An Adaptive Mutation Multi-particle Swarm Optimization for Traveling Salesman Problem

Ming-fang GAO [1, a], Xue-liang FU [1,b *], Gai-fang DONG [1,c], Hong-hui LI [1,d]

[1]College of Computer and Information Engineering, Inner Mongolia Agricultural University, Hohhot, 010018, P.R. China

[a]356293589@qq.com, [b]fxliang@126.com , [c]donggf@imau.edu.cn , [d]lihh@imau.edu.cn

Keywords: Particle Swarm Optimization; Adaptive mutation; Multi-particle swarm; Traveling Salesman Problem.

Abstract. Traveling Salesman Problem (TSP) is a well-known NP-hard combinatorial optimization problem. The Particle Swarm Optimization has been proven to succeed in lots of problems, but the PSO algorithm is challenging due to a variety of factors such as easy to fall into local optimal solution and the convergence speed is slow in the later. In this paper, we propose an adaptive mutation multi-particle swarm optimization algorithm (AMPSO) to the TSP. The experimental results show that the proposed algorithm can achieves better performance compared to the standard PSO method to solve the TSP.

Introduction

Particle swarm optimization (PSO) is a modeling method based on swarm intelligence that was developed by Dr. Kennedy and Eberhart who was inspired by nature bird populations and proposed actions in 1995 [1]. Compared with other swarm intelligence optimization algorithms, PSO is more intuitive and simple. It only needs to set fewer parameters and has more efficient.

Existing particle swarm algorithms mainly include particle swarm optimization algorithm with constriction factor, binary particle swarm algorithm[2], niching particle swarm optimization algorithm [3] and so on. The most standard algorithm is the global search particle swarm algorithm (model Gbest) which is introduced by Eberhart and Shi [4].

Traveling salesman problem (TSP) [5] is a famous NP-hard problem. TSP can be described as assuming there is a travel businessman to visit n cities, he must choose the path that he will go, the restriction of the path is to visit each city only once, and finally he must go back to the original departure city. The path selection objective is to get the minimum value among all paths. Due to the TSP is a classic combinatorial optimization problem[6], it has many applications in practice engineering, such as logistics distribution path optimization [7]. At present, the main method used to solve TSP is heuristic algorithm, such as genetic algorithm [8], ant colony algorithm [9], etc.

Particle Swarm Optimization(PSO)

Particle swarm optimization algorithm initializes a random population, and then constantly updates the population to search for the optimal solution. In addition, each potential solution which is called a particle is assigned a random velocity, these particles will fly over the entire solution space. The velocity of each particle is updated according to the best position of the particles by the iteration.

When the particles are flying over the solution space, the particle will fly to the optimal solution which is found by the adjacent particles and particle itself. If the search space is D dimensional and M particles form a population, then the position vector of the i_{th} particles in the solution space is $X_i = (x_{i1} , x_{i2} , \dots, x_{iD})^T$. The position of each particle is a possible solution. The velocity of each particle is also a D dimensional vector, expressed as $V_i = (v_{i1} , v_{i2} , \dots, v_{iD})^T$ $(i= 1, 2, \dots, M)$.The particle which finds the global optimal value expressed as $P_{gbest} =(p_{g1} ,p_{g2} , \dots, p_{gD})$. Each particle also keeps track of the solution of the optimal position which is searched by itself, and the optimal location which is searched by the i_{th} particle is represented as $P_{besti} = (p_{i1}, p_{i2},... p_{iD})$.The speed and position of the particles are updated according to the following formula:

$$v_{id}^{k+1}=w * v_{id}^{k}+c_1*rand()*(p_{id}-x_{id}^{k})+c_2*rand()*(p_{gd}-x_{id}^{k}), \quad (1)$$
$$x_{id}^{k+1}=x_{id}^{k}+v_{id}^{k+1} \quad (2)$$

Where ω is the inertia factor, p_{id} is the best solution this particle has reached, p_{gd} is the global best solution of all the particles, c_1 and c_2 is the accelerating factor which is the weight determine the influence of p_{id} and p_{gd}, and k is the maximum iteration number.

Adaptive Mutation Multi-Particle Swarm Optimization

Multi Particle Swarm Competition Mechanism

Despite the PSO has many advantages, such as simple operation, less parameters, faster convergence rate, the particle's convergence has the potential to lose the diversity of the population of particles [10] and causes the convergence rate of the algorithm in the later period of the algorithm is obviously decreased, which makes the accuracy of the algorithm is lower than other algorithms.

In order to overcome these problems, proposed the strategy which divides the particle swarm into several sub populations to improve the accuracy and convergence of the algorithm. Multi particle swarm competition mechanism divides the particle swarm into several sub populations, each sub population search in the solution space independently of each other. Then the particle "convergence" will be avoided in the search process of particle swarm and the algorithm can maintain the diversity of the population in the iterative process. The speed updating formula of the adaptive mutation multi particle swarm optimization is as follows:

$$v_{id}^{k+1}=w*v_{id}^{k}+c_1*rand()*(p_{id}-x_{id}^{k})+c_2*rand()*(p_{lgd}-x_{id}^{k}), \quad (3)$$

Adaptive Mutation Strategy

In the process of searching the optima solution, when particle is close to the optimal value, its search speed will become smaller, particle is easy to fall into local minimum and unable to jump out. This situation will result in that particle swarm can not search for new global extreme in the solution space. Therefore, an adaptive mutation strategy is proposed to update the speed and position of the particles which are in the lower fitness value of the half of each sub population.

Sorting all the particles in the population of each iteration. The top 50% of the particles are retained and the remaining particles are modified according to the formula (4) and (5).

$$v_{bd}^{k}=v_{gd}^{k}(1+\beta r), \quad (4)$$
$$x_{bd}^{k}=x_{gd}^{k}(1+\alpha r) \quad (5)$$

Where x_{gd}^{k} and v_{gd}^{k} are the position and velocity of the particle in the top 50% of the fitness value, x_{bd}^{k} and v_{bd}^{k} are the position and velocity of the particles in the rest of 50% worse particles which will be modified by adaptive mutation, r is a random number in the rage of[0 , 1], α and β are two given positive number.

The position and velocity of each particle is limited into a given range in the adaptive mutation particle swarm optimization . If the velocity or position of the particles is outside the given boundary value, they will be replaced by the boundary value.

Basic Flow of Adaptive Mutation Multi-Particle Swarm Algorithm

The specific process of adaptive mutation multi-particle swarm optimization is as follows:

Step1: Define the D-dimension target search space and the population size of M, divide the population of M into n sub populations, and the number of particles in each sub population is p;

Step 2: Initialize the initial position and velocity of the particles;

Step 3: Calculate the fitness value of each particle, save the local optimal position p_{id} and the corresponding position p_{gd} of the best fitness value of each sub population, then save the n p_{lgd};

Step 4: Adaptively mutate 50% worse particles v_{bd}^{k} and x_{bd}^{k} according to formula (4) and (5);

Step 5: Update velocity v^{k+1} and position x^{k+1} according to the formula (3) and (2);

Step 6: Compare the fitness value of each particle with the best position p_{id} which previously experienced and the best position P_{igd} in the sub populations, then update the global optimal position and the local optimal position;

Step 7: Repeat steps 3-6 until the termination criteria are met.

Simulation Experiment

In this paper , TSPLIB [11] provided berlin52, ch150, d198, lin105 as the test cases, the specific parameters in the experiment are as follows: population size m=40, learning factor c1 = c2 = 2, initial maximum inertia weight value $\omega_{max} = 0.9$, minimum inertia weight value $\omega_{min} = 0.35$, maximum speed $v_{max} = 1$, minimum speed $v_{min} = -1$, maximum position of $x_{max} = 10$, minimum position of $x_{min} = 0.05$, particle position variation factor α=0.0005, velocity mutation factor β =0.0005, maximum number of iterations maxgen = 100. In the simulation, the 40 particles are divided into 4 sub populations, and the number of particles in each sub population is 10. The software environment of experiment as follows: The Linux operating system of RHEL6.4, Eclipse4.3.2 , and JDK 1.6.0.Run 20 times for each example, as shown in Table 1.

Table1 The contrast of algorithm optimal solution

examples	PSO optimal solutio n	AMPSO optimal solution	AMPSO average value	TSBLIB optimal solution
berlin52	7674	7566	7584.6	7542
lin105	14655	14414	14497.4	14379
ch150	6784	6582	6616.4	6528
d198	16931	16270	16546.3	15780

As can be seen from table 1, the optimal solution obtained by AMPSO algorithm which is closer to the global optimal solution provided by TSPLIB is better than the basic PSO. Taking berlin52 which is in TSPLIB as an example, the optimal results of the basic PSO algorithm are 7674, while the results of the AMPSO algorithm are 7566, and the global optimal solution is 7542. This shows that the adaptive multi particle swarm optimization algorithm can improve the search accuracy and can give a better solution to TSP.

The relationship between path results and the number of iterations is obtained by running the two algorithms in the same parameter setting, as shown in figure 1:

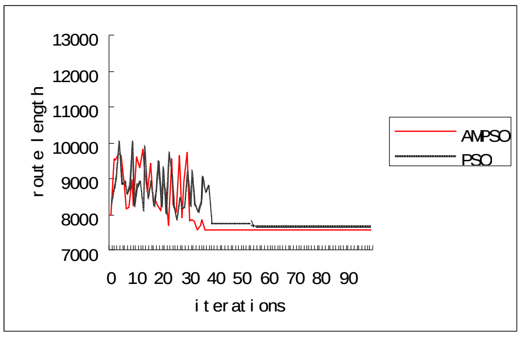

Fig.1 The optimal solution evolution of AMPSO and PSO

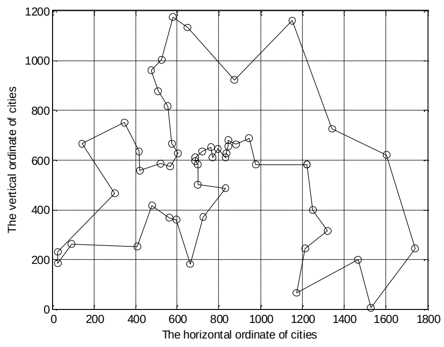

Fig.2 Berlin52 optimal solution path graph

As can be seen from Figure 1, the standard PSO algorithm begins to close the optimal value which is searched by the algorithm when the iteration is about 60th times and is trapped in local optimal solution. The adaptive mutation multi-particle swarm optimization algorithm is used to converge to the optimal value when the iteration is about 40th times. From this we can know that AMPSO not only improves the accuracy of the knowledge, but also has better searching speed, to some extent, the algorithm can improve the convergence ability of the algorithm.

Finally, the experiment takes berlin52 as an example, the optimal solution obtained by the AMPSO algorithm is used the Matlab7.0 tool to draw a path diagram, as shown in figure 2.

Conclusion

In this paper, an adaptive mutation multi-particle swarm optimization algorithm is proposed for the improvement of the standard particle swarm optimization algorithm, the algorithm sets multiple particle swarm search in the solution space independently, each particle updates the velocity and position by means of randomly adaptive mutation, then limit the velocity and position value in a reasonable range to improve the precision of the algorithm and the convergence speed.

Through we use a TSP benchmark problem with four standard TSP problem to test the validity of our approach, it can be seen that these improvements can improve the activity of particles in a certain extent and the search precision.

Acknowledgements

This research was financially supported by Chinese Natural Science Foundations (61363016, 61063004), Key Project of Inner Mongolia Advanced Science Research (NJZZ14100), Inner Mongolia Colleges and Universities Education Department Science Research (NJZC059), Natural Science Foundation of Inner Mongolia Autonomous Region of China (NO.2015MS0605, NO.2015MS0626 and NO.2015MS0627).

References

[1]Kennedy J, Eberhart R C, Particle swarm optimization, proceedings of IEEE Internationgnal Coference on the 1st European Conference on Neutral Networks, Perth, Australia.(1995) 1942- 1948.

[2] Sangwook Lee,Sangmoon Soak,Sanghoun Oh,Witold Pedrycz,Moongu Jeon,Progress in Natural Science,18(2008)1161-1166.

[3]Li Xiao-Yuan, Computer Engineering and Applications, 44(2008) 51-54, In Chinese.

[4]Shi Yuhui, Eberhart R C, Fuzzy Adaptive Particle Swarm Optimization[C]//Proc. of Congress on Evolutionary Computation. San Francisco, USA: IEEE Press, 2001.

[5]Deng Wei-Lin, Hu Gui-Wu, Computer and Modernization, 03(2012)1-4 ,In Chinese.

[6]Yu Liang-Liang, Wang,Computer Engineering, 36(2010)183-187 ,In Chinese.

[7]Wang Hua-Dong, Li Wei, Computer Simulation,29(2012) 243-246 ,In Chinese.

[8]Liu Shi-Qing, Journal of Beijing Information Science&Technology, 29(2014)46-50,In Chinese.

[9]Ye Jing, Computer Engineering, 36(2010)156-160 ,In Chinese.

[10]Liu Li-Fang, Modification and Application of Particles Swarm Optimization Algorithm, Taiyuan ，2008.

[11]TSPLIB[EB/OL]. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp