1-31-1991

Efficient multiprocessor scheduling by mean field theory neural networks

Pei-Ken Yi
New Jersey Institute of Technology

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

ABSTRACT

Title of Thesis :

Efficient Multiprocessor Scheduling by Mean Field Theory Neural Networks

Name:

Pei-Ken Yi

Master of Science in Electrical Engineering

Thesis Directed by : Dr. Nirwan Ansari

In this thesis, we develop an optimization method based on Mean Field Theory (MFT) Neural Networks to solve the Task Scheduling problem. The MFT algorithm combines characteristics of the Simulated Annealing (SA) algorithm and the Hopfield neural network. MFT exhibits rapid convergence and at the same time it preserves the solution quality afforded by SA. Since MFT has been successfully used to solve the Traveling Salesman Problem (TSP), a new modification to MFT is also presented which supports Task Scheduling problem. The temperature behavior of MFT during Task Scheduling is approximately analyzed and shown to possess a critical temperature (T_c) at which most of the optimization occurs. This temperature is analogous to the gain of the neurons in a neural network and may be used to tune such networks for better performance.

Efficient Multiprocessor Scheduling by Mean Field Theory Neural Networks

by
Pei-Ken Yi

Thesis submitted to the Faculty of the Graduate School
of the New Jersey Institute of Technology in partial
fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering

1991

Approval Sheet

Title of Thesis: Efficient Multiprocessor Scheduling by Mean Field Theory Neural Networks

Name of Candidate: Pei-Ken Yi

Degree: Master of Science in Electrical Engineering

Thesis and Abstract Approved:

Dr. Nirwan Ansari — Date
Assistant Professor
Department of Electrical and Computer Engineering

Dr. Edwin S. H. Hou — Date
Assistant Professor
Department of Electrical and Computer Engineering

Dr. Yun-Qing Shi — Date
Assistant Professor
Department of Electrical and Computer Engineering

VITA

NAME : Pei-Ken Yi

Address :

Date of Birth :

Place of Birth :

Eductation :

1. New Jersey Institute of Technology
   Electrical Engineering Department, M.S.E.E.
   Spring,1989 - January,1991
2. National Taipei Institute of Tech.
   Electrical Engineering Department
   Fall,1981 - Fall,1984

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to Dr. Nirwan Ansari for his invaluable guidance and encouragement throughout the course of this thesis. I am specially indebted to him for his insightful and constructive criticisms at every stage of this thesis without which this thesis would never have been completed. I also express may thanks to Dr. Edwin S. H. Hou and Dr. Yun-Qing Shi who served on this thesis committee and made appropriate suggestions.

I also express my sincere thanks to Mr. Z. Zhang for providing all the information of Mean Field Theory, which was vital to finish the work. I am also thankful to Mr. J. Yang for supplying the software of MS word and windows used in the thesis.

Last but not least I would like to thank all those who have helped me in one way or the other during the entire course of this thesis.

To My Mom and Wife.....

Contents

List of Figures

Chapter 1

Introduction

What is a Neural Network? Finding comprehensible answers has been difficult. Often expressed in technical jargon, many of the treatments assume a facility with branches of advanced mathematics that are seldom used in other specialties [10]. We can define it in this way, a neural network is a system composed of many simple processing elements, called neurons, operating in parallel[16][19]. The function of a neural network is determined by the network structure, interconnection strengths and the processing performed by the computing elements or nodes.

Artificial neural networks are biologically inspired. The network configurations and algorithms are usually derived from the organization of the brain [8]. The most important thing is that it can also be utilized to solve nonlinear problems. An artificial neural network is a computational structure modeled after the biological neural network. For example, they learn from experience, from previous examples to new ones, and abstract essential characteristics from inputs containing irrelevant data [9].

Despite these functional similarities between the artificial and biological, artificial neural networks will not duplicate the function like the biological's [20]. It is incorrect to ignore the surprisingly brainlike performance of certain

artificial neural networks. These abilities, however limited, hint that a deep understanding of human intelligence may lie close at hand, and along with it a host of revolutionary applications [17].

Artificial neural networks, as mentioned above, can adopt not only linear but also nonlinear systems. It is this factor, more than any other, that make many researchers interested in neural networks. Given a set of inputs, the neural network is initially trained to produce the predefined outputs. A wide variety of training algorithms have been developed, each with its own advantages and disadvantages. There are several existing models of neural networks such as Hopfield Nets (HN), Backpropagation (BP), Mean Field Theory (MFT), etc. They promise human-made devices that perform functions reserved for human beings. Dull, repetitive, or dangerous tasks can be performed by these machines and entirely new applications will arise as the technology matures. It is thus reasonable to expect that a rapid increase in our understanding of artificial neural networks lead to improved network paradigms and a host of application opportunities. We can also prove from the training that more neural networks are said to improve with experience. The more data they are fed, the more accurate or complete they will get [19].

In this thesis, we are going to focus on MFT. It has been shown that MFT exhibits better performance in pattern-recognition tasks than BP does [4]. The MFT algorithm combines characteristics of the simulated annealing [18] algorithm and the Hopfield neural network. Since MFT model is bidirectional, rather than feed-forward as BP, it can represent a major extension of computation.

MFT is an efficient algorithm, and it can be used to solve scheduling problems within the neural network paradigm. It was recently successfully applied to

solving the Traveling Salesman Problem (TSP) [12]. In this thesis, we consider optimizing the Task Scheduling problem by using MFT. MFT is an optimization technique that can be used to solve other combinatorial optimization problems. The detail of approximation and learning of MFT will be discussed later.

We shall next proceed to Chapter 2 with the discussion on MFT. The multiplexing encoding will be reviewed in Chapter 3. The mapping of the task scheduling problem into a MFT framework will be discussed in Chapter 4. Simulation results and conclusions will be presented in Chapter 5.

Chapter 2

Mean Field Theory

2.1 Introduction To Mean Field Theory

Since single layer networks proved severely limited in what they could represent and in what they could learn, the entire field went into virtual eclipse in the 60's and 70's. In contrast to single layer networks, multilayer networks have greater representational power. Present research efforts are geared towards multilayer networks [17][19]. MFT is a learning algorithm which can be incorporated in multilayer networks.

The feedback of MFT networks have recurrent paths from their outputs back to their inputs. The response of such networks is dynamic; after applying a new input, the output is then recalculated, and the process is repeated again and again. For stable networks, successive iterations produce smaller and smaller output changes until eventually the outputs become constant.

MFT was developed by "marrying" Hopfield Model (HM) and Boltzmann Machine (BM). In this thesis, we want to use the feature of MFT to optimize the Task Scheduling Problem. To solve the Task Scheduling problem by MFT, we need to map the problem into a neural network framework by encoding the tasks as neurons and modifying the Hopfield Model and the Boltzmann Machine.

In BM, learning takes place via two phases. First, the network is run with both the input and the output units clamped. Co-occurrence probabilities are measured after the system has reached a global energy minimum. This procedure is then repeated with only the input units clamped. The weights are then adjusted according to gradient-descent in an information-theoretical measure. In order to reach a global energy minimum in each phase of this process, it is necessary to use the very time-consuming simulated annealing method. This, combined with measuring the co-occurrence statistics, makes the Boltzmann machine one or two orders of magnitude slower than MFT [4]. Consequently, there has been relatively little experimentation with the Boltzmann machine.

In Hopfield model, all nodes in a network are fully connected. Consequently, it is worthwhile looking at the effects of limited connectivity. In addition, Hopfield leads to symmetric connection. Each connective neurons will influence each other. In contrast, the connections between biological neurons are asymmetric. Finally, learning rules are inadequate models for synaptic processes. It is known that only a restricted number of vesicles of neurotransmitter molecules are discharged at synaptic junctions, indicating that the synaptic efficacy cannot have a very broad spectrum of values. We will now consider extensions of the Hopfield model incorporating some of these features.

2.2 The Mean Field Theory Approximation

A brief description of the mean field theory approximation is given here. For more detailed treatment, we refer the readers to [4] [12] [13]. Consider the Hopfield energy function($E(S)$) [7] [14],

$$E(S) = -\frac{1}{2}\sum_i \sum_j T_{ij} S_i S_j \tag{2.1}$$

where each neuron is denoted by S_i, and $S_i = -1$ or 1. $T_{ij} = 1$ or 0, depending on whether neuron i and j are connected or not.

The stability of a network may be proven through an elegant mathematical technique. Suppose a function can be found that it always decreases each time the network changes rate. Eventually this function must reach a minimum and stop, thereby ensuring that the network is stable. The change rate in energy $E(S)$(Eq. 2.1), called the "local field" h_i [19], is defined by

$$h_i = -\frac{\partial E}{\partial S_i} = \sum_i T_{ij} S_j \tag{2.2}$$

where ∂S_i is the change in neuron i.

It can be shown [6] [7] that when the following updating rule,

$$S_i = sign(h_i) \tag{2.3}$$

is used, the energy described by Eq. 2.1 will reach the closest local minimum.

Other neural network applications, optimization problems and recognition tasks require that the global minimum of Eq. 2.1 is reached. There exist various stochastic procedures for performing the hill-climbing necessary to avoid getting stuck in local minima. A frequently used scheme is Simulated Annealing (SA) [18], where fluctuations of energy in Eq. 2.1 are allowed according to the Boltzmann distribution [13]

$$P(S) = \frac{1}{Z} e^{E(S)/T} \tag{2.4}$$

$$Z = \sum_s e^{E(S)/T} \tag{2.5}$$

where the "temperature" T has been introduced to set the magnitude of the fluctuations.

2.3 Mean Field Theory Learning

2.3.1 Simulated Annealing

Since SA is a probabilistic hill-climbing algorithm which finds a global minimum of $E(S)$ by combining gradient descent with a random process. Unlike traditional gradient descent method, SA allows, under certain conditions, move which causes an increase in $E(S)$, thus avoiding getting stuck at a local minima. The probability of allowing such a move which causes an increase of ΔE follows a Boltzmann distribution:

$$P_r\{uphill\ move = \Delta E\} = exp(\frac{-\Delta E}{T}) \tag{2.6}$$

that depends on a parameter, T, the temperature. The temperature is lowered as the SA algorithm proceeds, thus lowering the probability of accepting uphill moves and attempting to force the system into a global optimum.

Two conceptual operations are involved in SA. One is thermostatic operation which schedules a physical system decreasing in temperature. It takes a finite amount of time before thermal equilibrium is established. The other is relaxation operation which iteratively find the new equilibrium point at new temperature using the final state of the system at the previous temperature as a starting point.

The above stochastic mechanism which has been applied in the Boltzman

Machine [16] is also used in MFT. We have analytically and experimentally investigated the effect of temperature on the behavior of MFT on the proposed task scheduling problem that will be discussed in Chapter 4.

2.3.2 The Boltzmann Machine

The Boltzmann Machine (BM) [22] is a learning algorithm for neural networks with or without hidden units. The dynamics are based on the Hopfield energy function. The model learns by making an internal representation of its environment. The learning procedure changes weights so as to minimize the distance between output and desired probability distributions, as measured by the so-called G-function [13], given by

$$G = \sum_{\alpha} P_{\alpha} \log \frac{P_{\alpha}}{P\prime_{\alpha}} \tag{2.7}$$

where P_{α} is the probability that the visible units are collectively in state α when their states are determined by the environment. P_{α} represents the desired probabilities for these states. The corresponding probabilities when the network runs freely are denoted $P\prime_{\alpha}$. G is zero if and only if the distributions are identical; otherwise it is positive.

The Boltzmann Machine recipe for changing T_{ij} such that G is minimized is as follows:

1. Clamping Phase. The values of the input and output units of the network are clamped to a training pattern, and for a sequence of decreasing temperatures $T_n, T_{n-1}, ..., T_0$, the energy of the network shown in of Eq. 2.1 is allowed to relax according to the Boltzmann distribution Eq. 2.4. At $T = T_0$, statistics are collected for the correlations [13][18]

$$P_{ij} =< S_i S_j > \tag{2.8}$$

Relaxation at each temperature is performed by updating unclamped units according to the algorithm

$$P(Si \rightarrow 1) = [1 + e^{(\sum_j T_{ij} S_i / T)}]^{-1} \tag{2.9}$$

2. Free Running Phase. The same procedure as in Step 1, but this time the network runs freely or with only the input units clamped. Correlations

$$P\prime_{ij} =< S_i S_j > \tag{2.10}$$

are again measured at $T = T_0$

3. Updating. After each pattern has been processed through Steps 1 and 2, the weights are updated according to

$$\Delta T_{ij} = \eta(P_{ij} - P\prime_{ij}) \tag{2.11}$$

which is the learning rate parameter. Eq. 2.11 corresponds to the gradient descent of G. Steps 1,2 and 3 are repeated until no more changes in T_{ij} take place.

2.4 Mathematical Model of MFT

The general mathematical equations of MFT, modified from Hopfield model, will replace the h_i by U_i and S_i by V_i. In MFT, we use continuous variables rather than discrete variables [4]. In addition, we shall also develop that under every temperature(T), S_i is replaced by V_i as

$$V_i =< S_i >_T \tag{2.12}$$

where $< S_i >$ stands for average of the S_i values at temperature T. The state, V_i, and the local field of each neuron is related by a hyperbolic tangent function as follows:

$$V_i = tanh(U_i) \tag{2.13}$$

The change rate (local field) of total energy ($E(S)$) in MFT networks become

$$U_i = -\frac{1}{T}\frac{\partial E(S)}{\partial S_i} \tag{2.14}$$

The difference between Eq. 2.2 and 2.14 is the annealing temperature T.

These MFT mathematical model and parameters are modified from Hopfield model. After that, we anneal the system by lowering the temperature gradually, with the expectation that undesired local minima can be avoided through the stochastic mechanism discussed earlier, as in BM.

From Hopfield model, we can represent the relationship between neurons. The energy function($E(S)$) can also be set up. In BM model, the annealing algorithm can be used to minimize and clamp the energy function. We use the MFT that takes the advantages of these two models to solve the Task Scheduling problem. The MFT algorithm along with simulation results on the task scheduling problem will be discussed and presented in more details in Chapter 4 and 5.

Chapter 3

Neural Network Multiplexing Encoding

3.1 Graph Bisection

Neural networks have shown great promise as heuristic for solving difficult optimization problems [1]. In their pioneering work, Hopfield and Tank [6] formulated the Travelling Salesman Problem (TSP) on a highly interconnected neural network and made exploratory numerical studies on modest-sized samples. Here, we are going to review how to apply Hopfield model to solve the Graph Bisection (GB) problem [12] [18] [15].

The GB problem was used as a test for extensive numerical explorations, using a neural network mean field theory method [12]. The problem is defined as follows: Given a set of N nodes with a given connectivity, partition them into two halves such that the net connectivity (cutsize) is minimal between the two halves (Fig. 3.1). This problem is mapped onto a neural network by denoting each node as a neuron. Each neuron, denoted by S_i, is turned "on" or "off" respectively, depending on which of the two halves node i belongs to.

Thus, S_i is mathematically defined by:

Figure 3.1: A Graph Bisection Problem

$$S_i = \begin{cases} 1 & \text{node i in the right side set,} \\ -1 & \text{otherwise.} \end{cases} \tag{3.1}$$

The dynamics is governed by an appropriate choice of energy function together with the corresponding mean field theory equations.

The GB problem is mapped onto a Hopfield energy function by the following:

$$E(S) = -\frac{1}{2}\sum_i \sum_j T_{ij} S_i S_j. \tag{3.2}$$

For each node, assign a neuron S_i, and for each pair of neurons, S_i, S_j, $i \neq j$, we also assign a value, T_{ij}, which is defined below.

$$T_{ij} = \begin{cases} 1 & \text{if node i and j are connected,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

The product $T_{ij}S_iS_j$ is zero whenever nodes i and j are not connected. The product is positive when nodes i and j are connected, and are in the same partition, and is negative when they are in separate partitions. With this representation, the minimization of Eq. 3.2, the energy function, will maximize the connections within a partition while minimizing the connections between partitions.

However, such minimization will force all nodes into one partition. Hence, an addition of a "constraint term" to Eq. 3.2 must be introduced to penalize situations where the nodes are not equally partitioned. Note that $\Sigma S_i = 0$ when the partitions are balanced. Hence, a term proportional to $(\Sigma S_i)^2$ will increase the energy whenever the partition is unbalanced. Our neural network energy function for graph bisection then takes the form [12]:

$$E = -\frac{1}{2}\sum_{ij} T_{ij} S_i S_j + \frac{\alpha}{2}(\sum_i S_i)^2 \tag{3.4}$$

where the imbalance parameter α sets the relative strength between the cutsize and the balancing term. This balancing term represents a global constraint. The structure of Eq. 3.2 differs from that of Eq. 3.4 by the presence of the second term describing the balancing term. The generic form of Eq. 3.4 is

E = "cost" + "global constraint."

3.2 1-of-K Encoding; Neuron Multiplexing

The 1-of-K encoding problem is defined as follows. Given a set of N nodes with a given connectivity, partition them into K sets such that the net connectivity (cut size) is minimized among the K sets. To extend the neural network model used for the graph bisection poblem to that for the 1-of-K encoding problem, we first introduce a second index for the neurons

$$S_{ia} = 0, 1 \tag{3.5}$$

where index i denotes the node (i = 1,...., N), and a the set (a = 1,...,K). S_{ia} takes on 1 or 0 depending on whether node i belongs to set a or not. We use 0, 1 notation in order to get a more convenient form of the energy function which

is analogous to Eq. 3.2. The resulting energy function is thus written as [12],

$$E = \frac{1}{2}\sum_{ij}\sum_{ab} T_{iajb} S_{ia} S_{jb} + \frac{\alpha}{2}\sum_{a}(\sum_{i} S_{ia} - \frac{N}{K})^2 \tag{3.6}$$

As in the bisection case, the second term in Eq. 3.6 represents the global constraint of equipartition; it is zero only if each of the K sets contains N/K nodes. There is an additional syntax constraint term built into T_{iajb} ensuring that the 1-of-K encoding is satisfied.

$$T_{iajb} = T^{(1)} + T^{(2)} \tag{3.7}$$

The cutsize and syntax constraint terms, $T^{(1)}$ and $T^{(2)}$, are defined [12] as

$$T^{(1)} = T_{ij}(1 - \delta_{ab}) \tag{3.8}$$

and

$$T^{(2)} = \beta\delta_{ij}(1 - \delta_{ab}) \tag{3.9}$$

respectively, where $T_{ij} = 1$ or 0 depending on whether the ith and jth node are connected or not. The cutsize constraint (Eq. 3.8) is introduced to encourage connected neurons into one same set, thus minimizing the connectivity among the K sets. This constraint will, however, force all neurons into one set. Hence, the second term in Eq. 5.6 is introduced to enforce equipartition. The syntax term (Eq. 3.9) is zero if each neuron is assigned to no more than one set. The energy function shown in Eq. 3.6 now takes the form

$$E = \frac{1}{2}\sum_{ij}\sum_{a\neq b} T_{ij} S_{ia} S_{jb} + \frac{\beta}{2}\sum_{i}\sum_{a\neq b} S_{ia} S_{ib} + \frac{\alpha}{2}\sum_{a}(\sum_{i} S_{ia} - \frac{N}{K})^2 \tag{3.10}$$

In the 1-of-K encoding problem, the neurons take on 0 and 1 rather than -1 and 1 in the graph bisection problem. To solve this problem with in the framework of MFT, we define the following mean field variables which are analogous to those defined by Eqs. 2.12, 2.13, and 2.14:

$$V_{ia} =< S_{ia} >_T \tag{3.11}$$

$$V_{ia} = \frac{1}{2}[1 + \tanh(U_{ia})] \tag{3.12}$$

where $U_{ia} = -\frac{1}{T}\frac{\partial E(S)}{\partial S_{ia}}$. Thus,

$$V_{ia} = \frac{1}{2}[1 + \tanh[(-\sum_j \sum_{b \neq a} T_{ij} V_{jb} - \beta \sum_{a \neq b} V_{ib} - \alpha(\sum_j V_{ja} - \frac{N}{K}))/T]] \tag{3.13}$$

In constrast to Eqs. 2.12-2.14, the variables are indexed by double subscripts, and take on [0,1] rather than [-1,1]. Since the variables take on [0,1], the state and the local field of each neuron are related by Eq. 3.12.

3.3 Reduced 1-of-K Encoding; Graded Neurons

In this section, we restrict the allowed states for the neurons such that exactly one neuron at each set is on, and derive the corresponding K-state Potts glass mean field theory equations [21].

3.3.1 The Potts Glass

The above restriction on the neurons can be compactly written as

$$\sum_a S_{ia} = 1 \tag{3.14}$$

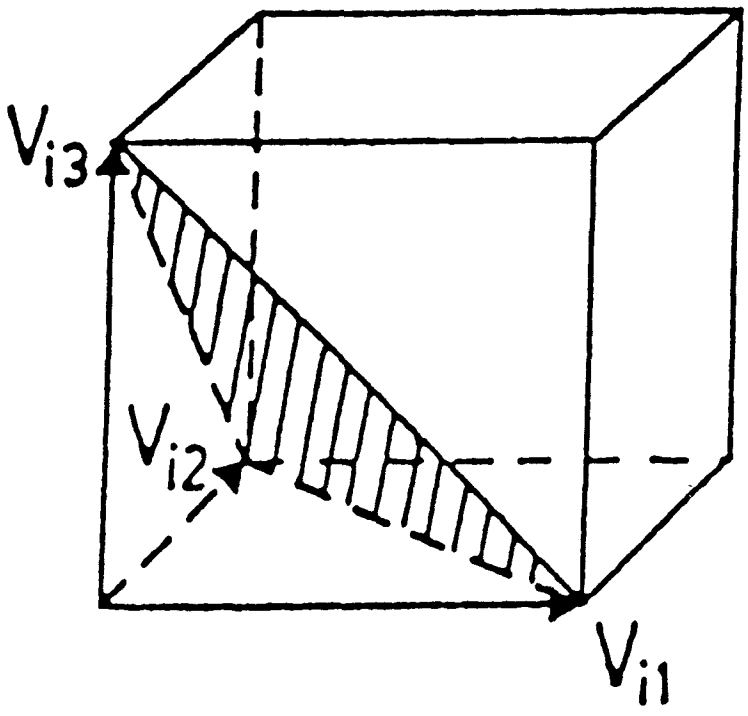

Figure 3.2: The volume of solutions corresponding to the neuron multiplexing for K = 3. The shaded plane corresponds to the solution space of the reduced 1-of-K encoding

For every i, S_{ia} is one for only one value of a, and zero for the remaining values of a. Therefore, the allowed values of the vector $\mathbf{S_i} = (S_{i1}, S_{i2}, ..., S_{iK})$ are the principal unit vectors $\mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_K}$ in an obvious vector notation.

The number of states available at every node is thereby reduced from 2^K to K, and technically we have a K-state Potts model at our hands. In Fig. 3.2 [12] we show the space of states at one node for the case K = 3.

The energy function of Eq. 3.10 can now be rewritten, using the constraint of Eq. 3.13, as

$$E = -\frac{1}{2}\sum_{ij}\sum_{a} T_{ij} S_{ia} S_{ja} - \frac{\beta}{2}\sum_{i}\sum_{a} S_{ia}^2 + \frac{\beta}{2}\sum_{ij}\sum_{a} S_{ia} S_{ja} \qquad (3.15)$$

or, in vector notation,

$$E = -\frac{1}{2}\sum_{ij} T_{ij}\mathbf{S_i}\mathbf{S_j} - \frac{\beta}{2}\sum_{i}\mathbf{S_i^2} + \frac{\alpha}{2}(\sum_{i}\mathbf{S_i})^2 \tag{3.16}$$

disregarding an unimportant constant term. Also note that the second term can now be dropped since it is a constant. We will however keep it since it will for some applications improve the solution quality. Also it turnouts to be a convenient regulator for avoiding chaotic behavior in synchronous updating. With $\beta = 0$, this expression has exactly the same structure as the energy (Eq. 3.4) for the graph bisection problem. Indeed, for K = 2, they are identical.

3.3.2 Mean Field Theory

As in the bisection case, we want to avoid getting stuck in local minima, by applying the mean field technique. For this reason we now derive the MFT equations corresponding to Eq. 2.12 for

$$V_i =< S_i >_T \tag{3.17}$$

Consider the Potts model partition function [11]

$$Z = \sum_{S} e^{-E(S_i)/T} \tag{3.18}$$

where the sum runs over all possible configurations satisfying the constraint of Eq. 3.13, i.e., $\mathbf{S_i}$ = (1,0,0,...),(0,1,0,...),(0,0,1,...), etc. The mean field theory trick is to rewrite this sum as an integral and to valuate its integrand at the saddlepoint. For simplicity we here initially limit the discussion to one spin S_i = S. A sum over $\mathbf{S} = e_1, e_2, ..., e_K$ can be rewritten in the following way:

$$\sum_{S} f(\mathbf{S}) = \sum_{S}\int_R d\mathbf{V}\delta(\mathbf{S}-\mathbf{V})f(\mathbf{V}) = C\sum_{S}\int_R d\mathbf{v}f(\mathbf{V})\int_I d\mathbf{U}e^{\mathbf{U(S-V)}} \tag{3.19}$$

Performing the sum, one obtains, for $\mathrm{f}(\mathbf{S}) = e^{-E(\mathbf{S})/T}$,

$$\sum_S e^{-E(\mathbf{S})/T} = \int_R d\mathbf{V} \int_I d\mathbf{U} e^{-E(\mathbf{V})/T - \mathbf{U}\cdot\mathbf{V} + \log Z_K(\mathbf{U})} \tag{3.20}$$

Where Z_k is the "local" partition function given by

$$Z_K(\mathbf{U}) = \sum_S e^{\mathbf{S}\cdot\mathbf{U}} = \sum_a e^{U_a} \tag{3.21}$$

For the partition function of Eq. 3.18 one then gets

$$Z = \sum_{\mathbf{S_i}} e^{E(S_i)/T} = C \int_R d\mathbf{V_i} \int_i d\mathbf{U_i} e^{(-E(\mathbf{V_i})/T + \sum_i (\log Z_K(\mathbf{V_i}) - \mathbf{U_i}\cdot\mathbf{V_i}))} \tag{3.22}$$

The saddle points of Eq. 3.22 are given by $\partial\mathbf{V_i} = 0$ and $\partial\mathbf{U_i} = 0$, yielding

$$\mathbf{V_i} = \mathbf{F}_K(-\frac{\partial \mathbf{E}}{\partial \mathbf{V_i}} \frac{1}{T}) \tag{3.23}$$

where $\mathbf{F_K}$ is a vector generalization of sigmoid function, defined as the average of $\mathbf{S}$ in the local partition function (Eq. 3.21):

$$\mathbf{F}_K(\mathbf{U}) = \frac{\sum \mathbf{S} e^{\mathbf{U}\cdot\mathbf{S}}}{\sum e^{\mathbf{U}\cdot\mathbf{S}}} \tag{3.24}$$

Writing this out in components,

$$\mathbf{F}_K^a(\mathbf{U}) = \frac{e^{U_a}}{\sum_b e^{U_b}} \tag{3.25}$$

which for K = 2 gives rise to a tanh function. Note that this expression automatically satisfies the constraint

$$\sum_a F_K^a(\mathbf{U}) = 1 \tag{3.26}$$

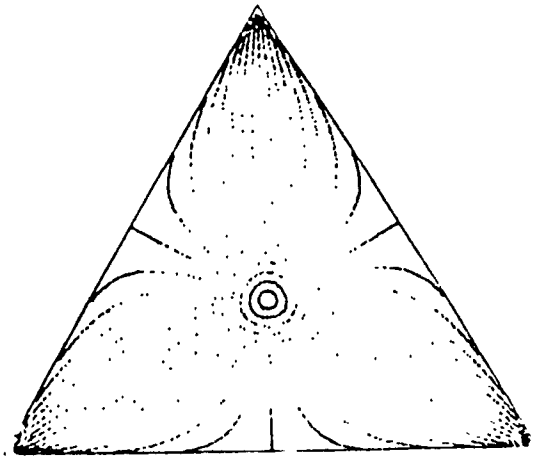

Figure 3.3: Contour map for the generalized function F_3 of Eq. 3.23

Thus, when integrating Eq. 3.23, the mean field variables, $\mathbf{V_i}$ will be forced to live in this (K - 1) - dimensional subspace of the original K - dimensional unit hypercube, as shown in Fig. 3.3 for the case K = 3.

The interpretation of V_{ia} as probabilities is obvious. In Fig. 3.3 we have plotted a contour map of $\mathbf{F}_K$ for K = 3. For the graph partitioning problem we now have the MFT equations:

$$\mathbf{V}_i = \mathbf{F}_K(\mathbf{U}_i) \tag{3.27}$$

$$\mathbf{U}_i = -\frac{\partial E}{\partial \mathbf{V}_i}\frac{1}{T} = [\sum_j (T_{ij} - \alpha)V_j + \beta \mathbf{V}_i]\frac{1}{T} \tag{3.28}$$

We note the following properties of the saddle point equations (Eqs. 3.27, 28):

* $\mathbf{V}_i$ automatically lies in the subspace $\Sigma_a V_{ia} = 1$.

* When interating Eqs. 3.27, 28 the component of $\mathbf{V}_i$ orthogonal to this subspace will be completely redundant.

Chapter 4

Multiprocessor Scheduling Problem

4.1 Scheduling Problem

Multiprocessor scheduling [3] [5] has been a source of challenging problems for researchers in the area of computer engineering. The general problem of multiprocessor scheduling can be stated as scheduling a set of partially ordered computation tasks onto a multiprocessor system so that an objective function will be optimized.

A task system can be represented by a directed acyclic task graph, TG, consisting of a finite nonempty set of vertices, V. The collection of vertices V= $T_1, T_2, ..., T_m$, and each connected pair, called T_{ij}. If T_{ij} exists and $i < j$, then task T_i must be completed before T_j can be initiated. A simple task graph with 8 tasks is illustrated in Fig. 4.1.

The problem of optimal scheduling a task graph and a multiprocessor system with P processors is to assign the computation tasks to the processors in such a way that the precedence relations are maintained and that all the tasks are completed in the shortest possible time. That is we want the last task completed, finish time, as short as possible. In this thesis, we are going to use the MFT

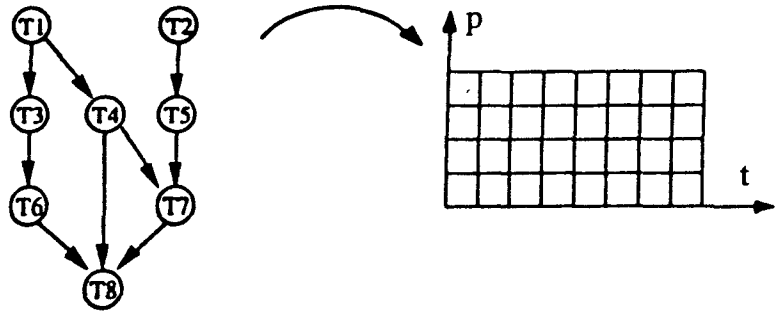

Figure 4.1: Mapping of Tasks T_i onto processor-time slot (p,t)

algorithm to solve the TG problem.

The MFT equations are governed by the annealing temperature T, and coefficients associated with the energy function which is problem dependent, such as α and β in the TSP problem. It is desirable to be able to estimate the values for these parameter so that a "trial-and-error" process can be avoided when applying the algorithm to different problems. Let us consider how to map task scheduling problem onto the processor-time-slot space, as shown in Fig. 4.1.

4.2 The Procedure For Scheduling Tasks

The spinsystems onto which we have mapped the optimization problems typically have two phases; at large enough temperatures the system relaxes into the trivial fixed point initial value of $V_{ia}(0)$ which is a completely symmetrical state, where all V_{ia} are equal [14]

$$V_{ia}(0) = \frac{1}{K} \tag{4.1}$$

As the temperature is lowered, a phase transition is passed at $T = T_c$ and

as $T \rightarrow 0$ fixed points $V_{ia}^{(*)}$ emerge representing specific decision made as to the solution to the optimization problems in question. These fixed points are characterized by

$$\prod \equiv \frac{1}{N} \sum_{ia} (V_{ia}^{(*)})^2 = 1 \tag{4.2}$$

where we have introduced the saturation $\prod$ [14] [18]. Let us now turn to the parameters α and β. The latter is redundant from the encoding point of view, as is obvious from Eq. 3.13. However, it turns out that for TSP the presence of the β-term has a constructive balancing effect in solving the MFT equations. In general the quality of the solutions are far less sensitive to the choice of α and β once T has been reliably estimated.

The parameter α governs the relative balance between "cost" and "rule" - term. The choice of balance is of course up to the "programmer." What are the guidelines? For graph bisection $\alpha = 1$ means that both "cost" and "rule" term are equally important, since the cost associated with an additional cut is $\frac{1}{2}\Sigma T_{ij} S_i S_j = 2$ and the corresponding cost for an imbalance is $\frac{\alpha}{2}(\Sigma S_i)^2 = 2\alpha$. In [14], $\alpha = 1$ was chosen for graph partition.

For TSP the situation is a little different since the "cost" is an analog number representing city distances and the absolute magnitude has to be chosen accordingly. For TSP testbeds consisting of cities randomly chosen in a unit square $\prod = 1$ again turned out to be a good choice.

For our simulations at least two alternatives are at our proposal. One is to use a fixed temperature approach as in Chapter 3 with T chosen below T_c. The other option is annealing where one chooses a temperature slightly above T_c and then anneals down to some value for the saturation $\prod$.

We have chosen this alternative with $\prod_{final} = 0.9$. The reason for not choosing $\prod = 1$ is that the performance of the final greedy heuristic improve with a not too "cold" solution.

As far as the number of sweeps per temperature goes, we have taken a dynamical approach. At each temperature T_n we interate the MFT equations until the convergence criteria ($N_{sweep}(T)$)

$$\frac{1}{NK} \sum \mid V_{ia}(t+1) - V_{ia}(t) \mid < \frac{0.004}{K} \tag{4.3}$$

is fulfilled. In other words N_{sweep} depends on T. It turns out the $N_{sweep}(T)$ stays constant at a very small number before and after phase transition, whereas it blows up around T_c. In summary , algorithm is implemented in the following manner in the case of serial updating [2][14][22]:

* For a given, determine T_{ij}.

* Set the coefficients of the respective energy function which will be discusses later.

* Determine T_c from linear expansion around trivial fixed point.

* Initialize with $V_{ia} = \frac{1}{K}$ + 0.001 * rand [-1,1].

* Anneal with T_n perform $N_{sweeps}(T)$ according to Eq. 4.3.

* After $\prod = 0.9$ is reached, perform the greedy heuristic.

By following the Mean Field theory and the learning from the Boltzmann Machine and approximation from the Hopfield Model, we know how to build the energy function under the task scheduling environment and constraint. We first consider the case in which task has the same execution time, and we shall then consider cases where different tasks have different execution times.

Case (1): Each Task Has The Same Execution Time.

Constraints:

1. Each task occupies one slot only.

2. There are P processors, and thus at most P tasks can be accomplished in each unit-time.

3. Task i and task j cannot be connected within one unit-time (set). If task i and task j are connected in the task graph, there is a precedence relationship between task i and task j. That is, task i and task j cannot be executed simultaneously. Thus, tasks carried out within the same unit time cannot be connected.

4. Event (i, a) must be connected to at least one of the $(j, a-1), (j, a-2), ..., (j, 1)$ events, where $a > 1$. That is, a task executed at the current unit time must be preceded by at least a task in one of preceding time slots.

Denote $S_{ia} = 0$ and 1 if task i is not in set and in set a, respectively. Also, $\Sigma_a S_{ia} = 1$ which means that each task is only allowed to occupy one time-slot. The value of δ_{ab} equal to 1 only when $a = b$. We can map the above constraints into the following respective energy functions:

$$1. E_1 = \frac{1}{2} \sum_{ij} \sum_{ab} \delta_{ij}(1 - \delta_{ab}) S_{ia} S_{jb} = \sum_{i} \sum_{a} \sum_{b \neq a} S_{ia} S_{ib} \tag{4.4}$$

$$2. E_2 = \sum_{a} [\sum_{i} S_{ia} - P]^2 \tag{4.5}$$

$$3. E_3 = \frac{1}{2} \sum_{ij} \sum_{a} T_{ij} S_{ia} S_{ja} \tag{4.6}$$

$$4. E_4 = \frac{1}{2} \sum_{i} \sum_{j=1}^{i-1} \sum_{a} \sum_{b=1}^{a-1} (1 - T_{ij}) S_{ia} S_{jb} \tag{4.7}$$

The total energy of the Task Graph E_T is the sum of the E_1 to E_4.

$$E_T = \alpha E_1 + \beta E_2 + \gamma E_3 + \xi E_4 \tag{4.8}$$

where α, β, γ, and ξ are constants.

When we finally apply the mean field, we obtain the following MFT reduced 1-of-K encoding (Eqs. 3.23, 25) equations for the mean field variable V_{ia} :

$$\begin{cases} V_{ia} = F_N(U_{ia}) \\ U_{ia} = -\dfrac{\partial E}{\partial V_{ia}} \dfrac{1}{T} \end{cases} \tag{4.9}$$

where the function of F_N is defined in Eq. 3.22.

Case (2): Tasks Having Different Execution Time.

Constraints:

1. If the execution time(t_i) for task i is larger than one unit time, this task will occupy the necessary number of time slots.

2. There are P processors, and thus at most P tasks can be accomplished in each unit-time.

3. Task i and task j cannot be connected within one unit-time (set). If task i and task j are connected in the task graph, there is a precedence relationship between task i and task j. That is, task i and task j cannot be executed simultaneously. Thus, tasks carried out within the same unit time cannot be connected.

4. Event (i, a) must connected to at least one of the $(j, a-1), (j, a-2), ..., (j, 1)$ events, where $a > 1$. That is a task executed at the current unit time must be preceded by at least a task in one of the preceding time slots. The energy functions corresponding to constraints (2) to (4) are the same as those in case (1). Constraint (1) which is different from case (1) thus has the following energy function:

$$E_1 = \sum_{ij} \sum_{ab} \delta_{ij}(t_i - \sum_{k=0}^{t_i-1} \delta_{a(b-k)}) S_{ia} S_{jb} \tag{4.10}$$

The total energy of the Task Graph E_T equals to the sum of E_1, E_2, E_3 and E_4:

$$E_T = \alpha E_1 + \beta E_2 + \gamma E_3 + \xi E_4 \tag{4.11}$$

where α, β, γ, and ξ are constants.

Thus, we can get E_T from the constraint condition. For the result V_{ia}, we shall follow the steps in Chapter 3. The whole procedure is illustrated by a flow chart as shown in Fig. 4.2.

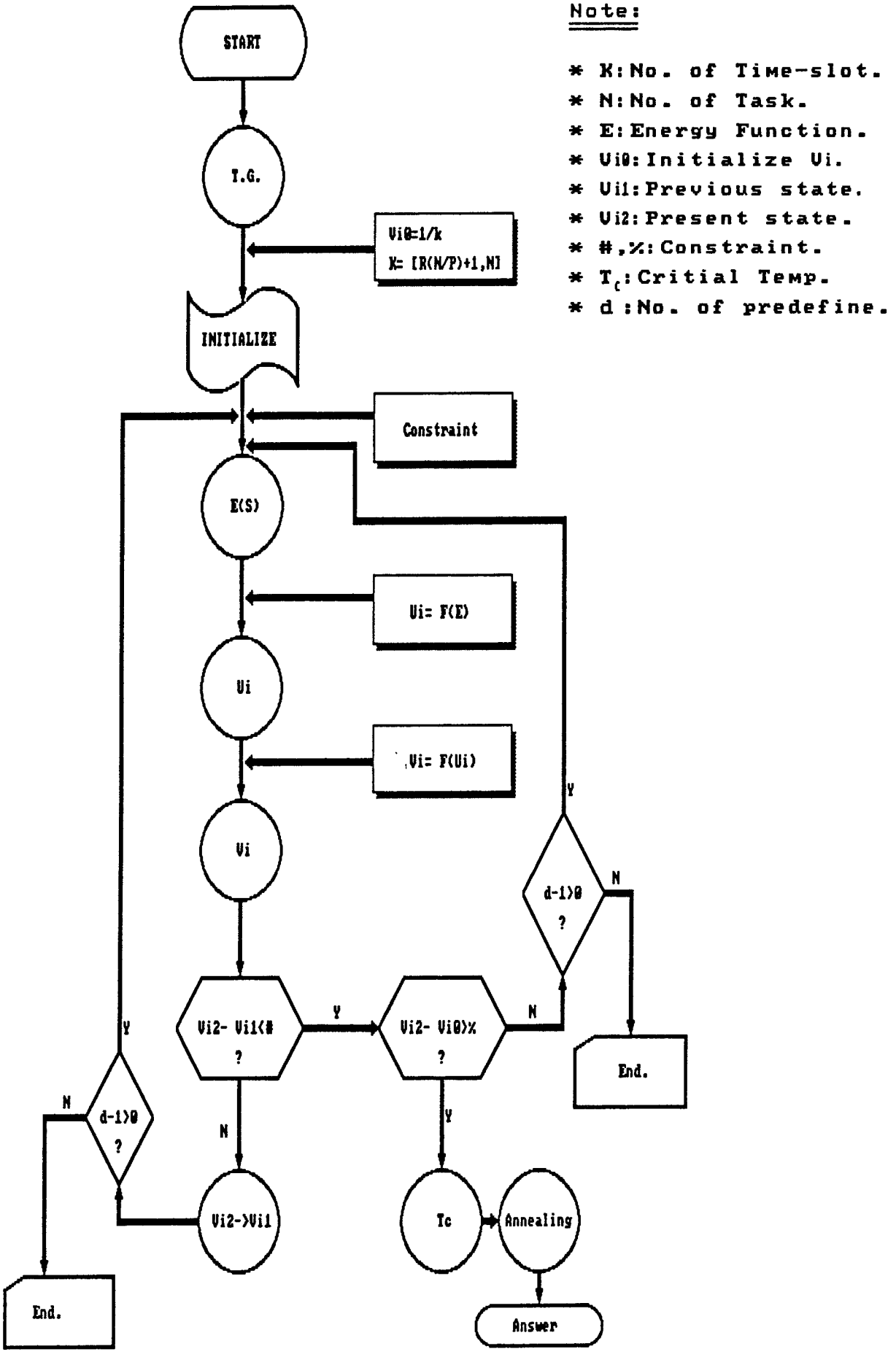

Figure 4.2: MFT Algorithm Flow Chart For Task Scheduling

Chapter 5

Simulation Results and Conclusions

5.1 Simulation Results

Consider the simple task graph, which consists of nine tasks, shown in Fig. 5.1. Following the algorithm procedure mentioned in Chapter 4, we can get the results shown in Fig. 5.2-5.5. In Fig. 5.2, the energy function ($E(S)$) was clamped in the right phase by the critical temperature (T_C). The local field (U_{ia}) versus the probability of each task (V_{ia}) by reduced encoding is shown in Fig. 5.3. The probability of each task (V_{ia}) is shown in Fig. 5.4 and 5.5 which are in same and different execution time. A more complicated task graph is shown in Fig. 5.6. The relationship between energy versus temperature and V_{ia} versus U_{ia} in the ninety tasks graph is shown in Fig. 5.7 and 5.8. Table 5.1 shows the finish time to complete the ninety tasks by using different number of processors.

The unit time in these examples is $\frac{1}{100}ms$, and the execution time for all task is shown on the right side of each task in the TG.

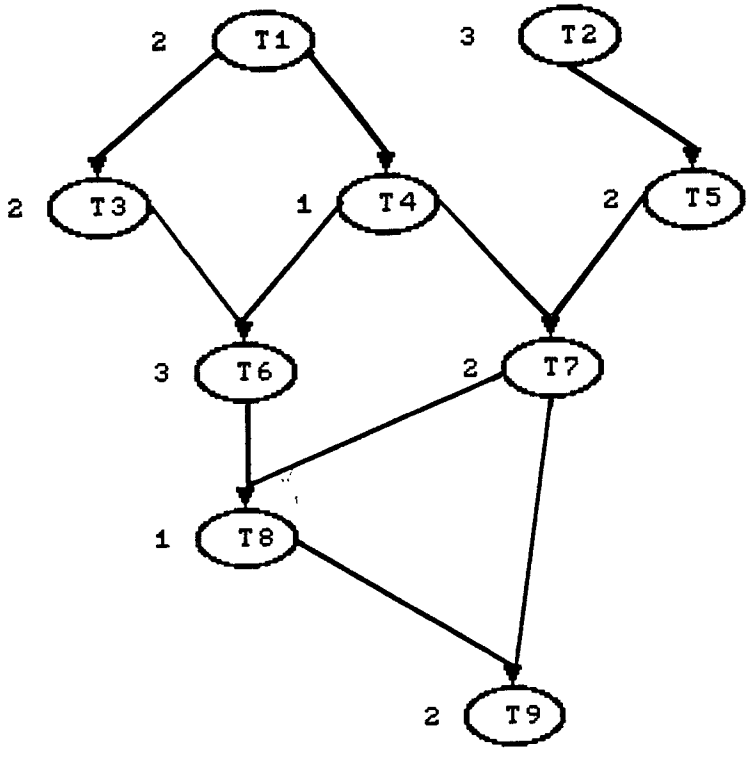

Figure 5.1: Nine Tasks Example.

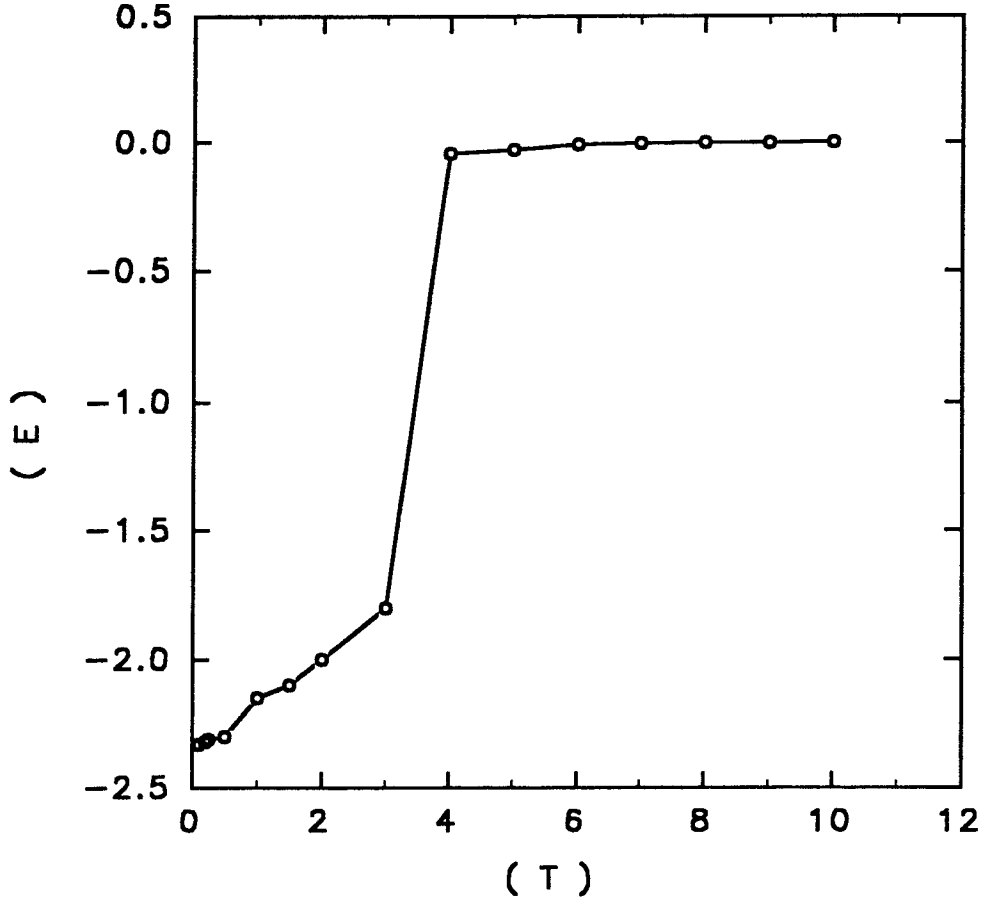

Figure 5.2: Energy versus Temperature

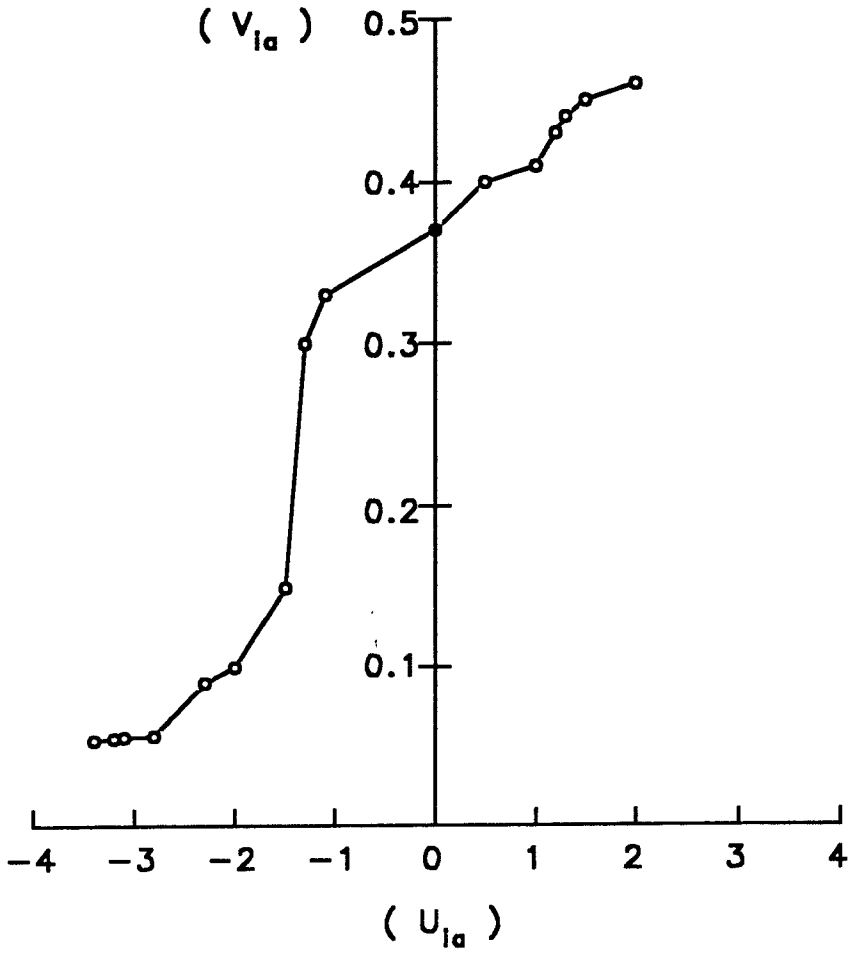

Figure 5.3: V_{ia} versus U_{ia}

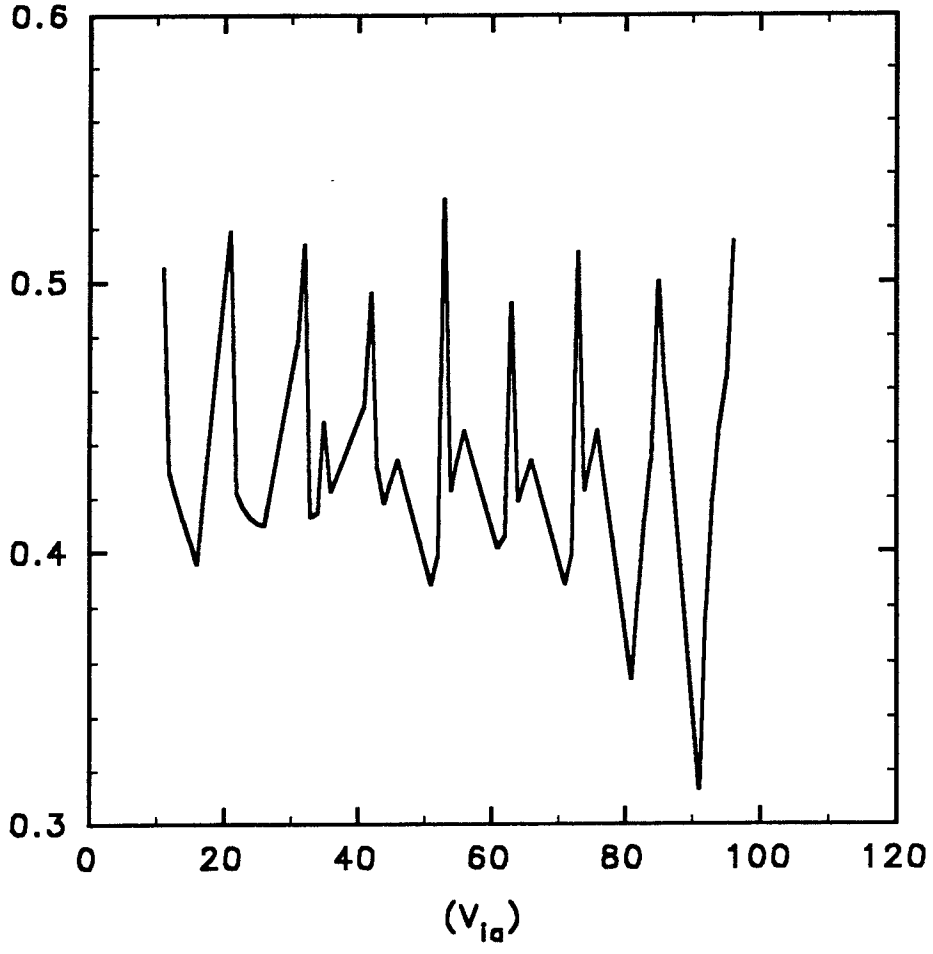

Figure 5.4: Same Execution Time Result

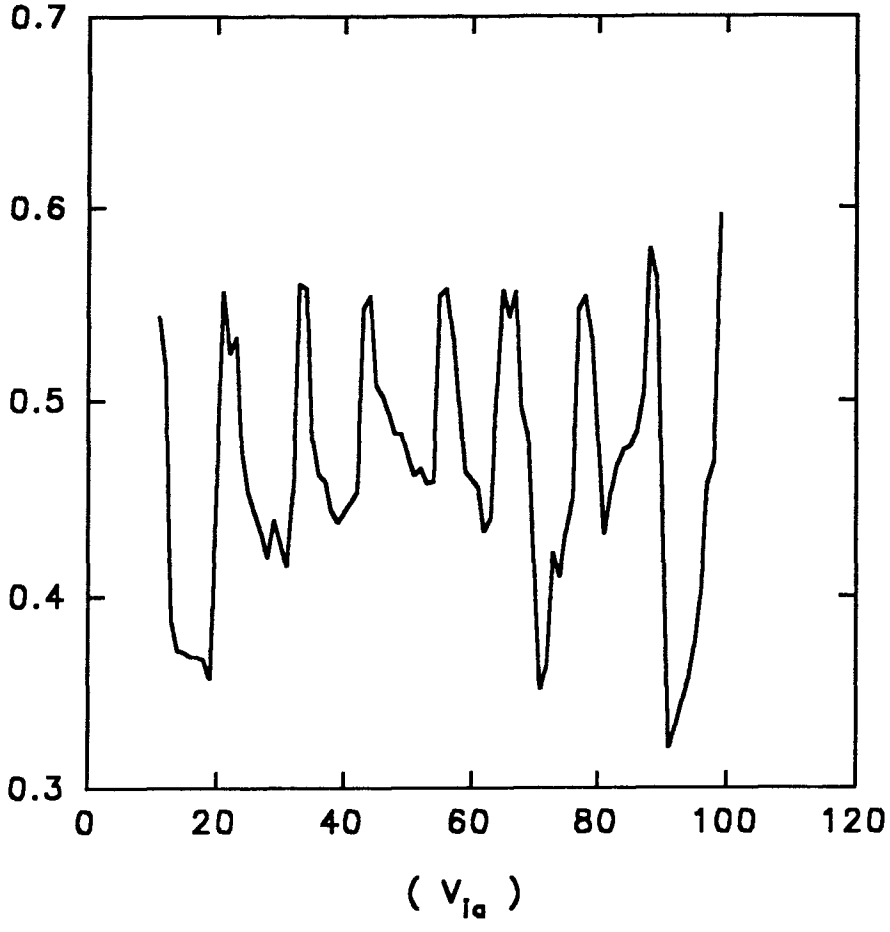

Figure 5.5: Different Execution Time Result

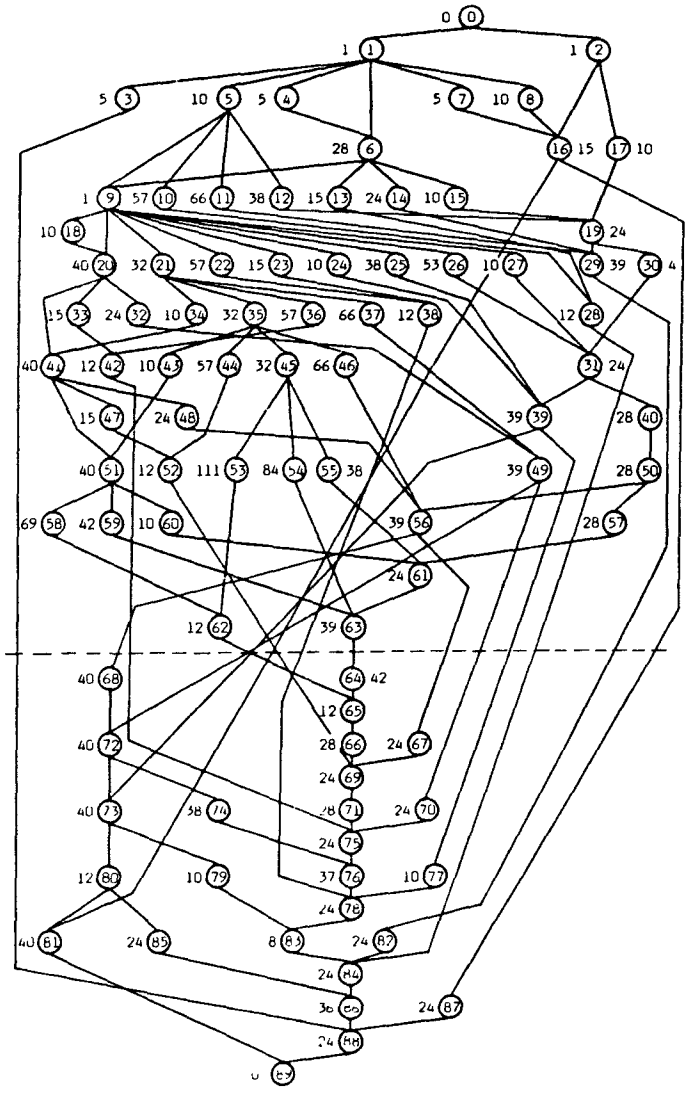

Figure 5.6: Ninety Tasks Example

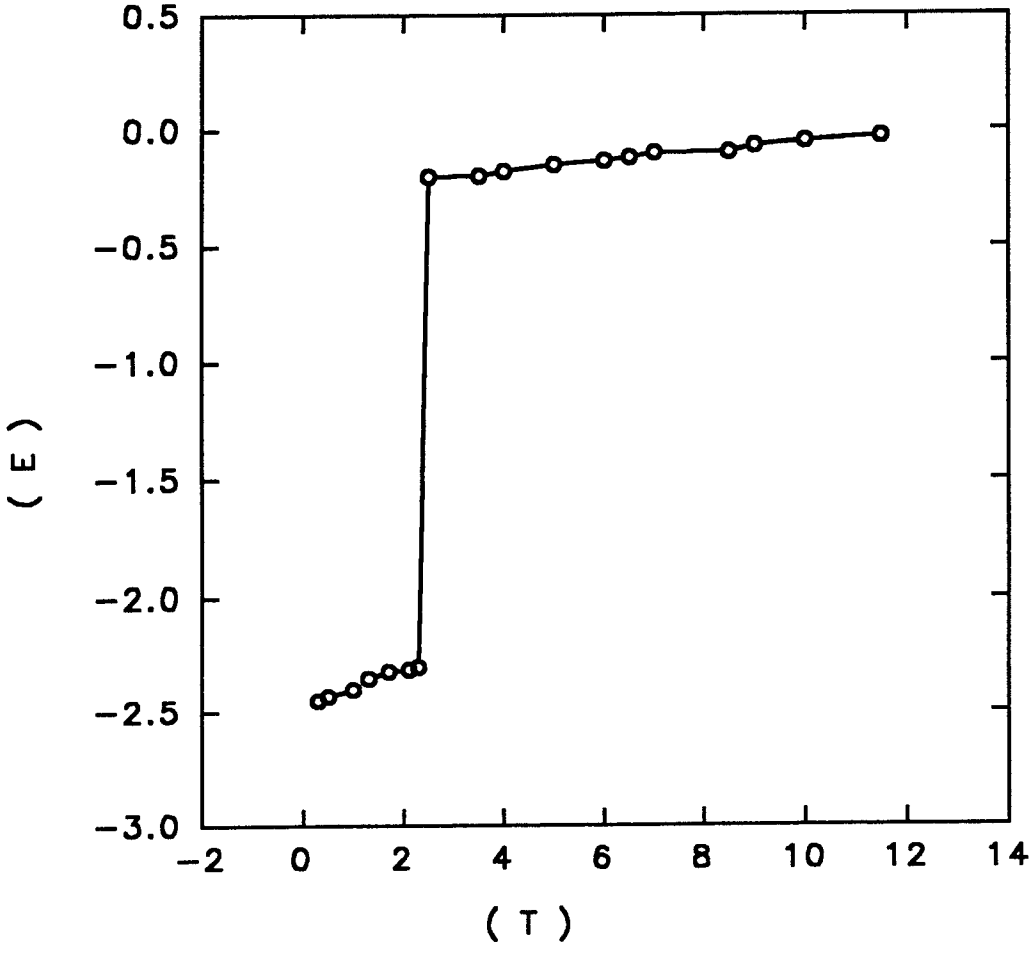

Figure 5.7: Energy versus Temperature

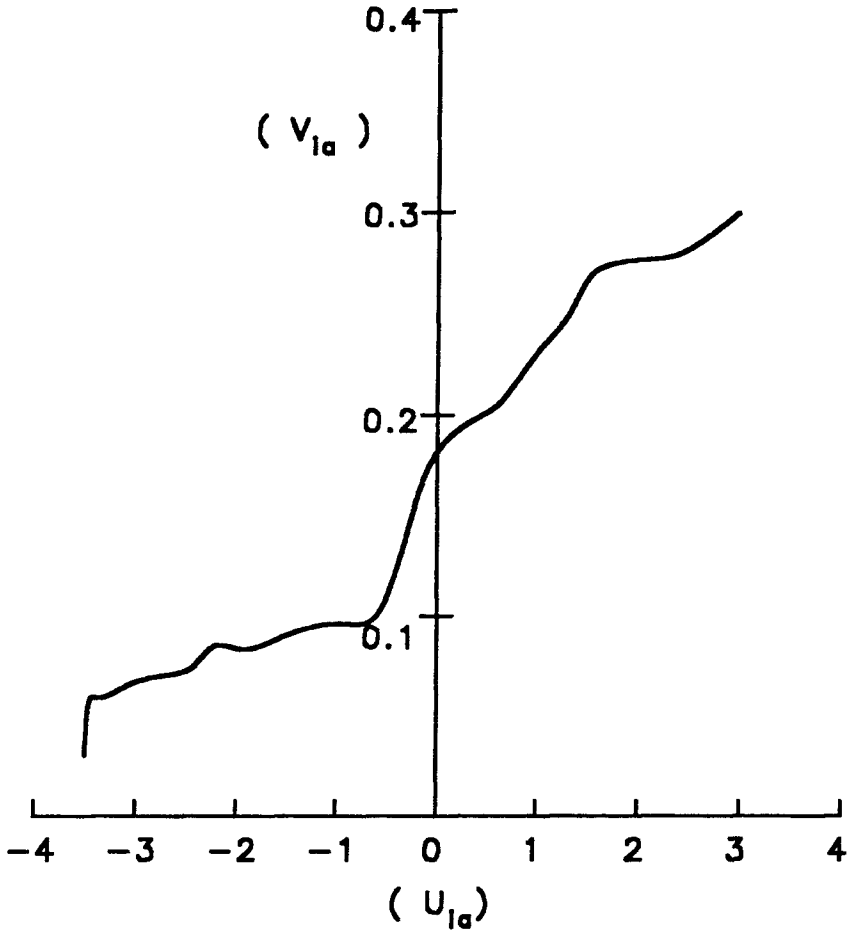

Figure 5.8: V_{ia} versus U_{ia}

Number of Processors	Processing Time (ms)	
	MFT solution	Optimal solution
2	14.21	12.42
3	10.65	8.43
4	8.45	6.59
5	7.11	5.86
6	7.35	5.73

Table 5.1: Comparison of MFT solution and optimal solution

5.2 Conclusions

In this thesis, we considered the problem of scheduling a task graph onto microprocessors system based on Mean Field Theory. The energy function, E(S), was set up to satisfy the physical constraints of the problem. Thus, from this the value of V_{ia} was found. It showed the probability of which task belongs to which time slot. We can get the simple results are the more accurate V_{ia} we got, the more efficient and reliable solutions we had and the more simple task graph we tested, the closer to optimal result we achieved.

After the simulation result, we can summary this result for future study. The constraint conditions, in Chapter 4, eliminated the precedence relations between tasks. That causes some tasks connecting illegally. The precedence of each task shall takes serious consideration. The other important condition is the value of slot (K). The K shall encode into the temperature annealing scheduling as a "cost" function.

Bibliography

[1] J. A. Anderson, and E. Rosenfeld, *Neurocomputing, Foundations of Research*, MIT Press, Cambridge, MA, 1988.

[2] G. Bilbro, R. Mann, *et al*, "Optimization By Mean Field Annealing," in *Advance In Neural Information Processing Systems 1*, ed. D. S. Touretzky, Morgan Kaufmann Pub., San Mateo, CA, 1989.

[3] C. L. Chen, C. S. G. Lee, and Edwin S. H. Hou, "Efficient Scheduling Algorithms for Robot Inverse Dynamics Computeration on a Multiprocessor Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 18, No. 5, pp. 729-743, Sept./Oct. 1988.

[4] C. C. Galland and G. E. Hinton, "Experiments on Discovering High Order Fertures with Mean Field Modules," *Technical Report CS-115 Dept. of Computer Science, U. of Toronto*, May 1989.

[5] E. S. H. Hou, H. Rong, and N. Ansari, "Efficient Multiprocessor Scheduling Base on Genetic Algorithms," *IECON* 90, *16th Conference of the IEEE Industrial Electronics Society,* November 27-30, 1990, Pacific Grove, CA, pp. 1239-1243.

[6] J. J. Hopfield, D. W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, Vol. 52, No. 4, pp. 141-156,

1985.

[7] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Science*, Vol. 79, pp. 2541-2554, 1982.

[8] C. Klimasauskas, J. Glluiver, and G. Pelton, "Neural-Works Professional-II User's Guide," *Neural Ware Inc.*, 1989.

[9] B. Kosko, "Bidirectional Associative Memories," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 18, No. 1, Jan./Feb. pp. 49-60, 1988.

[10] T. Kohonen, "An Introduction to Neural Computing," *Neural Networks*, Vol. 1, pp. 146-161, 1988.

[11] C. Peterson and B. Soderberg, "Teacher and Classes with Neural Networks," *International Journal of Neural Systems*, pp. 1-20, Oct. 1989.

[12] C. Peterson and B. Soderberg, "A New Method for Mapping Optimization Problems onto Neural Network," *International Journal of Neural Systems*, Vol. 1, No. 1, pp. 3-22, May 1989.

[13] C. Peterson and E. Hartman, "Explorations of the Mean Field Theory Learning Algorithm," *Neural Networks*, Vol. 2, pp. 475-494, 1989.

[14] C. Peterson, "Applications of Mean Field Theory Neural Networks," *Dept. of Theoretical Physics, Technical report CS-1153, U of Lund*, August, pp. 1-27, 1989.

[15] C. Peterson and J. Anderson, "Neural Networks and NP-Complete Optimization Problems; a performance study on the graph bisection problem," *Complex System*, Vol. 2, pp. 59-71, 1988.

[16] D. E. Rumelhart, and J. L. McClelland, *Parallel Distributed Processing*, Vol. 1, MIT Press, Cambridge, MA, 1986.

[17] J. Stanley, *Introduction to Neural Networks*, California Scientific Software, 1989.

[18] D. E. Van den Bout and T. K. Miller, "Graph Partitioning Using Annealed Networks," *IEEE Transactions on Neural Networks*, Vol 1. No. 2. pp. 192-203, June 1990.

[19] P. D. Wasserman, *Neural Computing Theory and Practice*, Van Nostrand Reinhold, NY, NY, 1989.

[20] C. Weiszmann, Technical Editor, *DARPA Neural Network Study*, AFCEA International Press, 1988.

[21] F. Y. Wu, "The Potts Model," *Review of Modern Physics*, Vol. 54, pp. 211-235, 1983.

[22] E. Yair and A. Gersho, "The Boltzmann Perceptron Network: A Multi-Layered Feed-Forward Network Equivalent To The Boltzmann Machine," in *Advances In Neural Information Processing Systems 1,* ed. D. S. Touretzky, Morgan Kaufmann Pub., San Mateo, CA, 1989.