ELSEVIER

Neurocomputing 8 (1995) 171–194

NEUROCOMPUTING

Circuit partitioning using mean field annealing

Tevfik Bultan [a], Cevdet Aykanat [b,*]

[a] *Department of Computer Science, University of Maryland, College Park, MD 20742, USA*
[b] *Department of Computer Engineering and Information Science, Bilkent University, 06533 Bilkent, Ankara, Turkey*

Received 6 April 1993; accepted 7 February 1994

Abstract

Mean field annealing (MFA) algorithm, proposed for solving combinatorial optimization problems, combines the characteristics of neural networks and simulated annealing. Previous works on MFA resulted with successful mapping of the algorithm to some classic optimization problems such as traveling salesperson problem, scheduling problem, knapsack problem and graph partitioning problem. In this paper, MFA is formulated for the circuit partitioning problem using the so called net-cut model. Hence, the deficiencies of using the graph representation for electrical circuits are avoided. An efficient implementation scheme, which decreases the complexity of the proposed algorithm by asymptotical factors is also developed. Comparative performance analysis of the proposed algorithm with two well-known heuristics, simulated annealing and Kernighan-Lin, indicates that MFA is a successful alternative heuristic for the circuit partitioning problem.

Keywords: Mean field annealing; Circuit partitioning; Net-cut model

1. Introduction

Partitioning of an electrical circuit, defined by its components and signal nets, is an extensively studied problem arising in various applications. Partitioning means to divide the components of a circuit into two or more evenly weighted partitions in such a way that the cost of the connections among the partitions is minimized. The cost of the connections can be measured by the set of nets that connect cells in different partitions, called the *cut-set*. The aim is to minimize the size of the cut-set while keeping the size of the partitions balanced. This problem, called the

* Corresponding author. Email: aykanat@bilkent.edu.tr

SSDI 0925-2312(94)00016-L

circuit partitioning problem, arises while dividing a circuit into parts that will be implemented separately. Also, divide-and-conquer algorithms, used in some VLSI layout problems as placement and floor-planning, necessitate dividing the circuits hierarchically into parts with different minimization criteria. Since the circuit partitioning is extensively used in these algorithms [10], the problem becomes more important.

A heuristic for circuit partitioning is given in the seminal paper by Kernighan and Lin [7]. In this work, the circuits are represented as graphs and the problem is treated as *graph partitioning problem*. Later Schweikert and Kernighan [13] showed the deficiencies of using the graph model for representing electrical circuits, and proposed a new model called the *net-cut* model. In the net-cut model circuits are represented by hypergraphs instead of graphs. The net-cut model represents the actual interconnection cost of distributing a circuit to different parts, whereas the graph model gives an approximation to the interconnection cost. This approximation gets worse as the sizes of the nets increase [13]. Hence, treating circuit partitioning problem as graph partitioning problem can decrease the performance of the algorithm used for partitioning. Modification of the Kernighan-Lin algorithm from the graph model to the net-cut model is extensively studied [3,13].

Since the circuit partitioning problem is NP-hard [10], finding efficient heuristics is an important research issue. In recent years, new neurocomputing approaches as maximum neural network and mean field annealing are successfully applied to several NP-hard problems such as bipartite subgraph problem [9,14], module orientation problem [14], traveling salesperson problem [12,15], scheduling problem [4], knapsack problem [11], mapping problem [1], and graph partitioning problem [5,12,16]. In this work, *mean field annealing* (MFA) is formulated for solving circuit partitioning problem using the net-cut model. Yih and Mazumder used the net-cut model when they applied *Hopfield neural network model* to the circuit partitioning problem [17]. MFA combines the collective computation property of Hopfield neural network model [6] with the annealing notion of *simulated annealing* [8] in order to form a better algorithm [15]. In MFA, discrete variables called *spins* (or *neurons*) 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. MFA is also a general strategy as simulated annealing, and can be applied to different problems with suitable formulations. We show that formulating the MFA algorithm for the net-cut model is not trivial but achievable, and the resulting algorithm is efficient both in solution quality and execution time.

The organization of the paper is as follows. Section 2 presents a formal definition of the circuit partitioning problem. The graph and the net-cut model representations of circuits and the deficiencies of the graph model are also discussed in Section 2. Section 3 presents the proposed formulation of the MFA algorithm for the circuit partitioning problem using the net-cut model. An efficient implementation scheme is also described in this section. Section 4 presents the

experimental performance evaluation of the proposed MFA algorithm for the circuit partitioning problem in comparison with two well-known heuristics: simulated annealing and Kernighan-Lin.

2. Circuit partitioning problem

An instance of the circuit partitioning problem consists of a circuit that is to be partitioned and an integer K representing the number of partitions. A circuit can be represented by a set of components called *cells*, and a list of *nets* which defines the connection relationships among the cells. Cells may represent different electrical components as transistors, standard cells or logic gates. Nets represent the connections among the cells that can be realized using different types of conductors depending on the application (e.g. wires, metal layers). Cells incident to a net are called the *terminals* of that net. Both cells and nets of a circuit have an attribute called the weight of a cell or a net. Weights of the cells may represent their areas if the partitioning is used for placement. Nets can be weighted due to their effect on the total delay of the circuit. An example circuit with 10 cells and 5 nets is given below.

cells (weights):
c_1 (4), c_2 (1), c_3 (4), c_4 (3), c_5 (4), c_6 (2), c_7 (3), c_8 (2), c_9 (1), c_{10} (2)
nets (weights):
n_1: $c_4 - c_5 - c_9$ (2) n_2: $c_1 - c_4 - c_6 - c_9$ (3) n_3: $c_2 - c_5 - c_7 - c_{10}$ (1)
n_4: $c_3 - c_8$ (2) n_5: $c_1 - c_3 - c_4 - c_6 - c_9$ (1)

A circuit Ω can be formally represented by a set of cells C, a set of nets N, a cell weight function w_{cell}: $C \rightarrow \mathcal{N}$, and a net weight function w_{net}: $N \rightarrow \mathcal{N}$, where $\mathcal{N}$ represents the set of natural numbers. Each element in the set N is a subset of set C, i.e. $N \subseteq 2^C$.

Given a circuit as defined above, the problem is to divide the cells of the circuit into $K (K \geq 2)$ evenly weighted partitions while minimizing the *cost* of the *external connections* (i.e. cut-set size) among partitions. The difference between the net-cut and graph models is in the computation of the cost of external connections.

In the graph representation of a circuit, each cell of the circuit is represented by a vertex and each net of the circuit is represented by a clique of vertices corresponding to its terminals. Cell weight function becomes the vertex weight function of the graph. Weights of the edges are equal to the weights of the nets that they represent. The graph representation of the circuits can be restricted to simple graphs. All edges between two vertices are represented by a single edge of which weight is the summation of the weights of the edges it represents. This simplification has no effect as far as the partitioning is concerned. If an edge between two vertices is in the cut set of a partitioning then all other edges between these two vertices are also in the cut set and vice versa. Therefore, a single edge with a weight equal to the summation of the weights of these edges can represent

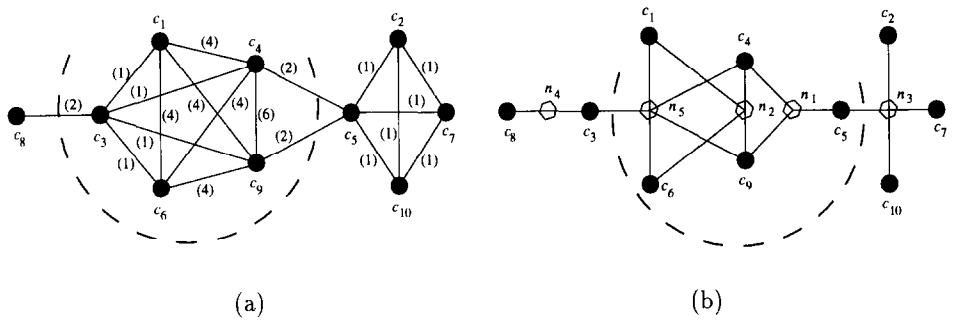


Fig. 1. (a) Graph and (b) hypergraph representations of the example circuit. Weights of the edges in the graph representation are shown in parenthesis.

their contribution to the cost. Fig. 1(a) illustrates the graph representation of the example circuit. Formally, a circuit $\Omega(C, N)$ is represented by a graph $G(V, E)$, where $V = C$, and $w_{vertex} = w_{cell}$. The edge set E is formed using the net set N as, $uv \in E$ if and only if there exists an $n \in N$ such that $u \in n$ and $v \in n$. The weight function w_{edge} is computed as $w_{edge}(uv) = \Sigma_{u,v \in n, n \in N} w_{net}(n)$ for all $uv \in E$. In the graph model, the connection cost is computed by simply adding the weights of the edges that have their vertices in different partitions.

In the net-cut model, electrical circuits are represented with hypergraphs. A hypergraph consists of a set of vertices V and a set of hyperedges $E \subseteq 2^V$. Hyperedges can be incident to more than two vertices. Note that, hypergraphs can represent the circuits exactly by representing cells by vertices and nets by hyperedges. Cell and net weight functions of the circuit become the vertex and hyperedge weight functions of the hypergraph. Hence, a circuit $\Omega(C, N)$ can be considered as a hypergraph where C is the set of vertices and N is the set of hyperedges. Fig. 1(b) illustrates the hypergraph representation of the example circuit.

In the net-cut model, the connection cost for K-way partitioning may be computed as follows. If the vertices incident to an hyperedge are in l different partitions, then that hyperedge contributes $(l-1)w_e$ to the connection cost, where w_e is the weight of the hyperedge. There are also some other alternatives for computing the connection cost. One of them is adding w_e to the connection cost if and only if $l \geq 2$. Another one is adding $(l(l-1)/2)w_e$. Note that, all choices are equivalent for bipartitioning.

The problem with the graph model is that it treats a net with s terminals as $s(s-1)/2$ two terminal nets. This strategy exaggerates the importance of the nets that have more than two terminals and the exaggeration grows with the square of the size of the net [13], where the size of a net denotes the number of terminals of a net. For example, the actual cost of a unit weight net of size 4 in the cut-set of a bipartitioning is 1 since such a net will cause a single connection between the two partitions. In the graph model, the same situation contributes a cost of 3 or 4 according to the distribution of the terminal cells of the net between the two partitions. This cost contribution in the graph model is far from the actual cost. In

general, the actual cost contribution of a unit weight net across a cut of a bipartitioning is 1, but the cost contribution of a clique, which is evenly split across a cut, rises quadratically with the size of the clique [10]. This quadratic growth does not adequately reflect the costs arising in practice. In fact, heuristics using the graph model for representing circuits will try to remove all nets with large sizes from the cut-set and try to put the smaller ones. This situation can cause performance degradation if the actual cut size is minimized when the nets with large sizes are in the cut-set. Experimentation shows that this occurs in most of the cases [13]. For example, using the net-cut model instead of the graph model increases the performance of the Kernighan-Lin heuristic drastically, reducing the connection costs by 19 to 50% [13].

Fig. 1(a) and (b) illustrate two bipartitionings of the example circuit. The bipartitioning, $P_1 = \{c_1, c_3, c_4, c_6, c_9\}$ and $P_2 = \{c_2, c_5, c_7, c_8, c_{10}\}$ with sizes $|P_1| = 14$ and $|P_2| = 12$, illustrated in Figure 1(a) is the global minimum of the graph model if the sizes of the two partitions are restricted to be between 12 and 14. The bipartitioning, $P_1 = \{c_1, c_4, c_5, c_6, c_9\}$ and $P_2 = \{c_2, c_3, c_7, c_8, c_{10}\}$ with sizes $|P_1| = 14$ and $|P_2| = 12$, illustrated in Fig. 1(b) is the global minimum of the net-cut model with the same restrictions on the partition sizes. Here, size of a partition denotes the summation of the weights of the cells assigned to that partition. As explained earlier, actual connection cost is the cost of the cut computed using the net-cut model. Hence, to compute the actual cost of the cut in Fig. 1(a) we transform it to the net-cut model and observe that the size of the cut is 4, whereas the size of the cut in Fig. 1(b) is 2. Note that, the global minimum solution of the graph model cuts two nets (n_1 and n_4) with smaller sizes (3 and 2, respectively) although their weights are high (2 for both of the nets). However, the global minimum solution of the net-cut model cuts two nets (n_3 and n_5) with larger sizes (4 and 5, respectively) but smaller weights (1 for both of the nets). Although both cuts give the global minimum according to the model used, min-cut bipartitioning using the graph model yields a suboptimal solution because of the incorrect representation of the problem. This demonstrates that even if one computes the global optimum using the graph model, the computed solution can be a suboptimal solution of the actual problem. It can be argued that some other representation scheme can be used to represent circuits with graphs which can give better approximations to the actual cost, but it can be shown that there is no good way of mapping a circuit instance into a graph [10].

3. Applying MFA to the circuit partitioning problem

Mean field annealing (MFA) merges collective computation and annealing properties of Hopfield neural networks [6] and simulated annealing [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 (neurons) can be decoded as a solution to the target problem. Then, an energy function is

constructed whose global minimum value corresponds to an optimum solution of the problem to be solved. MFA is expected to compute the optimum solution to the target problem, starting from a randomly chosen initial state, by minimizing this energy function. Steps of applying mean field annealing 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 in order to get an efficient algorithm.
(5) Select the energy function and the cooling schedule parameters.

The proposed formulation and implementation of the MFA algorithm for the circuit partitioning problem following these steps are presented in the following sections.

3.1. Encoding

The MFA algorithm is derived by analogy to *Ising* and *Potts* models which are used to estimate the state of a system of particles, called spins, in thermal equilibrium. In Ising model, spins can be in one of the two states represented by 0 and 1, whereas in Potts model they can be in one of the K states. For the circuit partitioning problem, Ising model can be used for bipartitioning whereas Potts model is suitable for K-way partitioning. In this work, we use Potts model which is more general, but for the case $K = 2$, it is easy to convert the formulations derived for Potts model to Ising model.

In the K state Potts model of S spins, the states of spins are represented using S K-dimensional vectors [12]

$$\mathbf{S_i} = [s_{i1}, \ldots, s_{ik}, \ldots, s_{iK}]^t \quad \text{for } 1 \le i \le S.$$

where 't' denotes the vector transpose operation. The spin vector $\mathbf{S_i}$ is allowed to be equal to one of the principal unit vectors $\mathbf{e_1}, \ldots, \mathbf{e_k}, \ldots, \mathbf{e_K}$, and can not take any other value. Principal unit vector $\mathbf{e_k}$ is defined to be a vector which has all its components equal to 0 except its kth component which is equal to 1. Spin $\mathbf{S_i}$ is said to be in state k if it is equal to $\mathbf{e_k}$. Hence, a K state Potts spin $\mathbf{S_i}$ is composed of K two state variables $s_{i1}, \ldots, s_{ik}, \ldots, s_{iK}$, where $s_{ik} \in \{0, 1\}$, with the following constraint

$$\sum_{k=1}^{K} s_{ik} = 1 \quad \text{for } 1 \le i \le S. \tag{1}$$

In our encoding of the circuit partitioning problem, each spin vector corresponds to a cell in the circuit $\Omega(C, N)$. Hence, number of spin vectors is $S = |C|$. Dimension K of the spin vectors is equal to the number of partitions. If a spin is in state k we say that the corresponding cell is assigned to partition k. Hence, $s_{ik} = 1$ means that cell i is assigned to partition k. For example, a 4 way partitioning of the circuit given in Section 2 can be represented by the following spin matrix $\mathbf{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_i, \ldots, \mathbf{S}_{|C|}]^t$ which consists of 10 spin vectors of dimension 4 representing the 10 cells in the example circuit.

	K Partitions			
	1	**2**	**3**	**4**
1	0	0	1	0
2	0	1	0	0
3	0	0	0	1
4	1	0	0	0
5	0	1	0	0
6	0	0	1	0
7	1	0	0	0
8	0	0	0	1
9	1	0	0	0
10	0	1	0	0

(|C| Cells)

If this spin matrix is decoded as described above, the resulting partitioning is $P_1 = \{c_4, c_7, c_9\}$, $P_2 = \{c_2, c_5, c_{10}\}$, $P_3 = \{c_1, c_6\}$ and $P_4 = \{c_3, c_8\}$ where P_1, P_2, P_3, P_4 are the sets representing partitions. Sizes of the partitions are $|P_1| = 7$, $|P_2| = 7$, $|P_3| = 6$ and $|P_4| = 6$. The size of partition k is defined to be $|P_k| = \Sigma_{c \in P_k} w_c$ where w_c denotes the weight of cell c. The interconnection cost computed according to the net-cut model is 8. This encoding is similar to the encodings used for graph partitioning problem [2,5,12,16] and bipartite subgraph problem [9,14] in the previous works. Although we have proposed a hypergraph partitioning formulation using this encoding in an earlier work [2], its energy function formulation was an approximation to the net-cut model as is the case for graph partitioning formulation. In the next section, we propose an energy function formulation according to the net-cut model using the encoding described above.

3.2. Energy function formulation

In the MFA algorithm, the aim is to find the spin values minimizing the energy function of the system. In order to achieve this goal, the average (expected) value

$\mathbf{V_i} = \langle \mathbf{S_i} \rangle$ of each spin vector $\mathbf{S_i}$ is computed and iteratively updated until the system stabilizes at some fixed point. Hence, we define

$$\mathbf{V_i} = \langle \mathbf{S_i} \rangle \quad \text{for } 1 \le i \le |C|$$

$$[v_{i1}, \ldots, v_{ik}, \ldots, v_{iK}]^t = [\langle s_{i1} \rangle, \ldots, \langle s_{ik} \rangle, \ldots, \langle s_{iK} \rangle]^t \quad \text{for } 1 \le i \le |C|$$

i.e. $v_{ik} = \langle s_{ik} \rangle$, for $1 \le i \le |C|$ and $1 \le k \le K$. Note that, $s_{ik} \in \{0, 1\}$, i.e. s_{ik} can take only two values 0 and 1, whereas $v_{ik} \in [0,1]$, i.e. v_{ik} can take any real value between 0 and 1. When the system is stabilized, v_{ik} values are expected to converge to 0 or 1. As the system is a Potts glass we have the following constraint similar to Eq. (1)

$$\sum_{k=1}^{K} v_{ik} = 1, \quad \text{for } 1 \le i \le |C|. \tag{2}$$

This constraint guarantees that each Potts spin $\mathbf{S_i}$ is in one of the K states at a time, and each cell is assigned to only one partition for our encoding of the circuit partitioning problem.

In order to construct an energy function it is helpful to associate the following meaning to the values v_{ik},

$$v_{ik} = \mathscr{P}\{\text{cell } i \text{ is in partition } k\} \quad \text{for } 1 \le i \le |C|, \quad 1 \le k \le K$$

i.e. v_{ik} is the probability of finding spin i at state k. If $v_{ik} = 1$ then spin i is in state k and the corresponding configuration is $\mathbf{S_i} = \mathbf{V_i}$.

Now, we formulate the interconnection cost of the circuit partitioning problem for the circuit $\Omega(C, N)$ as an energy term (E_C)

$$E_C(\mathbf{V}) = \sum_{n \in N} w_n \left(\sum_{k=1}^{K} \mathscr{P}\{\text{one or more cells of net } n \text{ is in partition } k\} - 1 \right) \tag{3}$$

$$= \sum_{n \in N} w_n \left(\sum_{k=1}^{K} (1 - \mathscr{P}\{\text{no cell of net } n \text{ is in partition } k\}) - 1 \right) \tag{4}$$

$$= \sum_{n \in N} w_n \left(\sum_{k=1}^{K} \left(1 - \prod_{i \in n} \mathscr{P}\{\text{cell } i \text{ is not in partition } k\} \right) - 1 \right) \tag{5}$$

$$= \sum_{n \in N} w_n \left(\sum_{k=1}^{K} \left(1 - \prod_{i \in n} (1 - v_{ik}) \right) - 1 \right) \tag{6}$$

where $\mathbf{V} = [\mathbf{V_1}, \ldots, \mathbf{V_i}, \ldots, \mathbf{V_{|C|}}]^t$ is the spin average matrix consisting of $|C|$ K-dimensional spin vectors. Here, $i \in n$ and w_n denote a terminal cell and the weight of net n respectively. In this formulation, cost of each net is computed one by one and added to the total interconnection cost. According to the net-cut model, as discussed in the Section 2, cost contribution of a net n to the total interconnection cost is $(l-1)w_n$ if the net is distributed to l different partitions. Eq. (3) follows by the observation $l = \sum_{k=1}^{K} \mathscr{P}\{\text{one or more cells of net } n \text{ is in partition } k\}$. The (-1) term in Eq. (3) is a constant term and can be eliminated. Another observation is

$\mathscr{P}\{\text{cell } i \text{ is not in partition } k\} = (1 - v_{ik})$ which follows from the probability interpretation of variable v_{ik}. Hence, $\prod_{i \in n}(1 - v_{ik})$ denotes the probability that no cell of net n is in partition k and $(1 - \prod_{i \in n}(1 - v_{ik}))$ denotes the probability that at least one cell of net n is in partition k. Note that, minimization of E_C corresponds to minimization of the actual interconnection cost of the circuit partitioning problem.

Another term of the energy function is the term for penalizing imbalanced partitions. We formulate this term (E_B) similar to the formulation of balance term proposed for the mapping problem [1].

$$E_B(\mathbf{V}) = \tfrac{1}{2} \sum_{i=1}^{|C|} \sum_{j \neq i} \sum_{k=1}^{K} v_{ik} v_{jk} w_i w_j \tag{7}$$

where w_i and w_j denote the weights of cells i and j. This triple summation term computes the summation of the inner products of the weights of the cells assigned to individual partitions. Global minimum of this term occurs when equal amounts of cell weights are assigned to each partition. If there is an imbalance in the partitioning, E_B term increases with the square of the amount of the imbalance, penalizing imbalanced partitionings.

The total energy function E can be defined in terms of E_C and E_B as

$$\begin{aligned} E(\mathbf{V}) &= E_C(\mathbf{V}) + r \times E_B(\mathbf{V}) \\ &= \sum_{n \in N} w_n \left(\sum_{k=1}^{K} \left(1 - \prod_{i \in n} (1 - v_{ik}) \right) - 1 \right) + \frac{r}{2} \sum_{i=1}^{|C|} \sum_{j \neq i} \sum_{k=1}^{K} v_{ik} v_{jk} w_i w_j \end{aligned} \tag{8}$$

where parameter r is introduced to maintain a balance between the two optimization objectives of the circuit partitioning problem. Hence, minimization of the energy function E corresponds to evenly distributing cells among K partitions while minimizing the interconnection cost among the partitions computed according to the net-cut model.

3.3. Derivation of the mean field theory equations

Mean field theory equations, needed to minimize the energy function E, can be derived as [12,16]

$$\phi_{ik} = -\frac{\partial E(\mathbf{V})}{\partial v_{ik}} \tag{9}$$

$$= -\sum_{n \in N_i} w_n \prod_{j \in n, j \neq i} (1 - v_{jk}) - r \sum_{j \neq i}^{|C|} v_{jk} w_i w_j \quad \text{for } 1 \leq i \leq |C|, \quad 1 \leq k \leq K \tag{10}$$

where N_i is defined to be the set of nets connected to cell i. The quantity ϕ_{ik} represents the kth element of the *mean field* vector effecting on spin i. Using the mean field values ϕ_{ik}, average spin values v_{ik} can be updated using the following

equation [12,16]

$$v_{ik} = \frac{e^{\phi_{ik}/T}}{\sum_{l=1}^{K} e^{\phi_{il}/T}} \quad \text{for } 1 \le i \le |C|, \quad 1 \le k \le K \tag{11}$$

where T is the temperature parameter which is used the relax the system iteratively. Eq. (11) enforces the summation of each row of the spin matrix to be unity, handling the constraint given in Eq. (2). Hence, it is guaranteed that all rows of the spin matrix will have only one spin with output value 1 when the system is stabilized.

Mean field ϕ_{ik} can be interpreted as the decrease in the energy function $E(\mathbf{V})$ when spin i is assigned to state k. Note that in Eq. (10), first summation term represents the increase in the total interconnection cost by assigning cell i to partition k. Second summation term represents the increase in the imbalance cost associated with partition k by assigning cell i to partition k. Hence, $-\phi_{ik}$ may be interpreted as the decrease in the overall solution quality by assigning cell i to partition k. Then, in Eq. (11), v_{ik} is updated such that the probability of assigning cell i to partition k increases with increasing mean field ϕ_{ik}.

After the mean field theory equations (Eq. (10), Eq. (11)) are derived, mean field annealing algorithm can be summarized as follows. First an initial, high temperature spin average is assigned to each spin, and an initial temperature is chosen. In general v_{ik} is initialized to $1/K$ plus a disturbance term (note that, $\lim_{T\to\infty} v_{ik} = 1/K$). In each iteration the mean field vector effecting on a randomly selected spin is computed using Eq. (10). Then, spin average vector is updated using Eq. (11). This process is repeated for a random sequence of spins until the system is stabilized for the current temperature. The system is observed after each spin vector update in order to detect the convergence to an equilibrium state for a given temperature. If energy function E does not decrease in most of the successive spin vector updates, this means that the system is stabilized for that temperature. Then, T is decreased according to the cooling schedule, and iterative process is re-initiated. Note that, the computation of the energy difference ΔE necessitates the computation of E (Eq. (8)) at each iteration. In general, the computation of the total energy (Eq. (8)) is much more expensive than the computation of the mean field vector. Hence, the computation of E at each iteration drastically increases the complexity of a MFA iteration. For example, the complexity of computing the energy function E is $\Theta(|N| s_{avg} K + |C|^2 K)$ for the proposed formulation (Eq. (8)). Here, s_{avg} denotes the average number of cells of a net (i.e. average size of a net). We present an efficient scheme [1] which reduces the complexity of energy difference computation by asymptotical factors.

The *incremental* energy change δE_{ik} due to the *incremental* change δv_{ik} in the value of v_{ik} is $\delta E = \delta E_{ik} = \phi_{ik} \delta v_{ik}$ from Eq. (9). Since $E(\mathbf{V})$ is linear in v_{ik} (see Eq. (8)), this equation is valid for any amount of change Δv_{ik} in the value of v_{ik}, that is

$$\Delta E = \Delta E_{ik} = \phi_{ik} \Delta v_{ik} \tag{12}$$

1. Get the initial temperature T_0, and set $T = T_0$

2. Initialize the spin averages $\mathbf{V} = [v_{11}, \ldots, v_{ik}, \ldots, v_{|C|K}]$

3. While temperature T is in the cooling range DO

   3.1 While E is decreasing DO

      3.1.1 Select a cell i at random.

      3.1.2 Compute mean field vector corresponding to the i-th spin

      $$\phi_{ik} = -\sum_{n \in N_i} w_n \prod_{j \in n, j \neq i} (1 - v_{jk}) - r \sum_{j \neq i}^{|C|} v_{jk} w_i w_j \quad \text{for } 1 \leq k \leq K$$

      3.1.3 Compute the summation $\sum_{l=1}^{K} e^{\phi_{il}/T}$

      3.1.4 Compute new spin average vector $\mathbf{V_i}$ as $v_{ik}^{(new)} = e^{\phi_{ik}/T} / \sum_{l=1}^{K} e^{\phi_{il}/T}$ for $1 \leq k \leq K$

      3.1.5 Compute the energy change $\Delta E = \sum_{k=1}^{K} \phi_{ik}(v_{ik}^{(new)} - v_{ik})$

      3.1.6 Update the spin average vector $\mathbf{V_i}$ as $v_{ik} = v_{ik}^{(new)}$ for $1 \leq k \leq K$

   3.2 $T = \alpha \times T$

Fig. 2. The proposed MFA algorithm for the circuit partitioning problem.

At each iteration of the MFA algorithm, K v_{ik} values in the same spin average vector are updated in a synchronous manner, and Eq. (12) is valid for all updates performed in a particular iteration. Thus, energy difference due to the spin vector update operation in a particular iteration can be computed as

$$\Delta E = \sum_{k=1}^{K} \phi_{ik} \Delta v_{ik} \tag{13}$$

where $\Delta v_{ik} = v_{ik}^{(new)} - v_{ik}^{(old)}$. The complexity of computing Eq. (13) is only $\Theta(K)$ since mean field (ϕ_{ik}) values are already computed for the spin updates.

The MFA algorithm derived from the proposed formulation of the circuit partitioning problem is shown in Fig. 2. The complexity analysis of one iteration of this algorithm (from step 3.1.1 to step 3.1.6 in Fig. 2) is as follows. The complexity of computing the first summation term in Eq. (10) is $\Theta(d_{avg} s_{avg})$ where d_{avg} denotes the average number of nets incident to a cell (i.e., average *degree* of a cell). The second summation in Eq. (10) is a $\Theta(|C|)$ operation. Thus, the complexity of a single mean field (ϕ_{ik}) computation is $\Theta(d_{avg} s_{avg} + |C|)$. Hence, the complexity of computing a mean field vector corresponding to a selected spin (step 3.1.2) is $\Theta(d_{avg} s_{avg} K + |C| K)$. Spin update computations (steps 3.1.3, 3.1.4 and 3.1.6) and energy difference computation (step 3.1.5) are both $\Theta(K)$ operations. Hence, the overall complexity of a single MFA iteration is $\Theta(d_{avg} s_{avg} K + |C| K)$.

3.4. An efficient implementation scheme

As mentioned earlier, the MFA algorithm proposed for the circuit partitioning problem is an iterative process. The complexity of a single MFA iteration is mainly due to the mean field vector computation. In this section, we propose an efficient implementation scheme which reduces the complexity of the mean field computations, and hence the complexity of the MFA iteration, by asymptotical factors.

Assume that, cell i is selected at random for updating the spin average vector $\mathbf{V_i}$ in a particular iteration. The expression given for ϕ_{ik} (Eq. (10)) can be rewritten as

$$\phi_{ik} = -\sum_{n \in N_i} \lambda_{ik}^n - r\psi_{ik} \quad \text{for } 1 \le k \le K \tag{14}$$

where

$$\lambda_{ik}^n = w_n \prod_{j \in n, j \neq i} (1 - v_{jk}) \quad \text{for } 1 \le k \le K \tag{15}$$

$$\psi_{ik} = \sum_{j \neq i}^{|C|} v_{jk} w_i w_j \quad \text{for } 1 \le k \le K. \tag{16}$$

For the sake of clarity of the representation, the overall mean field computations involved in a single iteration can be expressed using vector representation as

$$\boldsymbol{\Phi}_\mathbf{i} = -\sum_{n \in N_i} \boldsymbol{\Lambda}_\mathbf{i}^\mathbf{n} - r\boldsymbol{\Psi}_\mathbf{i}. \tag{17}$$

Here, $\boldsymbol{\Lambda}_\mathbf{i}^\mathbf{n}$ and $\boldsymbol{\Psi}_\mathbf{i}$ are column vectors with K elements, where

$$\boldsymbol{\Phi}_\mathbf{i} = [\phi_{i1}, \ldots, \phi_{ik}, \ldots, \phi_{iK}]^t \quad \boldsymbol{\Psi}_\mathbf{i} = [\psi_{i1}, \ldots, \psi_{ik}, \ldots, \psi_{iK}]^t$$
$$\boldsymbol{\Lambda}_\mathbf{i}^\mathbf{n} = [\lambda_{i1}^n, \ldots, \lambda_{ik}^n, \ldots, \lambda_{iK}^n]^t \quad \text{for } n \in N_i.$$

The complexity of computing the $\boldsymbol{\Psi}_\mathbf{i}$ vector can be reduced asymptotically [1] if the computation of ψ_{ik} in Eq. (16) is re-formulated as

$$\psi_{ik} = \sum_{j \neq i}^{|C|} v_{jk} w_i w_j = w_i \left(\sum_{j=1}^{|C|} v_{jk} w_j - v_{ik} w_i \right) = w_i (\gamma_k - v_{ik} w_i) \quad \text{for } 1 \le k \le K \tag{18}$$

where $\gamma_k = \sum_{j=1}^{|C|} v_{jk} w_j$. Here, γ_k represents the current size of partition k prior to the update of the spin average vector $\mathbf{V_i}$. Computationally γ_k represents the weighted sum of the individual v_{ik} values of the k-th column of the spin matrix. At the beginning of the MFA algorithm, the initial γ_k value for each column k ($1 \le k \le K$) can be computed using the initial spin values. Then γ_k values can be updated at the end of each iteration (i.e. after spin average vector $\mathbf{V_i}$ is updated) using

$$\gamma_k^{(new)} = \gamma_k^{(old)} - v_{ik}^{(old)} w_i + v_{ik}^{(new)} w_i = \gamma_k^{(old)} + w_i \Delta v_{ik} \quad \text{for } 1 \le k \le K. \tag{19}$$

This formulation proposed for the efficient computation of the $\boldsymbol{\Psi}_i$ vector, which is needed in Eq. (17), can be represented in vector notation as

$$\boldsymbol{\Psi}_i = w_i\left(\boldsymbol{\Gamma}^{(old)} - w_i \mathbf{V}_i^{(old)}\right) \tag{20}$$

$$\boldsymbol{\Gamma}^{(new)} = \boldsymbol{\Gamma}^{(old)} + w_i \boldsymbol{\Delta}\mathbf{V}_i \tag{21}$$

where $\boldsymbol{\Gamma} = [\gamma_1, \ldots, \gamma_k, \ldots, \gamma_K]^t$ and $\boldsymbol{\Delta}\mathbf{V}_i = [\Delta v_{i1}, \ldots, \Delta v_{ik}, \ldots, \Delta v_{iK}]^t$. The computation of initial γ_k values can be excluded from the complexity analysis since they are computed only once at the very beginning of the algorithm. In this scheme, the computation of an individual ψ_{ik} using Eq. (18) is a $\Theta(1)$ operation. Hence, the construction of the $\boldsymbol{\Psi}_i$ vector becomes a $\Theta(K)$ operation (Eq. (20)). The update of an individual γ_k value (using Eq. (19)) at the end of each iteration is a $\Theta(1)$ operation. Thus, the overall complexity of γ_k updates is $\Theta(K)$ since K weighted column sums should be updated (Eq. (21)). Hence, the proposed scheme reduces the complexity of computing the $\boldsymbol{\Psi}_i$ vector (needed in Eq. (17)) from $\Theta(|C|K)$ to $\Theta(K)$.

The complexity of computing $\boldsymbol{\Lambda}_i^n$ vector can be reduced asymptotically if the computation of an individual λ_{ik}^n value in Eq. (15) is reformulated as

$$\begin{aligned}\lambda_{ik}^n &= w_n \prod_{j \in n, j \neq i} (1 - v_{jk}) \\ &= \frac{1}{(1 - v_{ik})} w_n \prod_{j \in n} (1 - v_{jk}) = \frac{1}{(1 - v_{ik})} \pi_k^n \quad \text{for } 1 \leq k \leq K\end{aligned} \tag{22}$$

where $\pi_k^n = w_n \prod_{j \in n}(1 - v_{jk})$. Here, π_k^n represents the probability that no cell of net n is in partition k multiplied by the weight of the net n. At the beginning of the MFA algorithm, the initial $\boldsymbol{\Pi}^n = [\pi_1^n, \ldots, \pi_k^n, \ldots, \pi_K^n]^t$ vector for each net can be computed using the initial spin averages. Then, π_k^n values can be updated at the end of each iteration (i.e. after the spin average vector $\mathbf{V}_i$ is updated) using

$$\pi_k^{n(new)} = \frac{\left(1 - v_{ik}^{(new)}\right)}{\left(1 - v_{ik}^{(old)}\right)} \pi_k^{n(old)} \quad \text{for } 1 \leq k \leq K \text{ and } \forall n \in N_i. \tag{23}$$

This formulation proposed for the efficient computation of an individual $\boldsymbol{\Lambda}_i^n$ vector, which is needed in Eq. (17), can be represented in vector notation as

$$\boldsymbol{\Lambda}_i^n = \mathbf{U}_i \times \boldsymbol{\Pi}^{n(old)} \qquad \text{for } \forall n \in N_i \tag{24}$$

$$\boldsymbol{\Pi}^{n(new)} = \mathbf{R}_i \times \boldsymbol{\Pi}^{n(old)} \qquad \text{for } \forall n \in N_i. \tag{25}$$

Here, $\mathbf{U}_i = [u_{i1}, \ldots, u_{ik}, \ldots, u_{iK}]^t$ and $\mathbf{R}_i = [r_{i1}, \ldots, r_{ik}, \ldots, r_{iK}]^t$ are column vectors with K elements where $u_{ik} = 1/(1 - v_{ik}^{(old)})$ and $r_{ik} = (1 - v_{ik}^{(new)})/(1 - v_{ik}^{(old)})$ for $1 \leq k \leq K$ and the operation '$\times$' represents element-by-element multiplication of two column vectors. The computation of initial $\boldsymbol{\Pi}^n$ vectors can be excluded from the complexity analysis since they are computed only once at the very beginning of the algorithm. In this scheme, the computation of an individual λ_{ik}^n value for a particular net n using Eq. (22) is a $\Theta(1)$ operation. Hence, the construction of a $\boldsymbol{\Lambda}_i^n$ vector becomes a $\Theta(K)$ operation (Eq. (24)). The update of an individual π_k^n value

for a particular net n at the end of each iteration is a $\Theta(1)$ operation (Eq. (23)). Thus, the overall complexity of updating a particular Π^n vector is a $\Theta(K)$ operation (Eq. (25)). Hence, the proposed scheme reduces the complexity of computing an individual Λ_i^n vector (needed in Eq. (17)) from $\Theta(s_n K)$ to $\Theta(K)$ where s_n denotes the size of the net n.

The first summation term in Eq. (17) requires the addition of d_i Λ_i^n vectors where d_i denotes the degree of cell i. Furthermore, the proposed scheme necessitates the update of d_i Π^n vectors since cell i is connected to d_i different nets. Thus, the proposed scheme reduces the complexity of computing the first summation term in Eq. (17) from $\Theta(K\Sigma_{n\in N_i} s_n) = \Theta(Kd_i s_{avg}^i)$ to $\Theta(Kd_i)$. Here, $s_{avg}^i = (\Sigma_{n\in N_i} s_n)/d_i$ denotes the average size of the nets connected to cell i. The final addition of vectors $\Lambda_i = -\Sigma_{n\in N_i}\Lambda_i^n$ and $(-r)\times\Psi_i$ is a $\Theta(K)$ operation. Hence, the proposed scheme reduces the overall complexity of mean field vector computation in a single MFA iteration from $\Theta(d_{avg}s_{avg}K + |C|K)$ to $\Theta(d_{avg}K)$. Recall that spin update computations and energy difference computation involved in a MFA iteration are $\Theta(K)$ operations. Hence, the proposed implementation scheme reduces the overall complexity of an individual MFA iteration to $\Theta(d_{avg}K)$.

4. Performance of the MFA algorithm

This section presents the performance evaluation of the proposed Mean Field Annealing (MFA) algorithm for the circuit partitioning problem. To evaluate the performance of the proposed algorithm two well-known circuit partitioning heuristics are used: simulated annealing (SA) and Kernighan-Lin (KL). Each algorithm is tested using randomly generated circuit partitioning problem instances. Hypergraphs representing circuits are generated using two different schemes, resulting with two families of hypergraphs referred here as random hypergraphs and geometric hypergraphs.

Random hypergraphs are generated using the following parameters: number of cells ($|C|$), number of nets ($|N|$), maximum cell weight (W_c), maximum net weight (W_n), and maximum net size (s_{max}). Each net is generated by randomly selecting a net size between 2 and s_{max}. Then, that many cells are selected randomly from the cell set to form the net. If a new generated net contains exactly same cells as another net generated earlier, then it is discarded and another net is generated instead of it. Each cell or net is weighted randomly by choosing a number between 1 and W_c or 1 and W_n, respectively.

Geometric hypergraphs are generated using an algorithm similar to the one used for generating geometric graphs. Geometric hypergraphs may represent electrical circuits better than random hypergraphs as they present clustering and local connectivity properties. Parameters used for generating geometric hypergraphs are number of cells ($|C|$), number of nets ($|N|$), maximum cell weight (W_c), maximum net weight (W_n), and average net size (s_{avg}). A geometric hypergraph is generated using these parameters by randomly distributing $|C|$ cells and $|N|$ nets in a unit square. Then, the nets are formed using the following rule: a cell

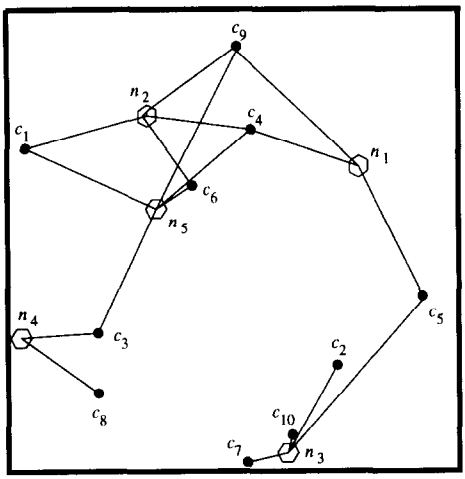


Fig. 3. An example geometric hypergraph.

is incident to a net if it is contained in the bounding box of that net. Bounding box of a net is a square with the net in its center. Sizes of the bounding boxes are fixed and computed using the input parameters s_{avg} and $|C|$ as $b = \sqrt{s_{avg}/|C|}$, where b denotes the length of the sides of the bounding box. Nets of which bounding boxes contain less than 2 cells are discarded. Also, as in the generation of random hypergraphs, if two or more nets have the same subset of cells, only one of them is accepted and all others are discarded. New nets are generated instead of the discarded ones. Note that, the average size of the nets in the resulting hypergraph may be slightly different than the input parameter s_{avg} as the nets near the borders of the unit square will have smaller sizes than expected, but also nets having less than 2 terminals are discarded which may compensate this effect. Each cell or net is weighted randomly by choosing a number between 1 and W_c or 1 and W_n, respectively. Fig. 3 illustrates a geometric hypergraph with 10 cells and 5 nets which corresponds to the example circuit given in Section 2.

4.1. MFA implementation

The MFA algorithm proposed for the circuit partitioning problem is implemented efficiently as described in Section 3. At the very beginning of the algorithm spin averages are initialized to $1/K$ plus a random disturbance term, so that the initial spin averages are uniformly distributed in the range

$$0.9 \times \frac{1}{K} \leq v_{ik}^{(initial)} \leq 1.1 \times \frac{1}{K} \quad \text{for } 1 \leq i \leq |C|, \quad 1 \leq k \leq K.$$

The initial temperature T_0 and the parameter r used in the mean field computations (Eq. (10)) are estimated using these initial random spin average values. Recall that, the parameter r is introduced in the energy function formulation (Eq.

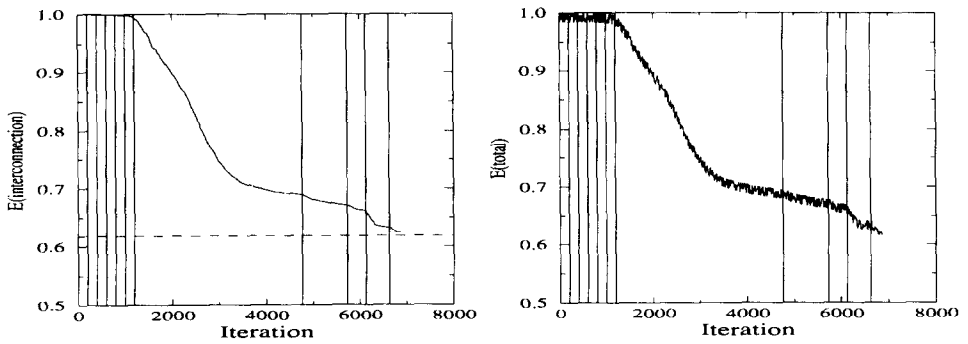
(8)) in order to determine a balance between the two optimization objectives of the circuit partitioning problem. Hence, in the mean field computations (Eq. (10)), the parameter r determines a balance between the terms

$$\phi_{ik}^C = -\sum_{n \in N_i} w_n \prod_{j \in n, j \neq i} (1 - v_{jk}) \quad \text{and} \quad \phi_{ik}^B = -\sum_{j \neq i}^{|C|} v_{jk} w_i w_j$$

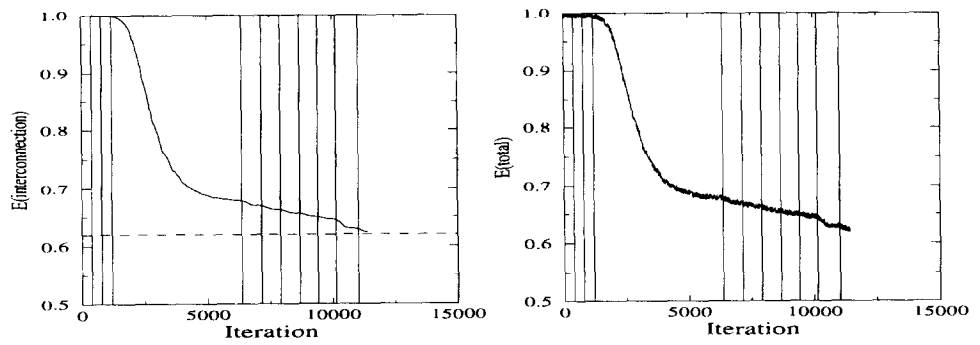
where $\phi_{ik} = \phi_{ik}^C + r \times \phi_{ik}^B$. We compute the averages $\langle \phi_{ik}^C \rangle = (\sum_{i=1}^{|C|} \sum_{k=1}^{K} \phi_{ik}^C)/(|C|K)$ and $\langle \phi_{ik}^B \rangle = (\sum_{i=1}^{|C|} \sum_{k=1}^{K} \phi_{ik}^B)/(|C|K)$ of these two terms using the initial v_{ik} values and compute r as $r = 8\langle \phi_{ik}^C \rangle / \langle \phi_{ik}^B \rangle$. Our experiments show that computing r using this method is sufficient for obtaining balanced partitions.

Selection of T_0 is crucial for obtaining good quality solutions. In previous applications of MFA [12,16], it is experimentally observed that spin averages tend to converge at a critical temperature. It is suitable to choose T_0 close to this critical temperature. Although there are some methods proposed for the estimation of critical temperature [12,16] we prefer an experimental way for computing T_0 which is easy to implement and successful as the results of the experiments indicate. After the parameter r is fixed, average mean field $\langle \phi_{ik} \rangle = (\sum_{i=1}^{|C|} \sum_{k=1}^{K} \phi_{ik})/(|C|K)$ is computed using initial v_{ik} values. Then, T_0 is computed as $T_0 = c\langle \phi_{ik} \rangle / K$. Our experiments indicate that it is suitable to choose the parameter c as 100 for geometric hypergraphs, and as 60 for random hypergraphs. Note that, T_0 is inversely proportional to the number of partitions (K) which is also observed for the critical temperature formulations presented in the other implementations of MFA [12,16].

After the spin averages (v_{ik} values) and the parameters T_0 and r are initialized, the cooling schedule of the algorithm proceeds as follows. At each temperature a random sequence is generated for the spins that are not converged yet. Then, averages of these spins are updated iteratively according to this random sequence. The number of iterations with energy decrease ($-\Delta E$) less than ϵ is counted in each random sequence, where ϵ is chosen as 0.1 and 0.001 for random and geometric hypergraphs, respectively. If this count is more than 90% of the number of unconverged spins, temperature is decreased according to $T = \alpha \times T$ where α is the parameter for adjusting the rate of the cooling. Average spin values are tested for convergence at the end of each random sequence. If one of the v_{ik} terms of the spin average vector $\mathbf{V}_i$ is greater than 0.999, it is assumed that spin i is converged to state k and its average is not updated in the future iterations. Cooling process is realized in two phases; slow cooling followed by fast cooling. In the slow cooling phase, temperature is decreased using $\alpha = 0.95$ until T is less than $T_0/1.5$. Then, in the fast cooling phase, α is set to 0.7 and cooling is continued until T is less than $T_0/5.0$. At the end of this cooling process, maximum element in each spin average vector is set to 1 and all other spin average values are set to 0. Then, the result is decoded as described in Section 3.1, and the resulting partitioning is found. Note that, all parameters used in this implementation are either constants or found automatically except the parameters c and ϵ. The parameters c and ϵ are also constants for each family of hypergraphs but differ for random and geometric hypergraphs.



(a)



(b)

Fig. 4. Evolution of interconnection and total energy terms with iterations for partitioning two random hypergraphs ((a) $|C| = 200$, $|N| = 200$, (b) $|C| = 400$, $|N| = 400$) into $K = 8$ partitions.

Fig. 4 illustrates the evolution of interconnection and total energy terms (E_C and E, respectively) with MFA iterations for partitioning two random hypergraphs ((a) $|C| = 200$, $|N| = 200$, (b) $|C| = 400$, $|N| = 400$) into $K = 8$ partitions. Fig. 4 is constructed by computing the E_C (Eq. (6)) and E (Eq. (8)) terms at each 10 MFA iterations for the given two circuit partitioning problem instances. The displayed energy values are normalized with respect to the initial energy values. The vertical solid lines in each curve denote the temperature changes according to the cooling schedule. As is seen in Fig. 4, interconnection energy (E_C) monotonically decreases as the iterations proceed. The oscillatory decrease of the total energy term (E) is due to the balance term (E_B given in Eq. (7)) oscillations superimposed on the monotonically decreasing E_C term. As is seen in Fig. 4, the major decrease in the energy terms occur during a single temperature which corresponds to the critical temperature mentioned earlier. Note that, the number of iterations performed during the critical temperature is substantially greater than

the numbers of iterations performed during other temperatures. The decrease in the energy terms during the last two temperatures demonstrate the merits of the fast cooling phase mentioned earlier. The horizontal dashed lines in the interconnection energy curves denote the normalized actual interconnection costs of the solutions decoded from the final spin matrices. Note that, interconnection energy terms (E_C) converge to the final actual interconnection costs confirming the correctness of the proposed formulation (Eq. (6)).

4.2. Simulated annealing implementation

In simulated annealing, starting from a randomly chosen initial configuration, configuration space is searched for the best solution using a probabilistic hill climbing algorithm [8]. A configuration of the circuit partitioning problem is a partitioning of the circuit to K partitions. In order to search the configuration space, neighborhood of a configuration must be defined. For the implementation in this work, neighborhood of a configuration consists of all configurations which result from moving one cell of the circuit from the partition with maximum size to any other partition. At each iteration of the simulated annealing algorithm, one of the possible moves is chosen randomly as a candidate move. Then, the resulting decrease in the total interconnection cost caused by the candidate move is calculated without changing the configuration. If the candidate move decreases the interconnection cost, it is realized. If it increases the interconnection cost, then it is realized with a probability which decreases with the amount of increase in the total interconnection cost. Acceptance probabilities of the moves that increase the cost are controlled with a temperature parameter T which is decreased using an annealing schedule. Hence, as the annealing proceeds acceptance probabilities of uphill moves decrease. An automatic cooling schedule is used in the implementation of the SA algorithm [12]. Note that, the SA algorithm implemented in this work implicitly achieves the balance among the sizes of the partitions by selecting the neighbor configurations as defined above. This increases computationally efficiency of the algorithm but decreases its flexibility since the amount of imbalance among the partitions can not be controlled.

4.3. Kernighan-Lin implementation

Kernighan-Lin (KL) heuristic is implemented efficiently as described by Fiduccia and Mattheyses [3]. In order to apply the KL heuristic to K-way partitioning a two phase approach is used which consists of recursive bisection and pairwise min-cut phases. In recursive bisection phase, the circuit is recursively partitioned into two partitions until K partitions are obtained. Then, in the pairwise min-cut phase, total interconnection cost is iteratively minimized by executing KL heuristic between each pair of partitions until no improvement can be achieved. In the KL heuristic, balance among partitions is maintained implicitly by the algorithm. Cell moves causing intolerable imbalances are not considered. In the implementation used in this work a move which increases or decreases the size of a partition more

Table 1
Interconnection cost averages (and standard deviations) normalized with respect to the MFA heuristic and percent imbalance ratio averages (and standard deviations) of the solutions found by the MFA, KL, and SA heuristics for random hypergraphs

Problem Size			Average Interconnection Cost			Average Percent Imbalance Ratio		
$\|C\|$	$\|N\|$	K	MFA	KL	SA	MFA	KL	SA
200	100	4	1.00 (0.08)	1.04 (0.09)	1.00 (0.08)	8.08 (3.60)	7.43 (1.00)	1.58 (0.70)
200	200	4	1.00 (0.04)	1.03 (0.04)	1.01 (0.04)	5.55 (2.13)	8.32 (0.68)	1.73 (0.70)
200	400	4	1.00 (0.02)	1.03 (0.02)	1.02 (0.02)	7.72 (2.65)	8.52 (0.43)	1.62 (0.77)
400	200	4	1.00 (0.06)	1.03 (0.06)	0.98 (0.06)	13.16 (5.50)	7.77 (1.42)	0.87 (0.36)
400	400	4	1.00 (0.03)	1.02 (0.03)	0.99 (0.03)	4.29 (1.88)	9.15 (0.50)	0.93 (0.41)
400	800	4	1.00 (0.03)	1.03 (0.03)	1.01 (0.03)	6.90 (1.80)	9.26 (0.22)	0.83 (0.37)
200	100	8	1.00 (0.08)	1.07 (0.09)	1.00 (0.08)	4.35 (1.53)	7.65 (0.48)	3.81 (1.03)
200	200	8	1.00 (0.05)	1.05 (0.05)	1.02 (0.05)	6.09 (1.75)	7.71 (0.50)	3.69 (1.24)
200	400	8	1.00 (0.03)	1.04 (0.03)	1.02 (0.03)	8.00 (1.62)	7.99 (0.45)	3.74 (1.25)
400	200	8	1.00 (0.07)	1.06 (0.07)	1.00 (0.07)	3.95 (1.16)	8.60 (0.27)	1.81 (0.56)
400	400	8	1.00 (0.03)	1.05 (0.03)	1.01 (0.03)	4.91 (1.37)	8.81 (0.31)	1.82 (0.58)
400	800	8	1.00 (0.03)	1.05 (0.03)	1.03 (0.03)	7.45 (1.44)	9.11 (0.60)	1.78 (0.60)
200	100	16	1.00 (0.09)	1.10 (0.10)	0.98 (0.09)	5.25 (1.55)	6.16 (0.44)	7.34 (2.13)
200	200	16	1.00 (0.05)	1.06 (0.07)	0.98 (0.04)	6.43 (1.60)	6.06 (0.95)	7.89 (2.10)
200	400	16	1.00 (0.07)	1.02 (0.03)	0.98 (0.03)	6.65 (1.57)	6.33 (0.30)	8.22 (1.94)
400	200	16	1.00 (0.07)	1.04 (0.08)	0.97 (0.07)	3.51 (1.04)	7.96 (0.38)	3.73 (1.01)
400	400	16	1.00 (0.03)	1.03 (0.03)	0.99 (0.03)	5.21 (1.21)	8.25 (0.87)	3.79 (1.07)
400	800	16	1.00 (0.03)	1.02 (0.03)	1.00 (0.03)	6.77 (1.42)	8.34 (0.67)	3.84 (0.99)

than 10% of the size of a perfectly balanced partition is considered as causing intolerable imbalance.

4.4. Experimental results

In this section, performance of the proposed MFA algorithm is experimentally evaluated in comparison with the Kernighan-Lin (KL) and the simulated annealing (SA) algorithms. These heuristics are experimented with a large number of randomly generated circuit partitioning problem instances.

Six different types of random hypergraphs and six different types of geometric hypergraphs are generated with $|C| = 200$, $|C| = 400$ and $|N| = |C|/2$, $|N| = |C|$ and $|N| = 2|C|$. For each type of hypergraph 10 different random instances are generated. That is, a total of 120 different hypergraph instances are generated randomly. In each hypergraph instance, maximum cell weight (W_c) and maximum net weight (W_n) are both selected as 4. The maximum net size (s_{max}) in random hypergraph instances and the average net size (s_{avg}) in geometric hypergraph instances are selected as 16 and 8, respectively. A total of $3 \times 120 = 360$ circuit partitioning problem instances are constructed by using these hypergraph instances and selecting the number of partitions as $K = 4$, $K = 8$, and $K = 16$.

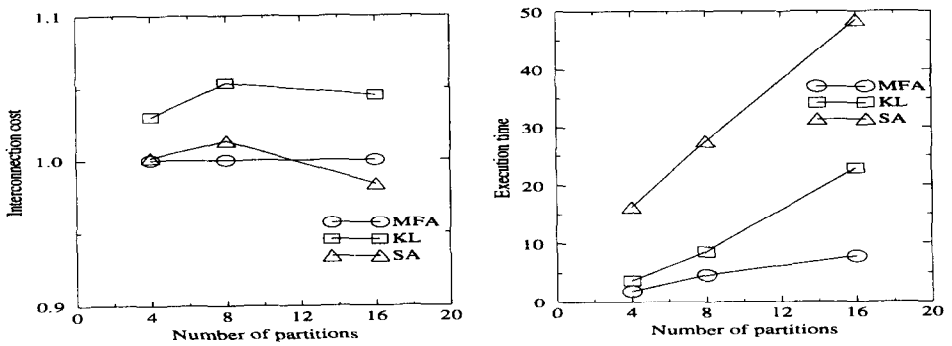
Tables 1–3 and Fig. 5 illustrate the performance results of the MFA, KL and SA heuristics for the circuit partitioning problem instances constructed using random and geometric hypergraphs. In Tables 1–3, $|C|$ (number of cells) and $|N|$

Table 2
Interconnection cost averages (and standard deviations) normalized with respect to the MFA heuristic and percent imbalance ratio averages (and standard deviations) of the solutions found by the MFA, KL, and SA heuristics for geometric hypergraphs

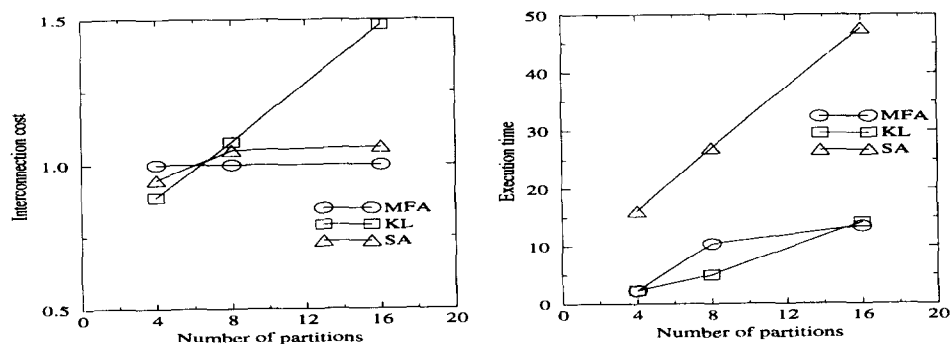
Problem Size			Average Interconnection Cost			Average Percent Imbalance Ratio		
$\|C\|$	$\|N\|$	K	MFA	KL	SA	MFA	KL	SA
200	100	4	1.00 (0.21)	1.02 (0.26)	1.00 (0.22)	8.09 (3.91)	5.55 (1.64)	1.85 (0.71)
200	200	4	1.00 (0.21)	0.94 (0.17)	0.98 (0.15)	6.65 (4.66)	6.41 (1.49)	1.62 (0.71)
200	400	4	1.00 (0.17)	0.88 (0.12)	0.98 (0.13)	7.63 (4.41)	6.36 (1.61)	1.46 (0.73)
400	200	4	1.00 (0.24)	1.02 (0.22)	0.98 (0.20)	8.09 (3.71)	5.77 (2.10)	0.77 (0.36)
400	400	4	1.00 (0.15)	0.77 (0.11)	0.88 (0.14)	5.46 (3.35)	6.94 (1.70)	0.82 (0.40)
400	800	4	1.00 (0.17)	0.69 (0.10)	0.86 (0.13)	7.03 (4.81)	7.32 (1.56)	0.73 (0.37)
200	100	8	1.00 (0.14)	1.29 (0.31)	1.09 (0.17)	6.08 (2.59)	7.13 (1.06)	3.14 (1.06)
200	200	8	1.00 (0.10)	1.13 (0.20)	1.05 (0.11)	5.16 (2.28)	7.32 (0.91)	3.48 (1.36)
200	400	8	1.00 (0.07)	1.09 (0.14)	1.07 (0.09)	6.42 (2.62)	7.28 (0.91)	3.64 (1.17)
400	200	8	1.00 (0.17)	1.10 (0.17)	1.02 (0.17)	7.70 (3.68)	7.29 (1.37)	1.83 (0.58)
400	400	8	1.00 (0.11)	0.96 (0.09)	1.05 (0.10)	4.97 (2.32)	8.46 (1.02)	1.72 (0.61)
400	800	8	1.00 (0.11)	0.90 (0.08)	1.02 (0.09)	6.01 (3.03)	8.53 (1.15)	1.80 (0.61)
200	100	16	1.00 (0.12)	2.23 (0.65)	1.13 (0.19)	6.16 (1.63)	6.38 (1.09)	7.68 (2.53)
200	200	16	1.00 (0.09)	1.60 (0.21)	1.03 (0.11)	6.68 (1.56)	6.55 (1.02)	7.08 (2.21)
200	400	16	1.00 (0.07)	1.68 (0.38)	1.01 (0.07)	7.62 (1.77)	6.21 (1.40)	7.72 (2.28)
400	200	16	1.00 (0.13)	1.21 (0.17)	1.10 (0.13)	6.10 (2.13)	8.26 (0.95)	3.47 (1.04)
400	400	16	1.00 (0.07)	1.12 (0.13)	1.06 (0.08)	5.51 (1.44)	8.64 (1.12)	3.47 (1.13)
400	800	16	1.00 (0.07)	1.05 (0.11)	1.04 (0.06)	6.50 (1.79)	8.55 (1.16)	3.75 (1.19)

Table 3
Execution time averages (in seconds) of the MFA, KL, and SA heuristics for random and geometric hypergraphs

			Average Execution Time					
Problem Size			Random Hypergraphs			Geometric Hypergraphs		
$\|C\|$	$\|N\|$	K	MFA	KL	SA	MFA	KL	SA
200	100	4	0.76	1.11	7.64	1.76	0.86	7.85
200	200	4	1.13	1.78	9.44	1.53	1.35	8.75
200	400	4	2.12	3.47	10.94	2.07	2.23	11.20
400	200	4	1.40	2.51	18.52	2.71	1.76	18.59
400	400	4	2.11	4.51	22.31	2.15	2.77	21.62
400	800	4	3.60	8.80	28.46	3.32	4.91	28.54
200	100	8	1.65	3.24	14.16	5.44	2.44	13.93
200	200	8	2.70	5.07	16.22	8.12	3.52	15.98
200	400	8	4.79	8.83	18.65	16.36	5.75	18.81
400	200	8	2.84	5.29	33.27	7.22	3.61	32.60
400	400	8	5.30	9.88	37.45	9.63	5.49	36.41
400	800	8	10.38	18.96	45.18	15.51	8.99	44.53
200	100	16	2.90	12.10	25.38	7.37	8.46	22.52
200	200	16	4.52	15.60	28.79	9.31	11.75	27.76
200	400	16	7.34	24.47	33.65	17.63	15.23	35.07
400	200	16	5.22	14.58	61.02	11.41	10.62	55.69
400	400	16	9.90	25.68	65.95	14.09	15.66	62.90
400	800	16	16.47	44.55	76.53	20.93	23.21	82.03



(a)



(b)

Fig. 5. Interconnection cost (normalized with respect to the MFA heuristic) and execution time (in seconds) averages of the MFA, KL and SA heuristics for (a) random and (b) geometric hypergraphs for different number of partitions.

(number of nets) determine the type of the 10 distinct hypergraph instances experimented for collecting the data in each row. Each algorithm is executed 10 times for each problem instance starting from different, randomly chosen initial configurations. Each entry in Tables 1–3 illustrates the overall average (and standard deviation) of the results of $10 \times 10 = 100$ executions of a particular algorithm for partitioning 10 different hypergraph instances of the same type into K partitions.

Tables 1 and 2 illustrate the quality of the solutions obtained by the MFA, KL and SA heuristics for random and geometric hypergraphs, respectively. Total interconnection cost averages (and standard deviations) of the solutions are normalized with respect to the results of the MFA heuristic. Percent imbalance ratio averages (and standard deviations) of the solutions displayed in these tables are computed using $100 \times (|P|_{max} - |P|_{min})/(2|P|_{avg})$ where $|P|_{max}$, $|P|_{min}$

and $|P|_{avg} = (\sum_{k=1}^{K} |P_k|)/K$ denote the maximum, the minimum and the average partition sizes. Table 3 displays the execution time averages of the MFA, KL and SA heuristics on a SUN Sparc Server 490, measured in seconds. Fig. 5 illustrates the change in the interconnection cost and execution time averages of the MFA, KL and SA heuristics with respect to the number of partitions for random and geometric hypergraphs.

As is mentioned earlier, circuit partitioning has two different optimization objectives: interconnection cost and imbalance cost. Hence, the quality of a solution of a particular circuit partitioning problem instance has two components: interconnection quality and balance quality. For random hypergraphs, as is seen in Fig. 5(a) and Table 1, interconnection qualities of the solutions found by the MFA and the SA heuristics are comparable and both better than the interconnection qualities of the solutions found by the KL heuristic. For geometric hypergraphs, as is seen in Fig. 5(b) and Table 2, the KL and SA heuristics produce solutions with slightly better interconnection qualities compared with those of the MFA heuristic for $K = 4$. However, for $K = 8$ and $K = 16$ interconnection qualities of the solutions obtained by the MFA heuristic are better than those of the KL and SA heuristics.

As is seen in Tables 1 and 2, the balance qualities of the solutions found by the MFA algorithm are comparable with those of the KL heuristic. Note that, the balance qualities of the solutions found by the SA heuristic are superior to those of the MFA and KL heuristics. This is due to the implementation of the SA heuristic (explained in Section 4.2) which compels balanced partitionings.

As is seen in Fig. 5 and Table 3, the MFA and KL heuristics are significantly faster than the SA heuristic. For the case of random hypergraphs, the MFA heuristic is always faster than the KL heuristic. For geometric hypergraphs, execution time averages of the MFA and KL heuristics are comparable for $K = 4$ and $K = 16$, whereas for $K = 8$ the KL heuristic is faster than the MFA heuristic. Note that, as the number of partitions increase, both the solution quality and the speed advantage of the MFA heuristic increases in comparison with those of the KL heuristic. The relative increase in the speed of the MFA heuristic is also observed in the literature [16] for the case of graph partitioning problem.

5. Conclusion

In this paper, a mean field annealing (MFA) algorithm is proposed for the circuit partitioning problem using the net-cut model. An efficient implementation scheme is also developed for the proposed algorithm. The proposed implementation scheme decreases the complexity of a single MFA iteration by asymptotical factors. The performance of the proposed algorithm is experimentally evaluated in comparison with two well-known heuristics (simulated annealing (SA) and Kernighan-Lin (KL)) for a large number of randomly generated circuit partitioning problem instances. The qualities of the solutions obtained by the MFA heuristic are comparable with those of the SA heuristic. In general, the MFA heuristic

produces better solutions than the KL heuristic. The proposed MFA algorithm is significantly faster than the SA algorithm. In general, the MFA algorithm is also faster than the KL algorithm. It is also observed that, as the number of partitions increase, the solution quality and the speed advantage of the proposed MFA heuristic increases in comparison with those of the KL heuristic.

References

[1] T. Bultan and C. Aykanat, A new mapping heuristic based on mean field annealing, *J. Parallel Distributed Comput.* 16 (1992) 292–305.

[2] T. Bultan and C. Aykanat, Circuit partitioning using parallel mean field annealing algorithms, in *Proc. 3rd IEEE Symp. on Parallel Processing* (1991) 534–541.

[3] C.M. Fiduccia and R.M. Mattheyses, A linear-time heuristic for improving network partitions, in *Proc. Design Automat. Conf.* (1982) 175–181.

[4] L. Gislen, C. Peterson and B. Soderberg, Complex scheduling with Potts neural networks, *Neural Computat.* 4 (1992) 805–831.

[5] L. Herault and J. Niez, Neural networks and graph K-partitioning, *Complex Syst.* 3 (1989) 531–575.

[6] J.J. Hopfield and D.W. Tank, 'Neural' computation of decisions in optimization problems, *Biol Cybern.* 52 (1985) 141–152.

[7] B.W. Kernighan and S. Lin, An efficient heuristic procedure for partitioning graphs, *Bell Syst. Tech.* J. 49 (1970) 291–307.

[8] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, Optimization by simulated annealing, *Science* 220 (1983) 671–680.

[9] K.C. Lee, N. Funabiki and Y. Takefuji, A parallel improvement algorithm for the bipartite subgraph problem, *IEEE Trans. Neural Networks* 3 (1) (1992) 139–145.

[10] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout* (Wiley, 1990).

[11] M. Ohlsson, C. Peterson and B. Soderberg, Neural networks for optimization problems with inequality constraints-the knapsack problem, *Neural Computat.* 5(2) (1993) 331–339.

[12] C. Peterson and B. Soderberg, A new method for mapping optimization problems onto neural networks, *Int. J. Neural Syst.* 1 (3) (1989) 3–22.

[13] D.G. Schweikert and B.W. Kernighan, A proper model for the partitioning of electrical circuits, in *Proc. 9th Design Automat. Workshop* (1979) 57–62.

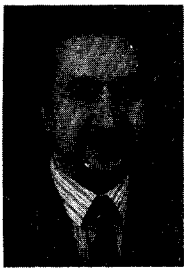
[14] Y. Takefuji, K.C. Lee and H. Aiso, An artificial maximum neural network: a winner-take-all neuron model forcing the state of the system in a solution domain, *Biol. Cybernet.* 67 (1992) 243–251.

[15] D.E. Van den Bout and T.K. Miller, Improving the performance of the Hopfield-Tank neural network through normalization and annealing, *Biol. Cybernet.* 62 (1989) 129–139.

[16] D.E. Van den Bout and T.K. Miller, Graph partitioning using annealed neural networks, *IEEE Trans. Neural Networks* 1 (2) (1990) 192–203.

[17] J.S. Yih and P. Mazumder, A neural network design for circuit partitioning, *IEEE Trans. Computer-Aided Design* 9 (1990) 1265–1271.

Tevfik Bultan received the B.S. degree in electrical engineering from the Middle East Technical University, Ankara, Turkey, and the M.S. degree in computer engineering and information science from the Bilkent University, Ankara, Turkey, in 1989 and 1992, respectively. He is currently working toward the Ph.D. degree in the Department of Computer Science at University of Maryland, College Park. His research interests are in parallel processing and non-deterministic optimization techniques.



Cevdet Aykanat received the B.S. and M.S. degrees from the Middle East Technical University, Ankara, Turkey, and the Ph.D. degree from The Ohio State University, Columbus, all in electrical engineering. He was a Fulbright scholar during his Ph.D. studies. He worked at the Intel Supercomputer Systems Division, Beaverton, as a research associate. Since October 1988 he has been with the Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey, where he is currently an associate professor. His research interests include parallel computer architectures, parallel algorithms, applied parallel computing, neural network algorithms, and fault-tolerant computing.