We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

154
Countries delivered to

Our authors are among the
TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™
Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

The Advantage of Intelligent Algorithms for TSP

Yuan-Bin MO
College of Mathematics and Computer Science, Guangxi University for Nationalities, China

1. Introduction

Traveling salesman problem (TSP) means that a travelling salesman needs to promote products in n cities (including the city where he lives). After visiting each city (each city can be visited once), he returns to the departure city. Let's suppose that there is one road to connect each two cities. What is the best route to follow in order to minimize the distance of the journey?
TSP has been proven to be a NP-hard problem, i.e. failure of finding a polynomial time algorithm to get a optimal solution. TSP is easy to interpret, yet hard to solve. This problem has aroused many scholars' interests since it was put forward in 1932. However, until now, no effective solution has been found.
Though TSP only represents a problem of the shortest ring road, in actual life, many physical problems are found to be the TSP. Example 1, postal route. Postal route problem is a TSP. Suppose that a mail car needs to collect mails in n places. Under such circumstances, you can show the route through a drawing containing n+1 crunodes. One crunode means a post office which this mail car departures from and returns to. The remaining n crunodes mean the crunodes at which the mails need to be collected. The route that the mail car passes through is a travelling route. We hope to find a travelling route with the shortest length. Example 2, mechanical arm. When a mechanical arm is used to fasten the nuts for the ready-to-assembling parts on the assembly line, this mechanical arm will move from the initial position (position where the first nut needs to be fastened) to each nut in proper order and then return to the initial position. The route which the mechanical arm follows is a travelling route in the drawing which contains crunodes as nuts; the most economical travelling route will enable the mechanical arm to finish its work within the shortest time. Example 3, integrated circuit. In the course of manufacturing the integrated circuit, we often need to insert thousands of electrical elements. It will consume certain energy when moving from one electrical element to the other during manufacturing. How can we do to arrange the manufacturing order to minimum the energy consumption? This is obviously a solution for TSP. Except for the above examples, problems like route distribution of transportation network, choice of tourist route, laying of pipelines needed for city planning and engineering construction are interlinked with the problems of finding the shortest route. So, it is of significance to make a study on the problem of the shortest route. This renders us a use value.
As finding a solution for TSP plays an important role in the real life, since the TSP appeared, it has attracted many scholars to make a study on it.

2. Mathematical description for the TSP and its general solving method

2.1 Mathematical description for the TSP

According to the definition of the TSP, its mathematical description is as follows:

$$\min \sum d_{ij} x_{ij} \tag{2.1.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,2,\cdots n \tag{2.1.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,2,\cdots n \tag{2.1.3}$$

$$\sum_{i,j \in S} x_{ij} \le |S| - 1 \quad 2 \le |S| \le n-2, S \subset \{1,2,\cdots n\} \tag{2.1.4}$$

$$x_{ij} \in \{0,1\} \quad i,j = 1,2,\cdots n \quad i \ne j \tag{2.1.5}$$

Where d_{ij} means the distance between the city i and city j; decision variable $x_{ij} = 1$ means the route the salesman passes through (including the route from city i and city j); $x_{ij} = 0$ means the route which isn't chosen by the salesman. Objective function (2.1.1) means the minimum total distance; (2.1.2) means that a salesman only can departure from the city i for one time; (2.1.3) means that a salesman only can enter the city j for one time; (2.1.2) and (2.1.3) only give an assurance that the salesman visits each city once, but it doesn't rule out the possibility of any loop; (2.1.4) requires that no loop in any city subset should be formed by the salesman ; $|S|$ means the number of elements included in the set S .

2.2 Traditional solving method for TSP

At present, the solving methods for TSP are mainly divided into two parts: traditional method and evolution method. In terms of traditional method, there are precise algorithm and approximate algorithm.

2.2.1 Precise algorithm for solving the TSP

Linear programming

This is a TSP solving method that is put forward at the earliest stage. It mainly applies to the cutting plane method in the integer programming, i.e. solving the LP formed by two constraints in the model and then seeking the cutting plane by adding inequality constraint to gradually converge at an optimal solution.

When people apply this method to find a cutting plane, they often depend on experience. So this method is seldom deemed as a general method.

Dynamic programming

S is the subset of the set $\{2,3,\cdots n\}$. $k \in S$ and $C(S,k)$ means the optimal travelling route (setting out from 1, passing through the points in S and ending to k). When $|S| = 1$, $C\{\{k\},k\} = d_{1k}$ and $(k = 2,3,\cdots n)$. When $|S| > 1$, according to the optimality principle, the

dynamic programming equation of TSP can be written as $C(S,k) = \min_{j \in S-\{k\}} [C(S-\{j,k\}, j) + d_{jk}]$ and the solution can be obtained by the iterative method based on dynamic programming. As the time resource (i.e. time complexity) needed for dynamic programming is $O(n^2 \cdot 2^n)$, and its needed space resource (i.e. space complexity) is $O(n \cdot 2^n)$, when n is added to a certain point, these complexities will increase sharply. As a result, except for the minor problem, this is seldom used.
Branch-bound algorithm
Branch-bound algorithm is a search algorithm widely used by people. It controls the searching process through effective restrictive boundary so that it can search for the optimal solution branch from the space state tree to find an optimal solution as soon as possible. The key point of this algorithm is the choice of the restrictive boundary. Different restrictive boundaries may form different branch-bound algorithms.
Branch-bound algorithm is not good for solving the large-scale problem.

2.2.2 Approximate algorithm for solving the TSP

As the application of precise algorithm to solve problem is very limited, we often use approximate algorithm or heuristic algorithm. The result of the algorithm can be assessed by $C / C^* \le \varepsilon$. C is the total travelling distance generated from approximate algorithm; C^* is the optimal travelling distance; ε is the upper limit for the ratio of the total travelling distance of approximate solution to optimal solution under the worst condition. The value of ε >1.0. The more it closes to 1.0, the better the algorithm is. These algorithms include:
Interpolation algorithm
Interpolation algorithm can be divided into several parts according to different interpolation criteria. Generally it includes following steps:

Step 1. Choose the insertion edge (i and j) and insertion point k through a certain way. Insert k into i and j to form $\{\cdots, i, k, j, \cdots\}$;

Step 2. Follow the process in an orderly manner to form a loop solution.

Interpolation algorithm mainly includes:

1. Latest interpolation effect $\varepsilon = 2$. Time complexity: $O(n^2)$.
2. Minimum interpolation effect $\varepsilon = 2$. Time complexity: $O(n^2 \lg n)$.
3. Arbitrary interpolation effect $\varepsilon = 21 \mathrm{g}\, n + 0.16$. Time complexity: $O(n^2)$.
4. Farthest interpolation effect $\varepsilon = 2 \lg n + 0.16$. Time complexity: $O(n^2)$.
5. Convex interpolation effect ε (unknown). Time complexity: $O(n^2 \lg n)$.

Nearest-neighbour algorithm

Step 1. Choose one departure point randomly;

Step 2. Choose the nearest point in an orderly manner to add to the current solution until the loop solution is formed.

Effect: $\varepsilon = (\lg n + 1)/2$. Time complexity: $O(n^2)$
Clark & Wright algorithm

Step 1. Choose one departure point P randomly to calculate $s_{ij} = d_{pi} + d_{pj} + d_{ij}$;

Step 2. Array s_{ij} in ascending order;

Step 3. Connect each (i, j) in an orderly manner upon arrangement to form a loop solution.

Effect: $\varepsilon = 2\lg n/7 + 2/21$. Time complexity: $O(n^2)$
Double spanning tree algorithm
Step 1. First determine the minimum spanning tree.
Step 2. Determine the Euler loop by adding a repetitive edge to each edge of the tree;
Step 3. Eliminate the repetitive point in the sequence of Euler loop point to form a loop solution.
Effect: $\varepsilon = 2$. Time complexity: $O(n^2)$
Christofides algorithm
Step 1. First determine the minimum spanning tree;
Step 2. Solve the minimum weight matching problem to all the singular vertexes of the tree;
Step 3. Add the matching edge to the spanning tree to determine its Euler loop;
Step 4. Eliminate the repetitive point in the sequence of Euler loop point to form a loop solution.
Effect: $\varepsilon = 2/3$. Time complexity: $O(n^3)$
$r - opt$ algorithm
This algorithm is a locally improved search algorithm and is put forward by Lin and other people (1965). Its thought is to improve the current solution by exchanging r edges each time according to the given initial loop. As for different r , we find from massive calculation that $3 - opt$ is better than $2 - opt$, and $4 - opt$ and $5 - opt$ are not better than $3 - opt$. The higher the r is, the more time the calculation will take. So we often use $3 - opt$.
Effect: $\varepsilon = 2$ $(n \geq 8, r \leq n/4)$. Time complexity: $O(n^r)$
Hybrid algorithm
Use a certain approximate algorithm to find an initial solution and then improve the solution by using one or several algorithms of $r - opt$ **.**Usually, Hybrid algorithm will help you to get better solution, but it takes a long time.
Probabilistic algorithm
Based on the given $\varepsilon > 0$, this algorithm is often used to solve the TSP within the range of $1 + \varepsilon$ .Suppose that G is in the unit square and function $t(n)$ is mapped to the positive ration number and satisfies the following two conditions: (1) $t \to \log_2 \log_2 n$; (2) to all n , n/t is the perfect square, so the steps are as follows:
Step 1. Form the network by using $[t(n)/n]^{1/2}$ as size. Divide the unit square into $n/t(n)$ and G into several $n/t(n)$ subgraphs;
Step 2. Use dynamic programming to find the optimal loop for each subgraph;
Step 3. Contract $n/t(n)$ subgraph into one point. The distance definition is the shortest distance of the optimal sub-loop of the original subgraph. In addition, determine the minimum generation number T of the new graph;
Step 4. See T $\cup$ {the optimal sub-loop of each optimal sub-loop of } as the close loop with repetitive point and edge. According to the condition of the triangle inequality, reduce the repetitive points and edges to find a TSP loop.
Effect: $\varepsilon = 1+$ (give the positive number randomly). Time complexity: $O(n \lg n)$.
As these traditional algorithms are local search algorithms, they only help to find a local optimal solution when used for solving the TSP. It is hard to reach a global optimal solution and solve large-scale problem. So, people started to look for an evolution algorithm to solve the TSP.

3. Evolution algorithm for solving the TSP

As stated above, the traditional algorithms used to solve the TSP have some limitation. With the development of evolution algorithm, many numerical optimization algorithms appear. They are ACA, GA, SA, TS, PSO and IA, etc. These algorithms are, to some extent, random search algorithms. ACA and PSO are typical parallel algorithms. Though they cannot guarantee to help you to obtain an optimal solution within the limited time, they can give you a satisfactory solution within the affordable time range. To figure out the effect of the solution for TSP obtained by using optimization algorithm, we should consider the algorithm's search ability. Algorithm with strong optimization will produce better effect. Algorithm which is easy to trap in local extremum often helps you to obtain the local optimal solution for TSP.

3.1 Ant colony algorithm for solving the TSP

Ant colony algorithm (ACA) is a relatively new analogy evolution algorithm, which was put forward by scholars such as Italian scholar Dorigo. They called it ant colony system and used this ant colony to solve the TSP, achieving fairly good experimental result. As for ACA, n represents the number of cities for the TSP; m represents the number of ant in the ant colony; d_{ij} $(i,j=1,2,\cdots,n)$ represents the distance between city i and city j; $\tau_{ij}(t)$ represents the concentration of pheromone on the line of city i and city j at the time of t.At the initial time, the concentration of pheromone on each route is similar to one another. When $\tau_{ij}(0)=C$, C is a constant. During the moving process, ant $(k=1,2,\cdots,m)$ will determine which direction it will change according to the concentration of pheromone on each route. $P_{ij}^k(t)$ represents the probability for ant to move from city i to city j at the time of t. Its formula is

$$P_{ij}^k(t)=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum\limits_{s\in \text{allowed}_k}\tau_{is}^{\alpha}(t)\eta_{is}^{\beta}(t)} & j\notin \text{tabu}_k \\ 0 & other\end{cases} \tag{3.1.1}$$

Wherein: tabu_k $(k=1,2,\cdots,m)$ means that ant k has passed through the set of the city. From the beginning, tabu_k has only one element, i.e. the departure city of ant k. With the process of evolution, the elements for tabu_k increase continuously; $\text{allowed}_k=\{1,2,\cdots,n\}-\text{tabu}_k$ means the next city that ant k is allowed to choose. η_{ij} represents the visibility, and is taken from the reciprocal of the length of the route (i,j); α,β regulates the relatively important degree of pheromone concentration τ and visibility η.

As time goes by, the pheromone on each route gradually disappears. Parameter $1-\rho$ is used to represent the volatility of pheromone. After ω time, the ants complete one circle. Pheromone concentration on each route can be adjusted according to the following formula:

$$\tau_{ij}(t+\omega)=\rho\cdot\tau_{ij}(t)+\Delta\tau_{ij} \qquad \rho\in(0,1) \tag{3.1.2}$$

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^k \tag{3.1.3}$$

Wherein: $\Delta\tau_{ij}^{k}$ means the pheromone concentration left on the route (i,j) by the k ants during the process of this circle; $\Delta\tau_{ij}$ means the total pheromone concentration released on the route (i,j) by all the ants during the process of this circle.

ACA not only uses the positive feedback principle which may, to some extent, quicken the evolution process, but also is a parallel algorithm in nature. The ongoing process of information exchange and communication between individuals helps to find a better solution. It is easy to converge at a local extremum when there is only one individual. However, through cooperation, multiple individuals will help us to get a certain subset of the solution space, which provide a better environment for us to carry out a further exploration on solution space. The movement of multiple individuals in the ant colony is random. Actually, the measures taken to avoid the possibility of appearance of local extremum slow down the velocity of convergence. When the scale of ant colony expands, it will take a longer time to look for a better route.

In the light of the above problems, many scholars at home and abroad make an improvement of the basic ACA. Though some achievements have been made, they are not enough as a whole. Some principles are still needed to found to make a proof and test in practice.

3.2 Solve the TSP through particle swarm optimization

Ant colony algorithm is a discrete random number algorithm, which is suitable for solving the discrete optimization problem. TSP is a typical discrete optimization problem, so, since the appearance of ant colony algorithm, many scholars have used this algorithm to solve the TSP. However, as the travelling salesman problem is a NP, and the pheromone needs to be updated when ant colony algorithm is iterated each time, so, when solving the large-scale TSP, it will meet some problems such as slow searching speed. Though scholars at home and abroad have made some efforts to accelerate the searching speed, but what they've done is not enough as a whole. Some principles are still needed to found to make a proof and test in practice. Particle swarm optimization is a continuous algorithm. Its iteration formula is simple and easy to achieve. A slight improvement of this algorithm will help you to solve the discrete optimization problem of the travelling salesman. As its iteration formula is very simple, a use of this algorithm may help you to solve the slow searching speed problem found from the ant colony algorithm.

At present, different improvement algorithms for PSO have been provided to solve the TSP. In particular, great result has been made by Maurice who used discrete PSO algorithm to solve the TSP.A hybrid PSO algorithm which is used to solve the TSP is provided on the basis of GA, AC and SA. Application of PSO algorithm to solve the travelling salesman problem is a fresh attempt. However, as the traditional PSO will easily trap in the local optimal solution, we provide two improve strategies for the standard PSO and use them to solve the TSP.

4. Solve the TSP through improved PSO algorithm

4.1 Solve the TSP through DPSO algorithm

4.1.1 DPSO principle

Dynamic programming is a typical deterministic algorithm for solving the optimization problem. It is provided on the basis of the optimality principle and non-aftereffect and used for the algorithm of multistage decision process. Optimality principle: any truncation of the optimal decision still remains the optimal state; non-aftereffect: after truncation in any stage, the decision made in the later stage is only connected to the initial state of this stage and has no connection to others. Dynamic programming, through optimality principle and non-aftereffect, analyze the optimization problem in stages to simplify the problem, which greatly reduce the calculation steps.

PSO algorithm is an interactive parallel searching algorithm as well as a good attempt to look for global extremum. However, when solving the optimization problem of high dimensional function, as the mutual restraint exists between each dimensional variable, disadvantage has been found when the PSO algorithm is used to solve this problem. According to the numerical value test result, this algorithm is proven to be very effective when the dimension is low. The solving process of dynamic programming is to simplify the complex problem to obtain the solution. A combination of this with the property of PSO algorithm will surely improve the optimal performance of the PSO algorithm.

As for the solution of the problem $\min f(\mathbf{x}) = f(x_1, x_2, \cdots, x_n),\ \ s.t.\ \ \mathrm{a_i} \le x_i \le b_i,\ \ i = 1,2,\cdots n.$ (4.1.1.1), a strategy should be provided to fix some variables and change the remaining variables; i.e. partition the variable and approximate the optimal solution of the majorized function through partitioning to convert the high dimensional optimization problem into low dimensional optimization problem to get the condition optimal solution. Then fix the other part to get the other group of condition optimal solution. Use this information to carry out a comprehensive optimization process. Be aware that this strategy is different from the principle of dynamic programming, because aftereffect exists when partition optimization is applied. So, a strategy method concerning reasonable approximation of global extremum should be provided for the partition optimization of aftereffect.

It is hard to decide the order of fixed variable in the process of calculation. Different strategies can be used during the process of practical operation; after the algorithm traps in the local extremum, it may pick some components to be fixed randomly from the local optimal solution, or choose some components alternately; at the same time, transform the original problem into two problems after some components are picked randomly. If the dimension is too high, this problem can also be transformed into multiple problems to find a solution. See the following problem

$$\min f(x_1, x_2, x_3, x_4, x_5, x_6)\ , \tag{4.1.1.2}$$

If PSO algorithm gives a local optimal solution $\mathbf{x}^{1^*} = (x_1^{1^*}, x_2^{1^*}, \cdots x_6^{1^*})$, the following two strategies can transform the high dimension optimization into low dimension optimization: (1) pick several components randomly, e.g. pick 3 components $x_1^{1^*}, x_2^{1^*}, x_4^{1^*}$, then the result is

$$\min f(x_1^{1^*}, x_2^{1^*}, x_3, x_4^{1^*}, x_5, x_6) \tag{4.1.1.3}$$

A local optimal solution $(x_1^{1^*}, x_2^{1^*}, x_3^{2^*}, x_4^{1^*}, x_5^{2^*}, x_6^{2^*})$ is given by using the PSO algorithm again. Then pick some components randomly or alternately (for example, if you pick components 1, 2 and 4 last time, you can pick components 3, 5 and 6 this time); in this way, a new optimal problem is found. Continue the run until you find a satisfactory result. (2) Pick some components randomly and divide the original problem into several problems, including: $\min f(x_1^{1^*}, x_2^{1^*}, x_3, x_4^{1^*}, x_5, x_6)$ and $\min f(x_1, x_2, x_3^{1^*}, x_4, x_5^{1^*}, x_6^{1^*})$. It may write down all the possible forms (i.e. $C_6^3 = 20$) of the three variables to divide the original problem into 20 optimization problems. If you think the dimension is too high, pick p (p is relatively high in number) components randomly and transform the original problem into several optimization problems. You can also list all the C_n^p optimization problems and use PSO algorithm to solve several optimization problems you get. Then compare the results of these optimization problems and pick the best one to use as the local optimal solution next step, and further analyze this solution until you find the satisfactory result.

4.1.2 Computational steps of the DPSO

As for the optimization problem of the formula (4.1.1.1), the key algorithm steps are as follows:

Step 1. Randomly generate the initial population m. Under normal circumstances, $m \geq 10$.

Step 2. After figure up certain algebras through PSO or after use PSO and find that the target values within several successive algebras remain the same, set the optimal solution as $\mathbf{x}^* = (x_1^0, x_2^0, \cdots x_n^0)$.

Step 3. Pick $[\frac{n}{2}]$ component randomly from the optimal solution $\mathbf{x}^*$ and set it as $x_{i_1}^0, x_{i_2}^0, \cdots x_{i_{[\frac{n}{2}]}}^0$.

Step 4. Use PSO to solve the following two optimization problems

$$\min f(\mathbf{x}) = f(x_1, x_2, \cdots, x_{i_1}^0, \cdots x_{i_2}^0, \cdots x_{i_{[\frac{n}{2}]}}^0, \cdots, x_n), \tag{4.1.2.1}$$

and

$$\min f(\mathbf{x}) = f(x_1^0, x_2^0, \cdots, x_{i_1}, \cdots x_{i_2}, \cdots x_{i_{[\frac{n}{2}]}}, \cdots, x_n^0). \tag{4.1.2.2}$$

In these two optimization problems, one is the function of $n-[\frac{n}{2}]$ dimension and the other is the function of $[\frac{n}{2}]$ dimension.

Step 5. Choose the best result from these two optimization problems to use as the current optimal solution $\mathbf{x}^*$ to see if it can reach a satisfactory result. If not, iterate the steps by starting from step 3; if a satisfactory result is obtained, terminate the computational process and get the optimal solution.

Note: Other strategies may be applied to Step 3, and here is only one of them. In order to ensure the rapid convergence of the algorithm, pick the optimal solution after each calculation to use it as a particle for the calculation next time.

4.1.3 Solve the TSP through DPSO

For the TSP with n cities ($a_1, a_2, \cdots a_n$), use $(a_{i1}, a_{i2}, \cdots, a_{in}, a_{i1})$ to represent the route (i.e. $a_{i1} \rightarrow a_{i2} \cdots \rightarrow a_{in}$). $a_{i1}, a_{i2}, \cdots, a_{in}$ is an array of $a_1, a_2, \cdots, a_n$ and is called solution sequence.

As stated above, DPSO algorithm is applicable to the continuous problem. As TSP is a typical discrete problem, its solution is a sequence or loop rather than a point within the solution space. In order to apply DPSO to TSP, we introduce to you some definitions and algorithms of the solution sequence.

Definition 1 Exchange and exchange sequence Exchange the j point and k point of the solution sequence to form a new solution sequence. This is called exchange and is indicated with $E(j,k)$.Exchange a_{ij} and a_{ik} in the solution sequence of $T=(a_{i1},a_{i2},\cdots,a_{ij},\cdots,a_{ik},\cdots,a_{in})$. The new solution after exchange is $T+E(j,k)$. The ordered sequence $Q=(E_1,E_2,\cdots,E_m)$ after m times of exchanges is called exchange sequence. Exchange T through the exchange sequence in an orderly manner to generate a new solution. i.e.

$$T+Q=T+(E_1,E_2,\cdots,E_m)=[(T+E_1)+E_2]+\cdots+E_m \tag{4.1.3.1}$$

When $m=0$, Q is equivalent to empty sequence. This means that formula (6.4.1.3.1) doesn't do any exchange for the solution sequence. Under such circumstances, you can add an exchange result to the exchange sequence and place this exchange result to the end of the sequence to form a new sequence.

Definition 2 Solution sequence difference As for any two solution sequences T_1 and T_2 of the same TSP, the exchange sequence Q always exists. As a result, $T_2=T_1+Q$ is formed. Q is the difference of the solution sequences T_2 and T_1, i.e. the result of T_2-T_1.When $T_1=(a_1,a_2,\cdots a_n)$ and $T_2=(b_1,b_2,\cdots b_n)$ are found, you can use the following procedure 1 to calculate $Q=T_2-T_1$.

```
Procedure 1    $Q$ = empty sequence
    for $j=1$ to $n$
    for $i=1$ to $n$
    if $a_i=b_j$ and $i\neq j$ then add $E(i,j)$ to $Q$
        end
  end
```

In respect of T_1 and T_2, there are many Qs to be used in the formula $T_2=T_1+Q$.

Definition 3 Product of decimal and exchange sequence $\eta\in(0,1)$ and exchange sequence is Q which has an exchange of n_0. If $\frac{m_0}{n_0}\leq\eta<\frac{m_0+1}{n_0}$ (m_0 is an integer from 0 to n_0-1), $\eta\cdot Q$ is the sequence formed by m_0 exchange before Q.

Through this operation, the above algorithm can be used to solve the discrete optimization problem like TSP.

4.1.4 Test and discussion of the performance of the algorithm

Use 14 points of the TSP provided by Huang Lan and other people to test the effectiveness of the algorithm. Description of the 14 points of the TSP is listed in table 1.

Point	1	2	3	4	5	6	7	8	9	10	11	12	13	14
X	16.47	16.47	20.09	22.39	25.23	22.00	20.47	17.20	16.30	14.05	16.53	21.52	19.41	20.09
Y	96.10	94.44	92.54	93.37	97.24	96.05	97.02	96.29	97.38	98.12	97.38	95.59	97.13	94.55

Table 1. Position data for 14 points

Use DPSO to carry out 8 times of tests and set the parameters as $\omega_1 = 0.53$, $\eta_1 = 0.35$ and $\eta_2 = 0.45$. The number of the initial population is 600. Set the maximum iterative number as 300. The result as follows:

Test serial number	1	2	3	4	5	6	7	8
Get the algebra of the optimum value 30.8785	58	30	58	58	58	58	93	196
Get the best route each time	6-12-7-13-8-11-9-10-1-2-14-3-4-5							

Tabela 2.

Algorithm analysis table

Number of the solution space	(14-1)!/2=3 113 510 400
Average iterative number	$(58\times5+30+93+196)/8=76.125$
Average search space for each test	$600+76.125\times200=15825$
Proportion of the search space to solution space	15825/3113510400=0.000508%

Tabela 3.

From the above test, we can see that DPSO may go beyond the local extremum to gen the final optimal solution for the problem. To achieve this, we should transform the high dimension optimization into low dimension optimization. We should optimize the remaining components while maintain some components unchanged; by doing this alternately, the ability for algorithm to optimize the high dimension problem will be strengthened. This improved algorithm only represents an improvement on the calculative strategy front. It does not add additional calculation and step to the algorithm, hence, maintaining the simplification of the PSO algorithm. At the same time, it helps to transform a high dimension optimization problem into several low dimension optimization problems, which will not complicate the calculation procedure.

4.2 Solve the TSP through MCPSO

When use MCPSO to solve the TSP, you also need to go through the relevant procedure which is used by continuous optimization algorithm to solve the discrete optimization problem; except for the above methods, MCPSO also has midpoint problem, so we introduce you the following definition:

Definition Midpoint solution sequence Set two solution sequences $T_1 = (a_1, a_2, \cdots a_n)$ and $T_2 = (b_1, b_2, \cdots b_n)$ for n cities of TSP and make the solution sequence as $T = (a_1, a_2, \cdots, a_{n_1}, b_{n_1+1}, b_{n_1+2}, \cdots, b_n)$ ($n_1 = [n/2]$).If repetitive point appears in the solution, adjustment can be made according to the procedure 2 to make it become a feasible solution sequence and call it a midpoint solution sequence of T_1 and T_2.

Procedure 2

Step 1. Search for the repetitive point of $a_1, a_2, \cdots, a_{n_1}$ from $b_{n_1+1}, b_{n_1+2}, \cdots, b_n$ and replace it with 0;

Step 2. Search for the points which are different from the points of $b_{n_1+1}, b_{n_1+2}, \cdots, b_n$ from $a_{n_1+1}, a_{n_1+2}, \cdots, a_n$ and replace the 0 point in an orderly manner.

4.2.1 Steps for MCPSO to solve the TSP

The steps for MCPSO algorithms to solve the TSP are as follows:

Step 1. Set relevant parameters l, β_1, β_2 and δ, and begin to conduct initialization complex. Each point is the solution sequence generated randomly and is indicated with $\mathbf{x}$;

Step 2. Pick l solution sequences, good and bad, for $\mathbf{x}_r$ and $\mathbf{x}_f$, and calculate the midpoint sequence $\mathbf{x}_m$ and ratio λ. Then determine the best solution sequence $\mathbf{x}_1$;

Step 3. Based on certain probability,
Pick formula $\mathbf{x}_p = \mathbf{x}_f + \phi_1(\mathbf{x}_r - \mathbf{x}_f) + \phi_2(\mathbf{x}_1 - \mathbf{x}_f)$ through probability β_1
Pick formula $\mathbf{x}_p = \mathbf{x}_r + \phi_1(\mathbf{x}_r - \mathbf{x}_f) + \phi_2(\mathbf{x}_1 - \mathbf{x}_f)$ through probability β_2
Pick formula $\mathbf{x}_p = \mathbf{x}_f + \phi_1(\mathbf{x}_f - \mathbf{x}_r) + \phi_2(\mathbf{x}_1 - \mathbf{x}_f)$ through probability $1 - \beta_1 - \beta_2$

to get m new solution sequences $\mathbf{x}_p$ to replace the bad solution sequence $\mathbf{x}_f$ to form a new complex;

Step 4. If the satisfactory result is reached, go to Step 5; otherwise, go back to Step 2;

Step 5. Show the optimal solution.

4.2.2 Test and discussion of the performance of the algorithm

Test the algorithm based on the 14 points of the TSP provided by Huang Lan and other people. The optimum value is 30.8785.We use this problem to test the optimal performance of MCPSO algorithm. Its parameters are $\delta = 0.85$, $\beta_1 = 0.675$, $\beta_2 = 0.175$ and $l = 50$. The pop-size is 600 and the upper limit of iterative number is 200. In order to facilitate comparison, we also use SGA and ACO to solve this problem. These two have the same pop-size and iteration upper limit as MCPSO. Each algorithm is run for 10 times. The parameter setting for these two algorithms are: SGA: multiplying probability $P_r = 0.2$,crossing probability $P_c = 0.6$ and mutation probability $P_m = 0.05$; ACO: constant $C = 20$, pheromone factor $\alpha = 1$, heuristic factor $\beta = 1$, and information keeping factor $\rho = 0.8$. The results of these algorithms are shown in the table 4.2.2.1 and the change curve of average mean fitness is shown in the figure 4.2.2.1.

ACO and integer-coded SGA can be directly used to solve the discrete optimization problems such as TSP. These algorithms have the ability to search for the global optimal solution, but the efficiency is relatively low as they can only make a change based on the probability. MCPSO is a continuous algorithm which introduces the group searching mechanism of PSO into the complex method. It considers the global property between solutions through geometry point, optimization and other principles so as to shorten the distance between the solution with poor adaptability and the solution with good adaptability. In order to avoid being trapped in the local extremum, certain probability will be considered. Shortening the distance between bad solution and good solution will help you to get the optimal solution in a more precise way within a short time, and greatly enhance the searching ability. The appearance of the algorithm targeted to TSP solution sequence not only helps to keep the above characteristics of MCPSO, but also guarantees the effective application of MCPSO to discrete problems. As for the 14 points of TSP, the running of MCPSO algorithm (7 out of 10 times) will help you to find the optimal solution with relatively low iterative number. However, after 10 times of running of ACO and SGA, no optimal solution is found. From this, we can see the advantage of MCPSO.

Algorithm	Number of times of reaching the optimum value	Minimum algebra for reaching the optimum value	Average algebra for reaching the optimum value	Best value	Average value	Standard deviation
ACO	0	N/A	N/A	31.8791	33.6888	3.7000
SGA	0	N/A	N/A	34.4509	35.9578	3.4209
MCPSO	7	35	143.57	30.8785	31.0262	0.7137

Table 4. Comparison of the results from three algorithms

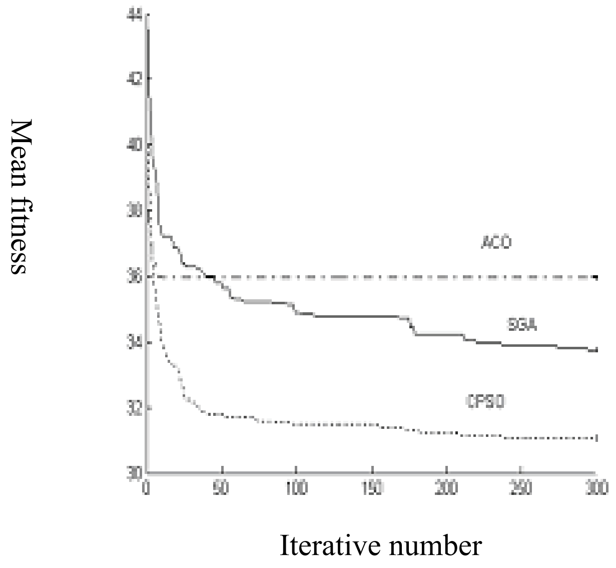


Fig. 1.

5. Application of the improved PSO algorithm for the TSP

PCB's digital control drilling problem can be described as follows: Process all the holes by starting from the tool changing point (repetition and omission are not allowed). After the processing, return to this point to do tool changing and processing for other aperture. In terms of digital control programming, we should consider the order of drill hole processing to minimize the time idle running, i.e. the best route problem of tool changing or the TSP problem in nature. With regard to the processing problem for a series of holes, the coordinate for these 20 holes has been listed in the figure 5.1. We use PCB-CAD software and PSO, SGA, ACO, DPSO and MCPSO to solve this problem. The parameter setting for PSO is: $\omega = 0.25$, $c_1 = 0.3$ and $c_2 = 0.45$. The speed will be indicated through exchange sequence. The parameters of the other three algorithms are the same as above. The tool changing routes generated are shown in figures 5.1 to 5.6. The latter five algorithms are run individually for 10 times with the upper limit of iterative number each time of 200 and pop-size of 600. The figures presented are their optimal structures. The path lengths for the tool changing routes generated from 6 algorithms are given in table 5.2.

No.	x	y	No.	x	y	No.	x	y
1	1	1	8	2.5	7.5	15	7	15.5
2	1	3	9	2.5	1	16	7	13.5
3	1	7	10	3.5	2	17	7	12.1
4	1	8	11	3.5	8.2	18	7	12
5	2.5	14	12	3.5	12.9	19	7	10
6	2.5	13.5	13	3.5	13.2	20	7	4
7	2.5	13	14	3.5	13.9			

Table 5. Position for 20 holes

Algorithm	PCB-CAD	ACO	SGA	PSO	MCPSO	DPSO
Average length	61.5555	60.5610	58.6334	59.4244	43.4923	44.4978
Minimum length	61.5555	56.7481	52.2687	53.2687	40.1203	40.1203

Table 6. Calculation result comparison

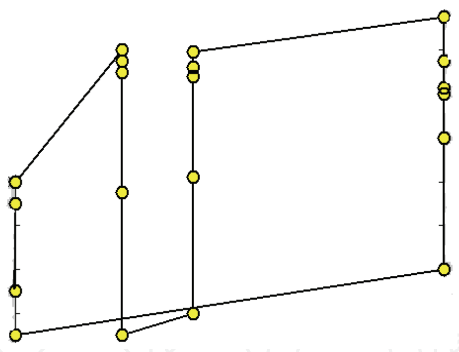


Fig. 5.1 Tool changing route chart generated from PCB-CAD

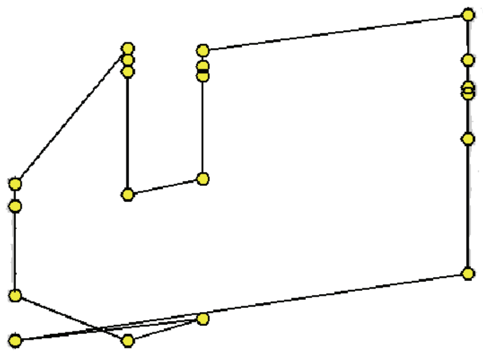


Fig. 5.2 Tool changing route chart generated from PSO

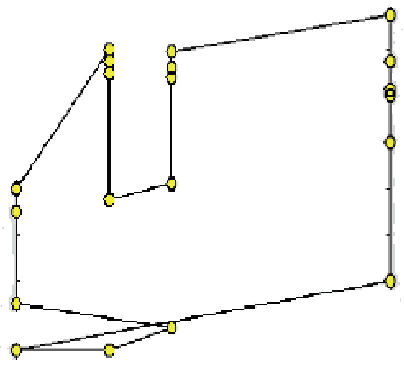


Fig. 5.3 Tool changing route chart generated from SGA

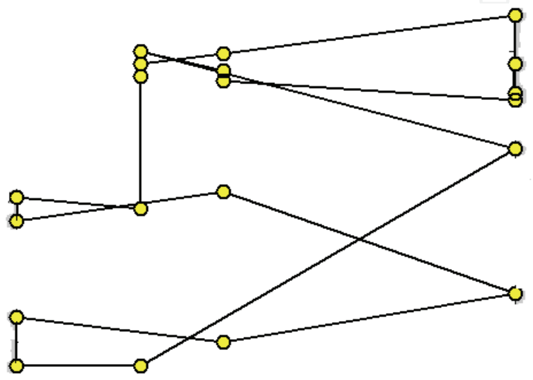


Fig. 5.4 Tool changing route chart generated from ACO

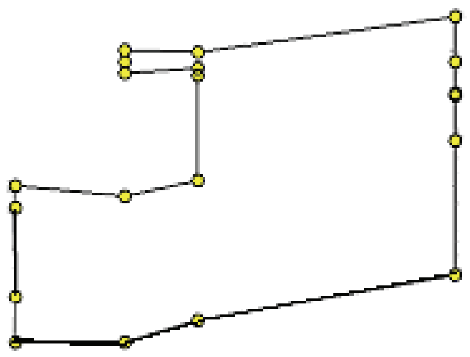


Fig. 5.5 Tool changing route chart generated from MCPSO

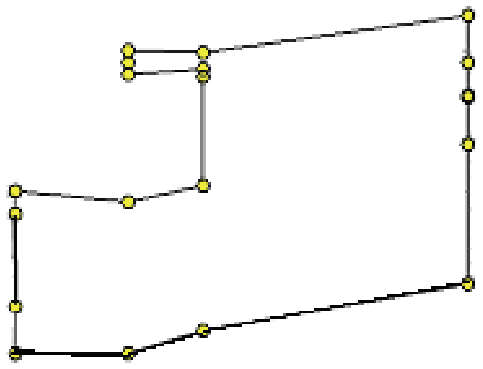


Fig. 5.6 Tool changing route chart generated from DPSO

From above, see can see that the lengths of tool changing routes generated from optimization algorithms are shorter than that generated from PCB-CAD software, of which the MCPCO enjoy the shortest length (about 29% shorter than others). Determination of the best route for PCB digital control drilling can effectively solve the optimization problem of the digital control programming in the course of PCB processing and develop a PCB automatic programming system.

6. Summary

This article consists of the definition of TSP, mathematical description methods, traditional solving methods for the TSP and problems existing in the traditional solving methods. At the same time, it introduces the evolution algorithms for solving the TSP. Based on this, two algorithms (MCPSO and DPSO) are provided. Finally, it shows us the best tool changing route for the digital control drilling by using the algorithms given.

7. References:

[1] Mo Yuanbin, Expansion and Application of PSO (D). Hangzhou: Zhejiang University. 2007

[2] Garey M R, Johnson D S. Computers and Intractabilitys: A Guide to the Theory of NP-Completeness [M]. San Francisco: Freeman WH , 1979.

[3] Maurice Clerc. Discrete Particle Swarm Optimization Illustrated by the Traveling Sales man Problem [DB/OL].
http: //www. mauriceclerc. net, 2000.

[4] Gao S , Han B,Wu X J . et al. Solving Traveling Sales2man Problem by Hybrid Particle Swarm Optimization Algorithm [J]. *Control and Decision*, 2004, 19 (11) :1286-1289.

[5] (US) Robert E. Larson (Writer). Chen Weiji (Translator). Dynamic Programming. 1st edition. Beijing: Tsinghua University Press, 1984

[6] (US) Leon Cooper and Mary W. Cooper (Writers). Zhang Youwei (Translator). Dynamic Programming. 1st edition. Beijing: National Defence Industrial Press, 1985

[7] Huang Lan and Wang Kangping, etc. Solve the Traveling Salesman Problem through PSO Algorithm [J]. Journal of Jilin University, 2003 (4): 477-480 ·

[8] Meng Fanzhen, Hu Yunchang and Xu Hui, etc. Genetic Algorithm of the Traveling Salesman Problem [J]. System Engineering Theory and Practice, 1997, 2 (7): 15-21.

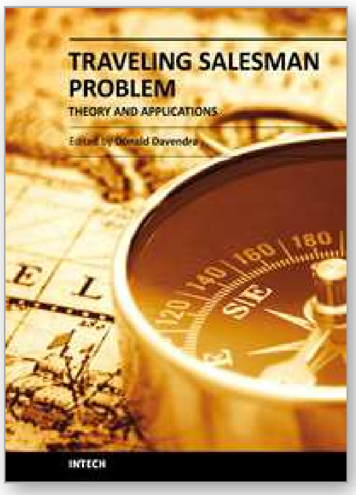
[9] Conley WC. Programming an automated punch or drill[J]. International Journal of Systems Science, 1991, 22(11): 2039-2025.

[10] Wang Xiao and Liu Huixia. Modeling and Solving of the Best Tool Changing Route for PCB Digital Control Drilling [J]. Journal of Computer Aided Design and Graphics. 2001, 13 (7): 590-593.

[11] Colorni, A., Dorigo, M., and Maniezzo, V. Distributed optimization by ant colonies. Proceedings of the First European Conference on Artificial Life, Paris, France, Varela, F. and Bourgine, P. (Eds.), Elsevier Publishing, 1991, 134-142.

IntechOpen

IntechOpen



Traveling Salesman Problem, Theory and Applications
Edited by Prof. Donald Davendra

ISBN 978-953-307-426-9
Hard cover, 298 pages
Publisher InTech
Published online 30, November, 2010
Published in print edition November, 2010

This book is a collection of current research in the application of evolutionary algorithms and other optimal algorithms to solving the TSP problem. It brings together researchers with applications in Artificial Immune Systems, Genetic Algorithms, Neural Networks and Differential Evolution Algorithm. Hybrid systems, like Fuzzy Maps, Chaotic Maps and Parallelized TSP are also presented. Most importantly, this book presents both theoretical as well as practical applications of TSP, which will be a vital tool for researchers and graduate entry students in the field of applied Mathematics, Computing Science and Engineering.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Yuan-bin Mo (2010). The Advantage of Intelligent Algorithms for TSP, Traveling Salesman Problem, Theory and Applications, Prof. Donald Davendra (Ed.), ISBN: 978-953-307-426-9, InTech, Available from: http://www.intechopen.com/books/traveling-salesman-problem-theory-and-applications/the-advantage-of-intelligent-algorithms-for-tsp

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen