Proceedings of the IEEE
International Conference on Automation and Logistics
August 18 - 21, 2007, Jinan, China

Chaotic Particle Swarm Optimization Algorithm for Traveling Salesman Problem

Zhenglei Yuan, Liliang Yang, Yaohua Wu, Li Liao, Guoqiang Li

The Logistics Institute
Shandong University
Jinan, Shandong Province, China

Zhenglei.yuan@mail.sdu.edu.cn

Abstract- **In this paper, a novel algorithm based on particle optimization algorithm (PSO) and chaos optimization algorithm (COA) is presented to solve traveling salesman problem. Some new operators are proposed to overcome the difficulties of implementing PSO into solving the discreet problems. Meanwhile embedded with chaos optimization algorithm (COA) it can enhance particle's global searching ability so as not to converge to the local optimal solutions too quickly. The experiment results of several benchmark test problems show its validity and satisfactory effect.**

Index Terms--Particle swarm optimization (PSO); Chaos optimization algorithm (COA); Traveling Salesman Problem (TSP); Position swap.

Ⅰ .INTRODUCTION

The particle swarm optimization (PSO) algorithm, creatively developed by Kennedy and Eberhart [1], is a population-based optimization method simulating social behavior of flocks of birds in searching for foods. The system is initialized in a set of randomly generated particles and then iteratively searches for the optimal solutions. During each loop, every particle follows the heretofore best local and global positions to update its own velocity and position. Compared with other heuristic algorithms, the PSO has a better intelligent strategy inherent in it. Therefore the PSO has been broadly used in both scientific research and engineering application [2] [3]. PSO shows great adaptability in continuous optimization problems but also inborn weakness for solving discrete problems where the velocity and position lose their conventional meaning and need to be redefined. Yet, more and more attention has been focused on discrete ones recently [4] [5].

As a well-known NP-hard combinatorial optimization problem, TSP is quite easy to describe but hard to solve. Give a weighed graph G = (S, D), where S = (S1, S2… S_m) is the set of cities and D= {(S_i, S_j): Si, S_j•S} is the set of edges. Let d (S_i, S_j) be a cost measure associated with the edge (S_i, S_j) •S. The object of TSP is to find a roundtrip of minimal total length visiting each city exactly once. TSP is a intensively studied benchmark in combinatorial optimization [7], including some of techniques in evolutionary computation, such as the nearest neighbourhood search (NNS), simulated annealing (SA), tabu search (TS), neural network (NN), ant colony system (ACS), genetic algorithm (GA) as well as several hybrid algorithms. As for PSO, Clerc[8], Hendtlass[9] and Wang[10] came up with revised PSO ways for solving TSP. Among these three methods, the basic equations were adjusted to fit the characteristics and requirements of the problem. Quite clear, it is only after remodelling that PSO can be suitably used in solving TSP.

Chaos [6] is a common nonlinear phenomenon with much complexity and similarity with randomness. Chaos is typical of being highly sensitive to the initial value so that it represents greatly diversity due to the ergodic property of the phase space-chaos can go through all states in certain ranges without repetition; the congenital randomness of the system means that chaos behaviour is similar to randomness which is out-of-order; at the same time it is generated through the deterministic iteration formula. Because of the characteristics chaos has, it can be applied in optimization.

In this paper, a new algorithm based on the canonical PSO is proposed. First of all, some new operators are introduced to overcome the gap between continuous and discrete problems when using PSO. With the help of these operators such as "Position swap" and "Swap loop", the difference between two travelling trips can be well identified and expressed so that the subtraction and addition operations in the original particle velocity and position updating equations have gained brand-new definitions. Moreover, combined with Chaos optimization algorithm, the particles can go through all places in the solution space without repetition under a deterministic iteration formula. Therefore the so-called chaotic particle swarm optimization can guarantee not only the global searching ability but also finding a preferable result in a reasonable time.

Ⅱ .STANDARD PARTICLE SWARM OPTIMIZATION (PSO) ALGORITHM

Suppose that the searching space is n dimensional with m randomly initialized particles in it. Every particle is represented by an n-dimensional vector X_i (i=1, 2… m) which stands for its location (x_{i1}, x_{i2}… x_{in}) in space and it is also regarded as a potential solution. The fitness value of each particle, which means the distance from the optimal solution, can be calculated by the fitness function. The higher the fitness value is, the better the particle is or the nearer it is to the best. During the iteration, the best position that a particle has been visited is defined as pbesti, while the best solution of the whole colony is defined as gbesti. The two best solutions attract particles to travel to them with different velocity which is a n-dimensional vector as well denoted as V_i= (v_1, v_2…

v_n).The PSO algorithm can be demonstrated by the following velocity and position updating equations:

$$X_i(k+1) = X_i(k) + V_i(k+1) \quad (1)$$

$$V_i(k+1) = c_0 V_i(k) + c_1 (pbest_i - X_i(k)) + c_2 (gbest - X_i(k)) \quad (2)$$

where i=1,2,…,m;c_o,c_1,c_2 are colony learning coefficients; generally ,c_0 is randomly generated between [0,1], while c_1, c_2 are randomly located in the interval[0,2]. The inertia coefficient c_o enables particles to keep their moving inertia so as to expand their searching scope and reach the optimal solution with more possibility. If c_o =0, the velocities of particles depends on $pbest_i$, $gbest_i$ and have nothing to do with their primitive ones. Assume that one particle takes the place of the temporary $gbest_i$, it'll stay still. However those which are not in $gbest_i$ would converge to the powered middle points of both $gbest_i$ and $pbest_i$. Under this condition, the PSO degrades into a kind of local searching algorithm. As for the acceleration coefficients c_1, c_2, they pull the particles close to the powered middle points of both $gbest_i$ and $pbest_i$. If c_1 =0, the particle would lost its own recognition ability which means particles would pool around the $gbest_i$ more quickly than the standard PSO. But for the large scale problems, the algorithm with c_1 =0 would be much more likely to run into local optimal solution. If c_2 =0, particles would be deprived of their social cooperation ability, which means that particles could not share the optimal solution of the colony but keep going on their own way. Particles can hardly converge to the global optimal solution so the algorithm would lose its efficiency. The iteration stops when the determination criterion meets such as whether the maximum generation or a designated value of the fitness is reached.

The algorithm above is quite suitable in continuous problems. However, it can be hardly applied into discreet problems without changing. In order to enlarge its utility, Kennedy and Eberhart brought up a discrete binary version of PSO algorithm by defining the particles' trajectories and velocities in terms of changes of probabilities that a bit will be in one state to the other [4].Thus a particle moves in a state spaces restricted to 0 to 1 in each generation, where v_{id} represents the probability of bit x_{id} taking 1.

Ⅲ.CHAOTIC PARTICLE SWARM OPTIMIZATION ALGORITHM

The first obstacle that needs to be surmounted is to redefine the subjection such as ($pbest_i - X_i(k)$) and addition in (2). Inspired by the concept of "Swap operator" and "Swap sequence" in Wang's paper [10], here some new definitions such as "Position swap" and "Swap loop" are introduced.

For a m-city TSP problem, natural numbers are chosen for coding. Denote $X_i = \{x_{i1}, x_{i2} \ldots x_{im}\}$ (x_i=1, 2…m) as the position of the *i*th particle, which means the cities traveling sequence of $x_{i1} \rightarrow x_{i2} \rightarrow \ldots \rightarrow x_{im}$.

*Definition1.*In two traveling sequences $X_i=\{x_{i1},x_{i2},\ldots,x_{im}\}$ and $X_j=\{x_{j1},x_{j2},\ldots,x_{jm}\}$, if the city numbers in the same position are different such as $x_{ia} \neq xj_a$ (a=1,2,…,m), two city numbers x_{ia} and x_{ia} form a position swap (x_{ia}, x_{ia}), namely D(x_{ia}, x_{ia}).

E.g.1 Suppose that for a 6-city TSP there are two distinct solutions X_1= (2, 3, 1, 5, 6, 4) and X_2= (3, 1, 4, 6, 5, 2). In the first position, x_{11}=2 is different from x_{21}=3 therefore x_{11} and x_{21} can constitute a position swap (2, 3).

Definition2. Several position swaps form a position swap sequence, namely DS.

E.g.2 For the above two solutions X_1, X_2, a position swap sequence is:

DS={D_1(2, 3), D_2(3,1), D_3(1,4), D_4(5,6), D_5(6, 5), D_6(4, 2)}.

Definition3. Take every position swap as a two-dimensional array. If the second digit in the array equals to the first digit in another one, place the two arrays adjacently. After going through all the position swaps that two different solutions may have, the results would be a closed loop. The first digit of the first array is equal to the second digit of the last array. The distinct city numbers in the arrays which constitute the closed loop form a swap loop, namely L.

E.g.3 Among all the position swaps of X_1 and X_2,D_1(2,3),D_2(3,1),D_3(1,4),D_6(4,2)can constitute L_1(2, 3, 1, 4) and D_4(5, 6) D_5(6, 5) form L_2(5, 6) as well.

Definition4. One or more swap loops constitute a swap loop sequence, namely LS.

Definition5. Subtraction means calculating the swap loop sequences between two solutions.

E.g.4 X_1 (2, 3, 1, 5, 6, 4) - X_2 (3, 1, 4, 6, 5, 2) =LS (LS_1, LS_2).

Definition6. A solution added up with a position swap means interchanging the positions of two city numbers, which are included in the position swap, in the original solution.

E.g.5 X_2 (3, 1, 4, 6, 5, 2) + D (5, 6) = X_2' (3, 1, 4, 5, 6, 2)

Definition7. A solution added up with a swap loop sequence means that orderly add the solution with the position swaps which constitute the swap loop sequence.

E.g. 6 X_2 (3, 1, 4, 6, 5, 2) + LS (LS1,LS2) = X_2+ L_1+L_2 = X_2 + D_1 + D_2 + D_3+ D_6 + D_4 = { X_2 (3, 1, 4, 6, 5, 2) + D_1(2, 3)}+ D_2 + D_3 + D_6 + D_4 = X_{21}(2, 1, 4, 6, 5, 3) +D_2+D_3+D_6+D_4=X_{22}(2, 3, 4, 6, 5, 1) +D_3+D_4=X_{23}(2, 3, 1, 6, 5, 4)+D_4=B_{24}(2, 3, 1, 5, 6, 4)=X_1（2 3 1 5 6 4）.

From E.g. 6, we can see the inherent relationship between different travelling sequences. Any two sequences are interchangeable. If one sequence plus one position swap, it can achieve a new or temporary state. If added with the swap loop between A and B, the sequence A can be changed into B.

With the redefinition of subtraction and addition, the differences between two tour sequences obtain new meaning so equation (2) becomes applicable for TSP.

The second barrier that should be solved is how to define $c_0 v_k$ in (2) which can endue particles with the ability to globally visit new positions. As mentioned before, without this part or c_0 = 0, the particles would quickly converge to the local optimal solutions, which greatly effects the efficiency of the algorithm. But Chaos Optimization Algorithm (COA) with its ergodic property shows its advantages and utility in global searching. So COA can be a considerably useful way to keep particle's global searching ability.

Chaos optimization is realized through chaos variables which can be obtained by many ways. Here the Logistic Mapping Method [11] is selected. Its equation is as follows:

$$Z(k+1) = \mu Z(k)(1-Z(k)), \quad (3)$$

where μ is a control parameter. When μ=4, equation (3) is in chaos state, which means all values between 0 and 1 except the fixed point (0.25, 0.5, 0.75) are produced randomly by iteration. Furthermore with its sensitivity to the initial value, n chaos variables of different orbits can be obtained in the iteration after setting n different initial values between 1 and 0 to Z (k) in (3).

In the chaotic particle optimization, every particle is loaded with a memory capacity to store the chaos variables. For an m-city TSP, the memory capacity has m units which compose a vector $Z_i = (z_{i1}, z_{i2} \ldots z_{im})$ (z_{ia} is closely bound up with the city) [12]. At the beginning, Z_i is randomly initialized with the number between (0, 1).Then sort the sequence (z_{i1}, $z_{i2} \ldots z_{im}$) descending (or ascending) and a new sequence can be attained. According to the sequence, the number of the original position in Z_i can constitute another sequence which can be taken as the initial position $X_{i1}=f(Z_{i1})$ for every particle. During every iteration, every particle can gain a new chaos variable vector $Z_i(k+1) = 4 Z_i(k)(1-Z_i(k))$. Sort the $Z_i(k+1)$ and a new sequence can be obtained as well as a new particle position $X_i(k+1)$. Just as Table1 represents, as for the Z_i (0, 0.23, 0.73, 0.65, 0.54, 0.12), there is a tour sequence (1, 6, 2, 5, 4, 3). After one iteration, Z_{i+1}(0, 0.71, 0.79, 0.91, 0.99, 0.42) can be derived from Z_i .Consequently another tour sequence (1, 6, 2, 3, 4, 5) can be obtained. With the COA embedded in the proposed algorithm, particles can come to more unvisited places. It is the advantage of COA that can help particles jump out of the local optimum position and search global optimal one effectively.

Table1
An example of COA in chaotic PSO

City	1	2	3	4	5	6
Z_i	0	0.23	0.73	0.65	0.54	0.12
Z_{i+1}	0	0.71	0.79	0.91	0.99	0.42

Now the particle position equation can be given out as follows:

$$X_i(k+1) = C(X_i(k)) + c_1(pbest_i - X_i(k)) + c_2(gbest - X_i(k)) \quad (4)$$

where $C(X_i(k))$ means that update the new chaos variables according to the Logistic Mapping and get a new particle position vector $X_i(\kappa)'$ after sorting chaos variable sequence $pbest_i$-$X_i(k)$ represents the swap loop between the local optimal position pbesti and $X_i(k)$; αis a random variable between [0, 1] and $\alpha(pbest_i - X_i(k))$ means all the position swap can be reserved with the probability of α.Much the same, gbest-$X_i(k)$ represents the swap loop between the global optimal position gbest and $X_i(k)$; βis a random variable between [0, 1] and $\beta(pbest_i - X_i(k))$ means all the position swap can be reserved with the probability of β.The biggerα, βis, every position swaps in the swap loops can be reserved with more possibility. While "+" means swap the city positions according to the two parts of reserved position swaps

So in the redefined position updating equation (4) α, βstand for acceleration rates much the same with c_1, c_2 in standard PSO, which have much effect on the attraction of particles to the local and global optimal positions, while C(X (k)) endues particles with global searching ability to enlarge the searching space and jump out of the local best positions.

In summary, the procedures of proposed chaotic PSO algorithm for TSP can be described as follows:

Step1: Initialize particle swarm. Randomly set chaos variable vectors for every particle and after sorting n initial particle position (X_i (1)) can be obtained.

Step2: Update the local optimal position $pbest_i$ and global optimal position gbest; if the determination criterion meets, turn to step5.

Step3: Update the particle position according to (4):

1) Exercise the operation C(X (k)). Update chaos variable vectors and get a temporary position $X_i(\kappa)'$.

2) Calculate the swap loop LS_1 between $pbest_i$ and Xi (k).

3) With the stochastic value α, compute the reserved position swaps DS1.

4) Calculate the swap loop LS_2 between gbest and Xi (k).

5) With the stochastic valueβ, compute the reserved position swaps DS2.

6) Compute $X_i(\kappa)' + DS_1 + DS_2$. Sequentially swap city positions in the $X_i(\kappa)'$ in term of position swaps included in DS_1 and DS_2.

Step4: Turn to Step (2).

Step5: Output the global optimal solution gbest.

Ⅳ. NUMERICAL RESULTS

To verify the validity of the proposed algorithm, some simulation experiments were carried out on a PC with 3.0GHz processor and 512M memory and in C++ language. The instances in the experiments are selected from the TSPLIB library [13] .Each instance have been run for 100 times.

Table2
Results of the proposed algorithm for TSP

Problem	Opt	Best	Worst	Avg	Err(%)
Burma14	30.8785	30.9062	31.022	30.9245	0.14
Oliver30	423.7406	425.6542	457.2354	432.2231	2.00
att48	33522	33534	39679	34556	3.08
EIL51	426	427	466	442	3.76

The numerical results of four different problems are listed in Table2. The second column contains the best known optimal tour length for each problem. The first and second columns show the best and worst solutions that derived from the experiments respectively. The fifth column stands for the average tour length of 100 times experiments and the sixth column demonstrates the relative error which is calculated according the following equation(5):

$$Err = (Avg - Opt)/Opt * 100\% \quad (5)$$

From Table2 it is clear that the chaotic particle swarm optimization algorithm can be used to solve the TSP. For the 4 problems, the experiment best results are quite close to the well known shortest tour length. And the relative errors of average tour lengths of 100 experiments from Opt for each problem are acceptable. Therefore the proposed algorithm represents its validity.

However, from Table2 it can be easily found that the bigger the scale of problem is or the more cities the problem has, the bigger error the algorithm would come up with. For the Burma14 with 14 cities, the Err is only 0.14%; while for Oliver30 with 30, the Err jumps to 2.00%. And the Err keeps rising to 3.76% for EIL51 with 51 cities. So the proposed algorithm may be inherent with its limitation, which means that it can perform better in small scaled TSP problems.

Table3
Comparison between different algorithms for Oliver30

Algorithm	Oliver30		
	Best	Worst	Avg
SA	424.6918	479.8312	438.5223
GA	467.6844	502.5742	483.4572
ACS	441.9581	499.9331	450.0346
Proposed Algorithm	425.6542	457.2354	432.2231

Table 4
Comparison between different algorithms for att48

Algorithm	att48		
	Best	Worst	Avg
SA	35176	40536	34958
GA	38732	42458	38541
ACS	36532	42234	35876
Proposed Algorithm	33534	39679	34556

Table 3 and Table 4 display the comparisons between different algorithms including Simulated Annealing (SA), Genetic Algorithm (GA), Ant Colony System (ACS) and the proposed Algorithm. The results of SA, GA, and ACS are cited from reference [14]. It is obvious that all the best, worst and average results of the proposed algorithm are less than others for both Oliver30 and att48 so it justifies its effectiveness in solving TSP.

Ⅴ. CONCLUSIONS

PSO algorithm has attracted extensive attention of researchers and engineers for its advantages and comprehensive applicability, and been broadly used in many optimization areas especially in continuous problems. However, the standard equations in PSO can't be applied directly in discreet problems such as TSP. Focused on TSP, some new operators are introduced and a revised equitation is given out so as to extend PSO to combinatorial problems. Furthermore, COA is embedded in the chaotic particle swarm optimization algorithm to prevent particles from converging too quickly. Numerical simulation results of benchmark test problems validate its effectiveness and advantage over some other algorithms such as conventional Simulated Annealing, Genetic Algorithm and Ant Colony System.

REFERENCES

[1] J.Kennedy, R.C.Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, vol.4, IEEE Services Center, Piscataway, NJ,1995, pp.1942-1948.

[2] P.J.Angeline, Evolutionary optimization versus particle swarm optimization: philosophy and performance differences, Evolutionary Programming 7 (1998), pp.601-610.

[3] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters 85 (2003), pp.317–325.

[4] J.Kennedy, R.C.Eberhart, Discreet binary version of the particle swarm algorithm, in: Proceedings of the IEEE International Conferences on Systems, Man and Cybernetics, vol.5, Orlando, Florida,USA, 1997,pp.4104-4108.

[5] X.H. Shi et al., Particle swarm optimization-based algorithms for TSP and generalized TSP, Information Processing Letters (2007), doi:10.1016/j.ipl.2007.03.010

[6] LU Hui-juan, A new optimization algorithm based on Chaos, J Zhejiang Univ SCIENCE A 2006 7(4), pp.539-542.

[7] M. Bellmore, G.L. Nemhauser, The traveling salesman problem: a survey, Operations Research 16 (1968), pp. 538–558.

[8] M. Clerc, Discrete particle swarm optimization illustrated by the traveling salesman problem, http://www.mauriceclerc.net, 2000.

[9] T. Hendtlass, Preserving Diversity in Particle Swarm Optimization, in: Lecture Notes in Computer Science, vol. 2718, Springer, 2003, pp. 4104–4108.

[10] K.P. Wang, L. Huang, C.G. Zhou, W. Pang, Particle swarm optimization for traveling salesman problem, International Conference on Machine Learning and Cybernetics 3 (2003) 1583–1585.

[11] Li, B., Jiang, W.S., Chaos optimization method and its application, Control Theory and Applications, 4(1997), pp, 613-615 (in Chinese).

[12] Zhang, G.P., Wang, Z.O., Yuan, G.L.,A Chaotic Search Method for a Class of Combinatorial Optimization Problems, Systems Engineering-Theory & Practice 2001 21(5), pp.102-105.

[13]http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/

[14] S.Gao and J.Yang, Swarm Intelligent Algorithm and Application, China Waterpub Press, 2006, pp.56-57