IMPROVED PARTICLE SWARM OPTIMIZATION FOR TRAVELING SALESMAN PROBLEM

Xin-Li XU, Xu CHENG, Zhong-Chen YANG, Xu-Hua YANG and Wan-Liang WANG
College of Computer Science and Technology
Zhejiang University of Technology
Hangzhou 310023, China
e-mail: xxl@zjut.edu.cn

KEYWORDS

Particle swarm algorithm, dynamic programming algorithm, scale-free fully informed network, travelling salesman problem.

ABSTRACT

To compensate for the shortcomings of existing methods used in TSP (Traveling Salesman Problem), such as the accuracy of solutions and the scale of problems, this paper proposed an improved particle swarm optimization by using a self-organizing construction mechanism and dynamic programming algorithm. Particles are connected in way of scale-free fully informed network topology map. Then dynamic programming algorithm is applied to realize the evolution and information exchange of particles. Simulation results show that the proposed method with good stability can effectively reduce the error rate and improve the solution precision while maintaining a low computational complexity.

INTRODUCTION

TSP (Traveling Salesman Problem) is a classical combinatorial optimization problem. It is hard to find the optimal solution within polynomial calculation time, as for large-scale problems people are more inclined to seek acceptable approximated optimal solution algorithm in a limited period of time. There are many ways to solve this problem, such as local search strategy, genetic algorithm, ant colony algorithm, particle swarm optimization, immune algorithm, neural network algorithm and dynamic programming algorithm. Traditional dynamic programming algorithm can obtain the optimal solution, but great time complexity and space complexity makes it only available to solve the small scale problem. Other algorithms like genetic algorithms and neural networks can obtain better solutions in a short period of time, but they have disadvantages in terms of the accuracy of solutions and the scale of problems.

To compensate for the shortcomings of existing methods used in TSP, such as the accuracy of solutions and the scale of problems, this paper proposed an improved particle swarm optimization by using a self-organizing construction mechanism and dynamic programming algorithm. Particles were connected in way of scale-free fully informed network topology map. And dynamic programming algorithm was used to realize the position updating of particles. Simulation results show that the proposed method has a good stability and it can effectively reduce the error rate besides it can improve the solution precision while maintaining a low computational complexity.

BASIC PRINCIPLES OF PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling. The basic idea of PSO is as follows: assume the population size is N, the current position of the particle can be expressed as $X_i^k = (x_1^k, x_2^k, \dots x_n^k, \dots x_N^k)$, $x_n^k \in [l_n, u_n]$, $1 \le n \le N$, l_n and u_n represent upper and lower bounds in n-th dimensional space, respectively. The current velocity can be expressed as $V_i^k = (v_1^k, v_2^k, \dots, v_N^k)$. V_i^k is between $V_{max}^k = (v^k_{max,1}, \dots v^k_{max,n}, \dots v^k_{max,N})$ and $V^k_{min} = (v^k_{min,1}, \dots v^k_{min,n}, \dots v^k_{min,N})$. The updating equation of the velocity and position of the particles are shown as equation (1) and equation (2), respectively.

$$V_i^{k+1} = wV_i^k + c_1 r_1 (P_i^k - X_i^k) + c_2 r_2 (P_g^k - X_i^k) \quad (1)$$

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

where, c_1 and c_2 are constants called the learning factor used to adjust the relative importance of the individual extremum and the global relative materiality. r_1 and r_2 are uniformly distributed random numbers in (0, 1). P_i^k and P_g^k represent the individual optimal position and the global optimal position of particles in the k-th iteration, respectively.

PARTICLE SWARM OPTIMIZATION ALGORITHM COMBINING WITH DYNAMIC PROGRAMMING METHOD

For traveling salesman problem, the state of particle (X) is represented as N-dimensional vector just like other particle swarm optimization algorithms, and the number of dimensions denotes the total number of cities. We can use the formula (3) to represent the state X.

$$X = (x_1, x_2, \dots x_i, \dots x_N),\ 1 \le i \le N,\ 1 \le x_i \le N \quad (3)$$

In this formula, N represents the number of the cities while x_i represents the corresponding city number. Start

Proceedings 27th European Conference on Modelling and Simulation ©ECMS Webjørn Rekdalsbakken, Robin T. Bye, Houxiang Zhang (Editors)
ISBN: 978-0-9564944-6-7 / ISBN: 978-0-9564944-7-4 (CD)

from the first city x_1, and visit x_{i+1} after x_i, until x_N is visited then revisit x_1. Finally, form the overall access sequence. In the initial phase of the algorithm, M different particles are randomly generated to compose the particle swarm.

Based on the updating equation of velocity and position of the particle are shown as equation (1) and (2), we can see new position of the particle depends on three parts: a) velocity and position of the particle in the previous iteration, that is, the previous state of the particle; b) cognition part, that is, self-learning of the particle; c) social part, that is, the collaboration among particles. In a word, new position of the particle is interdependence of previous state and best self-experience of the particle and best experience the population. So we can define the position updating formula for discrete particle swarm algorithm as follows:

$$X_i^{k+1}=c_2\otimes f(c_1\otimes g(w\otimes h(X_i^k),\ P_i^k),\ P_g^k) \qquad (4)$$

where w is the inertia weight, c_1 is the cognitive coefficient, and c_2 is social coefficient. In generally, w, c_1, $c_2 \in [0,1]$.

The formula (4) consists of three parts:

$$E_i^k=w\otimes h(X_i^k)=\begin{cases} h(X_i^k) & \text{rand}<w \\ X_i^k & \text{otherwise} \end{cases} \qquad (5)$$

The equation (5) represents the influence of previous state of the particle, where rand is the random number in [0, 1]. $h(X_i^k)$ indicates the velocity of the particles and its adjustment is as seen in section velocity updating operator combined with dynamic programming.

$$G_i^k=c_1\otimes g(E_i^k,P_i^k)=\begin{cases} s(E_i^k,P_i^k) & \text{rand}<c_1 \\ E_i^k & \text{otherwise} \end{cases} \qquad (6)$$

$g(E_i^k,P_i^k)$ represents that the particle is adjusted according to its optimal position P_i^k. For $s(E_i^k,P_i^k)$,we can see in the section of crossover operator with interparticle information interaction.

$$X_i^k=c_2\otimes s(G_i^k,P_g^k)=\begin{cases} s(G_i^k,P_g^k) & \text{rand}<c_2 \\ G_i^k & \text{otherwise} \end{cases} \qquad (7)$$

The equation (7) denotes the particle is adjusted based on the global optimal position P_g^k. The operation process of $s(G_i^k,P_g^k)$ is the same to $s(F_i^k,P_i^k)$.

Self-organizing construction mechanism with scale-free network graph

In 1999, Barabási and Albert in the process of research the dynamic evolution of the World Wide Web found that many large-scale complex networks have high ability about self-organization, and the node degrees of most complex network follow a "power law distribution". The probability of k connections of any node is in proportion to 1/k, namely p(k) ~ k-γ, where 2 <γ < 4. The network with the characteristic is called scale-free network.

The construction method of scale-free network topology graph is used as a self-organizing construction mechanism in this paper, that is, all particles are combined into a particle swarm in the form of scale-free network topology graph. The topology construction steps can be briefly described as follows. Firstly, we select m most accurate particles from M original particles. Then m particles are connected to form a complete graph, that is, any two particles have an edge connected. Lastly, insert a particle of the M-m remaining particles into the network every time according to a degree preferential attachment scheme. Specifically, for a newly added particle i and an arbitrary existing particle j in the network, there is a probability $P_i=k_j/\Sigma k_i$ to connect the two particles, where k_j denotes the degrees of the particle j.

Based on the above steps, we can adaptively generate the population topology exhibiting scale-free property, that is, the topology will be gradually generated as the construction process and the optimization process progress synchronously.

Velocity updating operator combined with dynamic programming

Velocity updating operator includes TSP sequence splitting and integrating with dynamic programming. TSP often involves hundreds or even tens of thousands of cities. If we carry out dynamic programming to solve TSP in the global situation, the over-scaled problems will consume excessive time, and even the optimal solution will not be found. To solve the above problem, we consider the method from the local situation and split the whole TSP sequence into a number of sub-sequences. In order to ensure more reasonable sub-sequences split, five different strategies of setting truncation places are considered as follow, and thus a TSP sequence including n cities is spitted into $m(1\leq m\leq n)$ subsequences.

(A) Randomly select m different truncation places in the TSP sequence.
(B) Randomly select a city in TSP sequence, and then set a truncation place after m subsequent cities.
(C) Select a random city in TSP sequence, find its m-1 closest cities, and then set a truncation place after the m cities found.
(D) Select the "oldest" truncation place not to be split, and then set a truncation place in the rear of the m/2 cities after it and the other truncation place in the rear of the m/2 cities before it.
(E) Select a city in front of the "oldest" truncation place not to be split, find its m-1 closest cities, and then

set a truncation place in the rear of the m cities found.

To illustrate the performance of different strategies, rat_783 (TSP case with city size of 783) in TSPLIB is selected to compare with different strategies. The simulation results in table 1 show that the error rate of randomly selecting strategy such as A, B, C, D and E in any one splitting is the smallest, and its CPU times spent is less, which is more than strategy B and D. Hence, randomly selecting strategy such as A, B, C, D and E is used as TSP sequence splitting in this paper.

Table 1: Results based on Different Strategies for Rat_783

Strategies	Error rate(%)	CPU time(s)
A	5.75	133.2
B	24.75	1.89
C	14.73	196.8
D	24.35	1.961
E	13.51	142.7
A, B, C, D, E	3.62	131.3

TSP sequence is divided once, and the problem is simplified how to connect those subsequences into the shortest circuit. At the same time, the sequence direction will be involved in when we integrate those subsequences result from the local splitting. Based on the traditional dynamic programming algorithm for TSP, this paper introduces a new variable c_i indicating the direction of the i-th subsequence. If the direction of the i-th subsequence is positive, then set $c_i=1$. Otherwise, set $c_i=0$.

As TSP path is a cyclic path, we can assume that the last access subsequence is the 0-th subsequence whose direction is positive. And thus a new state variable (i,c_i,k) can be obtained, which indicates that the current spot is the i-th subsequence whose direction is c_i. Where k represents several subsequences accessed from the 0-th subsequence to the i-th subsequence in positive direction. In the corresponding binary, if the j-th bit is 1, it means the j-th subsequence has been selected in the access path while 0 means not.

Here, function $f(i, c_i ,k)$ is defined as the shortest distance in a certain state (i, c_i ,k). Then we can get the function value of initial state, that is, $f(0,0,0)=0$, and the function value of target state is $f(0,0,2^m-1)$. Hence the transfer equation can be obtained as follow.

$$f(i,c_i,k) = \min_{i\neq j,k/2^j\%2=1} (f(j,c_j,k-2^i)+dist(P(i,c_i),P(j,1-c_j))) \tag{8}$$

In the equation (8), if $c_i=0$, then $P(i,c_i)$ represents the first city in the i-th subsequence. Otherwise, if $c_i=1$, then $P(i,c_i)$ represents the last city in the i-th subsequence. $dist(P(i,c_i), P(j,1-c_j))$ represents the distance between the i-th subsequence and the j-th subsequence. $k/2^j\%2=1$ means that the j-th bit of the binary of k is equal to 1, that is, the j-th subsequence has been selected in the previous path walked.

According to the equation (8), the optimal value of $f(0,0,2^m-1)$ can be obtained, and all subsequences can be recombined into a whole TSP sequence including n cities in accordance with the optimal path recorded in the solution.

Crossover operator with interparticle information interaction

We adjust the state of the particles according to their own best position P_i^k and the global best position P_g^k by using crossover operator with interparticle information interaction in the paper, which can be described as follows.

1. We assume that P_i^k or P_g^k is a certain particle Y and E_i^k or G_i^k is a certain particle X.
2. Extract the continuous sub-vector from particle Y, such as $(y_{i+1},y_{i+2},\ldots,y_{i+l})$ called as the vector Z, whose length is l (l is a random positive number less than 10). Each element of the vector Z can be regarded as a breakpoint used to split the vector X, so the vector X can be divided into l segments according to the breakpoints and every segment of the vector X does not contain any element of the vector Z.
   For example, there are two known particles, that is, X is (1,2,3,4,5,6,7,8,9) and Y is (4,3,2,1,4,6,5,7,8,9). We can extract a length-3 sub vector Z from Y, which is (1, 4, 6). And then X can be divided into three sections according to the vector Z, such as (2, 3), (5) and (7, 8, 9).
   Based on the above analysis, we can get l sub-vectors and a sub-vector extracted from the vector Y, so a total number of sub-vectors is l+1.
3. We can use the above dynamic programming method as seen in section B to reassemble into a new particle X'. If X' is superior to the original particle X, X will be replaced with X'.

Procedure of algorithm for TSP

The procedures of improved particle swarm algorithm to solve TSP are as follows:

Step1. Set the population size of particle swarm (M), the max number of iterations (ds), and the parameters (w, $c_1$and c_2). Then randomly generate M different particles, calculate fitness value of each particle X_i, and record the initial local optimal value $P_i=X_i$, $i\in\{1,2,\ldots, M\}$.

Step2. Set t=0, construct an initial scale-free network graph construction with m different particles, and record the global optimal particle $P_g=\min(P_j)$, $j\in\{1,2,\ldots, m\}$.

Step3. Update the position of each particle according to the formula (3), calculate their fitness value, and then update the local optimal value P_j and global optimal value P_g=min(P_j), $j \in \{1,2,\ldots, \min(m+t, M)\}$.

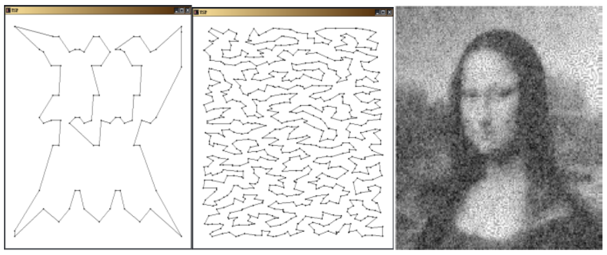
Step4. Gradually generate the population topology. At each time step t there is exactly one particle of the remainder M-m-t particles that will be connected with m+t existing particles.

Step5. Add 1 to the time step t. If t > ds, then output the optimal solution and quit. Otherwise, go to Step3.

SIMULATION RESULTS AND ANALYSIS

Effectiveness test of proposed algorithm

In order to validate the performance of the proposed algorithm, numerical simulation experiments of three different-scaled TSP examples are conducted, where the number of cities is 76, 783 and 100000, that is, small scale, medium scale and large scale, respectively. The proposed algorithm has been coded with VC++6.0, runs on a PC with Intel Pentium CPU 2.20GHz processor and 2G of memory and the simulation results are as shown in Figure1.



(a) 76 cities (b) 783 cities (c) 100000 cities

Figure 1: TSP with Different Number of Cities

As shown in Figure 1, the improved particle swarm algorithm can obtain good results of any scale TSP, which demonstrates the effective of the proposed algorithm.

In order to illustrate effectiveness of the proposed algorithm, we do further simulation on the rat_783 and extract the results of first 100 iterations and 100 to 300 iterations as shown in Figure 2 and Figure 3.

From Figure 2, we can see that the proposed algorithm has strong convergence at the early stage and an excellent solution can be quickly obtained. Moreover, the proposed algorithm does not lead to "premature" and it is still slowly evolving as shown in Figure 3.

Finally the proposed algorithm is applied to solve 14 different-scale TSP examples from 76 to 85900 in TSPLIB, and simulation results are compared with the optimal path lengths published in TSBLIB. "Error rate" deviating from the optimal solution is used to evaluate quality of the solution, which is the percentage ratio of the deviation between average path length obtained by the proposed algorithm and the optimal path length released in TSPLIB, that is, Error rate=(Average solution – Optimal solution)/Optimal solution×100%. So the smaller the error rate is, the better the quality of the solution will be. Considering path lengths released in TSPLIB are integers while the data aren't rounded in many references, path lengths are not rounded to facilitate comparison with similar methods in this paper, too. The simulation results of 20 independent calculations are shown in the following table 2. If the error rate in table 2 is less than 0, we can know that the solution value is smaller than the optimal solution in TSBLIB, which shows the superiority of the proposed algorithm.

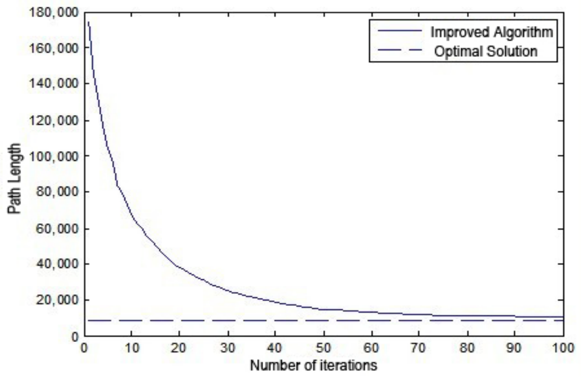


Figure 2: Convergent Graph of the First 100 Iterations

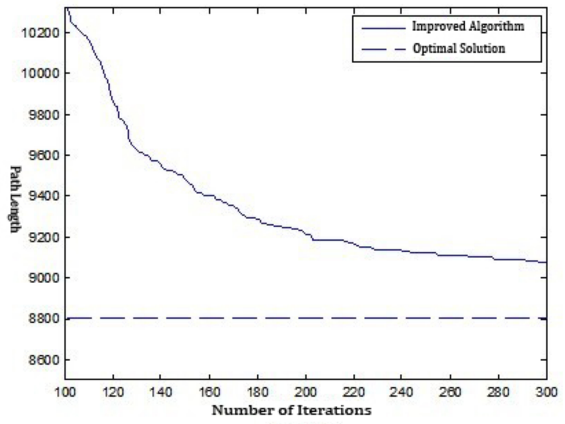


Figure 3: Convergent Graph of from 100 to 300 Iterations

We can find when the scale of problem is small, about 100 or so, the optimal solutions obtained by the improved particle swarm algorithm are extremely approximated to or even smaller than that of TSBLIB, which means the improved algorithm can almost find the optimal solution of the small scaled TSP. Error rate increases as the problem scale increases while remaining in a small scope, which shows good convergence of the algorithm. As is shown in Table 2, the improved algorithm is suitable for good solution of any scale TSP.

Table 2: Results of Different Numbers of Cities

Number	Cases	Number of cities	Optimal solution	Best solution	Average solution	Error rate (%)	Average CPU time (s)
1	pr_76	76	108159	108159.2	108159.4	0.00	1.53
2	kroA_100	100	21282	21282.44	21286.64	0.01	2.97
3	kroB_100	100	22141	22138.26	22139.07	-0.01	1.08
4	pr_107	107	44303	44301.68	44302.88	0.00	2.70
5	pr_144	144	58537	58536.44	58537.87	0.00	5.63
6	lin_318	318	42029	42632.13	42633.17	1.44	24.63
7	pcb_442	442	50778	51526.46	51528.35	1.48	27.29
8	rat_783	783	8806	9124.82	9124.91	3.62	300.55
9	pcb_1173	1173	56892	58335.36	58339.30	2.54	184.63
10	D_1655	1655	62128	64124.65	64126.35	3.22	341.23
11	pcb_3038	3038	137694	142403.53	142404.40	3.42	672.24
12	rl_5934	5934	556045	575345.36	575395.78	3.48	869.11
13	pla_33810	33810	66048945	68208524.23	69408534.49	5.09	2939
14	pla_85900	85900	142382641	154827063.35	154927063.95	8.81	8226

Simulation results Comparison with different algorithms

Simulation results of the improved particle swarm algorithm (IPSO), generalized chromosome genetic algorithm (GCGA), constructive-optimizer neural network algorithm (CONN) and improved elastic net algorithm (Improved-EN) are compared with those of corresponding cases in terms of error rate in Table 3.

An observation of Table 3 demonstrates that the improved algorithm is obviously as good as or better than other algorithms in terms of error rate, namely, the proposed algorithm is more accurate than other algorithms.

Table 3: Comparisons among Error Rate of Different Algorithms

Number	IPSO	GCGA	CONN	Improved-EN
1	0	0.72	4.34	--
2	0.02	1.23	2.57	2.09
3	-0.01	1.81	2.6	3.02
4	0	1.37	2.77	--
5	0	0.04	2.34	--
6	0.73	5.14	--	8.02
7	1.18	8.99	5.72	--
8	1.76	--	7.59	12.73
9	2.13	--	8.9	7.12
10	3.13	--	8.28	--
11	2.04	--	8.04	--
12	2.78	--	13.1	--
13	4.36	--	--	--
14	4.93	--	--	--

CONCLUSION

In this paper, a self-organizing construction mechanism is used to generate the population topology and dynamic programming method is used to update the position of particles in the optimization process progress, which makes the improved particle swarm algorithm with good fusion of the local optimum and the global optimum. Simulation results show that the proposed algorithm surpasses other methods in the accuracy of solution, such as generalized chromosome genetic algorithm (GCGA), constructive-optimizer neural network algorithm (CONN) and improved elasticity network algorithm (improved-EN). Unlike some existing algorithms which can commonly solve TSP with small scale (less than 100), the proposed algorithm can be applied to TSP with various scale. Simulation experiments of 14 TSP with the scale ranging from 76 to 85900 have verified that. In addition, the improved algorithm can be used widely, which can solve TSP in non-planar state, such as 3-dimensional space and spherical space.

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation of China (61070043, 61105073, and 61203371), and a sub-project of National Science and Technology Support Plan (2012BAD10B0101), and the Open Fund Project of Zhejiang University of Technology (20120814).

REFERENCES

Barabási, A-L. and Albert, R. “Emergence of scaling in random network”, Science, vol .286, pp.509-512, 1999.

Kennedy, J. and Eberhart, R. “Particle swarm optimization”, IEEE Int. Conf. on Neural Networks, Perth, Australia, pp.942-1948,1995.

Chen, S.Y., "Kalman Filter for Robot Vision: a Survey", IEEE Transactions on Industrial Electronics, Vol. 59, No.11, 2012, pp. 4409 - 4420.

Chen, S.Y. and Li, Y.F., "Automatic Sensor Placement for Model-Based Robot Vision", IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 34, No. 1, pp. 393-408, Feb 2004.

Eberhart, R. and Kennedy, J. “A new optimizer using particle swarm theory”, Proc of the Sixth International Symposium on Micro Machine and Human Science Conf. Nagoya, Japan, pp.39-43,1995.

Gao, S. Han, B. Wu, X.-J. and Yang, J.-Y. “Solving traveling salesman problem by hybrid particle swarm optimization algorithm”, Control and Decision, vol. 19, no.11, pp.1286-1289, 2004.

Xu, G. Segawa, E. and Tsuji, S. “Robust active contours with insensitive parameters”, Pattern Recognition, vol. 27, No.7, pp.879-884, 1994.

Hall, D. and Gal, L-C. “magicboard:A contribution to an intelligent office environment”, Robotics and Antonomous Systems, pp.211-220, 1999.

Su, J-R. and WANG, J-Z. “Improved particle swarm optimization for traveling salesman problem”, Computer Engineer and Applications, vol.46, no.4, pp.52-75, 2010.

Reinelt, G. “TSPLIB-A traveling salesman problem library”, ORSA Journal on Computing, pp.376-384, 1991.

Wen, S., et al., "Elman Fuzzy Adaptive Control for Obstacle Avoidance of Mobile Robots using Hybrid Force/Position Incorporation", IEEE Transactions on Systems, Man, and Cybernetics, Part C, Vol. 42, No. 4, 2012, pp. 603-608.

Yang, J. Wu, C. Lee, H.P. and Liang, Y. “Solving traveling salesman problem using generalized chromosome genetic algorithm”, Progress in Natural Science Conf. , vol.18, pp.887-892, 2008.

Saadatmand-Tarzjan, M. Khademi, M. Akbarzadeh-T, M. R. and Mghaddam, H. A. “A novel constructive-optimizer neural network for the traveling salesman problem”, IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol .37 (4), pp.754-770, 2007.

Yi, J. Yang, G. Zhang Z. and Tang,Z. “An improved elastic net method for traveling salesman problem”, Neurocomputing, vol.72, pp. 1329-1335, 2009.

Zhang, C-G. and Yi, Z. “Scale-free fully informed particle swarm optimization algorithm”, Information Sciences, vol.181, pp.4550-4568, 2011.

Zheng, Y., Chen, S., Ling, H., "Agent-based Cooperative Evolutionary Computation for Disaster Rescue Operation Planning", Disaster Advances, Vol. 5, No. 4, 2012, pp. 698-703.

Yao, C-Z. and Yang, J-M. “PSO Algorithm Based on Network Neighborhood Topology”, Computer Engineering, vol.36, no.19, pp.18-23, 2010.

AUTHOR BIOGRAPHIES

Xinli XU received the B.S. degree in automation from China University of Mining and Technology, China, in 2000, the M.Sc. and Ph.D. degrees in control theory and control engineering from Zhejiang University of Technology (ZJUT), China, in 2003 and 2009, respectively. Since 2010, she has been an Associate Professor in the College of Computer science and technology, ZJUT.

Her current research interests include optimization theory and algorithms in production scheduling and hydropower scheduling, and complex networks theory and its applications.