Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road. Ann Arbor. MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 1356272

Fast point pattern matching by heuristic and stochastic optimization techniques

Agrawal, Ashish, M.S.

New Jersey Institute of Technology, 1994

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

ABSTRACT

FAST POINT PATTERN MATCHING BY HEURISTIC AND STOCHASTIC OPTIMIZATION TECHNIQUES

by
Ashish Agrawal

This work is concerned with one of the methodologies used in the final stages of machine vision: the matching of model point patterns to observed point patterns. Conventional search methods not only fail to arrive at the optimal match, but are also computationally expensive and time consuming. To arrive at the optimal pattern match, *stochastic and heuristic optimization* as the search technique, exploiting Simulated Annealing (SA), Evolutionary Programming (EP) and Mean Field Annealing (MFA), are explored in detail. A comparison of results obtained using SA versus "hill-climbing" and "exhaustive search" techniques, and results of EP are presented. The relative effectiveness of these optimizing search algorithms over other conventional algorithms will be demonstrated. Finally, the limitations of MFA are discussed.

FAST POINT PATTERN MATCHING BY HEURISTIC AND STOCHASTIC OPTIMIZATION TECHNIQUES

by
Ashish Agrawal

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

January 1994

APPROVAL PAGE

FAST POINT PATTERN MATCHING BY HEURISTIC AND STOCHASTIC OPTIMIZATION TECHNIQUES

Ashish Agrawal

Dr. Nirwan Ansari, Thesis Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Edwin S. H. Hou, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Y. Shi, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Ashish Agrawal

Degree: Master of Science in Electrical Engineering

Date: January 1994

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1993
- Bachelor of Science in Electrical Engineering,
  Manipal Institute of Technology, Manipal, India, 1990

Major: Electrical Engineering

Presentations and Publications:

N. Ansari, E. S. H. Hou and A. Agrawal, "Point Pattern Matching using Simulated Annealing," *Proceedings of the 1993 Regional IEEE Control Conference*, pp.215–218, August 1993.

A. Agrawal, "Backpropagation Techniques in Neural Networks: A Quantitative Analysis of Various Algorithms," *1993 International MATLAB Conference*, Cambridge, MA, October 18–20, 1993.

I dedicate this thesis to my parents for
their eternal love, support and understanding

ACKNOWLEDGMENT

The author wishes to express his sincere gratitude to his advisor, Professor Nirwan Ansari, for his guidance, friendship, and moral support throughout this research.

Special thanks to Professors Edwin S. H. Hou and Y. Shi for serving as members of the committee.

The author would also like to thank Lisa Fitton for her help with proofreading, and Chris Peckham for carefully preparing the impeccable LaTeX infrastructure which enabled an *overnight* compilation of this document.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Point pattern matching is a troublesome but crucial task in machine vision. Existing algorithms employing conventional search techniques usually fail to arrive at a global optimal match and, moreover, require *a priori* knowledge of certain attributes of the search points. Algorithms that arrive at an optimal pattern match in the presence of noise and an incomplete data set are presented herein. The techniques of Evolutionary Programming (EP), Simulated Annealing (SA), and Mean Field Annealing (MFA) are exploited and the entire task is mapped on to a stochastic and heuristic optimization framework.

These techniques provide an efficient means of traversing the search space combining the elements of "gradient descent" and "random search," thereby incorporating strategies which not only exploit the concept of a "depth first search" but also bring in the advantage of a "breadth first search." In fact, it is this characteristic of these algorithms to prevent the search from being limited to a subset of the search nodes, thereby escaping local optima, which would otherwise result in an inefficient cost realization.

For illustrative purposes, a two-dimensional $(2-D)$ framework and a similarity transformation are considered. The pattern sets (*model* and *observed*) are compared for a similarity transformation such that the resulting error between the *model* points undergoing the similarity transformation and the observed pattern is minimized. The error is formulated as a cost function using the transform parameters as its variables. The task finally boils down to evaluating the search space such that this cost is minimized, thereby arriving at a global minima.

To enable an efficient realization of the above strategies, various parameters, such as, the energy function (a function that relates a solution node with its corre-

sponding cost), the acceptance rule (which determines whether a particular search node should be accepted), a cooling schedule (an algorithmic parameter used to model the concept of stability at lower temperatures) and a perturbation rule (which enables one to traverse the search space), etc. are defined. Most of these are problem specific and are best designed heuristically.

Results show that the above techniques perform well in the presence of noise (interestingly, when the number of points constituting a pattern is significantly higher, the computation of the transformation parameters is less prone to error, resulting in a convergence with a lower match error). Graphs of *model* and *observed* points along with a match error plot at every search attempt are presented for ideal and noisy cases in the presence of complete and partial data sets. Not only is the matching optimal, but also efficient in that the algorithm needs to search a fairly low number of nodes constituting the solution space.

CHAPTER 2

POINT PATTERN MATCHING

2.1 An Overview

Shape recognition is an important task in machine vision and pattern recognition. We use the term *shape* to refer to the invariant geometrical properties of the relative distances among a set of static spatial features of an object. These static spatial features are known as the *shape features* of the object. For the purpose of recognition, much of the visual data perceived by the human eye is highly redundant. It has been suggested from the point of the human visual system [1] that some dominant points along an object contour are rich in information content and sufficient to characterize the shape of the object.

Numerous studies on planar object recognition have been carried out. The recognition task can be modeled as searching for an assignment between two features. Commonly used features are holes and points [2, 3, 4, 5, 6, 7, 8], line-segments [9, 10, 11, 12, 13], curve-segments [14, 15, 16, 17, 18, 19], or a combination of these features [20, 21]. These features are obtained by a preprocessing step such as edge detection, polygonal approximation, and corner extraction. Among the methods mentioned above, [6, 8] which use relaxation labeling for point pattern matching do not assume, similar to our proposed algorithm, knowledge on the order of the points. However, a good estimate of the initial assignment between the points of the two point patterns is important relative to the convergence of the algorithm and the validity of the result. These methods [6, 8], inheriting the drawback of relaxation labeling, are complex, and computationally expensive because of their sequential nature. Moreover, existing methods that use points as their features, usually require *a priori* knowledge of the order of the arrangement of the points.

Evidently, the problem addressed here is that of recognizing and locating objects which are represented by a set of points. That is, each object is repre-

sented by a set of dominant points (shape features). Information about the order of these points is not known or provided. The task is to find a subset of points in a point pattern that match to a subset of points in another point pattern through a transformation in a certain optimal sense with the constraint that the mapping is single. In a general setting, the points are arranged in n-dimensional space, and the transformation is specified according to the geometric and environmental constraints of the problem. Exhaustive search to find the best assignment mapping one set of points to another set is, if the number of points that are to be matched is large, computationally expensive. Thus, point pattern matching, wherein points are used as shape features, is a crucial vision task.

The current literature identifies three main types of search methods:

- calculus-based
- enumerative
- random

Random search algorithms have achieved increasing popularity as researchers have recognized the shortcomings of calculus-based and enumerative schemes. Random search schemes, though robust, are computationally inefficient. The conventional search methods do not meet the robust requirement, because they are local in scope. However, as more complex problems are attacked, other methods will be necessary.

We take a similar view by posing the recognition task as a point-pattern matching problem. Our approach is more general in that it assumes no knowledge on the sequential order of the points. For the computations, only two-dimensional point patterns (because images are inherently $2-D$) and the similarity transformation is considered, however, the algorithm itself is not restricted to $2-D$ point patterns or/and the similarity transformation.

2.2 Problem Statement

Given two sets of points defined as follows:

$$\begin{aligned} P &= \{P_i : P_i \in R^N; i = 1, 2, 3, \cdots, m\} \\ O &= \{O_i : O_i \in R^N; i = 1, 2, 3, \cdots, n\}, \end{aligned} \tag{2.1}$$

find an assignment $P' \rightarrow O'$, where $P' \subseteq P$ and $O' \subseteq O$, such that the *match error* between $T(P')$ and O' is minimized. The match error, defined later, indicates the degree of match; the lower the match error, the better the quality of the match. T is a predefined similarity transformation; $T \equiv$ {rotation, translation, scaling}. It should be noted here that the problem is different from that of image registration wherein the objective is to align two images [22] through a geometric transformation.

Let O be an *observed* point pattern and P a *model* point pattern. The 2-D similarity transformation T is defined by the mapping $X \rightarrow U$; X and $U \in R^2$ such that

$$\begin{bmatrix} u \\ v \end{bmatrix} = S \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}, \tag{2.2}$$

where

$$\begin{aligned} X &= \begin{bmatrix} x & y \end{bmatrix}^T, \\ U &= \begin{bmatrix} u & v \end{bmatrix}^T, \end{aligned}$$

S = scale factor, θ = angle of rotation, e = translation in the x-axis and f = translation in the y-axis. By letting $a = S\cos\theta$ and $b = S\sin\theta$, the similarity transformation can be rewritten as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}. \tag{2.3}$$

2.3 Generic Approach

In order to determine the degree of match between the model and the observed points, the parameters of the similarity transformation which map P *optimally* to O

need to be found; in this case, in the *minimum least squared error sense*. Mapping P under the transformation, $T[(x_i, y_i)] = [(u'_i, v'_i)]$, we have

$$\begin{bmatrix} u'_i \\ v'_i \end{bmatrix} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}. \tag{2.4}$$

The squared error between the model and observed point sets may then be defined as

$$\begin{aligned} \varepsilon^2 &= \sum_{i=1}^{n} (u_i - u'_i)^2 + (v_i - v'_i)^2 \\ &= \sum_{i=1}^{n} (u_i - ax_i - by_i - e)^2 \\ &\quad + (v_i + bx_i - ay_i - f)^2. \end{aligned} \tag{2.5}$$

In order to find the parameters that minimize the squared error, we take partial derivatives w.r.t. a, b, e and f of the squared error defined in equation (2.5) above and equate them to zero. Thus, we have

$$\begin{aligned} \frac{\partial \varepsilon^2}{\partial a} &= \sum_{i=1}^{n} (u_i - ax_i - by_i - e)(-x_i) \\ &\quad + (v_i + bx_i - ay_i - f)(-y_i) = 0, \\ \frac{\partial \varepsilon^2}{\partial b} &= \sum_{i=1}^{n} (u_i - ax_i - by_i - e)(-y_i) \\ &\quad + (v_i + bx_i - ay_i - f)(-x_i) = 0, \\ \frac{\partial \varepsilon^2}{\partial e} &= \sum_{i=1}^{n} (u_i - ax_i - by_i - e)(-1) = 0, \text{ and} \\ \frac{\partial \varepsilon^2}{\partial f} &= \sum_{i=1}^{n} (v_i + bx_i - ay_i - f)(-1) = 0. \end{aligned}$$

Solving the above equations and rewriting the same in terms of matrices, we have

$$A \begin{bmatrix} a & b & e & f \end{bmatrix}^T = C, \tag{2.6}$$

where

$$A = \begin{bmatrix} \sum(x_i^2 + y_i^2) & 0 & \sum x_i & \sum y_i \\ 0 & \sum(x_i^2 + y_i^2) & \sum y_i & -\sum x_i \\ \sum x_i & \sum y_i & n & 0 \\ \sum y_i & -\sum x_i & 0 & n \end{bmatrix}$$

and

$$C = \begin{bmatrix} \sum(u_i x_i + v_i y_i) \\ \sum(u_i y_i - v_i x_i) \\ \sum u_i \\ \sum v_i \end{bmatrix}.$$

Then the optimal transformation parameters yielding the minimum least squared error may be obtained as

$$\begin{bmatrix} a & b & e & f \end{bmatrix}^T = \begin{bmatrix} A^{-1} & C \end{bmatrix}. \tag{2.7}$$

The least squared error thus obtained, however, quantifies the degree of match only between the model points' subset P' and the observed points' subset O'. To obtain a measure of the overall match between the two pattern sets, the following *heuristic measure* [2, 3] is used

$$\hat{\varepsilon} = \begin{cases} \frac{\varepsilon^2}{S*k}(1 + (\frac{m-2}{k-2})\log_2(\frac{m-2}{k-2}) & k \geq 3 \\ \infty & k = 0, 1, 2. \end{cases} \tag{2.8}$$

defined as the match error, which penalizes for an incomplete match. Herein, k denotes the number of model points that match the observed points, m denotes the number of model points, and S denotes the scale factor. From the equation above, it should be noted that a match between two or fewer points is considered an under-determined case. The logarithmic term serves as the penalty factor for incomplete matching of the pattern sets. When all points of the two patterns match ($k = m$), the match error equals the normalized least squared error. The *problem* defined above is a *combinatorial optimization problem* (COP) and can be best approached by techniques suited for problems of this class.

CHAPTER 3

SIMULATED ANNEALING

3.1 General Framework

Simulated annealing, first introduced by Kirkpatric *et al* [23] is analogous to the way liquids crystallize: at high temperatures the *energetic* molecules are free to move and rearrange, and with a decrease in temperature, lose mobility as a result of decreasing energy, finally settling down to an *equilibrium state* resulting in the formation of a crystal having *minimum energy.* Equivalently, in *simulated* annealing, there are two operations involved: a thermostatic operation [24] which schedules the decrease of the temperatures (an algorithm parameter), and a random relaxation process which searches for the equilibrium solution at each temperature.

The SA technique [25] eliminates most disadvantages of the "hill-climbing" methods: solutions do not depend on the starting point any longer and are usually close to the optimal point. Moreover, the SA algorithm can escape local optima for a COP. It is essentially a *stochastic search algorithm; at a given temperature, it arrives at a possible subsequent state* Π_{i+1} *(i.e.,* Π_j*) by 'perturbation' of the present state* Π_i . The transition is carried out based on the following rule:

$$\Pi_{i+1} = \begin{cases} \Pi_j & \text{if } A_{ij} \geq \eta \\ \Pi_i & \text{if } A_{ij} < \eta, \end{cases} \tag{3.1}$$

where $\eta \in [0, 1]$ and is a uniform random number, and A_{ij}, the *acceptance probability*, is defined as

$$A_{ij} = e^{\frac{\Delta f}{T}}, \tag{3.2}$$

where $T \equiv$ *scheduling temperature* and Δf is a measure of the energy change between subsequent states and is defined as

$$\Delta f = -(f(\Pi_j) - f(\Pi_i))^+, \tag{3.3}$$

where $f(\Pi_n) \equiv$ *energy at state* n, and

$$(x)^+ = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0. \end{cases} \tag{3.4}$$

The number of transitions at a particular temperature is determined heuristically as some fraction of the total number of search nodes that constitute the solution space. The scheduling temperature T is determined using the cooling schedule which, again, is chosen heuristically based on the size of the solution space and the number of transitions involved at each successive temperature.

The search is said to converge to its optimal solution when the scheduling temperature reaches its minimum level and all transitions at this temperature have been exploited for a possible subsequent state. The number of transitions at a given temperature may be curbed to a minimum if, for a predefined number of such consecutive transitions, the cost of the subsequent states does not change much (again a problem-dependent, predefined value).

3.2 Implementation of SA to Point Pattern Matching

To map the point pattern matching problem onto the SA framework we now explicitly define the *coding scheme* (a way of representing the point pattern), a *perturbation rule* for generating new assignments (configurations or states), the *acceptance rule*, the *cooling schedule*, and the *convergence criterion*.

3.2.1 The Coding Scheme

We code the point pattern as a string representation constituting of nodes (i.e., labels for different point co-ordinates), thereby forming the search/solution space. Thus each code is an assignment between two sets of points; each *cell value* of a string indicates an observed point that is assigned to a model point which is, in turn, denoted by the *cell number*. Suppose we have m model points and n observed points. Accordingly, we choose a code consisting of m cells, wherein each cell may

take on any integer value from 0 to n (a value of zero denotes that no observed point could be matched with a model point). Thus, the cell position-value (corresponding to the model point) in the string and the value within each cell (corresponding to the observed point) indicates an assignment of match from the model point to an observed point, with the constraint of one to one mapping. For example, consider the following assignment for 12 cells as shown in Fig. 3.1. The cell position from

2	5	7	0	9	1	3	4	6	11	10	12

Figure 3.1 An Arbitrary Assignment of *Observed* Point Labels to *Model* Point Labels Depicting the Scheme of "String" Representation.

the left indicates the label of the model point; for example, the sixth cell corresponds to the sixth model point. Then, accordingly, the first model point is assigned to the second observed point, the second to the fifth, the third to the seventh, the fourth is *not* assigned, the fifth to the ninth, and so on. Thus, each code (an assignment) is analogous to the state of a liquid.

3.2.2 The Energy Function

The energy (cost associated with each node of the solution space) function, analogous to the energy of a state of the liquid, is defined as the match error of the assignment. Restating equation (2.8), it is defined as

$$E = \begin{cases} \frac{e^2}{S*k}(1 + (\frac{m-2}{k-2}) \log_2(\frac{m-2}{k-2}) & k \geq 3 \\ \infty & k = 0, 1, 2 \end{cases} \tag{3.5}$$

3.2.3 The Perturbation Rule

Consider the string assignment depicted above in Fig. 3.1. We generate randomly, two numbers within the ranges 1 and m (the number of model points) and 1 and n (the number of observed points), respectively; say, "1" and "6." The first random number indicates the cell of the string whose value will be replaced by the second number representing the observed point. The next step is to substitute the replaced

number in the cell from which the original replacement came from. Thus, after the replacement we have the sting assignment as depicted in Fig. 3.2.

1	5	7	0	9	2	3	4	6	11	10	12

Figure 3.2 A "String" Representation Illustrating the Technique of *Perturbation*.

3.2.4 The Acceptance Rule

This rule is used for deciding whether to accept or ignore the subsequent search node. Assignments with lower energies are always accepted, and to provide a mechanism to escape a local optima, a new assignment with a higher energy is occasionally accepted. Equations (10), (11), and (12) define the probability of accepting a new assignment based on the acceptance probability A_{ij}. It should be noted that as the temperature T is decreased $A_{ij} \rightarrow 0$, thereby reducing significantly the probability of accepting assignments with higher energy states at low temperatures. This is the reason why, a search space with local minima, whose energy is *nearly* equal to that of the global minima, may sometimes yield solutions close to the optimal one at very low temperatures.

3.2.5 The Cooling Schedule

As mentioned earlier, the cooling schedule is problem-dependent. Based on a cooling schedule where the temperature is decreased linearly, such that its final value for a heuristically pre-determined number of maximum search attempts (i.e., total number of transition attempts) is zero, we define the scheduling temperature T as follows:

$$T = T_0(1 - (n/n_{max})), \tag{3.6}$$

where T_0 is chosen (again, heuristically) as the *starting (initial) temperature*, n denotes the nth search attempt, and n_{max} denotes the *maximum number* of such search attempts.

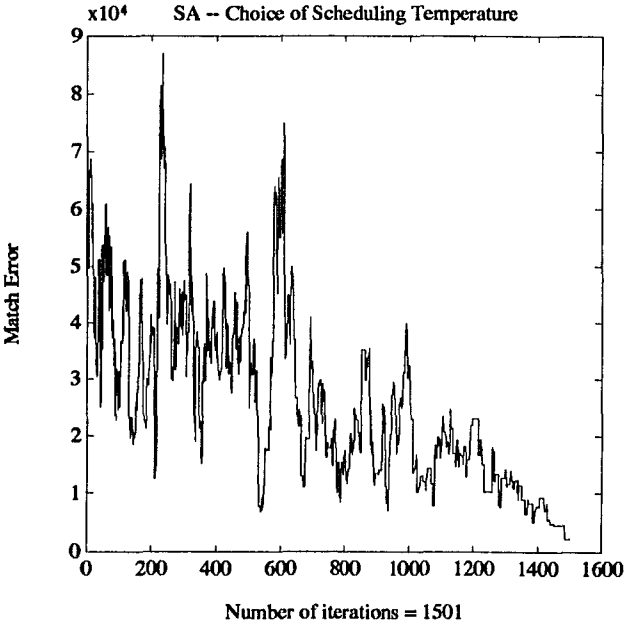

Figure 3.3 Plot Illustrating the Result of A *High* Scheduling Temperature T. Herein *Every Other Search Node is Accepted* as a Solution Thereby Failing to Converge Even after Over 1500 Iterations.

The choice of this parameter plays a crucial role in the entire algorithm and often proves to be the deciding factor behind the convergence speed. As an illustration of this fact, consider Fig. 3.3. Choosing a high value of T_0 results in almost an unstable state (*i.e.*, almost every other search node is accepted as a possible solution, causing the search to oscillate through the solution domain, proving detrimental to the convergence speed. Furthermore, a low value of T_0 may cause the search to get *stuck* in a local minima, as illustrated by Fig. 3.4, affecting the convergence speed.

3.2.6 The Stopping Criterion

In this algorithm, the process of annealing is terminated naturally when an optimal solution (with match error $\hat{\epsilon} \approx 0$) is obtained, or else when the scheduling temperature T cannot be decreased anymore (i.e., when it reaches its lowest value of zero), which happens when the predetermined number of search attempts have

been exhausted. Usually, termination due to the latter case is an indication that the scheduling temperature needs to be remodeled or/and a larger number of search attempts be allowed.

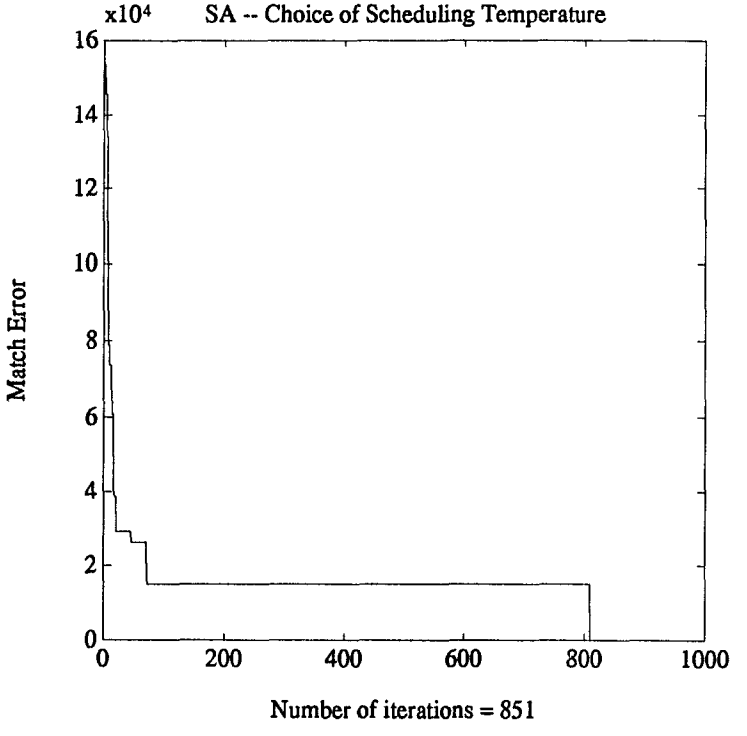

Figure 3.4 Plot Illustrating the Result of A *Low* Scheduling Temperature T. Herein the Solution Gets "Stuck" at A Local Minima for Over 750 Iterations.

3.3 Simulation Results

Experimental outcomes depicting average results have been presented. For any particular simulation the same may vary by 20 to 30 iterations on an average. First we present results which compare the SA technique with "hill-climbing", and "exhaustive search," for the ideal case, *i.e.*,

- *clean* and *complete* pattern set.

Algorithm used	Iterations	Matching	Error
Exhaustive search	refer eqn. (15)	100%	≈ 0
Hillclimbing	$>$ 10,000 (rarely converges)	≤ 25 %	$\geq 3 \times 10^4$
SA	≈ 130	100%	≈ 0

Table 3.1 SA Results Compared with "Hill-climbing" and "Exhaustive Search"

Next, we present results for four cases; namely, (1) *clean* and *complete* pattern set, (2) *clean* and *incomplete* pattern set, (3) *noisy* and *complete* pattern set, and (4) *noisy* and *incomplete* pattern set. We define as an *incomplete* pattern set wherein 2 *observed* points are missing out of a total of 12 points and as a *noisy* pattern set wherein the *observed* points have been corrupted with a Gaussian random variable of $mean = 0$ and $variance = 4$.

The results mentioned in Table 3.2 are depicted by their respective *model* pattern,

Case	Iterations	Matching	Error
(1) *clean* and *complete*	105	100%	$= 0$
(2) *clean* and *incomplete*	298	100%	≈ 0
(3) *noisy* and *complete*	197	100%	≈ 0
(4) *noisy* and *incomplete*	500	100%	≈ 0

Table 3.2 SA Results Compared for the Mentioned *Incomplete* and *Noisy* Cases

observed pattern, and match error plots, for each of the four cases. One should note that a complete match is always obtained regardless of the quality of the data set; it is almost always possible to realize this at the cost of a slightly higher convergence value, using SA. Situations where this might fail are those solution spaces wherein exist local optimas of nearly the same low energy as that of the global optima, regardless of the physical (co-ordinate) proximity. This happens because at energies close to the global optima the low scheduling temperature might not allow further "breadth first search" to be incorporated in the algorithm.

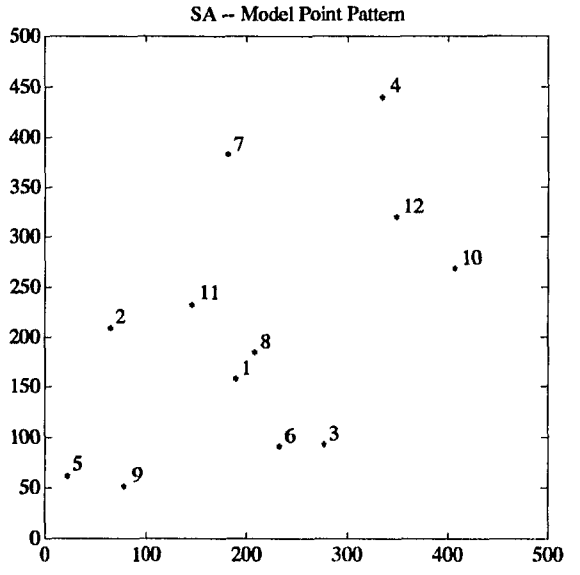

Figure 3.5 Case 1: Model Point Pattern

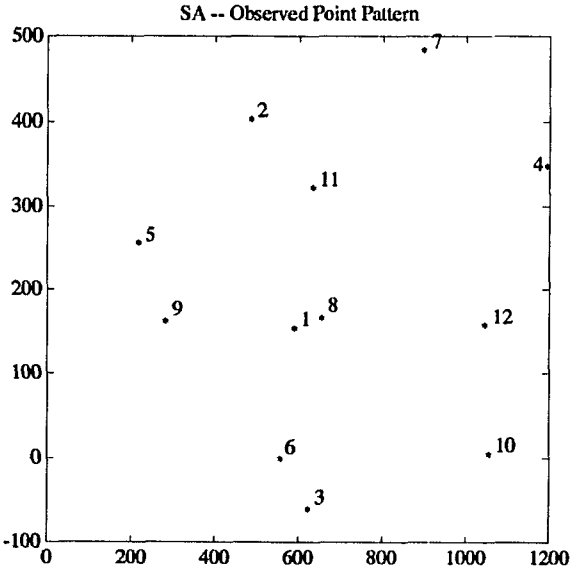

Figure 3.6 Case 1: Observed Pattern = T[Model Pattern]

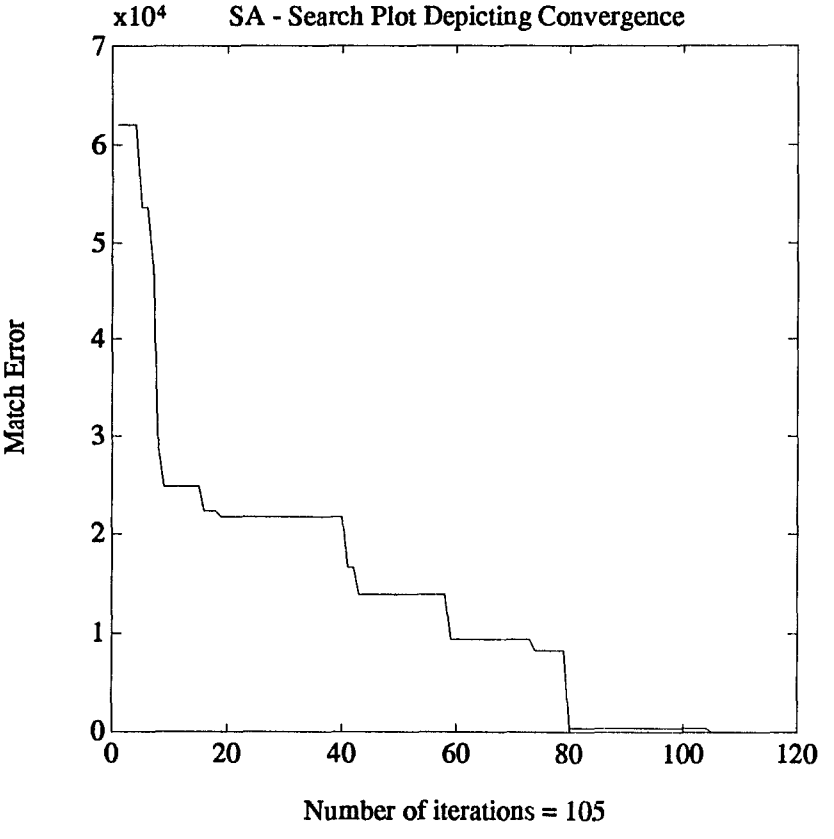

Figure 3.7 Case 1: Plot Showing the Error Convergence with Each Iteration for the Clean and Complete Pattern Set.

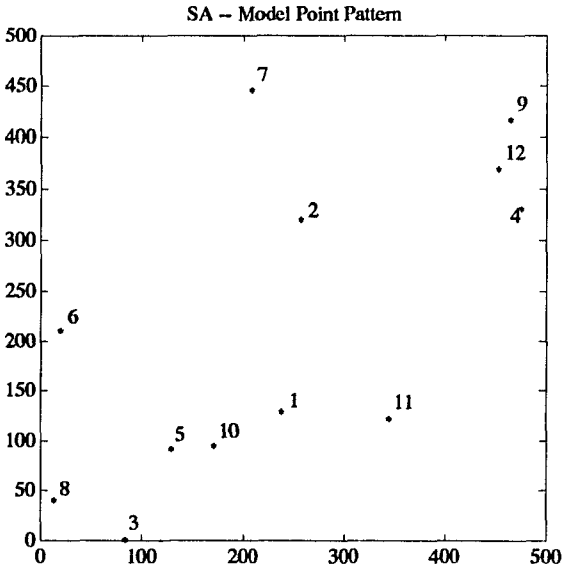

Figure 3.8 Case 2: Model Point Pattern

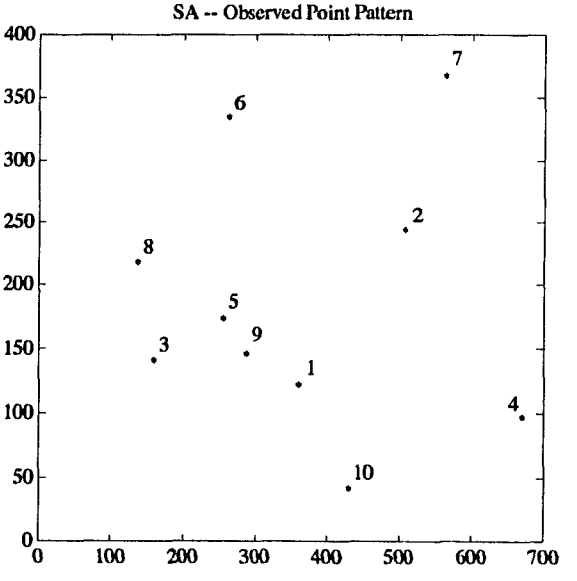

Figure 3.9 Case 2: Observed Pattern = T[Model Pattern] - 2 Points

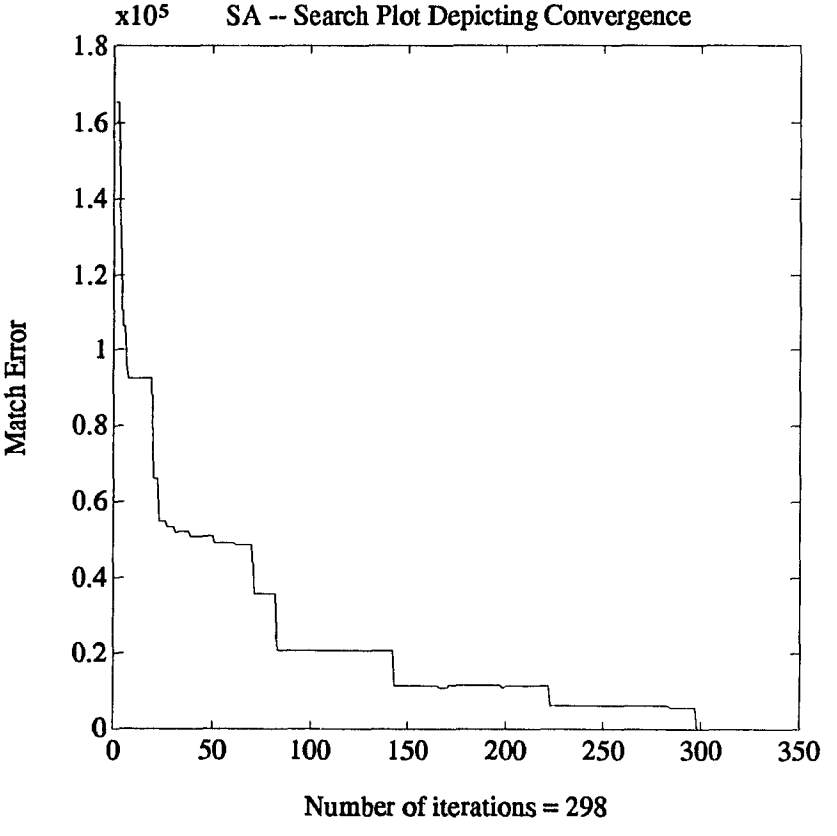

Figure 3.10 Case 2: Plot Showing the Error Convergence with Each Iteration for the Clean and Incomplete Pattern Set.

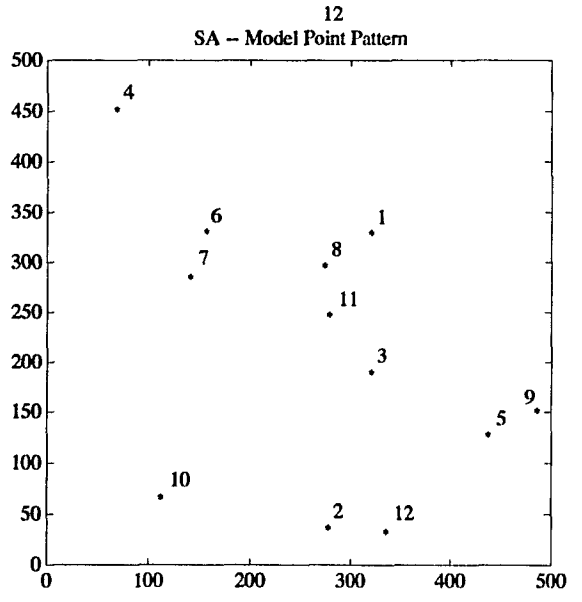

Figure 3.11 Case 3: Model Point Pattern

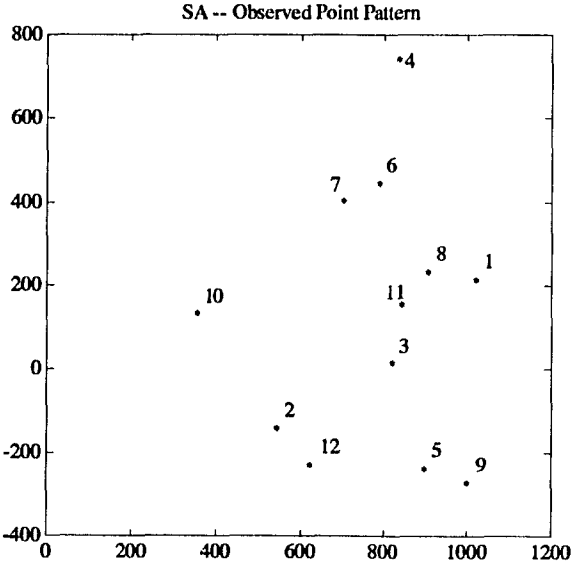

Figure 3.12 Case 3: Observed Pattern = T[Model Pattern] + Noise

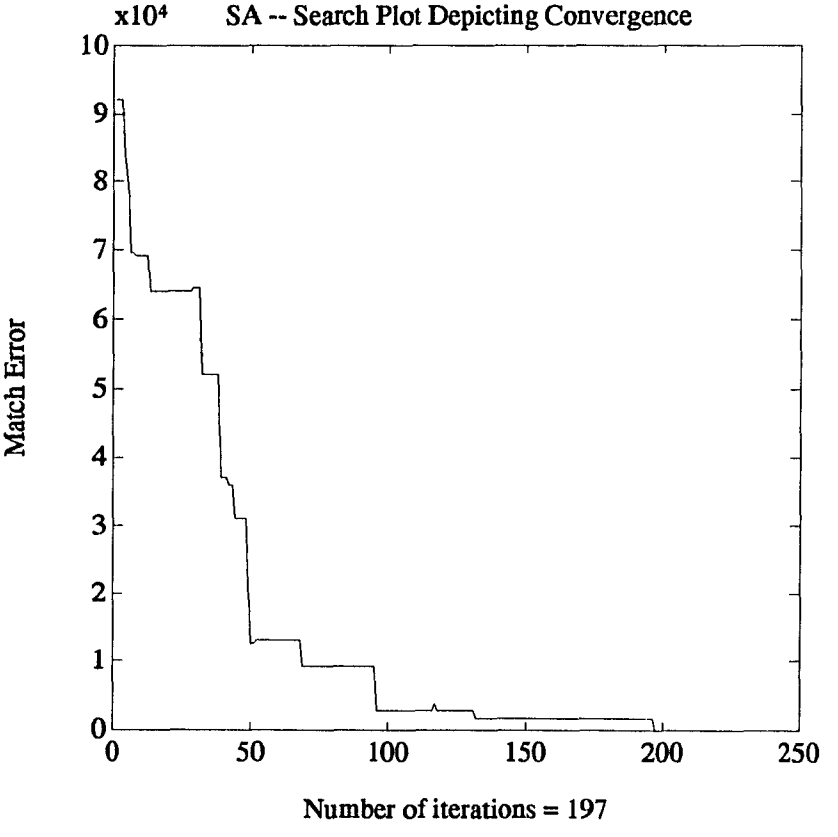

Figure 3.13 Case 3: Plot Showing the Error Convergence with Each Iteration for the Noisy and Complete Pattern Set.

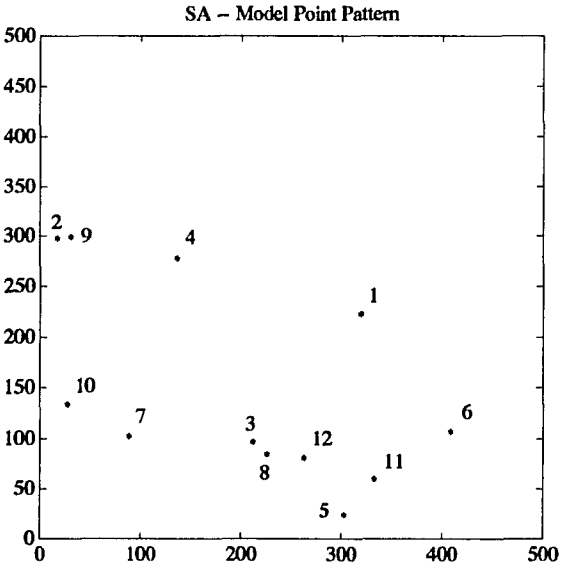

Figure 3.14 Case 4: Model Point Pattern

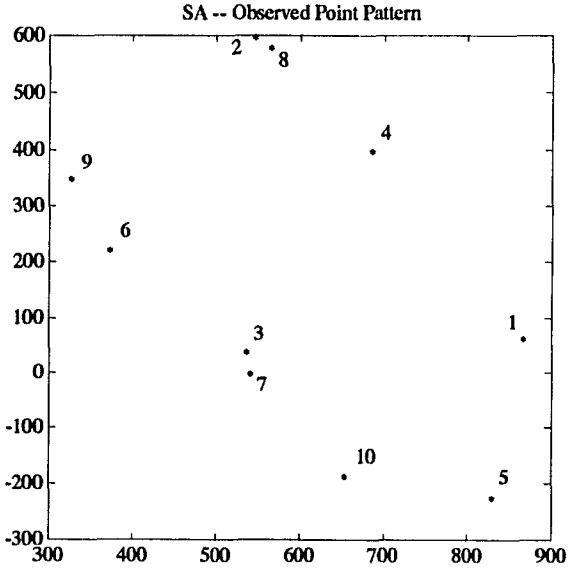

Figure 3.15 Case 4: Observed Pattern = T[Model Pattern] + Noise - 2 Points

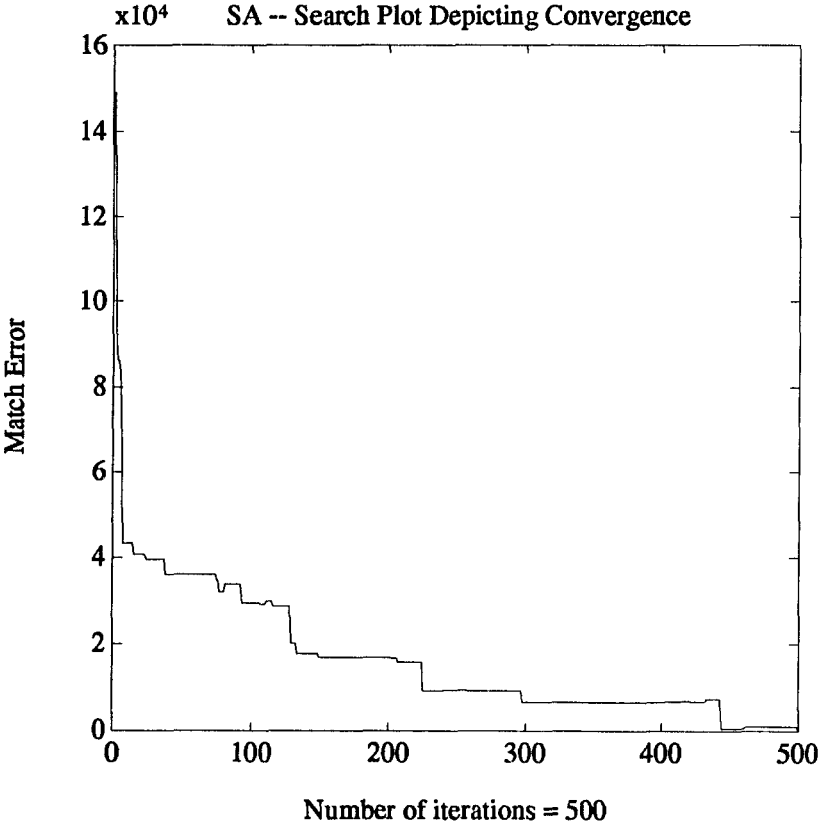

Figure 3.16 Case 4: Plot Showing the Error Convergence with Each Iteration for the Noisy and Incomplete Pattern Set.

3.4 Summary

A heuristic method for point pattern recognition is introduced. The robustness (ability to carry out the matching efficiently even in case of *noisy* or/and *incomplete* observed point sets) and fast convergence of the algorithm is established through the presented results. As mentioned earlier, the SA algorithm yields results that are close to the global optima. Exhaustive search would surely get to the global optima, but it would require a total of

$$\sum_{i=0}^{m} \binom{m}{m-n+i} \binom{n}{n-i} (n-i)! \quad \text{for } n < m,$$

and

$$\sum_{i=0}^{m} \binom{n}{m-i} \binom{m}{i} (m-i)! \quad \text{for } n \geq m \tag{3.7}$$

search nodes to be evaluated. Hill-climbing, on the other hand, depends on the starting point of the search space, and the search herein is almost always not able to converge to an optimal (or even close to optimal) global minima, as it always gets *stuck* in a local minima, there being no mechanism to escape the same in this algorithm.

CHAPTER 4

EVOLUTIONARY PROGRAMMING

4.1 Introduction

Evolutionary Programming stemmed as a result of limitations inherent in the theory of *classical* Genetic Algorithms. [1] As will be made clear shortly, GAs *evolved* into Evolutionary Programs as a result of multiple modifications, incorporating domain knowledge with a view to performance enhancement of the former. To understand and appreciate the mechanism of EPs, one needs to recognize the strengths and weaknesses of GAs. The following sections introduce the reader to the practical (implementation) aspects of the GA, with comments on its shortcomings. Where appropriate, the EP approach as applied to our task of point pattern matching, will be introduced and built upon. A detailed discussion of the mathematical foundations of GAs can be found in [26]. It should be mentioned that adequate theoretical basis for the theory of EPs is not established as of this work [27],

> "... On the other hand we have to admit the poor theoretical basis of evolutionary programs."

nevertheless, EPs outperform GAs significantly, as will be evident from the results presented herein.

4.2 Genetic Algorithms - Precursors to Evolutionary Programs

As opposed to conventional search techniques which evaluate potential search nodes from the solution space sequentially, a genetic algorithm evaluates them simultaneously; that is to say, genetic algorithms are parallel operationally and algorithmically. They [26] were first introduced by John Holland, his colleagues, and his students at the University of Michigan. They are a class of general purpose (domain

[1] The term Genetic Algorithm (GA) refers to *classical* Genetic Algorithms, unless stated otherwise.

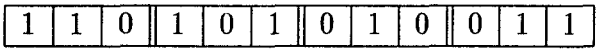

Figure 4.1 *GA*– Coded String: Binary Representation of Integer Values

independent) search methods which strike a remarkable balance between exploration and exploitation of the search space. Herein, each search node or potential chromosome is represented as a string, and with subsequent iterations search nodes are arrived at by using genetic operators on these strings. A search node replaces the previous one if it is judged as a better chromosome. Finally, after many such iterations one arrives at an optimal search node which is taken to represent the best assignment. This technique is based on the mechanics of natural selection and genetics combining the notion of survival of the fittest; random and yet structured search; and parallel evaluation of search nodes in the solution space. A genetic algorithm consists of

- a coded binary string representation (*genes*) of the search node in the solution space,
- an evaluation function that plays the role of the environment, rating chromosomes in terms of their "fitness,"
- fitness function to evaluate the search nodes,
- a set of binary genetic operators for generating new search nodes, and
- a stochastic assignment to control the genetic operators.

Suppose, we want to represent values 0 to 4. Consider an arbitrary binary string representation, as in Fig. 4.1. Each group of three bits represents an integer value, namely, 6, 5, 2, and 4. We see that *illegal* codes are inherent in this scheme and it would need repair algorithms to be able to work efficiently. *The coding of the problem often moves the GA to operate in a different space than that of the problem*

itself. The application of genetic operators may again give rise to strings outside the solution space, requiring recursive use of repair algorithms which results in wasteful computational resources.

In 1985 De Jong remarked [51]:

> "What should one do when elements in the space to be searched are most naturally represented by more complex data structures such as arrays, trees, digraphs, etc. Should one attempt to 'linearize' them into a string representation or are there ways to creatively redefine mutation and crossover to work directly on such structures. I am unaware of any progress in this area."

Realizing the above major shortcoming of GAs, reports citing implementations of modified GAs appeared, e.g., "A Modified Genetic Algorithm ... " [52], "Specialized Genetic Algorithms ... " [53], "A Non-Standard Genetic Algorithm ... " [54], paving the way for a technique coined as Evolutionary Programs.

Evolution programs can be perceived as custom genetic algorithms. Classical genetic algorithms operate on fixed length binary strings, which need not be the case for evolution programs. Also, evolution programs usually incorporate a variety of "genetic" operators, whereas genetic algorithms use binary crossover and mutation. The major factor behind the failure of GAs is the same one responsible for their success: domain independence. EPs, by virtue of being domain dependent, result in a better constraint satisfaction. The construction of an EP framework for any problem can be separated into four distinct and yet related tasks:

- choice of the data-structure which represents the solution domain most naturally and efficiently,
- the design of problem specific, knowledge-based genetic operators incorporating environment information with each iteration,
- choice or formulation of a robust fitness function enabling efficient evaluation of potential chromosomes, and

- preferably, dynamic determination of the stochastic parameters controlling the genetic operators in order to 'explore' and 'exploit' efficiently.

The algorithm for an evolutionary program may be depicted [27] as follows:

The structure of an Evolution Program

```
procedure evolutionary programming
    begin
        t ← 0
        initialize P(t)
        evaluate P(t)
        while (not termination-condition) do
        begin
            t ← t + 1
            select P(t) from P(t − 1)
            recombine P(t)
            evaluate P(t)
        end
    end
```

Evolution Programs outperform other traditional algorithms because:

- they work directly on the data-structure of the problem domain and not on any coding or linearization of the same
- they search from a genetically-rich population of chromosomes rather than evaluating genetically-poor population as in the case of GAs or individual solution points as in the case of linear algorithms
- they effectively use payoff (objective function) information built into the knowledge-based genetic operators

- they use probabilistic and knowledge-based transition rules and not arbitrary deterministic rules.

4.2.1 The Point Pattern Matching Procedure

The core of any Evolutionary Program consists of the following necessary steps as implemented herein (detailed descriptions are provided in the following subsections):

1. *Initialization of the starting population:* The pool of an initial population of the search nodes is randomly generated.

2. *Cost evaluation of strings in the population:* The fitness function determines the fitness of each chromosome in the population space.

3. *Reproduction of offspring:* Based on the fitness values of strings in the population, a population of strings is produced as outlined in the selection strategy for EP.

4. *Re-combination of reproduced offspring:* The reproduced population then undergoes a subsequent re-combination using genetic operations such as crossover, mutation and inversion.

5. *Convergence criteria:* Steps (2)–(4) are repeated until convergence or a predefined number of generations have been reached.

From the above description, we can see that the notion of survival of the fittest, passing *good* genes to the next generation of strings, and combining different strings to explore new search nodes are present in an evolutionary program.

4.3 Implementation of EP to Point Pattern Matching

To map the task of pattern matching onto the EP framework, we need to define a representation scheme, a fitness function, a set of genetic operators, and the rules to control the genetic operators.

4.3.1 Representation of Solution Space

As we are interested in finding the best assignment, a data-structure depicting assignments between the model points and the observed points forms the most ideal representation in that it:

- is a simple and direct labeling and assignment in the problem domain and a compact (minimum storage space) representation of the pattern points,
- is robust as it spans the entire range of all possible assignments and hence the solution space, without the possibility of illegal assignments, with the natural provision of a null label in case of incomplete pattern sets, and
- can be efficiently operated on by genetic operators yielding only valid assignments, thereby avoiding the need for computationally expensive repair algorithms.

Thus by using a string representation as explained in section (3.2.1), Fig. 3.1, each assignment is depicted by a label; that is, as discussed for the case of SA, each string indicates which of the model points are assigned to the observed points, and vice-versa.

4.3.2 Population Size

Theoretically, the number of chromosomes available at each iteration should be infinite in order to realize the highest degree of operational parallelism of the evolution program. Practically, this is impossible and a reasonable balance has

to be struck between the number of such available chromosomes and computational resources. Moreover, desired are the efficient utilization of the available population size by avoiding overcrowding, and maintaining a pool of chromosomes which is significantly exploratory in nature initially and evolves into an exploitative search of the solution domain. In our algorithm, the size chosen is 30. Efficient utilization of this population space is brought about by an efficient reproduction of the chromosome strings based on deterministic, knowledge-based, stochastic and heuristic genetic operators.

4.3.3 Initial Population

The initial population is a set of chromosomes generated by randomly meeting the one-to-one mapping constraint and it contains both good and bad strings (assignments). The population size is chosen to be $N = 30$.

4.3.4 Fitness Function

A customized fitness (objective or cost) function is necessary to ensure efficient utilization of computational resources and quick convergence to a global optima. For our problem it should incorporate the error (based on the optimal parameters) due to improper assignments and also reflect the existence of incomplete data sets. Furthermore, a good assignment should yield high fitness values and vice-versa. The match error as defined earlier in equation (2.8) meets all the above requirements and is used as the fitness evaluator. A fitness value that is inversely proportional to the match error is used.

4.3.5 Reproduction

After each iteration the solution set undergoes a selection process based on the principal of survival of the more fit individual, whereby good chromosomes are chosen to contribute their gene inherited knowledge to form potential chromosomes for the

next generation after undergoing re-combination. This selection process in a GA is quite straightforward wherein the selection is performed as follows:

- Normalize the fitness value of each string such that the sum of the fitness values of all the strings in the current population equal 1.
- Partition a unit-length scale into N (the number of population size) slots, each slot size in proportion to the normalized fitness value of a string in the current population.
- Generate N random numbers ranging from 0 to 1 and see where the number falls on the scale. The string corresponding to the division where a random number falls is selected to be a member for the new population.[2]
- The best chromosome is always passed on to the next population pool, deterministically with a probability 1.

The above technique has its drawbacks. A chromosome whose fitness is considerably higher than the rest of the population stands a good chance of being selected many times according to the above mechanism, thereby overcrowding the population space and also narrowing down the search space considerably as its repetitive selection (or the repetitive selection of a few of its kind) would force out potential promising chromosomes from the population pool, thereby throwing away useful genetic information! Furthermore, a large number of the same set of chromosomes results in wasteful use of computational resources (as the information utilization factor from a good chromosome is not infinite) along with an effective reduction in the population space utilization; infinite population size being one of the requirements of the theoretical foundations of the GAs.

[2]Note that strings which have higher fitness values and hence good gene representation are more likely reproduced in the new population.

To overcome these aforementioned shortcomings of genetic algorithms, the evolutionary programming technique presented herein makes use of a particular mechanism to bring about efficient offspring reproduction. Not only does the following remedy the previous mechanism's shortcomings [27], but it also has its advantages that contribute significantly to faster convergence leading decidedly to a global optima:

1. Select r (not necessary distinct) chromosomes from $P(t)$ as parents for reproduction. This selection is based on the selection technique described above utilizing relative fitness. This brings in r chromosomes with good genetic information.

2. Select r (distinct) chromosomes from $P(t)$ to die (independent of the above selection). This is achieved by deterministically choosing the best $(N - r)$ individuals from the population and passing them on to the reproduction pool and leaving the other r chromosomes to die.

3. From this reproduction pool of N strings obtained from $P(t)$, as a result of the above steps, the r parents from the first step are now recombined (by being operated upon exclusively by one of the genetic operators) to yield r offspring.

4. The new population $P(t + 1)$ is then formed from the $(N - r)$ chromosomes from the second step and the r offspring from the third step. This completes the selection process of the new population, an important preprocessing step before re-combination. The advantage of this technique is providing for re-combination r offspring generated from highly fit parents, each offspring different from the other in some manner. This mechanism results in formation of offspring which may have a slightly higher value than the parents and yet be passed on to the next generation pool for re-combination. This may be taken as analogous to the concept of occasionally accepting a chromosome

with higher fitness, as in simulated annealing. This concept is crucial to fast convergence.

Furthermore, by virtue of the second step, the best chromosome is always passed on to the new generation along with an $(N - r - 1)$ number of highly fit individuals. From the above discussion it is clear that this population space is richer in genetic information and has a greater potential to produce offspring, thus leading decidedly to global optima.

4.3.6 Genetic Operators

The genetic or the re-combination operators control the means by which new information is formed and also existing information exchanged between chromosomes to facilitate their subsequent evolution into chromosomes with higher survival probability with every iteration. For the EP algorithm, three such operators are used, which facilitate a fast evolution leading to quick and optimal convergence. It is a point to observe that all operators satisfy the constraint that the mapping is single and there is no need for repair algorithms. Each of these will now be discussed in detail.

4.3.6.1 Mutation In a GA the concept of mutation refers to *randomly* flipping *each* bit of a chromosome to one or zero. Moreover, the probability of mutation is kept low since its random nature and successive implementation on each gene of the chromosome serves more to explore the solution space rather than exploit it. Contrary to the above notion the mutation operator implemented herein operates once on a chromosome on a *single* gene, mostly chosen *deterministically* based on the knowledge extracted from the fitness computation of each gene of the assignment. Thus, the role of this operator is more that of exploitation rather than exploration; it is then implicit that this knowledge-based operator be applied with high probability.

Furthermore, as mutation herein is only on a single gene basis per chromosome, as opposed to generating a probability and testing every bit for mutation as in GA, the former is much more efficient and faster. Moreover as the string size grows (for real-world cases) the latter technique is computationally expensive. There are three types of mutation operators defined for our problem and each is used with a varying degree as explained below.

Mutation-1 Herein the gene contributing the most to the square error of the chromosome, as per equation (2.5), is chosen to undergo mutation; it is exchanged with the gene with the next highest contribution to the overall square error. This is deterministic and is always performed on the best chromosome of the population, at every iteration. It is also performed with some probability on the other chromosomes within the population. The rationale behind this can be explained by considering the following example. Suppose after a number of iterations we have the string shown in Fig. 4.2. It

1	2	3	4	5	9	7	8	6	10	11	12

Figure 4.2 Solution Depicting the Need for Mutation-1 Operator

is evident that the genes in positions nine and six need to be mutated in order to arrive at the optimal match. This is efficiently achieved by the technique described above as only these genes are likely to contribute most to the square error. Had conventional mutation been carried out on this string, the probability that it selected for swapping, the sixth gene followed by the choice of the ninth gene, would be very low. Furthermore, convergence would result only if the mutation operator left the other genes untouched. Thus, it is evident that a large number of iterations may be required for convergence, whereas in mutation-1 convergence would be obtained in the next iteration with proba-

bility one. Therefore, the contribution of this operator to convergence speed is quite significant prior to the stages of achieving global optima.

Mutation-2 The selection of the first gene remains the same, as in the case of mutation-1. The second gene is chosen randomly from the chromosome string and is mutated with the first. The need for this version of the operator arises wherein a gene with the highest error contribution needs to be mutated with a gene other than the one with the next highest error contribution. This is possible when three or more genes are not in their proper positions. This operator may also be devised such that the second gene selected is the one with the third highest error contribution. In fact, one could extend this principle to include all combinations of the genes with high contributions to the square error. The search would then be faster. For our illustrative case of 12 points, this extension has not been implemented as a random pick of one right gene is highly possible considering the population size. This technique may yield significant improvement in real-world cases wherein the strings are large and t he knowledge and successful manipulation of high error contributing genes might result in a significant saving of computational resources.

Mutation-3 It may happen that mutating the highest error contributing gene with any other gene will not result in a proper assignment and increase significantly the overall square error. This situation would render the above two variations useless and the solution would get stuck in a local optima. The mutation-3 version of mutation remedies this situation by selecting both the genes randomly for mutation. Thus, its contribution is more of an exploratory nature rather than to exploit the solution space; hence its probability of execution is kept quite low. Nevertheless, it plays an important role, as it provides a mechanism to escape from such apparent local optima.

4.3.6.2 Uniform Crossover The classical *Crossover* operator is an effective tool when the genetic order in a chromosome needs to be preserved. Digressing briefly from the topic of point pattern matching to the dilemma of The Travelling Salesman, wherein a salesman needs to visit each city exactly once and return to the starting point, it is the order of the cities that is important and not their respective positions as illustrated by identical tours.

1	3	2	5	4	7	8	6

4	7	8	6	1	3	2	5

Figure 4.3 Identical Tours of a TSP

It should be clear, then, that crossover results in effectively combining good partial tours with lower costs from two or more complete tours to form another complete tour with a lower anticipated cost. Returning back to our task of point pattern matching, it should then be implicit that a high crossover rate will result in poor performance, as here the genetic positions and not the genetic order is important. This is implicit in the representation wherein, for example, a value of 5 in cell 7 means an assignment of the 5th observed point to the 7th observed point and hence is position dependent. However, each positionwise mapping does form a complete assignment, based on which the square error is computed, and this aspect may make it seem that the individual error is dependent on the relative order of the points. This is not true because preserving the partial genetic order and changing their positions results in a fitness change which provides a mechanism of changing connectivity order along the length of the chromosome; a useful concept in The Travelling Salesman Problem but an useless one for our task. Furthermore, with the age of the population it is noticed that the individual genes need to be exchanged with other individual genes, and not an ordered group of genes with another ordered

group of genes. Thus, the importance of the crossover operator is not pronounced for this problem.

However, in large solution representations, where the points are quite close by or cluttered in groups, a clustering of neighboring points might occur (as the low Euclidean distance between them results in an apparently optimal set of transformation parameters) leading to local minimas. The crossover operator may then be useful, but these cases are rare and usually don't survive long enough to hamper the convergence speed. In fact, as soon as the central gene from the cluster changes position, the entire solution shifts to a different search space, driven by the drastic change in the optimal transform parameters.

It is interesting to note that when incomplete data sets are being considered, situations do arise where the positional assignments of a group of genes appears shifted by one or two gene positions (depending on the number of missing points) and the use of the crossover operator seems promising, but then choosing the correct crossover site randomly is a very low probability. Moreover, the determination of the fact that some genes have shifted assignments is not possible because no *a priori* knowledge of the sequential order of the pattern sets is provided to the algorithm. Thus, interestingly, even though this situation is evident to an observer viewing the iterations, not much can be done to remedy it.

The above discussion of the crossover operator makes it clear that any segment-based operator (or order-preserving operator, for that matter) is futile and one should design a problem specific, position-based crossover that results in positionwise-genetic-information interchange among chromosomes. Based on the concept of generalization of a multi-point crossover [55] [56], a problem specific uniform crossover is defined as follows:

- select two strings for crossover based on the probability of the crossover operator

- for each gene position in the 1st chromosome, if the error contribution of the gene is *high* relative to the other genes, and the error contribution of the corresponding gene in the 2nd chromosome is *low* relative to the other genes of this chromosome, copy the gene from second chromosome into the corresponding position of the first chromosome; simultaneously replace the original location occupied by the second chromosome gene in the first chromosome by the replaced gene (to preserve one-to-one mapping)

- repeat the above step for the 2nd chromosome

This crossover generates two offspring from two parents *preserving the good gene positions and overwriting the bad gene positions with the good gene positions from the other chromosome.* As is evident, this form of crossover can be very powerful since it can replace multiple gene positions leading decidedly to a lower error value. Its use in real-world application would significantly improve the convergence speed and with some fine tuning of the error comparisons, this operator can also be used effectively to escape local minimas, as the mechanism of overwriting a higher error contributing gene with a lower error contributing gene would surely result in an improvement of the resulting offsprings. According to [27] it has been emphasized that the role of the mutation operators is stronger citing [57] than that of the crossover operator to the point of ignoring it altogether, but it seems that such a generalization is not applicable since in any EP the operators are most useful when they embody knowledge-based domain information, and because of their problem dependent nature their efficient design depends on the implementor of the algorithm.

4.3.6.3 Inversion The *Inversion* operator brings about a complete change in the search space being traversed as it carries out an $(n + 1)$ complement of the chromosome string it operates upon. It is best to shut off the probability of this operator with the age of the population (provided that a certain meaningful fitness value has been obtained); the use of this operator would then render the previous iterations futile, as it is analogous to generating a new search node altogether without the concept of evolution from the previous iteration. However, the use of this operator can be quite meaningful if the solution seems stuck for over a large number of iterations. Moreover, based on simulations, it is somehow felt that this operator would prove useful if there exists some geometric axis-symmetry between the pattern sets resulting in a mirror image assignment for near points, resulting in a local optima.

4.4 Simulation Results

Experimental results demonstrate that the EP technique highly outperforms all other conventional algorithms in convergence speed. This is because of the exploitation of domain-knowledge by the genetic operators and the elimination of prohibitively inefficient computing environments inherent in other algorithms. The results tend to substantiate the robustness of the proposed approach using EP for point pattern matching, as even with high error and missing points the convergence speed does not undergo significant degradation. The new versions of mutation and crossover operators defined are the instrumental factors contributing to the speed of the algorithm. A large number of simulations, which can be categorized into several cases, has been carried out. For *illustrative purposes*, results using four point patterns sets, each consisting of 12 points are shown and tabulated. Average results have been presented; best cases have iteration values around 6 and worst cases go up to iteration values of around 30 (provided the solution does not get stuck in a local minima).

Case	Iterations	Matching	Error
(1) *clean* and *complete*	8	100%	0
(2) *clean* and *incomplete*	11	100%	0
(3) *noisy* and *complete*	6	100%	43.62
(4) *noisy* and *incomplete*	12	100%	90.6721

Table 4.1 EP Results Compared for the Mentioned *Incomplete* and *Noisy* Cases

We consider four cases, namely:

- Case 1: Observed Pattern = T[Model Pattern].
  Scale factor = 3; Rotation = $\pi/4$; translation in the x-axis = 100;
  translation in the y-axis = 300.

- Case 2: Observed Pattern = T[Model Pattern] - n points.
  Scale factor = 3; Rotation = $\pi/4$; translation in the x-axis = 100; translation in the y-axis = 300; $n = 3$; 2nd, 5th and 10th point is missing.

- Case 3: Observed Pattern = T[Model Pattern] + noise.
  Scale factor = 3; Rotation = $\pi/4$; translation in the x-axis = 100; translation in the y-axis = 300; Gaussian noise with zero mean and variance equal nine is introduced into the observed point pattern.

- Case 4: Observed Pattern = T[Model Pattern] + noise - n points.
  Scale factor = 3; Rotation = $\pi/4$; translation in the x-axis = 100;
  translation in the y-axis = 300; Gaussian noise with zero mean and variance equal nine is introduced into the observed point pattern; $n = 3$; 2nd, 10th and 11th point is missing.

The following pages present the plots of model and observed points followed by the convergence and error plots for each of the above cases. They should be self explanatory.

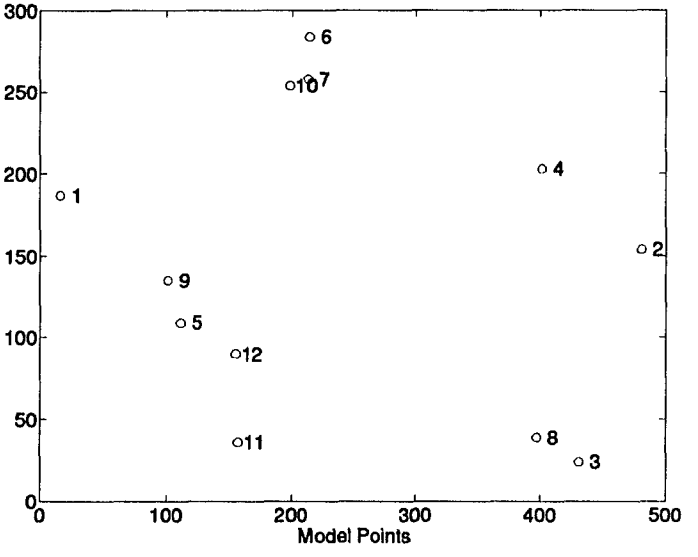

Figure 4.4 Case 1: Model Point Pattern

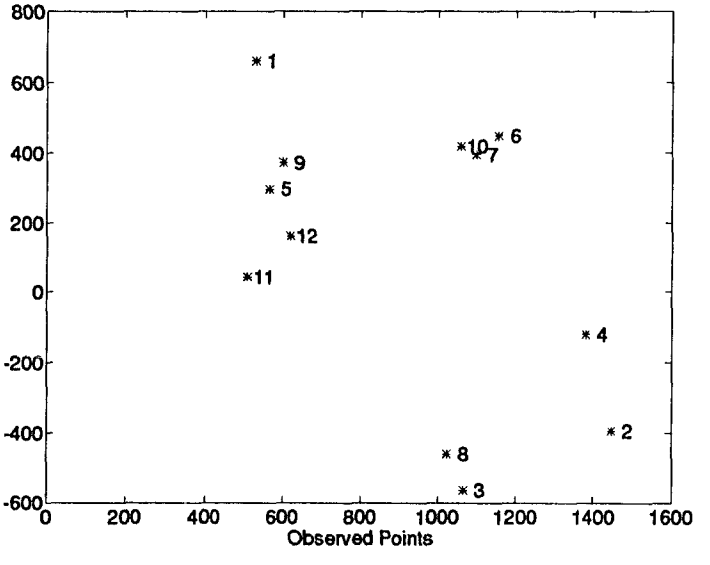

Figure 4.5 Case 1: Observed Pattern = T[Model Pattern]

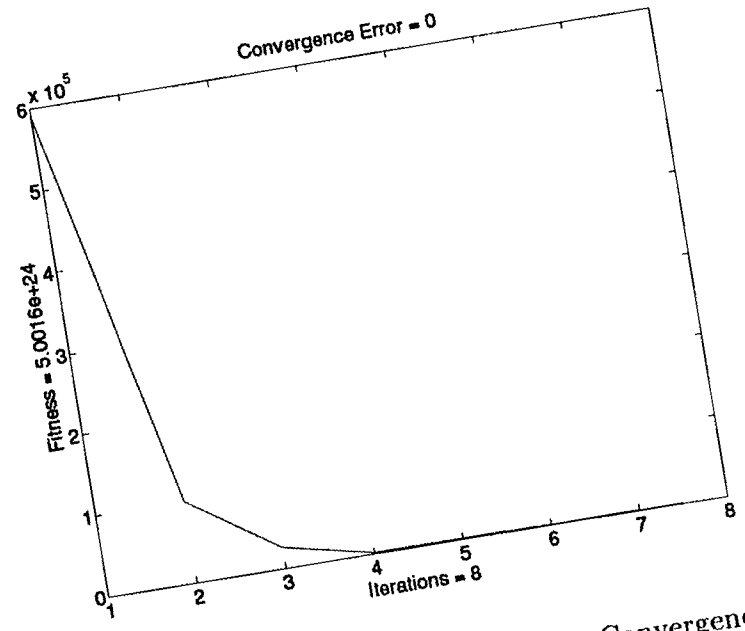

Figure 4.6 Case 1: Plot Showing the Rapid Error Convergence with Each Iteration

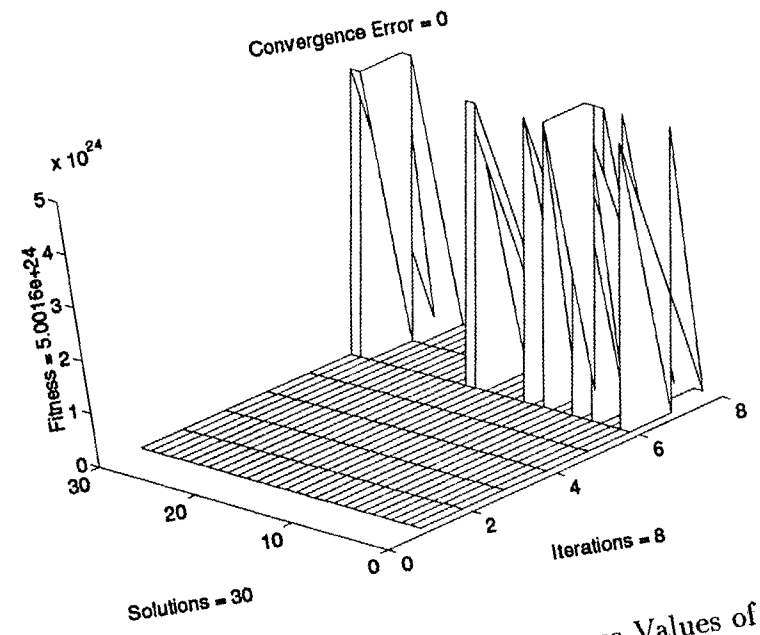

Figure 4.7 Case 1: Mesh Plot Showing the Fitness Values of Solutions for Each of the 8 Iterations

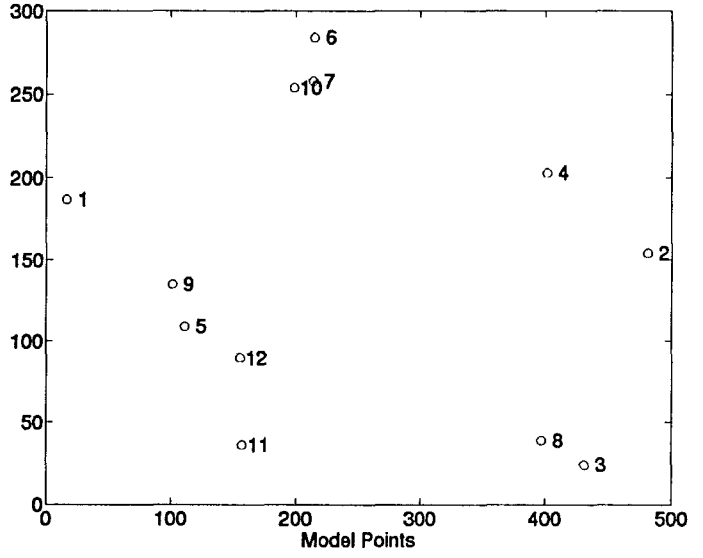

Figure 4.8 Case 2: Model Point Pattern

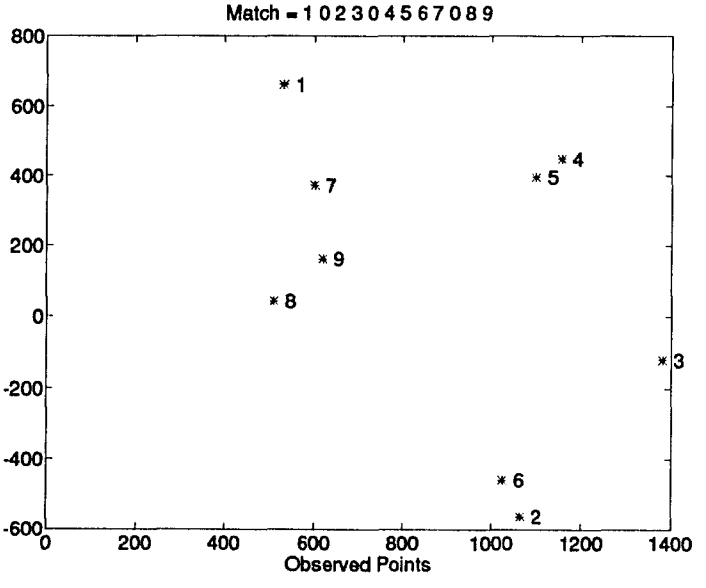

Figure 4.9 Case 2: Observed Pattern = T[Model Pattern] - 3 Points

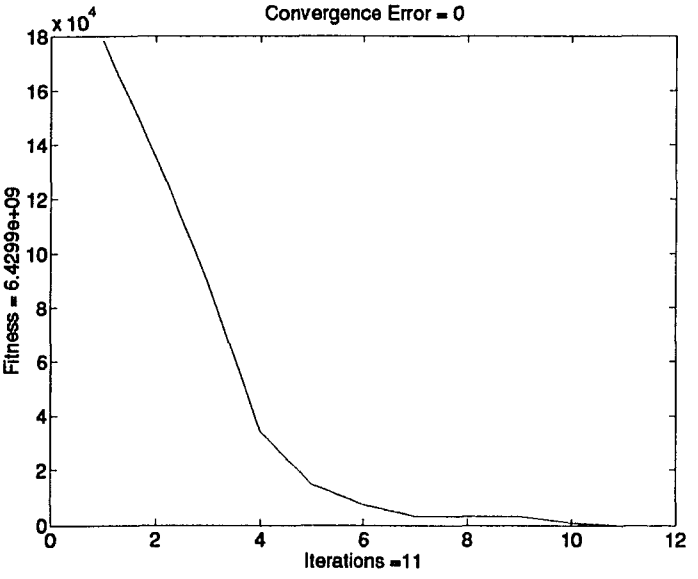

Figure 4.10 Case 2: Plot Showing the Rapid Error Convergence with Each Iteration

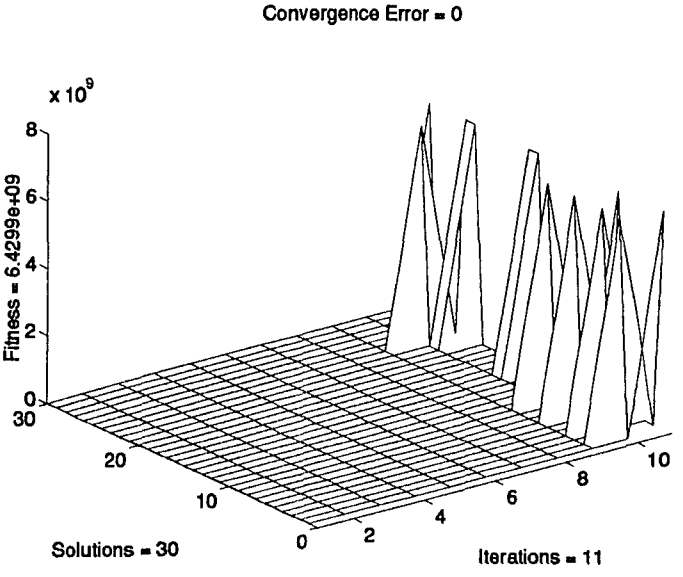

Figure 4.11 Case 2: Mesh Plot Showing the Fitness Values of Solutions for Each of the 11 Iterations

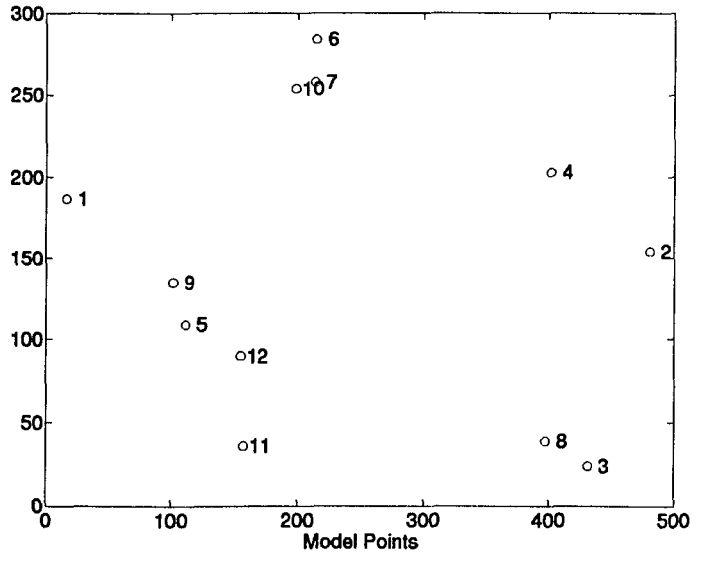

Figure 4.12 Case 3: Model Point Pattern

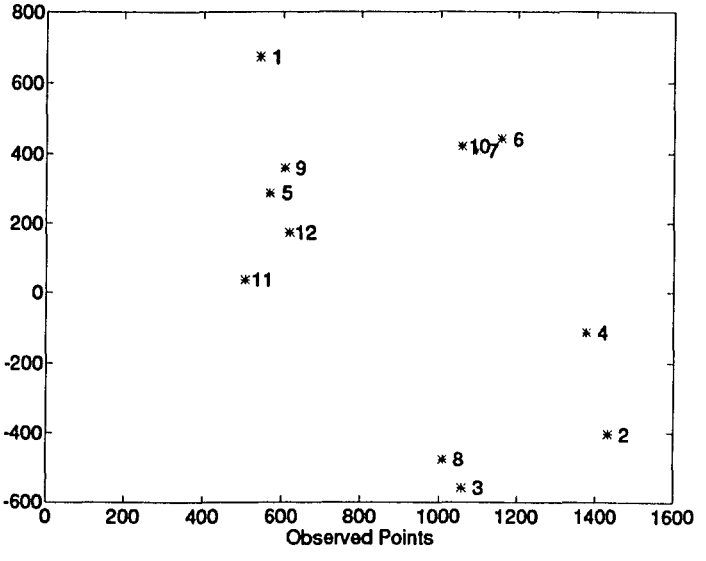

Figure 4.13 Case 3: Observed Pattern = T[Model Pattern] + Noise

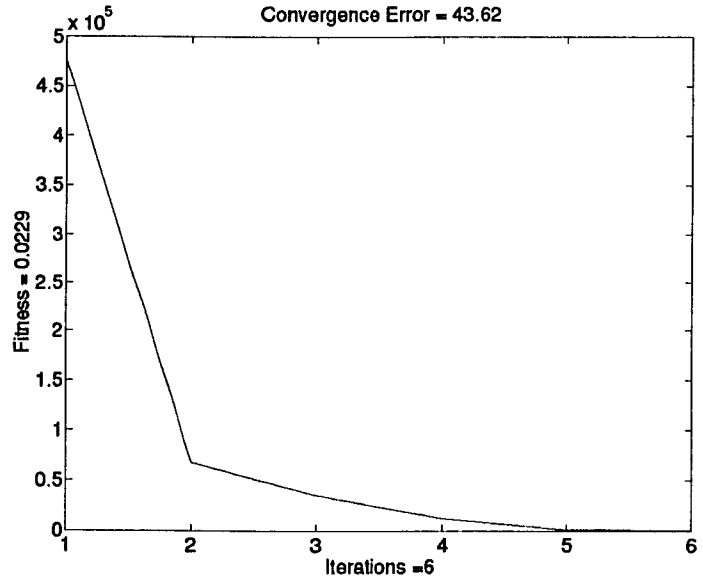

Figure 4.14 Case 3: Plot Showing the Rapid Error Convergence with Each Iteration

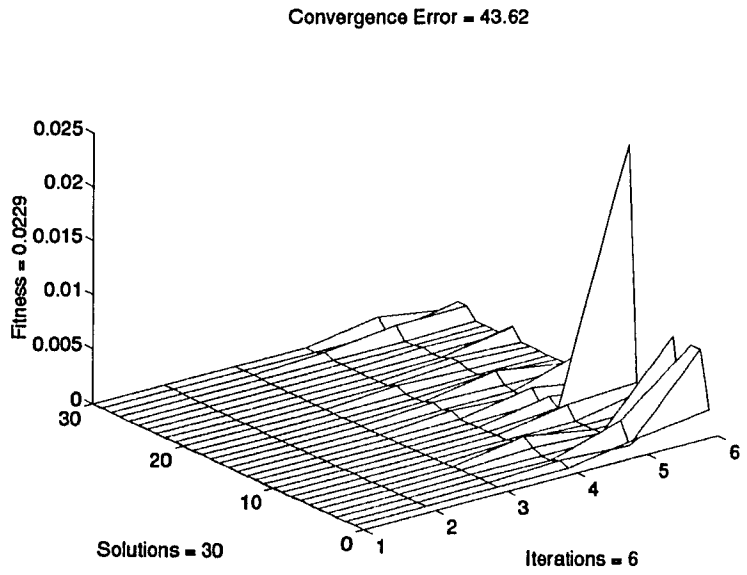

Figure 4.15 Case 3: Mesh Plot Showing the Fitness Values of Solutions for Each of the 6 Iterations

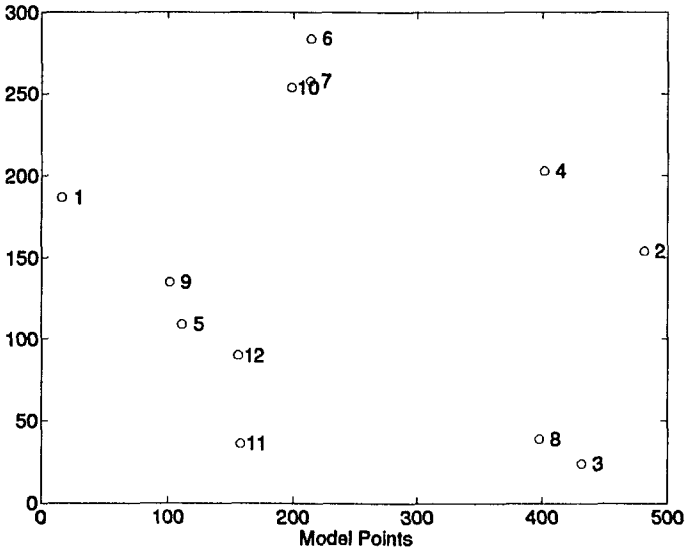

Figure 4.16 Case 4: Model Point Pattern

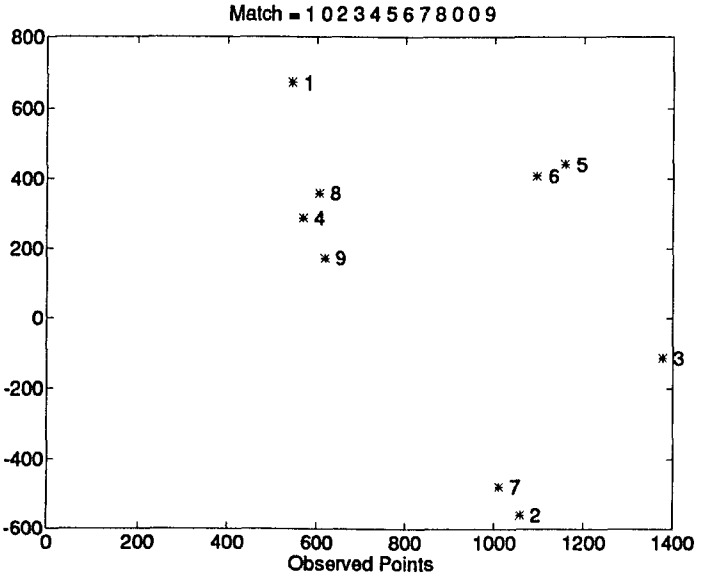

Figure 4.17 Case 4: Observed Pattern = T[Model Pattern] + Noise - 3 Points

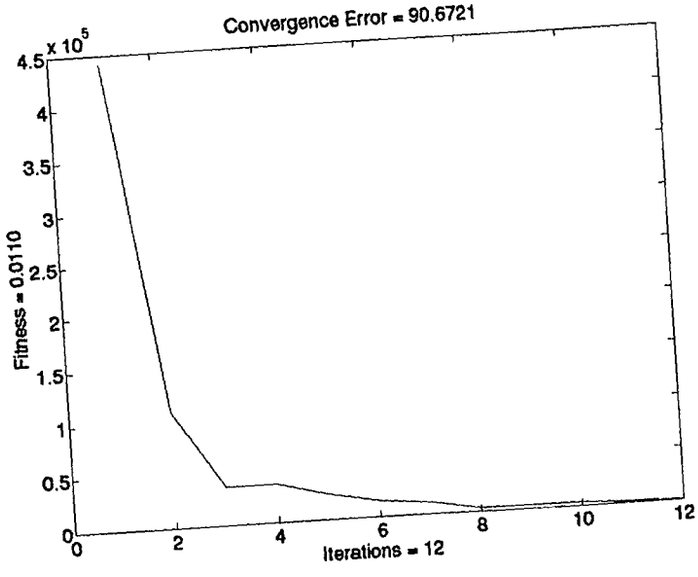

Figure 4.18 Case 4: Plot Showing the Rapid Error Convergence with Each Iteration

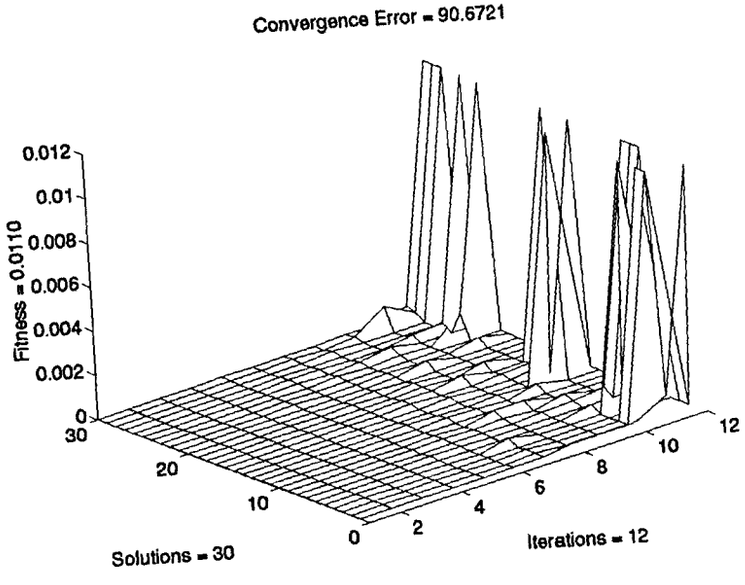

Figure 4.19 Case 4: Mesh Plot Showing the Fitness Values of Solutions for Each of the 12 Iterations

4.5 Summary

A highly robust, knowledge-based and efficient technique has been introduced as a new approach to point pattern recognition. This technique has the uncanny ability to perform without significant degradation in the presence of noise and in the absence of complete pattern sets. Additionally, pattern sets with a higher number of model points have been worked on and the results are quite interesting. As mentioned in the previous sections, the involvement of some otherwise lesser used genetic operators becomes quite significant as the data set increases. The robustness and fast convergence rates speak highly of the proposed algorithm. To find the *best* match between a set of m model points and a set of n observed points, the algorithm only needs to evaluate $N \times G$ search nodes where N is the population size and G is the number of generations at which the algorithm converges or stops. It should be noted that not all search nodes are unique as multiple copies of a good chromosome do exist due to the very nature of the evolution program.

There is no unique way of predicting the probabilities of the reordering (crossover), mutation, and inversion operations. The most efficient use of these operators would result from dynamic variation of the operation probability. This gives a greater control over the utilization of the environment knowledge with increase in the age of the population. The probability is best determined heuristically and also depends on the strength and the capability (the amount of knowledge used effectively) of the operator in question. The results presented herein are from an illustrative point of view only and the algorithm is not limited to 2-D pattern sets or the similarity transformation. It can be applied to n-dimensions and to any transformation.

CHAPTER 5

MEAN FIELD ANNEALING

5.1 Introduction

The Hopfield neural network is a recurrent network that has been successfully applied to many optimization problems [28]. Using the Hopfield neural network to solve an optimization problem involves two major tasks:

1. Constructing an appropriate *Hopfield Energy Function* for the problem.

2. Adopting and designing a recursive mechanism for minimizing the energy function.

The construction of the energy function should incorporate all aspects (*i.e.*, *constraints*) of the solution domain, which, when met, tend to lower the energy of the function, along with the *cost*. This gives an indication of the feasibility of the potential solution. For the minimization of the *Hopfield Energy Function*, Mean Field Annealing (MFA) has been proposed [29] [30] and the approach has been demonstrated to be robust and efficient.

5.2 The Hopfield Energy Function

One of the most important contributions in Hopfield's pioneering work [28] [32] [33] is the concept of an energy function. Hopfield and Tank [28] formulated the Travelling Salesman Problem (TSP) to a highly interconnected neural network,[1] and made exploratory numerical studies on modest-size problems by minimizing the Hopfield energy associated with the network.

The most important property of an energy function is that it should decrease dynamically as the system evolves, and must be minimized once a stable or optimum state is reached.

[1]Neural networks are being considered as a good alternative for solving difficult optimization problems [31].

5.3 The Hopfield Neural Network

A simple Hopfield neural network is depicted by the block diagram shown in Fig. 5.1.

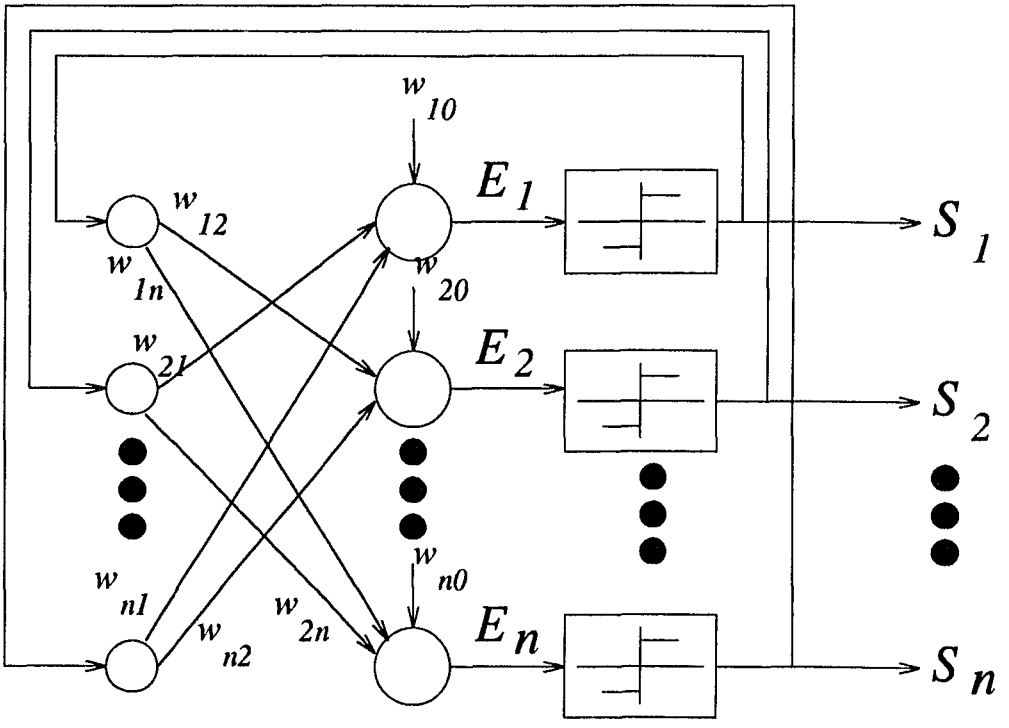

Figure 5.1 The Hopfield Neural Network Model

After each iteration,

- neuron excitation = weighted sum of its inputs,

$$E_j = \sum_{i \neq j} w_{ij} S_i + w_{j0}, \tag{5.1}$$

where E_j is defined as the jth component of the energy function; w_{ij} is the weight connection between neuron i and neuron j; S_i is the state of neuron i

- S_i, the output of a neuron, equals one if the weighted sum of its inputs is greater than a threshold w_{j0}, and zero otherwise:

$$S_j = \begin{cases} 1, & \text{if } E_j > 0, \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

- equilibrium is obtained when all the constraints (weights) are met

The Hopfield neural network can be operated in two modes. In the first mode, where the state of the neurons is assumed known, the corresponding weights of the connections are to be determined. In the second mode, where the connections among the neurons are given, the corresponding states of the neurons (i.e., some specific constraints or requirements are met) are to be determined. The second mode is used for our task.

5.4 Mean Field Theory

The mean field theory approximation is a well known technique in physics. It was first introduced for neural network applications by Peterson [37]. A brief description of the mean field theory, as relevant to our task, is given below. For an elaborate analysis, readers are referred to [29] [30] [36] [37] [38] [39].

The Mean Field Theory was derived incorporating the SA technique of thermostatic operation, which schedules the decrease of temperature and a relaxation process that searches for the mean value of the solution as the equilibrium solution, at each temperature. Thus the statistical mechanism of thermostatic operation is maneuvered according to the Boltzmann distribution,

$$B(S') = \frac{\exp[-E(S')/T]}{Z}, \tag{5.3}$$

where $S' \equiv$ one of the possible configurations

$E(S) \equiv$ the energy of the configuration

$T \equiv$ the temperature, and

$Z \equiv$ the partition function given by

$$Z = \sum_{S'} \exp[-E(S')/T], \tag{5.4}$$

where $\sum_{S'}$ denotes the sum over all possible neuron state configurations.

In the MFA theory, we consider the *means* of the neuron outputs; defined by

$$\begin{aligned} V_i &= < S_i > \\ &= 1 \cdot P(S_i = 1) + (0) \cdot P(S_i = 0) \\ &= P(S_i = 1), \end{aligned} \tag{5.5}$$

where the value, S_i, is 1 for a neuron that is *ON* and 0 for a neuron that is *OFF*; $P(S_i = 1)$ and $P(S_i = 0)$ are the probabilities for $S_i = 1$ and $S_i = 0$, respectively; and V is the *mean configuration* corresponding to S.

Then the Boltzman Distribution, in terms of *means*, can be expressed as:

$$P(V) = \frac{\exp[-E(V)/T]}{Z}, \tag{5.6}$$

as given by equation (5.3). The *Local Field* may then be defined as

$$h_i = -\frac{\partial E}{\partial S_i}, \tag{5.7}$$

wherein

$$S_i = \begin{cases} 1 & \text{if } h_i > 0 \\ 0 & \text{if } h_i < 0 \end{cases}. \tag{5.8}$$

The probability of any neuron S_i being "on" or "off" is given by

$$P(S_i = 1) = \frac{e^{h_i^{MFT}/T}}{e^{-h_i^{MFT}/T} + e^{h_i^{MFT}/T}} \tag{5.9}$$

$$P(S_i = 0) = \frac{e^{-h_i^{MFT}/T}}{e^{-h_i^{MFT}/T} + e^{h_i^{MFT}/T}}, \tag{5.10}$$

where

$$h_i^{MFT} = < h_i > = < -\frac{\partial E}{\partial V_i} > \tag{5.11}$$

wherein the *local field* h_i is approximated by its thermal average (*i.e., the mean field*).

Combining equation (5.5) and equations (5.9) (5.10) and (5.11), the neuron outputs may be determined from

$$V_i = \frac{e^{h_i^{MFT}/T}}{e^{-h_i^{MFT}/T} + e^{h_i^{MFT}/T}}, \tag{5.12}$$

or, equivalently,

$$V_i = \frac{1}{2}[1 + \tanh(h_i^{MFT}/2T)]. \tag{5.13}$$

5.5 Modeling the PPM Task onto the Hopfield Network

In brief, the construction of the framework for any problem consists of:

1. Defining an appropriate neuron encoding.
2. Constructing the Hopfield energy function which appropriately reflects the constraints and costs of the problem.
3. Deriving the mean field equations from the *Hopfield Energy Function.*
4. Selecting the Lagrange parameters in accordance with the significance of each component of the energy function.
5. Updating the mean field variables by annealing according to a cooling schedule.

The difficulty generally lies in mapping a problem onto a framework that is solvable by mean field annealing.

In order to employ the MFA algorithm, the task needs to be mapped onto a neural network. The output of the neurons of the network thus obtained is then monitored according to the problem specific *constraints* and *cost*, embodied in an *energy function.* The algorithm proceeds by minimizing this energy function and at the same time striving to maintain a stable output value for the neurons in the network, leading to a global optima. This optimal value of the neural net indicates a mapping between the model and the observed points such that the error

is minimum. To incorporate the above strategy, the neurons are encoded to represent the parameters used in modeling the energy function.

5.5.1 Neuron Encoding

A neuron, i.e., its output, is denoted by V_{ij}, where the subscript i corresponds to the ith cell (model point) and the subscript j denotes the corresponding cell value (observed point).

$$V_{ij} = \begin{cases} 1 & \text{if the } i \text{ th model point is matched with} \\ & \text{the } j \text{ th observed point,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.14}$$

The *Associative Matrix, A,* can then be defined as:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N_N} \\ a_{21} & a_{22} & \cdots & a_{2N_N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{N_N1} & a_{N_N2} & \cdots & a_{N_NN_N} \end{bmatrix}, \tag{5.15}$$

where

$$a_{ij} = \begin{cases} 1 & \text{if there is an assignment between the} \\ & i\text{th model point and } j\text{th observed point.} \\ 0 & \text{otherwise.} \end{cases} \tag{5.16}$$

For our problem the associative matrix, defined above, must meet certain constraints, which are described in section (5.6.1).

5.6 Formulation of the Energy Function

As mentioned earlier, the *Energy Function* is made up of the cost and constraint terms which, ideally, must characterize the solution space completely in order for fast convergence to a global minima.

5.6.1 Cost and Constraint Terms

The *cost* term is modeled as the weighted sum of all outputs V_{ij} times the cost C_{ij}:

$$E_0 = \sum_i \sum_j V_{ij} C_{ij}, \tag{5.17}$$

where V_{ij} is the *mean* of a neuron assigning the ith model to the jth observed point, and C_{ij} is the cost associated with the above defined assignment.

The cost C_{ij} is based on equation (2.5) and is given by the error matrix E:

$$E = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1N_N} \\ C_{21} & C_{22} & \cdots & C_{2N_N} \\ \vdots & \vdots & \vdots & \vdots \\ C_{N_N1} & C_{N_N2} & \cdots & C_{N_NN_N} \end{bmatrix}. \tag{5.18}$$

The following are the constraints characterizing the associative matrix completely:

1. The total number of neurons, V_{ij}, that may be ON equal to the number of *observed* points, $N_{ob.}$:

$$E_1 = (\sum_i \sum_j V_{ij} - N_{ob.})^2 \tag{5.19}$$

2. Each *model* point can be assigned to only one *observed* point:

$$E_2 = \sum_i \sum_j \sum_{j \neq j'} V_{ij} V_{ij'} \tag{5.20}$$

3. Similarly, an *observed* point, once assigned, cannot be assigned to any other *model* point:

$$E_3 = \sum_j \sum_i \sum_{i \neq i'} V_{ij} V_{i'j} \tag{5.21}$$

4. Lastly, the requirement that a neuron's output value be either 1 (ON) or 0 (OFF) is monitored by the following equation:

$$E_4 = \sum_j \sum_i V_{ij}(1 - V_{ij}) \tag{5.22}$$

The *Total Energy* is then given by

$$E = (\alpha \times E_0) + (\beta \times E_1) + (\gamma \times E_2) + (\kappa \times E_3) + (\zeta \times E_4), \tag{5.23}$$

where α, β, γ, κ, and ζ are the Lagrange parameters which serve to maintain a proportional contribution of each of the cost or constraint terms to the energy function. The choice of the Lagrange parameters is critical because otherwise not all constraints will be able to contribute proportionately to the energy function, thereby hampering the convergence speed and/or failing to achieve convergence to a global minima.

5.7 Evaluation of the Thermal Average

The *Thermal Average* may then be evaluated as

$$h_{ij}^{MFT} = < h_{ij} > = < -\frac{\partial E}{\partial V_{ij}} >, \tag{5.24}$$

where

$$\frac{\partial E_0}{\partial V_{ij}} = C_{ij}, \tag{5.25}$$

$$\frac{\partial E_1}{\partial V_{ij}} = 2(\sum_i \sum_j V_{ij} - N_{ob.}), \tag{5.26}$$

$$\frac{\partial E_2}{\partial V_{ij}} = \sum_{j' \neq J} V_{Ij'}, \tag{5.27}$$

$$\frac{\partial E_3}{\partial V_{ij}} = \sum_{i' \neq I} V_{i'J}, \tag{5.28}$$

$$\frac{\partial E_1}{\partial V_{ij}} = 2(\sum_i \sum_j (1 - 2 \times V_{ij}). \tag{5.29}$$

From the above, the *Total Thermal Average* may then be written as

$$\frac{\partial E}{\partial V_{ij}} = \alpha \frac{\partial E_0}{\partial V_{ij}} + \beta \frac{\partial E_1}{\partial V_{ij}} + \gamma \frac{\partial E_2}{\partial V_{ij}} + \kappa \frac{\partial E_3}{\partial V_{ij}} + \zeta \frac{\partial E_4}{\partial V_{ij}}. \tag{5.30}$$

5.8 Cooling Schedule

As is the case with SA, a cooling schedule specifying the initial temperature, the stopping criterion, the number of iterations at each temperature before moving on to the next temperature, and a temperature updating rule are required.

5.8.1 Initial Temperature

The initial temperature is preferably chosen just below the *critical temperature* where the energy function has a significant low value. The proper choice of this parameter is essential to preserve computational resources.

5.8.2 Stopping Criterion

The annealing process comes to an end when the following conditions are met.

1. *All* neuron outputs $\in$ [0.0,0.1] or $\in$ [0.9,1.0]; signifying that the outputs have converged to a minimum or maximum value.

2. When

$$\frac{\sum_i \sum_j (V_{ij})^2}{N} > 0.95, \tag{5.31}$$

where N is the number of neurons with output values $\in$ [0.9,1.0].

5.8.3 Number of Iterations at each Temperature

At each temperature, the network is annealed until the following convergence criterion is met

$$\sum_i \sum_j \mid V_{ij}(t+1) - V_{ij}(t) \mid < 0.001 N_{on}, \tag{5.32}$$

where N_{on} is the number of neurons with non-zero output values.

5.8.4 Temperature Updating Rule

The updating rule is defined exactly as used in SA:

$$T = T_0(1 - (n/n_{max})), \tag{5.33}$$

where n is the number of iterations at which the temperature is being decreased and n_{max} is the maximum number of iterations after which the annealing is brought to an end. It should be noted that *linear* updating is used.

5.9 The Mean Field Annealing Algorithm

The task of annealing can be broken down into the following steps:

1. Initialize the neurons randomly

$$V_{ij} = rand[0, 1]n_{ij}. \tag{5.34}$$

2. Anneal the network until the saturation criterion, as defined above, is met.

3. At each temperature, iterate the MFT equations until convergence:

$$V_{ij} = \frac{n_{ij}}{2}[1 + \tanh \frac{h_{ij}^{MFT}}{2T} n_{ij}]. \tag{5.35}$$

5.10 Inherent Limitations in the MFA Algorithm

The above formulation of the MFA infrastructure *could not be applied* to the task of point pattern matching because of the following limitations inherent in the MFA algorithm:

1. Judging from step (1) in section (5.9) we see that each neuron will have an initial value $\in$ [0,1] and will violate the implicit requirement of the problem; namely, any valid assignment should form a one-to-one mapping from the observed to the model points.

2. During the course of the algorithm as well, the constraint of one-to-one mapping will never be met, as shown by the following illustrative output of the neurons during a sample run:

$$\begin{bmatrix}
0.51 & 0.00 & 1 & 0.28 & 0.39 & 0.79 & 1 & 0.31 & 0.38 & 0.94 & 0.52 & 0.45 \\
0.88 & 0.77 & 0.76 & 0.77 & 0.60 & 0.69 & 1 & 0.13 & 0.27 & 0.32 & 1 & 0.81 \\
0.43 & 0.73 & 0.62 & 0.78 & 1 & 0.75 & 0.58 & 0.52 & 1 & 0.46 & 0.01 & 0.02 \\
0.46 & 0.31 & 0.21 & 0.42 & 0.82 & 0.66 & 0.32 & 0.31 & 0.36 & 0.51 & 0.27 & 1 \\
0.80 & 1 & 0.21 & 0.28 & 0.15 & 0.63 & 0.72 & 1 & 0.25 & 0.66 & 0.64 & 0.70 \\
0.36 & 1 & 0.08 & 0.19 & 0.98 & 1 & 0.14 & 0.51 & 1 & 0.40 & 0.54 & 0.70 \\
0.21 & 0.68 & 0.38 & 0.01 & 0.25 & 0.59 & 0.16 & 0.43 & 1 & 0.60 & 0.91 & 0.43 \\
1 & 0.20 & 0.95 & 0.19 & 0.23 & 0.22 & 0.48 & 0.25 & 0.51 & 0.98 & 0.26 & 0.58 \\
0.15 & 0.83 & 0.94 & 0.98 & 0.10 & 0.31 & 0.86 & 0.37 & 0.72 & 1 & 1 & 0.75 \\
0.63 & 1 & 0.39 & 1 & 0.21 & 0.69 & 1 & 0.39 & 0.72 & 0.67 & 0.24 & 0.99 \\
0.61 & 0.82 & 1 & 0.81 & 0.63 & 0.11 & 0.55 & 0.44 & 0.94 & 0.34 & 0.84 & 0.69 \\
0.00 & 1 & 1 & 0.13 & 0.69 & 0.76 & 0.73 & 0.47 & 0.46 & 0.54 & 0.69 & 0.27
\end{bmatrix}$$

Here each row, i, corresponds to a model point and each column, j, to an observed point. It is clear that one-to-one mapping is *not* preserved as a model point is assigned to more than one observed point.

3. The absence of desired mapping makes it impossible to calculate the optimal parameter vectors and hence the cost C_{ij}, which forms the error matrix as defined in equation (5.18), cannot be computed. So the algorithm fails as soon as the constraints of the solution domain are not met because further annealing is impossible as the required parameters become unavailable.

4. There is no remedy to this problem in the algorithm, as constraints can never be matched throughout the algorithm; in fact, the necessity of the constraint as

an algorithm parameter would have been obviated had there been a technique to satisfy it with probability one.

5.11 Summary

From the discussions above it is clear that this technique cannot be applied to our task of point pattern matching. Had the cost function *not* been dependent on the constraint of one-to-one mapping the MFA technique could have proved a useful tool in that the convergence to the global optima is faster when the *mean value* is used by the relaxation parameter to search for the solution. Nevertheless, this chapter serves as a guide to how the problem may be mapped to a neural network and what constraints would come into play. Perhaps it may be possible to use some other algorithm based on the neural-network-mapped approach of the task of point pattern matching, in which the basic constraint used to calculate the error is not violated along the length of the algorithm.

CHAPTER 6

CONCLUSION

6.1 Overview

The techniques of SA and EP were successfully applied to the task of Point Pattern Matching. MFA could not be applied beacause of the constraint limitations inherent in the algorithm, as discussed. *Fast* convergence was obtained using the EP approach and it was noted that the implementation of environmental knowledge results in a robust algorithm, as it performs well even in the presence of high noise and in the absence of a complete data set.

New, knowledge-based variations of the *Mutation* and *Crossover* operator were defined in the EP algorithm, which contributed significantly to the convergence speed. It should be mentioned again that the importance of an operator lies in its definition and the way it is used. Hence, it would not be quite appropriate to classify operators as useful or useless in general, specifically in the case of EPs, where each operator can be custom engineered.

6.2 Extensions on this Work

In this work, only 2-D patterns and similarity transformations have been considered. It would be interesting to extend the algorithms presented herein, to handle n-dimension cases and other non-linear or affine transformations.

In the SA technique, instead of regular (*random*) perturbation, one could use knowledge-based perturbation. This, however, would need to be done carefully, as perturbation is the only mechanism of traversing the solution space in the algorithm and the search is done one node at a time. It is evident that during the use of knowledge-based perturbation, not all regions of the solution space stand an equal chance of being searched, and any over-zealous knowledge-based technique might actually slow down the convergence speed rather than improve on it.

Furthermore, based on heuristics, more specific versions of the mutation and crossover operators can be defined in the case of EP, so as to control the recombination of offspring at different ages of the evolving population more effectively and efficiently, leading to faster and more robust algorithms.

REFERENCES

1. F. Attneave, "Some Informational Aspects of Visual Perception," *Psychol. Rev.*, vol.61, no.3, pp.183–193, 1954.

2. N. Ansari and E. J. Delp, "Partial Shape Recognition: A Landmark-Based Approach," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.12, no.5, pp.470–483, May 1990.

3. N. Ansari and E. J. Delp, "Recognizing Planar Objects in 3-D Space," *Proc. SPIE Automated Inspection and High-Speed Vision Architectures III*, vol.1197, Nov.5–10, 1989, Philadelphia, PA, pp.127–138.

4. R. C. Bolles and R. A. Cain, "Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method," *Int. J. Robotics Res.*, vol.1, no.3, pp.57–82, Fall 1982.

5. D. P. Huttenlocher and S. Ullman, "Object Recognition Using Alignment," *Proc. IEEE 1st Int. Conf. Computer Vision*, London, pp.102–111, 1987.

6. M. C. Ibison and L. Zapalowski, "On The Use of Relaxation Labeling in the Correspondence Problem," *Pattern Recognition Letters*, pp.103–109, April 1986.

7. M. W. Koch and R. L. Kashyap, "Using Polygons to Recognize and Locate Partially Occluded Objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.PAMI-9, no.4, pp.483–494, July 1987.

8. S. Ranade and A. Rosenfeld, "Point Pattern Matching by Relaxation," *Pattern Recognition*, vol.12, pp.269–275, 1980.

9. N. Ayache and O. D. Faugeras, "HYPER: A New Approach for The Recognition and Positioning of Two-Dimensional Objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.PAMI-8, no.1, pp.44–54, Jan. 1986.

10. B. Bhanu and O. D. Faugeras, "Shape Matching of Two-Dimensional Objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.PAMI-6, no.2, pp.137–156, Mar. 1984.

11. B. Bhanu and J. C. Ming, "Recognition of Occluded Object: A Cluster-Structure Algorithm," *Pattern Recognition*, vol.20, no.2, pp.199–211, 1987.

12. L. S. Davis, "Shape Matching Using Relaxation Techniques," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.PAMI-1, no.1, pp.60–72, Jan. 1979.

REFERENCES
(Continued)

13. K. E. Price, "Matching Closed Contours," *Proc. Seventh Int. Conf. Pattern Recognition*, Montreal, Canada, July 30–Aug. 2, 1984, pp.990–992.

14. G. J. Ettinger, "Large Hierarchical Object Recognition Using Libraries of Parameterized Model Sub-Parts," *Proc. IEEE Comput. Soc. Conf. Computer Vision and Pattern Recognition*, Ann Arbor, MI, pp.32–41, June 5–9, 1988.

15. J. W. Gorman, O. R. Mitchell, and F. P. Kuhl, "Partial Shape Recognition Using Dynamic Programming," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.PAMI-10, no.2, pp.257–266, Mar. 1988.

16. W. E. L. Grimson, "On Recognition of Curved Objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.PAMI-11, no.6, pp.632–643, June 1989.

17. A. Kalvin, E. Schonberg, J. T. Schwartz, and M. Sharir, "Two-Dimensional Model-Based, Boundary Matching Using Footprints," *Int. J. Robotics Res.*, vol.5, no.4, pp.38–51, Winter 1986.

18. T. F. Knoll and R. C. Jain, "Recognizing Partially Visible Objects Using Feature Indexed Hypotheses," *IEEE Trans. Robotics and Automation*, vol.RA-2, no.1, pp.3–13, Mar. 1986.

19. J. L. Turney, T. N. Mudge, and R. A. Volz, "Recognizing Partially Occluded Parts," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.PAMI-7, no.4, pp.410–421, July 1985.

20. Y. Lamdan, J. T. Schwartz, and H. J. Wolfson, "Object Recognition by Affine Invariant Matching," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Ann Arbor, MI., June 5–9, 1988, pp.335–344.

21. Y. Lamdan, J. T. Schwartz, and H. J. Wolfson, "On Recognition of 3-D Objects From 2-D Images," *Proc. IEEE Int. Conf. Robotics and Automation*, Philadelphia, PA., Apr. 1988, pp.1407–1413.

22. W. K. Pratt, *Digital Image Processing, 2nd edition*, New York: Wiley, 1991.

23. S. Kirkpatric, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220 (1983): 671–680.

24. D. I. Ackley, G. E. Hinton, and T. J. Seijnowski, "A Learning Algorithm For Boltzman Machines," *Cognitive Science*, Vol. 9 (1983): 147–169.

REFERENCES
(Continued)

25. P. J. M. van Laarhoven, and E. H. L. Aarts, *Simulated Annealing: Theory And Applications*, Netherlands: Kluwer Academic Publishers, 1992.

26. D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Reading, MA: Addison Wesley, 1989.

27. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin Heidelberg, 1992.

28. J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, Vol. 52, No. 4, p.141–156, 1985.

29. C. Peterson and E. Hartman, "Explorations of The Mean Field Theory Learning Algorithm," *Neural Networks*, Vol. 2, p.475–494, 1989.

30. C. Peterson and B. Soderberg, "A New Method for Mapping Optimization Problems Onto Neural Network," *International Journal of Neural System*, Vol. 1, No. 1, p.3–22, May 1989.

31. J. A. Anderson and E. Rosenfeld, *Neurocomputing, Foundations of Research.* Cambridge, MA:MIT Press, 1988.

32. J. J. Hopfield, "Neural Networks and Physical Systems With Emergent Collective Computational Abilities," *Proceedings of The National Academy of Science*, Vol. 79, p.2541–2554, 1982.

33. J. J. Hopfield, "Neurons With Graded Response Have Collective Computational Properties Like Those of Two-State Neurons," *Proceedings of The National Academy of Science*, Vol. 81, p.3088–3092, May 1984.

34. R. Hecht-Nielson, *Neurocomputing*, Reading, MA: Addison Wesley, 1989.

35. M. Cohen and S. Grossberg, "Absolute Stability of Global Pattern Works," *IEEE Trans. System, Man, and Cybernetics*, SMC-13, p. 815–926, 1983.

36. C. Peterson and J. Anderson, "Neural Networks and NP-Complete Optimization Problems: A Performance Study on The Graph Bisection Problem," *Complex System*, Vol. 2, p. 59–71, 1988.

37. C. Peterson, "A Mean Field Theory Learning Algorithm For Neural Networks," *Complex System*, Vol. 1, p. 995–1019, 1987.

38. C. C. Galland and G. E. Hinton, "Experiments on Discovering High Order Features With Mean Field Modules," *Technical Report CS-115, Dept. of Computer Science*, U. of Toronto, May 1989.

REFERENCES
(Continued)

39. C. Peterson, "Applications of Mean Field Theory Neural Networks," *Dept. of Theoretical Physics, Technical Report CS-1153*, University of Lund, August, p.1–27, 1989.

40. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, "Equations of State Calculations by Fast Computing Machines," *J. of Chemical Physics*, no.21, pp.1087–1091, 1953.

41. S. Kirkpatrick, C. D. Gelatt Jr. and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol.220, pp.671–680, 1983.

42. S. Gemen and D. Gemen, "Stochastic Relaxation, Gibbs Distributions and Baysian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.6, pp.721–741, 1984.

43. B. Gidas, "Non-stationary Markov Chains and Convergence of The Annealing Algorithm," *J. of Statistical Physics*, no.39, pp.73–131, 1985.

44. S. B. Gelfand, "Analysis of Simulated Annealing Type Algorithms," *Ph.D. Thesis*, Massachusetts Institute of Technology, 1987.

45. P. J. M. Van Laarhoven and E. H. L. Arts, *Simulated Annealing: Theory and Applications*, Dordrecht, Holland: Reidel Publishing Company, 1987.

46. C. Peterson, "A Mean Field Theory Learning Algorithm for Neural Networks," *Complex System*, Vol. 1, pp.995–1019, 1987.

47. C. Peterson, "Neural Networks and NP-Complete Optimization Problem; a Performances Study on The Graph Bisection Problem," *Complex System*, Vol. 2, pp.59–89, 1988.

48. C. Peterson and E. Hartman, "Explorations of The Mean Field Theory Learning Algorithm," *Neural Network*, Vol. 2, pp.475–494, 1989.

49. G. Bilbro and R. Mann, et al, "Optimization by Mean Field Annealing," *Advances in Neural Information Processing System 1*, D. S. Touvetzky (ed.), pp.91–98, 1989.

50. J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, Vol. 52, pp.141–152, 1985.

51. K. A. De Jong, "Genetic Algorithms: A Ten Year Perspective," *Proceedings of The First International Conference on Genetic Algorithms*, pp.169–177, J. J. Grefenstette (ed.), Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.

REFERENCES
(Continued)

52. Z. Michalewicz, C. Janikow, and J. Krawczyk, "A Modified Genetic Algorithm for Optimal Control Problems," *Computers and Mathematics With Applications*, vol.23, no.12, pp.83–94, 1992.

53. C. Janikow, and Z. Michalewicz, "Specialized Genetic Algorithms for Numerical Optimization Problems," *Proceedings of The International Conference on Tools For AI*, pp.798–804, 1990.

54. Z. Michalewicz, G. A. Vignaux, and M. Hobbs, "A Non-Standard Genetic Algorithm for the Non-Linear Transportation Problem," *ORSA Journal on Computing*, vol.3, no.4, pp.307–316, 1991.

55. W. M. Spears and K. A. De Jong, "On The Virtues of Parameterized Uniform Crossover," *Proceedings of The Fourth International Conference on Genetic Algorithms*, pp.230–236, Morgan Kaufman Publishers, Los Altos, CA, 1991.

56. G. Syswerda, "Uniform Crossover in Genetic Algorithms," *Proceedings of The Third International Conference on Genetic Algorithms*, pp.2–9, Morgan Kaufman Publishers, Los Altos, CA, 1989.

57. J. Schaffer, R. Caruana, L. Eshelman, and R. Das, "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization," *Proceedings of The Third International Conference on Genetic Algorithms*, pp.51–60, Morgan Kaufman Publishers, Los Altos, CA, 1989.