A Novel Particle Swarm Optimization approach for VLSI Routing

Abhinandan Khan
Dept. of Electronics & Telecommunication Engineering
Jadavpur University
Kolkata, India
khan.abhinandan@gmail.com

Sulagna Laha
Dept. of Electronics & Telecommunication Engineering
Jadavpur University
Kolkata, India
lahasulagna@gmail.com

Subir Kumar Sarkar
Dept. of Electronics & Telecommunication Engineering
Jadavpur University
Kolkata, India
su_sircir@yahoo.co.in

***Abstract*—Rapid advances in VLSI technology has increased the chip density by constantly increasing the number of constituents on a single chip, as well as decreasing the chip feature size. In such a complex scenario the primary objective is to limit the power-delay product of the system. It can be done by reducing the interconnect delay by optimizing the wire lengths i.e. by the proper interconnection of all the nodes. The minimum cost of interconnection of all nodes can be found by a Rectilinear Steiner Minimal Tree (RSMT) formed by the nodes. The problem of finding a RSMT is an NP-complete one. Particle Swarm Optimization (PSO) is an efficient swarm intelligence algorithm which boasts of fast convergence and ease of implementation, capable of solving such a problem. This paper presents a novel discrete particle swarm optimization (DPSO) to solve the NP-complete problem i.e. finding the RSMT. A modified Prim's Algorithm has been adopted for the purpose of finding the cost of the RSMT. A unique modification to the traditional PSO has been done by introducing the Mutation operation of Genetic Algorithm (GA) which produces up to 20% reduction in the wire lengths or cost of interconnections. Two versions of the DPSO algorithm – one with linearly decreasing inertia weight and another with self-adaptive inertia weight - have been employed and their results have been compared. Comparisons have also been made between the results available from recent work and our algorithm and the latter has established itself to superior in optimizing the interconnect lengths and thereby finding the lowest wire lengths.**

Keywords—DPSO, RSMT, Mutation, VLSI, Routing

I. INTRODUCTION

The number of transistors that can be fabricated on a single chip is continuously increasing with the rapid advances and improvements in VLSI technology. This combined with the continuous shrinkage in the chip feature size has led to very high chip densities. Keeping pace with these advancements, switching speeds are also uninterruptedly improving.

However, the major bottleneck in achieving the advantages of the aforementioned improvements is the interconnect delay. Thus, arises the essential need for efficient interconnect routing algorithms in VLSI physical design. Minimizing the length of interconnects is the only way to reduce the interconnect delay when the width of interconnects and other feature parameters are assumed to be non-varying. Therefore achieving the minimum interconnect length becomes the ultimate goal of any efficient routing algorithm. This complex problem provides us with the opportunity to use a fast, efficient and robust optimization technique, the Particle Swarm Optimization [1] in the field of VLSI routing.

While, the standard PSO is designed for continuous problems, the discrete PSO (DPSO) [2] achieves promising results for multiple destination routing problems with discrete constraints. It is now being increasingly used for the RSMT [3] problem and hence also being applied in VLSI Routing optimization problems. This paper proposes a routing algorithm based on DPSO to find the optimum interconnect length and hence minimize the interconnect delay. Our algorithm is novel in the sense that mutation has been incorporated and the algorithm has been applied for routing optimization in VLSI. Though such an algorithm has already been employed for the RSMT problem [3], this is the first in the field of VLSI routing optimization.

The rest of this paper has been organised as follows. In section II, the basic theories have been explained. Our DPSO algorithm has been explained step by step in section III. Section IV gives the experimental results. Finally the paper concludes with Section V.

II. PRELIMINARIES

A. Particle Swarm Optimization (PSO)

PSO is a robust stochastic optimization technique based on the movement and intelligence of swarms. It applies the concept of social interaction to problem solving. It was developed in 1995 by James Kennedy (a social-psychologist) and Russell Eberhart (an electrical engineer) [1]. It uses a number of agents (particles) that constitute a swarm moving around in the search space looking for the best solution. Each particle keeps track of its coordinates in the solution space which are associated with the best solution (fitness) that has been achieved so far by that particle. This value is called the personal best (*pbest*). Another best value that is tracked by the PSO is the best value that has been obtained so far by any particle in the neighborhood of that particle. This value is called the global best (*gbest*). The basic concept of PSO lies in accelerating each particle toward its personal best and the global best locations, with a random weighted acceleration at

978-1-4673-4529-3/12/$31.00 ©2012 IEEE

each step. Each particle is treated as a point in an N-dimensional space which adjusts its flight according to its own flying experience as well as the flying experience of other particles.

Considering a swarm S containing 'n' particles in a D-dimensional continuous solution space, each particle is represented by its D–dimensional position vector X_i, while the velocity of the particle is represented by another D-dimensional vector, V_i. The position of any particle X_i represents a possible solution to the optimization problem under study, while the velocity V_i gives the rate of change of position of a particle for the next iteration. A fitness value for a particle position can be obtained by evaluating the objective function at the interested particle position. The fitness value gives an indication of the quality of the particle position as a solution candidate to the optimization problem under study. The best position of a particle at a given iteration stage corresponds to the position that gives the best fitness value among all its historical positions so far. Mathematically, the entire process is summarised by the following two equations:

$$\mathbf{V_{i,j+1} = w*V_j + c_1*r_1*(pbest - X_{i,j}) + c_2*r_2*(gbest - X_{i,j}) \dots (i)}$$

$$\mathbf{X_{i,j+1} = X_{i,j} + V_{i,j} \dots (ii)}$$

Where w is the inertia weight, r_1 and r_2 are two random numbers between 0 & 1; c_1 and c_2 are both positive constants, which are known as acceleration coefficients.

This standard form of PSO was designed for continuous problems only. The concept of discrete PSO or DPSO [2] came into existence in 1997 which is best suited for multi-dimensional routing problems like ours.

The inertia weight w can be updated using two completely different techniques:

- Linearly decreasing inertia weight: In this technique the inertial weight decreases, as the name suggests, linearly from a set initial value (w_{max}) to a final value (w_{min}) depending on the current generation or iteration (*iter*) and the maximum number of iterations allowed (*maxiter*). Mathematically,

$$\mathbf{w = w_{max} - [(w_{max} - w_{min}) * iter] / maxiter \dots (iii)}$$

- Self-adaptive inertia weight: This technique [4] for the update procedure of the inertial weight depends on the swarm size (S), the dimension size of the solution search space (D) and the fitness of the solution generated during each iteration (R). For complex large dimensional problems PSO has a tendency to be trapped at local optima, leading to premature convergence. A large inertia weight helps the algorithm to escape from the local optima. On the other hand for low dimensional problems, a small inertia weight reinforces the local search capability enhancing convergence. Thus, we it can be seen that the inertia weight has an underlying relation with swarm size, fitness and dimension of search space which can be modelled mathematically as,

$$\mathbf{w = [3 - \exp(-S/200) + (R/8*D)^2]^{-1} + 0.8 \dots (iv)}$$

An additional 0.8 is added as our problem is pretty complex and has a relatively high dimension size of search space. Experiments as well as previous research [5] have shown that this is the ideal value for the specific type of problem dealt herein.

B. Rectilinear Steiner Minimal Tree (RSMT)

The Steiner tree problem is essentially a special case of multidimensional routing that can be defined as: for any given set T of n terminal points, the objective is to find another set S of additional points called Steiner points such that the length of a minimum rectilinear spanning tree or RSMT is minimized. As already mentioned our ultimate goal is the reduction of interconnect length, the RSMT problem becomes our fundamental problem also. The RSMT problem has been proved to be NP-complete by Garey and Johnson [6] which implies that a polynomial-time algorithm to compute an RSMT is unlikely to exist.

III. OUR ALGORITHM APPLIED TO THE ROUTING PROBLEM

In this section, our novel DPSO with mutation (DPSOM) algorithm has been introduced to solve the NP-complete problem i.e. the RSMT problem. The operations of PSO have been somewhat redefined to design a novel routing algorithm based on an improved DPSO with mutation.

This algorithm is elaborated as follows:

- The entire search space is represented by a null matrix, X of n-by-n dimensions.It has been assumed n = 100 in this paper.

- The terminal nodes are taken as input from the user or are generated randomly in the form of x and y-coordinates. The elements of the matrix with this coordinates are set to be equal to 1. Two sets of coordinates have been randomly generated in this work.

- The matrix is then reduced by deleting those rows and columns which do not have any terminal nodes. This is done solely to reduce the computational load. The matrix after reduction looks like this.

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

Proper steps are taken to store the indexes of the rows and columns that are not deleted, as it is required to calculate the weight of the paths between each terminal nodes.

➢ A swarm of 'm' particles is generated where each particle is this reduced matrix with some elements, selected at random, set to 1. The original terminal nodes are kept unchanged. One such random particle is:

X_{20} =

0	0	0	0	0	1	0	0	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	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	1	0	1
1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	0

➢ These 'm' matrices are both the position and the velocity as well as the *pbest* of each of these 'm' particles for the first iteration. The cost of these solutions are calculated and the matrix with the least cost is assigned as the *gbest* for the first iteration.

➢ The fitness value of each particle is calculated by using a modified version of Prim's Algorithm.

➢ The calculated fitness value of each of the particles is compared with the particle's previous best value, *pbest*. If the current value is better than *pbest*, then the current value is set to be the new *pbest* for each of the particles.

➢ Similarly, the *pbest* of all the particles is compared to find the minimum among all the *pbest*-s. If this value is lower than the previous global best value, the *gbest* value is also updated to the new minimum value.

➢ The inertia weight is updated according to either equation (iii) or equation (iv) depending on which technique we are applying. In this paper, it has been assumed w_{max} = 0.95 and w_{min} = 0.40.

➢ The velocity of each of these m particles is updated according to equation (i) for the next iteration.

➢ However equation (ii) is not used to update the postion of all particles for the next iteration. Here we use a novel approach for updating the postion by selecting only some bits of the postion matrix of the particles from the previous generation. This process is analogous to the mutation process used in GA. This mutation process increases the efficiency as well as the robustness of standard DPSO.

IV. RESULTS AND DISCUSSION

Two independent experiments have been performed with two randomly selected sets of 10 terminal points – one a bit more sparsely situated in the 100-by-100 search space than the other. Our novel DPSO algorithm with mutation has been applied to optimally interconnect the terminal points of these two sets and return the minimum cost. Each experiment is run 25 times.

The two sets of terminal points on which the experiments have been conducted are shown in Table I and Table II. These two sets have been randomly generated.

The algorithm has been applied using various values of the different parameters like swarm size and the maximum of iterations. Three swarm sizes have been used: 150, 200 and 250. The maximum number of iterations has been set as 75, 100 and 125 respectively for the three swarm sizes. The experiments have been performed using both the inertia weight update techniques individually which gives us four sets of results. This has been purposefully done to observe the performances of both the inertia weight update techniques when used in conjunction with mutation. The parameters and their used values are tabulated in Table III.

TABLE I. COORDINATES FOR EXPERIMENT 1

No.	1	2	3	4	5	6	7	8	9	10
X	81	91	63	76	68	17	99	54	21	27
Y	11	65	50	89	83	62	42	35	57	79

TABLE II. COORDINATES FOR EXPERIMENT 2

No.	1	2	3	4	5	6	7	8	9	10
X	65	99	39	22	61	55	29	58	85	94
Y	48	40	96	63	89	91	78	69	35	58

TABLE III. PARAMETERS FOR EXPERIMENTAL WORK

LDW-DPSOM		SAW-DPSOM	
SWARM SIZE	ITERATIONS	SWARM SIZE	ITERATIONS
150	75	150	75
200	100	200	100
250	125	250	125

The four sets of results obtained have explicitly been shown in Tables IV and V. The minimum cost and the mean cost over the 25 runs of the experiment are only shown.

TABLE IV. RESULTS OF EXPERIMENT 1

LINEARLY DECREASING WEIGHT DPSOM			
	S = 150 IT = 75	S = 200 IT = 100	S = 250 IT = 125
Min	220	224	225
Mean	247.92	247.72	245.68
SELF ADAPTIVE WEIGHT DPSOM			
	S = 150 IT = 75	S = 200 IT = 100	S = 250 IT = 125
Min	226	219	223
Mean	248.84	246.32	248.68

TABLE V. RESULTS OF EXPERIMENT 2

LINEARLY DECREASING WEIGHT DPSOM			
	S = 150 IT = 75	S = 200 IT = 100	S = 250 IT = 125
Min	182	182	186
Mean	195.36	195.76	197.00
SELF ADAPTIVE WEIGHT DPSOM			
	S = 150 IT = 75	S = 200 IT = 100	S = 250 IT = 125
Min	187	175	183
Mean	195.48	193.68	194.36

Figures 1 and 2 graphically compare the results obtained from the first experiment. While Fig. 1 compares the minimum cost of interconnects obtained by the two different techniques for inertia weight update, Fig. 2 compares the mean values over the 25 runs for the first experiment only. Figures 3

and 4 similarly compare the results obtained from the second experiment graphically.

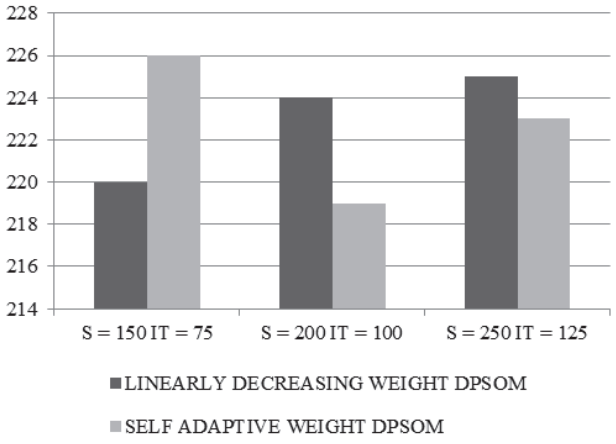

Figure 1. Comparison of Minimum Cost Obtained using different parameter values

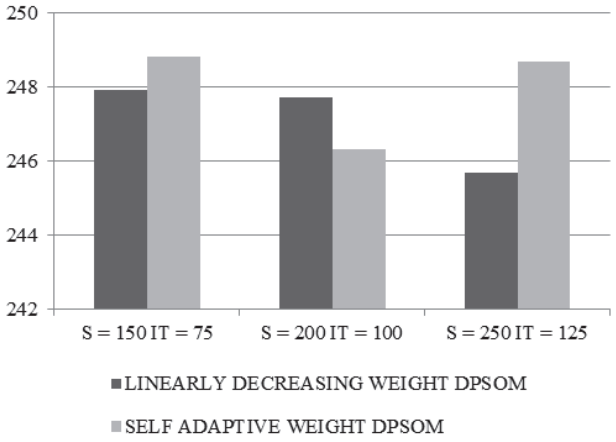

Figure 2. Comparison of Mean Cost Obtained using different parameter values

It can be seen from the results that the self-adaptive weight DPSOM with mutation performs better in both the cases from the point of view of minimum as well as mean interconnection costs. In our experimental runs, the self-adaptive weight DPSOM has outperformed the linearly decreasing weight DPSOM with swarm size of 200 and maximum number of iterations set to 100 in both the cases. Thus we can say that this is the optimum setting for the self-adaptive weight DPSOM. Moreover the self-adaptive weight DPSOM also performs in a more reliable manner in both cases giving the best results with the same settings which the linearly decreasing weight DPSOM does not. The linearly decreasing weight DPSOM gives the best minimum cost for the first experiment with swarm size set to 150 and maximum number of iterations set to 75 but gives the best mean value with them being set to 250 and 125 respectively. In the second experiment the linearly decreasing weight DPSOM achieves the best minimum value with swarm size set to 150 and maximum number of iterations set to 75 as well as them being set to 200 and 100 respectively. However the algorithm is capable of achieving the best mean value with only one parameter settings .

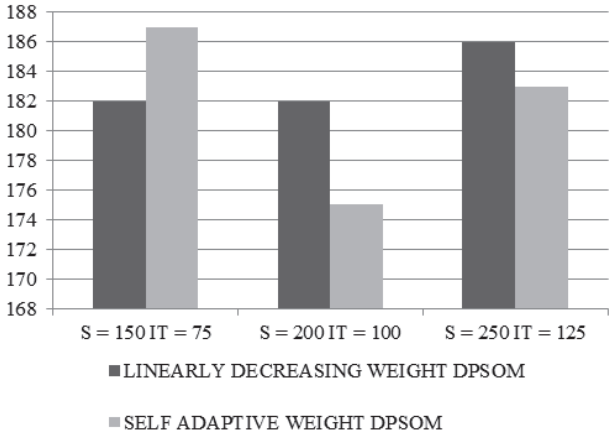

Figure 3. Comparison of Minimum Cost Obtained using different parameter values

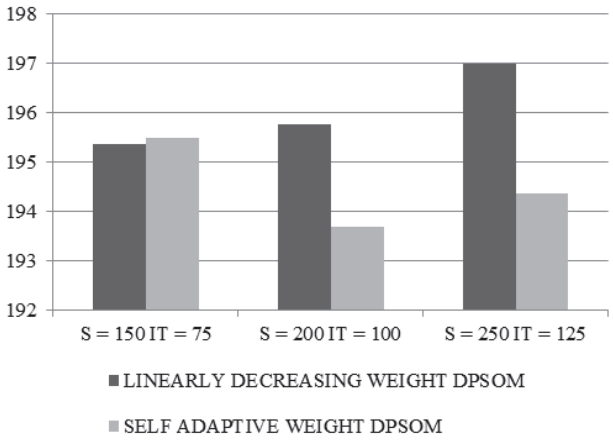

Figure 4. Comparison of Mean Cost Obtained using different parameter values

Thus, it can be deduced that the self-adaptive weight DPSOM works more efficiently and reliably when we need to find out the optimum routing cost. It can also be safely deduced that the best performance of the self-adaptive weight DPSOM is achieved with a swarm size of 200 and 100 iterations. Also the comparison of our work with [5] where the DPSOM was used without incorporating mutation has been shown in Fig. 5.

V. CONCLUSION

With the rapid advancements in all aspects of VLSI Physical design, routing in ICs has a great impact on the entire chip performance. Efficient routing algorithms are obviously the key for future success in VLSI physical design. This paper presents an application of Discrete Particle Swarm Optimization with Mutation to find the optimum routing cost.

A novel update operation for the position of a particle in the swarm has been adopted. The parameters have been varied to find out the optimum settings for the most efficient results that our algorithm can obtain. Prim's Algorithm has also been modified accordingly so that it can work with only the incidence matrix and does not require the weight matrix and can also deal with rectilinear distances instead of only Euclidean distances.

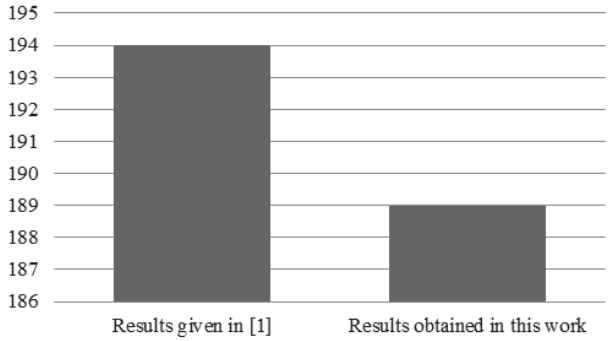

Figure 5. Comparison of Minimum Cost Obtained using different parameter values

The results of the experiments that have been carried out, demonstrate the feasibility of our algorithm for implementation in VLSI routing optimization. This algorithm also has the flexibility to be used in much more complex situations in the future dealing with variable interconnect weights and also avoiding obstacles in the routing space.

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization", *IEEE International Conference on Neural Networks* (Perth, Australia), IEEE Service Center, Piscataway, NJ, vol. 4, 1995, pp. 1942-1948.

[2] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*, 1997, pp 4101~4109.

[3] W. L. Zhong, J. Huang, and J. Zhang, "A Novel Particle Swarm Optimization for the Steiner Tree Problem in Graphs," *IEEE World Congress on Evolutionary Computation*, pp. 2460-2466, 2008.

[4] Chen Dong, Gaofeng Wang, Zhenyi Chen and Zuqiang Yu, "A Method of Self-Adaptive Inertia Weight For PSO" *IEEE International Conference on Computer Science and Software Engineering*, vol 1, Dec 2008, pp 1195-1198.

[5] C. Dong, G. F. Wang, Z. Y. Chen, S. L. Sun, and D. W. Wang, "A VLSI Routing Algorithm Based on Improved DPSO", *IEEE International Conference on Intelligent Computing and Intelligent Systems*, vol. 1, pp. 802-805, 2009.

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