第 30 卷第 8 期
2013 年 8 月

控 制 理 论 与 应 用
Control Theory & Applications

Vol. 30 No. 8
Aug. 2013

DOI: 10.7641/CTA.2013.12167

一种求解旅行商问题的离散状态转移算法

阳春华[1], 唐小林[1], 周晓君[1,2†], 桂卫华[1]

(1. 中南大学 信息科学与工程学院, 湖南 长沙 410083;
2. 巴拉瑞特大学 科学与信息技术及工程学院, 维多利亚 巴拉瑞特 3350, 澳大利亚)

摘要: 本文提出了一种求解旅行商问题的离散状态转移算法, 设计了交换、平移、对称等3种转移算子, 讨论了算法的收敛性和时间复杂度等问题, 研究了参数对算法的影响. 实验结果表明, 与模拟退火算法及蚁群算法等经典组合优化算法相比, 该算法具有耗时短、寻优能力强等优点, 这也表明了状态转移算法的适应性很好.

关键词: 状态转移算法; 旅行商问题; 参数学习; 组合优化

中图分类号: TP273　　**文献标识码**: A

A discrete state transition algorithm for traveling salesman problem

YANG Chun-hua[1], TANG Xiao-lin[2], ZHOU Xiao-jun[1,2†], GUI Wei-hua[1]

(1. School of Information Science and Engineering, Central South University, Changsha Hunan 410083, China;
2. School of Science, Information Technology and Engineering, University of Ballarat, Ballarat Victoria 3350, Australia)

Abstract: A discrete version of state transition algorithm is proposed to solve the traveling salesman problem. Three special operators named swap, shift and symmetry transformations are presented for discrete optimization problem. Convergence analysis and time complexity of the algorithm are also considered. To make the algorithm efficient, a parametric study is investigated. Experiments are carried out to test its performance, and comparisons with simulated annealing and ant colony optimization have demonstrated the effectiveness of the proposed algorithm. The results also show that the discrete state transition algorithm consumes much less time and has better search ability than other traditional combinatorial optimization methods, indicating that state transition algorithm has strong adaptability.

Key words: state transition algorithm; traveling salesman problem; parametric study; combinatorial optimization

1 Introduction

As one of the most significant combinatorial optimization problems, traveling salesman problem (TSP) has been paid great attention and extensively studied, due to its importance in manufacturing, distribution management and scheduling[1–2]. Traveling salesman problem can be described as: given a set of n nodes and distances for each pair of nodes, find a roundtrip of minimal total length visiting each node exactly once, and according to whether the distance from node i to node j is the same as the distance from node j to node i, a TSP can be symmetric or asymmetric[3]. For an n nodes asymmetric TSP, there exist $(n-1)!$ possible solutions; while for the corresponding symmetric TSP, $(n-1)!/2$ solutions are possible. Because of its NP-hard property, emphasis has shifted from the aim of finding a global optimal solution to the goal of obtaining 'good solutions' in reasonable time and establishing the 'the degree of goodness'[4–5].

Taking the impractical exhaustive search for TSP into consideration, heuristic algorithms are introduced to speed up the process of finding a satisfactory solution, of which, simulated annealing (SA), tabu search (TS), genetic algorithm (GA), ant colony optimization (ACO), particle swarm optimization (PSO) have found widest applications in the field[6–10]. In terms of the concepts of state and state transition, a new heuristic search algorithm named state transition algorithm (STA) is proposed recently, and it exhibits good search performance in continuous space[11–13]. In the initial version of state transition algorithm, a special transformation matrix named general elementary matrix was also proposed to solve the discrete optimization problem. In this paper, we will continue to develop its details to promote a deep study of STA for solving traveling salesman problem.

The paper is organized as follows: In Section 2, we review the unified form of the state transition algorithm, and establish the discrete version of STA. Then details and implementations of three special transformation operators are put forward in Section 3, and a parametric study is also investigated. In the next section, some experiments are executed to test the performance of the proposed algorithm. Conclusion is derived in the end.

Received 14 July 2012; revised 2 April 2013.
†Corresponding author. E-mail: E-mail: tiezhongyu2010@gmail.com; Tel.: +61 0416 370650.
This work was supported by the the National Science Found for Distinguished Young Scholars of China (No. 61025015), and the China Scholarship Council.

2 State transition algorithm in discrete version

As stated in [11], a solution to a specific optimization problem can be described as a state, and the process to update solutions will become a state transition process. Without loss of generality, the unified form of state transition algorithm can be described as follows:

$$\begin{cases} x_{k+1} = A_k x_k + B_k u_k, \\ y_{k+1} = f(x_{k+1}), \end{cases} \tag{1}$$

where, $x_k \in \mathbb{R}^n$ stands for a state, corresponding to a solution to a optimization problem; A_k, $B_k \in \mathbb{R}^{n\times n}$ are state transition matrixes, which are usually transformation operators; $u_k \in \mathbb{R}^n$ is the function of x_k and historical states; f is the cost function or evaluation function.

As for traveling salesman problem, the cost f is usually expressed as the function of a sequence, which corresponds to an ordered traveling route. That's to say, after a transformation using A_k or B_k, a new state x_{k+1} should be a sequence too. Due to the particularity of TSP, only a state transition matrix is considered, avoiding the complexity of 'adding' one sequence to another.

According to the theory of linear algebra, we know that the special matrix must have only one position with value 1 in each column and each row. The special random matrix is called general elementary matrix, since it is generated from the identity matrix and has the function of transforming a sequence into another one. For simplicity and specificity, G_k is denoted in state transition algorithm for discrete optimization problem as follows:

$$\begin{cases} x_{k+1} = G_k x_k, \\ y_{k+1} = f(x_{k+1}), \end{cases} \tag{2}$$

where, $x_k = [x_{1k}\ \ x_{ik}\ \cdots\ x_{nk}]^{\mathrm{T}}$, $x_{ik} \in \{1, 2, \cdots, n\}$ is a sequence, and the G_k is general elementary matrix with only one position value 1 in each column and each row. For instance, G_k has the following styles[11]

$$\begin{pmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&0&0&1\\ 0&0&0&1&0\\ 0&0&1&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&1&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&0&1\\ 1&0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&1&0\\ 0&1&0&0&0\\ 1&0&0&0&0\\ 0&0&0&0&1\\ 0&0&1&0&0 \end{pmatrix}.$$

If an initial sequence is $\{1, 2, 3, 4, 5\}$, after transformation by the above general elementary matrices, it will becomes $\{1, 2, 5, 4, 3\}$, $\{4, 2, 3, 5, 1\}$, $\{4, 2, 1, 5, 3\}$, respectively.

3 State transition operators and corresponding adjusting strategies

As can be seen from above, the general elementary matrix can behave in various types, but arbitrary types may have neither validity nor efficiency for discrete optimization problem. To make the state transition process controllable and efficient, three special transformation operators for traveling salesman problem are designed.

3.1 Three transformation operators

1) Swap transformation

$$x_{k+1} = G_k^{\mathrm{swap}}(m_{\mathrm{a}})x_k, \tag{3}$$

where, $G_k^{\mathrm{swap}} \in \mathbb{R}^{n\times n}$ is called a swap transformation matrix; m_{a} is a constant called swap factor to control the maximum number of random positions to be exchanged. For example, if $n = 5$, $m_{\mathrm{a}} = 3$, the swap transformation matrix has the following styles:

$$\begin{pmatrix} 0&0&1&0&0\\ 0&1&0&0&0\\ 1&0&0&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1 \end{pmatrix}, \begin{pmatrix} 1&0&0&0&0\\ 0&0&1&0&0\\ 0&1&0&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1 \end{pmatrix}, \begin{pmatrix} 1&0&0&0&0\\ 0&0&0&0&1\\ 0&0&1&0&0\\ 0&1&0&0&0\\ 0&0&0&1&0 \end{pmatrix}.$$

If an initial sequence is $\{1, 2, 3, 4, 5\}$, then after the above swap transformation matrices, it will becomes $\{3, 2, 1, 4, 5\}$, $\{1, 3, 2, 4, 5\}$, $\{1, 5, 3, 2, 4\}$, respectively. It can be found that the 1th and 3th rows are exchanged, the 2th and 3th rows are exchanged, and the 2th, 4th, and 5th rows are interchanged, respectively, which indicates that the maximum number of positions exchanged is 3.

2) Shift transformation

$$x_{k+1} = G_k^{\mathrm{shift}}(m_{\mathrm{b}})x_k, \tag{4}$$

where, $G_k^{\mathrm{shift}} \in \mathbb{R}^{n\times n}$ is called a shift transformation matrix; m_{b} is a constant called shift factor to control the maximum length of consecutive positions to be shifted. By the way, the selected position to be shifted after and positions to be shifted are chosen randomly. To make it more clearly, let $n = 5$, $m_{\mathrm{b}} = 2$, then the shift transformation matrix has the following styles:

$$\begin{pmatrix} 1&0&0&0&0\\ 0&0&1&0&0\\ 0&1&0&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1 \end{pmatrix}, \begin{pmatrix} 1&0&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&1&0&0&0\\ 0&0&0&0&1 \end{pmatrix}, \begin{pmatrix} 1&0&0&0&0\\ 0&0&0&1&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&0&1 \end{pmatrix}.$$

If an initial sequence is $\{1, 2, 3, 4, 5\}$, then after the above shift transformation matrices, it will becomes $\{1, 3, 2, 4, 5\}$, $\{1, 3, 4, 2, 5\}$, $\{1, 4, 2, 3, 5\}$, respectively. In the first two cases, the position to be shifted is $\{2\}$, and the positions to be shifted after are $\{3\}$ and $\{4\}$, while in the third case, the positions to be shifted is $\{2, 3\}$, and the position to be shifted after is $\{4\}$. In the last case, the length of consecutive positions to be shifted is 2.

3) Symmetry transformation

$$x_{k+1} = G_k^{\mathrm{sym}}(m_{\mathrm{c}})x_k, \tag{5}$$

where, $G_k^{\mathrm{sym}} \in \mathbb{R}^{n\times n}$ is called a symmetry transformation matrix; m_{c} is a constant called symmetry factor to control the maximum length of subsequent positions as a center. By the way, the components before the subsequent positions and consecutive positions to be symmetrized are both created randomly. For instance, $n = 5$, $m_{\mathrm{c}} = 1$, then the symmetry transformation matrix has the following styles:

$$\begin{pmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&0&1&0\\ 0&0&1&0&0\\ 0&0&0&0&1 \end{pmatrix}, \begin{pmatrix} 1&0&0&0&0\\ 0&0&0&0&1\\ 0&0&0&1&0\\ 0&0&1&0&0\\ 0&1&0&0&0 \end{pmatrix}, \begin{pmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&0&0&1\\ 0&0&0&1&0\\ 0&0&1&0&0 \end{pmatrix}.$$

If an initial sequence is $\{1, 2, 3, 4, 5\}$, then after the above symmetry transformation matrices, it will becomes

$\{1,2,4,3,5\}$, $\{1,5,4,3,2\}$, $\{1,2,5,4,3\}$, respectively. All of the three cases, the component before the subsequent positions is $\{3\}$. In the first two cases, the subsequent position (or the center) is $\{\varnothing\}$, while the consecutive positions to be symmetrized are $\{4\}$ and $\{4,5\}$, respectively. In the third case, the subsequent position (or the center) is $\{4\}$, while the consecutive position to be symmetrized is $\{5\}$. It is not difficult to find that the length of subsequent positions is 1 in the last case, which is the maximum length of subsequent positions as indicated by $m_{\rm c}$.

We propose the above three transformation operators due to their geometrical significance. In geometry, swap, shift and symmetry operators have different topological structures, that is to say, they possess various functions, which play significant roles in perturbing current solutions. By the way, it should be noted that the swap operator with $m_{\rm a}=2$ is also widely used in simulated annealing.

3.2 Adjusting strategies

To accept a new solution, 'greedy criterion' is commonly adopted, in other words, only a solution better than previous one is accepted. However, in SA, a bad solution is accepted probabilistically. Although the strategy in SA can jump out of a stagnant solution and it creates a big perturbation to current solution, to guarantee the convergence of SA becomes difficult. In state transition algorithm for discrete optimization problem, 'greedy criterion' is inherited in this paper. The main procedure of the STA for TSP can be outlined in pseudocode as follows:

```
State ← initiation(SE, n);
[Best, fBest] ← fitness(State, cities, SE);
k ← 0
repeat
    [Best, fBest] ← op_swap(cities, Best,
                            fBest, SE, n, m_a)
    [Best, fBest] ← op_shift(···, m_b)
    [Best, fBest] ← op_symmetry(···, m_c)
    k ← k + 1
until the maximum number of iterations is met
```

where, in initiation, a set of states are created randomly; cities is the information about the TSP; the SE is the times of transformation, called search enforcement. As for more detailed explanations, op_swap function of above pseudocode is also given in MATLAB scripts

```
for i = 1: SE
    Tranf = swap_matrix(n, ma);
    State(i,:) = (Tranf*Best')';
end
[newBest, fGBest] = fitness(State, cities, SE)
  if fGBest < fBest
    Best = newBest;
    fBest = fGBest;
  end
function y = swap_matrix(n, ma)
y = eye(n);
R = randperm(n);
T = R(1:ma);
S = T(randperm(m));
y(T, :) = y(S, :);
```

4 Theoretical analysis of discrete STA

Next, we analyze the convergence performance, the global search ability, and time complexity of the discrete state transition algorithm.

4.1 Convergence properties of discrete STA

Firstly, we define the concept of convergence for discrete STA

$$|f(x_k)-f(x^*)|\leqslant\epsilon,\ \forall\, k>N, \tag{6}$$

where, x^* is the global minimum solution of a traveling salesman problem, ϵ is a small constant, and N is a natural number. If $\epsilon>0$, we can say that the algorithm converges to an ϵ-optimal solution; if $\epsilon=0$, we can say that the algorithm converges to a global minimum. However, if x^* is a local minimum solution, then we can say that the algorithm converges to a ϵ-suboptimal solution and a local minimum for $\epsilon>0$ and $\epsilon=0$, respectively.

Theorem 1 The discrete STA can at least converge to a local minimum.

Proof Let suppose the maximum number of iterations (denoted by M) is big enough, then there must exist a number $N(N<M)$ such that if $k>N$, no update of the current solution will happen. That is to say, $f(x_k)=f(x^{\rm best}),\ \forall\, k>N$, where $x^{\rm best}$ is the solution at the Nth iteration. The $x^{\rm best}$ is just the local minimum solution denoted as x^*, and $|f(x_k)-f(x^{\rm best})|=0$.

4.2 Global search ability of discrete STA

It is not difficult to find that whether the discrete STA converges to a global minimum depends on the $x^{\rm best}$ in the Nth iteration. If the $x^{\rm best}$ is the global minimum, then according to the 'greedy criterion' used to update the $Best$ in pseudocode, when $k>N$, $Best$ will always be $x^{\rm best}$. In other words, the global convergence performance has much to do with the three operators we have designed.

Theorem 2 The discrete STA can converge to a global minimum in probability.

Proof Let suppose $x^*=(a_1,\cdots,a_n)$ is the global minimum solution, and $x_k=(b_1,\cdots,b_n)$ is the kth best solution. Then we discuss separately that how the state x_k can be transformed to the best state x^*.

If there exists the same sub-sequence in both x_k and x^*, for instance, $x_k=(a_n,a_2,\cdots,a_{n-1},a_1)$, then both swap transformation with small swap factor and appropriate shift transformation have the probability to swap or shift components a_n,a_1 in the sequence so that x_k can be transformed exactly to the corresponding positions in the x^*.

If there exists no same sub-sequence in x_k and x^*, for instance, $x_k=(a_{n-1},a_{n-2},\cdots,a_2,a_1)$, then swap transformation, shift transformation or proper symmetry transformation have the probability to swap, shift or symmetrize proper positions in the sequence so that x_k can be transformed exactly to the corresponding positions in the x^*.

The local property indicates that at each iteration, the current solution in STA is always feasible and better than or the same to previous solutions. The global property indicates that whether the STA can capture the global solution

depends on the current solution as well as the three special transformation operators.

4.3 Time complexity of discrete STA

Because of the NP-hard property of TSP, it is impossible to solve the problem in polynomial time. As described in the Section 1, to obtain a 'good solution' in reasonable time is a optional choice, and this is the same case to the discrete STA, which aims at getting a satisfactory solution in as short time as possible. In the pseudocode as described above, we can find that in the outer loop, there are M iterations, and in the inner loop, there exist three times of SE transformations. It is not difficult to find that the time complexity of the discrete STA is $\mathcal{O}(M \cdot SE)$, that is to say, the discrete STA can achieve the global optimum in polynomial time with probability.

Traditional methods to solve combinatorial optimization problems are based on the branch-and-bound framework[4], and they are essentially in exponential time. For the majority of heuristic algorithms, like STA, a procedure terminates when the maximum number of iterations is met; therefore, the time complexity is polynomial, although at then the global solution may not be captured.

5 Experimental results and discussion

From the MATLAB scripts in Section 3, we can find that as the m_{a} increases, the swap transformation matrix becomes less efficient. The same phenomenon can be observed in other two transformations, and repetition of the transformation matrices occurs frequently (for example, if $m_{\mathrm{a}} = 3$, the transformation matrices happen the same to that of $m_{\mathrm{a}} = 2$ with a high probability).

To investigate the effect of these parameters on the performance of the discrete STA, a parametric study is conducted. Different groups of $(m_{\mathrm{a}}, m_{\mathrm{b}})$ are tested on some TSPLIB[3] instances. and the experiment results are given in Tables 1–3.

From the below results, it indicates that for a fixed m_{a}, as m_{b} increases, the performance of STA becomes worse, and for a fixed m_{b}, the performance becomes worse as well when m_{a} increases. The results confirm the speculation we have mentioned before (by the way, additional tests have testified the same phenomenon for m_{c}). As a result, $m_{\mathrm{a}} = 2, m_{\mathrm{b}} = 1, m_{\mathrm{c}} = 0$ are specified as a good choice in the remaining of this paper.

Table 1 A parametric study for the instance ulysses16

m_{a}	Statistic	m_{b}		
		1	2	3
2	best	73.9876	73.9876	73.9876
	mean	74.0779	74.5528	74.6858
	st.dev.	0.1626	0.4306	0.5465
3	best	73.9876	73.9876	73.9876
	mean	74.2369	74.3590	74.4254
	st.dev.	0.2766	0.4362	0.4575
4	best	73.9876	73.9998	73.9876
	mean	74.3344	74.3893	74.3427
	st.dev.	0.4287	0.4421	0.4521

Table 2 A parametric study for the instance att48

m_{a}	Statistic	m_{b}		
		1	2	3
2	best	3.3724e4	3.4787e4	3.4557e4
	mean	3.4872e4	3.5707e4	3.6137e4
	st.dev.	668.7553	640.9893	910.9418
3	best	3.4337e4	3.4763e4	3.5193e4
	mean	3.5459e4	3.6392e4	3.6521e4
	st.dev.	902.6357	965.4966	933.0365
4	best	3.4695e4	3.5131e4	3.5384e4
	mean	3.6040e4	3.7340e4	3.7242e4
	st.dev.	1.0757e3	1.2148e3	1.3545e3

Table 3 A parametric study for the instance berlin52

m_{a}	Statistic	m_{b}		
		1	2	3
2	best	7.5444e3	7.8739e3	8.0072e3
	mean	8.2472e3	8.5904e3	8.5888e3
	st.dev.	273.4509	259.1225	326.3185
3	best	7.7934e3	8.1545e3	8.3861e3
	mean	8.4674e3	8.6732e3	8.8258e3
	st.dev.	320.0697	294.7743	275.5957
4	best	8.0510e3	8.3657e3	8.4120e3
	mean	8.5308e3	8.9084e3	8.9106e3
	st.dev.	253.1811	342.2800	358.0091

In order to test the performance of the proposed algorithm, SA, ACO, which are recognized as distinguished algorithms for TSP, are used for comparison with STA. In SA, we set initial temperature at 5000, cooling rate at 0.97, and in ACO, $\alpha = 1$, $\beta = 5$, $\rho = 0.9$, where, α, β are used to control the relative weight of pheromone trail and heuristic value, and ρ is the pheromone trail decay coefficient. In ACO and STA, the number of ants or the search enforcement is 20, and the maximum number of iterations is fixed at 200. Considering that SA is usually not a population-based algorithm, the maximum iterations is extended especially for fairness. Therefore, the threshold, or the total number of iterations in SA is set at 4000.

Programs are run independently for 20 trails for each algorithms in MATLAB R2010b (version of 7.11.0.584) on Intel(R) Core(TM) i3-2310M CPU @2.10GHz under Window 7 environment, and comparison results for STA with SA and ACO are listed in Table 4. Some statistics and the run time are utilized to evaluate the performance of algorithms. The best means the minimum of the results, and then it follows the mean, st.dev. (standard deviation). The run time is the average time used in 20 trails, which is measured in seconds.

As can be seen from the Table 4, STA outperforms SA and ACO in almost every performance index. Taking the burma14 for instance, all of these algorithms can achieve the same best solution. But as indicated by the mean, only STA can gain the best solution in a random run. The position location of ulysses16 and ulysses22 is similar, but the best results are quite different.

Table 4 Results for benchmark test problems

Instances	Statistic	SA	ACO	STA
burma14	best	30.8785	30.8785	30.8785
	mean	31.1483	32.0654	30.8785
	st.dev.	0.3360	0.5476	7.9022e−15
	time/s	1.9482	9.2727	1.0965
ulysses16	best	73.9998	74.6287	73.9876
	mean	74.4481	76.0864	74.0779
	st.dev.	0.4105	1.1062	0.1626
	time/s	2.9975	11.3038	1.2223
ulysses22	best	75.6525	76.1971	75.3097
	mean	76.3843	78.9508	76.1147
	st.dev.	0.5118	1.5545	0.8561
	time/s	7.0002	21.6290	1.5883
att48	best	3.5266e+4	3.7015e+4	3.3724e+4
	mean	3.9667e+4	3.8449e+4	3.4872e+4
	st.dev.	2.7453e+3	862.4546	668.7553
	time/s	14.7605	102.4784	3.0462
eil51	best	455.0951	465.2753	432.0332
	mean	481.2016	501.2494	451.1813
	st.dev.	15.5326	18.0181	9.6923
	time/s	134.5754	114.3018	3.2173
berlin52	best	8.1864e+3	8.2404e+3	7.5444e+3
	mean	8.9838e+3	8.7776e+3	8.2472e+3
	st.dev.	380.1004	267.1124	273.4509
	time/s	139.8399	118.0948	3.3438

To be more specific, the details of some benchmark problems are discussed separately in the following:

ulysses16.tsp: The best of the results are obtained by STA, with the sequence of {7, 6, 14, 13, 12, 16, 1, 3, 2, 4, 8, 15, 5, 11, 9, 10}, which gets the best length of route at 73.9876, and the best route is plotted in Fig.1. To be more careful, we can find that the mean solution gained by STA is even better than the best of ACO, which indicates the strong search capability of STA. The st.dev. of STA is almost approaching zero, and it shows that STA is also stable for this test problem. By the way, the STA consumes the half time of SA, and 1/9 time of ACO. The curves of the average fitness are illustrated following in Fig.2. The data created by SA are condensed to the same iterations (the same method is applied to other three results obtained by SA). It is interesting to find that the ups and downs in the curve of SA at the early stage, because SA accepts a relatively worse solution in probability. Taking the condensed data of SA into consideration, we can perceive that SA needs quite a long time for its steadily descending trend. However for ACO, after quickly to reach a good fitness, it is trapped into a stagnation point. But for STA, neither of the phenomena occurs, and it keeps decreasing before not a short time.

att48.tsp: STA also acheives the best, with the sequence of {9, 40, 15, 12, 11, 23, 3, 22, 16, 41, 34, 48, 5, 29, 2, 42, 26, 4, 35, 45, 10, 24, 32, 39, 25, 14, 13, 21, 47, 20, 33, 46, 36, 30, 43, 17, 27, 19, 37 , 6, 28, 7, 18, 44, 31, 38, 8, 1} and length of route at 3.3724e+4, which is illustrated in Fig.3. The same situation is observed in the results between ACO and STA, that is, the mean solution gained by STA is better than the best of ACO. SA outperforms ACO in the best, but mean in ACO is better than that of SA. In this case, the st.dev. of STA is not very satisfactory, although it is better than its competitor. The time consumed by STA is much shorter than other two algorithms, only around 1/4 and 1/30 time of that of SA and ACO, respectively. From Fig.4, we can find low degree of ups and downs in the curve of SA, and SA keeps good descending trend at the first stage. ACO is confronted the premature convergence again, few changes happen in the late process. For STA, the fitness decreases sharply at the early stage, but can still keep decreasing in the later, which indicates the excellent performance of the designed operators in the discrete STA.

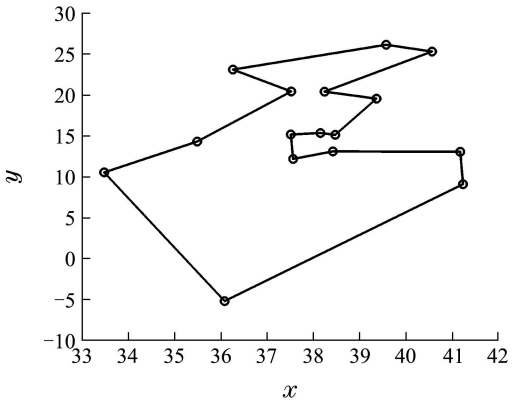

Fig. 1 The best route of ulysses16.tsp obtained by STA

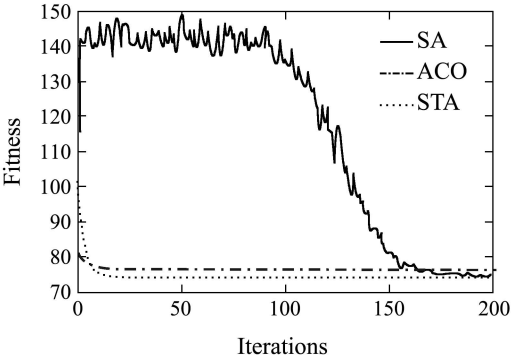

Fig. 2 Curves of the average fitness for ulysses16.tsp

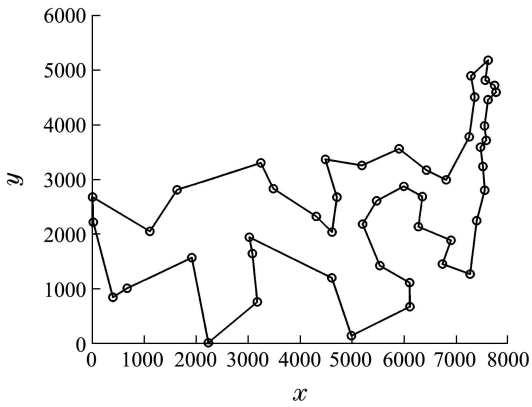

Fig. 3 The best route of att48.tsp obtained by STA

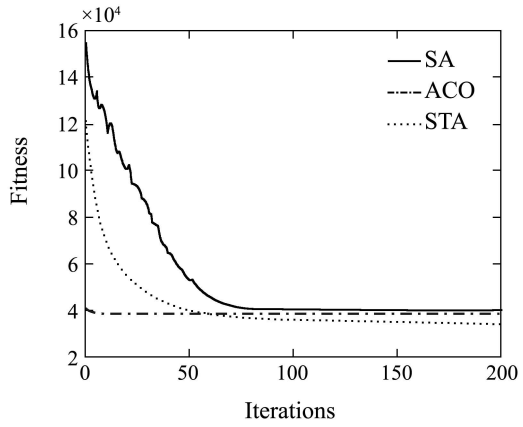

Fig. 4 Curves of the average fitness for att48.tsp

eil51.tsp: The best results are obtained by STA, with the sequence of {32, 11, 38, 5, 37, 17, 4, 18, 47, 12, 46, 51, 27 , 6, 48, 23, 7, 43, 24, 14, 25, 13, 41, 40, 19, 42, 44, 15, 45, 33, 39, 10, 49, 30, 34, 21, 50, 9, 16, 2, 29, 20, 35, 36, 3, 28, 31, 26, 8, 22, 1}, and the corresponding length of route is 432.0332, which can be found in Fig.5. We can see that, in this case, the statistical performance of SA is a little better that that of ACO, but the average time consumed is in a opposite way. Anyway, the time cost by STA is extraordinary short, when compared with SA and ACO, just around 1/40 and 1/35 time of them, respectively. In Fig.6, it is also observed that the curve of SA floats quite a long time before its rapid descending. In this case, the curve in STA is similar to that in the att48 instance, with a continued decline.

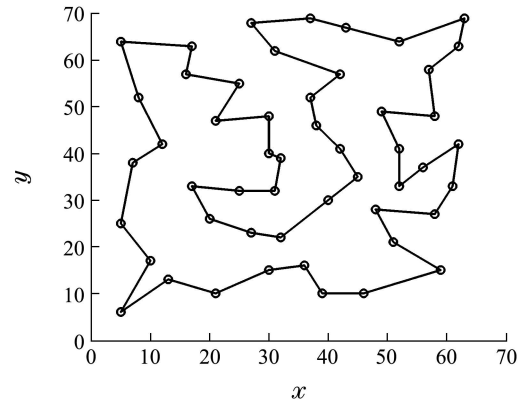

Fig. 5 The best route of eil51.tsp obtained by STA

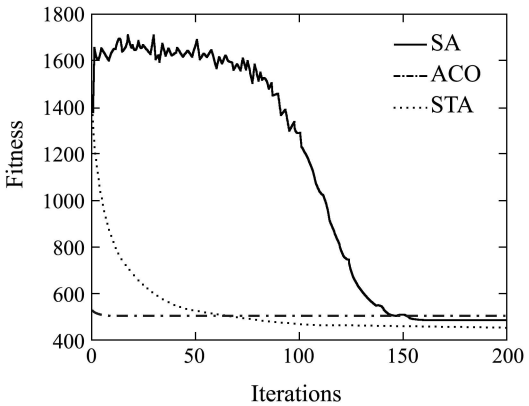

Fig. 6 Curves of the average fitness for eil51.tsp

berlin52.tsp: The STA wins the best again, with the best sequence of {3, 17, 21, 42, 7, 2, 30, 23, 20, 50, 29, 16, 46, 44, 34, 35, 36, 39, 40, 37, 38, 48, 24, 5, 15, 6, 4, 25, 12, 28, 27 26, 47, 13, 14, 52, 11, 51, 33, 43, 10, 9, 8, 41, 1945, 32, 49, 1, 22, 31, 18} and length of route at 7.5444e+3, which can be observed in Fig.7. At this time, ACO exhibits much better than SA in other performance except for the best. But, STA achieves the best results on the whole, especially for the computational time. For the problem, STA consumes 1/40 and 1/35 of the time cost by SA and ACO, respectively. As for TSP, the time complexity is really important, so the results gained by STA shows again that the discrete STA is really promising. In Fig.8, we can find that SA need quite a long time to reach a relatively good fitness, and then it becomes stagnated. On the contrary, the fitness of STA can decease quickly and keep descending till the end of the process.

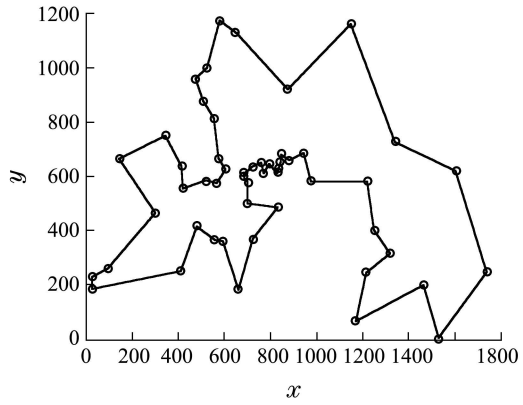

Fig. 7 The best route of berlin52.tsp obtained by STA

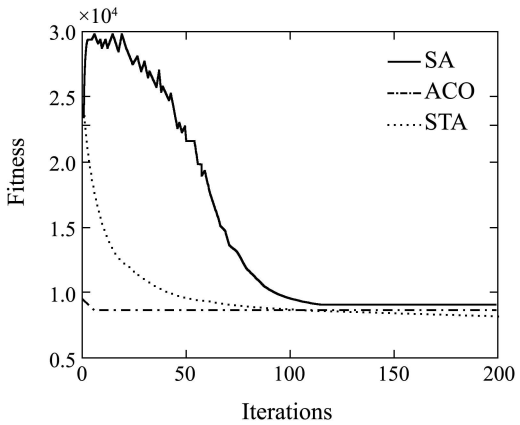

Fig. 8 Curves of the average fitness for berlin52.tsp

6 Conclusion

Different from continuous search space, the space for traveling salesman problem is discrete, which corresponds to a permutation of a sequence. In discrete version of state transition algorithm, three special state transition operators are designed to manipulate the permutation. First, we investigate a parametric study of the discrete STA. With the gained parameters, then some experiments are done to evaluate the proposed algorithm, and the results show that the discrete version of STA has much better performance not only in the search ability but also in the time consuming.

Premature phenomenon is extensively existing in heuristic algorithms. To escape from a stagnation point, this paper focuses on the designing of operators. Accepting a relatively bad solution is a good idea, as can be seen in SA; however, it increases the computational time and risks the non-convergence. It is really excited to see the fantastic performance of discrete STA only with 'greedy criterion', due to the excellent operators designed. On the other hand, difficulties are still existing when confronted with large size problems, so other strategies are necessary to be introduced to improve its performance. By the way, in current version of STA, three transformations are only for local permutation, and effective global permutation need to be found in our future work as well as the equilibrium between them.

References:

[1] STUETZLE T. The traveling salesman problem:state of the art [R]. *TUD–SAP AG Workshop on Vehicle Routing*, TUD–SAP, 2003.

[2] LAPORTE G. A concise guide to the traveling salesman problem [J]. *Journal of the Operational Research Society*, 2010, 61(1): 35 – 40.

[3] REINELT G. TSPLIB–A traveling salesman problem library [J]. *ORSA Journal on Computing*, 1991, 3(4): 376 – 384.

[4] PAPADIMITRIOU C H, STEGLITZ K. *Combinatorial Optimization: Algorithms and Complexity* [M]. New York: Dover Publications, Inc, 1982.

[5] AHMED Z H. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator [J]. *International Journal of Biometrics & Bioinformatics*, 2010, 3(6): 96 – 105.

[6] KIRKPATRICK S, GELATT C D, VECCHI M P. Optimization by simulated annealing [J]. *Science*, 1983, (220): 671 – 680.

[7] KNOX J. Tabu search performance on the symmetric traveling salesman problem [J]. *Computers & Operational Research*, 1994, 21(8): 867 – 876.

[8] YANG J H, WUA C G, LEE H P, et al. Solving traveling salesman problems using generalized chromosome genetic algorithm [J]. *Progress in Natural Science*, 2008, 18(7): 887 – 892.

[9] DORIGO M, GAMBARDELLA L M. Ant colony system: a cooperative learning approach to the traveling salesman problem [J]. *IEEE Transactions on Evolutionary Computation*, 1997, 1(1): 53 – 66.

[10] SHI X H, LIANG Y C, LEE H P, et al. Particle swarm optimization-based algorithms for TSP and generalized TSP [J]. *Information Processing Letters*, 2007, 103(5): 169 – 176.

[11] ZHOU X J, YANG C H, GUI W H. Initial version of state transition algorithm [C] *//International Conference on Digital Manufacturing and Automation (ICDMA)*. Zhangjiajie: IEEE, 2011: 644 – 647.

[12] ZHOU X J, YANG C H, GUI W H. A new transformation into state transition algorithm for finding the global minimum [C] *//International Conference on Intelligent Control and Information Processing(ICICIP)*. Harbin: IEEE, 2011: 674 – 678.

[13] ZHOU X J, YANG C H, GUI W H. State transition algorithm [J]. *Journal of Industrial and Management Optimization*, 2012, 8(4): 1039 – 1056.

作者简介:

阳春华 (1965–), 女, 教授, 博士生导师, 主要研究方向为复杂工业过程建模与优化控制、智能自动化控制系统与装置、智能信息处理技术, E-mail: ychh@csu.edu.cn;

唐小林 (1989–), 女, 博士研究生, 目前研究方向为进化计算、多目标优化及其应用, E-mail: xiaolin5789@126.com;

周晓君 (1986–), 男, 博士研究生, 主要研究方向为复杂工业过程建模、优化与控制、对偶理论、优化算法及其应用, E-mail: tiezhongyu2010@gmail.com;

桂卫华 (1950–), 男, 教授, 博士生导师, 主要研究方向为工业大系统递阶和分散控制理论及应用、复杂生产过程建模与控制, E-mail: gwh@csu.edu.cn.