International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 3 Issue 2, February-2014, pp: (205-213), Impact Factor: 1.252, Available online at: www.erpublications.com

Performance comparison of firefly algorithm in Global Routing Optimization

Abhinandan Khan[1], Pallabi Bhattacharya[2], Subir Kumar Sarkar[3]
[123]Dept. of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata, India

Abstract: Rapid technological advancements are leading to a continuous reduction in integrated chip sizes. An additional steady increase in the chip density is resulting in device performance improvements as well as severely complicating the fabrication process. The interconnection of all the components on a chip, known as routing, is done in two phases: the global and the local routing phases. This phase affects chip performance significantly and hence researched into extensively. Swarm based algorithms have proven to be quite useful in this domain. It is an emerging area in the field of optimization and this paper presents one such algorithm, the Firefly Algorithm for solving the routing optimization problem. The performance of the algorithm has been compared with other well-known swarm based algorithms. The proposed algorithm shows remarkable improvement in minimizing the interconnect lengths however, computationally, it is much more expensive compared to the other algorithms in question.

Keywords: Swarm Intelligence, Firefly Algorithm, Particle Swarm, Artificial Bee Colony, Global Routing.

I. Introduction

With the increase in the number of transistors in any electronic circuit, the length of the interconnecting wires also increases. These interconnects have a fixed area thus, making it very difficult to reduce the dependant capacitive and resistive effects. These effects play a significant role in determining the overall chip delay. The only parameter that can be controlled is their length. So for efficient chip fabrication we need an optimum interconnect routing strategy to achieve the minimum possible interconnect length. Thus, arises the essential indigence for efficient routing algorithms in VLSI physical design. The routing problem in VLSI is broadly divided into global and local routing phases. This paper deals with the global routing phase which is essentially a case of finding a Minimal Rectilinear Steiner Tree (MRST) by joining all the terminal nodes, known to be an NP-hard problem [1].

There are several algorithms which return near optimal results [2, 3]. Recently algorithms based on Evolutionary Algorithms (such as Genetic Algorithm [4]) and based on Swarm Intelligence (such as PSO [5], ACO [6], ABC[7], etc.) are being increasingly used in the domain of global routing optimization in VLSI Design.

Swarm Intelligence is a modern optimization techniques that can be used to solve critical real world optimization problems. Swarms use their environment and resources effectively using collective intelligence. Self-organization is the key feature of a swarm. Some of the established and popular optimization techniques based on swarm intelligence are PSO (particle swarm optimization) [5], ACO (ant colony optimization) [6], ABC (artificial bee colony algorithm) [7], FA (firefly algorithm) [8], etc. PSO and ACO have been in existence for more than a decade. On the other hand, ABC and FA are one of the most recently introduced swarm-based algorithms the former being marginally older.

In this work, we propose a global routing algorithm based on the FA and compare the results obtained against those of ABC and PSO. ABC was developed by Karaboga, inspired by the intelligent foraging behaviour of a honey bee hive. PSO was introduced by Kennedy and Eberhart, based on the flocking behaviour of birds. FA, on the other hand was developed by Dr Xin-She Yang at Cambridge University, based on the flashing characteristics and social behaviour of fireflies. Being a decade newer, the FA is much simpler both in concept and implementation compared to PSO. Improvements have been introduced to enhance the optimization capacity of each of the algorithms in our work.

The paper is organized as follows: Section II introduces the basic theories related to routing in VLSI circuits, swarm intelligence and the three algorithms namely: ABC, PSO and FA. Section III illustrates the problem and how the algorithms have been applied to solve the same with the help of their pseudocodes. Section IV presents the experimental setup and the relevant parameters and Section V presents the experimental results. The paper concludes with in Section VI.

II. PRELIMINARIES

A. Routing in VLSI Systems

Routing in integrated circuits (ICs) is done after the placement phase, which determines the location of each active elementor component of an IC, on a PCB. The task of all routers is the same, working upon some pre-existing polygons consisting cells and their corresponding terminals, known as pins on cells, and optionally upon some pre-existing wiring called pre-routes. Each of these polygons are associated with a net. The primary task of the router is to create geometries such that all terminals assigned to the same net are connected, no terminals assigned to different nets are connected, and all design rules are obeyed. The first step in routing is to determine an approximate course for each net, known as global routing which may optionally include layer assignment. Global routing limits the size and complexity of the following detailed routing steps, which can then be accomplished grid square- by-grid square.

B. Swarm Intelligence

'Swarm' as a term is used for an aggregation of animals like fishes, birds and insects like ants, termites and bees performing collective behaviour. The individual agents of such swarms behave without supervision and each of them has a stochastic behaviour due to his/her perception in the neighbourhood. Swarm Intelligence (SI) can be defined as the collective behaviour of swarms which are decentralized and self-organized. Well known examples of SI are bird flocks and colony of social insects like ants and bees. The intelligence of swarm lies in the network of social interactions between the agents themselves and between the agents and their environment. SI is increasingly becoming an area important for research in engineering, science, statistics, bioinformatics, operational research as well as many other disciplines. This is primarily due to the fact that intelligent swarms can solve problems like finding food, dividing labour among nest mates, building nests, etc. which have important counterparts in several real world engineering areas. Two important SI approaches were proposed in the 1990s: one based on colonisation of ants, referred to as ACO by Dorigo et al. [6] and the other based on bird flocking, known as PSO by Kennedy and Eberhart [5]. Both the optimization approaches have been extensively studied over the next two decades and several refined and modified versions have been proposed and successfully applied to find the solution of complex real-world problems across various domains. SI based algorithms proposed in this decade itself include ABC: based on the forging behaviour of honeybees and proposed by Karaboga and FA: based on the path planning of fireflies. Millonas [9] defined some principles in 1999 that need to be satisfied by a swarm to have an intelligent behaviour:

- The proximity principle: a swarm should be able to do simple space and time computations.
- The quality principle: a swarm should be able to respond to quality factors of the environment like the quality of foodstuffs or safety of location.
- The principle of stability: a swarm should not change its mode of behaviour for every fluctuation of the environment.
- The principle of adaptability: a swarm 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, developed by Karaboga [7], uses the intelligent foraging behaviour of honeybees. In this 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:

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

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:

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

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:

$$p_i = \frac{\text{fitness }_i}{\sum \text{fitness }_i} \ldots (3)$$

Here$\text{fitness}_i$is the fitness of the solution or 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.

D. Particle Swarm Optimization

Particle swarm optimization is an efficient and well-performing algorithm in the field of optimization, developed by Kennedy and Eberhart in 1995. It is a population based optimization technique. When aschool of birds are searching for food in an unknown space, they share their information within the group, and thus the moving direction of each bird affected by both the global optimist position information and the individual optimist position information. This forms the basis of the algorithm.

In PSO the potential solution to a problem is represented by a particle in the search space, with its position represented by a vector,x_{id} and the rate of change of its position i.e. velocity represented by v_{id} in a D-dimensional search space.The fitness of each particle is represented by the quality of its position.The particles fly around in the search space with a certain velocity, which (both direction and speed) depends on its own historical best position obtained so far as well as the best position obtained in the swarm. The position as well as the velocity of a particle are modified in an iterative process. The local best position, p_{id} of a particle and the global best position of the swarm, g_{id} is memorized in each iteration. The velocity and position of a particle are obtained by the following equations:

$$v_{i+1} = wx_i + \varphi_1\beta_1(p_i - x_i) + \varphi_2\beta_2(g_i - x_i) \quad (4)$$

$$x_{i+1} = v_{i+1} + x_i \qquad (5)$$

Here the constants, $\varphi_1$and $\varphi_2$are responsible for the influence of the individual particle’s own knowledge (φ_1) and that of the group (φ_2), both usually initialised to 2. The variables β_1&$\beta_2$are uniformly distributed random numbers in [0,1] and $p_i$and $g_i$are the particle’s previous best position and the swarm’s previous best position respectively.The inertia weight w controls the balance between local and global search performed by the algorithm and is calculated by:

$$w = \left\{3 - e^{-\frac{S}{200}} + \left(\frac{R}{8} \times D\right)^2\right\}^{-1} \quad (6)$$

Here, S is the size of the swarm, D is the dimension of the optimization problem and R is the relative quality of each solution normalised to [0,1].

PSO is a metaheuristic algorithm as it makes few or no assumptions about the problem being optimized and can search very large candidate solution spaces. However, meta-heuristics such as PSO do not guarantee an optimal solution will ever be found. More specifically, PSO does not use the gradient of the problem being optimized, which means PSO does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods. PSO seems to be very good in terms of speed of convergence. This might be due to the effect from generating completely different random numbers to be used in the iterative procedures of the algorithm. PSO can therefore, also be used on optimization problems that are partially irregular, noisy, change over time, etc. However, there are certain disadvantages [10] in this algorithm. The method easily suffers from the partial optimism, which causes the less exact at the regulation of its speed and the direction. Also the method cannot work out problems of scattering and optimization and problems of non-coordinate systems such as the solution to energy fields and the moving rules of the particles in the energy field. However, due to its many advantages including its simplicity and ease of implementation, the algorithm can be used widely in the fields of function optimization, model classification, machine study, neutral network training, signal procession, vague system control, automatic adaptation control and etc.

E. Firefly Algorithm

The Firefly Algorithm (FA) is a heuristic, stochastic optimization algorithm, based on the social behaviour of fireflies. They communicate with each other using varying flashing patterns that uses bioluminescence. The two fundamental functions of such flashes are to attract mating partners (communication) and to attract potential prey. In addition, flashing may also serve as a protective warning mechanism.

There are certain characteristics in the flashing of real fireflies, such as fireflies are unisex, and thus, every firefly will move towards the more attractive and brighter ones regardless their sex. The relative attractiveness proportional to firefly‘s brightness. For a pair of flashing fireflies the one who is less bright will move towards the brighter one. Attractiveness will decrease with increasing space distance between fireflies. If there are no brighter fireflies than a firefly itself, it will move randomly in the search space. And finally the brightness of a firefly is somehow determined by the fitness function value of the objective problem.

In the FA, the process of optimization is simulated with the attraction and the position changing of the individual fireflies. The fitness of the solution is determined by the advantages and disadvantages of the fireflies' location, and the process of finding good feasible solutions is presented by the process of the fireflies searching the better locations in the sky in an iterative process. There are two important factors involved: the variation of light intensity and the formulation of the attractiveness. For simplicity, suppose that the attractiveness of a firefly is determined by its brightness and which in turn is associated with the objective function. The higher is the brightness and the better is the location, the more fireflies will be attracted to the direction. If the brightness's are equal, the fireflies will move randomly in the search space. The light intensity decreases monotonically as the distance from the source increases and thus attractiveness also decreases monotonically.

In the simplest case for maximum optimization problems, the brightness I of a firefly for a particular location x could be chosen as $I(x)$. The attractiveness β is relative, and it should be judged by the other fireflies. Thus, it will differ with the distance $r_{i,j}$between firefly,i and firefly,j. In addition, light intensity decreases with the distance from its source, and light is also absorbed by the media so light intensity depends on the absorption coefficient of the media. So attractiveness will also vary with the varying degree of absorption.

In the simplest form, the light intensity I(r) will vary according to the inverse square law:

$$I(r) = {}^{I_s}/_{r^2} \qquad (7)$$

Here $I_s$represents the intensity at the source. For a medium with a fixed light absorption coefficient γ, the light intensity I will vary with the distance r. That is:

$$I = I_o e^{-\gamma r} \qquad (8)$$

Here I_o is the initial light intensity, In order to avoid the singularity at r = 0 in the expression, the combined effect of both the inverse square law and absorption can be approximated as the following Gaussian form.

$$I = I_o e^{-\gamma r^2} \qquad (9)$$

A firefly's attractiveness is proportional to the light intensity seen by adjacent fireflies. So we can now obtain the attractiveness β of a firefly as:

$$\beta = \beta_o e^{-\gamma r^2} \qquad (10)$$

In the above equation,β_o is the attractiveness at r = 0. It is often faster to calculate$1/(1 + r^2)$ than an exponential function. So the above function, if necessary, can be approximated as:

$$\beta = \frac{\beta_o}{1+\gamma r^2} \qquad (11)$$

Characteristic distance is the distance r(=Γ=1/γ) over which the attractiveness changes significantly from β_o to $\beta_o e^{-1}$ for (10) and $\beta_o/2$ for equation (11).The parameter γ now characterizes the contrast of the attractiveness, and its value is crucial in determining the speed of the convergence and how the FA algorithm behaves. In theory,$\gamma \in [0, \infty)$, but in actual practice, γ = O(1) is determined by the characteristic length Γ of the system to be optimized. Thus, for most applications, it typically varies from 0.1 to 10.

The distance between any two fireflies, i and j at x_i and x_j, respectively is the Cartesian distance:

$$r_{i,j} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d}(x_{ik} - x_{jk})^2} \qquad (12)$$

Here is $x_{i,k}$ the kth component of the spatial coordinate, x_i of the ith firefly. In 2-D case, we have:

$$r_{i,j} = \sqrt{(x_i - x_j\ )^2 - (y_i - y_j)^2} \quad (13)$$

The movement of the firefly i is attracted to another more attractive (brighter) firefly, j is determined by:

$$x_i = x_i + \beta_o e^{-\gamma r_{i,j}^2}(x_j - x_i) + \alpha\varepsilon_i \qquad (14)$$

Here the second term is due to the attraction and the third term is randomization with α being the randomization parameter, and is a vector of random numbers being drawn from a Gaussian distribution or uniform distribution. For example, the simplest form is ε_i could be replaced by (rand − ½) where rand is a random number generator uniformly distributed in [0, 1]. For most of our implementation, we can take β_o =1 and $\alpha \in [0, 1]$.

This algorithm has many advantages such asa high convergence rate. Ever agent i.e. firefly works individually and finds a better position for itself in consideration with its current position as well as the position of other fireflies. Thus, it escapes from the local optima and finds a global optimum ina lesser number of iterations. And also it is a robust algorithm. It has two superiorities [11]: automatic subdivision and random reduction. The fireflies in the FA can automatically divide into subgroups so that they swarm around the multimodal optima. This makes it possible for the algorithm to find all thepossible global optima simultaneously. Thus, the algorithm is particularly suitable for nonlinear and multimodal optimization problems. In addition, the randomness is graduallyreduced, using a similar strategy as that used in simulated annealing and this speeds up convergence when the global optimality is approaching.

III. Problem Formulation And Application of Proposed Algorithms

The search space or the grid graph can be represented by a 2D matrix where each element represents a node of the grid graph as shown below. The elements of the matrix corresponding to the nodes that need to be connected, have a value of 1; all the other elements being zero. Thus, it can be seen that all the algorithms require their discrete versions to be implemented as only 0 and 1 are allowed. 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 generality from a grid graph. 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. The algorithms have been explained below with the help of their pseudocodes. The variables, x and n are used to globally denote the agents and the population of the swarm respectively.

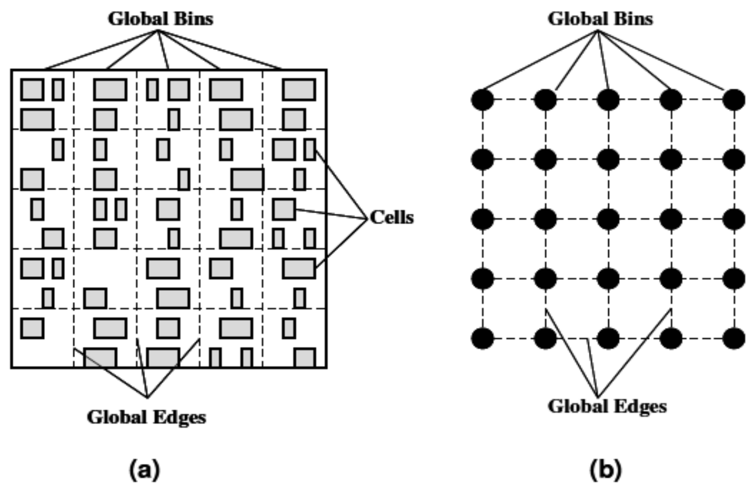

Figure 1. Global bins and corresponding global routing grid graph.

The agents of the swarm in each of the algorithms are initialised based on the matrix used to represent the grid graph.

$$\text{grid\_graph} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots\cdots & a_{1,q} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots\cdots & a_{2,q} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots\cdots & a_{3,q} \\ . & . & . & & . \\ . & . & . & & . \\ . & . & . & & . \\ a_{p-2,1} & a_{p-2,2} & a_{p-2,3} & \cdots\cdots & a_{p-2,q} \\ a_{p-1,1} & a_{p-1,2} & a_{p-1,3} & \cdots\cdots & a_{p-1,q} \\ a_{p,1} & a_{p,2} & a_{p,3} & \cdots\cdots & a_{p,q} \end{bmatrix}$$

A. Pseudocode for ABC

```
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
```

B. Pseudocode for PSO

```
Objective function $f(x)$ = weight of grid_graph
Initialise swarm of particles $x_i (i=1,2,..,n)$
Calculate $f(x_i)$ for all i and then calculate w
Set $p_i = f(x_i)$ and $g_i$ = minimum of all $f(x_i)$
While (iter<maxgen)
   For i = 1:n
      Calculate $v_{i+1}$ using eq. (4)
      Calculate $x_{i+1}$ using eq. (5) and modification in [12]
      Evaluate $f(x_i)$
      If $(f(x_i)<p_i)$
            Replace $p_i$ with $f(x_i)$ and update $p_{best}$
      End If
      If $((f(x_i)<g_i)$
            Replace $g_i$ with $f(x_i)$ and update $g_{best}$
      End If
   End for
   For i = 1:n
      Calculate w using eq. (6)
```

```
    End for
    iter = iter + 1
End while
```

C. Pseudocode for FA

```
Objective function f(x) = weight of grid_graph
Initialize a population of fireflies $x_i$(i=1,2,..,n)
Define light absorption coefficient $\gamma$
While (iter<maxgen)
    For i = 1:n
        For j = 1:i
            Light intensity $I_i$ at $x_i$ is determined by $f(x_i)$
            If ($I_i > I_j$)
                Move firefly i towards j in all d dimensions. (eq. (12))
            Else
                Move firefly i randomly
            End if
            Attractiveness varies with distance r. (eq. (11))
            Evaluate new solutions and update light intensity
        End for j
    End for i
    Rank the fireflies and find the current best
      iter = iter + 1
    End while
```

IV. Experimental Setup

A total of two different sets of coordinates for terminal nodes are randomly generated for connection. Each algorithm: ABC, PSO and FA, is applied on the two different setups to optimally connect all the terminal nodes mentioned in the sets. Each experiment is run 30 times for each algorithm. The population of swarm is set to 200 and the number of iterations to 100 for each of the SI algorithms. All the terminal node coordinates have been shown in Table I.

TABLE I. Experimental Coordinates for Terminal nodes

EXPERIMENT 1: COORDINATES										
NO.	1	2	3	4	5	6	7	8	9	10
X	58	32	67	27	99	46	23	16	54	08
Y	80	53	61	66	75	09	92	83	01	45
EXPERIMENT 2: COORDINATES										
NO.	1	2	3	4	5	6	7	8	9	10
X	11	31	82	09	26	44	19	41	87	55
Y	97	78	87	40	81	92	27	14	58	15

V. Results and Discussion

The results obtained have been summarised in Table II. The normalised values have been depicted graphically in figures 2 and 3. It can be seen from the results that theFA fares better than ABC whose performance in turn exceeds that of PSO when it comes to optimally routing or interconnecting the given nodes. The mean wirelength obtained over 30 runs of each of the algorithms also follow the same trend. The standard deviation values obtained points to the fact that there is much variation in the results obtained by the FA compares to the ABC and the PSO in both the cases.This means that the randomness among the results obtained in each run is highest for the FA and least for the ABC, the PSOcoming in between. It can be thus safely concluded that the ABC is the most robust among the three algorithms used here. The proposed FA betters the results of PSO by 25% and 30% respectively in the two cases where AC can improve by only 11% and 15% respectively. The FA thus is twice as effective as the ABC. However, the computational load results discredit the FA's performance to a large extent. While the ABC is uses a maximum 10% surplus of time, the FA requires more than 400% of PSO's runtime in each case to optimally interconnect the given nodes.

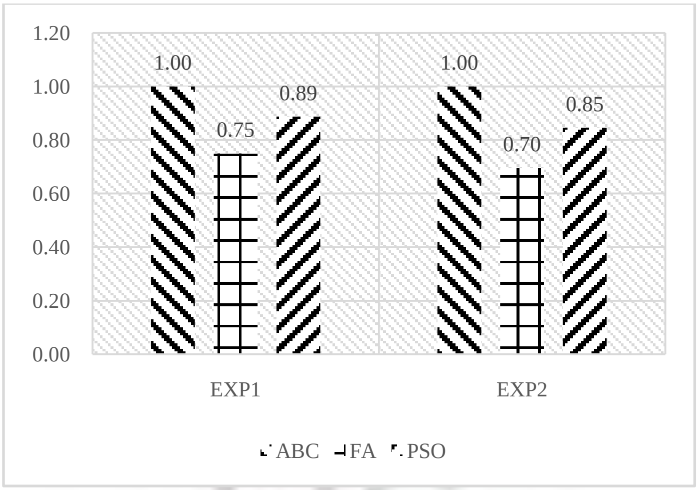

Figure 2. Normalised Minimum Wirelengths Obtained

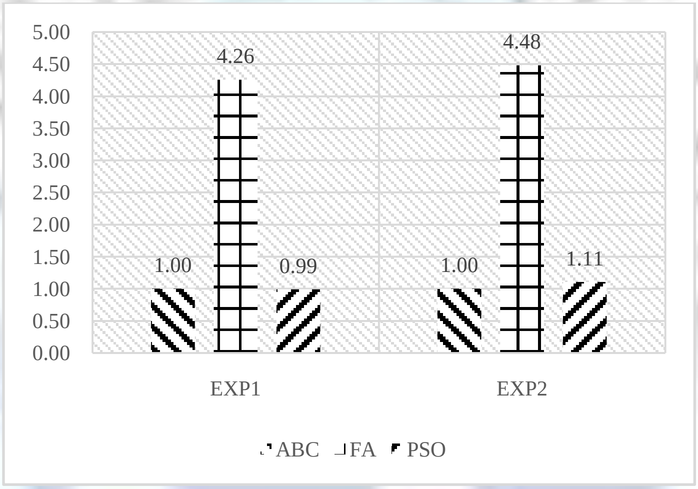

Figure 3. Normalised Computaional Load

VI. CONCLUSIONS

Routing, both global and local, has a significant impact on the entire chip performance and efficient routing algorithms (with further modifications leading to improved performance) [12] are obviously the key to success in the future in the domain of VLSI physical design. In this work, our main objective has been to obtain the minimumpossible wirelength with the help of a global routing algorithm inspired from fireflies. We have presented an efficient FA based global routing approach and have compared its performance against that of ABC and PSO. Prim's Algorithm is used to find the weight of the Steiner trees obtained. It has been suitably modified to work on the incidence matrix and not the weight matrix and also to deal with rectilinear distances instead of only Euclidean distances. The results obtained show that the proposed approach has a potential for further extension. The results of the experiments carried out, demonstrate the feasibility of our algorithm for implementation in VLSI routing optimization and it also has the flexibility to be used in much more complex situations in the future dealing with variable interconnect weights and obstacles in the routing path.Based on simulation results, the solution quality and reliability show the superiority of the FA over the other approaches.

The proposed algorithm however lacks in robustness when compared to the ABC. Also a glaring chink in its armour is its premium computational load requirement. The presented FA has already been modified by the authors of this paper and they have managed to reduce 50% of the time requirement as opposed to their previous work. Further research into this algorithm will hopefully lead to the reduction of runtime for this particular case. Once it becomes comparable to the other evolutionary and swarm based algorithms w.r.t. to the time required to return an optimal result, it will prove to be one of the best solutions to the problem dealt herein.

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

[6]. 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

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

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

[9]. M M Millonas, "Swarms, phase transitions, and collective intelligence," C. G. Langton, Ed., Artificial Life III, Addison Wesley, Reading, MA, 1994.

[10]. Qinghai Bai, "Analysis of particle swarm optimization algorithm," Computer and Information Science, vol. 3, no. 1, pp. 180-184, 2010.

[11]. XS Yang, Seyyed Soheil Sadat Hosseini, and Amir Hossein Gandomi, "Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect," Applied Soft Computing,vol. 12, no. 3, pp. 1180-1186, 2012.

[12]. 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.

TABLE II. EXPERIMENTAL RESULTS

EXPERIMENT 1: RESULTS											
Best Value			Mean Value			Standard Deviation			Computational Load (in sec)		
ABC	FA	PSO	ABC	FA	PSO	ABC	FA	PSO	ABC	FA	PSO
213	180	240	223.63	215.43	255.57	4.9023	14.0631	6.6679	48.298	207.568	48.782
EXPERIMENT 2: RESULTS											
Best Value			Mean Value			Standard Deviation			Computational Load (in sec)		
ABC	FA	PSO	ABC	FA	PSO	ABC	FA	PSO	ABC	FA	PSO
186	153	220	207.17	184.50	244.37	10.1163	15.0991	14.021	51.543	208.594	46.551