Applied Mathematics & Information Sciences
An International Journal

A Novel Discrete Cuckoo Search Algorithm for Spherical Traveling Salesman Problem

Xinxin Ouyang[1], *Yongquan Zhou*[1,2], *Qifang Luo*[1] *and Huan Chen*[1]

[1]College of Information Science and Engineering, Guangxi University for Nationalities, Nanning Guangxi 530006 P.R. China
[2]Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning 530006, P.R. China

Received: 20 Sep. 2012; Revised 26 Oct. 2012; Accepted 28 Nov. 2012
Published online: 1 Mar. 2013

Abstract: In this paper, we propose a novel discrete cuckoo search algorithm (DCS) for solving spherical Traveling Salesman Problem (TSP) where all points are on the surface of a sphere. The algorithm is based on the Lévy flight behaviour and brood parasitic behaviour. The proposed algorithm applies study operator, the ”A” operator, and 3-opt operator to solutions in the bulletin board to speed up the convergence. Optimization results obtained for HA30 (an instance from TSPLIB) and different size problems are solved. Compared with GA, DCS is better and faster.

Keywords: discrete cuckoo search algorithm (DCS), spherical geometry, spherical TSP

1. Introduction

The Traveling Salesman Problem (TSP) is classical and most widely studied problem in the field of Combinatorial Optimization and attracts computer scientists, mathematicians, and others [1]. The idea of problem is to find shortest route of salesman starting from a given city, visiting n cities only once and finally arriving at origin city. Euclidean TSP is a NP-hard problem as the search space is huge viz. $n!$, so that it is calculating infeasible to obtain optimal solution to the problem. [2] used local search operators best part collect, C2Opt, smallest square evolution for TSP, the results perform good. [3] Combined the GGA and LGA as an operator to solve TSP. [4] proposed a greedy heuristic algorithm with regret and the cheapest insertion algorithm for TSP.

In metric TSP the nodes lie in a metric space (i.e., the distances satisfy the triangle inequality), whereas in Euclidean TSP the nodes lie in R^2(or more generally, in R^4 for each d). Much of work on TSP lie in R^2 has been done with heuristic algorithms to produce an optimal or close to optimal solution. The commonly used heuristic approaches are greedy algorithms; 2-opt, 3-opt[1]; simulated annealing(SA); ant colony algorithm(ACA)[5]; genetic algorithm (GA); particle swarm optimization (PSO)[6]; Artificial neural network (ANN) [7]. While the work on multi dimensional TSP is seediness [8] solved 3D-TSP for the multi-dimensional city location. [9] solved TSP with genetic algorithm on a sphere. [10] has proved TSP^S (TSP on the curve) is NP hard and can be polynomial time approximated. [11] has described the distribution of distances between random points on a sphere. [12] gave an upper bound and a lower bound of the optimal value to the random spherical TSP.

In this paper, we propose a novel discrete cuckoo search algorithm (DCS) for solving the Euclidean TSP, where all points are on the surface of a sphere. The cuckoo search (CS) algorithm proposed by Yang and Deb (2009, 2010) is based on Lévy flight behavior and brood parasitic behavior [13–15]. CS performed excellent on function optimization, engineering design, training neural network and other continuous target optimization problems [16–18]. Now CS is proposed to solve knapsack problem and nurse scheduling problem [19,20].

The proposed algorithm does not save the initial routes in the bulletin board, until the segment between each city and its around cities partially reversed. In every generation, every cuckoo searches a new nest and abandons the old one to decrease the TSP route. In order to accelerate the convergence, the proposed algorithm applies study opera-

* Corresponding author: e-mail: yongquanzhou@126.com

tor, the ”A” operator, and 3-opt operator to solutions in the bulletin board. The search-new-nest operator and the study operator derived from the idea of the Inver-Over operator [21]. Optimization results obtained for HA30 (an instance from TSPLIB) and different size problems are presented.

2. Spherical Geometry

A sphere is a set of points in 3D space equidistant from a point called the enter of the sphere [22]. The distance from the center to the points on the sphere is called the radius r of the sphere. All the points satisfying the following.

$$x^2+y^2+z^2=r^2 \tag{1}$$

2.1. Representation of a Point on the Surface of a Sphere

We can represent coordinate positions on a surface using the following cartesian vector point function [9]:

$$P(u,v)=(x(u,v),y(u,v),z(u,v)) \tag{2}$$

Each of the coordinate is a function of the two surface parameters u and v. In most cases, we can normalize the three coordinate functions so that parameters $u,v\in[0,1]$. A spherical surface with radius r and center at the coordinate origin can be described with the equations:

$$x(u,v) = r\cos(2\pi u)\sin(\pi v) \tag{3}$$
$$y(u,v) = r\sin(2\pi u)\sin(\pi v) \tag{4}$$
$$z(u,v) = r\cos(\pi v) \tag{5}$$

Parameter u describes lines of constant longitude over the surface, while parameter v describes lines of constant latitude. We used a unit sphere which is simply a sphere of radius one for simplification of calculations. Results or path lengths can be evaluated easily for spheres of radius r, multiplying by r.

Equations 3 to 5 can transform to Equations 6 to 7, we obtain

$$u = \arctan(y/x)/2\pi \tag{6}$$
$$v = \arccos(z/r)/\pi \tag{7}$$

where x,y,z are parameters.

Note: In the Equations 6, where $u\in[-0.25,0.25]$, function is not one-to-one mapping, so we need do some changes to make $u\in[0,1]$. If $x<0$, $u\in[0.25.0.75]$. If $x>0$ and $y>0$, $u\in[0,0,25]$. If $x>0$ and $y<0$, $u\in[0.75,1]$.

2.2. Finding Geodesics between All Pairs of Points on the Surface of Unit Sphere

A great circle is the intersection a plane and a sphere where the plane also passes through the center of the sphere [23]. A geographic example of a great circle is the earth’s equator. Great circles become more important when we realize that the shortest distance between two points on the sphere is along the segment of the great circle. This shortest path is called a geodesic. The curves that minimize the distance between points are called geodesics on any surface. Every meridian of longitude is exactly half a great circle. The parallels of latitude are smaller circles except for the equator.

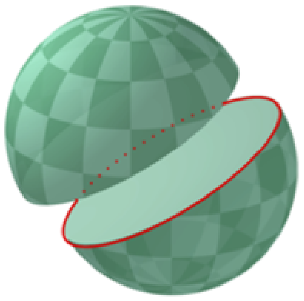

Figure 1 A great circle divides the sphere in two equal hemispheres

Shortest distance between two points (P_1,P_2) on a spherical surface is along the arc of a great circle. So, it can be used the value of angle theta (θ) in radians between two vectors $\overrightarrow{v_1}=(x_1,y_1,z_1)$ and $\overrightarrow{v_2}=(x_2,y_2,z_2)$. Scalar product of two vectors is

$$\overrightarrow{v_1}\bullet\overrightarrow{v_2}=|\overrightarrow{v_1}||\overrightarrow{v_2}|\cos\theta \tag{8}$$

where θ is the angle (smaller one) between two vector directions. Scalar product is calculated as

$$\overrightarrow{v_1}\bullet\overrightarrow{v_2}=x_1x_2+y_1y_2+z_1z_2 \tag{9}$$

So shortest distance formula is

$$\hat{d}_{12}=r\theta \tag{10}$$

From Equations 8 to 10, we can get Equations 11:

$$\hat{d}_{12}=r\arccos\left(\frac{x_1x_2+y_1y_2+z_1z_2}{r^2}\right) \tag{11}$$

Distance from any point P_i to any point P_j is the same as the distance from point P_j to point P_i on the sphere. In this step, $N\times N$ symmetric distance matrix D is

$$D = \begin{bmatrix} \hat{d}_{11} & \hat{d}_{12} & \cdots & \hat{d}_{1N} \\ \hat{d}_{21} & \hat{d}_{22} & \cdots & \hat{d}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{d}_{N1} & \hat{d}_{N2} & \cdots & \hat{d}_{NN} \end{bmatrix} = \begin{bmatrix} \infty & \hat{d}_{12} & \cdots & \hat{d}_{1N} \\ \hat{d}_{12} & \infty & \cdots & \hat{d}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{d}_{N1} & \hat{d}_{N2} & \cdots & \infty \end{bmatrix} \quad (12)$$

which gives the geodesics between all pairs of points on the spherical surface was computed. Especially, $\hat{d}_{i,i} = \infty$, means that city to itself is unreachable.

3. Cuckoo Search Strategies

In nature, cuckoos use an aggressive strategy of reproduction that involves the female hack nests of other birds to lay their eggs fertilized [13]. Sometimes, the egg of cuckoo in the nest is discovered and the hacked birds discard or abandon the nest and start their own brood elsewhere. The CS is based on the following three ideas lased rules (Yang and Deb, 2010):

(1) Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest.

(2) The best nests with high quality of eggs (solutions) will carry over to the next generations.

(3) The number of available host nests is fixed, and a host can discover an alien egg with a probability $p_a \in (0,1)$. In this case, the host bird can either throw the egg away or abandon the nest so as to build a completely new nest in a new location.

The last assumption can be approximated by a fraction p_a of the n nests being replaced by new nests (with new random solutions at new locations). The generation of new solutions x_{t+1} is done by using a Lévy flight. Lévy flights essentially provide a random walk while their random steps are drawn from a Lvy distribution for large steps which has an infinite variance with an infinite mean. Here, the consecutive jumps (steps) of a cuckoo essentially form a random walk process [16] which obeys a power-law step-length distribution with a heavy tail.

$$x_i^{t+1} = x_i^t + \oplus L(\lambda) \quad (13)$$

$$L(\lambda) \oplus u = t^{-\lambda} \quad (14)$$

where $\alpha > 0$ is the step size which should be related to the scales of the problem of interest. Generally, we take $\alpha = O(1)$. The product $\oplus$ means entry-wise multiplications. This entry-wise product is similar to those used in PSO, but here the random walk via Lévy flight is more efficient in exploring the search space as its step length is much longer in the long run.

4. The Proposed Algorithm for TSP on A Sphere

Discrete cuckoo search algorithm is recently proposed to solve knapsack problem [19] and nurse scheduling problem [20]. In this paper we will propose a discrete cuckoo search algorithm to solve Euclidean TSP where all points are on the surface of a sphere.

4.1. Discrete Cuckoo Search Algorithm

Discrete cuckoo search algorithm (DCS) produce a city number every call. If given a city C as input, DCS will produce another city D as output. It means a cuckoo will fly from C to D, and the probability relevant to their distance. Detailed description is as following.

Firstly, suppose the cuckoo at present locate at city C, and the distances between city C and m cities are given by $A = [d_{C,1}, d_{C,2}, \cdots, d_{C,m}]$. Secondly, calculate

$$p = \frac{u}{|v|^{1/\lambda}}, \; 1 < \lambda \leq 3 \quad (15)$$

for real elements of λ is in the range $[0, +\infty]$.

1) When $\|p\| < 1$, If $d_{C,D} = \min(A)$, choose the nearest cite D to visit next time.

2) When $\|p\| < 1$, compute

$$L = d_{C,D} \times p \quad (16)$$

$$B = \{j \mid d_{C,j} \leq L, \; 1 \leq j \leq m\} \quad (17)$$

$$d_{C,E} = \max(d_{C,j}), \; j \in B \quad (18)$$

Note that L is a flight length, set B contains all cities a flight can reach, $d_{C,E}$ means a cuckoo always choose city E (which is the forest city but within L) to visit after visited city C.

The algorithm transforms the random-walk vector (p) to a scalar (L) and chooses a city (no farther than L) as the next visited city. Here the consecutive jumps (steps) of a cuckoo essentially form a random walk process which obeys a power-law step-length distribution with a heavy tail, avoiding the tedious time consuming of probability density formula.

4.2. Representation of Solution

A cuckoo flies between cities obeying Lévy distribution. At the beginning, a cuckoo visited n cities once and saved its traved order in the bulletin. *Individual* is a nest which represents the first city (the end city) of a route (solution). *Population* is all n nests which represent n cities. A solution represents an available route. Bulletin board save all n solutions and refresh by offspring's generation by generation. In every generation, n cuckoos start from their nests flying around to get new nests replacing of the olds. By partially reversing of segment of route (start from old nest to the new nest) the algorithm eliminates crossover edges to find shorter route (better solution). *Fitness* defined as its route length: the smaller, the better. To improvement its effectiveness, the proposed algorithm applies study operator, the "A" operator, 3-opt operator to the bulletin board. The search-new-nest operator and study operator derived from the idea of the Inver-Over operator [21].

4.3. Steps of DCS to TSP

The proposed discrete CS algorithm to solve TSP problem described as follow:

Discrete Cuckoo Search to solve Traveling Salesman Problem:

```
Begin
    Objective function $f(\pi)$, city distances array;
    Initial a population of $n$ host nests (cities)) $x_i$,
    $i = 1,2,\cdots,n$;
    Cuckoos fly via DCS to find an initial route (solution).
    Mend initial solutions and saved in the bulletin board.
    Evaluate the route length (fitness) of solutions $F_i$;
While ($t < MaxGeneration$) or (Stop Criterion)
        Cuckoos start from their nest to search new nests;
    If $F_i' < F_i$
        Replace old nest by new one,
        Reverse the segment between old nest and new one,
        Refresh bulletion board.
    End
        Study operator to the bulletin board,
        The "A" operator to the bulletin board.
        3-opt operator to the bulletin board.
        Host birds abandon $p_a \in (0,1)$ nests, and search $p_a$
        new nests;
        Refresh the bulletin board and keeping the best solutions (and nests).
        Rank the solutions, and find the best route (solution).
        $t = t+1$;
End While
        Postprocess results and visualization.
End
```

Following we describe detail how operators work.

4.3.1. Initial Solution

Cuckoos starting from its nest fly to a new city via DCS. Suppose this cuckoo arrive at city C, the algorithm regards C as take-off city and choose a new city D to visit next time; then regards D as take-off city and choose another city to visit$\cdots$. until all cities have been visited. Every city can be visited only once and be signed unreachable after visited.

After that the algorithm saves a travelling order as a solution. Next, for every C equal to $1,2,\cdots,m$, produce an other city D, judge whether reversing the segment between C and D can decrease the route length. If yes, save the new solution (route) in the bulletin board. If not, save the old one.

4.3.2. The Search New Nest Operator

A cuckoo starting from its nest fly to a new city via DCS, checking whether the route decrease by reversing the segment between take-off city and touch-down city or not. If yes, the new route will be saved in the bulletin board; or not it will continuously fly and check until touch-down city adjoin the starting nest. At the end, the last visited city will be regard as new nest in the next generation. Note that, the algorithm only reverse the segment between two dis-adjoin cities.

Assume that the current individual is

$$S_1(1,2,3,4,5,6,7,8)$$

Cuckoo's nest is city 2, and touch-down city is 7. The result of inversion of segment starts after city 2 and terminates after city 7 is

$$S_1 \rightarrow S_1'(1,2,7,6,5,4,3,8)$$

If the reversing result is shorter ($F_1' < F_1$), new route will be saved, or not cuckoo starting from city 7 (see as take-off city) will fly to a new city (see as touch-down city). But if the touch-down city adjoin with its nest or find a shorter route, cuckoo stops fly. For example, suppose the current individual is

$$S_2(1,5,6,7,2,3,4,8)$$

Cuckoo's nest is city 7; this cuckoo starts from city 4 just now, and arrive at city 6. Since city 6 is already before (adjoin with) city 7, the continual flights (inversions) terminate.

When searching new nest operator finished, the inversion may happen several times or may not once (that means the route may not change). In either case, a new nest is produced. Note again, a cuckoo's every movation is looking for the next city to visit. A far-away city is selected in small probability and neighboring cities is selected very likely. The selected probability and distance from take-off city to touch-down city fit Lévy distribution. This method instructs the proposed algorithm works effectively. Note also, the search-new-nest operator of proposed algorithm is different from the one of basic CS, because the former is to find a proper visiting order. While the city location has nothing to do with route order, the order of visiting has great influence on solution. A cuckoo may fly several times until find a good order.

4.3.3. The Abandon Operator

The search new nest operator and abandon operator imitate the cuckoo's brood behavior. The former carried out by cuckoos, while the latter performed by host birds. If the generated number $r \sim U[0,1]$ is less than a given fixed probability $p_a \in (0,1)$, a host bird abandon its nest and search for a new one. Note, the abandon probability of old nest irrelevant to its coordinate but relate to the route length (fitness). The longer the route, the more likely its nest is abandoned. Though the old nest is abandoned, but the corresponding solution won't change; solution will change when searching a new nest.

4.3.4. The Study Operator

The study operator acts on two solutions in the bulletin board at the same time. Assume they are S_1 and S_2. If a substring in the S_2 is better than S_1, copy it to S_1 and change the bad of S_1. If this substring is worse, copy corresponding substring from S_1 to S_2 and mend the bad of S_2. They study well from each other and change bad.

For example, assume that S_1 is

$$S_1(1,2,3,4,5,6,7,8)$$

Assume that S_2 is

$$S_2(1,3,5,6,8,4,2,7)$$

where S_1, S_2 are choose randomly from the board. Suppose the current city is $C = 2$, the operator search for the city 7 in the S_2 which is next to 2, so the substring is "27". Thus the operator reverse the segment starts *after* city 2 and terminates *after* city 7 in the S_1. Consequently, a new solution is

$$S_1 \to S_1'(1,2,7,6,5,4,3,8)$$

If the fitness of S_1' is less than $S_1(F_1' < F_1)$, S_1' would be saved. If not, it means that the substring "27"+"38" is worse than "23"+"78". So the corresponding substring is "78". Therefore the segment for inversion in the S_2 starts after city 7 and terminates after city 8. But "8" is in front of "7", the operator reverse the segment starts *before* 8 and ends *before* 7. As a result, a new solution is produced

$$S_2 \to S_2'(1,3,5,6,2,4,8,7)$$

If the fitness of S_1' is less than S_2, S_2' would be saved. The study operator make C equal to $1,2,\cdots,m$ in turn. For each value, the inversion caused by "study well" and "change bad" has been applied several times in the process of executing.

4.3.5. The "A" Operator

The study operator acts on only one solution in the bulletin board every time. The operator can mend the route well which looks like "A". Assume a solution is

$$S(1,2,3,4,5,6,7,8).$$

A cuckoo fly from city $C = 2$ to city 6 via DCS, judge whether delete 2 from original order and inserting 2 after 6 can decrease the route length. After insertion, new solution is produced

$$S \to S'(1,3,4,5,6,2,7,8).$$

If the new route is better $(F' < F)$, save it into board; or not don't change. Following is a schematic diagram

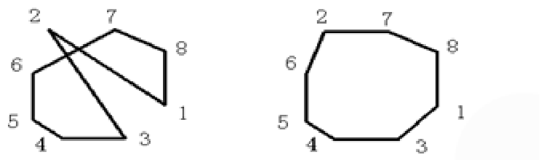

Figure 2 A schematic diagram of the "A" operator

The "A" operator make C equal to $1,2,\cdots,m$ in turn. For each value, the inserting executed most once.

4.3.6. The 3-opt Operator

The 3-opt operator acts on a solution once. It removes 3 edges from the tour and divided the tour into 3 pieces. It means there are two ways of reconnecting the 3 substring into a valid tour. If a tour is 3 optimal it is also 2 optimal (Helsgaun) [24].

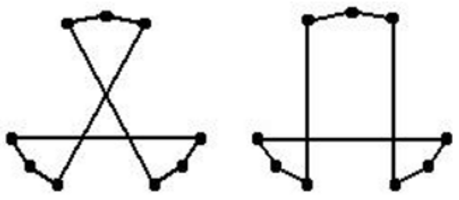

Figure 3 The 3-opt operator reconnect the substrings in two ways

5. Simulations

We test our algorithm in the two simulations: Part A and Part B. Part A show an instance HA30 from TSPLIB. In Part B represents instances which were obtained for $N = 100,150,200,250,300,350,400$ points on the unit sphere, and a new random point set was generated for each trial.

Simulations were repeated 10 times for each value of N. The coordinate of these points are saved and the best solution is presented. Hardware environment: INSPIRON 530, 1G memory, 2.2 GHZ. Software environment: Microsoft Windows XP SP3. Programming environment: MATLAB 2008a.

Part A

Population size $n = 4$, abandon rate $p_a = 0.5, \beta = 2, r = 39.59$. There are 30 cities in the HA30. We use these coordinates and distances matrix as input. If the route length of the best solution hasn't change for 10 generations, the algorithm stops. Figure 4 show the best route from different directions.

Please focus on the red part of sphere, which rotates in a direction that is horizontal counterclockwise. The shortest route length of HA30 is 503. And solution is:
"$1 \to 13 \to 22 \to 24 \to 3 \to 18 \to 11 \to 6 \to 2 \to 4 \to 14 \to 28 \to 30 \to 9 \to 10 \to 25 \to 27 \to 12 \to 17 \to 20 \to 8 \to 19 \to 16 \to 21 \to 26 \to 5 \to 23 \to 7 \to 1$"

Table 1 The comparison results of GA and DCS to solve $N = 100, 150, 200, 250$ points on the surface of a sphere

Generation	Number of Points							
	100		150		200		250	
Size	GA	**DCS**	GA	**DCS**	GA	**DCS**	GA	**DCS**
10	90.1	**25.5**	157.6	**31.1**	227.2	**36.5**	299.1	**41.1**
	194	**412**	434	**714**	337	**940**	799	**404**
20	72.0	**25.4**	132.9	**31.1**	187.7	**36.4**	265.2	**40.9**
	467	**120**	872	**952**	657	**697**	366	**506**
30	53.0	**25.3**	100.9	**30.9**	162.8	**36.4**	224.2	**40.8**
	306	**846**	841	**975**	873	**538**	341	**280**
40	42.6	**25.3**	82.2	**30.9**	141.7	**36.3**	200.0	**40.7**
	922	**409**	588	**768**	211	**856**	556	**622**
50	37.1	**25.3**	70.5	**30.9**	115.1	**36.3**	165.5	**40.7**
	942	**341**	737	**735**	155	**792**	674	**557**

Table 2 The comparison results of GA and DCS to solve $N = 300, 350, 400$ points on the surface of a sphere

Generation	Number of Points					
	100		150		200	
Size	GA	**DCS**	GA	**DCS**	GA	**DCS**
10	374.1841	**45.6111**	441.3127	**45.6336**	532.6288	**52.2950**
20	333.5504	**45.3540**	393.6967	**45.3357**	472.2036	**51.9555**
30	288.0024	**45.2773**	353.1749	**45.2284**	439.4960	**51.8304**
40	259.6622	**45.2367**	323.2888	**45.1079**	393.7457	**51.7429**
50	226.1183	**45.2005**	291.1789	**45.0918**	354.3750	**51.6725**

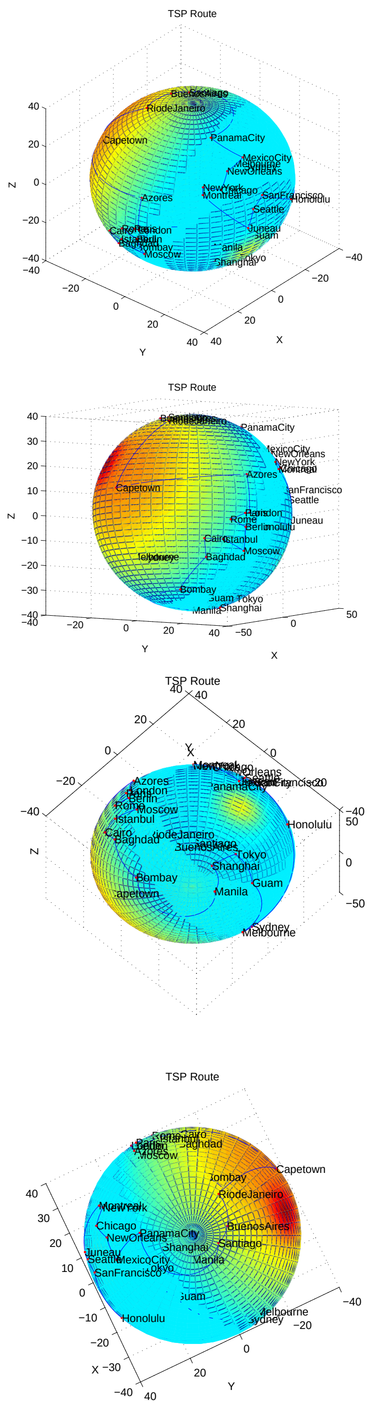

Figure 4 The best route for HA30 is 503, 10 generations

So the best traveling route is:
"Azores→London→Paris→Rome→Berlin→Moscow →Istanbul→Cairo→Baghdad→Bombay→Manila→ Shanghai→Tokyo→Guam→Melbourne→Sydney→ Honolulu→ SanFrancisco→Seattle→Juneau→Montreal

→New York→Chicago →New Orleans→Mexico City→ Panama City→Santiago→ Buenos→Aires→Capetown→ Azores".

Part B

Table 1 and Table 2 are obtained in the 10 runs. Population size $n = 200$ abandon rate $p_a = 0.5, \beta = 2, r = 1$. Data in these two tables is average spherical TSP tour lengths. The results of DCS signed by bold letters. From Table 1 and Table2, we note that DCS is far better than GA.

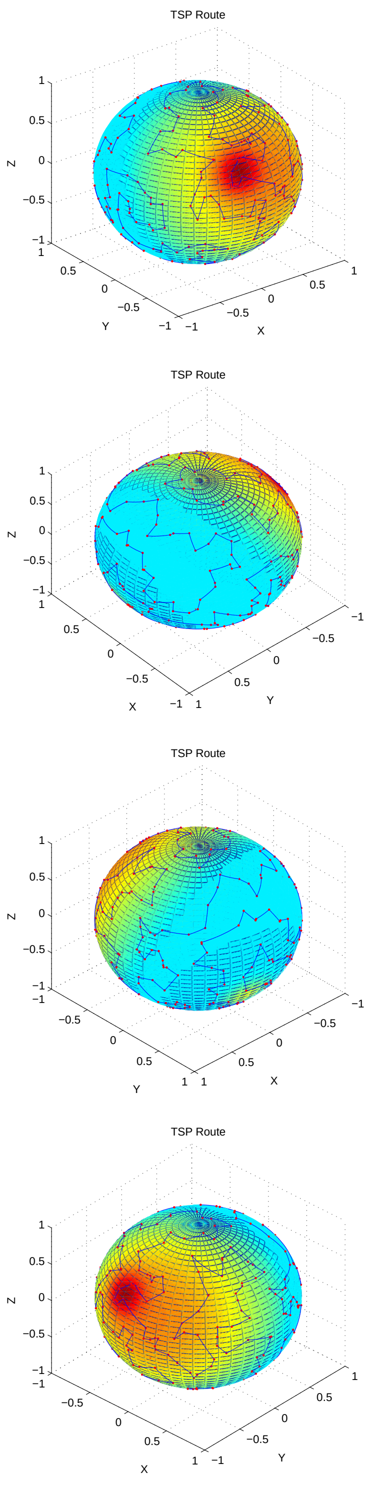

Figure 5 The best route for HA30 is 503, 10 generations

6. Conclusions

Two important contributions of this paper are to be the first serious application of HA30 from TSPLIB which is on the earth and to propose a simple and efficient cuckoo search algorithm based solving method for spherical TSP which is different from 3-dimensional Euclidean TSP. Cuckoo search algorithm is based on the Lvy flight behavior and brood parasitic behavior. Another contribution is to develop MATLAB tool to experiment TSP on the surface of the sphere.

The search new nest operator and study operator are different from the inver-over operator though they borrow the idea of it. The biggest difference is that the operators in the proposed algorithm applying DCS to search around for the current city. In addition, the study operator is more effective than original crossover operator, because it not only can improve the solution S_1, but also can improve the solution S_2.

Acknowledgement

This work is supported by National Science Foundation of China under Grant No. 61165015. Key Project of Guangxi Science Foundation under Grant No. 2012GXNSFDA0530 28, Key Project of Guangxi High School Science Foundation under Grant No. 20121ZD008 and the Funded by Open Research Fund Program of Key Lab of Intelligent Perception and Image Understanding of Ministry of Education of China under Grant No. IPIU01201100. Thanks for the help.

References

[1] Rajesh Matai1. Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches. Traveling Salesman Problem, Theory and Applications, 1-24 (2011).

[2] Peng Gang, Ichiro Iimura, Shigeru Nakayama. An Evolutionary Multiple Heuristic with Genetic Local Search for Solving TSP. International Journal of Information Technology, **14**, 1-11 (2008).

[3] Peng Gang. Genetic Local Search Based on Genetic Recombination: A Case for Traveling Salesman Problem.LNCS, **3320**, 202-212(2004).

[4] Refael Hassin. Greedy Heuristics with Regret, with Application to the Cheapest Insertion Algorithm for the TSP. Oper. Res. Lett. **36**, 243-246 (2008).
[5] Ji Junzhong. A Fast Ant Colony Optimization Algorithm for Traveling Salesman Problems. Journal of Computer Research and Development, **46**, 968-978 (2009).
[6] Yuhong Duan, Ying Sun. A Particle Swarm Optimization Algorithm with Ant Search for Solving Traveling Salesman Problem. International Conference on Computational Intelligence and Security. 11-14 (2009).
[7] Arindam Chaudhuri and Kajal De. A Study of Traveling Salesman Problem Using Fuzzy Self Organizing Map. Industrial and Information Systems, 8-10 (2008).
[8] S.Takahash. The SOM-TSP Method for the There-Dimension City Location Problem. Proceedings of the 9th international Conference on Neural Information Processing, **5**, 2552-2555 (2002).
[9] Aybars Uğur. Genetic Algorithm Based Solution for TSP on a Sphere. Mathematical and Computational Applications, **14**, 219-228 (2009).
[10] Wang Gang, Luo Zhigang. A polynomial time approximation scheme for the traveling salesman problem in the sphere surface. Proceedings of the 3rd International Conference on Computer Engineering and Applications. IEEE Press. (2011).
[11] Cinna Lomnitz. On the Distribution of Distances between Random Points on a Sphere. Bulletin of the Seismological Society of America, **85**, 951-953(1995).
[12] Wang Gang, Luo Zhigang. Spherical traveling salesman problem constant and its experimental analysis. Application Research of Computer, **28**, 1001-3695 (2011).
[13] Xin-She Yang, Suash Deb. Cuckoo Search via Lvy Flights, Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009, India), IEEE Publications, 210-214(2009).
[14] Xin-She Yang, Suash Deb. Engineering Optimisation by Cuckoo Search. International Journal of Mathematical Modelling and Numerical Optimisation. **3**, 330-343 (2010).
[15] Xin-SheYang, Nature-Inspired Metaheuristic Algorithms, Luniver Press. 2010.
[16] Ramin Rajabioun. Cuckoo Optimization Algorithm. Applied Soft Computing. **11**, 5508-551 (2011).
[17] Ehsan Valian. Improved Cuckoo Search Algorithm for Feedforward Neural Network Training. International Journal of Artificial Intelligence & Applications. **2**, 36-43 (2011).
[18] S.Walton. Modied Cuckoo Search: A New Gradient Free Optimisational Gorithm, Chaos, Solitons & Fractals. **44**, 710-718(2011).
[19] Abdesslem Layeb. A novel quantum inspired cuckoo search for knapsack problems, Int. J. Bio-Inspired Computation. **3**, 297-305 (2011).
[20] Lim Huai Tein. Recent advancements of nurse scheduling models and a potential path, Proceedings of the 6th IMT-GT Conference on Mathematics, Statistics and its Applications. 395-409 (2010).
[21] Guo T, Michalewicz Z. Invor-Over Operator for the TSP-Proceedings of the 5th Parallel Problem Solving from Nature Conference, Berlin: Springer,1498-1520 (1998).
[22] Wikipedia, Spherical coordinate system. http://en.wikipedia.org/wiki/Spherical_coordinate_system(2012).
[23] Wikipedia, Great circle. http://en.wikipedia.org/wiki/Great_circle, 2012.
[24] Johnson D.S. McGeoch L.A. The Traveling Salesman Problem: A Case Study in Local Optimization. Local Search in Combinatorial Optimization, 215-310 (1997).

Yongquan Zhou Ph.D & Professor. He received the MS degree in Computer science from Lanzhou University in 1993, and the PhD degree in Computation Intelligence from the Xiandian University in 2006. He is currently a professor in Guangxi University for Nationalities. His research interests are in the areas of computation intelligence, neural networks, and intelligence information processing.

Xinxin Ouyang MS. Her research interests are in the areas of swarm intelligence, functional networks, and information systems.

Qifang Luo Associate Professor. She received the MS degree in computer science from Guangxi University in 2005, Her research interests are in the areas of computation intelligence, and intelligence information systems.

Huan Chen MS. His research interests are in the areas of swarm intelligence algorithm, functional networks, and information systems.