Multiprocessor Scheduling Using Mean-Field Annealing

Shaharuddin Salleh
Dept of Mathematics
Faculty of Science
University of Technology Malaysia
KB 791, 80990 Johor Bahru
Malaysia

Albert Y. Zomaya
Parallel Computing Research Lab.
Dept of Electrical & Electronics Eng.
University of Western Australia
Perth, WA 6907
Australia

ABSTRACT

This paper presents our work on the static task scheduling model using the mean-field annealing (MFA) technique. Mean-field annealing is a technique of thermostatic annealing that takes the statistical properties of particles as its learning paradigm. It combines good features from the Hopfield neural network and simulated annealing, to overcome their weaknesses and improve on their performances. Our MFA model for task scheduling is derived from its prototype, namely, the graph partitioning problem. MFA is deterministic in nature and this gives the advantage of faster convergence to the equilibrium temperature, compared to simulated annealing. Our experimental work verifies this finding on various network and task graph sizes. Our work also includes the simulation of the MFA model on several network topologies and parameters.

Keywords: *task scheduling, mean-field annealing and graph partitioning.*

1. INTRODUCTION

Task scheduling problem is a combinatorial optimization problem that is known to have large interacting degrees of freedom and is generally classified as NP-complete [1]. Most solutions to the problem have been proposed in the form heuristics, using integer programming, queueing theory, graph theoretic and enumerated search.

In this paper, we present an approach using the mean-field annealing technique for scheduling tasks onto a set of processing elements (PEs). The objectives are to obtain a near-optimal solution for the minimum interprocessor communication, while trying to balance the load on all PEs. The paper is organized into six sections. Section 1 is the introduction which describes the work in general. Section 2 describes the background to mean-field annealing. Section 3 is our mean-field model for the problem, while Section 4 is its algorithm. In Section 5, we present the MFA model that supports the machine topological parameters. The results from simulation using both models are presented in Section 6. Finally, Section 7 is the summary and conclusion.

2. BACKGROUND TO MEAN-FIELD ANNEALING

Statistical mechanics is an area in physics which describes the slow Ising Hamiltonian process of thermal cooling for spin particles with many degrees of freedom until they reach their equilibrium states [2]. The annealing particles in solids provide a framework for optimization of the properties of very large and complex systems. This idea is now incorporated into algorithms for solving several prototype combinatorial optimization problems such as the graph partitioning problem. *Graph partitioning* is a problem where nodes from an undirected graph are partitioned into their bins so as to minimize the number of inter-bin links of the nodes. The graph partitioning problem provides a very useful framework for solving the multiprocessor scheduling problem as it relates the minimum cut size criteria of the former to the minimum interprocessor communication of the latter.

Hopfield neural network (HNN) (Hopfield and Tank [3]) is a fully-connected, recurrent network which has a strong potential for hardware implementation due to the collective dynamical properties of its interacting neurons. In terms of graph theory, HNN is a complete graph that is based on simple asynchronous updating where only one unit of its neurons, selected at random, is involved at each time step. Unlike feedforward networks, the network involves cyclic connections that make them behave as a nonlinear dynamic system.

A Hopfield neural network minimizes the cost function encoded in its weights by performing gradient descent. It has been shown in the Lypanov stability theory that the energy $E(\mathbf{x})$ converges to a stable solution if ${dE}/{dt} < 0$. However, this energy function may contain plenty of local minima. The probability of getting stuck to a local minimum is very high, and this makes it difficult to converge to its global minimum which corresponds to the desired solution. The network generates reasonably good solutions for small problems but performs poorly when the size of the problem increases.

The *stochastic simulated annealing* (SSA) ([2,4]) combines the gradient descent technique which is a probabilistic hill-climbing algorithm with a random process to find the global minimum for its energy function E. Simulated annealing models the degrees of freedom as a collection of atoms slowly being cooled into their stable states with the temperature T as the controlling parameter. The energy surface defined as $E(s)$ for a particle state s is a Boltzmann distribution function that allows changes in s to increase E, thus providing the network with a mechanism to escape from being trapped in a local minimum. This is made possible since changes to $\mathbf{x}$ which decrease E are always accepted, whereas a move which causes an increase ΔE will be taken with the Boltzmann probability, $\Pr\{\text{uphill move}\} = \exp(-\Delta E/T)$.

The main drawback in simulated annealing is the long computation time before converging to its stable state. This is due to the fact that it performs a large number of random searches at each temperature step before arriving at its equilibrium state.

3. THE SCHEDULING MODEL

In their work, Peterson and Anderson [5,6] introduced the *mean-field annealing* (MFA) approach as a new method to speed up the Boltzmann machine learning algorithm. Originally, this method was derived from the well-known mean-field theory approximation in physics for spin systems.

MFA performs an approximation to simulated annealing to replace some of the stochastic properties with deterministic values. This approach combines the characteristics of simulated annealing process of relaxation with a suitable neural network structure such as the Hopfield network. The technique is said to be superior to SSA as it approximates solutions deterministically rather than performing the slow stochastic hill-climbing search and, as a result, leads to a faster convergence to its solution. In addition, MFA can be represented as an intrinsically parallel algorithms which can then be mapped to a set of nonlinear differential equations.

In their work, Van den Bout and Miller [7] successfully produced a graph partitioning model based on mean-field annealing. This work has been widely referred as a prototype to several other difficult problems in the area. Its application in the multiprocessor scheduling problem using MFA has been demonstrated in Bultan and Aykanat [8]. The proposed energy function in [8] considers the general scheduling condition where the precedence constraint of the tasks is less emphasized. We extend the work by proposing a new equation that supports this important element, as follows [7]:

$$E(\mathbf{s}) = \sum_{j=1}^{J} \sum_{i \neq j} \sum_{k=1}^{K} c_{ij} (1 - s_{ik} s_{jk}) s_{jk} - \lambda \sum_{j=1}^{J} \sum_{i \neq j} \sum_{k=1}^{K} (1 - s_{ik} s_{jk}) s_{jk} \tag{1}$$

Equation (1) represents the energy function for the static task scheduling problem which supports the task precedence constraint. The first term in the equation is applied to minimize the interprocessor communication, while the second is to balance the load with λ as the repulsion coefficient. The constant c_{ij} is the intertask communication for $\mathrm{TS}_i<\mathrm{TS}_j$, where its value is 0 if the two tasks are not related, and $s_{jk} = \begin{cases} 1 & \text{if } \mathrm{TS}_j \in \mathrm{PE}_k \\ 0 & \text{if } \mathrm{TS}_j \notin \mathrm{PE}_k \end{cases}$. The first term of Equation (8) evaluates the total sum of the interprocessor communication cost. The product $s_{ik}s_{jk}$ is the probability that both TS_j and its predecessor TS_i are mapped to the same PE. The term $(1-s_{ik}s_{jk})$ implies that the communication cost is 0 if both TS_j and its predecessor TS_i are mapped to the same PE. The term s_{jk} inside $(1-s_{ik}s_{jk})s_{jk}$ suggests that the communication cost is 0 if TS_j is not mapped on PE_k.

MFA simulates the Hopfield analog model and modifies the simulated annealing technique where the stochastic binary neurons whose states defined as $s_{jk}\in\{0,1\}$ at the corners of the hypercube are replaced by analog neurons with deterministic outputs $v_{jk}\in[0,1]$. Peterson and Anderson [5,6] demonstrated that the discrete sum in the Boltzmann probability distribution function can be replaced by multiple nested integrals over the continuous variables. Using the saddle point expansion of the partition function F and by setting $\partial F/\partial u_j = 0$ and $\partial F/\partial v_j = 0$, we obtain the mean-field theory equations as follows:

$$u_{jk} = -\frac{1}{T}\frac{\partial E}{\partial v_{jk}} = \frac{1}{T}\left(\sum_{j=1}^{J} \sum_{i \neq j} \sum_{k=1}^{K} c_{ij} (1 - 2v_{ik} v_{jk}) - \lambda \sum_{j=1}^{J} \sum_{i \neq j} \sum_{k=1}^{K} (1 - 2v_{ik} v_{jk}) \right) \tag{2}$$

where u_j is called the mean field of the spin. It follows that the new spin values at the jth row using Equation (2) are given by

$$v_{jk} = \frac{1}{2}\left[1 + \tanh\left\{ \frac{1}{T}\left(\sum_{j=1}^{J} \sum_{i \neq j} \sum_{k=1}^{K} c_{ij} (1 - 2v_{ik} v_{jk}) - \lambda \sum_{j=1}^{J} \sum_{i \neq j} \sum_{k=1}^{K} (1 - 2v_{ik} v_{jk}) \right) \right\} \right] \tag{3}$$

Finally, the energy change as a result of the spin updates in the spin row j for k elements is given by:

$$\Delta E = \sum_{k=1}^{K} u_{jk}(v_{jk}^{\text{new}} - v_{jk}^{\text{old}}) \tag{4}$$

4. MFA TERMINOLOGY AND ALGORITHM

Several terminologies are used in this work. We denote l_j as the length of the task TS$_j$. The parameter δ_{ij} represents the partial order TS$_i$<TS$_j$ which indicates TS$_i$ is the predecessor of TS$_j$, defined as $\delta_{ij} = \begin{cases} 1 \text{ if } \mathrm{TS}_i < \mathrm{TS}j \\ 0 \quad \text{otherwise} \end{cases}$. The intertask communication cost c_{ij} is the cost, in unit time, in transmitting data from TS$_i$ to TS$_j$, for TS$_i$<TS$_j$. From this relationship, the following terminology is used [9]:

$$\mathrm{TS}_j \text{ high ready time} = \mathrm{TS}_j.\mathrm{hrt} = \max_{i=1}^{J}\{\delta_{ij}\mathrm{TS}_j.\mathrm{ct} + c_{ij}\} \tag{5a}$$

$$\mathrm{TS}_j \text{ low ready time} = \mathrm{TS}_j.\mathrm{lrt} = \max_{r \neq i}^{J}\{\delta_{rj}\mathrm{TS}_r.\mathrm{ct} + c_{rj}\} \tag{5b}$$

for i,k=1,2,...,J and TS$_s$ denotes the predecessor task with TS$_j$.hrt (that is, TS$_j$.hrt= TS$_s$.ct+c_{is}). In addition, we define the *processor ready time* PE$_k$.prt as the earliest time PE$_k$ becomes available and is ready to accept TS$_j$. The value of PE$_k$.prt is determined from the maximum of the completion time of PE$_k$ last task, TS$_{Lk}$.ct and the high or low ready time, as follows:

$$\mathrm{PE}_k.\mathrm{prt} = \begin{cases} \max(\mathrm{TS}_{Lk}.\mathrm{ct}, \mathrm{TS}_j.\mathrm{hrt}) \text{ if } k \neq s \\ \max(\mathrm{TS}_{Lk}.\mathrm{ct}, \mathrm{TS}_j.\mathrm{lrt}) \text{ if } k = s \end{cases} \tag{6}$$

Our model is implemented as Algorithm `TS_MFA-1` in [9], as follows:

```
/* Algorithm TS_MFA-1 */
Initialise $\Gamma$ and $\Pi$;
Evaluate TS$_j$.lev for $j$=1,2,...,$J$;
Select suitable values for $T$=$T_0$, and the reducing
     parameter $\alpha$, for 0<$\alpha$<1;
Initialise $v_{jk}$ with random values in 0≤$v_{jk}$≤1;
While $T$ is in the cooling range
   While $E$ is decreasing
      Select a row $j$ at random for the spin $v_{jk}$;
      For $k$=1 to $K$
         Compute the mean fields $u_{jk}$ from (2);
         Compute the new spin values $v_{jk}$ from (3);
         Compute the energy change $\Delta E$ from (4);
         Update the spin values given by $v_{jk} := v_{jk}^{new}$;
   Repeat until $\Delta E$ is not significant;
   Update $T$ according to $T^{(new)} := \alpha T^{(old)}$;
Evaluate the new schedule;
```

MFA starts with initial random spin values $0 \leq v_{jk} \leq 1$. Some suitable initial values for the thermostatic temperature T_0 and the reducing parameter α are also chosen. Once the initial values are set, we begin the relaxation process at $T=T_0$ by choosing a row j at random. The neuron is perturbed and this generates new values for u_{jk} and v_{jk} for $k=1,2,...,K$, using Equations (2) and (3), respectively. The change in the neuron energy ΔE from its earlier value is computed using Equation (4). If $\Delta E \leq 0$ then the spin values v_{jk} for the row are updated with the new values. Otherwise, the old values are maintained. The process repeats with the perturbation of another neuron at random. At every cycle, the energy change is observed closely as it determines whether the selected neuron updates its values or not.

When random perturbations of neurons bring no significant energy change, the energy is said to have stabilized at the current temperature. The next step is to repeat annealing. The temperature T is reduced according to $T^{(\text{new})} := \alpha T^{(\text{old})}$ for $0<\alpha<1$, and the process in the last paragraph is applied until the energy stabilizes again. This whole process is repeated for several iterations until a stopping criteria is reached. In [5] and [7], the stopping criteria is determined when several annealing iterations fail to produce a significant energy change. The temperature that first exhibits this observation is called the *critical temperature* T_c, and MFA is said to have reached its equilibrium at this temperature.

The new schedule for TS_j is determined by evaluating its actual start time TS_j.ast according to the following equation:

$$\text{TS}_j.\text{ast} = \max_{i=1}^{J} \left[\delta_{ij} \text{TS}_i.\text{ct} + c_{ij}(1 - v_{ik} v_{jk}) v_{jk} \right] \tag{7}$$

5. MACHINE TOPOLOGY DEPENDENCE

The main obstacle to the high performance of a scheduling algorithm is the presence of communication delay in transmitting a message through the machine network. The problem arises when information is to be transmitted from a task in one PE to another task in a different PE. The information comes in the form of data and instructions, and it is tranmitted through the network in packets.

The magnitude of the delay varies according to several factors, such as the algorithm used, the network topology and the structure of the problem. The following equation represents the communication delay $C(j_0, j_n, k_0, k_n)$ between two tasks TS_{j_0} and TS_{j_n}, located in PE_{k_0} and PE_{k_n}, respectively, by hopping through $\text{PE}_{k_1}, \text{PE}_{k_2}, ..., \text{PE}_{k_{n-1}}$ in the network (El-Rewini, [10]):

$$C(j_0, j_n, k_0, k_n) = \left(\frac{d(j_0, j_n)}{R} + IM \right) * H(k_0, k_n) + CD \tag{8}$$

In the above equation, n-1 is the number of PEs hopped for the transmission, $d(j_0, j_n)$ is the amount of data for the transmission and $H(k_0, k_n)$ is the distance, or the number of hops between PE_{k_0} and PE_{k_n}. It is assumed that the network consists of homogeneous PEs with the same processing speed, the same rate IM for the PEs in initiating a message, and the same transmission rate R and contention delay CD between any two PEs.

We propose a scheduling model called TS_MFA-2 [9] that considers the machine topology factor. This model is based on TS_MFA-1, with the addition of the machine dependent factor. The same objective function of Equation (1) is used in the model. However, the communication cost c_{ij} between two tasks TS_i and TS_j is now a variable that depends on the assigned PEs.

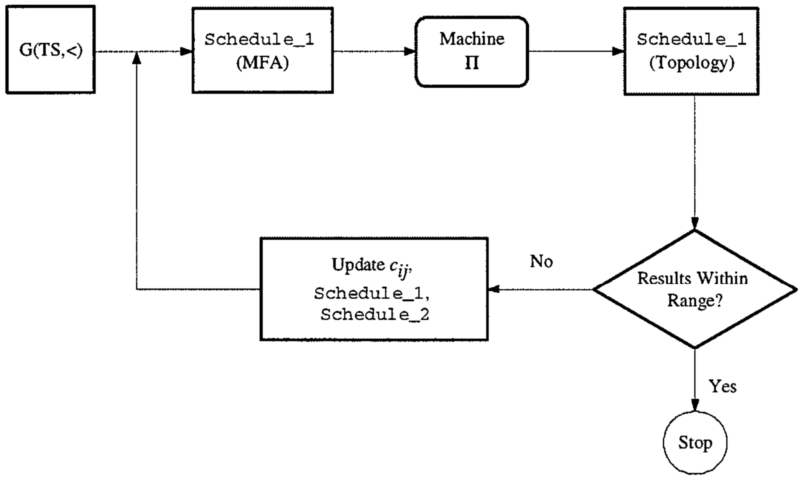


Fig. 1: TS_MFA-2 scheduling organization

Figure 1 shows the organization of TS_MFA-2. The process begins with the mapping of the task graph **G**(TS,<) using Algorithm TS_MFA-1, with the initial values of c_{ij} taken from **G**(TS,<). At equilibrium, Schedule_1 is generated. The schedule is then inserted into the machine on the given topology, to produce Schedule_2. This step also produces $C(i,j,k,l)$ from Equation (8), for $TS_i \in PE_k$ and $TS_j \in PE_l$. The next step is to check if the PE most minimum completion time result in Schedule_2 is within 5% of the result in Schedule_1. If this is the case, then Schedule_2 is accepted as the final schedule and the process stops.

If the result in Schedule_2 is not within 5% of Schedule_1 then the values of c_{ij} in Equation (8) are updated according to $c_{ij} := C(i,j,k,l)$. The current results of Schedule_1 and Schedule_2 are stored as Schedule_1' and Schedule_2', respectively, to be used as the testing criteria against the results in the following cycle. The process continues by re-evaluating Schedule_1 using TS_MFA-1, and the same cycle is repeated until the best result from Schedule_2 is obtained. The stopping criteria can be determined in many ways, such as limiting the number of iterations and comparing the results of Schedule_2 against Schedule_1. In our model, the stopping criteria is determined as follows:

(1) the result in Schedule_2 is within 5% of Schedule_1, or
(2) if the above criteria is not met then the cycle is repeated to some number of iterations M. In our case, we set M to be not more than 10. The best result produced by Schedule_2 within the iterations is the final schedule.

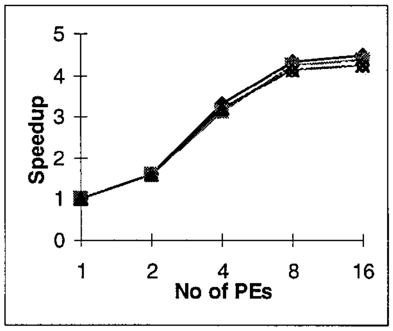
6. EXPERIMENTAL WORK

6.1 TS_MFA-1 IMPLEMENTATION

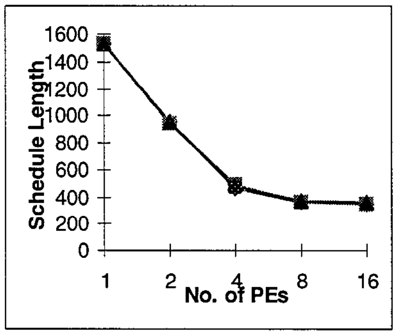
In this section, Algorithm TS_MFA-1 is implemented to simulate MFA scheduling. The simulation runs on Intel 80486 single-processor machine, and it assumes a fully-connected network with no machine topology factors. The performance of the mean-field annealing technique is measured in terms of speedup, schedule length, and the PE utilization rate (pur) [9].

We compare the performances of the mean-field annealing technique against the general list heuristic (GL) in El-Rewini [10], and the simulated annealing technique using Algorithm TS_SSA in [9]. GL is a list scheduling technique that gives high priority for scheduling to tasks that are ready in their priority list. In its implementation, the first ready task in the list is assigned to the first ready processor, and the process repeats until all tasks have been assigned.

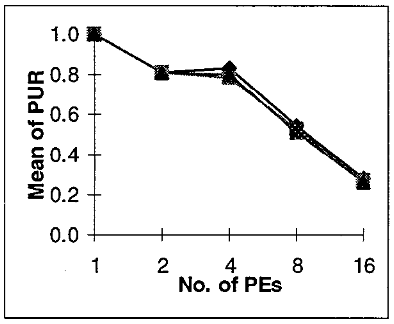
The experimental work consists of a study on the performances of the three algorithms in cases of 1, 2, 4, 8 and 16 PEs. Initial values in TS_SSA and TS_MFA-1 are given as follows: T_0=100 and α=0.95. A task graph consisting of 400 tasks is used in all cases of the experimental work..



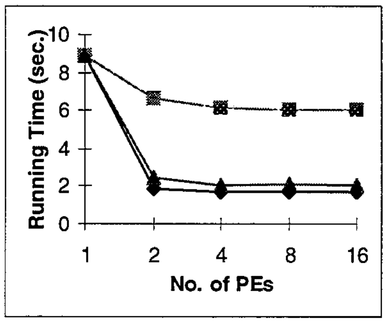
(a)



(b)



(c)



(d)

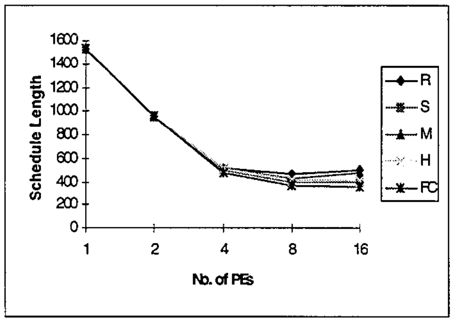
Fig. 2: Comparison in the performances of the three methods

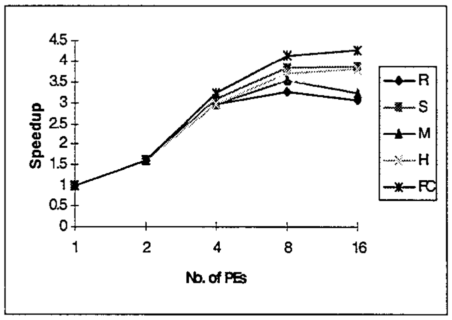
Figure 2(a) compares the speedup performances of TS_MFA-1 against TS_SSA and GL. In general, the speedup in all the three models increases as the number of PEs is increased. It is noted that the speedup performances of GL and TS_SSA are both better than TS_MFA-1. The results from TS_MFA-1 are acceptable as they are within 5% of the other two.

Figure 2(b) shows the schedule length performances from the three algorithms. The lengths tend to decrease significantly when the number of PEs increases. Figure 2(c) analyses the mean of the PE utilization rate (μ_{pur}). From the graph, the performances of all the three algorithms are almost similar: the mean drops as the number of PEs increases in each case. The results show that the distribution of tasks deteriorates with this increase. Clearly, this problem is caused by the increase in the communication cost. Finally, Figure 2(d) gives the comparison in the running time (in seconds) for the 400 tasks from the same task graph model. From the figure, the fastest scheduling times are recorded in the order GL, TS_MFA-1 and TS_SSA, in the network models. The results also show TS_MFA-1 performs about 300% faster than TS_SSA, in all cases.

6.2 TS_MFA-2 IMPLEMENTATION

We perform a simulation using TS_MFA-2 on five models of computing platform: fully-connected (FC), ring (R), star (S), mesh (M) and hypercube (H). The experimental work involves a task graph of 400 tasks with the precedence and communication constraints. The input data includes the tasks length, their precedence relationship with other tasks and the communication cost involved. The same set of data is run on the five topological models, on cases of 1, 2, 4, 8 and 16 PEs. The mesh (M) network assumes a rectangular 4x2 grid in the 8-PE model, and square grid in all other PE models.





(a) (b)

Fig. 3: Comparison of the schedule length and speedup performances

In the experiment, we set the following values in the initialization: R=1, IM=0, and CD=0. Figure 3(a) shows the schedule length results obtained from the five networks. In each of these cases, all models show improvement in their schedule length when the number of PEs is increased. The best performances are recorded in descending order, as follows: FC, S, H, M and R. This observation is expected as the interprocessor communication is at its best in FC, and worst in R. The same analysis is produced in the speedup performances of the model, as shown in Figure 3(b).

7. SUMMARY AND CONCLUSION

In this paper, we presented the mean-field annealing approach to solve the multiprocessor scheduling problem. The Hopfield neural network has a strong hardware potential where maximum parallelism can be be applied, but the network does not have a good scaling property. Simulated annealing, on the other hand, is easy to implement as it uses the Boltzmann machine for the search to escape from being trapped in a local minimum. However, its stochastic mechanism is too slow for this convergence. Mean-field annealing performs better as it combines the best features from these two approaches.

Our experimental shows some convincing results from the mean-field annealing approach. In general, MFA produces good load balancing results, and performs within 5% of the general list heuristic (El-Rewini, 1990) and simulated annealing approaches, in terms of schedule length. The model also converges to the equilibrium 300% faster than simulated annealing.

REFERENCES

[1] M.R.Garey and D.S.Johnson, "Computers and intractability: a guide to the theory of NP-completeness", *W.H.Freeman and Co.*, 1979.

[2] S.Kirkpatrick, C.D.Gelatt and M.P.Vecchi, "Optimization by simulated annealing", *Science*, vol.220, no.4598, 1983, pp.671-678.

[3] J.J.Hopfield and D.W.Tank, "Neural computation of decisions in optimization problems", *Biological Cybernetics*, vol.52, 1985, pp. 141-152.

[4] A.Y.Zomaya and R.Kazman, "Simulated annealing techniques", to appear in *Handbook of Algorithms and the Theory of Computation*, M.J.Atallah (ed.), CRC Press, Florida.

[5] C.Peterson and J.R.Anderson, "A mean-field theory learning algorithm for neural networks", *Complex Systems*, vol.1, 1987, pp. 995-1019.

[6] C.Peterson and J.R.Anderson, "Neural networks and NP-complete optimization problems; a performance study on the graph bisection problem", *Complex Systems*, vol.2, 1988, pp. 59-89.

[7] D.E.Van den Bout and T.K.Miller, "Graph partitioning using neural network", *IEEE Trans. Neural Networks*, vol.1, no.2, 1990, pp. 192-202.

[8] T.Bultan and C.Aykanat, "A new mapping heuristic based on mean field annealing", *Journal of Parallel and Distributed Computing*, vol.16, pp. 292-305.

[9] S.Salleh, "Task scheduling for parallel processing systems using neural network models", Ph.D Thesis, Dept of Mathematics, University of Technology Malaysia, 1998 (submitted).

[10] H.El-Rewini, "Task partitioning and scheduling on arbitrary parallel processing systems", *Ph.D Thesis*, Dept of Computer Science, Oregon State University, 1990.