Obstacle-avoiding Rectilinear Steiner Minimum Tree Construction Based on Discrete Particle Swarm Optimization

Yulong Shen, Qingzhen Liu
College of Electrical Engineering and Automation
Fuzhou University
Fuzhou, China

Wenzhong Guo
College of Mathematics and Computer Science
Fuzhou University
Fuzhou, China

***Abstract*—The obstacle-avoiding rectilinear Steiner minimum tree (OARSMT) problem is one of the fundamental problems in the physical design of Very Large Scale Integrated (VLSI) circuits, especially in routing, which is known to be NP-complete. It becomes more important than ever for VLSI designs which need to consider numerous obstacles such as macro cells, IP blocks and pre-routed nets. Particle Swarm Optimization (PSO) has been proved to be an efficient intelligent algorithm for optimization designs. In this paper, an improved Discrete Particle Swarm Optimization (DPSO) algorithm is proposed for solving the OARSMT problem optimally. Meanwhile, the corresponding evaluation function and the operators inspired from the Genetic Algorithm (GA) are designed. The obstacle-avoiding measure is then presented to construct an OARSMT, which avoiding obstacles by generate virtual vertexes of the real vertexes. Experimental results on several benchmarks show that the proposed method achieves good performance in routing optimization significantly.**

Keywords-OARSMT; DPSO; VLSI; routing; Steiner tree

I. INTRODUCTION

As integrated circuit (IC) technology moves to deep sub-micron (DSM) designs, a circuit may easily contain hundreds of thousands of interconnecting wires. This makes routing in Very Large Scale Integrated (VLSI) physical design difficult to manage. Construction of rectilinear Steiner minimum tree (RSMT) is one of the fundamental problems in integrated circuit computer-aided design (CAD). It has been widely used in the domain of VLSI design automation. Finding an RSMT is useful for both the detailed and global routing steps, and is also important for congestion estimations, wire length estimations and timing estimations in floorplanning or placement. The original RSMT problem assumes no obstacle in the routing region. However, with the progress of IC technology, a modern nanometer VLSI design will contain more and more routing obstacles on the chip, such as macro cells, IP blocks and pre-routed nets. Therefore, the obstacle-avoiding RSMT (OARSMT) problems become more important and received increasing research attentions.

The RSMT problem has been shown to be NP-complete [1], and the introduction of obstacles has made this problem even more complicated. So, many heuristic approaches for OARSMT have been represented in last twenty years [2]-[10]. Hwang et al. [2] introduced an extensive survey of RSMT heuristics. Kahng et al. [3] proposed the Batched Iterated 1-Steiner heuristic with an average improvement over the minimum spanning tree (MST). Hu et al. [4] proposed another non-deterministic local search heuristic, namely An-OARSMan, to solve small-scale OARSMT problems with complex obstacles. CDCTree was proposed by Shi et al. [5], this CDCTree is based on a current driven circuit model and it can obtain shorter wire length than An-OARSMan. Shen et al. [6] introduced the spanning graph. In their algorithm, an obstacle-avoiding spanning graph (OASG) was first constructed and then a clever heuristic was proposed to transform the OASG into an OARSMT. Lin et al. [7] extended the approach in [6] by identifying edges which can lead to more desirable solutions in the construction of the OASG. Li et al. [8] proposed a maze routing based approach. Their approach keeps multiple paths between vertexes until all the vertexes were reached. An MST based method was then used to select between those paths to create an OARSMT. Long et al. [9] proposed a four-step algorithm to construct an OARSMT. Very recently, Ajwani et al. [10] presented the FOARS, a top down approach for the OARSMT problem. In their work, they applied the OASG to partition a solution and constructed the OARSMT by using the obstacle-aware version of the fast lookup table based wire length estimation. In comparison with heuristics, few traditional exact algorithms have been proposed. Li et al. [11] proposed an exact algorithm for multi-terminal nets. Huang et al. [12] presented a two-phase based obstacle-avoiding full Steiner trees (OAFSTs) based on [11] by using OAFSTs to construction an OARSMT.

Particle Swarm Optimization (PSO) was first proposed by Kennedy and Eberhart [13] in 1995, and it has been proved to be an efficient intelligent algorithm for optimization designs. However, the standard PSO is designed for continuous problems, so it can not be applied to discrete problems. Since 1997 discrete PSO (DPSO) is presented [14], the DPSO achieves promising results for multiple destination routing problems, Zhong et al. [15] introduced a DPSO-STP algorithm for solving MST, different form standard PSO, the DPSO-STP includes four steps to resolve MST problem. Dong et al. [16] presented a DPSO-RA algorithm to find out an MRST. Chen et al. [17] proposed an intelligent decision algorithm based on PSO to solve Floorplanning in VLSI design. However, none of them is for OARSMT.

978-1-4244-9953-3/11/$26.00 ©2011 IEEE

The main contribution of this paper is to propose an efficient swarm intelligence algorithm based on DPSO to construct an obstacle-avoiding rectilinear Steiner minimum tree in VLSI routing, the corresponding evaluation function and the operators inspired from the Genetic Algorithm (GA) are designed. An obstacle-avoiding measure is then presented, which avoiding obstacles by generate virtual vertexes of the real vertexes, and the experimental results on several benchmarks show that the proposed method achieves good performance in routing optimization significantly. The rest of this paper is organized as follows. Section 2 the background of standard PSO algorithm is introduced and the OARSMT problem is formulated. Section 3 describes the construction of OARSMT problem, then the DPSO algorithm for OARSMT is proposed in section 4 and the experimental results and discussion are presented in section 5, followed by a conclusion in section 6.

II. THE STANDARD PSO ALGORITHM AND THE OARSMT PROBLEM FORMULATION

A. Standard PSO

The standard Particle Swarm Optimization (PSO) is a computational paradigm based on the phenomenon of collective intelligence exhibited by swarms of insects, and schools of fish, that has been applied to a variety of domains. In the standard PSO model, a swarm S contains n particles (S = 1; 2; 3; ...; n) in a D-dimensional continuous solution space. Each particle has a position X_i and a velocity V_i, both of which are D-dimensional vectors denoting a solution and moving step, respectively, then

$$X_i = (x_i^1, x_i^2, ..., x_i^n), \text{ and } V_i = (v_i^1, v_i^2, ..., v_i^n).$$

Here variable i stands for the i-th particle. The position of a particle X_i represents a possible solution to the optimization problem, while the velocity V_i denotes the change rate for the position of the i-th particle in the next iteration. A particle adjusts its position X_i through the solution space attracted by *gbest* and *pbest*, which are the best position that the whole population and the i-th particle have reached, respectively. A fitness value for a particle can be obtained by evaluating the objective function. The fitness value is an important index of the quality of the particle. A particle is updated through these two equations:

$$V_i^{k+1} = \omega V_i^k + c_1 r_1 (pbest_i^k - X_i^k) + c_2 r_2 (gbest^k - X_i^k) \quad (1)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \quad (2)$$

Where k denotes the k-th generation, ω is the inertia weight, which is help to control the influence of the pre-velocity. Both c_1 and c_2 are constant real positive number, which are used to direct the i-th particle to $pbest_i$ and *gbest*. r_1 and r_2 are random real numbers belonging to [0, 1). After updated, the new position will be evaluated again, $pbest_i$ and *gbest* may be replaced if necessary.

B. OARSMT Problem

DEFINITON 1: OARSMT Problem: let $P = \{P_1, P_2, ..., P_m\}$ be a set of pins of a m-pin net in a 2-dimensional space. $O = \{O_1, O_2, ..., O_n\}$ is a set of n obstacles. The objective of the problem is to give an obstacle-avoiding tree with the shortest length that connects all of those n pins using only horizontal and vertical lines (namely Manhattan distance). This tree is known as an OARSMT.

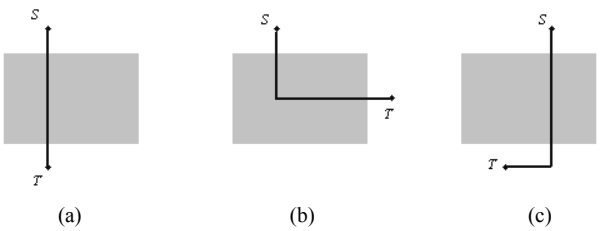


Figure 1. Three structure types that violate the constraint.

In the OARSMT problem, a pin can at the corner or on the boundary but it cannot lie inside an obstacle. No obstacle can overlap with each other, but they can be line-touch at the boundary or pin-touch at the corner. The OARSMT cannot intersect with any obstacles except their boundaries.

III. OARSMT CONSTRUCTION

The selection of Steiner points is the most important step to solve the rectilinear Steiner minimum tree (RSMT), but this paper mainly focuses on the handling of avoiding obstacles for solving ORASMT.

A. the First Step

The first step of our algorithm is to divide the pins of the net P into k groups and generate OARSMTs of each group. We choose the pins whose position are near with each other and put them into the same group, assume that $P = \{P_1^k, P_2^k, ..., P_k^k\}$, and P_i^k $(i \in 1,2,...,k)$, $(k \leqq m)$ has two or more pins. In order to ensure the quality of our algorithm, the pins contained in group P_i^k should be as fewer as possible.

The objective is to construct an obstacle-avoiding tree of P_i^k and ensure that the tree has a smaller total wire length. There are three structures that violate the constraint of OARSMT, as we can see in Fig. 1. S is the source pin, T is the terminal pin, and the rectangle region denotes obstacle on the plane. In Fig. 1 (a), (b) and (c), all of the tree edges intersect with the obstacles, while, we must construct edges between source pin $S \in P_i^k$ and each of its neighbors of P_i^k with no edge blocked by the obstacles.

For the OARSMT construction, the virtual vertex of a real vertex (in the following, only a real vertex refers to a pin and a virtual vertex not a pin) is defined as follows:

DEFINITION 2. A virtual vertex $p' \notin P$ is a neighbor of a real vertex $p \in P$ that outside or on the boundary of the obstacles intersecting with the RSMT edges.

Since an RSMT is directly a shortest path between these vertexes, the RSMT edges certainly pass through one of the virtual vertexes.

LEMMA 1. Given a source $S \in P_i^k$, a target $T \in P_i^k$ $(S \neq T)$,

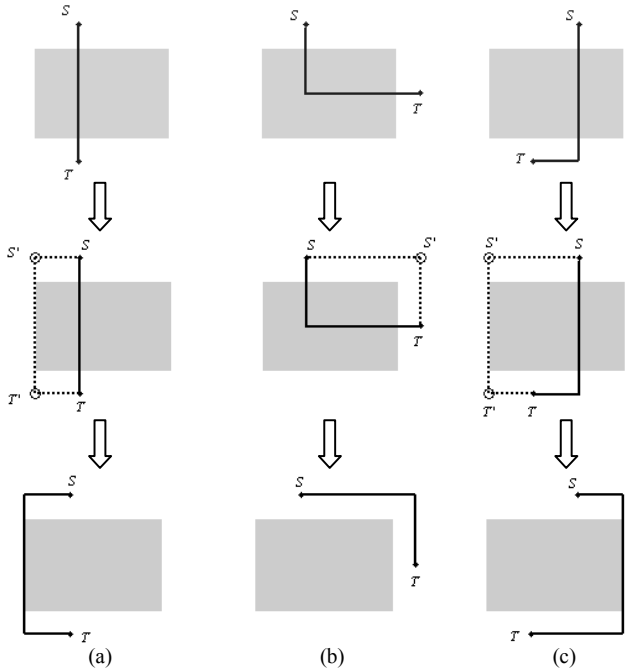


Figure 2. Obstacle-avoiding construction by generating virtual vertexes.

their must exist virtual vertexes (*P'*, *T'*) that the shortest rectilinear length $L^k = l(S, S') + l(S', T') + l(T', T)$.

In our algorithm, we take obstacle-avoiding measures when construct the RSMTs, it is clearly shown in Fig. 2. Fig. 2 (a) shows that, after the initialization step in Fig. 1 (a), vertex *S* generates a virtual vertex *S*′, which has the same horizontal coordinate with the left (or right) edge of the obstacle and the same vertical coordinate with *S*. Similarly, vertex *T* generates a virtual vertex *T*′ that has the same horizontal coordinate with the left (or right) edge of the obstacle and the same vertical coordinate with *T*. The next is to connect *S*, *S*′, *T*′ and *T* sequentially and dismantle the violate edges. In (b), virtual vertex *S*′ coincides with *T*′, while, *S*′ has the same horizontal coordinate with *T* and the same vertical coordinate with *S*, then we connect *S*, *S*′ and *T* sequentially and dismantle the violate edges. The processing mode of (c) is similar to the way (a) utilizes.

B. the Second Step

The second step of our algorithm is to use the OARSMTs of P_1^k, P_2^k, ..., P_k^k generated in the first step to construct an OARSMT of *P* with a smaller total wire length. It's similar to obstacle-avoiding construction method in step one, we see a group P_i^k as a real vertex of *P*, then connect all of the groups considered as real vertexes to construct the final OARSMT.

IV. Discrete Particle Swarm Optimization for OARSMT Problem

The standard PSO was originally proposed for continuous problems, but the OARSMT is typically a discrete problem in VLSI routing. So a DPSO algorithm is proposed in this paper, we use the novel updating method proposed in [18], which is inspired from the Genetic Algorithm (GA). A particle's velocity is changed by mutation, the individual optimal solution and global optimal solution are influenced by crossover.

A. Representation

This paper uses Prüfer number [19] to code the solution of the RSMT, and the definition of Prüfer number is elaborated as follows:

DEFINITION 3. Given a labeled tree, the vertexes of the tree are denoted by digits {1,2,…,n}. Cayley's theorem states that there are $n^{(n-2)}$ distinct labeled trees on a complete graph with *n* vertexes. Prüfer provided a constructive proof of Cayley's theorem by establishing a one-to-one correspondence between such trees and the set of all strings of $n-2$ digits. This means that there is only $n-2$ digits permutation to uniquely represent a tree with *n* vertexes where each digit is an integer between 1 and *n* inclusive. This permutation is known a Prüfer number.

We attempt to select Steiner points from the scheme of 0-1 encoding. A particle is a feasible solution in VLSI routing, a Hanan point will be selected as Steiner point if the correspond vertex is denoted by 1, not if the correspond vertex is denoted by 0. For example, a particle is represented by:

$$X_i^t = \begin{bmatrix} 2 & 5 & 4 & 5 & 3 & 100 \\ 0 & 1 & 0 & 0 & 1 & 15 \end{bmatrix}$$

The Prüfer number is {2,5,4,5,3}, and the Steiner point selection scheme is {0,1,0,0,1}, the total length of this tree is 100 and the obstacles quantities this particle avoiding is 15.

B. Optimization Objectives

DEFINITION 4. Let $G = (V, E, L)$ is a weighted, undirected and connected graph, where *V*, *E* are the set of vertexes and edges respectively, and *L* is the weight function. $L(e_{ij})$ is the weight of edge $e_{ij} \in E$, which connects vertexes *i* and *j*.

Weight cost is the total wire length here. Generally, the target of routing is to minimize the total wire length while ensuring 100% routability. In this paper, we take the wire length as our optimization objective directly, then, the evaluation function can be described as follows:

$$f = \min \sum_{e \in T} L(e) \tag{3}$$

For any solution tree *T* that is not an OARSMT, there is always at least one OARSMT *T'* with the same vertexes as *T*, and its weight cost is smaller than *T*. So it is obvious that the optimal tree must be an OARSMT, because the evaluation function ensures a better particle must have shorter total wire length than its previous generation.

C. Discrete Particle Swarm Optimization

The proposed discrete Particle Swarm Optimization in this paper introduces crossover and mutation operators inspired from Genetic Algorithm (GA) to handle Prüfer numbers which are discrete magnitude. We divide (1) into three parts by the

operator "+", the notion of mutation operator from GA is then adopted in the first part, as (4) shows.

$$\lambda_i^t = \omega \oplus F(X_i^{t-1}) = \begin{cases} F(X_i^{t-1}), r_0 \le \omega \\ X_i^{t-1}, else \end{cases} \quad (4)$$

Here F represents the mutation operator from GA with the probability of ω, and r_0 is a random number on the interval [0, 1).

The second part and the third part adopt crossover operator from GA.

$$\delta_i^t = c_1 \oplus F_1(\lambda_i^t, P_i^{t-1}) = \begin{cases} F_1(\lambda_i^t, P_i^{t-1}), r_1 \le c_1 \\ \lambda_i^t, else \end{cases} \quad (5)$$

$$X_i^t = c_2 \oplus F_2(\delta_i^t, G_i^{t-1}) = \begin{cases} F_2(\delta_i^t, G_i^{t-1}), r_2 \le c_2 \\ \delta_i^t, else \end{cases} \quad (6)$$

Here F_1, F_2 represent the crossover operators that influence self-cognitive and social-cognitive respectively, and the probabilities are c_1 and c_2. r_1 and r_2 are random numbers on the interval [0, 1). Then the position updating of particles can be defined as follows:

$$X_i^t = c_2 \oplus F_2(c_1 \oplus F_1(\omega \oplus F(X_i^{t-1}), P_i^{t-1}), G_i^{t-1}) \quad (7)$$

D. *Description of the Algorithm*

Our algorithm is comprised of five steps, which are described as follows:

Step 1: Initialization.

Step1-1: Load the circuit netlist data, and then initialize the number of the particles and the maximum iterations.

Step1-2: Each particle gets a random position and a random velocity according to the representation method we have proposed above. Set $pbest_i = X_i^0$ and $gbest$ is the best of all $pbest_i$.

Step 2: If the termination condition is satisfied, go to Step 5.

Step 3: Calculate the new position and velocity of all the particles.

Step 3-1: Calculate the new velocity and new position according to (4)-(7).

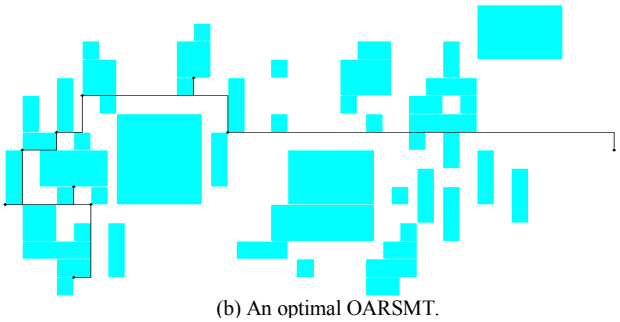
Step 3-2: Calculate the total wire length of each particle.

If the total wire length is less than that of the local best position of the particle, then update the local best position with the new position.

Step 4: If the total wire length of the local best position of some particles is less than that of the global best position, then update the global best with the local best. Go to Step 2.

Step 5: Stop the test run and output the best routing total wire length.

(a) The original test case ind3.



(b) An optimal OARSMT.

Figure 3. An optimal result generated by our algorithm.

V. EXPERIMENTAL RESULTS

We implemented our algorithm in MTALAB on a PC with 2.2 GHz Intel Pentium processor and 1 GB memory running in the Windows XP environment. There are 10 benchmark circuits were tested, five benchmarks (ind1-ind5) are industrial test cases from Synopsys, and the other five benchmarks (rc01-rc05) are provided by authors [12]. We set w decreased linearly from 0.9 to 0.4, c_1 decreased linearly from 0.85 to 0.4, c_2 increased linearly from 0.5 to 0.9, meanwhile, the swarm size as 100, the maximum iterations as 200 in our experiments. Then we compared our algorithm with those presented in [6], [7], [8], [9] and [12], the experimental results are directly quoted from these papers.

Fig. 3 is an optimal result for the test case ind3 generated by our algorithm, the cyan blocks are obstacles and the filled black circles denote vertexes. (a) is the original test case and (b) shows that our algorithm constructed an optimal OARSMT.

Table 1 lists the basic features of the benchmarks and the total wire length of these algorithms. We run 10 times for each benchmark. As can be observed from the table, compared with [6], our algorithm obtains further improves by 2.9% in total wire length on average, which is significant considering the increasing scale of VLSI design. We also can see that the total wire length of our algorithm is at most 8.98% shorter than that of [6] (ind1), but at most only 6.93% longer (ind2). Compared with [7] and [9], our algorithm also achieves further improves by 1.11% and 1.60% on average respectively. The total wire length generated by our algorithm is 6.6% and 1.4% longer on average than that of [9] and [12] respectively, but our algorithm is evolutionary algorithm, therefore, the solution quality of our algorithm could be more stable when considering the large and increasing scale of modern VLSI design.

TABLE I. THE TOTAL WIRE LENGTH OF THESE ALGORITHMS

Test cases	m	k	Total wire length						Improvement (X-F)/X (%)				
			[6] (A)	*[7] (B)*	*[8] (C)*	*[9] (D)*	*[12](E)*	*Ours(F)*	*X=A*	*X=B*	*X=C*	*X=D*	*X=E*
ind1	10	32	646	632	619	639	604	588	8.98	6.96	5.01	7.98	2.65
ind2	10	43	10100	9600	9500	10000	9500	10800	-6.93	-12.5	-13.68	-8.00	-12.5
ind3	10	50	623	613	600	623	600	600	3.69	2.12	0	3.69	0
ind4	25	79	1121	1121	1096	1126	1086	1053	6.07	6.07	3.92	6.48	3.04
ind5	33	71	1392	1364	1360	1397	1341	1319	5.24	3.30	3.01	5.58	1.64
rc01	10	10	27730	26900	25980	27540	25980	25805	6.94	4.07	0.67	6.30	0.67
rc02	30	10	42840	42210	42010	41930	41350	42641	0.47	-1.02	-1.50	-1.70	-3.12
rc03	50	10	56440	55750	54390	54180	54160	56529	-0.16	-1.40	-3.92	-4.34	-4.37
rc04	70	10	60840	60350	59740	59050	59070	59198	2.70	1.91	0.91	-0.25	-0.22
rc05	100	10	76970	76330	74650	75630	74070	75406	2.03	1.60	-1.01	0.30	-1.80
Average									2.90	1.11	-6.6	1.60	-1.4

m is the number of real vertexes, *k* is the number of obstacles and Average is the improves in total wire length on average

VI. CONCLUSIONS

In this paper, we propose an efficient swarm intelligence algorithm based on DPSO to construct an obstacle-avoiding rectilinear Steiner minimum tree in VLSI routing. Unlike those previous woks, ours is a novel framework of swarm intelligence algorithm to solve OARMST problem, which extends the application of the swarm intelligence algorithm in VLSI physics design, and the experimental results show that our framework can achieve effective solutions in most test cases. We will be committed to improving the performance of our algorithm in OARSMT problem continuously, which is regarded as our future work.

ACKNOWLEDGMENT

This work was supported in part by the National Basic Research Program of China under Grant No. 2011CB808000 and No.2006CB805904, the National Natural Science Foundation of China under Grant No.10871221, the Technology Innovation Platform Project of Fujian Province under Grant No.2009J1007, the project development foundations of Fuzhou University under Grant No. 2010-XQ-21.

REFERENCES

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

[2] F. K. Hwang, D. S. Richards, and P. Winter, The Steiner Tree Problem, Annals of Discrete Mathematics. Amsterdam: North Holland. The Netherlands. 1992.

[3] A. B. Kahng and G. Robins, A new class of iterative Steiner tree heuristics with good performance. IEEE Trans. Computer Aided Design, July 1992, 11: 893-902.

[4] Y. Hu, T. Jing, X. Hong, Z. Feng, X. Hu, and G. Yan, "An-OARSMan: Obstacle-avoiding Routing Tree Construction with Good Length Performance", Proceedings ASP-DAC, pp. 7-12, 2005.

[5] Y. Shi, T. Jing, L. He, Z. Feng, and X. Hong, "CDC Tree: Novel Obstacle-avoiding Routing Tree Construction based on Current Driven Circuit Model", Proceedings ASP-DAC, 2006.

[6] Z. Shen, C. Chu, and Y. Li, "Efficient Rectilinear Steiner Tree Construction with Rectilinear Blockages," in Proceedings ICCD, pp. 38-44, 2005.

[7] C. W. Lin, S. Y. Chen, C. F. Li, Y. W. Chang, and C. L. Yang, "Efficient Obstacle-avoiding Rectilinear Steiner Tree Construction," in Proc. Int. Symp. Phys. Des., pp.380-385, 2007.

[8] L. Li and Evangeline F. Y. Young, "Obstacle-avoiding Rectilinear Steiner Tree Construction," in Proc. Int. Conf. Compute.-Aided Des., pp.523-528, 2008.

[9] J. Y. Long, H. Zhou, and S. O. Mimic, "EBOARST: An Efficient Edge-Based Obstacle-Avoiding Rectilinear Steiner Tree Construction Algorithm," IEEE Transaction on Compute-Aided Design, Vol.27, No.12, pp.2169-2182, 2008.

[10] G. Ajwani, C. Chu, and W. K. Mak, "FOARS: FLUTE Based Obstacle-Avoiding Rectilinear Steiner Tree Construction," in Proc. Int. Symp. Phys. Des., pp. 27-34, 2010.

[11] L. Li, Z. Qian, and Evangeline F. Y. Young, "Generation of Optimal Obstacle-avoiding Rectilinear Steiner Minimum Tree," in Proc. Int. Conf. Compute.-Aided Des., pp.21-25, 2009.

[12] T. Huang and Evangeline F. Y. Young, "Obstacle-avoiding Rectilinear Steiner Minimum Tree Construction: An Optimal Approach," in Proc. Int. Conf. Comput.-Aided Des., pp. 610-613, 2010.

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

[14] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in: Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics 1997. IEEE Service Center, Piscataway, NJ., 1997: 4101~4109.

[15] 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.

[16] 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, . pp. 802-805, 2009.

[17] G. L. Chen, W. Z. Guo, and Y. Z. Chen, A PSO-based Intelligent Decision Algorithm for VLSI Floor planning. Soft Computing. 2010, 14(12):1329-1337.

[18] W. Z. Guo, G. L. Chen, M. Huang, and S. L Chen, "A discrete particle swarm optimization algorithm for the multiobjective permutation flow-shop sequencing problem," Proceedings of the second international conference of Fuzzy Information and Engineering, pp.323-331, 2007.

[19] Prüfer. H., "Neuer beweis eines satzes uber permutationen", Arch. Math., vol.27, pp. 742-744, 1918.