Parallel Processing: CONPAR 94 - VAPP VI
B. Buchberger, J. Volkert (Eds.)

An Efficient Mapping Heuristic for Mesh-Connected Parallel Architectures Based on Mean Field Annealing

İsmail Haritaoğlu and Cevdet Aykanat

Dept.of Comp. Eng. & Info. Scn., Bilkent University, Ankara, TURKEY
hismail@bilkent.edu.tr

Abstract. A new Mean Field Annealing (MFA) formulation is proposed for the mapping problem for mesh-connected architectures. The proposed MFA heuristic exploits the conventional routing scheme used in mesh interconnection topologies to introduce an efficient encoding scheme. An efficient implementation scheme which decreases the complexity of the proposed algorithm by asymptotical factors is also developed. Experimental results also show that the proposed MFA heuristic approaches the speed performance of the fast Kernighan-Lin heuristic while approaching the solution quality of the powerful simulated annealing heuristic.

1 Introduction

The mapping problem arises as parallel programs are developed for distributed memory architectures. Various classes of problems can be decomposed into a set of interacting sequential subproblems (tasks) which can be executed in parallel. In these classes of problems, the interaction patterns among the tasks is static. Hence, the decomposition of the algorithm can be represented by a static undirected task graph referred here as *Task Interaction Graph* (TIG). Vertices of this graph represent the atomic tasks and the edge set represents the interaction pattern among the tasks. Vertices can be associated with weights which denote the relative computational costs of the respective tasks. Each edge denotes the need for the bidirectional interaction between the corresponding pair of tasks at the completion of the execution of those two tasks. Edges can also be associated with weights which denote the amounts of bidirectional information exchanges involved between the respective pairs of tasks. In a distributed-memory architecture, a pair of processors communicate with each other over a shortest path of links connecting them. Hence, communication between each pair of processors can be associated with a relative unit communication cost (communication cost per unit information). Unit communication cost between a pair of processors can be assumed to be linearly proportional to the shortest path distance between those two processors. Hence, the communication topology of the parallel architecture can be modeled by an undirected complete graph, referred here as *Processor Communication Graph* (PCG). The nodes of the PCG represent the processors and the weights associated with the edges represent the unit communication costs between processor pairs.

The objective in mapping TIG to PCG is the minimization of the expected execution time of the parallel program on the target architecture. Thus, the mapping problem can be modeled as an optimization problem by associating the following quality measures with a good mapping : (i) interprocessor communication overhead should be minimized, (ii) computational load should be uniformly distributed among processors in order to minimize processor idle time.

A mapping problem instance can be formally represented with two undirected graphs; TIG and PCG. The TIG $G_T(T, I)$, has $|T| = N$ vertices, labeled as $(1, 2, \ldots, i, j, \ldots, N)$, which represents the atomic tasks of the parallel program. Vertex weight w_i denotes the computational cost associated with task i for $i = 1, 2, \ldots, N$. Edge weight e_{ij} denotes the volume of interaction between tasks i and j connected by edge $(i, j) \in I$. The PCG $G_{\mathcal{R}}(\mathcal{R}, \mathcal{D})$, is a complete graph with $|\mathcal{R}| = K$ nodes and $|D| = C(K, 2)$ edges where $C(\cdot, \cdot)$ denotes the combinational operator. Nodes of the $G_{\mathcal{R}}$, labeled as $(1, 2, \ldots, k, l, \ldots, K)$, represent the processors of the target multicomputer. Edge weight d_{kl}, for $k, l = 1, 2, \ldots, K$ and $k \neq l$, denotes the unit communication cost between processors k and l. Given an instance of the mapping problem with the TIG $G_T(T, I)$ and the PCG $G_{\mathcal{R}}(\mathcal{R}, \mathcal{D})$, the question is to find a many-to-one mapping function $M : T \rightarrow \mathcal{R}$, which assigns each vertex of the graph G_T to a unique node of the graph $G_{\mathcal{R}}$, and minimizes the total interprocessor communication cost

$$C = \sum_{(i,j) \in I, M(i) \neq M(j)} e_{ij} d_{M(i),M(j)}$$

while maintaining the computational load of each processor k

$$W_k = \sum_{i \in T, M(i) = k} w_i, \qquad \text{for} \qquad k = 1, 2, \ldots, K$$

balanced. Here, $M(i) = k$ denotes the label of the the processor that task i is mapped to. Each edge (i, j) of the G_T contributes to the communication cost, only if vertices i and j are mapped to two different nodes of the $G_{\mathcal{R}}$, i.e., $M(i) \neq M(j)$. The amount of contribution is equal to the product of the volume of interaction e_{ij} between these two tasks and the unit communication cost $d_{M(i),M(j)}$ between processors $M(i)$ and $M(j)$. The computational load of a processor is the summation of the weights of the tasks assigned to that processor. Perfect load balance is achieved if $W_k = (\sum_{i=1}^{N} w_i)/K$ for each processor k.

Since the mapping problem is NP-hard [8, 10], heuristics giving suboptimal solutions are used to solve the problem [2, 3, 12]. Kernighan-Lin (KL) [7] and Simulated Annealing (SA) [8] heuristics are two attractive algorithms widely used for solving the mapping problem [3]. In a recent work [1], we have successfully formulated a recently proposed algorithm, called Mean Field Annealing (MFA) for solving the mapping problem. MFA merges collective computation and annealing properties of Hopfield neural networks [6] and SA [8], respectively, to obtain a general algorithm for solving combinatorial optimization problems. MFA can be used for solving a combinatorial optimization problem by choosing a representation scheme in which the final states of the spins can be decoded as a solution to the target problem. Then, an energy function is constructed whose global minimum value corresponds to the *best solution* of the target problem. MFA is expected to compute the best solution to the target problem, starting

from a randomly chosen initial state, by minimizing this energy function. Steps of applying MFA technique to a problem can be summarized as follows.

1) Choose a representation scheme which encodes the configuration space of the target optimization problem using spins. In order to get a good performance, number of possible configurations in the problem domain and the spin domain must be equal, i.e., there must be a one-to-one mapping between the configurations of spins and the problem.
2) Formulate the cost function of the problem in terms of spins, i.e., derive the energy function of the system. Global minimum of the energy function should correspond to the global minimum of the cost function.
3) Derive the mean field theory equations using this energy function, i.e., derive equations for updating averages (expected values) of spins.
4) Minimize the complexity of update operations.
5) Select the energy function and the cooling schedule parameters.

We propose an efficient encoding scheme which asymptotically reduces the number of variables used in the representation for mesh-connected architectures. Section 2 presents the proposed MFA formulation for the mapping problem for mesh-connected architectures using the proposed encoding. An efficient implementation scheme is also described in this section. The proposed formulation is asymptotically faster than the general formulation as discussed in Section 2. Section 3 presents the experimental performance evaluation of mesh-topology specific MFA algorithm proposed for the mapping problem in comparison with the well-known mapping heuristics KL, SA and the general MFA formulation.

2 MFA formulation for Mesh-Connected Architectures

Consider a P by Q two-dimensional mesh-connected architecture with P rows and Q columns. The encoding in the general MFA formulation in [1] necessitates $N \times K = N \times P \times Q$ variables for the problem representation. In this section, we propose a MFA formulation for mesh-connected architectures which exploits the conventional routing scheme in mesh interconnection topologies to introduce a much more efficient encoding scheme. Each processor in a $2D$-mesh can be identified with a two tuple (p, q) where $1 \leq p \leq P$ and $1 \leq q \leq Q$ denote its row and column indices, respectively. The communication distance between any two processors is equal to the Manhattan distance between those two processors on the processor grid. Hence, the unit communication cost between any two processors can be expressed as the sum of two components: horizontal and vertical communication costs. Horizontal and vertical unit communication costs are equal to the column and row distances between the processor pairs, respectively. Thus, any edge $(i, j) \in I$ with weight e_{ij} of the TIG will contribute

$$C_{ij} = C_{ij}^h + C_{ij}^v = e_{ij} \times |col(i) - col(j)| + e_{ij} \times |row(i) - row(j)| \qquad (1)$$

to the total communication cost, where $row(i)$ and $col(i)$ denote the row and column indices of the processor that task i is mapped to and $|\cdot|$ denotes the absolute value function. That is, $M(i) = (row(i), col(i))$. Here, C_{ij}^v and C_{ij}^h denote the horizontal and vertical communication costs due to edge $(i, j) \in I$.

2.1 Encoding

The MFA is derived by analogy to *Ising* and *Potts* models which are used to estimate the state of system of particles (spins). In Ising model, spins can be in one of the two states, whereas in Potts model they can be in one of the K states. In the proposed encoding, we use two Potts spins of dimensions P and Q to encode the row and column mappings, respectively, of each vertex (task) of the TIG. Spins with dimensions P and Q are called row and column spins which are labeled as $\mathbf{S}_i^r = [s_{i1}^r, \ldots, s_{ip}^r, \ldots, s_{iP}^r]^t$ and $\mathbf{S}_i^c = [s_{i1}^c, \ldots, s_{iq}^c, \ldots, s_{iQ}^c]^t$, respectively, for $i = 1, 2, \ldots, N$. Each spin vector is allowed to be equal to one of the principal unit vectors $\mathbf{e}_1, \ldots, \mathbf{e}_k, \ldots, \mathbf{e}_K$, and cannot take any other value, where $K = P$ and $K = Q$ for row and column spin vectors, respectively. Principal unit vector $\mathbf{e}_k$ is defined to be a vector which has all its component equal to 0 except its kth components which is equal to 1. Spins $\mathbf{S}_i^r$ and $\mathbf{S}_i^c$ are said to be in states p and q if $\mathbf{S}_i^r = \mathbf{e}_p$ and $\mathbf{S}_i^c = \mathbf{e}_q$, respectively, which means that $M(i) = (p, q)$. This encoding is much more efficient since it uses a total of $N \times (P + Q)$ two state variables instead of $N \times P \times Q$ two state variables of the general encoding [1].

2.2 Energy Function Formulation

The following spin average vectors are defined for energy function formulation.

$$\mathbf{V}_i^r = \left[v_{i1}^r, \ldots, v_{ip}^r, \ldots, v_{iP}^r\right]^t = \langle \mathbf{S}_i^r \rangle = \left[\langle s_{i1}^r \rangle, \ldots, \langle s_{ip}^r \rangle, \ldots, \langle s_{iP}^r \rangle\right]^t$$

$$\mathbf{V}_i^c = \left[v_{i1}^c, \ldots, v_{iq}^c, \ldots, v_{iQ}^c\right]^t = \langle \mathbf{S}_i^c \rangle = \left[\langle s_{i1}^c \rangle, \ldots, \langle s_{iq}^c \rangle, \ldots, \langle s_{iQ}^c \rangle\right]^t$$

Note that $s_{ip}^r, s_{iq}^c \in \{0, 1\}$ are discrete variables taking only two values 0 and 1, whereas $v_{ip}^r, v_{iq}^c \in [0, 1]$ are continuous variables taking any real value between 0 and 1. We have the following constraints for Potts spins;

$$\sum_{p=1}^{P} v_{ip}^r = 1 \quad \text{and} \qquad \sum_{q=1}^{Q} v_{iq}^c = 1$$

These constraints guarantee that each Potts spin $\mathbf{S}_i^r$ ($\mathbf{S}_i^c$) is in one of the P (Q) states at a time, and each task is assigned to only one row (column) for the proposed encoding. In order to construct an energy function it is helpful to associate the following meanings to the v_{ip}^r and v_{iq}^c values,

$$v_{ip}^r = \mathcal{P}(row(i) = p) \qquad \text{and} \qquad v_{iq}^c = \mathcal{P}(col(i) = q)$$

That is, v_{ip}^r (v_{iq}^c) denotes the probability of finding row (column) spin i in row p (column q). Formulation of horizontal and vertical communication costs due to edge (i, j) of the TIG as energy terms are:

$$E_{(i,j)}^h = e_{ij} \sum_{k=1}^{Q-1} \sum_{l=k+1}^{Q} (l - k) \left[\mathcal{P}\Big(col(i) = k \wedge col(j) = l\Big) + \mathcal{P}\Big(col(j) = k \wedge col(i) = l\Big)\right]$$

$$= e_{ij} \sum_{k=1}^{Q-1} \sum_{l=k+1}^{Q} (l - k)(v_{ik}^c v_{jl}^c + v_{jk}^c v_{il}^c) \tag{2}$$

$$E_{(i,j)}^v = e_{ij} \sum_{k=1}^{P-1} \sum_{l=k+1}^{P} (l - k)(v_{ik}^r v_{jl}^r + v_{jk}^r v_{il}^r) \tag{3}$$

The derivation of the mean field theory equations using the formulation of the energy terms $E^h_{(i,j)}$ and $E^v_{(i,j)}$ given in Eqs. (2) and (3) results in substantially complex expressions. Hence, we simplify the expressions for $E^h_{(i,j)}$ and $E^v_{(i,j)}$ in order to get more suitable expressions for the mean field theory equations. A close examination of Eqs. (2) and (3) reveals the symmetry between the expressions for $E^h_{(i,j)}$ and $E^v_{(i,j)}$ terms. Hence, algebraic simplifications will only be discussed for the $E^h_{(i,j)}$ term. Similar step can be followed for the $E^v_{(i,j)}$ term. We introduce the following notation for the simplification of the communication cost terms:

$$F^c_{i,k} = \sum_{l=1}^{k} v^c_{il} \qquad L^c_{i,k} = \sum_{l=k}^{Q} v^c_{il}, \qquad F^r_{i,k} = \sum_{l=1}^{k} v^r_{il} \qquad L^r_{i,k} = \sum_{l=k}^{P} v^r_{il} \tag{4}$$

Here, F^c_{ik} (F^r_{ik}) and L^c_{ik} (L^r_{ik}) denote the probabilities that task i is mapped to one of the processor in the first k columns (k rows) and the last $Q-k+1$ columns (P–k+1 rows), respectively. Using this notation and thru some algebraic manipulations the expression for $E^h_{(i,j)}$ and $E^v_{(i,j)}$ simplifies as:

$$\begin{aligned} E^h_{(i,j)} &= e_{ij}\{\sum_{k=1}^{Q-1}\sum_{l=k+1}^{Q}(l-k)v^c_{ik}v^c_{jl} + \sum_{k=1}^{Q-1}\sum_{l=k+1}^{Q}(l-k)v^c_{jk}v^c_{il}\} \\ &= e_{ij}\{\sum_{k=1}^{Q-1}\sum_{l=1}^{k}v^c_{il}\sum_{m=k+1}^{Q}v^c_{jm} + \sum_{k=1}^{Q-1}\sum_{l=1}^{k}v^c_{jl}\sum_{m=k+1}^{Q}v^c_{im}\} \\ &= e_{ij}\sum_{k=1}^{Q-1}(F^c_{i,k}L^c_{j,k+1} + F^c_{j,k}L^c_{i,k+1}) \end{aligned} \tag{5}$$

$$E^v_{(i,j)} = e_{ij}\sum_{k=1}^{P-1}(F^r_{i,k}L^r_{j,k+1} + F^r_{j,k}L^r_{i,k+1}) \tag{6}$$

We formulate the energy term corresponding to the imbalance cost using the same inner product approach adopted in the general formulation as follows:

$$\begin{aligned} E^B &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} w_i w_j \mathcal{P}\Big(M(i) = M(j)\Big) \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} w_i w_j \sum_{p=1}^{P}\sum_{q=1}^{Q}\mathcal{P}\Big(M(i) = (p,q)\Big) \times \mathcal{P}\Big(M(j) = (p,q)\Big) \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} w_i w_j \sum_{p=1}^{P}\sum_{q=1}^{Q} v^r_{ip}v^c_{iq}v^r_{jp}v^c_{jq} \end{aligned} \tag{7}$$

Total energy term can be defined in terms of the communication cost terms and the imbalance cost term as

$$E(\mathbf{V}^r, \mathbf{V}^c) = E^h(\mathbf{V}^c) + E^v(\mathbf{V}^r) + \beta E^B(\mathbf{V}^r, \mathbf{V}^c) \tag{8}$$

Here, $\mathbf{V}^r = [\mathbf{V}^r_1, \ldots, \mathbf{V}^r_p, \ldots, \mathbf{V}^r_P]^t$ and $\mathbf{V}^c = [\mathbf{V}^c_1, \ldots, \mathbf{V}^c_q, \ldots, \mathbf{V}^c_Q]^t$ denote the row and column spin-average matrices consisting of N P and Q dimensional vectors as their rows, respectively.

2.3 Derivation of the Mean Field Theory Equation

The expected values $\mathbf{V}_i^r$ and $\mathbf{V}_i^c$ of each row and column spins $\mathbf{S}_i^r$ and $\mathbf{S}_i^c$ are iteratively updated using the Boltzmann distribution as

$$(a) \quad v_{ip}^r = \frac{e^{\phi_{ip}/T^r}}{\sum_{k=1}^{P} e^{\phi_{ik}/T^r}}, \qquad (b) \quad v_{iq}^c = \frac{e^{\phi_{iq}/T^c}}{\sum_{k=1}^{Q} e^{\phi_{ik}/T^c}}, \tag{9}$$

for $p = 1, 2, \ldots, P$ and $q = 1, 2, \ldots, Q$, respectively. Here, T^r and T^c denote the temperature parameters used for annealing. Recall that, the number of states of the row and column spins are different (P and Q for row and column spins, respectively) in the proposed encoding. As the convergence time and the temperature parameter of the system depends on the number of states of the spins we interpret the row and column spins as different system. Note that Eqs. (9.a) and (9.b) enforce each row and column Potts spins $\mathbf{S}_i^r$ and $\mathbf{S}_i^c$ to be in one of the P and Q states, respectively, when they converge. In the proposed MFA formulation, row and column spins are updated in an alternative manner, i.e., each row spin update follows a column spin update and vice versa.

In the proposed formulation, row and column mean field vectors Φ_i^r and Φ_i^c are to be computed in row and column iterations, respectively. Each element ϕ_{ip}^r and ϕ_{iq}^c of the row and column mean field vectors $\Phi_i^r = [\phi_{i1}^r, \ldots, \phi_{ip}^r, \ldots, \phi_{iP}^r]^t$ and $\Phi_i^c = [\phi_{i1}^c, \ldots, \phi_{iq}^c, \ldots, \phi_{iQ}^c]^t$ experienced by row and column Potts spins i denote the decrease in the energy function by assigning $\mathbf{S}_i^r$ to $\mathbf{e}_p$ and $\mathbf{S}_i^c$ to $\mathbf{e}_q$, respectively. Hence, $-\phi_{ip}^r$ ($-\phi_{iq}^c$) may be interpreted as the decrease in the overall solution quality by mapping task i to row p (column q). Then, in Eq. (9.a) (Eq. (9.b)), v_{ip}^r (v_{iq}^c) is updated such that the probability of mapping task i to row p (column q) increases with increasing mean field value ϕ_{ip}^r (ϕ_{iq}^c). Using the simplified expressions for the proposed energy functions in Eqs. (5), (6) and (7)

$$\begin{aligned}
\phi_{ip}^r &= -\frac{\partial H(\mathbf{V}^r, \mathbf{V}^c)}{\partial v_{ip}^r} = \phi_{ip}^{r(C)} + \beta^r \phi_{ip}^{r(B)} \\
&= -\sum_{j \in Adj(i)} e_{ij} Z_{jp}^r - \beta^r w_i \sum_{j=1, j \neq i}^{N} w_j v_{jp}^r \sum_{q=1}^{Q} v_{iq}^c v_{jq}^c \quad \text{where} \quad Z_{jp}^r = \sum_{k=1}^{p-1} F_{jk}^r + \sum_{k=p+1}^{P} L_{jk}^r
\end{aligned} \tag{10}$$

$$\begin{aligned}
\phi_{iq}^c &= -\frac{\partial H(\mathbf{V}^r, \mathbf{V}^c)}{\partial v_{iq}^c} = \phi_{iq}^{c(C)} + \beta^c \phi_{iq}^{c(B)} \\
&= -\sum_{j \in Adj(i)} e_{ij} Z_{jq}^c - \beta^c w_i \sum_{j=1, j \neq i}^{N} w_j v_{jq}^c \sum_{p=1}^{P} v_{ip}^r v_{jp}^r \quad \text{where} \quad Z_{jq}^c = \sum_{k=1}^{q-1} F_{jk}^c + \sum_{k=q+1}^{Q} L_{jk}^c
\end{aligned} \tag{11}$$

As seen in Eqs. (10) and (11), different balance parameters β^r and β^c are used in the mean field computations of row and column iterations since row and column spins are interpreted as different systems.

2.4 An Efficient Implementation Scheme

As mentioned earlier, the proposed MFA algorithm is an iterative process. The complexity of a single MFA iteration is due mainly to the mean field computations. As is seen in Eqs. (10) and (11), calculation of mean field values is computationally very intensive. In this section, we propose an efficient implementation

scheme which reduces the complexity of mean field computations, and hence the complexity of the MFA iteration, by asymptotical factors. Mean field theory equations given in Section 2.3 reveals the symmetry between the mean field vector computations in row and column iterations. Hence, the proposed implementation scheme will only be discussed for computing the mean field vector $\Phi_i^r = [\phi_{i1}^r, \ldots, \phi_{ip}^r, \ldots, \phi_{iP}^r]^t$ in row iterations. Similar discussion applies to the computation of the $\Phi_i^c = [\phi_{i1}^c, \ldots, \phi_{iq}^c, \ldots, \phi_{iQ}^c]^t$ vector in column iterations.

Assume that row Potts spin i is selected at random in a row iteration for updating its expected value vector $\mathbf{V}_i^r$. We will first discuss the mean field computations corresponding to the vertical communication cost. As is seen in Eq. (10), these computations require the construction of the $\mathbf{Z}_j^r = [Z_{j1}^r, \ldots, Z_{jp}^r, \ldots, Z_{jP}^r]^t$ vector for each vertex j adjacent to i in TIG. The computation of an individual $\mathbf{Z}_j^r$ vector necessitates the construction of $\mathbf{F}_j^r = [F_{j1}^r, \ldots, F_{jp}^r, \ldots, F_{jP}^r]^t$ and $\mathbf{L}_j^r = [L_{j1}^r, \ldots, L_{jp}^r, \ldots, L_{jP}^r]^t$ vectors. These two vectors can be constructed in $\Theta(P)$ time using the recursive equation

$$F_{jk}^r = F_{j,k-1}^r + v_{jk}^r, \quad \text{for} \quad k = 2, 3, \ldots, P \quad \text{where} \quad F_{j1}^r = v_{j1}^r \tag{12}$$

$$L_{jk}^r = L_{j,k-1}^r + v_{jk}^r, \quad \text{for} \quad k = P-1, P-2, \ldots, 1 \quad \text{where} \quad L_{jP}^r = v_{jP}^r \tag{13}$$

The computation of an individual Z_{jp}^r value takes $\Theta(P)$ time. Hence, the complexity of computing an individual $\mathbf{Z}_j^r$ vector becomes $\Theta(P^2)$. However, in the proposed scheme the elements of the $\mathbf{Z}_j^r$ vector are computed in only $\Theta(P)$ time by exploiting the recursive equation

$$Z_{jq}^r = Z_{j,q-1}^r - L_{jq}^r + F_{j,q-1}^r \quad \text{for} \quad k = 1, 2, \ldots, P \quad \text{where} \quad Z_{j1}^r = \sum_{l=2}^{P} L_{jl} \tag{14}$$

Hence, the complexity of mean field computations corresponding to the vertical communication cost term is $\Theta(d_i P)$ in a row iteration since the first summation term in Eq. (10) requires the computation and weighted addition of d_i such $\mathbf{Z}_j^r$ vectors. Here, d_i denotes the degree of vertex i in the TIG. Similarly, the complexity of mean field computations corresponding to the horizontal communication cost term is $\Theta(d_i Q)$ when column spin i is selected at random in a column iteration.

As is seen in Eq. (10), the complexity of computing an individual mean field value corresponding to the imbalance cost term is $\Theta(NQ)$. Since P such values are computed in a row iteration, the total complexity of mean field computations corresponding to the imbalance cost term becomes $\Theta(NPQ)$. However, the complexity of these computations can be asymptotically reduced as follows. The second summation term in Eq. (10) can be re-written by interchanging the order of summations as

$$\begin{aligned} w_i \sum_{j=1, j\neq i}^{N} w_j v_{jp}^r \sum_{q=1}^{Q} v_{iq}^c v_{jq}^c &= w_i \sum_{q=1}^{Q} v_{iq}^c \sum_{j=1, j\neq i}^{N} w_j v_{jp}^r v_{jq}^c \\ &= w_i \sum_{q=1}^{Q} v_{iq}^c (W_{pq} - w_i v_{ip}^r v_{iq}^c) \end{aligned} \tag{15}$$

$$\text{where} \qquad W_{pq} = \sum_{j=1}^{N} w_j v_{jp}^r v_{jq}^c \tag{16}$$

Here, W_{pq} denotes the total computational load of processor (p,q) for the current row and column spin values. In Eq. (15), $W_{pq} - w_i v_{ip}^r v_{iq}^c$ denotes the weight of processor (p,q) excluding task i. Hence, Eq. (15) represents the increase in the imbalance cost term if task i is assigned to row p (i.e., $\mathbf{V}_i^r$ is set to $\mathbf{e}_p$). In the proposed implementation scheme, we maintain a P by Q processor weight matrix $\mathbf{W}$ consisting of W_{pq} values. The entries of this matrix are computed using Eq. (16) only at the beginning of the algorithm. Then, while updating the expected value vector $\mathbf{V}_i^r$ of an individual Potts spin i, the $\mathbf{W}$ matrix is updated in $\Theta(PQ)$ time using

$$W_{pq}^{(new)} = W_{pq}^{(old)} + w_i v_{iq}^c (v_{ip}^{r(new)} - v_{ip}^{r(old)})$$

for $p = 1, 2, \ldots, P$ and $q = 1, 2, \ldots, Q$. Hence, computing Eq. (15) for each ϕ_{ip}^r value takes $\Theta(Q)$ time. Since P such values are to be computed to construct the mean field vector, the total complexity of mean field computations corresponding to the imbalance cost term reduces to $\Theta(PQ)$ in a row iteration.

It should be noted here that, column iterations also use and update the same weight matrix $\mathbf{W}$ as is used and maintained in row iterations. The complexity of mean field computations corresponding to the imbalance cost term is also $\Theta(QP)$ in column iterations. Thus, the proposed scheme reduces the overall complexity of mean field computations to $\Theta(d_{avg}P + PQ)$ and $\Theta(d_{avg}Q + PQ)$ in row and column iterations, respectively. Here, d_{avg} denotes the average vertex degree in TIG . After computing the mean field vectors Φ_i^r and Φ_j^c, expected value vectors $\mathbf{V}_i^r$ and $\mathbf{V}_j^c$ of row and column Potts spins i and j can be updated using Eq. (9.a) and Eq. (9.b) in $\Theta(P)$ and $\Theta(Q)$ times, in a row and column iteration, respectively.

Therefore, the proposed implementation scheme reduces the complexity of an individual row and column iteration to $\Theta(d_{avg}P + PQ)$ and $\Theta(d_{avg}Q + PQ)$, respectively. The proposed MFA scheme asymptotically reduces the complexity of a single MFA iteration from $\Theta(d_{avg}PQ + (PQ)^2)$ of the general MFA formulation to $\Theta(d_{avg}(P+Q)+PQ)$ for a P by Q mesh. For a square mesh with K processors, this corresponds to an asymptotical complexity reduction from $\Theta(d_{avg}K + K^2)$ to $\Theta(d_{avg}\sqrt{K} + K)$.

3 Performance Evaluation

This section presents the performance evaluation of the efficient MFA formulation proposed for the mapping problem for mesh-connected architectures in comparison with the well known mapping heuristics: Simulated Annealing (SA), Kernighan-Lin (KL) and the general MFA formulation. The following paragraphs briefly present the implementation details of these algorithms.

The MFA algorithm proposed for the mapping problem for mesh topology is implemented efficiently as described in Section 2.4. At the beginning of the algorithm row and column spin averages are initialized to $1/P$ and $1/Q$ plus a random disturbance term, so that the initial spin averages are uniformly distributed in the range $0.9/P \leq v_{ip}^{r(initial)} \leq 1.1/P$ and $0.9/Q \leq v_{iq}^{c(initial)} \leq 1.1/Q$

for $i = 1,2,\ldots,N$, $p = 1,2,\ldots,P$ and $q = 1,2,\ldots,Q$ respectively. Note that $\lim_{T^r \to \infty} v_{ip}^r = 1/P$ and $\lim_{T^c \to \infty} v_{iq}^c = 1/Q$. The initial temperatures and balance parameters used in the mean field computation of row and column iterations are estimated using these initial random spin average values. As is seen in Eq. (10) (Eq. (11)), the parameter β^r (β^c) determine a balance between the terms $\phi_{ip}^{r(C)}$ and $\phi_{ip}^{r(B)}$ ($\phi_{iq}^{c(C)}$ and $\phi_{iq}^{c(B)}$) in the mean field computations of row (column) iterations. We compute these averages as $\langle \phi_{ip}^{r(C)} \rangle = (\sum_{i=1}^{N} \sum_{p=1}^{P} \phi_{ip}^{r(C)})/NP$ and $\langle \phi_{ip}^{r(B)} \rangle = (\sum_{i=1}^{N} \sum_{p=1}^{P} \phi_{ip}^{r(B)})/NP$, respectively, using the initial v_{ip}^r values. Averages $\langle \phi_{iq}^{c(C)} \rangle$ and $\langle \phi_{iq}^{c(B)} \rangle$ are computed similarly using the initial v_{iq}^c values. Then, balance parameters are computed as $\beta^r = C_B \langle \phi_{ip}^{r(C)} \rangle / \langle \phi_{ip}^{r(B)} \rangle$ and $\beta^c = C_B \langle \phi_{jq}^{c(C)} \rangle / \langle \phi_{jq}^{c(B)} \rangle$, where constant C_B is chosen as 20. Our experiments show that computing β^r and β^c using this method is sufficient for obtaining balanced partitions.

Selection of initial temperature parameters T_0^r and T_0^c is crucial for obtaining good quality solutions. In previous applications of MFA [8, 11], it is experimentally observed that spin averages tend to converge at a critical temperature. We prefer an experimental way for computing T_0^r and T_0^c which is easy to implement and successful as the results of experiments indicate. After the balance parameters β^r and β^c are fixed, average row and column mean fields are computed as $\langle \phi_{ip}^r \rangle = \langle \phi_{ip}^{r(C)} \rangle + \beta^r \langle \phi_{ip}^{r(B)} \rangle$ and $\langle \phi_{iq}^c \rangle = \langle \phi_{iq}^{c(C)} \rangle + \beta^c \langle \phi_{iq}^{c(B)} \rangle$. Then, T_0^r and T_0^c are computed using $T_0^r = C_T \langle \phi_{ip}^r \rangle / P$ and $T_0^c = C_T \langle \phi_{iq}^c \rangle / Q$ where constant C_T is chosen as 20.

The same cooling schedule is adopted for row and column iterations as follows. At each temperature, row and column iterations proceed in an alternative manner for randomly selected unconverged row and column spin updates until $\Delta E^r < \epsilon$ and $\Delta E^c < \epsilon$ for M consecutive iterations, respectively, where $M = N$ initially and $\epsilon = 0.05$. Average spin values are tested for convergence after each update. If one of the v_{ik} terms of a row or column spin average vector is detected to be greater than 0.95, that spin is assumed to converge to state k. The cooling process is realized in two phases, slow cooling followed by fast cooling, similar to the cooling schedules used for SA [11]. In the slow cooling phase, row and column temperatures are decreased using $\alpha = 0.9$ until $T < T_0/1.5$ for both row and column iterations. Then, in the fast cooling phase, M is set to $M/4$, α is set to 0.7 and cooling for row and column iterations are continued until 90% of the row and column spins converge, respectively. At the end of this cooling process, the maximum element in each unconverged spin average vector is set to 1 and all other elements in that vector are set to 0. Then, the result is decoded as described in Section 2.1, and the resulting mapping is found.

The general MFA formulation is implemented efficiently as described in [1]. The initialization of spin averages, the selection of the balance parameter β and the initial temperature T_0 are performed as is described for the mesh-specific MFA implementation. The parameters C_T and C_B are chosen as 0.5. The same cooling schedule described for mesh-specific MFA implementation is used in the implementation of the general MFA formulation.

Two-phase approach is used to apply KL to the mapping problem. KL heuristic is implemented efficiently as described by Fiduccia and Mattheyses (FM) [5] for the clustering phase. The recursive bisection scheme implemented for the first phase recursively partitions the initial TIG into two clusters until $K = P \times Q$ clusters are obtained. In the KLFM heuristic, computational load balance among clusters is maintained implicitly by the algorithm. Vertex moves causing intolerable load imbalance are not considered. The one-to-one mapping heuristic used in the second phase is a variant of the KL heuristics. In this heuristic, communication cost is minimized by performing a sequence of cluster swaps between the processor pairs after an initial random mapping of K clusters.

The SA algorithm implemented in this work implicitly achieves the load balance among processors by setting a neighborhood configuration consisting of all configurations which result from moving one task from the processor with maximum load to any other processor. Randomly selected possible moves which decrease the communication costs are realized. Acceptance probabilities of randomly selected moves increasing the communication cost are controlled with a temperature parameter T which is decreased using an automatic annealing schedule [9]. Hence, as the annealing proceeds acceptance probabilities of uphill moves decrease.

3.1 Experimental Results

The mapping heuristics are experimented by mapping some test TIG's onto various size meshes. Test TIG's correspond to the undirected sparse graphs associated with the symmetric sparse matrices selected from *Harwel Boeing sparse matrix test collection* [4]. Weights of the vertices are assumed to be equal to their degrees. These test TIG's are mapped to 8×8, 8×16 and 16×16 $2D$-meshes.

Table 1 illustrates the performance result of the KL, SA, general and mesh-specific MFA heuristics for the generated mapping problem instances. In this table, "Gen" and "Mesh" denote the general and mesh-specific MFA formulations, respectively. Each algorithm is executed 5 times for each problem instance starting from different, randomly chosen initial configurations. Averages are illustrated in Table 1. Total communication cost averages of the solutions are normalized with respect to the results of the mesh specific MFA heuristic developed in this work. Percent computational load imbalance averages of solutions displayed in this table are computed using $100 \times (W_{max} - W_{avg})/W_{avg}$. Here, W_{max} denotes the maximum processor load and W_{avg} denotes the computational loads of processors under perfect load balance conditions. Execution time averages are measured on a SUN SPARC 10 workstation. Execution time averages are normalized with respect to those of mesh-specific MFA heuristic. Table 2 is constructed for a better illustration of the overall relative performances of the heuristics. Percent load imbalance averages of the solutions are also normalized with respect to the results of the mesh-specific MFA heuristic. Then, the overall averages of the normalized averages of Table 1 are displayed in Table 2. Tables 1 and 2 confirm the expectation that mesh-specific MFA formulation is significantly faster (7.26 times on the average) than the general MFA formulation

while producing solutions with considerably better qualities. As seen in these tables, the mesh specific MFA heuristic produces significantly better solutions than the KL heuristic whereas the MFA heuristic is slightly slower.

Table 1. Total communication cost and execution time averages normalized with respect to mesh-specific MFA, and percent computational load imbalance averages of the solutions found by SA, KL, general MFA and mesh-specific MFA for various mapping problem instances. Here, N and M denote the number of nodes and edges in the TIGs, respectively.

Problem		Communication Cost				% Load Imbalance				Execution Time			
	Mesh			MFA				MFA				MFA	
TIG	$P \times Q$	KL	SA	Gen.	Mesh	KL	SA	Gen.	Mesh	KL	SA	Gen.	Mesh
DWT758	8x8	1.79	0.95	2.02	1.00	9.4	5.3	5.7	4.2	0.2	16.0	3.4	1.0
N=758	8x16	2.85	1.10	2.75	1.00	12.3	14.4	9.3	7.7	0.3	5.7	1.6	1.0
M=1332	16x16	3.34	1.38	4.03	1.00	16.5	26.3	15.0	9.0	1.7	5.4	2.7	1.0
DWT1242	8x8	1.42	1.00	2.01	1.00	8.9	3.9	6.3	2.6	0.2	25.4	7.8	1.0
N=1242	8x16	2.53	1.05	2.62	1.00	12.1	5.3	8.2	5.3	0.2	8.7	2.7	1.0
M=4592	16x16	2.91	1.08	2.94	1.00	16.3	11.5	10.3	9.4	0.8	7.0	3.8	1.0
JAGMESH7	8x8	1.40	0.95	1.89	1.00	8.1	2.9	6.4	2.8	0.2	26.6	19.4	1.0
N=1138	8x16	2.74	1.06	3.25	1.00	11.1	7.7	5.9	4.4	0.3	11.0	4.2	1.0
M=3156	16x16	3.48	1.20	3.77	1.00	13.8	18.8	12.6	9.1	1.2	9.6	6.8	1.0
BSCPWR09	8x8	1.87	0.90	2.43	1.00	10.6	1.4	8.0	4.1	0.2	59.6	8.2	1.0
N=1723	8x16	2.33	1.01	3.13	1.00	12.4	2.7	10.8	4.0	0.2	23.5	6.6	1.0
M=2394	16x16	4.75	1.80	5.06	1.00	17.3	5.6	18.9	4.0	1.3	32.1	14.9	1.0
LSHP2233	8x8	1.37	0.81	1.88	1.00	8.1	1.3	5.4	2.3	0.2	34.5	17.3	1.0
N=2233	8x16	2.20	0.97	3.63	1.00	9.5	3.0	3.9	2.4	0.3	17.5	7.2	1.0
M=6552	16x16	3.31	1.12	2.68	1.00	10.2	7.9	12.0	3.7	0.7	14.0	2.2	1.0

Table 2. Average performance measures of the solutions found by SA, KL, general MFA and mesh-specific MFA for mapping problem instances in Table 1.

	KL	SA	MFA	
			Gen.	Mesh.
Communication Cost	2.55	1.08	2.94	1.00
Load Imbalance	2.34	1.5	1.85	1.00
Execution Time	0.5	19.7	7.26	1.00

The qualities of the solutions obtained by the mesh-specific MFA heuristic are comparable with those of the SA heuristic. However, the mesh-specific MFA heuristic is faster (19.7 times on the average). Hence, the proposed MFA heuristic approaches the speed performance of the fast KL heuristic while approaching the solution quality of the powerful SA heuristic.

4 Conclusion

In this paper, we have proposed an efficient mapping heuristic for mesh-connected parallel architectures based on Mean Field Annealing (MFA). We have also developed an efficient implementation scheme for the proposed mapping formulation. The performance of the proposed mapping heuristic is evaluated in comparison with the well-known heuristics Kernighan-Lin (KL), Simulated Annealing (SA) and general MFA formulation for a number of mapping problem instances generated using *Harwell-Boeing sparse matrix test problems*. The proposed mesh-specific MFA formulation is found to be significantly faster than the general MFA formulation as is expected. The proposed MFA heuristic is slightly slower than the fast KL heuristic. However, it always produces significantly better solutions than the KL heuristic. The qualities of the solutions obtained by the proposed MFA heuristic are comparable to those of the powerful SA heuristic. However, the proposed MFA heuristic is significantly faster than the SA heuristic.

References

1. Bultan, T., and Aykanat, C., "A new mapping heuristic based on mean field annealing," *Journal of Parallel and Distributed Computing*, vol. 10, pp. 292-305, 1992.
2. Bokhari, S.H., "On the mapping problem," *IEEE Trans. Comput.,* vol. 30, no. 3, pp. 207-214, 1981.
3. Camp, W.J, Plimpton, S.J, Hendrickson, B.A., and Leland, R.W., "Massively parallel methods for Enineering and Science problem," *Communication of ACM*, vol. 37, no. 4, April 1994.
4. Duff, I.S., and Grius, R.G., "Sparse matrix test problems," *ACM Trans. on Matematical software*, vol. 17, no. 1, pp. 1-14, March 1989.
5. Fiduccia, C. M., and Mattheyses, R. M., "A linear heuristic for improving network partitions," *Proc. Design Automat. Conf.,* pp. 175-181, 1982.
6. Hopfield, J. J., and Tank, D. W., "Neural Computation of Decisions in Optimization Problems," *Biolog. Cybern.,* vol. 52, pp. 141-152, 1985.
7. Kernighan, B. W., and Lin, S., "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.,* vol. 49, pp. 291-307, 1970.
8. Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., "Optimization by simulated annealing," *Science*, vol. 220, pp. 671-680, 1983.
9. Johnson, A., Aragon, G.R., McGeoch, L.A., and Scheoven, C., "Optimization by simulated annealing," *Operations Research*, vol. 37, no. 6, Nov 1989.
10. Peterson C., and Anderson, J.R., "Neural networks and NP-complete optimization problems; a performance study on the graph bisection problem," *Complex Syst.* vol. 2, pp. 59-89, 1988.
11. Peterson, C., and Soderberg B., "A new method for mapping optimization problems onto neural networks" *Int. J. Neural Syst.*, vol. 3, no. 1, pp. 3-22, 1989.
12. Van den Bout, D. E., and Miller, T. K. "Improving the performance of the Hopfield-Tank neural network through normalization and annealing," *Biolog. Cybern.*, vol. 62, pp. 129-139, 1989.

This article was processed using the LaTeX macro package with LLNCS style