An Efficient Mean Field Annealing Formulation for Mapping Unstructured Domains to Hypercubes [⋆]

Cevdet Aykanat and İsmail Haritaoğlu

Computer Engineering Department, Bilkent University, Ankara, Turkey
aykanat@cs.bilkent.edu.tr

Abstract. We propose an efficient MFA formulation for mapping unstructured domains to hypercube-connected distributed-memory architectures. In the general MFA formulation, $N \times P$ spin variables are maintained and an individual MFA iteration requires $\Theta(d_{avg}P + P^2)$ time for the mapping of a sparse domain graph with N vertices and average vertex degree of d_{avg} to a parallel architecture with P processors. The proposed hypercube-specific MFA formulation asymptotically reduces the number of spin variables and the computational complexity of an individual MFA iteration to $N lg_2 P$ and $\Theta(d_{avg} lg_2 P + P lg_2 P)$, respectively, by exploiting the topological properties of hypercubes.

1 Introduction

Parallelization schemes for many applications on distributed memory architectures are characterized by the *mapping* of the problem domain to processors with locality of communication. These schemes employ *data parallelism* by breaking the data structures supporting a computation into pieces and then assigning those pieces to different processors. These decomposition and assignment tasks constitute the *domain mapping* problem [2]. The objective in the domain mapping is to find a mapping which minimizes the communication overhead while maintaining almost the same workload for each processor. Problem domain representing the data structure is represented with an undirected graph $G = (T, I)$, referred here as *domain graph*. Vertices $i, j \in T$ represent atomic computations (tasks) which can be executed simultaneously and independently. Vertex weight w_i denotes the estimated computational cost of task i. Each edge $(i, j) \in I$ denotes the need for the bidirectional interaction between tasks i and j. Edge weight e_{ij} denotes the volume of interaction between tasks i and j connected by edge $(i, j) \in I$. This graph usually represents the repeated execution of the computations corresponding to the vertices with intervening partial result exchanges denoted by the edges. An edge incurs interprocessor communication only if the respective pair of computations are mapped to two different processors. Simultaneous *single-hop* communications between distinct adjacent pairs of processors can be performed concurrently. However, simultaneous *multi-hop* communications between distant pairs of processors may introduce congestion to the interconnection network, thus increasing the communication overhead. Multi-hop

[⋆] This work is partially supported by the Commission of the European Communities,Directorate General for Industry under contract ITDC 204-82166 communications between distant processors are usually routed over the shortest paths of links between the communicating pairs of processors. Hence, multi-hop messages are usually weighted with the distances between the respective pairs of processors in the network, while considering their contribution to the overall communication cost. Here, distance refers to the number of communication links and switching elements along the communication route in *static* and *dynamic* interconnection networks, respectively. Thus, in our communication cost model, an edge $(i,j) \in I$ contributes $e_{ij} d_{M(i),M(j)}$ to the overall communication cost where $M(i) = p$ and $M(j) = q$ denote that processors that tasks i and j are mapped to, respectively, and d_{pq} denotes the distance between processors p and q. This model is widely used in the literature [2].

The domain-mapping problem is known to be NP-hard for unstructured domains. Hence, heuristics giving suboptimal solutions are used to solve the problem. Two distinct approaches have been considered in the context of mapping heuristics: *one-phase* and *two-phase*. In one-phase approaches, referred as *many-to-one* mapping, vertices of the domain graph are directly mapped onto the processors. In two-phase approaches, *clustering* phase is followed by *one-to-one* mapping phase. In the clustering phase, vertices of the domain graph are partitioned into P equally weighted clusters while the total number of edges among clusters is minimized. Here, P denotes the number of processors. The problem solved in the clustering phase is identical to the P-way graph partitioning problem. In the one-to-one mapping phase, each cluster is assigned to an individual processor so that the total communication overhead is minimized. As expected, two-phase approaches suffer from delaying the processor distance factor to the second phase.

Mean field annealing (MFA) algorithm, proposed for solving combinatorial optimization problems, combines the characteristics of neural networks with the annealing notion of simulated annealing (SA). Previous works on MFA resulted with successful formulation of the algorithm to some classic combinatorial optimization problems including the mapping problem. In MFA, discrete variables, called spins are used for encoding the combinatorial optimization problems. An energy function written in terms of spins is used for representing the cost function of the problem. Then, using the expected values of these discrete variables, a gradient descent type relaxation scheme is used to find a configuration of the spins which minimizes the associated energy function. The MFA formulation proposed for the mapping problem [1] is a general formulation which works for any interconnection topology. In this formulation, $N \times P$ spin variables are maintained and an individual MFA iteration takes $\theta(d_{avg}P + P^2)$ time, for the mapping of a sparse domain graph with N vertices and average vertex degree of d_{avg} to a parallel architecture with P processors. In this work, we propose an efficient MFA formulation for mapping unstructured domains to hypercube-connected distributed-memory architectures. The proposed hypercube-specific MFA formulation asymptotically reduces the number of spin variables and the computational complexity of an individual MFA iteration to Nlg_2P and $\Theta(d_{avg}lg_2P + Plg_2P)$, respectively, by exploiting the topological properties of hypercubes.

2 MFA Formulation For Hypercubes

A D-dimensional hypercube H consists of $P = 2^D$ processors labeled from 0 to $2^D - 1$ such that there is a direct connection between two processors if and only if the binary representation of their labels differ exactly in one bit. Each processor p is represented by a D bit binary number $(p_{D-1}, \ldots, p_d, \ldots, p_0)$, where $p = \sum_{d=0}^{D-1} p_d 2^d$. A D-dimensional hypercube H can be split into two $(D-1)$-dimensional subcubes H_d^0 and H_d^1 so that each processor of H_d^0 is connected to exactly one processor of H_d^1. This splitting operation is called *tearing* along dimension (channel) d [5]. Note that there are D such tearings for $d = 0, 1, \ldots, D-1$. Here, H_d^0 and H_d^1 both contain $P/2$ processors whose dth bits are 0 and 1, respectively. The $P/2$ communication links connecting the processors of H_d^0 and H_d^1 in a one-to-one manner are referred here as channel d for $d = 0, 1, \ldots, D-1$.

In hypercubes, communication distance between a processor pair p and q is equal to the number of bits that differ between their labels. Consider a processor pair p and q with a distance of k. Let $\{d_1, d_2, \ldots, d_k\}$ denotes the set of k differing bit positions (dimensions) between processors p and q. Then, each differing bit position d_i corresponds to the use of a communication link along channel d_i in the communication route between processors p and q, for $i = 1, 2, \ldots, k$. Each differing bit position d_i also means that processors p and q are in different subcubes $H_{d_i}^1$ and $H_{d_i}^0$, or vice-versa, respectively. These properties of hypercube topology can be exploited to decompose the communication cost of each edge $(i,j) \in I$ into its channel components. Let the processor mapping $M(i)$ for a task $i \in T$ be represented as a D-bit binary label $M(i) = [m_{D-1}(i), \ldots, m_0(i)]^t$, where $m_d(i)$ represents the dth bit of processor $M(i)$. Here, $m_d(i) = 1$ and $m_d(i) = 0$ mean that task i is in the subcubes H_d^1 and H_d^0, respectively, for a tearing over channel d. Using this notation, communication volume contribution of edge $(i,j) \in I$ to channel d is $C_{ij}^d = e_{ij}|m_d(i) - m_d(j)|$ where $|\cdot|$ denotes the absolute value function. Edge (i,j) incurs a communication over channel d only if processors $M(i)$ and $M(j)$ are in different $(D-1)$-dimensional subcubes H_d^0 and H_d^1. Hence, the total communication cost C can be written as $C = \sum_{(i,j)\in I} C_{ij}$ where $C_{ij} = \sum_{d=0}^{D-1} C_{ij}^d$. Here, C_{ij} denotes the contribution of edge $(i,j) \in I$ to the total communication cost. This communication cost formulation is exploited in this work to propose an efficient encoding scheme for mapping to hypercubes.

2.1 Encoding

In the proposed encoding, we assign an Ising spin to each task for each channel. Effectively, we assign D Ising spins $\{s_{id}\}_{d=0}^{D-1}$ to each task $i = 1, 2, \ldots, N$. Here, Ising spin s_{id} corresponds to $m_d(i)$ mentioned earlier. That is, the spin configuration $s_{id} = 1$ $(s_{id} = 0)$ means that task i is assigned to one of the $P/2$ processors in the $(D-1)$-dimensional subcube H_d^1 (H_d^0) defined by the tearing over channel d. Note that $M(i) = [s_{i,D-1}, \ldots, s_{i0}]^t$ corresponds to a distinct mapping of task i for each possible configuration of the D spins $\{s_{id}\}_{d=0}^{D-1}$ assigned to task i. Hence, the proposed encoding constructs a one-to-one mapping between the configuration space of the problem domain and the spin domain. The proposed encoding requires $N \times D = N lg_2 P$ Ising spins where each Ising spin contains a single spin variable. Hence, the proposed encoding scheme asymptotically reduces the number of spin variables from $N \times P$ of the general MFA formulation to $N lg_2 P$.

2.2 Energy Function Formulation

The average (expected) value of each spin is defined as $v_{id} = \langle s_{id} \rangle$. Recall that $s_{id} \in \{0,1\}$ are two-state discrete variables, whereas $v_{id} \in [0,1]$ are continuous variables. In order to construct an energy function it is helpful to associate the following meanings to the v_{id} values:

$$v_{id} = \mathcal{P}(\text{task } i \text{ is mapped to } H_d^1) \qquad 1 - v_{id} = \mathcal{P}(\text{task } i \text{ is mapped to } H_d^0) \tag{1}$$

That is, v_{id} and $(1 - v_{id})$ denote the probabilities of finding task i in a processor in H_d^1 and H_d^0, respectively. Thus, the formulation of communication cost due to edge $(i,j) \in I$ as an energy term is:

$$E_{ij}^C = \sum_{d=0}^{D-1} E_{ij}^d = e_{ij} \sum_{d=0}^{D-1} [v_{id}(1 - v_{jd})] + [(1 - v_{id})v_{jd}] \tag{2}$$

Here, E_{ij}^d is the energy term corresponding to C_{ij}^d which denotes the total communication cost of edge (i,j) over channel d. Energy formulation for total communication is $E^C = \sum_{(i,j)\in I} E_{ij}^C$. We formulate the energy term corresponding to the imbalance cost term using the same inner product approach adopted in the general formulation [1] as follows

$$\begin{aligned} E^B &= \frac{1}{2} \sum_{i=1}^{N} \sum_{j \neq i}^{N} w_i w_j \sum_{p=0}^{P-1} \mathcal{P}(\text{tasks } i \text{ and } j \text{ are mapped to processor } p) \\ &= \frac{1}{2} \sum_{i=1}^{N} \sum_{j \neq i}^{N} w_i w_j \sum_{p=0}^{P-1} Z_{ip} Z_{jp} \end{aligned} \tag{3}$$

Here, Z_{ip} denotes the probability of finding task i at processor p, i.e.,

$$Z_{ip} = \prod_{d=0}^{D-1} u_{ip}^d, \qquad \text{where} \qquad u_{ip}^d = \begin{cases} v_{id} & \text{if} \quad p_d = 1 \\ 1 - v_{id} & \text{if} \quad p_d = 0 \end{cases} \tag{4}$$

Recall that p_d denotes the dth bit of the label of processor p. Hence, u_{ip}^d is the probability of finding task i in the subcube $H_d^{p_d}$. That is, u_{ip}^d is the probability of finding task i in the subcube H_d^1 (H_d^0) if $p_d = 1$ ($p_d = 0$). The total energy can be defined in terms of E^C and E^B as $E(\mathbf{V}) = E^C(\mathbf{V}) + \beta E^B(\mathbf{V})$, where β is introduced to maintain a balance between the two conflicting optimization objectives of the mapping problem.

2.3 Derivation of Mean Field Theory Equations

Using the expression for the proposed energy function, the expression for the mean field experienced by an Ising spin s_{id} can be obtained as

$$\phi_{id} = -\frac{\partial E(\mathbf{V})}{\partial v_{id}} = \phi_{id}^C + \beta \phi_{id}^B \qquad \text{where} \tag{5}$$

$$\phi_{id}^C = \sum_{j \in Adj(i)} e_{ij}(v_{jd} - 0.5) \qquad \phi_{id}^B = \sum_{j=1, j\neq i}^{N} \sum_{p=0}^{P-1} (-1)^{p_d} w_i w_j Z_{jp} Z_{ip}^d \tag{6}$$

Here, ϕ_{id}^C and ϕ_{id}^B denotes the communication cost and imbalance cost components of the mean field ϕ_{id}. The expression for Z_{ip}^d used in (6) is $Z_{ip}^d = \prod_{c=0,c\neq d}^{D-1} u_{ip}^c$. Note that Z_{ip}^d denotes the probability of finding task i in one of the two processors $p \in H_d^{p_d}$ and $q \in H_d^{1-p_d}$ which are connected by a communication link over channel d. Here, Z_{ip}^d can also be interpreted as the probability of finding task i at processor $p \in H_d^{p_d}$ if task i is mapped to $H_d^{p_d}$, i.e., $v_{id} = s_{id} = p_d$.

In the general MFA formulation, updating a single Potts spin updates the expected mapping of the task [1]. However, the MFA formulation proposed for hypercubes, $D = lg_2 P$ Ising spin updates are required to update the expected mapping of a particular task. That is, D Ising spins $\{s_{id}\}_{d=0}^{D-1}$ should be selected for update in order to update the overall mapping of task i. Hence, the complexity of D iterations of the proposed hypercube formulation should be compared to the complexity of a single iteration of the general formulation. Thus, the proposed hypercube formulation, together with the efficient implementation scheme proposed in [4], asymptotically reduces the complexity of a simple MFA iteration of the general formulation from $\Theta(d_{avg}P + P^2)$ to $\Theta(d_{avg}lg_2P + Plg_2P)$.

3 Experimental Results

This section presents the performance evaluation of the proposed hypercube-specific MFA formulation for the mapping problem, in comparison with the well-known mapping heuristics: general MFA formulation, Simulated Annealing (SA) and Kernighan-Lin (KL). Each algorithm is tested by mapping some test domain graphs to various dimensional hypercubes. Test graphs used for experimentation correspond to the sparse graphs associated with the symmetric sparse matrices selected from *Harwell Boeing sparse matrix test collection* [3]. Weights of the vertices are assumed to be equal to their degrees. These test graphs are mapped to various dimensional hypercubes (D=5, 6, 7, 8).

Table 1 illustrates the performance results of the KL, SA, general and hypercube-specific MFA heuristics for the generated mapping problem instances. In this table, “Gen” and “Hyp” denote the general and hypercube-specific MFA formulations, respectively. Each algorithm is executed 10 times for each mapping instance starting from different, randomly chosen initial configurations. Averages are illustrated in Table 1. Percent computational load imbalance averages of the 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 workstation. In Table 1, values displayed in parentheses denote the communication cost and execution time averages normalized with respect to those of the hypercube-specific MFA heuristic. Actual communication cost averages and execution time averages (in seconds) are displayed only for the solutions generated by the hypercube-specific MFA heuristic. In Table 1, bold values indicate the best results for the respective mappings. Table 1 confirms the expectation that hypercube-specific MFA formulation is significantly (36 times on the overall average) faster than general MFA formulation while producing mappings with considerably better qualities. The hypercube-specific MFA heuristic is as fast as the fast KL heuristic while

Table 1. Communication cost, percent load imbalance and execution time averages of the solutions found by KL, SA, general MFA and hypercube-specific MFA heuristics for various mapping problem instances. N and E denote the number of vertices and edges in the respective domain graph instances. Values in parentheses denote the communication cost and execution time averages normalized with respect to those of the hypercube-specific MFA heuristic. Bold values represent the best results for the respective mapping instances

Problem			Communication Cost				% Load Imbalance				Execution Time(sec)			
Domain	H				MFA				MFA				MFA	
Graph	D	P	KL	SA	Gen.	Hyp	KL	SA	Gen.	Hyp	KL	SA	Gen.	Hyp
	5	32	(2.13)	(**0.77**)	(1.03)	2760	8.1	**1.1**	4.0	2.6	(**0.7**)	(660)	(47.0)	60
DWT-1242	6	64	(3.53)	(1.09)	(1.50)	**2941**	8.9	**2.7**	5.2	4.3	(**0.6**)	(213)	(29.4)	201
N=1242	7	128	(7.48)	(1.36)	(1.71)	**3605**	12.1	**6.1**	6.6	7.2	(1.1)	(83)	(24.8)	**617**
E=4592	8	256	(9.47)	(1.60)	(1.88)	**4829**	16.3	**11.9**	14.0	12.4	(1.5)	(54)	(25.7)	**1247**
	5	32	(1.64)	(**0.62**)	(0.87)	1861	6.3	**1.3**	3.6	**1.3**	(**0.6**)	(530)	63.52	82
JAGMESH6	6	64	(2.43)	(**0.86**)	(1.22)	2298	7.4	**2.4**	4.1	2.7	(**0.5**)	(204)	(53.2)	224
N=1377	7	128	(4.95)	(1.07)	(1.59)	**2890**	12.0	**4.9**	5.6	5.8	(**0.8**)	(102)	(33.7)	547
E=3808	8	256	(8.01)	(1.27)	(1.67)	**3863**	13.4	11.7	12.1	**10.1**	(2.6)	(60)	(34.4)	**1283**
	5	32	(2.56)	(**0.59**)	(1.32)	1045	5.5	**0.5**	2.8	1.3	(**0.6**)	(783)	(15.2)	97
BCSPWR09	6	64	(3.09)	(**0.72**)	(1.37)	1477	10.5	**1.6**	5.6	3.2	(**0.5**)	(350)	(18.7)	225
N=1723	7	128	(5.10)	(1.08)	(1.62)	**1701**	12.4	**4.1**	7.2	6.0	(**0.7**)	(149)	(20.5)	637
E=2394	8	256	(8.22)	(1.56)	(1.74)	**2155**	17.2	**8.7**	12.8	11.1	(1.9)	(77)	(26.7)	**1563**
	5	32	(1.60)	(**0.52**)	(0.64)	3740	6.3	**0.7**	3.6	0.9	(**0.9**)	(887)	(26.9)	112
LSH2233	6	64	(2.24)	(**0.68**)	(0.99)	4490	8.0	**1.6**	3.1	2.1	(**0.6**)	(332)	(38.1)	322
N=2233	7	128	(3.66)	(**0.84**)	(1.34)	5375	9.4	**2.9**	4.4	3.6	(**0.9**)	(107)	(69.9)	800
E=6552	8	256	(6.21)	(1.01)	(1.42)	**6865**	10.2	**5.3**	8.8	6.8	(1.8)	(52)	(59.2)	**1783**
Overall normalized averages on the basis of number of processors														
	5	32	(1.98)	(**0.63**)	(0.97)	(1.0)	(4.3)	(**0.6**)	(2.3)	(1.0)	(**0.7**)	(715)	(37.7)	(1.0)
	6	64	(2.82)	(**0.84**)	(1.27)	(1.0)	(2.9)	(**0.5**)	(1.5)	(1.0)	(**0.9**)	(274)	(34.8)	(1.0)
	7	128	(5.30)	(1.10)	(1.57)	(**1.0**)	(2.0)	(**0.8**)	(1.1)	(1.0)	(**0.9**)	(110)	(37.2)	(1.0)
	8	256	(7.97)	(1.36)	(1.68)	(**1.0**)	(1.4)	(**0.9**)	(1.2)	(1.0)	(1.3)	(60)	(36.2)	(**1.0**)
Overall normalized averages														
			4.50	1.06	1.37	**1.0**	2.6	**0.7**	1.5	1.0	(**0.9**)	(274)	(36.4)	(1.0)

producing extremely better mappings. The proposed heuristic produces comparable mappings with powerful SA heuristic, however it is drastically (274 times on the average) faster. As is also seen Table 1, the relative quality difference between the mappings produced by hypercube-specific MFA and SA heuristics decreases with increasing number of processors in favor of the proposed heuristic.

4 Conclusion

An efficient MFA formulation was proposed for mapping unstructured domains to hypercube-connected distributed-memory architectures. The proposed hypercube-specific MFA formulation asymptotically reduces the number of spin variables and the complexity of the mean filed computations compared to the general MFA formulation. The proposed hypercube-specific MFA heuristic was found to be significantly faster than the general MFA heuristic as was expected.

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. 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.
3. 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.
4. Haritaoglu, I., and Aykanat Cevdet., "An efficient Mean Filed annaealing formulation for mapping unsrtuctured domains " *Technical Report,BU-CEIS-9511*
5. Saad Y. and Schultz M. " Topological Properties of Hypercubes " *IEEE Transaction on Computer*, vol 37, July, 1988