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 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.

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 \(X^k = (x_1^k, x_2^k, \ldots, x_N^k)\), where \(x_i^k \in [l_i, u_i]\), and \(u_i\) represent upper and lower bounds in n-th dimensional space, respectively. The current velocity can be expressed as \(V^k = (v_1^k, v_2^k, \ldots, v_N^k)\), \(V^k\) is between \(V_{\text{max}} = (v_{\text{max},1}, \ldots, v_{\text{max},n}, \ldots, v_{\text{max},N})\) and \(V_{\text{min}} = (v_{\text{min},1}, \ldots, v_{\text{min},n}, \ldots, v_{\text{min},N})\). The updating equation of the velocity and position of the particles are shown as equation (1) and equation (2), respectively.

\[
\begin{align*}
V_{i}^{k+1} &= wV_{i}^{k} + c_{1}r_{1}(P_{i}^{k} - X_{i}^{k}) + c_{2}r_{2}(P_{g}^{k} - X_{i}^{k}) \quad (1) \\
X_{i}^{k+1} &= X_{i}^{k} + V_{i}^{k+1} \quad (2)
\end{align*}
\]

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.

\[
X = (x_1, x_2, \ldots, x_N), 1 \leq i \leq N, 1 \leq x_i \leq N \quad (3)
\]

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

$$X_{i}^{k+1} = c_2 \oplus f(c_1 \oplus g(w \oplus h(X_i^k), P_i^k), P_g^k)$$

(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:

$$E_i^k = w \oplus h(X_i^k) \begin{cases} h(X_i^k) \text{ rand} < w \quad \text{or otherwise} \end{cases}$$

(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.

$$G_i^k = c_2 \oplus g(E_i^k,P_i^k) \begin{cases} s(E_i^k,P_i^k) \text{ rand} < c_2 \quad \text{or otherwise} \end{cases}$$

(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.

$$X_i^{k} = c_2 \oplus s(G_i^k,P_g^k) \begin{cases} s(G_i^k,P_g^k) \text{ rand} < c_2 \quad \text{or otherwise} \end{cases}$$

(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 $p(k) \sim k^{-\gamma}$, where $2 < \gamma < 4$. 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 $P_i = k_i / 2k$ to connect the two particles, where $k_i$ 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 $m(1 \leq m \leq n)$ sub-sequences.

(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 $m$ 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
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

<table>
<thead>
<tr>
<th>Strategies</th>
<th>Error rate(%)</th>
<th>CPU time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5.75</td>
<td>133.2</td>
</tr>
<tr>
<td>B</td>
<td>24.75</td>
<td>1.89</td>
</tr>
<tr>
<td>C</td>
<td>14.73</td>
<td>196.8</td>
</tr>
<tr>
<td>D</td>
<td>24.35</td>
<td>1.961</td>
</tr>
<tr>
<td>E</td>
<td>13.51</td>
<td>142.7</td>
</tr>
<tr>
<td>A, B, C, D, E</td>
<td>3.62</td>
<td>131.3</td>
</tr>
</tbody>
</table>

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 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 \( Z \).
3. For example, there are two known particles, that is, \( X = (1, 2, 3, 4, 5, 6, 7, 8, 9) \) and \( Y = (4, 3, 2, 1, 4, 6, 5, 7, 8, 9) \).

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 \( l+1 \).

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 $P_g = \min(P_j), j \in \{1, 2, \ldots, \min(m+t, M)\}$.

Step 4. 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.

Step 5. Add 1 to the time step $t$. If $t > ds$, then output the optimal solution and quit. Otherwise, go to Step 3.

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 Figure 1.

![Figure 1: TSP with Different Number of Cities](image)

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

![Figure 2: Convergent Graph of the First 100 Iterations](image)

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 TSPLIB. "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, \( \text{Error rate} = \frac{\text{Average solution} - \text{Optimal solution}}{\text{Optimal solution}} \times 100\% \). 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 TSPLIB, which shows the superiority of the proposed algorithm.

![Figure 3: Convergent Graph of from 100 to 300 Iterations](image)

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 TSPLIB, 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.
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

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

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 (2012BAD1OB0101), and the Open Fund Project of Zhejiang University of Technology (20120814).

REFERENCES


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.