A Swarm Based Global Routing Optimization Scheme

Abhinandan Khan, Pallabi Bhattacharya and Subir Kumar Sarkar
Dept. of Electronics and Telecommunication Engineering
Jadavpur University
Kolkata, India
khan.abhinandan@gmail.com, pallabi.a1@gmail.com and sksarkar@jdvu.ac.in

***Abstract*—Swarm Intelligence (SI), modelled upon the behaviours of various swarms of animals and insects such as ants, termites, bees, birds, fishes, fireflies, etc. is an emerging area in the field of optimization. SI based algorithms are proclaimed to be robust and efficient optimization tools. This fact is corroborated by a number of practical engineering problems where these algorithms give very satisfactory results. Nowadays VLSI Design has become one of the most intriguing and fervent research field for engineers. Efficient development of a system of a billion chips and blocks on a printed circuit board requires extensive use of optimization in various areas of design such as chip size, separation among components, interconnect length etc. One of the most significant among these is the interconnect wirelength, which determines the overall delay in transmission within the chip. The routing phase in the VLSI Physical Design strives to optimize the interconnect length. Several studies have been and are being conducted to improve the performance of VLSI chips by optimally interconnecting the various components. Various SI based algorithms have already proved their efficiency in this field of routing optimization. In this paper we have proposed a global routing scheme based on contemporary SI algorithms: Firefly Algorithm (FA), and Artificial Bee Colony (ABC) algorithm and have compared the performance of the two. FA produces superior optimization results in comparison to ABC although proving to be quite expensive, computationally.**

Keywords—artificial bee colony, firefly algorithm, global routing, swarm intelligence

I. INTRODUCTION

With the rapid growth in the number of transistors in a VLSI system, the design process becomes very complex and requires optimization at various stages of the design cycle. One such important phase is where the various passive and active components are connected with each other in lieu of all design specifications and constraints. This stage is known as Routing. The connections are made with the help of metal wires, known as interconnects. Any delay in transmission of signals via these interconnects arises due to their capacitive, resistive or inductive effects. All such effects depend on the dimensions of the interconnecting wires and they also lead to considerable power dissipation. Now these interconnects have a fixed area but a manoeuvrable length. So in order to minimize the undesirable effects, the length of the interconnecting wires need to be minimised as greatly as possible. Thus, originates the necessity of efficient routing algorithms in VLSI Physical Design.

The problem of global routing is actually the problem of finding a Minimal Rectilinear Steiner Tree (MRST) by joining all the terminal nodes or points of the components. Obtaining a MRST is a NP-complete problem [1] i.e. a real time solution does not exist. Several algorithms exist which give satisfactory results to such problems [2-3]. Swarm Intelligence [4] draws inspiration from nature to provide robust and efficient solutions to such critical real world optimization problems. Swarms use their environment and resources effectively using collective intelligence to obtain near satisfactory results for such NP-hard problems [5]. Some of the popular optimization techniques based on swarm intelligence are PSO (particle swarm optimization [6]), ACO (ant colony optimization [7]), ABC (artificial bee colony algorithm [8]), FA (firefly algorithm [9]), etc. In this work, we propose a global routing algorithm based on the FA and compare the results against those of ABC.

The paper is organized as follows: Section II introduces the basic theories related to routing in VLSI circuits, swarm intelligence and the SI-based algorithms: ABC and FA. Section III illustrates the problem formulation and our proposed approach. Section IV presents the experimental setup and the relevant parameters. Section V presents the experimental results and required discussions. The paper concludes with Section VI.

II. PRELIMINARIES

A. Routing

There are two phases of routing in the physical design process, namely, global routing and detailed routing. Global routing allocates routing resources for connections between components. Detailed routing assigns routes to specific metal layers and routing tracks within the global routing resources. Global routing is one of the most stimulating optimization problems using various approaches based on evolutionary computation and swarm intelligence. In the global routing phase of VLSI design, we assume that the circuits are in a one-layer frame [1]. A chip is simulated as a lattice graph, where each channel in the chip relates to an edge in the lattice graph. Pins of the chip components are obtained at the intersections of these edges, which correspond to vertices in the lattice graph. A net is defined to be a group of pins which are to be connected. In any instance, a set of nets is given, each of which has pins to be connected by wires. Additionally, there are constraints on the number of wires that may pass through any given channel and timing constraints. A solution is a set of MRSTs in the lattice graph, one for each net, corresponding to the wires in the chip routing the given nets satisfying all constraints. The goal is to minimize the length of the MRSTs.

B. Swarm Intelligence

Evolutionary computation [10] is a sub-part of artificial intelligence, mainly computational intelligence. It deals with continuous as well as combinatorial optimization problems. Evolutionary computation uses iterative progress, such as growth or development in a population. This population is then selected in a guided random search using parallel processing to achieve the desired end. Such processes are often inspired by biological mechanisms of evolution. SI is the collective behaviour of decentralized, self-organized systems, natural or artificial. SI systems consist typically of a population of simple agents interacting with each other and with their environment. The inspiration often comes from nature. A swarm should be able to do simple space and time computations, should be able to respond to quality factors of the environment like the quality of food source or the safety of the food source location, should not change its mode of behaviour with every fluctuation in its environment and must be able to change its behaviour when the investment in energy is worth the computational price.

C. Artificial Bee Colony Optimization

The ABC algorithm was developed by Karaboga [8] and it uses the intelligent foraging behaviour of honeybees. In the algorithm the artificial bees fly around in a search space to find probable food sources (solutions) with high nectar amount (fitness) and finally obtain the food source with the highest nectar (the best solution). The colony of artificial bees consists of three categories of bees: employed, onlooker and scout bees. The number of employed artificial bees is equal to the number of probable food sources as well as the number of artificial onlookers. During each cycle of the search, employed bees search for food sources and return with their nectar amounts (i.e. the fitness of each random solution). They share the nectar and position information of the food sources with the onlookers. The onlooker bees then choose food sources with a probability based on the nectar amounts calculated using equation (3). As and when a food source is exhausted, the visiting employed bee becomes a scout bee. The scout bee searches for a new food source in the search space previously not visited, and when it does find a new food source, the scout becomes employed. These iterative processes are repeated until the termination condition is satisfied. The ABC algorithm has only two input parameters; population of the colony and the counter for a solution that does not change over a predetermined number of cycles. Initially, ABC algorithm generates random solutions using:

$$x_{i,j} = x_{j,max} - rand(0,1) \times \left(x_{j,max} - x_{j,min}\right) \qquad (1)$$

In the above equation, i = {1, 2, 3, …, S} and j = {1, 2, 3, …, D}. S is the number of possible solutions and equals to half of the population. D is the dimension of the solution space, $x_{j,max}$ and $x_{j,min}$ are the lower and upper bounds for the dimension, j. After initialization, each solution of the population is evaluated and the best solution is memorized. Each employed bee produces a new food source in the neighbourhood of its present position by using:

$$v_{i,j} = x_{i,j} - \emptyset_{i,j} \times \left(x_{i,j} - x_{k,j}\right) \qquad (2)$$

Here k = {1, 2, 3, …, N} and j = {1, 2, 3, …, N} are randomly chosen indexes; k ≠ i. $\emptyset_{i,j}$ is a random number in the range [-1,1]. New solution $v_{i,j}$ is modified with the previous solution $x_{i,j}$ and a random position in its neighbourhood $x_{i,k}$. The new solution $v_{i,j}$ is evaluated and compared to the fitness of $x_{i,j}$. If the fitness is better than that of $x_{i,j}$, it is replaced with $x_{i,j}$ otherwise, $v_{i,j}$ is retained. After the searching process the employed bees share nectar amounts and positions of the food sources with the onlookers. For each onlooker bee, a food source is chosen by a selection probability. The probability of the food source is calculated:

$$p_i = \frac{fitness_i}{\sum fitness_i} \qquad (3)$$

Here $fitness_i$ is the fitness of the solution (food source), *i*. Finally, the best solution is memorized and the algorithm repeats the search processes of the employed bees, onlookers and the scouts until the termination condition is met. The artificial bee colony algorithm can be explained as below:

Define objective function f(x), maximum number of iterations, maxit
Initialize a population of bees x_i(i=1,2,..,n)
Initialize iteration count iter=0
Define a counter, c
While (iter<maxit)
  For i = 1:n/2(employed bees)
    Calculate new solution by eqn. (1)
    Calculate f(x_i) for all i
    Calculate p_i by eq. (3)
  End for
  For i = 1:n/2(onlooker bees)
    Select solution based on p_i
    Calculate new solution by eq. (2)
    Calculate f(x_i) for all i
    Use greedy selection
  End for
  If for a particular x_i, f(x_i) does not improve until counter
    Scout produces new solution using eq. (1)
  End if
  iter = iter +1
End while

D. Firefly Algorithm

Fireflies, one of the most fascinating creatures in nature, use bioluminescent signals for courtship rituals, enamouring other fireflies; as methods of prey attraction; or as a warning signal. Based on the biochemical and social aspects of fireflies, X. S. Yang [9] introduced the FA. In the FA, the objective function of a given optimization problem is based on the light intensity. It helps the fireflies to move towards brighter and more attractive locations in order to obtain optimal solutions. All fireflies are characterized by their light intensity associated with the objective function. Fireflies are unisex and move towards more attractive and brighter ones. The attractiveness of a firefly is proportional to its brightness which decreases as the distance from other fireflies, increases. The attractiveness of a firefly is described by a monotonically decreasing function of the distance r between any two fireflies:

$$\beta = \beta_0 e^{-\gamma r^2} \qquad (4)$$

In the above equation, β_0 denotes the maximum attractiveness (at r = 0) and γ is the light absorption coefficient,

which controls the light intensity. The distance between any two fireflies, *i* and *j* residing at positions, x_i and x_j, respectively, can be defined as follows:

$$r_{i,j} = \| x_i - x_j \| = \sqrt{\sum_{k=1}^{d} \left(x_{i,k} - x_{j,k}\right)^2} \quad (5)$$

Here $x_{i,k}$ is the kth component of the spatial coordinate x_i, and d denotes the dimension of the problem. The movement of a firefly i is determined by the following form:

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r^2}\left(x_j^t - x_i^t\right) + \alpha_t \varepsilon_i \quad (6)$$

Here the first term is the current position of a firefly *i*, the second term is due to the attraction and the third term is randomization with α_t being the randomization parameter, and ε_i is a vector of random numbers being drawn from a Gaussian distribution or uniform distribution. The FA can be presented by the following pseudo-code:

Initialize parameters: number of fireflies (n), γ, β_0, α (random parameters) and maximum number of iterations, maxit
Define the objective function f(x), x = (x1, . . . , xd)^T .
Generate initial population of fireflies x_i (i = 1, 2, . . . , n)
Light intensity of firefly, I_i at x_i is determined by value of objective function f(x_i)
While k < maxit
For i = 1:n //all n fireflies
For j = 1:n
If ($I_j > I_i$)
Move firefly, i towards firefly, j according to eqn. (6)
End if
Obtain attractiveness according to eqn. (4)
Find new solutions and update light intensity
End for j
End for i
Rank the fireflies and find the current best
End while
Find the firefly with the highest light intensity

Some of the many advantages of the algorithm are: High convergence rate, Automatic subdivision and Random reduction.

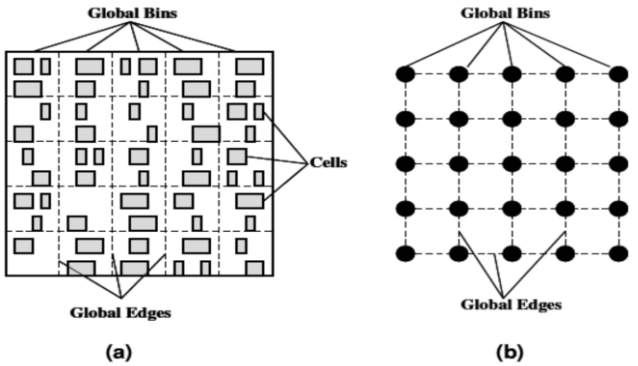

Fig. 1. Illustration of routing.

III. Problem Formulation and Proposed Approach

The problem of global routing optimization can be formulated as the problem of finding an MRST connecting all the terminals or pins, without any severe loss of. As illustrated in fig. 1, the whole routing region is usually partitioned into a number of global bins. Each global bin is represented by a node and each common boundary is represented by an edge in the grid graph. The edges are known as global edges. We have assumed for our work that the capacity of the edges is always greater than the demand or in other words, there is no congestion. This grid graph can be represented by a binary matrix where the elements represent the nodes of the grid graph. This matrix represents the search space for our algorithms. An element is assigned the value of 1 or 0 depending on whether the particular node needs to be connected or not respectively. Such a matrix can be easily reduced by eliminating the rows and columns which do not contain any connectable nodes. This is done solely to reduce the computational load. We have assumed in our work that the search space is a 100-by-100 point VLSI system. The subsequent matrix after reduction looks like this:

$$Z = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

For both the algorithms, this represents the constituent of the swarm: bee for ABC and firefly for FA. However to apply these algorithms, some random points need to be added to the above matrix which shall act as the Steiner points of the MRST. The matrix thus created shall be the initial population for both the algorithms. One such matrix is shown with the bold elements the random points added:

$$Z_{20} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Both the algorithms are then applied according to their pseudocodes already presented. The equations (1) to (6) yield continuous values, however, we can use only the binary values of 0 or 1. Thus, necessary corrections have been done for the algorithms to work on a discrete binary environment. Prim's Algorithm has been used to find the cost of the MRSTs formed by these matrices.

IV. Experimental Setup

Two different sets of coordinates of terminal nodes are randomly generate. Both the ABC and the FA are applied on the two different setups to optimally connect all the terminal nodes. Each experiment is run 30 times for each algorithm. The populations of bees and fireflies are set to 200 and the number of iterations is set to 100 for the SI. The algorithms have been

run in Matlab 2012a running on a desktop computer with a Core i7 3.4GHz CPU and 8GB of RAM. All the generated terminal node coordinates have been shown in Table I.

TABLE I. EXPERIMENTAL COORDINATES

EXPERIMENT 1: COORDINATES										
NO.	1	2	3	4	5	6	7	8	9	10
X	17	32	67	27	69	46	23	16	54	08
Y	80	53	61	66	75	09	92	83	100	45
EXPERIMENT 2: COORDINATES										
NO.	1	2	3	4	5	6	7	8	9	10
X	76	51	90	55	15	85	82	93	20	62
Y	46	70	96	14	66	26	25	55	28	48

V. RESULTS AND DISCUSSION

The results obtained have been summarised in Table II in terms of the minimum interconnect length, mean over the 30 runs, the standard deviation and the time taken. It can be seen from the results that the FA gives better results than ABC w.r.t. finding the most optimum MRST of the terminal nodes. However, the FA is much slower than ABC. The FA brings about an improvement of 11% and 15% in the weight of the optimal MRST found compared to ABC for the first and second experiments respectively. In context of the mean weight of the MRST obtained over 30 runs of the algorithm, the FA outperforms the ABC by 15% and 6% respectively. However, when it comes to the time taken for a single run of the algorithm, the performance of the ABC eclipses the performance of the FA in all other categories. The FA is nearly 6 times and 5 times slower than the ABC for the first and the second instances respectively. Although finding out the most optimum interconnection scheme is the ultimate objective here, the huge computational cost incurred by the FA needs to be considered. It does minimize the interconnection cost better than the ABC, however, the amount of extra computational time needed to better the results is humongous. Moreover, there is simply much more variation in the results obtained by the FA as can be seen from the standard deviation figures. So it can be concluded that if the ultimate need is only time delay reduction via interconnect wirelength minimization, FA is the way to go. However, if the slight improvement can be sacrificed, ABC can optimise nearly six VLSI systems in the time FA optimizes one. Ultimately it comes down to a trade-off between delay and computational cost incurred.

VI. CONCLUSION

SI based algorithms are now widely used to solve many critical real world problems and VLSI design is one such problem in today's age of ever advancing technology. An efficient and robust VLSI chip must have a manageable delay and power dissipation levels. These can partly be achieved through efficient routing optimization which obviously is the key for future success in VLSI physical design. In this work, the main objective is to obtain the minimum possible interconnection cost via a global routing scheme based on SI. For this we have utilised the firefly algorithm (FA) and the artificial bee colony (ABC) algorithms. Comparison of obtained results show that the FA performs better than the ABC but is much slower than its competitor. FA is a pretty new algorithm and in the future, if researchers are able to reduce its runtime considerably, it can become a very attractive alternative to other algorithms in the field of VLSI physical design. Until then ABC provides an excellent choice as a global routing optimization technique.

ACKNOWLEDGMENT

Subir Kumar Sarkar thanks the "Devices and Systems Programme" under the UGC funded "University with Potential for Excellence - Phase II" scheme of Jadavpur University.

REFERENCES

[1] G M Garey and D Johnson, "The rectilinear Steiner tree problem is NP-complete," SIAM J. Appl. Math, vol. 32, 1977, pp. 826-834,.

[2] M Pan, and C Chu, "FastRoute 2.0: A High-quality and Efficient Global Routing," 12th Asia and South Pacific Design Automation Conference, pp. 250–255, 2007.

[3] Chris Chu and Yiu-Chung Wong, "FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design," IEEE Transactions on Computer-Aided Design, vol. 27, no. 1, pages 70-83, January 2008.

[4] J H Holland, "Adaptation in Natural and Artificial Systems," University of Michigan Press, Ann Arbor, MI (1975).

[5] Abhinandan Khan, Sulagna Laha, Subir Kumar Sarkar, "A novel particle swarm optimization approach for VLSI routing," 2013 IEEE 3rd International Advance Computing Conference (IACC), pp. 258-262, 22-23 Feb. 2013.

[6] J Kennedy, R Eberhart, "Particle swarm optimization," IEEE international conference on neural networks, vol. 4, pp. 1942–1948, 1995.

[7] M Dorigo, A Colorni and V Maniezzo, "Positive feedback as a search strategy," Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1991.

[8] D Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical report. Computer Engineering Department, Engineering Faculty, Erciyes University, 2005.

[9] X S Yang, "Nature-Inspired Metaheuristic Algorithms," Luniver Press, (2008).

[10] http://en.wikipedia.org/wiki/Evolutionary_computation

TABLE II. EXPERIMENTAL RESULTS

EXPERIMENT 1: RESULTS													
Best Value				Mean Value				Standard Deviation		Computational Load (in sec)			
Absolute Values		Normalised Values		Absolute Values		Normalised Values		Absolute Values		Absolute Values		Normalised Values	
ABC	FA	ABC	FA	ABC	FA	ABC	FA	ABC	FA	ABC	FA	ABC	FA
162	144	1.00	0.89	178.30	151.77	1.00	0.85	07.02	07.68	47.5491	271.9847	1.00	5.72
EXPERIMENT 2: RESULTS													
Best Value				Mean Value				Standard Deviation		Computational Load (in sec)			
Absolute Values		Normalised Values		Absolute Values		Normalised Values		Absolute Values		Absolute Values		Normalised Values	
ABC	FA	ABC	FA	ABC	FA	ABC	FA	ABC	FA	ABC	FA	ABC	FA
205	174	1.00	0.85	213.37	200.80	1.00	0.94	03.90	14.21	52.8531	271.8425	1.00	5.14