(0) ;
    }
```

The acceptance strategy is represented by the function $f_T : \mathbb{R} \rightarrow (0,1]$. f_T is a family of functions of the argument Δc_{ij}. The shape of f_T is controlled by adjusting the parameter T.

Remark 2.1. Simulated annealing [KGV83], belongs to the PHC class. In simulated annealing, the control parameter T, called *temperature*, is updated by means of the following law

$$T_{m+1} = \alpha T_m \qquad 0 \leqslant \alpha \leqslant 1 \tag{2.1}$$

and hence it satisfies the property that every element of the sequence $\{ T_m \}$ $m = 1,2,\ldots$ is such that $T_i \geqslant 0$ and that the sequence of updates is monotonically decreasing and converges to zero. The acceptance function for the

simulated annealing is

$$f_T(\Delta c_{ij}) = e^{-\frac{c(j)-c(i)}{T}} . \qquad (2.2.a)$$

for $\Delta c_{ij} > 0$ and

$$f_T(\Delta c_{ij}) = 1 \qquad (2.2.b)$$

otherwise. Hence, it takes values on the interval (0,1].

■

PHC algorithms applied to a combinatorial optimization problem can be represented by Markov chains [Kar73], [Fel70], [Fre71]. We derive this model on a simple example: a linear placement problem.

Suppose that 3 interconnected modules $\{a, b, c\}$ have to be placed on a mono-dimensional grid, so that the global length of interconnections is minimized. The state space S consists of 6 configurations, all the possible placements of the three modules (3!), i.e., $S = \{1, \ldots, 6\}$

$$1 = \{a, b, c\}$$
$$2 = \{c, a, b\}$$
$$3 = \{b, c, a\}$$
$$4 = \{a, c, b\}$$
$$5 = \{c, b, a\}$$
$$6 = \{b, a, c\}.$$

We assume that the generation of new configurations is done by a PHC algorithm, applied to this problem by exchanging the positions of two elements. In this case, the set of states reachable from the first state is equal to $\{4,5,6\}$. The set of neighbors for the other states can be determined easily. Note that for all $i \in S$, the cardinality of each set of neighbors is 3.

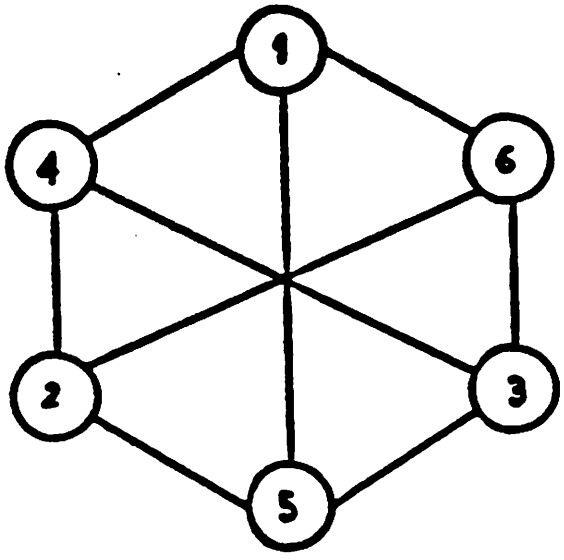

Fig. 2.1. Schematic representation of the generating rule.

The graph shown in Figure 2.1 is a schematic representation of the generating rule. Each node of the graph represents one of the possible placements; there is an edge (i, j) in the graph if j can be obtained from i by interchanging the positions of two modules. For each placement i, we denote by $S(i) \subseteq S$ the set of all the states $j \in S$ such that there is an edge (i, j) in the graph.

In general we define $S(i)$ to be the set of all the states that can be generated by a PHC algorithm with the generating rule introduced before in one step starting from i. $S(i)$ is defined to be the set of neighbors of i under the selected generation rule. Note that the generation rule induces a metric on the state space S and

turns it into a metric space.

Up to this point nothing has bees said about the cost associated to each one of the configurations. Assume now, for the sake of simplicity, that in the previous example the configurations have been numbered so that

$$c(i) \leqslant c(j) \qquad \forall i \leqslant j \, , i \, , j = 1,....6 \, . \tag{2.3}$$

We can now append to the edges of the graph the probability that the transitions between two configurations occurs when a PHC algorithm is applied. For example, if simulated annealing is applied, there are certain transitions which are independent of T and Δc_{ij}, i.e., the transitions corresponding to a decrease in cost, and others which are dependent on T and Δc_{ij}, e.g., the "hill climbing" transitions. If the states of the above example are numbered such that (2.3) holds the graph of Figure 2.1 becomes the graph represented in Figure 2.2. The dashed edges represent transitions which are dependent on T and Δc_{ij}, while solid edges represent independent transitions.

The application of a PHC algorithm to a combinatorial optimization problem can be represented by a graph. The nodes of the graph are the possible configurations, the edges represent configurations which can be obtained by the generation rule of the PHC algorithm. The edge labels represent the probability that the corresponding transition is generated and accepted by the algorithm.

The probability that the configuration selected by a PHC algorithm at the $(k+1)-st$ iteration be j given that at the $k-th$ iteration was i, is defined by

$$Prob\{ X_{k+1} = j \mid X_k = i \} \triangleq P_{ij}(T) \, , \tag{2.4}$$

where

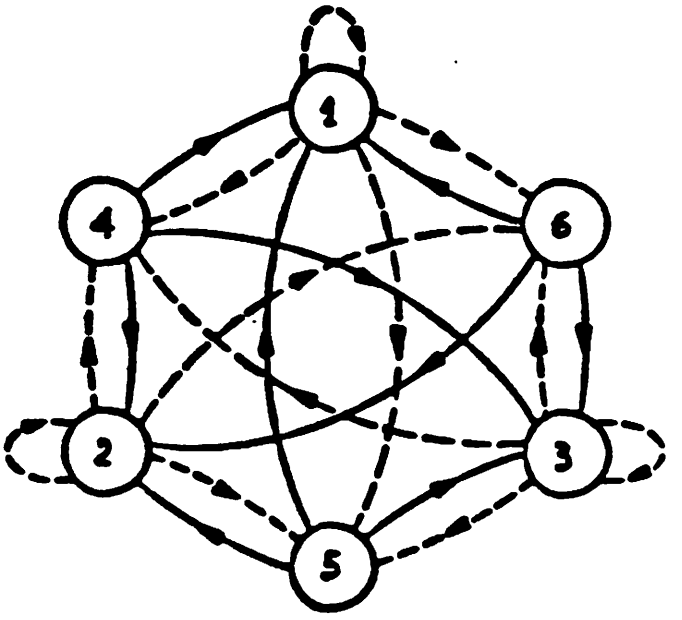

Fig. 2.2. Transition graph. The dashed edges represent transitions dependent both on T and Δc_{ij}. The solid edges represent independent transitions

$$P_{ij}(T) = G_{ij}(T) \bullet f_T(\Delta c_{ij}) \quad \forall j \in S(i) \ . \tag{2.5}$$

where $G_{ij}(T)$ is the probability of *generating* state j being in state i, possibly dependent on the parameter T. Δc_{ij} is defined as follows

$$\Delta c_{ij} = c(j) - c(i) \ .$$

and X_k is the value taken on by the random variable X, representing the generic solution given by the algorithm, at the k-th iteration. Since $G_{ij}(T)$ must be a probability, the following relation must hold

$$\sum_{j \in S(i)} G_{ij}(T) = 1 \tag{2.6}$$

moreover by definition $G_{ij}(T) = 0$ for the states that are not in $S(i)$.

Eq. (2.5) is the product of two different terms : the first term, i.e. $G_{ij}(T)$, is the probability that j is generated by the algorithm, the second term, i.e. $f_T(\Delta c_{ij})$ is the probability that the new configuration is accepted. An example of $G_{ij}(T)$ is given by

$$G_{ij}(T) = \begin{cases} 1/|S(i)| & \forall j \in S(i) \\ 0 & \text{otherwise} \end{cases}$$

where the probability of generating all the states that can be generated is uniform and independent of T. In our example, we assume that the generation probability is uniform and independent of T, and since $|S(i)| = 3, \forall i \in S$,

$$G_{ij}(T) = \begin{cases} 1/3 & \forall j \in S(i) \\ 0 & \forall j \notin S(i) \end{cases}$$

We assume also that f is given by (2.2).

Note that since f is not, in general, identically equal to one, there is a finite probability that the algorithm will remain in configuration i. The following equation determines this probability:

$$P_{ii}(T) = 1 - \sum_{j \in S(i)} P_{ij}(T). \tag{2.7}$$

The stochastic process represented by the evolution of the random variable X_k produced by PHC algorithms is a *Markov process*. In fact, eqs. (2.4) and (2.5) imply that, given the value X_k, the value of X_{k+1} depends only on the value of X_k, i.e., the probability of any particular future behavior of the process, when its present state is known exactly, is not altered by additional knowledge concerning

its past behavior. Furthermore it is easily seen that if we extend eq. (2.4) to describe an n-*step* transition of the PHC algorithm as follows

$$Prob\{ X_{k+n} = j \mid X_k = i \} \triangleq P_{ij}^{(n)}(T) \quad .$$

the Chapman-Kolmogorov equation [Fel70]

$$P_{ij}^{(n)} = \sum_{k=1}^{k=n} P_{ik}^{(m)} P_{kj}^{(n-m)} \quad \forall m : 0 \leqslant m \leqslant n \quad .$$

is automatically satisfied by the matrix multiplication rules.

Finally since the configuration space of combinatorial optimization problems is a countable and in general finite set, the process is a *discrete time Markov process with a finite state space*. Finite time Markov processes are named *Markov Chains*.

CHAPTER 3

Basic Definitions and Results on Markov Chains

In this section, a few basic definitions and theorems on Markov chains which are relevant to the discussion of PHC algorithms are reviewed. All the theorems are presented without proofs since they can be found in any of the standard probability theory texts, e.g. [Kar73], [Fel70], [Fre71].

Definition 3.1. State j is said to be *accessible* from state i if for some integer $n \geq 0$, $P_{ij}^{(n)} > 0$. Two states i and j, accessible to each other, are said to *communicate*.

■

The relation induced by this definition is an equivalence relation. The equivalence classes induced by this relation consist of all those states for which there exist a probability greater than zero to go from one state to the other in both directions in a finite number of steps.

Definition 3.2. A Markov chain is said to be *irreducible* if the equivalence relation induces a unique class.

■

Example 3.1. The Markov chain represented by the following probability transition matrix

$$P = \begin{vmatrix} 0 & P_2 \\ P_1 & 0 \end{vmatrix} ,$$

where the elements of P_i, $i = 1,2$ are all non zero, is irreducible. The Markov chain represented by

$$P = \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix} .$$

where P_1 and P_2 are two matrices with elements not all zero, is not irreducible.

■

Definition 3.3. The period of the state i is said to be the greatest common divisor of all integers $n \geq 1$ such that

$$P_{ii}^{(n)} > 0 .$$

■

Theorem 3.1 If the Markov chain is irreducible and there exists a state, say i, such that

$$P_{ii} > 0 .$$

then all states have period one.

■

Definition 3.4. A Markov chain in which each state has period one is said to be *aperiodic*.

■

Example 3.2. The Markov chain represented by the following probability transition matrix

$$P = \begin{bmatrix} 0 & P_2 \\ P_1 & 0 \end{bmatrix} .$$

where P_1 and P_2 are two matrices which elements are not all zero, is periodic with period two.

■

It is easy to show that periodicity is a *class property* i.e., all the states in an equivalence class have the same period.

Definition 3.5 Let $h_{ii}^{(n)}$ be the probability that starting from state i, the first return to state i occurs at the n-th transition, i.e.,

$$h_{ii}^{(n)} = Prob\{X_n = i, X_j \neq i, j = 1,2,....n-1, | X_0 = i\}.$$

h_{ii} is defined by means of the following recursion

$$P_{ii}^{(n)} - \sum_{k=0}^{n} h_{ii}^{(n-k)} P_{ii}^{(k)} = \begin{cases} 1 & \text{if } n=0 \\ 0 & \text{if } n>0 \end{cases} . \qquad (3.1)$$

A state i is recurrent if $\sum_{n=1}^{\infty} h_{ii}^{(n)} = 1$.

■

This definition says that a state i is recurrent if , starting from state i, the probability of returning to state i after some finite length of time is one.

Example 3.3. The following probability transition matrix

$$P_{ij} = \begin{cases} q & \text{if } i = j+1 \\ p & \text{if } i = j-1 \\ 0 & \text{otherwise} \end{cases} \qquad (3.2)$$

represents the Markov process known as one-dimensional random walk on the positive and negative integers where, at each transition, a particle moves with probability q one unit to the right and with probability p one unit to the left with ($p + q = 1$). If the process starts from the origin, if $p \neq q$, there is a non zero probability that a particle initially at origin will drift to $+\infty$ if $q > p$ ($-\infty$ in the other case) without ever coming back to the origin. Hence the origin is not recurrent.

The following condition

$$p = q = \frac{1}{2} \quad . \tag{3.3}$$

is necessary to ensure the recurrence of the Markov chain determined by eq. (3.2). In [Kar73] a formal proof of condition (3.3) is given.

■

Recurrence as periodicity is a class property, i.e., all the states in an equivalence class are either recurrent or non recurrent.

The properties that have been introduced above define a large class of Markov chains for which an ergodic theory has been developed. The two main results of this theory are recalled below.

Theorem 3.2. Let i be the initial state and let $P_{ii}^{(0)} \triangleq 1$. If a Markov chain is irreducible, aperiodic and recurrent, then

a) the following limit exists

$$\lim_{n \to \infty} P_{ii}^{(n)} = \frac{1}{\sum_{n=0}^{\infty} n \; h_{ii}^{(n)}} \quad ,$$

where $h_{ii}^{(n)}$ is defined recursively by eq. (3.1)

b) For each j,

$$\lim_{n \to \infty} P_{ji}^{(n)} = \lim_{n \to \infty} P_{ii}^{(n)} = \pi_i \, . \tag{3.4}$$

■

An intuitive explanation of Theorem 3.2 is as follows. If n is large enough, $P_{ji}^{(n)}$, the probability of being at the n-th iteration in state i, starting from state j, depends only on the state itself and is totally independent on the initial state

j. Note that π_i defined in Theorem 3.2 is always larger than or equal to zero. If it is larger than zero, then the following important result holds.

Theorem 3.3. If π_i of eq. (3.4) is greater than zero $\forall i$ and the Markov chain is recurrent, irreducible and aperiodic then

$$\lim_{n \to \infty} P_{ii}^{(n)} = \pi_i = \sum_{j=1}^{|S|} \pi_j P_{ji} \quad .$$

and the π_i's are uniquely determined by the following set of equations

$$\sum_{i=1}^{|S|} \pi_i = 1 \quad . \tag{3.5.a}$$

$$\sum_{i=1}^{|S|} \pi_i P_{ij} = \pi_j \quad . \tag{3.5.b}$$

$$\pi_i \geq 0 \qquad \forall i \quad . \tag{3.5.c}$$

■

The set $\{\pi_i, i = 1,....|S|\}$ determined by Theorem 3.2 is called the *stationary probability distribution* of the Markov chain and $\pi' = [\pi_1, \pi_2, \cdots, \pi_n]$ is called the stationary probability distribution vector.

The stationary probability distribution is very important since it completely characterizes the asymptotic behavior of a Markov chain.

The last result we report is related to the rate of convergence of an arbitrary initial probability distribution vector $\mathbf{p}^{(0)}$ to the asymptotic probability distribution vector π.

Theorem 3.4. (Frobenius-Perron Theorem) Let $\mathbf{P}$ be the transition probability matrix of a finite irreducible Markov chain. Let π be the stationary probability distribution for the Markov Chain then:

a) $\lambda_0 = 1$ is an eigenvalue of $\mathbf{P}$ with geometric multiplicity 1;

b) let $\lambda_i \quad i = 0,1,2,....n-1$ be the eigenvalues of $\mathbf{P}$. The following relation holds

$$| \lambda_i | < \lambda_0 \qquad i = 1,2,....n-1 \quad ;$$

c) $\vec{\pi}$ and 1 are the left and right eigenvectors of $\mathbf{P}$ corresponding to λ_0. 1 is a vector with all the entries equal to 1;

Proof. Many different proofs of the Frobenius-Perron Theorem are available in the literature on Markov chains and non-negative matrices. One of these can be found for example on [Sen80].

■

From the Frobenius-Perron Theorem follows immediately

Corollary 3.1 Let $\mathbf{p}^{(0)}$ be the initial probability distribution vector of the states in the Markov chain and let $\mathbf{p}^{(n)}$ be the probability distribution vector of the states after n iterations have been performed. If π is the asymptotic probability distribution vector then $\|\pi - \mathbf{p}^{(n)}\|$ goes to zero at least as fast as $o(| \lambda_j^n |)$ as $n \rightarrow \infty$, where λ_j is the largest eigenvalue of $\mathbf{P}$ such that

$$| \lambda_i | < 1 \qquad i = 1,2,....n-1 \quad .$$

CHAPTER 4

Asymptotic Properties of Probabilistic Hill Climbing Algorithms

Results quoted in Section 3 cannot be applied directly to the Markov chain representing the stochastic process generated by PHC algorithms. In fact, these results are in general valid for *stationary* transition probability matrices, i.e., for transition probability matrices that are independent of time. Note that the transition probabilities defined in Section 2 depend on the parameter T which is updated during the evolution of the algorithms and hence are dependent on time. However, when the parameter is kept constant, i.e., in the inner loop of PHC algorithms, the transition probabilities are constant and the associated Markov chain stationary.

Our strategy to prove the properties of PHC algorithms is to determine first which conditions PHC algorithms must satisfy so that a stationary probability distribution exists for each given value of the parameter T. Then we will introduce a function $\pi_i(T)$, defined on the set of configurations, with the property that, when T approaches zero, $\pi_i(T) \neq 0$ only for those configurations that are global optima for the combinatorial optimization problem. Finally conditions on the acceptance function f will be determined such that $\pi_i(T)$ is the stationary probability distribution of the Markov chain describing the PHC Algorithm. If we can prove that, given a function $\pi_i(T)$ with the above mentioned properties and a suitable generation function $G_{ij}(T)$, it is possible to determine an acceptance function f with the features outlined in section 2, then a PHC algorithm is obtained which generates with probability one, asymptotically, a global optimal solution for the combinatorial optimization problem.

The first assumption on the PHC algorithms is related to the generation rule and the acceptance rule. The rules must be such that for all T different from zero, the Markov chain induced by the algorithm is irreducible. This means that for each pair of configurations, say i, j, there must be integers $0 \leqslant m, n < \infty$, so that $P_{ij}^{(n)} \neq 0$ and $P_{ji}^{(m)} \neq 0$. In other words, the graph obtained by representing the generation rule with directed edges as in Section 2, must be strongly connected. In addition, we must be sure that the acceptance rule does not eliminate edges of the graph by assigning them probability zero which cause the labeled graph obtained by removing the edges with zero weight not to be strongly connected. Note that the acceptance rule specified by simulated annealing assigns a non zero probability to all the edges of the graph corresponding to the generation rule and hence, for this PHC algorithm we only need to verify that the generation rule produces a strongly connected graph in the configuration space.

The next condition is related to the aperiodicity of the Markov chain.

Proposition 4.1. Let the Markov chain corresponding to a PHC algorithm be irreducible for all $T \geqslant 0$. If the acceptance function f of the PHC algorithm is such that there exists at least a pair of states i and j for which

$$0 < f_T(\Delta c_{ij}) < 1, \qquad \forall T > 0, \tag{4.1}$$

then the Markov chain is aperiodic for all $T > 0$.

Proof. If (4.1) holds, then according to (2.4), $P_{ii}(T) > 0, \forall T > 0$ and the proof follows from Theorem 3.1 and the irreducibility of the Markov chain.

■

The condition of Proposition 4.1 is always satisfied by simulated annealing, since there is at least one state for which (4.1) holds: the global optimum.

The acceptance functions which satisfy the condition of Proposition 4.1 and which take values in the set (0,1] are said to be *admissible*.

Proposition 4.2. Let S, the configuration space of the optimization problem, be finite and the Markov chain associated to a PHC algorithm be irreducible $\forall T > 0$, then the Markov chain is recurrent $\forall T > 0$.

Proof. Since the state space of the Markov chain is finite, then after a number of steps greater than $|S|$ at least a state of the Markov chain has been visited twice. By definition, that state is recurrent. Since recurrence is a class property and the Markov chain is irreducible by hypothesis, the Markov chain is recurrent.

■

According to Theorem 3.3, the Markov chain associates with a PHC algorithm which satisfies the conditions of Propositions 4.1 and 4.2, has a stationary probability distribution.

Now we look for a form of the stationary probability distribution which, when T goes to zero, is different from zero only in global minima of c.

To this end, we have to find under which conditions a stationary probability distribution $\pi_i(T)$ is different from zero, as T goes to zero, only if the i-th configuration is the global optimal solution.

Theorem 4.1. Let $\pi_i(T)$, be a function that $\forall i \in S$ maps $\mathbb{R}+$ into $(0\ ,\ 1/\ |\ S\ |\ ]$. $\pi_i(T)$ is defined by

$$\pi_i(T) = \Lambda(T)\, g(c(i),T) \tag{4.2.a}$$

where $\Lambda(T)$ is a normalizing factor such that

$$\sum_{i \in S} \pi_i(T) = 1 \tag{4.2.b}$$

and g is such that

$$g(c,T) > 0 \qquad \forall T > 0, \forall c. \tag{4.3.a}$$

$$\lim_{T \downarrow 0} g(c,T) = \begin{cases} 0 & \text{if } c \geq 0 \\ \infty & \text{if } c < 0 \end{cases} \tag{4.3.b}$$

$$\frac{g(c_1,T)}{g(c_2,T)} = g(c_1 - c_2, T) \quad . \tag{4.3.c}$$

Then

$$\lim_{T \downarrow 0} \pi_i(T) = \begin{cases} 1/|M| & \text{if } i \in M \\ 0 & \text{if } i \in S \cap M^c \end{cases} \tag{4.4.a}$$

where the set M is defined by

$$M = \{ i \mid c(i) \leq c(j) \ , \forall \ j \in S \} \quad . \tag{4.4.b}$$

■

Proof. The proof of (4.4) is straightforward because of the properties of the function g. In fact $\forall i \in M$, by (4.3.c), eqs. (4.2) can be rewritten as

$$\pi_i(T) = \frac{1}{|M| + \sum_{j \in (S \cap M^c)} g(c(j) - c(i), T)} \tag{4.5}$$

but since

$$c(j) - c(i) > 0, \quad \forall i \in M \ , \forall j \in (S \cap M^c)$$

from (4.3.b) follows

$$\lim_{T \downarrow 0} \pi_i(T) = \frac{1}{|M|} \qquad \forall i \in M \quad .$$

If the same reasoning is applied for an $i \in (S \cap M^c)$ then at least one element of the summation in (4.5) goes to ∞ as T goes to 0. This completes the proof of the Theorem.

■

Now, we have to specify under which conditions on f and G a $\pi_i(T)$ of the form described above is indeed the stationary probability function of the Markov chain associated to the PHC algorithm.

Theorem 4.2. Let $S'(j)$ be defined by

$$S'(j) = \{i : j \in S(i)\},$$

if f is admissible and $\forall\, j$

$$\sum_{i \in S'(j)} g(c(i),T)\, G_{ij}(T)\, f_T(\Delta c_{ij}) = \tag{4.6}$$

$$= g(c(j),T) \sum_{i \in S(j)} G_{ji}(T)\, f_T(\Delta c_{ji})$$

then $\pi_i(T)$ defined by eqs. (4.2) is the stationary probability distribution of the Markov chain whose one-step transition probability is given by eqs. (2.4-2.7).

Proof. The proof of Theorem is carried out by verifying that g , G and f satisfy the conditions of Theorem 3.3. In view of the assumptions made on function G and of Propositions 4.1 and 4.2, the Markov chain defined by eq. (2.4, 2.7) is irreducible, aperiodic and positive recurrent. It is now immediate to see that $\pi_i(T)$ defined by eq. (4.2.a) satisfies eq. (3.5.a) because of (4.2.b) and (3.5.c) because of (4.3.a). Finally it takes just a little thinking to see that (3.5.b) is satisfied automatically once f is chosen as specified by (4.6).

■

Theorem 4.2 is important since it suggest a way to construct a PHC algorithm with guaranteed convergence properties. In fact one first selects a function $\pi_i(T)$ that ensures the convergence to the global optima (Theorem 4.1) and then selects an acceptance function f and a generation rule G such that Propositions

4.1, 4.2 and Theorem 4.2 are satisfied.

Up to now we did not place any assumptions on $G_{ij}(T)$ and we have obtained a general result. We now assume that the rule to generate new states is such that the existence of $G_{ij}(T)$ implies the existence of $G_{ji}(T)$. Under this assumptions we can prove the following

Corollary 4.1. If the function $G_{ij}(T)$ is such that

$$G_{ij}(T)\, G_{ji}(T) \neq 0 \tag{4.7}$$

then an admissible function f defined by

$$\frac{f_T(\Delta c_{ij})}{f_T(\Delta c_{ji})} = \frac{G_{ji}(T)}{G_{ij}(T)} g(c(j)-c(i)). \tag{4.8}$$

satisfies eq. (4.6).

■

The proof of the Corollary follows directly from eqs. (4.2) (4.3.c) and (2.7).

Remark 4.1. Simulated Annealing as proposed by Kirkpatrick [KGV83] has g, G and f defined as follows

$$g(c(i),T) = e^{-\frac{c(i)}{T}} \tag{4.9}$$

$$G_{ij}(T) = \begin{cases} 1/|S(i)| & \forall j \in S(i) \\ 0 & \forall j \notin S(i) \end{cases}$$

$$f_T(\Delta c_{ij}) = \min[1, e^{-\frac{c(j)-c(i)}{T}}] \quad \forall j \in S\,. \tag{2.2}$$

These functions satisfy the conditions of Corollary 4.1 and then, *a fortiori*, of Theorem 4.2. Hence the configurations generated by simulated annealing asymptotically converges to the global optimum.

Another PHC algorithm can be generated just by replacing the acceptance function of simulated annealing (2.20 with the following one

$$f_T(\Delta c_{ij}) = \frac{e^{-\frac{c(j)-c(i)}{T}}}{1 + e^{-\frac{c(j)-c(i)}{T}}} \tag{4.10}$$

and leaving functions G and g the same.

■

The result stated in Corollary 4.1 is of interest when a PHC algorithm has to be implemented. In fact with (4.7) and (4.8) it is possible to compute the ratio between the acceptance probabilities using only informations related to the two states involved in the transition. There is still a degree of freedom in fixing the actual value of f and in Remark 4.1 it is shown how this degree of freedom can be exploited to generate two different acceptance strategies that lead to the same asymptotic behavior.

A slightly different PHC algorithm related to simulated annealing has been used in a package for standard cell placement [SeS85]. In this algorithm, the acceptance function is the same as simulated annealing and $G_{ij}(T)$ satisfy the following relation:

$$G_{ij}(T) = G_{ji}(T)$$

but are not uniform. This algorithm is again described by a Markov chain with the same stationary probability distribution as simulated annealing.

Note that when the control parameter T, approaches zero, the acceptance functions f given by (2.2) and (4.10) become a unitary step function that assigns probability one only to those transitions which improve the cost function and probability zero to the others. Hence, when T is set to zero, f s degenerate into the usual greedy strategy and select, among all the new configuration that are generated, the ones with cost lower than the present configuration only.

Unfortunately Theorems 4.1 requires the algorithm to perform an infinite number of iterations every time the parameter T is updated. It is clear that a strategy of this kind is practically inapplicable. In fact, a PHC algorithm performing an infinite number of iterations for each value of the parameter T is a conceptual, non implementable algorithm [Pol71], in the sense that an internal loop is never exited.

If we assume that the function g is continuous in its second argument, then also $\pi_i(T)$ is a continuos function. The continuity of $\pi_i(T)$ implies that the stationary probability distribution for a particular value of the controlling parameter, say $\hat{T}$, is a good approximation for the stationary probability distribution for all the values of T sufficiently close to $\hat{T}$. This result suggests a strategy for the control of T: start with a value of T for which the stationary probability distribution is easy to estimate, and update T so that only a few iterations are needed to obtain a good approximation to the new stationary probability distribution. We will elaborate more on this point in Chapter 6 where we discuss how to derive an "efficient" control strategy for the algorithm.

Remark 4.2. Simulated annealing as proposed in [KGV83], follows the strategy outlined above. In fact, starting with a "high temperature" guarantees an easy estimation of $\pi_i(T)$. The acceptance function (2.2) implies that for T sufficiently large, the probability of accepting a move is close to one. Hence, if the generation probability is uniform and the associated Markov chain irreducible, all the states are equally likely and the stationary probability distribution trivial to estimate.

In simulated annealing, the "temperature" is slowly decreased. Since in this case, the function g is given by (4.9) which is obviously continuous in T, we can interpret the control strategy as a direct application of Proposition 4.3.

In addition, the updating rule of simulated annealing is

$$T_{m+1} = \alpha(T_m)T_m$$

where α can be a constant as in (2.1) or a function of T but always less than one and larger than zero. Values of α that yielded good results are in general in the interval [0.9,0.99], which forces the updating to become slower and slower as the algorithm approaches $T = 0$.

■

All the results presented in this section are asymptotic and hence they can only help deciding the control strategy and how long the inner loop of PHC algorithms should be run. However, no sharp bound is given on the actual choice of the number of steps to be taken in the inner loop.

CHAPTER 5

Number of Iterations at Each Value of T

In the previous section we have presented theoretical results which can be used to explain the success of PHC algorithms and to derive qualitative reasoning on the strategy for controlling the parameter T. In this section, we present results which can be used to estimate how many steps should be attempted for each value of T to "minimize" the risk that the algorithm is trapped in a local minima fort the cost function. The results presented are derived from the general properties of the Markov Chains.

PHC algorithms can be used to compute an estimation to the actual stationary probability distribution in the following way. Store how many times each of the states has been visited by the algorithm during its evolution. The ratio between this number and the total number of iterations gives an estimate on the $\pi_i(T)$'s. Actually, when the number of iterations goes to infinity, the result of the calculation converges to the $\pi_i(T)$'s.

The ideal approach to the determination of the number of iterations to take, would be to detect when the approximation is within a specified distance from the stationary probability distribution. Theorem 3.4 and Corollary 3.1 gives us an upper bound on the distance between the actual state distribution and the asymptotic. Unfortunately the bound can be computed only knowing the second largest eigenvalue of the probability transition matrix. Clearly this is beyond the knowledge of the problem we have. Thus we have to resort to another technique. This technique has been obtained by observing that, in order to obtain a good final result, the PHC algorithms have to be able to leave local minima where they could

end up during the computation. Thanks to the properties of Markov chains, it is indeed possible to estimate the number of iterations needed to get out of a local minima at a given value of T with probability $1-\epsilon$, $\epsilon > 0$.

Proposition 5.1. Given a state , say i, such that $P_{ii}(T) \leqslant 1$, then a PHC algorithm has probability $1 - \epsilon$ to leave i if at least

$$\hat{N}_i = \frac{\ln \epsilon}{\ln P_{ii}(T)} \tag{5.1}$$

iterations are performed by the algorithm with T constant.

Proof. For the sake of notational simplicity, in this proof and in the proof of Proposition 5.2, the dependence of P on T will be implicit. Let

$$Q_i^N = \sum_{n=1}^{N} P_{ii}^{n-1}(1-P_{ii}) \tag{5.2}$$

be the probability of leaving the i after at most N iterations. Taking the sum of the series, (5.2) becomes

$$Q_i^N = \frac{(1-P_{ii})(1-P_{ii}^N)}{1-P_{ii}} .$$

Now if a number of iterations $\hat{N}_i$, given by (5.1) are performed, then

$$Q_i^{\hat{N}_i} = 1 - P_{ii}^{\hat{N}_i} =$$

$$= 1 - P_{ii}^{\frac{\ln \epsilon}{\ln P_{ii}}} =$$

$$= 1 - P_{ii}^{\frac{\log_{P_{ii}} \epsilon}{\log_{P_{ii}} P_{ii}}} = 1 - \epsilon.$$

■

The evaluation of $P_{ii}(T), \forall i \in S$ requires to know the values taken by the cost function on the configuration space, which is obviously out of the question. Assuming that the acceptance function is monotonic decreasing in the first argument as is the case of the acceptance functions given in (2.2) and (4.10), there are a number of possible techniques to estimate $P_{ii}(T)$. For the result stated by Proposition 5.1 to hold, we need a conservative estimate of $P_{ii}(T)$. Unfortunately, the techniques we have been able to discover cannot be guaranteed to obtain an upper bound on $P_{ii}(T)$. The most conservative bound is obtained by assuming that $\Delta c_{ij} \quad \forall\ j \in S(i), \forall i \in S, i \neq j$ is constant and equal to $\Delta c_{\hat{i}\hat{j}}$ where $\hat{j}$ is the worst configuration and $\hat{i}$ is the best configuration found so far. To use this estimate, we have to insert a step in the PHC algorithm to record the best and worst configuration.

Proposition 5.1 is a "worst case" result; a similar approach is useful to determine the *expected value* of the number of iterations necessary to leave i.

Proposition 5.2. Let i be a state such that $P_{ii}(T) < 1$ and let R_i^N be the probability that a PHC algorithm leave i after N iterations. The expected value of the number of iterations required to leave i, $\tilde{N}$, is given by

$$\tilde{N}_i = \frac{1}{1 - P_{ii}(T)}$$

Proof. By the definition of expected value

$$\tilde{N}_i = \sum_{n=0}^{\infty} n \; R_i^n \tag{5.3}$$

Substituting in (5.3) the expression for R_i^N given by

$$R_i^N = P_{ii}^{N-1} (1 - P_{ii})$$

we obtain

$$\tilde{N}_i = \sum_{n=0}^{\infty} n\, P_{ii}^{n-1}(1 - P_{ii}) =$$

$$= (1 - P_{ii}) \sum_{n=0}^{\infty} n\, P_{ii}^{n-1} =$$

$$= (1 - P_{ii}) \frac{d}{dP_{ii}} \sum_{n=0}^{\infty} P_{ii}^{n} =$$

$$= (1 - P_{ii}) \frac{d}{dP_{ii}} \frac{1}{1 - P_{ii}} =$$

$$= \frac{1}{1 - P_{ii}}$$

■

As in the previous case, P_{ii} must be estimated. The approximations introduced above are also needed to estimate $\tilde{N}_i$.

Note that a Markov chain represents a stochastic dynamical system. The time (or number of iterations, since the configuration space is finite and discrete) necessary to leave a particular state plays a role which is similar to the time constant in a linear dynamical system. If a linear dynamical system is controlled by a piece-wise constant function, a time as long as a few time constants will bring the system to a new steady state condition. A similar reasoning can be applied here if we assume that a number of iterations which is few times $\tilde{N}_i$ is needed to reach a stationary probability distribution.

$\hat{N}$ and $\tilde{N}$ defined as the the largest values of of $\hat{N}_i$ and $\tilde{N}_i$ respectively determine two different estimates of the number of iterations necessary for the algorithm to obtain a good estimate of the stationary probability distribution.

It is important to note that both $\hat{N}$ and $\tilde{N}$ increase as T approaches zero.

Finally in this section we want to point out that $\hat{N}$ and $\tilde{N}$ depend upon P_{ii} which is unknown. Estimations of P_{ii} which are too crude easily generate values for $\hat{N}$ and $\tilde{N}$ that are unrealistically large. This make the use of this type of estimation impracticable when a PHC algorithm is implemented to solve a realistic combinatorial optimization problem.

CHAPTER 6

A Dynamical Control Strategy for the Annealing Process

In the previous Chapters we concentrated on the study of the structural properties of PHC algorithms and on the conditions on the generation and acceptance function that have to be satisfied to guarantee the convergence of the solution to the global optimum of the cost function. In this Chapter we will focus, for the sake of comparison, on a particular PHC algorithm, the original simulated annealing. We will show how a carefully designed annealing process can produce good results saving a considerable amount of computation time. However it must be noted that the control strategy for the annealing process is not restricted to the original simulated annealing algorithm but applies to every PHC algorithms as well.

The improvements that have been proposed to reduce the CPU time necessary to the simulated annealing to produce good solutions may be subdivided into two groups: improvements involving the move-set design and improvements of the annealing process.

In the former approach, clever move-set based on the idea of range-limiting [OtG84], [SeS84] or changes in the cost function [GrS86] is employed to reduce the chances of generating a next state which is likely to be rejected. This would lead to a significant reduction in CPU time at low values of parameter T where the acceptance probability for up-hill moves decreases rapidly as $\Delta c_{ij} > 0$ increases. call for the enlargement of the move-set obtained by combining several simple moves to obtain a complex one. The idea being that a larger move-set allows a faster exploration of the solution space and an higher probability to escape from

local minima. These improvements of the move set allows the use of more aggressive cooling schedule and this cuts down the CPU time. Unfortunately the approach of complex move-set is usually problem-dependent and is unlikely to be generally applicable to various combinatorial problems. As a result, it has not been widely used.

The latter approach involves careful control of the annealing process. The conditions determining an annealing process are the initial value of T sometimes referred in the literature as "hot condition", the updating rule for T, the equilibrium condition, and the stopping condition. For an annealing process to be problem-independent, the parameters used in the four conditions should be determined by the system itself and should not have any predefined values. For a cooling to be efficient, an early detection of equilibrium and an aggressive strategy to decrement T are desirable. For a cooling to be reliable, quenching must be avoided.

6.1. The Hot Condition

The condition proposed by White [Whi84] is used to determine the starting value of T for the annealing process. The system is considered to be "hot" enough if

$$\sigma << T \quad ,$$

where σ is the standard deviation of the cost distribution. Hence, to determine the initial value of T say T_∞, an initial exploration of the configuration space is performed. During the exploration, all the generated states are accepted, i.e. T is assumed to be infinite. The standard deviation of cost distribution is computed and T_∞ is determined as

$$T_\infty = k * \sigma \quad .$$

k is determined as follows: The cost is assumed to be normally distributed and k is chosen so that the corresponding value of T guarantees the acceptance of a configuration with cost which is 3 σ worse than the present one with probability P. The assumption on the cost distribution leads to the following expression for k

$$k = -\frac{3}{\ln P} \quad .$$

A typical value of k, corresponding to a value of $P = .85$, is 20. Furthermore it should be noted that since at high values of T only a limited number of moves is necessary to achieve the equilibrium distribution, the overhead for using a high value of k is insignificant.

6.2. The Decrement of T

Methods to determine the next value of T based on the standard deviation of the cost distribution at the present value of T have been reported in the literature [AaL85], [OtG84]. The advantage of these type of approaches is that the updating of T is dynamically controlled by the system itself and hence the schemes are generally applicable to a wide variety of problems. Although the idea of quasi-equilibrium was used to guide the T decrement, the extent of the disturbance that could be tolerated by the system with out perturbing too much the quasi-equilibrium it was never mentioned. As a result, an explicit guideline for the choice of a small parameter which controlled the disturbance has not been reported.

In our approach, we use the annealing curve - the curve of average cost $<c(T)>$ versus the logarithmic of T - to guide the decrease of T. The idea is to control T so that the average cost decreases in a uniform manner. The slope of the annealing curve is given by

$$\frac{d<c(T)>}{d(ln(T))} = T \frac{d<c(T)>}{dT} .$$

From the well known statistical physics relation [Rei5] we have:

$$\frac{d<c(T)>}{dT} = \frac{\sigma^2(T)}{T^2} \tag{6.1}$$

where $\sigma^2(T)$ is the variance of the cost at value T of the parameter. From (6.1) it follows that

$$\frac{d<c(T)>}{d(ln(T))} = \frac{\sigma^2(T)}{T}$$

The slope of the annealing curve can be approximated by

$$\frac{<c(T_{k+1})> - <c(T_k)>}{ln(T_{k+1}) - ln(T_k))} = \frac{\sigma^2(T_k)}{T_k} \tag{6.2}$$

where $<c(T_k)>$ and $\sigma^2(T_k)$ are computed recursively, as the algorithm proceeds, with the following relations

$$<c_{(n+1)}(T_k)> = \frac{1}{n+1} c_{(n+1)}(T_k) + \frac{n}{n+1} <c_{(n)}(T_k)> .$$

$$\sigma^2_{(n+1)}(T_k) =$$

$$= \frac{1}{n+1}(c_{(n+1)}(T_k))^2 + \frac{1}{n+1} \sum_{i=1}^{n} (c_{(i)}(T_k))^2 - (<c_{(n+1)}(T_k)>)^2 .$$

The superscript n is the iteration counter, namely the number of moves attempted at the value T_k of the parameter. Eq (6.2) leads to

$$T_{k+1} = T_k \exp\left(\frac{T_k(<c(T_{k+1})> - <c(T_k)>)}{\sigma^2(T_k)} \right) \tag{6.3}$$

Note that Eq. (6.3) is an implicit relation that cannot be used to determine T_{k+1} because $<c(T_{k+1})>$ is not known. Therefore (6.3) must be replaced by the following approximation

$$T_{k+1} = T_k \exp\left(\frac{T(<c(T_k)> - <c(T_{k-1})>)}{\sigma^2(T_k)} \right)$$

To maintain quasi-equilibrium, we require that the expected decrease in the average cost be less than the standard deviation of the cost. For instance:

$$<c(T_k)> - <c(T_{k-1})> = -\lambda\,\sigma(T_k)$$

where $\lambda \leqslant 1$. Finally,

$$T_{k+1} = T_k \exp\left(-\frac{\lambda T_k}{\sigma(T_k)}\right)$$

In the actual implementation, the ratio of T_{k+1}/T_k is lower bounded by a small number (typically 0.5) to prevent a drastic decrease of T caused by the flat annealing curve at high values of T.

6.3. The Equilibrium Condition

To reach an equilibrium means to establish the steady-state probability distribution of the accessible states. However, dynamic monitoring of the steady-state condition for all the accessed states is in practice hardly feasible.
Partly because of this the condition that has to be satisfied for the system to reach the equilibrium is the least addressed issue in the annealing processes in the literature. Typically, either a fixed number of generated configurations [AaL85] or certain minimum number of new accepted configurations [KGV83] is used, in the hope that the system will reach equilibrium by then. The condition we present in this report is completely based on statistical information collected during the operation of the algorithm and has been shown, on a set of examples to which we have applied it, to be fast and reliable in detecting the equilibrium.

Our equilibrium condition is based on the observation that once equilibrium is established, the ratio of the number of the new states generated with their costs within a certain range δ from the average cost to the total number of the newly accepted states will also reach a stable value, say χ. This value depends on the nature of the cost distribution and the sampling range δ. For the case of a normal

distribution, as is the case at high values of T, the ratio between the number of the accepted states having an energy in the range ($<c> - \delta$, $<c> + \delta$) (hereafter referred to as the *within count*), and the accepted states is $erf(\delta / \sigma)$ where $erf(x)$ is the error function [Fel70]. Based on this value and on the size of the problem, a target acceptance *within count* and a maximum tolerance limit are established as the equilibrium parameters. The equilibrium is considered maintained if the *within count* reaches the target value before the tolerance count (i.e. the number of the accepted states with their cost outside the designated range) exceeds the maximum tolerance limit. On the other hand, if the maximum tolerance count is exceeded, both the *within count* and the tolerance count are reset to zero and the counting is initiated again. Ideally, both the χ and the *within count* should be updated dynamically to reflect the change in the cost distribution when T is decreased. For simplicity, in the present implementation they are determined at the beginning of the annealing process and remain constant throughout the annealing process.

The parameter δ should be a fraction of σ so that the final state at any value of T, i.e. the "chosen" equilibrium state, is a state with a cost close to the average cost (within the range of δ) and hence is a highly accessible state at that value of T. In other words, in such case, the final state will be a good representation of the system at that value of T and hence, since the change of T is small, it will be also a good representation of the system at the lower value of T. Such control over the quality of the equilibrium state is missing in the traditional approach where the number of moves attempted at each T is kept fixed. χ is defined by $\chi = erf(\delta/\sigma)$. The *within count* is set to be χ* 3* *size of the problem* and the maximum tolerance is $(1 - \chi)$* 3* *size of the problem*. An upper limit on the number of the new states generated is needed in this case to terminate further generation for low values of T because the target *within count* may never be satisfied at low

values of T without dynamic adjustments of χ and the target *within count*. A typical value for this maximum number of generated moves is $\dot{M}$, where M is the number of states which may be reached in one move, i.e. the cardinality of the set of neighbors $S(i)$.

To guarantee the validity of the statistics, a criterion based on the minimum number of accepted moves is implemented. Detection of equilibrium based on the target *within count* is initiated only when the criterion is satisfied. A typical value for the minimum number of accepted moves is the size of the problem, e.g. the number of cells in a placement problem. At low values of T, the minimum acceptance criterion cannot be met when the maximum generation limit is exceeded. In such case, generation is allowed to extend to an ultimate limit, which is typically few times M. During this extended generation, the regular equilibrium detection based on the *within count* is still in force. Hence, equilibrium may be regarded established at low T if the target *within count* fails in one of the following conditions: (1) the minimum acceptance criterion is satisfied and the maximum generation limit is exceeded; or (2) the minimum acceptance criterion is not met when the ultimate generation limit is exceeded.

The effect of using the new equilibrium condition mentioned above is that the number of moves generated at each value of T is dynamically adjusted throughout the annealing process and a noticeable saving in CPU time is achieved by reducing the number of moves performed at high values of T where the equilibrium is reached quickly.

6.4. The Stopping Criterion

A typical stopping criterion for termination of the annealing process found in the literature is as follows: The algorithm is terminated when the average cost does not change significantly for few consecutive values of T. A different

approach is introduced in this annealing process. When equilibrium is established, we compare the difference between the maximum and minimum costs among the accepted states at that value of T with the maximum change in cost in any accepted move at the same value of T. If they are the same, apparently all the states accessed are of comparable costs and there is no need to use simulated annealing. T is then set to zero and the algorithm becomes a standard "greedy" random selection algorithm. An optional exhaustive search follows which ends the optimization process. This mechanism of stopping the annealing has been found to be quite successful without any negative side effects.

CHAPTER 7

Applications of the Dynamical Control Strategy

The annealing process described in the previous section has been applied to solve the Traveling-salesman problem and the standard cell placement problem. The two problems have been chosen because for both of them are available in literature algorithms based on the standard simulated annealing technique. Furthermore the nature of the problems is different enough to prove the generality of the proposed strategy to control the annealing process.

7.1. Traveling-salesman Problem

The Traveling-salesman problem is a well known NP-complete problem [GaJ79]. It consist in determining a tour on a graph such that every node is visited only once and the length of the tour is minimized. In the particular instance of the traveling-salesman problem we selected, the cities, nodes of the graph, are located at the vertices of a square array. The cost is the total Manhattan distance of a closed tour divided by the number of the cities. The intercity distance is unitary. A move is carried out by randomly selecting two cities in the tour and reversing the order in which the cities in between are visited. The control parameters of the annealing schedule are reported in Table 7.1

parameter	value
k	20
λ	0.7
δ	$0.5\,\sigma$
within count	$erf\ (0.5) * 3N$
maximum tolerance	$(\ 1 - erf\ (0.5)\ ) * 3N$
maximum generation limit	$N(N\ -1)/2$
ultimate generation limit	$2N(N\ -1)$

Table 7.1. Parameter setting for the traveling salesman example.

where N is the number of cities. Problems with N ranging from 49 to 400 are analized.

For the particular traveling-salesman problems considered, the global minima are known. Hence the quality of the solution produced by the algorithm can be carefully assessed. Using the control parameters reported in Table 7.1, the standard deviation from the global minimum is less than 2 per cent for each problem studied. A typical annealing curve is shown in Figure 7.1 for a problem with 400 cities. The curve appears to be rather smooth which is an *a posteriori* indication of a good control strategy for the parameter T.

In Figure 2, the CPU time required for the annealings is compared with the result published in [AaL85] in which a different annealing process is used. Care has to be taken when comparing the CPU time from various studies. In fact two issues may affect the CPU time required to produce the result in addition to the actual efficiency of the algorithms compared. The first issue involves the targeted quality for the solution. The simulated annealing may produce oscillations of the solution in the vicinity of the optimum and a stopping criterion that promptly detects such a situation can save few percents of the global running time. The second issue is related to the differences in the computing environment, both in computer hardware and software. Different efficiency of both operating systems and compilers may produce significant differencies in the CPU time. In the above

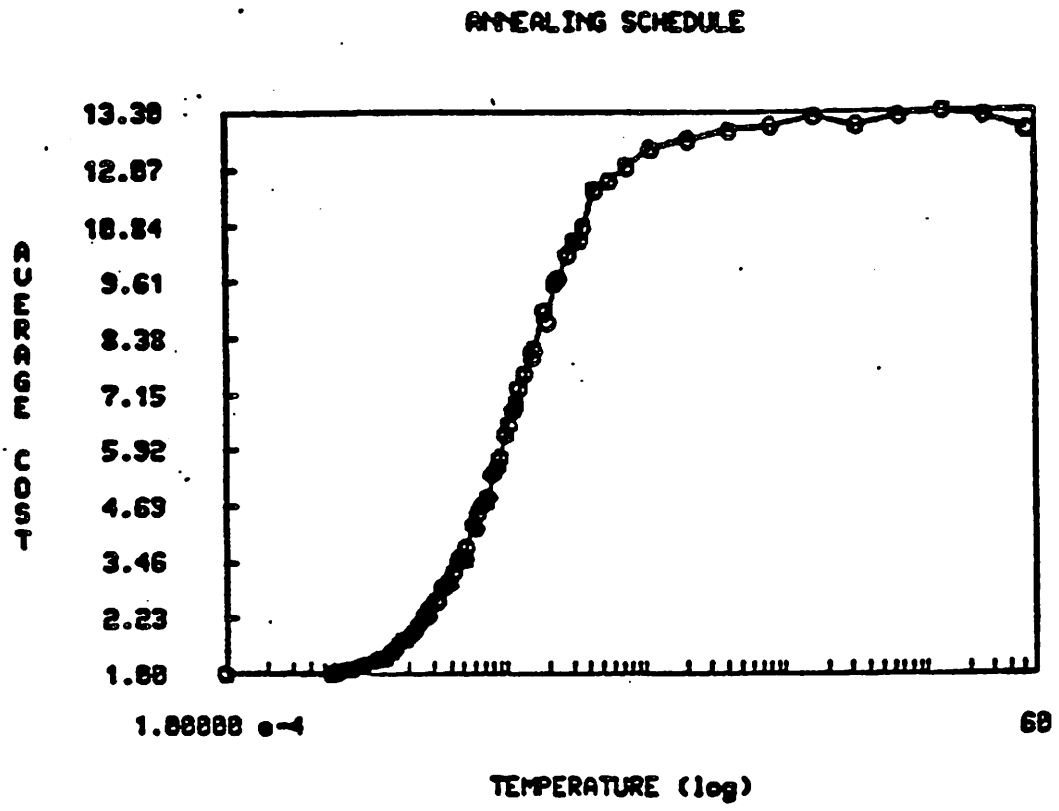

Figure 7.1 Annealing curve for a 400-city traveling-salesman problem

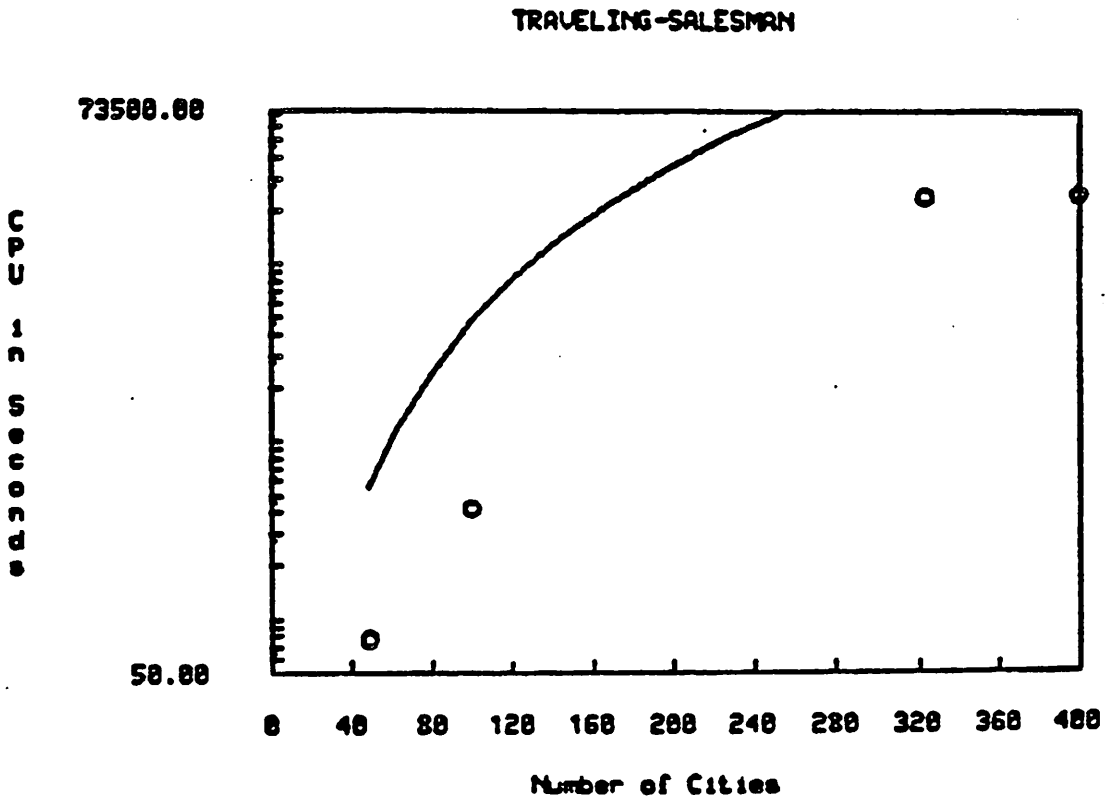

Figure 7.2 Comparison of performance of the new annealing process with the one reported in literature. Solid line is from literature; circles are from the new annealing process.

comparison, the quality of the solution found by the algorithm is comparable in both studies. The computer hardware used is the same, namely a Digital Vax 11-780. The operating system and the programming languages used in algorithm implementation are, however, different. In our case we used C programming language and UNIX BSD 4.3 operating system. The program by Aarts et al. is written in Pascal with VMS operating system.

Besides the above described differencies, a uniform speed-up of more than five times clearly demonstrates the efficiency of our annealing process. The CPU time saving is obtained mainly by reducing the number of moves attempted at high values of T and by using a more aggressive updating strategy for T.

7.2. Placement of the Standard Cells

The standard cell layout stile is used to assemble subblocks of integrated circuits. In this style the cells, each of which implements an elementary logic function, are placed in rows (or columns) and are connected to realize more complex logic functions. The connections are performed in the routing channels available between the rows. The objective of the placement is to select the position of the cells in the rows so that the area of the subblock, namely the area of the cells plus the routing area, is minimized. Of course the minimization is obtained by reducing the routing area since the cell areas are data of the problem.

TimberWolf3.2 [SeS86] is a very efficient simulated annealing based package for the placement and global routing of standard cells. TimberWolf3.2 features an annealing process in which the number of moves that are attempted at each value of T is proportional to the complexity of the circuit, i.e. number of cells, as shown in Table 7.2.

No. of cells	attempts per cell
<= 200	100
<= 500	200
<= 1000	300
<= 1500	400
<= 2000	500
<= 2500	600
<= 3000	700
<= 3500	800
<= 4000	900
<= 4500	1000
> 4500	1200

Table 7.2. TimberWolf optimal number of attempts per cell.

T is updated according to predetermined exponential law.

To test the efficiency of the new proposed annealing process it has been installed in TimberWolf3.2. All the other features of the original program namely the move-set, the penalty term to account for overlap, as well as the range-limiting feature in TimberWolf3.2 are left unchanged. The control parameters for our annealing process are the same as those used in the traveling-salesman problem discussed in the previous section. N is this case the total number of cells.

Four circuits of size ranging from 183 to 800 cells are analyzed using both algorithms. In Table 7.3 the results obtained with the TimberWolf3.2 annealing process and those obtained with the new annealing process are indicated with the subscript 1 and 2 respectively. The quantities Δ_t and Δ_w, given by

$$\Delta_t = \left(1 - \frac{t_2}{t_1}\right) 100 \quad ,$$

$$\Delta_w = \left(1 - \frac{w_2}{w_1}\right) 100 \quad ,$$

represent the saving in CPU time and in estimated wire length of the new annealing process with respect to the one used in TimberWolf3.2.

Size	CPU time (sec.)			Wire length		
	t_1	t_2	Δ_t	w_1	w_2	Δ_w
183	2301	984	57.2%	295585	300448	-1.6%
286	7330	4536	38.1%	317306	330263	-4.1%
469	11117	5140	53.8%	487934	480981	+1.4%
800	38526	32400	15.9%	1270561	1262354	+0.7%

Table 7.3. Comparisons with TimberWolf Annealing Process.

The results collected in Table 7.3 show that savings in CPU time up to 57% are obtained by applying the new annealing process while the quality of the solution is maintained. The improvement in CPU time does not exhibit a constant trend with respect to the size of the problem to be solved. This was expected since the proposed annealing process is adaptive and the number of attempted moves is varied according to the statistics of the problem to be solved, in order to obtain good quality solutions. The new dynamically adjusted annealing schedule compares well with the original schedule as indicated by the annealing curves in Figure 7.3.

As stated in the previous section, CPU time required for simulated annealing varies drastically with the quality of the solution. The speed-up ratio may be reduced if a setting different from the one reported in Table 7.2 for the maximum number of generated moves is used for the original annealing schedule. To test the above conjecture TimberWolf3.2 with the original schedule was allowed to run for an amount of time comparable to the time required by the new schedule. The results we have obtained are collected in Table 7.4.

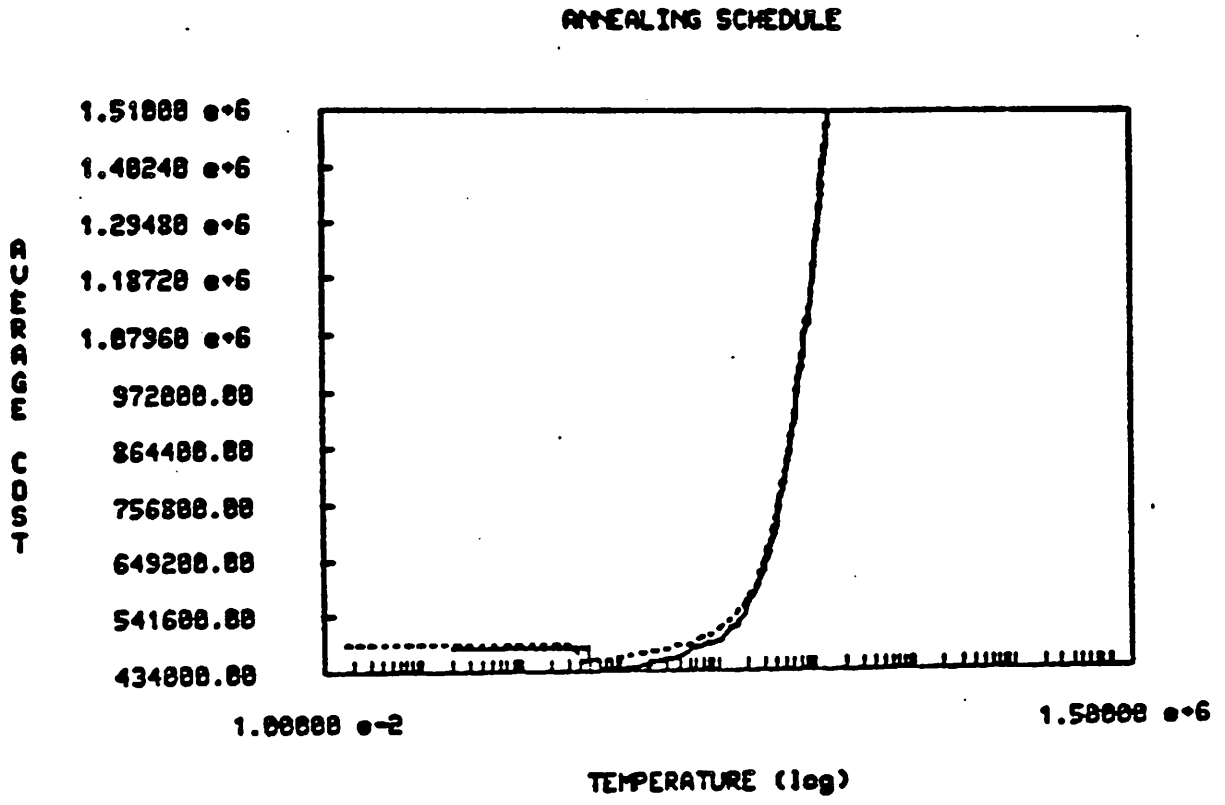

Figure 7.3 Annealing curves in TimberWolf: solid line new annealing process; dotted line original annealing process.

Size	Wire length saving new vs. original	CPU time new vs. original
183	+2 %	97 %
286	-3 %	115 %
469	+42 %	102 %
800	+1 %	97 %

Table 7.4. Comparisons between the two schedules with the same amount of CPU time.

The results represented in Table 7.4 show that the two schedules produce solutions with comparable quality in three out of four cases. In the fourth case the

quality of the solution obtained with the new schedule is considerably better. The conclusion we derive is that the adaptive schedule succeeds in the task of determining the exact number of moves to be performed at each value of T necessary to produce good quality results. Furthermore the set of experiments conducted so far demonstrate the capability of our general annealing process to compete with handicrafted cooling schedules like the one used in the original TimberWolf3.2. The general applicability and efficiency of the new annealing technique is also realized.

Finally is necessary to note that the new annealing process has still a lot of margin for improvements. In fact all the parameters that control the process are determined under the assumption that the normal distribution for the cost function is maintained through the annealing process. This assumption is not realistic when T is small. In this case in fact the cost distribution is not symmetric any more and the Gaussian distribution tends to a Gamma distribution.

CHAPTER 8

Conclusions

A theory of a class of algorithms for the solution of combinatorial optimization problems inspired by the technique known as simulated annealing has been developed. The class of algorithms has the characteristic of being probabilistic and of being able to climb "hills", i.e. to accept intermediate solutions which increase the cost. For this reason, these algorithms have been called Probabilistic Hill Climbing (PHC) algorithms. The mathematical model used in the study of their properties is a Markov chain with finite state space.

Based on the key results on stationary probability distributions of Markov chains, we have derived conditions on the parameters of PHC algorithms to guarantee that an optimum configuration is found with probability one. The theory requires that an infinite number of iterations be performed as intermediate steps.

Some guidelines in the selection of the number of iterations used in the intermediate steps have been given. Finally an annealing process with parameters that are dynamically determined according to the statistics produced by the algorithm during its evolution is presented. The new annealing process has been substituted to the traditional one in two algorithms used to solve the Traveling Salesman problem and to determine the optimal placement of standard cells. The solutions obtained by the new annealing process have been compared with the solutions obtained by the same algorithm with the traditional annealing process. In both cases the results obtained with the new annealing process were of the same or better quality than those obtained with the traditional annealing process.

Much work remains to be done to exploit fully the mathematical model. We are exploring new control strategies and techniques which we hope will give tighter bounds on the number of iterations needed to maintain a given level of confidence in the optimality of the results. In particular, we are looking at the theory of nonhomogeneous Markov chains to be able to find stronger results than the ones so far obtained.

Furthermore we are investigating the possibility of determining new acceptance functions that guarantee the same asymptotic probability distribution but a faster rates of convergence. In addition, a set of placement and routing packages is being developed which incorporates the control strategies suggested by the theory.

References

[AaL85] E. H. L. Aarts and P. J. M. Laarhoven, "Statistical Cooling: A General Approach To Combinatorial Optimization Problems", *Philips J. Res 40* (1985), 193-226.

[AJM84] C. R. Aragon, D. S. Johnson, L. A. McGeoch and C. Schevon, "Simulated Annealing Performance Studies", *Workshop on Statistical Physics in Engineering and Biology*, Yorktown Heights, April 1984.

[Bin78] K. Binder, *Monte Carlo Methods in Statistical Physic*, Springer-Verlag, 1978.

[Coo71] S. A. Cook, "The Complexity of Theorem-Proving Procedure", *Proc. 3rd Ann. ACM Symp. On Theory of Computing*, New York, 1971, 151-158.

[Doo51] J. L. Doob, *Stochastic Processes*, J. Wiley, 1951.

[Fel70] W. Feller, *An Introduction to Probability Theory and Applications* , J. Wiley, 3rd Edition 1970.

[Fre71] D. Freedman, *Markov Chains* , Holden-Day, 1971.

[GaJ79] M. R. Garey and D. S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*, Freeman, 1979.

[GrS86] J. W. Greene and K. J. Supowit, "Simulated Annealing Without Rejected Moves", *IEEE Trans. on Computer Aided Design CAD-5*, 1 (Jan. 1986).

[Kar73] S. Karlin, *A First Course in Stochastic Processes*, Academic Press, 1973.

[Kar72] R. M. Karp, " "Reducibility Among Combinatorial Problems"", *Complexity of Computer Computations*, New York, 1972.

[KGV83] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing", *Science 220*, 4598 (13 May 1983), 671-680.

[LLK85] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan and D. B. Shmoys, *The Traveling Salesman Problem*, J. Wyley, Chichester, 1985.

[LuM86] M. Lundy and A. Mees, "Convergence of the Annealing Algorithm", *Mathematical Programming 34* (1986), 111-124.

[MRR53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth and A. H. Teller, "Equation of State Calculations by FAst Computer Machines", *Journal of Chem. Physics 21* (1953), 1087.

[OtG84] R. H. J. M. Otten and L. P. P. P. Ginneken, "Floorplan Design using Simulated Annealing", *Proc. IEEE Int. Conference on Computer Aided Design*, Santa Clara, Nov. 1984, 96-98.

[Pol71] E. Polak, *Computational Methods in Optimization a Unified Approach*, Academic Press, 1971.

[Rei5F. Reif, *Statistical and Thermal Physics*, McGraw Hill Inc. , New York , 1965

[Sch80] J. T. Schwartz, "Fast Probabilistic Algorithms for Verification of Polynomial Identities", *Journal of ACM 27*, 4 (Oct 1980).

[SeS84] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package ", *Proc. 1984 Custom Integrated Circuit Conference*, Rochester N. Y, May 1984.

[SeS85] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package", *IEEE Journal of Solid-State Circuits 20*, 2 (April 1985), 510.

[SeS86] C. Sechen and A. Sangiovanni-Vincentelli, "TimberWolf3. 2: A New Standard Cell Placement and Global Routing Package", *Proc. 1986 Design Automation Conference*, Las Vegas, Nevada, June 29 - July 2, 1986, 432-439.

[Sen80] E. Seneta, *Non-negative Matrices and Markov Chains*, Springer-Verlag, New York, Second Edition, 1980.

[VeK83] M. P. Vecchi and S. Kirkpatrick, "Global Wiring by Simulated Annealing", *IEEE Transactions on Computer-Aided Design CAD-2*, 4 (Oct. 1983).

[Whi84] S. White, "Concept of Scale in Simulated Annealing", *Proc. IEEE Int. Conf. on Computer Design*, , Port Chester, New York, 1984, 646-651.