Journal of Theoretical and Applied Information Technology
15 May 2012. Vol. 39 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A MODIFIED HYBRID PARTICLE SWARM OPTIMIZATION ALGORITHM FOR SOLVING THE TRAVELING SALESMEN PROBLEM

[1]SAID LABED , [2]AMIRA GHERBOUDJ , [3]SALIM CHIKHI

[1,2,3] Computer Science Department, MISC Laboratory,
Mentouri University of Constantine – Algeria

E-mail : [1]s.labed@umc.edu.dz, [2]a.gherboudj@umc.edu.dz , [3]chikhi@umc.edu.dz

ABSTRACT

The traveling salesman problem (TSP) is a well-known NP-hard combinatorial optimization problem. The problem is easy to state, but hard to solve. Many real-world problems can be formulated as instances of the TSP, for example, computer wiring, vehicle routing, crystallography, robot control, drilling of printed circuit boards and chronological sequencing. In this paper, we present a modified hybrid Particle Swarm Optimization (MHPSO) algorithm in which we combine some principles of Particle Swarm Optimization (PSO), the Crossover operation of the Genetic Algorithm and 2-opt improvement heuristic. The main feature of our approach is that it allows avoiding a major problem of metaheuristics: the parameters setting. In the aim to prove the performance and convergence of the proposed algorithm, we have used it to solve some TSP instances taken from TSPLIB library. Moreover, we have compared our results with those obtained by other algorithms based PSO.

Keywords: *Traveling Salesman Problem, Particle Swarm Optimization, Optimization, Meta-heuristics.*

1. INTRODUCTION

The traveling salesmen problem (TSP) is a classical NP-Hard combinatorial optimization problem [1]. It can be formulated as follows: giving a set of cities, and distances between them, the goal is to find the shortest tour visiting every city only once and returning to the starting city. The problem where distance between two cities does not depend on the direction is called symmetric TSP; and asymmetric otherwise. The distance can be replaced by another notion, such as time or money. In all cases, we talk about cost.

The TSP finds application in a variety of situations such as vehicle routing, logistics, automatic drilling of printed circuit boards, x-ray crystallography, robot control, and chronological sequencing. TSP problem has been used during the last years as a basis for comparison in order to improve several optimization techniques.

TSP is considered to be NP-Hard problem, whose computational complexity rises exponentially by increasing the number of cities, making it impossible to solve very large instances in reasonable times. For a symmetric problem with n cities there are (n-1)!/2 possible tours. The time complexity of TSP problem is $O(n!)$. Assuming that the time required to evaluate a path is 1µs, Table 1 shows the combinatorial explosion of the TSP.

Table 1. Number of possible paths, and computation time estimated

Number of cities	Number of possibilities	Computational time
5	12	12 µ
10	181440	0,18 s
12	19958400	5,54 hours
15	43 billions	12 hours
20	60 E+15	1928 years

The proposed methods in the literature for solving the TSP can be classed into two classes: exact methods and approximated methods (or heuristics). The exact methods are guaranteed to find the optimal solution in a bounded number of steps, but with a limited number of cities. These methods provide the optimal solution but they are

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

very greedy in term of computation time required to reach the optimal solution. The time complexity of the proposed exact methods in the literature grows exponentially with n (the number of cities). For example, the exact solution of a symmetric problem with 2392 cities was determined over a period of more than 27 hours on powerful super computer [2]. The most effective algorithms in this class are "cutting-plane" and "branch and bound" [3]. However, approximated methods (heuristics and metaheuristics) have the advantage of finding a near optimal solution in a reasonable time. In fact, several heuristics are proposed in the literature. These heuristics can be classed in three classes: construction heuristics, improvement heuristics and composite algorithms. The construction heuristics built a cities tour by adding a new city at each step. They stop when the solution is found and do not try to improve it. In this category we can cite the nearest neighbor heuristic and the greedy algorithm. The improvement heuristics try to enhance the tour that has been generated by a construction heuristic, in the aim to obtain better tour quality. The local search algorithms 2-opt and 3-opt [2] are examples of the most used. In the same class, we find also Lin-Kernighan [2] algorithm, Tabu Search [4], and Simulated Annealing [5]. In the same class, we can cite other metaheuristics such as: genetic algorithm [6]-[7], ant colony algorithms [8] and particle swarm optimization [15]. Several approaches based particle swarm optimization algorithm are proposed in the literature as in [9]-[10]-[11]-[12]-[13]-[14].

In this contribution, we proposed a new hybrid Particle Swarm Optimization (MHPSO) algorithm in which we combine some principles of Particle Swarm Optimization (PSO), the Crossover operation of the Genetic Algorithm and 2-opt improvement heuristic. The main feature of the proposed algorithm is that it allows avoiding a major problem of metaheuristics: the parameters setting. In fact it requires few parameters to be set.

The remainder of this paper is organized as follows. The PSO principle is described in section 2. In the third section we describe the proposed algorithm. Experimental results are provided in section 4 and a conclusion is provided in the fifth section of this paper.

2. PSO PRINCIPLE

Particle Swarm Optimization (PSO) is a recent metaheuristic. It was developed by Kennedy and Eberhart in 1995 for solving optimization problems [15]. It mimics the collective behavior of animals living in groups such as bird flocking and fish schooling. The PSO method involves a set of agents for solving a given problem. This set is called swarm, each swarm is composed of a set of members called particles. Each particle is characterized by position $x_{id}= (x_{i1}, x_{i2},\ldots, x_{id},\ldots, x_{iD})$ and velocity $v_{id}= (v_{i1}, v_{i2},\ldots, v_{id},\ldots, v_{iD})$ in a search space of D-dimension. During the search procedure, the particle tends to move towards the best position (solution) found. At each iteration of the search procedure, the particle moves and updates its velocity and its position in the swarm based on experience and the results found by the particle itself, its neighbors and the swarm. It therefore combines three components (Fig 1): its own current velocity, its best position $p_{bestid}= (p_{besti1}, p_{besti2},\ldots, p_{bestid},\ldots, p_{bestiD})$ and the best position obtained by its informants or by the swarm $g_{best} = (g_{best1}, g_{best2},\ldots, g_{bestd},\ldots, g_{bestD})$.

The research process is based on two rules:
- Each particle has a memory which can store the best position in which it has already passed and it tends to return to that position.
- Each particle is informed by the best known position within its neighborhood and it will tend to move towards this position.

Each particle will move depending on its velocity and the two best positions she know (his own and that of the swarm) according to the following two equations [16]:

$$v_{id}(t)= \omega\, v_{id}(t-1) + c_1\, r_1\, (p_{bestid}(t-1) - x_{id}(t-1)) + c_2\, r_2\, (g_{bestd}(t-1) - x_{id}(t-1)) \quad (1)$$

$$x_{id}(t)= x_{id}(t-1) + v_{id}(t) \quad (2)$$

ω is an inertia coefficient. (x_{id} (t), x_{id} (t-1)), (v_{id} (t), v_{id} (t-1)): position and velocity of particle i in dimension d at times t and t-1, respectively. p_{bestid} (t-1), g_{bestd}(t-1) : the best position obtained by the particle i and the best position obtained by the swarm in dimension d at time t-1, respectively. c_1, c_2: two constants representing the acceleration coefficients. r_1, r_2: random numbers drawn from the interval [0,1[. v_{id} (t-1), $c_1\, r_1\, (p_{bestid}$ (t-1) - x_{id} (t-1)), $c_2\, r_2\, (g_{bestd}$(t-1) - x_{id} (t-1)): the three components mentioned above, respectively.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

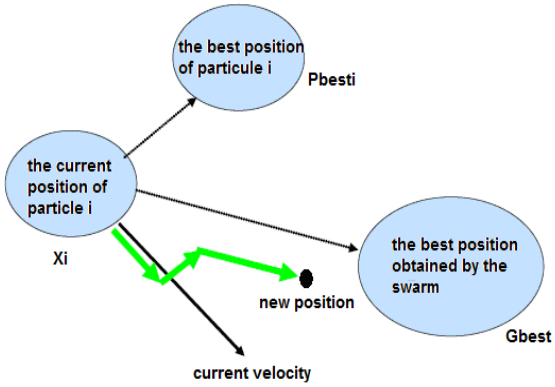

Figure 1. A particle displacement

The position of particle i indicates the possible solution in the multidimensional space of the addressed problem, and the velocity indicates the amount of change between the current position and the previous one. We could calculate the particle's fitness by putting its position into a designated objective function f (x_{id}).. When the fitness is higher, the corresponding position is better. A pseudo code of PSO algorithm is shown in Algorithm 1[17].

Algorithm 1 : pseudo code for basic PSO

1. Initialization :
   - Parameters and size of the swarm (S);
   - Randomly initialize particles positions and velocities;
   - For each particle, let $p_{bestid} = x_{id}$;
   - Calculate f (x_{id}) of each particle;
   - Calculate g_{bestd}; // the best of p_{bestid}
2. While (termination criterion is not met) {
   For (i = 1 to S) {
   - Calculate the new velocity using equation (1);
   - Calculate the new position using equation (2);
   - Calculate f (x_{id}) of each particle;
   - If (f (x_{id}) < f (p_{bestid}))
     $p_{bestid} = x_{id}$; // Minimization case
   - If (f (p_{bestid}) < f (g_{bestd}))
     $g_{bestd} = p_{bestid}$;

   }
   }
3. Show the best solution found $g_{bestd;}$

The PSO was originally developed for continuous valued spaces but many problems are, however, defined for discrete valued spaces where the domain of the variables is finite. Kennedy and Eberhart introduced a discrete binary version of PSO in 1997 for discrete optimization problems [18]. In binary PSO, each particle represents its position in binary values which are 0 or 1. The velocity v_{id} of the particle i is calculated from equation (1). v_{id} is a set of real numbers that must be transformed into a set of probabilities, using the sigmoid function as follows:

$$sig(v_{id}) = \frac{1}{1+\exp(-v_{id})} \qquad (3)$$

Where sig(v_{id}) represents the probability of bit x_{id} takes the value 1. The position x_{id} of particle i is updated as follows:

$$x_{id} = \begin{cases} 1 \text{ If } r < \text{sig}(v_{id}) \\ 0 \text{ Otherwise} \end{cases} \qquad (4)$$

Where, r is a random number taken from the interval [0, 1[.
To avoid the problem of the divergence of the swarm, the velocity v_{id} is generally limited by a maximum value Vmax and a minimum value -Vmax, i.e. $v_{id} \in$ [-Vmax, Vmax]

3. THE PROPOSED ALGORITHM (MHPSO)

PSO is a recent metaheuristic that has proved its simplicity of implementation, its effectiveness and its very fast convergence [17]. However, the selection and adaptation of the large number of PSO parameters such as: swarm size, inertia coefficient ω, acceleration coefficients c_1 and c_2, play a crucial role for good and efficient operation of PSO. On the other hand, PSO may be easily trapped into local optima if the global best and local best positions are equal to the position of particle over a number of iterations.

In the aim to benefit from all these advantages and escape all these shortcomings, we are inspired by the PSO principle, the crossover operation of the Genetic algorithm which allows a good exploration of the search space and the 2-opt heuristic that has proved its efficiency in dealing with TSP. Our objective is to propose a New Hybrid Particle Swarm Optimization algorithm that is characterized by its simplicity, few parameters to be set and which provides a good balance between

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

exploitation and exploration of the search space. The proposed algorithm is described below.

3.1. The problem formulation

The Traveling Salesmen Problem can be formulated as follows: assuming n the number of cities to visit, C=(c_{ij}) the matrix of distances (costs), where c_{ij} is the distance between city i and city j (i, j = 1, ... n). The problem that we want to resolve is to find a permutation (i_1,i_2,i_3,…,i_n) of integers from 1 to n which minimize the expression: $c_{i1i2}+c_{i2i3}+c_{i3i4}+…..+c_{ini1}$. If $c_{ij} = c_{ji}$ for all i and j, the problem is symmetric, otherwise it is asymmetric.

3.2. The 2-opt method

2-opt is a local search method. It is a well-known method for solving TSP. It provides feasible results within a reasonable time. 2-opt is used primarily to improve a solution in hybrid approaches. The 2-opt algorithm starts with a given tour and then searches in the neighborhood of the current solution all round improving the current configuration.

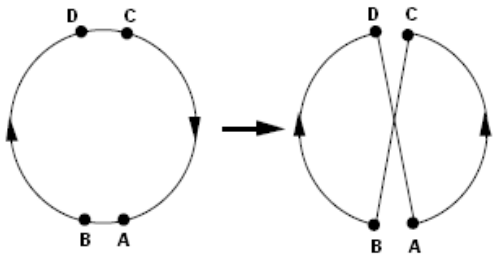

Figure 2. 2-opt move

During the improvement process, the algorithm examines whether the exchange of two edges produces a shorter tour. The algorithm continues until no improvement is possible (Figure 2). The steps of the algorithm are presented below:

1. Take two pairs of consecutive nodes, pairs (A,B) and (C,D) from a tour.
2. Check if the distance (AB + CD) is higher than (AD + BC)
3. If that is the case, swap A and C, resulting in reversing the tour between the two nodes.
4. If improvement of the tour, then go to 1, otherwise stop.

The time complexity for 2-opt algorithm is *O* (n^2).

3.1. Crossover Operation

The Crossover operation is one of the genetic algorithm operations which has introduced by John Holland in 1960 [21. The main role of the crossover operation is to produce a new population (a set of individuals). It consists in combining the characteristics of two individuals (parents) to produce one or two new individuals (childs). In the proposed algorithm, we have introduced the crossover operator in the aim to produce a new population. We have used the crossover operation between the best position p_{bestid} of particle i and its current position x_{id} to produce a new particle. Then, we apply the crossover operator between the best position obtained by the swarm g_{bestd} and the current position x_{id} of the particle i to produce another particle. Afterward, we choose the best solution (child) to represent the new particle. A variety of crossover operators has been proposed in the literature to solve the traveling salesman problem. We cite as example: the ERX (Edge Recombination Crossover, the PMX (Partially Mapped Crossover), the OBX (Order-Based Crossover), the DPX (Distance Preserving Crossover), the MPX (Maximal Preservative Crossover) and the ECH (Exchange Crossover Heuristic) [6]. For our part we inspired by the ECH algorithm and we proposed a modified version, which is more suited to the traveling salesmen problem. It allows the generation of feasible solutions that meet strict constraints of the traveling salesman problem.

Assuming that we have two particles (solutions) x_1, x_2 and we want to cross them together to get a new particle. We first choose randomly a city v from the n cities to visit. Next, we generate two new particles by moving the city v in the beginning of x_1 and x_2. After this, we initialize the new particle x (the child or the new solution) by the city v. And we continue the construction of x, by concatenating the cities one after another.

At each iteration of the construction procedure of x, we are faced with two cities to choose. The city v_1 of the first parent (x_1) and the city v_2 of the second parent (x_2). To opt for one of the cities v_1 and v_2, we choose the city closest to the last city of x. During the construction process of x, we must ensure that the two candidate cities (v_1 and v_2) are not included in the new solution x because we must respect the principal the constraint of the traveling salesman problem (every city must be visited only once). The proposed crossover algorithm is noted by MECH (Modified ECH). A pseudo code of the MECH algorithm is presented in Algorithm 2 and its principle is explained by an example in Figure 3.

Journal of Theoretical and Applied Information Technology
15 May 2012. Vol. 39 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

v=1
x_1= 6 2 1 5 4 3
x_1= 1 6 2 5 4 3
x_2= 1 3 4 5 2 6
x_2= 1 3 4 5 2 6
x=1
Choose 1-6 or 1-3 :
x=1 6
Choose 6-2 or 6-3 :
x=1 6 3
Choose 3-2 or 3-4 : x=1 6 3 4
Choose 4-2 or 4-5 : x=1 6 3 4 2
Choose 2-5 or 2-5 : x=1 6 3 4 2 5
Choose 5-4 or 5-2 : x=1 6 3 4 2 5
Choose 5-3 or 5-6 : x=1 6 3 4 2 5

x=1 6 3 4 2 5

0		13			12
	0				
	13	0	9		
	8		0	9	
				0	
	11	10			0

Distance Matrix

Figure 3. An example of the MECH crossover

Algorithm 2: MECH

Input: Two solutions x_1 and x_2
Output: One solution x

Choose a random city v;
moving the city v in the beginning of x_1 and x_2;
Initialize x by v : x=v;
i=2; j=2;
While [(i&j) <= n]{
If [(x_1(i) & x_2(j))∈ x]{
i++; j++;
Otherwise-if [x_1(i) ∈ x]
concatenate x_2 (j) to x;
j++;
Otherwise-if [x_2 (j) ∈ x]
concatenate x_1 (i) to x;
i++ ;
Otherwise-if [(x_1 (i) & x_2 (j))∉ x]
u= the last city in x;
If distance [u,x_1 (i)] < distance[u,x_2 (j)]{
concatenate x_1 (i) to x; i++;
Otherwise
concatenate x_2 (j) to x; j++;
}
}
}

3.4. The MHPSO Algorithm

As any algorithm, the first step in the MHPSO algorithm is to initialize some necessary parameters for good and efficient operation of the algorithm. The main characteristic of the MHPSO algorithm is its simplicity. In fact, comparing with other population metaheuristics such as PSO and GA, in the proposed algorithm there are few parameters to be set. Steps of the proposed algorithm (MHPSO) are presented below.

Step1:
- Initialize a swarm size S
- Random position of each particle.
- For each particle, let $p_{bestid} = x_{id}$

Step 2: Calculate the fitness of each particle
Step 3: Calculate g_{bestd}
Step 4: Four each particle, calculate her new position using the following equation (5):

$$x_{id} = \text{Max}[(p_{bestid} \otimes x_{id}),(g_{bestd} \otimes x_{id})] \quad (5)$$

Where «⊗» is the MECH crossover.

Step 5: Improving 10% of particles positions (solutions) using 2-opt method.

Step 6: Update p_{bestid} and g_{bestd} as follows:

If ($f(x_{id}) < f(p_{bestid})$)
$p_{bestid} = x_{id}$; (6)
If ($f(p_{bestid}) < f(g_{bestd})$)
$g_{bestd} = p_{bestid}$; for i=1,…,n (7)

Step 7: Stop iterations if stop condition is verified. Return to Step 4, otherwise.

The solution of the problem is the last g_{bestd}. Figure 4 shows the flowchart of the proposed algorithm.

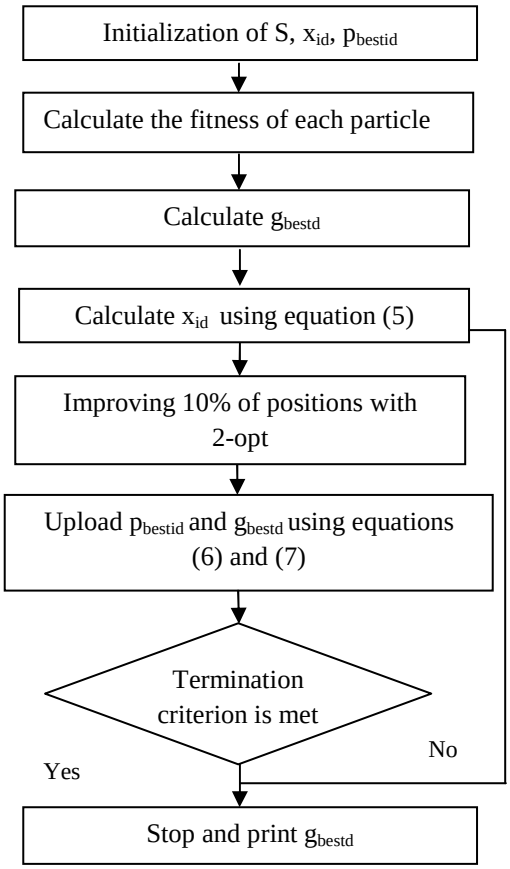

Figure 4. Flowchart of the proposed algorithm (MHPSO)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4. EXPERIMENTAL RESULTS

To validate the feasibility and effectiveness of the proposed approach, we have applied it on some TSP instances taken from TSPLIB [22]. The proposed algorithm was implemented in Matlab 7 with the following parameters: 50 particles, 200 iterations in maximum. Two parts of experiments were performed. First, we have tested the MHPSO algorithm on some TSP instances from TSPLIB. In the second part of experiments, we have compared the obtained results in terms of best solution with the exact solution (The best known), the obtained solution by the DPSO [9] and the obtained solution by the CDPSO [14].

Table 2 shows the experimental results of our MHPSO algorithm with some instances taken from TSPLIB. The first column, indicates the instance name, the second column indicates the problem size i.e. number of cities. The third column indicates the best known solution from TSPLIB. The fourth column indicates the best results obtained by the MHPSO. Table 2 shows that the proposed algorithm is able to find the best known result of the major instances.

Instance	Problem size	Best Known	MHPSO result
bays29	29	2020	**2020**
berlin52	52	7542	**7542**
dantzig42	42	699	**699**
eil51	51	426	**426**
eil76	76	538	**538**
eil101	101	629	**629**
fri26	26	937	**937**
kroA100	100	21282	**21282**
kroB100	100	22141	**22141**
kroC100	100	20749	**20749**
kroD100	100	21294	21309
kroE100	100	22068	**22068**
kroB150	150	26130	**26130**
pr107	107	44303	44391
Pr124	124	59030	**59030**
Pr76	76	108159	**108159**
Rat99	99	1211	1212
Pr144	144	58537	**58537**
Suiss42	42	1273	**1273**
St70	70	675	**675**

Table 2. MHPSO results

Table 3 shows a comparison in terms of best solution between the exact solution (best known), our MHPSO algorithm, the DPSO [9] algorithm and the CDPSO [14] algorithm. The first column indicates the instance name. The second column indicates exact solution as declared in TSPLIB. The third fourth and fifth columns record the best results obtained by the MHPSO, the DPSO and the CDPSO. N/A in Table 3 means "Not Available" result in paper reference.

Table 3 shows that the proposed algorithm outperforms the DPSO and the CDPSO algorithms. In fact, the MHPSO is able to find the best result of all instances.

The obtained results are very encouraging; they prove the effectiveness of the proposed algorithm.

Instance	Best Known	MHPSO Best solution	DPSO Best solution	CDPSO Best solution
eil51	426	**426**	427	**426**
berlin52	7542	**7542**	**7542**	**7542**
st70	675	**675**	**675**	**675**
eil76	538	**538**	546	**538**
Pr76	108159	**108159**	108280	**108159**
kroA100	21282	**21282**	N/D	**21282**
kroE100	22068	**22068**	N/D	**22068**
Pr124	59030	**59030**	N/D	**59030**
Pr144	58537	**58537**	N/D	**58537**
kroB150	26130	**26130**	N/D	26141

Table 3. Comparative results between MHPSO, DPSO and CDPSO.

5. CONCLUSION

In this paper, we proposed a new hybrid algorithm based on particle swarm optimization to solve the traveling salesman problem. In this algorithm, we were inspired by the PSO principle, the crossover operation of the genetic algorithm and also the improvement heuristic 2OPT. The proposed algorithm is characterized by its simplicity and a good balance between exploitation and exploration of the search space. It requires few parameters to be set. In the aim to testing and proving the efficiency of the proposed algorithm in dealing with the combinatorial optimization problems, we have used it to solve the well known NP-Hard combinatorial optimization problem of traveling salesmen problem .The obtained results are very encouraging, they prove the effectiveness of the proposed algorithm. Based on these promising results, our fundamental outlook moving towards the application of the proposed algorithm on large instances and using it to resolve other NP-Hard combinatorial optimization problems.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFERENCES

[1] M. R. Garey, and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H. Freeman, New York, 1979.

[2] S.Lin and B.Kernighan , "An effective heuristic algorithm for the traveling-salesman Problem", *Operations Research*, vol. 21, no. 2, April 1973, pp. 498-516.

[3] L. Gilbert, « La petite et la grande histoire du problème du voyageur de commerce", *Congrés de la ROADEF*, Lille février 2006.

[4] F.Glover, " Tabu Search-Part II", *ORSA Journal on Computing*, no. 2,1990,pp. 4-32.

[5] S. Kirkpatrik,"Optimization by simulated annealing: quantitative studies". *Journal of Statistical Physics*, vol. 34, 1984, pp. 975-986.

[6] L. Li & Y. Zhang, "An Improved Genetic Algorithm for the Traveling Salesman Problem", *Communications in Computer and Information Science*, vol. 2, Part 5, 2007, pp. 208-216.

[7] R.Thamilselvan & P.Balasubramanie , "Genetic Algorithm with a Tabu Search (GTA) for Traveling Salesman Problem ", *International Journal of Recent Trends in Engineering*, vol. 1, no. 1, May 2009.

[8] M. Dorigo & L. M. Gambardella, "Ant Colony System : A cooperative learning approach to the traveling salesman problem" , *IEEE Transactions on Evolutionary Computation*, no. 1,1997, pp. 53-66.

[9] X.H. Shi,Y.C. Liang, H.P. Lee, C. Lu & Q.X. Wang, "Particle swarm optimization-based algorithms for TSP and generalized TSP", *Information Processing Letters*, pp. 169–176, ELSEVIER 2007.

[10] W. Cheng, Z. Maimai ,& L. Jian, "Solving Salesman Traveling Problems with Time Windows by Genetic Particule Swarm Optimization" *IEEE Congress on Evolutionary Computation*, CEC 2008.

[11] X. Bin, & L. Zhaohui, "An Improved Hybrid Discrete Particle Swarm Optimization Algorithm to Solve the TSP Problem", *Applied Mechanics and Materials*, vols. 130-134, 2012 ,pp. 3589-3594.

[12] W. Chen, J. Zhang, S. H. Chung ,W. Zhong, W. Wu & Yu-hui Shi, "A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems", *IEEE Transactions on evolutionary computation*, vol. 14 no. 2, April 2010.

[13] D. Chaojun & Q. Zulian, "Particle Swarm Optimization Algorithm Based on the Idea of Simulated Annealing", *IJCSNS International Journal of Computer Science and Network Security*, vol. 6 no. 10, October 2006

[14] Y. Xu, Q. Wang & J. Hu, "An improved Discrete Particle Swarm Optimization Based on Cooperative Swarms", *International Conference on Web Intelligence and Intelligent Agent Technology*, 2008 IEEE/WIC/ACM

[15] J. Kennedy & R.C. Eberhart. "Particle Swarm Optimization." In: Proc. *IEEE Int. Conf. On Neural Networks*, WA, Australia, 1995, pp. 1942–1948.

[16] Y. Shi. & R. Eberhart. "Parameter Selection in Particle Swarm Optimization", *Proceedings of the 7th Annual Conference on Evolutionary Programming*, pp. 591-600, LNCS 1447, Springer (1998)

[17] A. Gherboudj & S. Chikhi, "BPSO Algorithms for Knapsack Problem". A. Özcan, J. Zizka, and D. Nagamalai (Eds.): *WiMo/CoNeCo 2011*, CCIS 162, pp. 217–227, 2011. Springer (2011)

[18] J. Kennedy, & R. Eberhart. "A discrete binary version of the particle swarm algorithm." *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*, 1997, pp. 4104-4109.

[19] M.A. Khanesar , M. Teshnehlab & M.A Shoorehdeli, "A Novel Binary Particle Swarm Optimization". *In proceedings of the 15th Mediterranean Conference on Control & Automation*, July 27– 29, 2007, Athens – Greece.

[20] J. Olamaei, T. Niknam, & G. Gharehpetian, "Application of particle swarm optimization for distribution feeder reconfiguration considering distributed generators". *Applied Mathematics and computation*, vol. 201, no. 1-2,2008, pp.575-586.

[21] J H. Holland., "Adaptation in natural and artificial system". Ann Arbor, The University of Michigan Press, (1975).

[22]TSPLIB, http://comopt.ifi.uni-idelberg.de/software/TSPLIB95/

[23] Y. Cooren, "Perfectionnement d'un algorithme adaptatif d'Optimisation par Essaim Particulaire. Applications en génie médical et en électronique ",Thèse de Doctorat , Université Paris 12 Val de Marne, France.