We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter 1

Mean Field Annealing Based Techniques for Resolving VLSI Automatic Design Problems

Gholam Reza Karimi and Ahmad Azizi Verki

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/52092

1. Introduction

The neuro-computing approaches based on Hopfield model were successfully applied to various combinatorial optimization problems such as the traveling salesman problem [1-3], scheduling problem [4], mapping problem [5], knapsack problem [6,7], communication routing problem [8], graph partitioning problem [2,9,10], graph layout problem [11], circuit partitioning problem [12,13].

MFA, as a neuro-computing technique, is applied for solving combinatorial optimization problems [1,2,4-8,10,13] , cell placement problem [14].

MFA combines the *annealing* notion of SA approach with the collective computation property of *Hopfield neural network*s to obtain optimal solution for np-hard problems.

We begin our study with the review of basic concepts of MFA techniques and describe the applied use of this technique to solve the problems in high speed Integrated Circuits (IC) design and in addition we applied a modified MFA algorithm to solve VLSI relocation problem [15].

1.1. Annealing

Annealing is a mechanical process in which material is slowly cooled allowing the molecules to arrange themselves in such a way that the material is less strained thereby making it more stable.

If materials such as glass or metal are cooled too quickly its constituent molecules will be under high stress lending it to failure (breaking) if further thermal or physical shocks are encountered. Slowing the cooling of the material allows each molecule to move into a place it feels most comfortable, i.e., less stress.

INTECH
open science | open minds

© 2012 Karimi and Verki, licensee InTech. This is an open access chapter distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

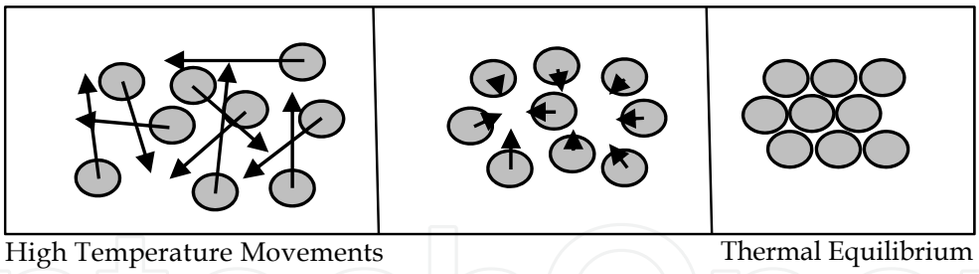


Figure 1. Molecules movement at the cooling process

As the material is kept at a high temperature the molecules are able to move around quite freely thus reducing stress on a large scale, indeed if the material is made too hot it will move into the liquid state allowing free movement of the molecules. As the material is cooled the molecules are not able to move around as freely but still move limited distances reducing stress in regional areas. The result is a material with significantly less internal stress and resistant to failure due to external shock.

The statistic mechanic is a domain in physics that describes the process of slow cooling of *Hamiltonian Ising* for particles or spins with high degree of freedom until they accede on their equilibrium states. The particles that are cooling, on solid state, provide a framework to characteristics improvisation of intricate and large systems. Now this idea is stated inside optimization algorithms to resolve various cases of problems.

1.2. Hopfield Neural Network (HNN)

The Hopfield Network is a fully connected network of simple processing units, V_i , with numerically weighted symmetric connections, T_{ij}, between units V_i and V_j. processing units have states (either discrete in {0, 1}, or continuous in [0, 1] depending on whether the discrete or the continuous version of the network is being considered). Each processing unit performs simple and identical computations which generally involve summing weighted inputs to unit, applying an internal transfer function, and changing state if necessary. The power of the Hopfield model lies in the connections between units and the weights of these connections [16]. An Energy function was defined by Hopfield on the states of the network (values of all units). The energy function, *E*, in its simplest form is:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}V_jV_i + \sum_{i=1}^{N} V_iI_i \quad (1)$$

WhereV_i denotes the current state (value) of the *ith* neuron and I_i denotes its bias. Hopfield utilized the fact that the $E(\vec{V_i})$ is a *Liapunov* function (bounded from below) to show that, from any starting state, the network would always converge to some energy function minimum upon applying a sequence of asynchronous local state updates (that locally reduce energy).

To solve any particular problem, first a decision must be made on how to set the network parameters *T* and *I*, so that minimization of the problem objective function and enforces

satisfaction of the problem constraints; this process is termed 'mapping' the problem onto the network. Hopfield gives the motion equation of the *ith* neuron:

$$\frac{dU_i}{dt} = \frac{U_i}{\tau} - \frac{\partial E(\vec{V_i})}{\partial V_i}, U_i = \frac{\partial E_H(\vec{V_i})}{\partial V_i} \quad (2)$$

Where E is energy function in term of V_i and E_H is *Hopfield term* of energy function. Totally Eq. 2.is motion (updating) equation of state of neurons and its output is U_i.Usually a simple nondecreasing monatomic output function in term of U_i like $g(U_i)$ is applied torelate U_i to the states. Typically this function is a step function or a hyperbolic tangent function. τ is a constant number as the weighting factor of U_i.Thereforea Hopfield Neural Network minimizes a cost function that is encoded with its weights by implementation of *gradient descent.* For more details see [16]

2. MFA technique

As it mentioned before, MFA merges collective computation and annealing properties of Hopfield neural Networks and SA, 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 discrete variables (spins or neurons) can be decoded as a solution to the problem. In fact the space of problem is mapped to the space of MFA variables (spins) and there will be a one-to-one relation between two spaces. This is called *encoding*. Then, an energy function is formulated in term of spins with a structure that is based on essence of problem whose global minimum value corresponds to an optimum solution of the problem. 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 MFA technique to a problem can be summarized as follows:

1. Choose a representation plan 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. That means 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 to 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 formulated energy function. Derive equations are used for updating averages (expected values) of spins.
4. cooling schedule
5. Set suitable parameters of the energy function and the cooling schedule to obtain efficient algorithm.

These main steps are same for various types of optimization problems and are explained at the following sections.

2.1. 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 by0 and 1, whereas in Potts model they can be in one of the K states and the configuration of the problem determines which one has to be used.

For *K*-state *Potts* model with n_S spins, the states of spins are represented using n_S K-dimensional vectors.

$$S_i = [S_{i1}, \dots, S_{ik}, \dots, S_{iK}] \text{ for } 1 \le i \le n_S \tag{3}$$

Just one of the components of S_i is 1 and the others are 0. That means *ith* spin must be at one of the K- states.

$$\sum_{k=1}^{K} S_{ik} = 1 \; for \; 1 \le i \le n_S \tag{4}$$

For encoding of VLSI circuit design problem, for example, each spin vector corresponds to a cell in the circuit or a module in the placement. Hence, number of spin vectors is equal to the number of cells or modules; n_C. Dimension *K* of the spin vectors is equal to the number of empty part of overall circuit space or empty spaces of the placement. That means we can divide the circuit space (chip area or die surface)to *K* parts and fill every part just by one and only one of the circuit elements [12, 13]. Therefore when a spin is assigned in *kth* state that means its corresponding cell or module (circuit element) is placed on *kth* space or part of circuit or placement.

2.2. 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 $V_i = \langle S_i \rangle$ of each spin vector S_i is computed and iteratively updated until the system stabilizes at some fixed point.

$V_i = \langle S_i \rangle$ *for* $1 \le i \le n_C$ and $V_i = [v_{i1}, \dots, v_{ik}, \dots v_{iK}]$ So:

$$[v_{i1}, \dots, v_{ik}, \dots v_{iK}] = [\langle S_{i1} \rangle, \dots \langle S_{ik} \rangle, \dots, \langle S_{iK} \rangle] \; for \; 1 \le i \le n_C \tag{5}$$

v_{ik} is probability of finding spin *i* at state *k* and 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:

$$\sum_{k=1}^{K} v_{ik} = 1 \; for \; 1 \le i \le n_C \tag{6}$$

This constraint guarantees that each Potts spin S_i is in one of the *K* states at a time, and each cell is assigned to only one position for encoded configuration of the problem. In order to construct an energy function it is helpful to associate the following meaning to the values v_{ik}, for example:

$$v_{ik} = P\{\text{spin } i \text{ is in position } k\} \; for \; 1 \leq \text{i} \leq \text{n}_C \text{ , } 1 \leq \text{k} \leq \text{K(P\{\} is probability function)} \quad (7)$$

$v_{ik}$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 $V_i = S_i$.

Locating spin i at stat k relevant to type of target problem has some costs and actually energy function calculates these costs. Example given, for circuit partitioning problem, utilizing the interconnection cost and the wire-length cost for VLSI placement problem are common cost functions and are used to formulating energy function of these target problems [12-14].

The interconnection cost is represented by E_c that for the circuit is total length of internal connections between circuit components or the cost of the connections among the circuit partitions. It is clear that if all of the circuit elements are located in one place and overlaps together, the interconnection cost (total wire length) becomes 0 and it is not acceptable. This is what we mean *illogical minimization* of interconnection cost energy function. So another term of the energy function must be applied for penalizing illogical minimization of first cost function. This term is represented by E_p. For example, this term is imbalanced partitioning for circuit partitioning problem and overlap between modules for VLSI placement problem [13, 14].The total energy function, E_t, is sum of both terms:

$$E_t = E_c + \alpha \times E_p \quad (8)$$

Where α parameter is introduced to maintain a balance between the two opposite terms of total energy function.

2.3. Derivation of the mean field theory equations

Mean field theory equations, needed to minimize the total energy function E_t, can be derived as follow:

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

The quantity ϕ_{ik} represents the *kth* element of the *mean field vector* effecting on spin *i*. Using the mean field values, average spin values, v_{ik}, can be updated.

$$v_{ik} = \frac{e^{\phi_{ik}/T}}{\sum_{n=1}^{K} e^{\phi_{in}/T}} \text{for} 1 \leq \text{ i } \leq n_C \text{ , } 1 \leq \text{ } k \leq K \quad (10)$$

Where T is the temperature parameter which is used the relax the system iteratively and is managed with a cooling schedule program.

2.4. Energy difference and cooling schedule

A teach iteration of algorithm, the mean field vector effecting on a randomly selected spin is computed. Then, spin average vector is updated. 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 the total energy 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 by a decreasing factor and the iterative process restarted again with new temperature. To reduce the complexity of energy difference computation an efficient scheme could be used.

$$\Delta E_{ik} = \phi_{ik}\,\Delta v_{ik}\ so\ \Delta E = \sum_{k=1}^{K} \phi_{ik}\,\Delta v_{ik} \text{where } \Delta v_{ik} = v_{ik}(new) - v_{ik}\,(old) \tag{11}$$

Depending to complexity of problem, the cooling program could be in one stage or more stages in order to reach faster and better result. In some problems like circuit partitioning problem the applied cooling schedule is simply in one stage (t_f is decreasing factor):

$$T_{new} = T_{old} \times t_f\ for\ 0 < t_f < 1 \tag{12}$$

Actually cooling schedule controls amount of acceptable cost increasing moves and the efficiency of the algorithm. Clearly for very large temperatures almost any change will be accepted while as the temperature is reduced the chance that a positive cost change will also be accepted is reduced.

2.5. Total MFA algorithm

The total format of MFA for various kind of problem is represented as:

1. Get the initial temperature T_0, and set $T = T_0$
2. Initialize the spin averages $V=[v_{11},\ldots, v_{ik},\ldots,v_{n_cK}]$
3. While temperature T is in the cooling range DO:
3.1. While E is decreasing DO:
3.1.1. Select the *ith* spin randomly.
3.1.2. Compute mean field vector corresponding to the *ith* spin: $\phi_{ik} = -\frac{\partial E(V)}{\partial v_{ik}}$
3.1.3. Compute the summation: $\sum_{n=1}^{K} e^{\phi_{in}/T}$
3.1.4. Compute new spin average vector: $v_{ik}(new) = e^{\phi_{ik}/T} / \sum_{l=1}^{K} e^{\phi_{il}/T}$
3.1.5. Compute new spin average vector: $\Delta E = \sum_{k=1}^{K} \phi_{ik}\ (v_{ik}(new) - v_{ik})$
3.1.6. Update the spin average vector: $v_{ik}(new) = v_{ik}$
3.2. Decrease the temperature: $T = T \times t_f$

Inside the algorithm some notes must be considered. Selection of initial temperatures is crucial for obtaining good quality solutions. Typically spin averages initialize with an equal values plus a small disturbing part that is randomly valued but this is not an eternal rule. Adding this disturbing part causes the spins exit their stable states and their movement starts. Selecting balance factors in energy function has important role for efficiency of the algorithm.

3. VLSI Relocation problem using MFA technique

In modern VLSI physical design, Engineering Change Order (ECO) optimization methods are used to mitigate model placement problems such as hot spots and thermal dissipation that are identified at a given layout at post-routing analysis that is an evaluation stage after placement stage. The relocation problem is defined as adding an additional module to a model placement in order to solve problems at a manner that similarity of the resultant placement to the model placement is kept.

Our presented MFA-based technique is modified form which was applied for cell placement problem in [14] by adding some considerations relating to particular characteristics of the local relocation problem.

3.1. Cell placement problem

Placement is the process of determining the locations of circuit devices on a die surface. ItisanimportantstageintheVLSIdesignflowbecauseitaffectsroutability, performance, heat distribution, and to a less extent, power consumption of a design.

Traditionally, it is applied after the logic synthesis stage and before the routing stage. Since the advent of deep submicron process technology around mid-1990, interconnect delay, which is largely determined by placement, has become the dominating component of circuit delay. As a result, placement information is essential even in early design stages to achieve better circuit performance.

The circuit is presented with a hyper-graph Ω(C, N), that consists of a set C representing the cells circuit, a cell weight function of the circuit, a hyper-edge set N representing the nets of the $\omega_{cell}: C \to \mathcal{N}$ and a net weight function $\omega_{net}: N \to \mathcal{N}$ where $\mathcal{N}$ represents the set of natural numbers. Space of circuit is a rectangular grid of clusters with P rows and Qcolumns where the cells will be placed.

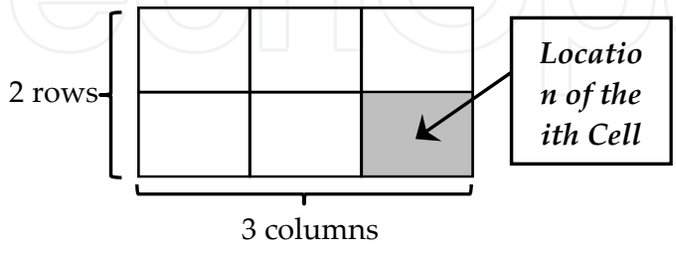


Figure 2. Cell location on spin space configuration

As presented before in the K-state Potts model of S spins, the states of spins re represented using S K-dimensional vectors. To apply MFA technique for cell placement problem the circuit layout space is mapped to a grid space with P rows and Q columns. If the number of

cells be C_L, the number of spins that encode the configuration of problem is C_L $(P \times Q)$-dimensional Potts spins so there would be a total of $|C_L| \times P \times Q$ two-state variables. To decreasing the number of spins that encode the configuration of problem, they are separated to two types: row and column spins. Therefore there would be P row spins and Q column spins and totally $|C_L| \times (P+Q)$ spins[14].For example for a circuit space with 2 rows and 3 columns if the row spin vector of *ith* cell is $v_{ip}^{r} = [0,1]$ and its column spin vector is $v_{iq}^{c} = [0,0,1]$ that means this cell is located at second row and third column of configuration space as Fig. 2.

3.1.1. Energy function formulation

Energy function in the MFA algorithm corresponds to formulation of the cost function of the cell placement problem in terms of spins. Since the MFA algorithm iterates on the expected values of the spins, the expected value of the energy function is formulated. The gradient of the expected value of the energy function is used in the MFA algorithm to compute the new values to update spin vectors in order to minimize the energy function. The applied cost energy for this problem is routing cost energy that is calculated approximately. It is not feasible to calculate the exact routing length for two reasons. Firstly, a feasible placement is not available during the execution of some algorithms; secondly, the computation of the exact routing cost necessitates the execution of the global and the detailed routing phases which are as hard as the placement phase. Commonly used approximations are the *semi-perimeter* method or *Half Perimeter Wire Length* (*HPWL*) method.

Using the expected values of spins, the probability of existence of one or more cells of *nth* net in *pth* row and *qth* column is calculated and applying *HPWL* method routing length cost is obtained. Different weights for row and column routing length costs could be considered.

If the routing cost is used as the only factor in the cost function, the optimum solution is mapping all cells of the circuit to one location in the layout. This placement will reduce the routing cost to zero but obviously it is not feasible. Hence, a term in the energy function is needed which will penalize the placements that put more than one cell to the same location. This term is called the overlap cost. This term is calculated by multiplying the probabilities of being *ith* and *jth* cells in same location. The total energy functionE_t, is:

$$E_t = E_{vrc} + E_{hrc} + \beta \times E_o \tag{13}$$

where E_{vrc}, E_{hrc} and E_o are vertical routing cost, horizontal routing cost and overlap cost respectively.The parameter β is balance factor between routing and overlap cost functions.

3.1.2. Half Perimeter Wire Length (HPWL) method

A very simple and widely used cost function parameter is the interconnect wire length of a placement solution; this can be easily approximated using the bounding box method. This wire length estimation method draws a bounding box around all ports in a given net, half

the perimeter of this box is taken as the net's interconnect length approximation. The *half perimeter wire length* (HPWL) estimation for minimally routed two and three port nets gives an exact value.

3.2. Local relocation using MFA technique

Our method executes local relocation on a model placement where an additional module is added to it for modification with minimum number of displacement. The model placement is a given placement of the circuit that needs modification. MFA based method resolves the problem in less time and hardware in compare to SA-based method. In addition, the runtime of solution is mostly independent of size and complexity of input model placement. Our proposed MFA algorithm is optimized by adding the ability of rotation of modules inside an energy function called *permissible distances preservation energy* that will be defined at section 3.2.6. This in turn allows more options in moving the engaged modules. Finally, a three-phase cooling process governs convergence of problem variables called neurons or spins.

The relocation problem is formulated as follows:

Input: A model placement including a set of modules and a net list or hypergraph representation of circuit, the additional module with its coordinates and the incident nets.

Output: Local relocated placement

Objective: Fast relocation with minimum number of displacements and more similarity

Constraint: No overlap between modules and preservation of permissible distances

There are four classified approaches to the problem of inserting an extra module into a model placement.

i. The additional elements are inserted into unoccupied "whitespace" areas as much as possible.
ii. Before additional logic elements are inserted, an effort is made to predict the amount of whitespace area required; this whitespace is distributed over the chip. If the prediction is accurate (or conservative), the added elements can be placed within the available space.
iii. The third approach is to simply insert or resize the required logic elements, and begin the optimization process from scratch.
iv. The fourth approach is to insert additional logic elements without considering overlaps.

Our approach matched the fourth approach above. The MFA relocation algorithm removes overlaps by moving or rotating modules. Note that all of the movements and rotations must observe some permissible distances that will be explained in the following sections. Feasibility of problem depends on topology of placement and similarity. It is clear that

selecting a big part of model placement as the relocation range may cause a feasible solution but causes more unsimilarity.

3.2.1. Local relocation algorithm

The proposed relocation algorithm consists of two stages:

i. Construction of MFA vectors and calculation of permissible distances from a proper relocation range around additional module.
ii. Local Relocation with MFA

At first stage, given the model placement and an additional module with its coordinates, the small area around the additional module is scanned to find proper range that has enough free space as the *local relocation range*, then necessary information that will be used at the second stage are extracted. At second stage, MFA algorithm starts to move or rotate some modules (movable modules) considering critical distances criteria using information of first stage. All of the seconcepts like movable modules, permissible distances and critical distances are defined at the following sections.

3.2.2. Calculation of permissible distances and construction of MFA vectors

The first stage of local relocation algorithm has to extract information of hypergraph representation of selected part of model placement as inputs of second stage, such as P, Q and sets C and N and MFA input vectors. The selected part of model placement is called the local *relocation range* and must has enough free space or dead space for inserting an extra module.

Selecting size and position of relocation rang depends on size of additional module and desirable similarity between model placement and relocated placement. It is clear that selecting bigger part of a model placement as a relocation range may cause more unsimilarity. So, this algorithm seeks around additional module in different directions considering relocation range limitation to find desirable range.

After relocation range determination, its underlying modules are classified into two groups:

First group includes modules that are completely inside the relocation range and are *movable modules*. Second group consists of modules that just overlap with relocation range and must have fixed position during relocation because they form a frame around movable modules and are *fixed modules*.

Actually if we assume the model placement as a puzzle, this frame is just a piece of it. It's clear that after local relocation, this piece must fit on its location again so any movement or rotation from inside modules must preserve vertical and horizontal distances between outer ones. Fig. 3.a shows the relocation range and its underlying modules on the model placement. Fig. 3.b shows local relocated placement of Fig. 3.a.

Dashed square is the relocation range and black module is the additional module. Modules marked as "o" are outer modules and those marked as "i" are inner modules. In our method we have used MFA with discrete variable for relocation, so the problem's configuration must encode to discrete space. As a result, the width and height of relocation range are divided into equal spans that form some columns and rows respectively. The rows and columns that are occupied with modules are marked. The outer modules are then separated into four sets: up boundary modules, down boundary modules, left boundary modules and right boundary modules.

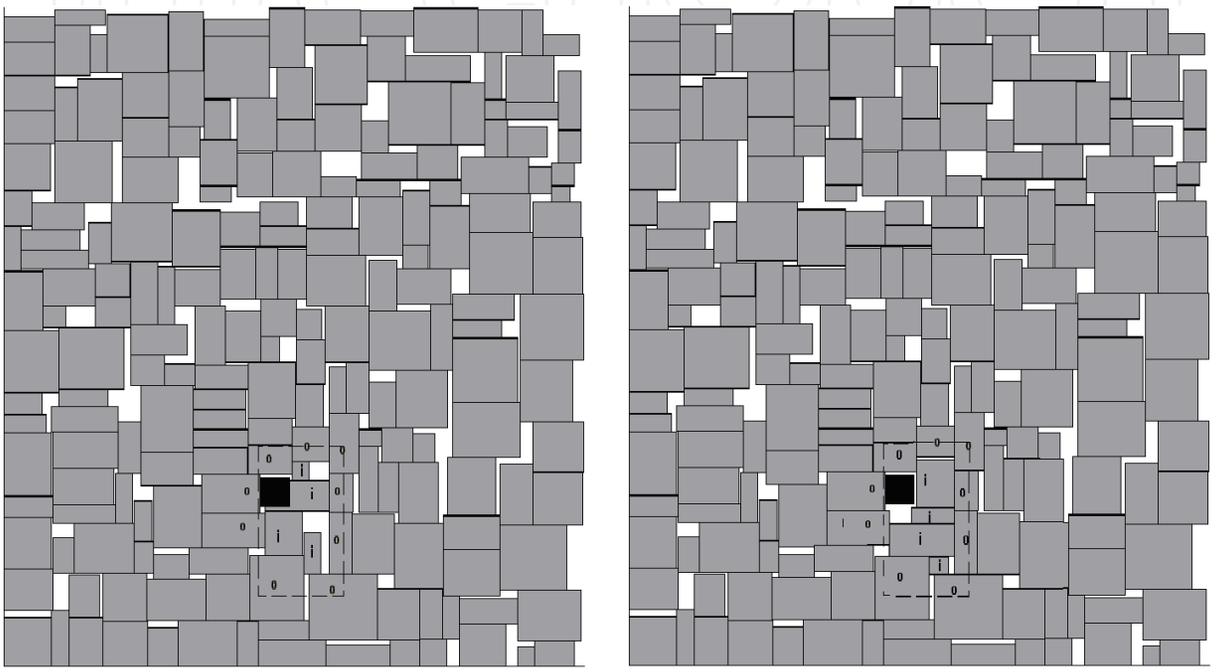


Figure 3. a) Relocation range and it sunder lying modules. b) Local relocated placement using MFA

3.2.3. Calculating permissible distances

For each row or column, two modules are determined as its boundary module. *Permissible distance* of every row or column is obtained with calculating distance between left boundary module and right boundary module of that row or distance between up boundary module and down boundary module of that column respectively. Fig. 4.a shows coordinates of a module. Left-down corner and right-top corners of a module are considerable here. Right-top corner coordinate of module "i" is obtained.

$$xw_i = x_i + w_i \ , yh_i = y_i + h_i \tag{14}$$

For each row or column, two modules are determined as its boundary modules. Fig. 4.b represents boundary modules of the relocation range shown in Fig. 3.

In Fig. 4.b row and column permissible distances are computed using Eq. 15 considering coordinates of the boundary modules of that row or column. Subscribe "*o*" Refers to outer modules, $Rpd_i$and Cpd_irepresent *ith* row's *jth* column's permissible distances.

$$Rpd_i = x_{oi} - xw_{oi}\,,\; Cpd_i = y_{oj} - yh_{oj} \tag{15}$$

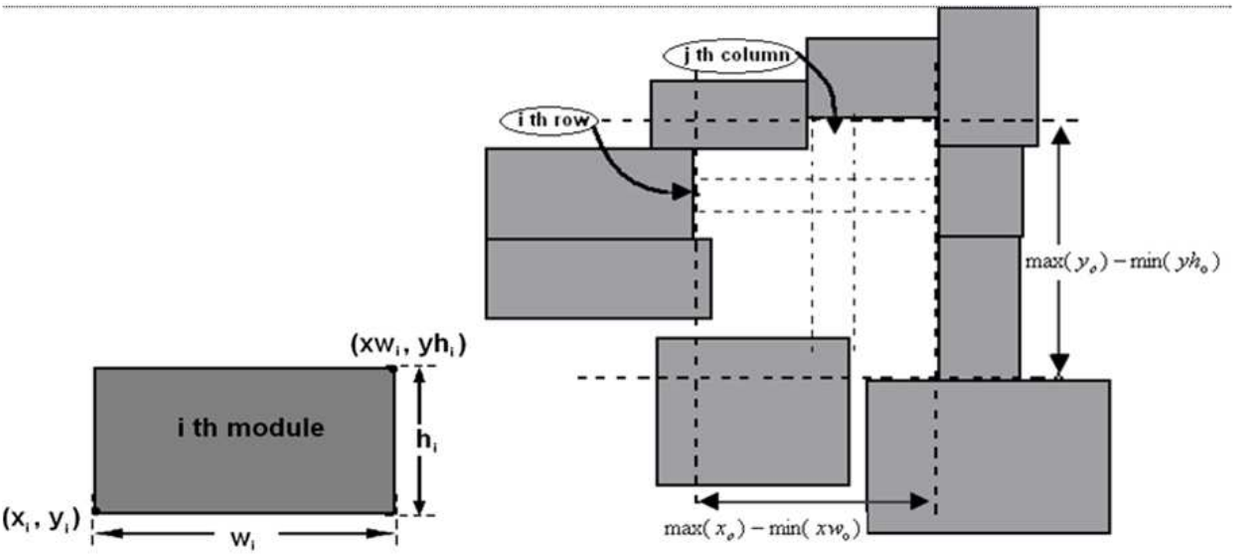


Figure 4. a) Coordinates of a module b) Boundary modules

In main algorithm sum of widths or heights of modules that are located in the same row or column are calculated and results are not permitted to exceed permissible distance of that row or column. For decreasing number of variables and calculations, outer modules that must have fixed position are laid aside and just inner modules that are movable enter MFA algorithm. In addition extra module as an overlap maker module enters the algorithm but it stays on its location during algorithm. Some of outer modules that advance inside the inner modules area could enter MFA algorithm to prevent some undesirable locating.

3.2.4. Construction of MFA initial average spin vectors based on the position of movable modules (mapping)

In addition extra module as an overlap maker module enters the algorithm too but it stays on its location during algorithm. Some of outer modules that advance inside the inner modules area could enter MFA algorithm to prevent some undesirable locating. We divided inner modules area to *P* rows and Q columns. Minimum value between all of the heights and widths of the modules is obtained. Then the width and height of relocation range are divided to this obtained value and rounded to integer values that are number of columns and rows; Q and P. We define position of a module with two vectors at MFA space, one for representing its vertical position and another one for its horizontal position. These vectors have *P* and *Q* elements respectively and for module *"m"* these vectors are shown with v_m^r andv_m^c that finally form overall matrices as v^randv^c. Every element of above mentioned vectors called spin (neuron) and sum of values of these elements is equal to 1. Left-down corner coordinate of a module determines its position, that means if this point locates in range of *ith* row and *jth* column, *ith* element of $v_m^r$and *jth* element of v_m^c is set to 1 and others to 0 as:

$$v_m^r = [\underbrace{0 \quad \dots}_{1:\,i-1} \quad \underbrace{1}_{i} \quad \underbrace{\dots \quad 0}_{i+1:\,P}] \quad , \quad v_m^c = [\underbrace{0 \quad \dots}_{1:\,j-1} \quad \underbrace{1}_{j} \quad \underbrace{\dots \quad 0}_{j+1:\,Q}] \tag{16}$$

To construct precision vertical and horizontal vectors we used a pseudo-trigonometric method. Module position is determined using its left-down corner distance with left-down corner of relocation range with coordinate as(x_{rr}, y_{rr}).Fig. 5 shows the relocation range of Fig. 3 and its incident inner modules that are darker one. We used a special value to normalize these distances. This value is Euclidean distance between left-down corner of relocation range and a point with coordinate of inner modules maximum "x" and maximum "y" as:

$$Sd = \sqrt{(\max(x_{in}) - x_{rr})^2 + (\max(y_{in}) - y_{rr})^2} \tag{17}$$

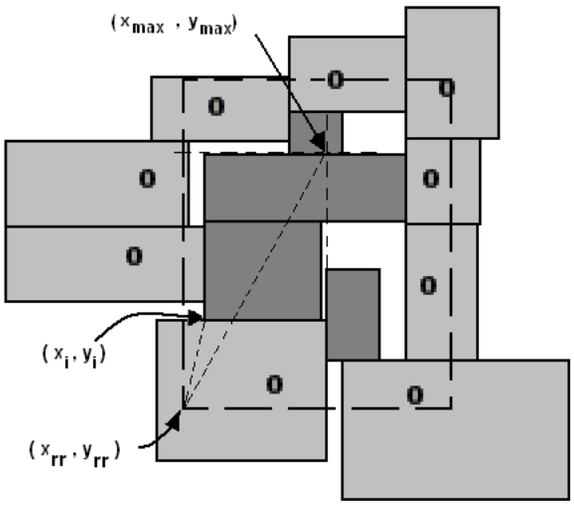


Figure 5. Relocation range and its inner modules for construction of MFA initial average spin vectors

Then for calculating row vector of a module, its vertical distance with left-down corner of relocation range is obtained and then normalized as Eq. 18. Same calculation is done for column vector.

$$vd_{\mathrm{i}} = \frac{y_{\mathrm{i}} - y_{rr}}{Sd} \, , hd_j = \frac{x_j - x_{rr}}{Sd} \tag{18}$$

Eq. 19 represents normalized total horizontal and vertical ranges. Horizontal range is divided into P parts and vertical range into Q parts. The algorithm then determines position of modules based on their $vd_{\mathrm{i}}$and hd_jvalues in comparison to P and Q obtained spans.

$$\text{Horizontal range} = \left(\frac{\min(x_{\mathrm{in}}) - x_{\mathrm{rr}}}{\mathrm{Sd}} \, , \frac{\max(\mathrm{x_{in}}) - \mathrm{x_{rr}}}{\mathrm{Sd}}\right), \tag{19}$$

$$\text{Vertical range} = (\frac{\min(\mathrm{y_{in}}) - \mathrm{y_{rr}}}{\mathrm{Sd}} \, , \frac{\max(\mathrm{y_{in}}) - \mathrm{y_{rr}}}{\mathrm{Sd}})$$

For module *"m"*, being in the *ith* vertical span causes the *ith* element of v_m^r to become 1 and being in the *jth* horizontal span causes the *jth* element of v_m^c to be equal to 1. In MFA space that means probability of finding module *"m"* at row "*i*" and column "*j*" is 1. v^r and v^c are initial average spin vectors as two inputs of MFA algorithm. Fig. 6 shows the flowchart of first stage of MFA local relocation algorithm.

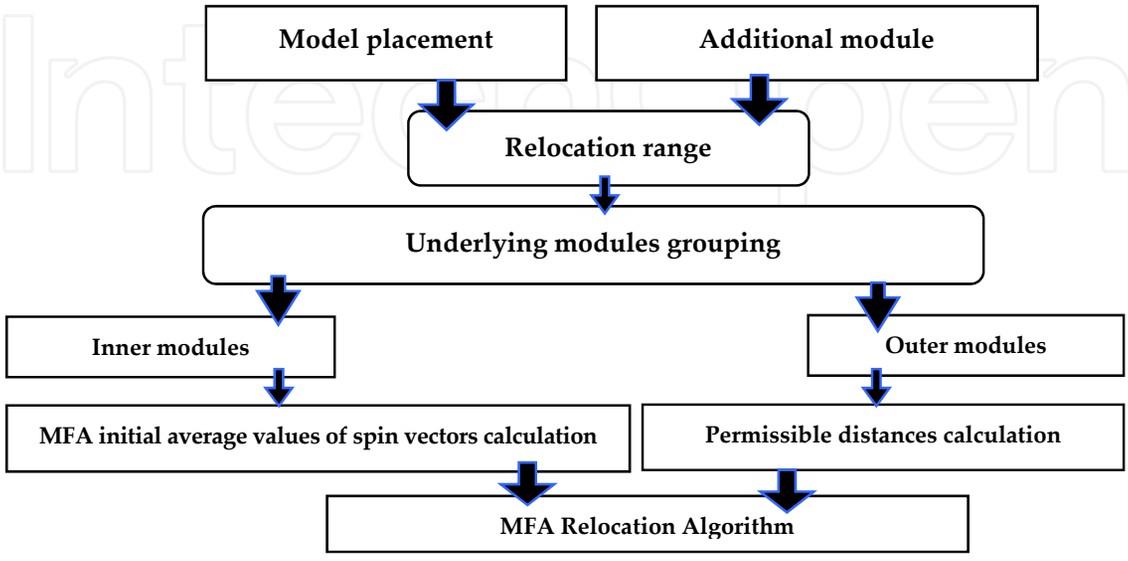


Figure 6. The flowchart of first stage of Local relocation

3.2.5. MFA relocation algorithm

At every epoch of MFA Algorithm one of the movable modules is selected randomly for mean field vector calculation from a random select list that includes movable modules with unconverged average spin vectors, and then selected module's average spin vector are updated using this vector. At the end of every epoch spin of every average vector that is greater than "0.9" is set to 1 and others are set to 0 and this vector is deleted from random select list because it has converged.

3.2.6. Energy functions

MFA Algorithm moves modules to minimize a total energy function. Our MFA relocation algorithm's total energy function is summation of three energy functions. First of all is routing cost function or wire length energy that is sum of vertical and horizontal routing costs and the algorithm minimizes it. Second one is the overlap cost and avoids algorithm to locate more than one module in same location. In MFA probability of being a module in row "*i*" and column "*j*" in the same location is computed for all of the modules. The energy term is formulated corresponding to the overlap cost as Eq. 7 in cell-placement problem [14]. In Eq. 20, ω_i and ω_j are constant values as the weights of modules "*i*" and "*j*" and are given from a module weight function that is used to encode the areas of modules. These values are some of input values of the algorithm and ω_i for module "*i*" is related to its area. v_{ip}^r is the probability of finding module "*i*" in one of the *Q* locations at row "*p*",

and v_{jq}^{c} is the probability of finding module "i" in one of the P locations at column "q", respectively.

Last energy function that supervises preserving permissible distances is *permissible distances preservation energy* or E_{pd}. When a selected module moves to a location, the summation of widths and heights of the modules that are in the same column or row are calculated and are compared to permissible distance of that row and column. If these values exceed the permissible distances first the selected module is rotated and the summation and comparison is done again. If the problem still exists the value of $E_{pd}$and total energy increases respectively. In Eq. 21, E_t, E_w and E_o are total energy function, routing cost or wire length energy function and overlap energy function, respectively. α and β are balance factors between E_w , E_o and E_{pd} .α and β are constant during simulation and are used to increase or decrease importance of every energy functions in total energy function related to others.

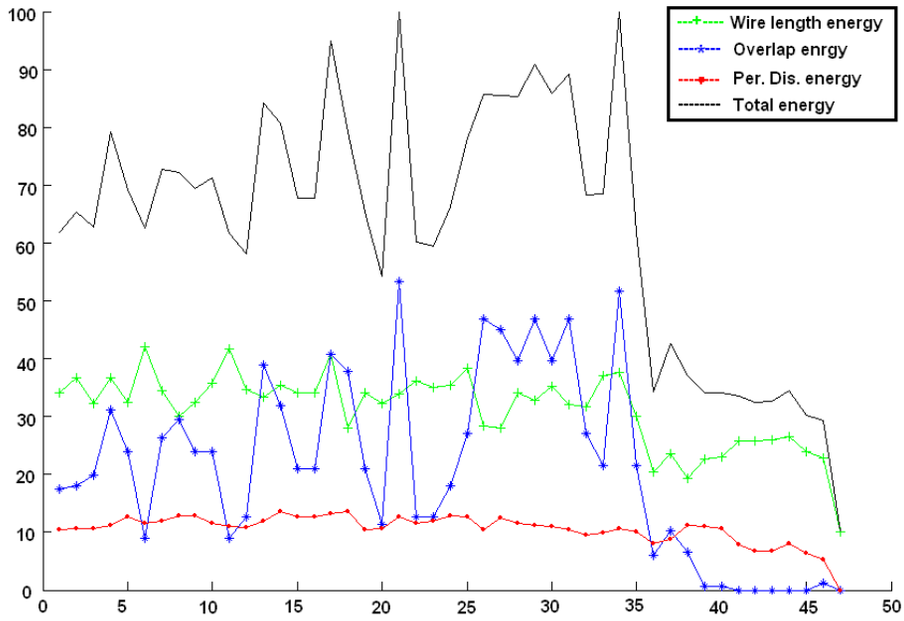


Figure 7. The energy function minimization: $E_t, E_w, E_o$and E_{pd}

$$E_o = \frac{1}{2}\sum_i \sum_{j\neq i} \omega_i\omega_j \times P\{\text{Modules i and j are in the same location}\} =$$

$$\frac{1}{2}\sum_i \sum_{j\neq i} \omega_i\omega_j \sum_{p=1}^{P}\sum_{q=1}^{Q} P\{\text{Module i is in location pq}\}\times P\{\text{Module j is in location pq}\} \quad (20)$$

$$= \frac{1}{2}\sum_i \sum_{j\neq i} \omega_i\omega_j \sum_{p=1}^{P}\sum_{q=1}^{Q} v_{ip}^{r} v_{iq}^{c} v_{jp}^{r} v_{jq}^{c}$$

$$E_t = E_w + \alpha \times E_o + \beta \times E_{pd} \quad (21)$$

Fig. 7 shows energy function and its parameters, wire length cost, overlap energy and permissible distances preservation energy. At the final epoch, where all of the spins

converge, overlap and permissible distances preservation energies become 0 and wire length cost is minimized, therefore total energy is minimized too.

3.2.7. Cooling Schedule

For local relocation problem the cooling process is realized in three phases, slow cooling followed by fast cooling and then very fast cooling(or *quenching*).Eq. 22 shows the cooling schedule algorithm. T_{r0}, T_{c0}, T_r and T_c are horizontal and vertical initial temperatures and horizontal and vertical current temperatures of system, respectively.

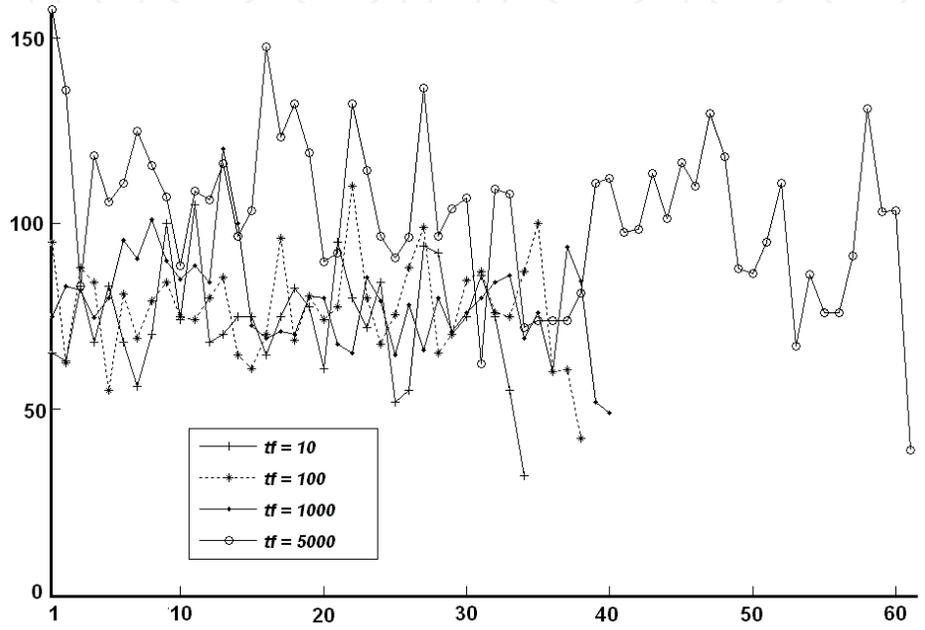


Figure 8. The total energy function minimization for three values of t_f: 10, 100, 1000 and 5000

$$if\ (0.8 \times T_{r0} \le T_r \& \ 0.8 \times T_{c0} \le T_c) : T_r = 0.95 \times T_r\ ;\ T_c = 0.95 \times T_c$$

$$elseif\ (0.35 \times T_{r0} \le T_r \le 0.8 \times T_{r0} \& \ 0.35 \times T_{c0} \le T_c \le 0.8 \times T_{c0}):$$

$$T_r = 0.8 \times T_r\ ;\ T_c = 0.8 \times T_c$$

$$elseif\ (0.35 \times T_{r0} \ge T_r \& \ 0.35 \times T_{c0} \ge T_c)\ : T_r = 0.65 \times T_r\ ;\ T_c = 0.65 \times T_c \qquad (22)$$

end

Due to having vertical and horizontal spins two initial temperatures are calculated at the first of algorithm according to vertical and horizontal sizes of problem and a constant factor is called initial temperature factor or t_f like Eq. 23.

$$T_{r0} \propto t_f \times P\ , T_{c0} \propto t_f \times Q \qquad (23)$$

Fig. 8 represents total energy minimization during algorithm iterations for three different values of $t_f$as 10, 100 and 1000. It is clear that changing this factor causes changing number of iterations and also minimum value of total energy function.

On the other hand, setting this factor to insufficient values (specially too high values) may cause unconvergence or unacceptable results, so the range of this factor is limited and according to our experiments is less than 5000.

The cooling process continues until either 90% of the spins are converged or temperature reduces below 1% of initial temperature. So when current temperature is below the 35% of initial temperature, a very fast phase of cooling process moderates the unconverged spins very fast.

At the end of this process, the variable with maximum value in each unconverged spin is set to 1 and all other variables are set to 0.

Fig. 9 shows the flowchart of second stage of MFA local relocation algorithm.

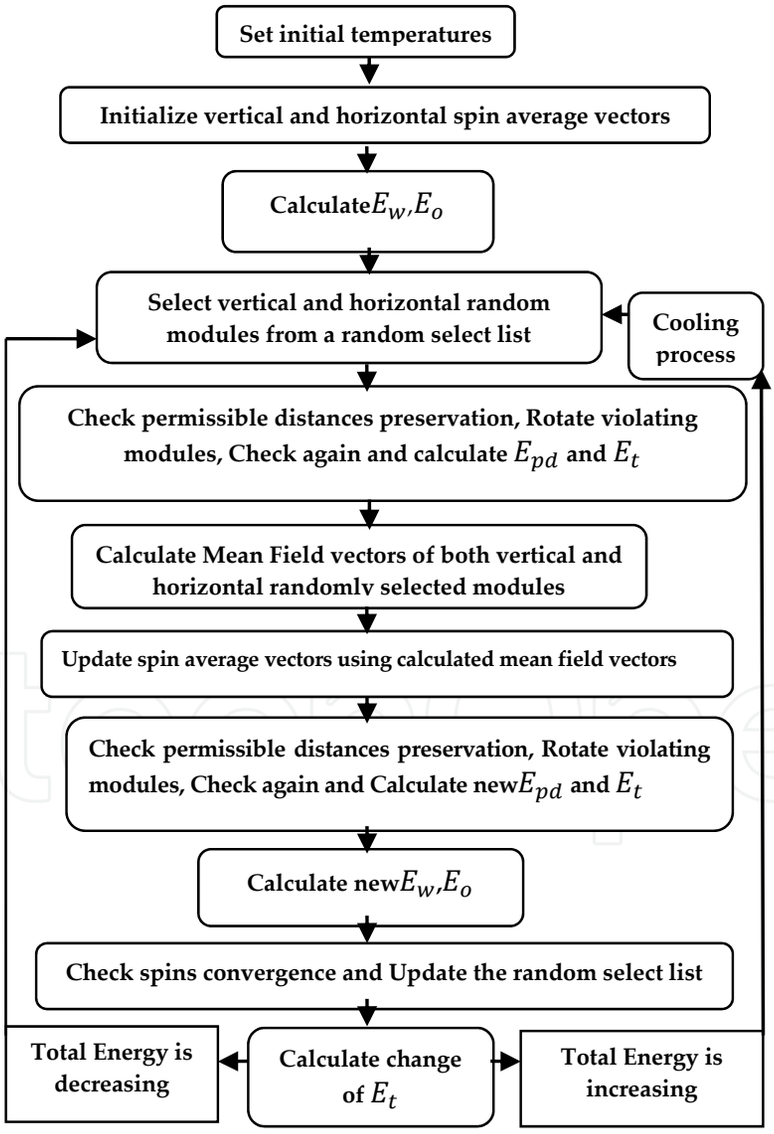


Figure 9. The flowchart of second stage of MFA local relocation algorithm

3.2.8. Experimental results

We implemented the proposed algorithm on a 2.4GHz Intel Pentium IV with 512MB memory using MATLAB 7.2.0.232 (R2006a) in WINDOWS operating system. We applied the proposed algorithm to the relocation of n300a, n200a, and n100a, which are distributed in GSRC benchmarks in [17].

For every benchmark five different problems were resolved using our proposed algorithm and maximum and average runtime of 10 runs of them are presented in Table 1. Results show that our MFA based algorithm is faster than SA-based proposed method in SA-based relocation method in [18] because the number of displacements is limited to the number of movable modules of problem and the problem is local relocation. Actually relocation range reflects on number of displacements and also similarity of resultant placement with model placement.

Results show runtimes of our proposed algorithm almost do not depend on the size of benchmark circuit in compare to the method represented in SA-based proposed method, actually size of local relocation range and numbers of movable modules of each problem are the main parameters here. Also feasibility of local relocation solution, to guarantee the similarity of resultant placement with model placement depends on the existence of enough dead space near additional module so that the relocation rage becomes limited and small.

	MFA Local Relocation			SA
Benchmark	Min. runtime (Sec.)	Average runtime (Sec.)	Max. runtime(Sec.)	Min. runtime (Sec.)
n100	2.0	2.37	2.52	1.0
n200	3.2	3.62	3.72	9.0
n300	3.7	3.92	3.96	60.8

Table 1. MFA Local Relocation results for GSRC benchmarks

4. Conclusion

Briefly, Our proposed method as a local solution method has less displacement and by taking advantages of MFA algorithm in comparison to SA algorithm and localizing problem (that reduces number of engaged modules) and therefore by having less variables, is faster. Also having less number of movable modules causes more similarity *if the solution is feasible*.

Selection of modules for relocation is based on the range that includes enough free space around the extra module so the runtimes of our proposed algorithm almost do not depend on the size of benchmark circuit in compare to the SA-based method, actually size of local relocation range and numbers of movable modules of each problem are the main parameters. Applying ability of rotation of modules inside a fixed distance controller energy function as permissible distances preservation energy and three phases cooling process are main properties of our employed MFA algorithm. Results show our method is almost independent of size and complexity of model placement.

Although the use of SA provides for escaping from the local minima, it results in an excessive computation time requirement that has hindered experimentation with the Boltzmann machine. In order to overcome this major limitation of the Boltzmann machine, a mean field approximation may be used. In mean field network, the binary state stochastic neurons of the Boltzmann machine are replaced by deterministic analogue neurons. A simple formulation of the Traveling Salesman Problems energy function is described which, in combination with a normalized Hopfield-Tank neural network, eliminates the difficulty in finding valid tours[1]. This technique, as the one of the bases of MFA algorithm, is applicable to many other optimization problems involving n-way decisions (such as VLSI layout and resource allocation) and is easily implemented in a VLSI neural network. The solution quality is shown to be dependent on the formation of elements of the problem configuration which are influenced by the constraint penalties and the temperature as what is borrowed from SA technique. The applied algorithm for local relocation problem is modified form of which is applied for cell placement problem. The cooling schedule has three stages that the final stage is very fast cooling with decreasing factor 0.65 that may be what you mean *quenching*. Otherwise other two stages with decreasing factors 0.95 and 0.8 are not so fast and have *annealing* essence. For more information about this topic, one can refer to [1].

Author details

Gholam Reza Karimi and Ahmad Azizi Verki
Electrical Engineering Department, Engineering Faculty- Razi University, Kermanshah, Iran

5. References

[1] VandenBout, D. E. & Miller, T. K. (1989). Improving the performance of the Hopfield-Tank neural network through normalization and annealing, *Biological Cybernetics*, Volume 62, Number 2, Pages 129-139.

[2] Peterson, C.& Soderberg, B. (1989). A new method for mapping optimization problems on to neural networks, *International Journal of Neural Systems,* vol.1 (3), pp. 3–22.

[3] Takahashi, Y. (1997). Mathematical improvement of the Hopfield model for TSP, feasible solutions by synapse dynamical systems. *Neurocomputing,* vol. 15 pp. 15–43.

[4] Gislen, L.; Peterson, C. & Soderberg, B. (1992).Complex scheduling with Potts neural networks," *Neural Computation*, vol. 4, pp. 805–831.

[5] Bultan, T. & Aykanat, C. (1992). A new mapping heuristic based on mean field annealing, *Journal of Parallel and Distributed Computing,* vol. 16, pp. 292–305.

[6] Ohlsson, M.; Peterson, C. & Soderberg, B. (1993). Neural networks for optimization problems with inequality constraints - the knapsack problem, *Neural Computation,* vol.5 (2), pp. 331–339.

[7] Ohlsson, M. & Pi, H. (1997).A study of the mean field approach to knapsack problems, *Neural Networks,* vol. 10(2), pp. 263–271.

[8] Hokkinen, J.; Lagerholm, M.; Peterson, C. & Soderberg, B. (1998).A Potts neuron approach to communication routing, *Neural Computation,* vol. 10, pp. 1587–1599.
[9] Herault, L. & Niez, J. (1989). Neural networks and graph k-partitioning, *Complex Systems,* vol. 3, pp. 531–575, 1989.
[10] VandenBout, D.E. & Miller, T.K. (1990). Graph partitioning using annealing neural networks, *IEEE Transaction on Neural Networks,* vol.1(2), pp. 192–203.
[11] Cimikowski, R. &Shope, P. (1996).A neural-network algorithm for a graph layout problem, *IEEE Transactions on Neural Networks,* vol. 7 (2), pp. 341–345.
[12] Yih, J.S. & Mazumder, P. (1990). A neural network design for circuit partitioning, *IEEE Transactions on Computer-Aided Design,* vol.9, pp. 1265–1271.
[13] Bultan, T. & Aykanat, C. (1995). Circuit partitioning using mean field annealing, *Neurocomputing,* vol. 8, pp. 171–194.
[14] Aykanat, C.; Bultan, T. & Haritaoglu, I. (1998). A fast neural-network algorithm for VLSI cell Placement, *Neural Networks,* vol. 11 pp. 1671–1684.
[15] Karimi, G.R.; AziziVerki, A. & Mirzakuchaki, S. (2010). Optimized Local Relocation for VLSI Circuit Modification Using Mean Field Annealing, *ETRI Journal*, Volume 32, Number 6.
[16] Hopfield, J.J. & Tank, D.W. (1985). "Neural" Computation of Decisions in Optimization Problems, *Biological Cybernetics,* vol. 52, pp. 141-152.
[17] http://vlsicad.eecs.umich.edu/BK/CompaSS
[18] Yanagibashi, K.; Takashima, Y. & Nakamura, Y. (December 2007). A Relocation Method for Circuit Modifications, *IEICE TRANS, FUNDAMENTALS,* vol.E90–A, NO.12.

IntechOpen