A Discrete State Transition Algorithm for Generalized Traveling Salesman Problem

Xiaolin Tang, Chunhua Yang, Xiaojun Zhou, and Weihua Gui

Abstract Generalized traveling salesman problem (GTSP) is an extension of classical traveling salesman problem (TSP), which is a combinatorial optimization problem and an NP-hard problem. In this paper, an efficient discrete state transition algorithm (DSTA) for GTSP is proposed, where a new local search operator named *K-circle*, directed by neighborhood information in space, has been introduced to DSTA to shrink search space and strengthen search ability. A novel robust update mechanism, restore in probability and risk in probability (Double R-Probability), is used in our work to escape from local minima. The proposed algorithm is tested on a set of GTSP instances. Compared with other heuristics, experimental results have demonstrated the effectiveness and strong adaptability of DSTA and also show that DSTA has better search ability than its competitors.

Keywords Generalized traveling salesman problem • Discrete state transition algorithm • K-circle • Double R-probability

1 Introduction

Generally speaking, GTSP can be described as follows: given a completely undirected graph $G = \{V, E\}$, where V is a set of n vertices and has been partitioned into m clusters $V = \{V_1, V_2, \cdots, V_m\}$, E is a set of m edges, and the goal of GTSP is to find a tour visiting each cluster exactly once while minimizing the sum of the route costs. In this paper, the symmetric GTSP is concerned, that is to say, $c_{i,j} = c_{j,i}$, here, the associated cost $c_{i,j}$ for each pair of vertices (i, j) represents the distance from one vertex in V_i to another vertex in V_j. Since each cluster has at least one vertex and each vertex can only belong to one cluster, we have $m \leq n$. If $m = n$, GTSP is restored to TSP and both of them are NP-hard problems [1]. The sole task of dealing with TSP is to optimize the sequence of the clusters. While in the process

X. Tang • C. Yang (✉) • X. Zhou • W. Gui
School of Information Science and Engineering, Central South University, Changsha 410083, People's Republic of China
e-mail: xiaolin5789@126.com; ychh@csu.edu.cn; tiezhongyu2005@126.com; gwh@csu.edu.cn

D. Gao et al. (eds.), *Advances in Global Optimization*, Springer Proceedings in Mathematics & Statistics 95, DOI 10.1007/978-3-319-08377-3_15

of solving GTSP, it requires determining the sequence of the clusters and a vertex to be visited in each cluster simultaneously, which indicates that GTSP is more complex than TSP. Nonetheless, GTSP is extensively used in many applications, such as task scheduling, airport selection, and postal routing, etc [2, 3].

According to the characteristics of GTSP, the process of solving GTSP can be decomposed into two phases. One is to determine the visiting order of all the clusters, which is similar to TSP; the other is to find the optimal vertex in each cluster in a given order. Many reputed heuristic searching algorithms, like genetic algorithm (GA) [4], particle swarm optimization (PSO) [5], simulated annealing (SA) [6], ant colony optimization (ACO) [7], have been varied into discrete versions to solve GTSP. Though these algorithms have their own mechanisms to deal with continuous optimization problems, they have to adapt themselves to GTSP with some classic operators, such as *swap* and *insert*. These search operators change the visiting order of clusters in particular ways. Lin–Kernighan (L–K) is a well-known method to solve TSP and GTSP [8], which focuses on changing the edges instead of the visiting order of clusters. The number of edges that L–K impacts in a single operation is unknown; as result, the depth of L–K is usually limited within a constant [9]. The majority of these methods focus on finding an optimal sequence of clusters, while to solve GTSP, it still has to choose a vertex from each cluster to make the minimal cost simultaneously. This is a well-known shortest path problem in operations research which is also called cluster optimization (CO) in GTSP. The most common method to deal with this problem is dynamic programming that can give us a definitively best result which is named as layer network method in other literatures [10].

State transition algorithm is a new optimization algorithm, according to the control theory and state transition [11]. The efficiency of STA in application to continuous optimization problems has been proved [12]. In [13], discrete version of state transition algorithm has been introduced to solve a series of discrete optimization problems such as TSP and boolean integer programming. In this study, we will extend DSTA to solve the GTSP.

In Sect. 2, we give a brief description of DSTA and some transformation operators. Section 3 introduces relevancy and correlation index to describe K-Neighbor. In Sect. 4, a DSTA is presented to solve GTSP with a new updating mechanism. Some experimental results are given in Sect. 5, and the final part is the conclusion.

2 Discrete State Transition Algorithm

2.1 Description of DSTA

State transition algorithm comes from control theory. It regards a solution to an optimization problem as a state and updating of the solution as state transition. The unified form of discrete state transition algorithm is given as follows:

$$\begin{cases} \boldsymbol{x}_{k+1} = A_k(\boldsymbol{x}_k) \bigoplus B_k(\boldsymbol{u}_k) \\ y_{k+1} = f(\boldsymbol{x}_{k+1}) \end{cases}, \tag{1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n$ denotes a current state, corresponding to current solution of an optimization problem; $\boldsymbol{u}_k \in \mathbb{R}^n$ is a function of $\boldsymbol{x}_k$ and historical states; both A_k, $B_k \in \mathbb{R}^{n\times n}$ are transition operators which are usually state transition matrixes; $\bigoplus$ is an operation, which is admissible to operate on two states; f is the cost function or evaluation function.

In general, the solution to discrete optimization problem is a sequence, which means a new state $\boldsymbol{x}_{k+1}$ should also be a sequence after transformation by A_k or B_k. For the TSP, only a state transition matrix is considered, avoiding the complexity of adding one sequence to another. So the form of DSTA for TSP is simplified as follows:

$$\begin{cases} \boldsymbol{x}_{k+1} = G_k \boldsymbol{x}_k \\ y_{k+1} = f(\boldsymbol{x}_{k+1}) \end{cases} \tag{2}$$

where $\boldsymbol{x}_k = [x_{1,k}, x_{2,k}, \cdots, x_{m,k}]^T, x_{i,k} \in \{1, 2, \cdots, m\}$; G_k is the state transition matrix which is created by transformation operators. State transition matrixes are variants of identity matrix with only position value 1 in each column and each row. Multiplying a state transition matrix by a current state will get a new state which is still a sequence and the process is like this:

Current state $\boldsymbol{x}_k$: $[1\ 2\ 3\ 4\ 5]^{\mathrm{T}}$
State transition matrix G_k:

$$\begin{pmatrix} 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&0&0&1 \\ 0&0&0&1&0 \\ 0&0&1&0&0 \end{pmatrix} \quad \begin{pmatrix} 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&0&0&1 \\ 0&0&0&1&0 \\ 0&0&1&0&0 \end{pmatrix} \quad \times \quad \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

New state:$[1\ 2\ 5\ 4\ 3]^{\mathrm{T}} \leftarrow G_k \times \boldsymbol{x}_k$

2.2 State Transformation Operators

DSTA solves TSP with three efficient operators, which is the foundation of study on GTSP. All of the three operators, swap, shift, symmetry, belong to G_k .

(1) Swap

$$(x_1, x_2, x_3, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_{m-1}, x_m)$$

$$\rightarrow (x_1, x_i, x_3, \cdots, x_{i-1}, x_2, x_{i+1}, \cdots, x_{m-1}, x_m)$$

This is an operator to exchange several vertices in the tour and the number of the vertices to be changed is limited by a parameter m_a. With this operator, the number of edges to be changed is twice as that of vertices to be exchanged.

(2) Shift

$$(x_1, x_2, x_3, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_{m-1}, x_m)$$

$$\rightarrow (x_1, x_3, \cdots, x_{i-1}, x_i, x_2, x_{i+1}, \cdots, x_{m-1}, x_m)$$

This operator first removes a segment of sequence from a given tour and then inserts this segment into a random position of the remaining sequence. The length of the removed sequence is restricted to less than m_b. Three edges will be changed through this operator, of which two edges are adjacent and the last edge is non-adjacent to them.

(3) Symmetry

$$(x_1, x_2, \cdots, x_{i-3}, x_{i-2}, x_{x-1}, x_i, x_{i+1}, x_{x+2}, x_{x+3} \cdots, x_m)$$

$$\rightarrow (x_1, x_2, \cdots, x_{i-3}, x_{i+2}, x_{x+1}, x_i, x_{i-1}, x_{x-2}, x_{x+3} \cdots, x_m)$$

Symmetry is to choose a vertex in a given tour as center, and then mirror a small segment on the left side of the center to the opposite side and so does a small segment on the right side. The length of these small segments is restricted to m_c. For symmetric GTSP, the symmetry operator can change two edges every time.

(4) Circle

At the final stage of search process, a tour is usually locally optimal, which indicates that this tour has many similar segments to the global tour; therefore, the entire sequence can be regarded as the combination of these segments. To further optimize a tour, we have to change several conjoint vertices in some segments simultaneously. Considering that the number of changing vertices in swap or shift is no more than 2, a new operator called circle is proposed here to enhance the global search ability.

Circle consists of two steps. First, we divide a given tour into two circles randomly, and then break one of the circles and insert it into another to create a new complete tour. Effects of this operator can be summarized as follows:

1. One of the circles contains only one vertex (Fig. 1a);
2. Both of the circles contain more than one vertex, and change the connection at the interfaces of each circle (Fig. 1b);
3. Both of the circles contain more than one vertex, break one of the circles at its interface and insert it into another from a random position except the interface (Fig. 1c, d).

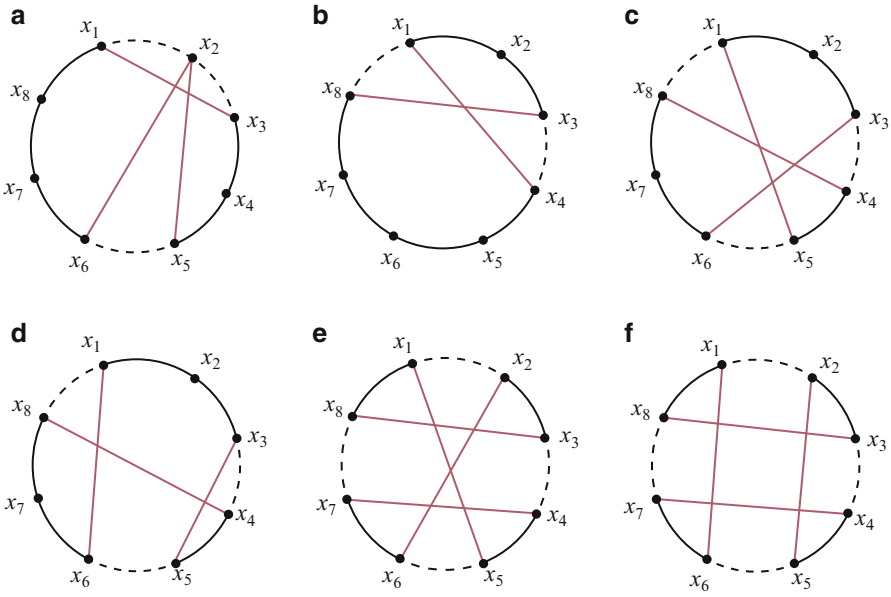

Fig. 1 Effects of circle operator in different cases

4. Both of the circles contain more than one vertex, break one of the circles and insert it into another. Both of the breaking position and inserting position are randomly chosen except two interfaces. Figure 1e, f show the results of this case.

   Obviously, circle is much more flexible than the other operators since it can gain six different kinds of cases.

(5) Cluster optimization (CO)

This is a sole operator to find the best path of given visiting order of clusters. A tour $[x_{1,k}, x_{2,k}, \cdots, x_{m,k}]$ with costs $W(\boldsymbol{x}_k)$ will be optimized into a new tour $\boldsymbol{x}_k^{'}$ after running CO, here, $W(\boldsymbol{x}_k^{'}) \leq W(\boldsymbol{x}_k)$ and $cluster(\boldsymbol{x}_k^{'}) = cluster(\boldsymbol{x}_k)$. In general, few of visiting orders of clusters will be changed from $\boldsymbol{x}_k$ to $\boldsymbol{x}_k^{'}$, thus we only need to optimize a small segment around the changed clusters.

3 K-Neighbor

To improve the global search ability for large-scale problems in limited computational time, it is necessary to avoid some potential bad search space. In this paper, the correlation index and relevancy are proposed, where correlation index is used to assess the correlation of every two clusters and the relevancy is applied to define K-Neighbor which will guide search direction as heuristic information.

Definition 1: *Let define the distance between the geometric centers of cluster i and cluster j as $d_{i,j}$, the sum of distance from the geometric centers of cluster i to the geometric centers of other clusters as d_i and denote $r_{i,j}$ as the correlation index of cluster i to cluster j:*

$$r_{i,j} = \frac{1 - \frac{d_{i,j}}{d_i}}{n-1}, \tag{3}$$

$$\sum_{j=1}^{n} r_{i,j} = 1, d_i = \sum_{j=1}^{n} d_{i,j}.$$

Definition 2: *Given $r_{i,j}$ as the correlation index of cluster i to cluster j and $r_{j,i}$ as the correlation index of cluster j to cluster i, then the relevancy $p_{i,j}$ of cluster i to cluster j can be formulated as:*

$$p_{i,j} = \frac{r_{i,j} \times r_{j,i}}{\sum_{j=1}^{n} r_{i,j} \times r_{j,i}}. \tag{4}$$

Calculating each $p_{i,j}$, we can get a relevancy matrix. The ith row of the relevancy matrix shows the relevancy of cluster i to other clusters. A big $p_{i,j}$ indicates cluster i is with high possibility connecting with cluster j. After sorting the relevancy matrix in descending order by row, the top k clusters in row i will be the K-Neighbor of cluster i. Using K-Neighbor as heuristic information, the global search ability can be improved significantly.

4 DSTA for GTSP

To adapt DSTA to GTSP and to make the algorithm more efficient, K-Neighbor is used as heuristic information to guide the search. Thus, k-shift, k-symmetry, and k-circle which are all guided by K-Neighbor are included in DSTA. The core procedure of the DSTA for GTSP can be outlined in pseudocode as follows:

1: **repeat**
2: $[Best, Best^*] \leftarrow$ swap($SE, Best, Best^*, m_a$)
3: $[Best, Best^*] \leftarrow$ shift($SE, Best, Best^*, m_b$)
4: $[Best, Best^*] \leftarrow$ k-circle($SE, Best, Best^*, K\text{-}Neighbor$)
5: $[Best, Best^*] \leftarrow$ k-symmetry($SE, Best, Best^*, K\text{-}Neighbor$)
6: $[Best, Best^*] \leftarrow$ k-shift($SE, Best, Best^*, K\text{-}Neighbor$)
7: **until** the specified termination criterion is met

where SE is the search enforcement, representing the times of transformation by a certain operator; $Best$ is the best solution from the candidate state set created

by transformation operators; $Best^*$ is the best solution in history. There are five operators in DSTA to optimize the sequence of clusters. A short cluster optimization which only optimizes a small segment (no more than five vertices) around the changed vertices is contained in every transformation operator to find a minimum path of a given sequence in further. To escape from local minima, a new robust update mechanism, restore and risk in probability, called double R-Probability for short, is introduced. Risk in probability is to accept a bad solution with a probability p_1. To ensure the convergence of DSTA, restore in probability p_2 is designed to recover the best solution in history.

5 Computational Results

Instances used in this paper all come from GTSPLIB [14]. The number of clusters in these instances varies from 30 to 89. Algorithms including DSTA, SA and ACO are coded in matlab and run on an Intel Core i5 3.10 GHz under Window XP environment. In order to test the performance of the proposed operators and approach, DSTA is compared with SA and ACO, and ten runs are carried out for the experiment. Some statistics are used as follows:

Opt. : the best known solution,
Best: the best solution obtained from the experiment,
Δ_{avg}: the relative error of the average solution,

$$\Delta_{avg} = \frac{mean(values) - Opt.}{Opt.} \times 100\,\%$$

t_{avg}: average time consumed.

Results of comparison among DSTA, SA, and ACO are listed in Table 1. In DSTA, we set $k = 8$, $m_a = 2, m_b = 1$. The initial temperature in SA is 5,000 and the cooling rate is 0.97. In ACO, $\alpha = 1$, $\beta = 5$, $\rho = 0.95$, where α, β are used to control the relative weight of pheromone trail and heuristic value, and ρ is the pheromone trail decay coefficient. As can be seen from Table 1, DSTA is superior to SA and ACO in both time consumption and solution quality. The Δ_{avg} of DSTA is very small, which indicates DSTA has good robustness and can obtain good solutions with high probability. In the ten runs, DSTA obtains the optimal solution at almost each run for every instance, but SA and ACO seldom find the Opt. except for 30kroB150. Δ_{avg} of SA is smaller than that of ACO because SA accepts a bad solution with probability which can help it escape from local minima.

Table 1 Results of comparison for SA, ACO, and DSTA

Instance	Opt.	SA			ACO			DSTA		
		Best	Δ_{avg}	t_{avg}	Best	Δ_{avg}	t_{avg}	Best	Δ_{avg}	t_{avg}
30kroA150	11,018	11,027	0.16	152	11,331	5.99	104	11,018	0	13
30kroB150	12,196	12,196	0.02	78	12,532	6.02	67	12,196	0.18	18
31pr152	51,576	51,584	1.12	79	51,734	1.60	69	51,576	0	25
32u159	22,664	22,916	1.90	89	24,285	8.68	75	22,664	0.74	33
39rat195	854	857	1.09	198	884	5.86	145	854	0.05	56
40d198	10,557	10,574	0.53	112	11,458	9.31	103	10,557	0.06	74
40kroA200	13,406	13,454	0.62	107	14,687	10.77	99	13,406	1.10	69
40kroB200	13,111	13,117	0.38	108	13,396	8.34	99	13,111	0.20	28
45ts225	68,340	68,401	1.57	325	70,961	5.83	223	68,340	0.66	72
45tsp225	1,612	1,618	1.77	122	1,736	8.15	119	1,612	1.35	88
46pr226	64,007	64,062	2.70	130	66,458	7.51	124	64,007	0	35
53gil262	1,013	1,047	5.24	142	1,148	15.92	148	1,013	1.30	85
53pr264	29,549	29,725	1.87	146	32,388	12.06	150	29,546	0.07	58
60pr299	22,615	23,186	7.00	165	25,296	15.97	184	22,618	2.54	114
64lin318	20,765	21,528	5.73	166	23,365	13.57	199	20,769	2.62	117
80rd400	6,361	6,920	10.36	225	8,036	21.67	299	6,361	2.52	141
84fl417	9,651	10,099	9.95	282	10,122	10.14	345	9,651	0.51	158
88pr439	60,099	66,480	13.13	276	69,271	16.14	368	60,099	2.95	156
89pcb442	21,657	23,811	11.15	253	26,233	19.48	376	21,664	3.80	163

6 Conclusions

We added a new operator and heuristic information to DSTA to solve GTSP. K-Neighbor can guide the search direction, in a way to ignore all possible connections among vertices. A flexible operator k-circle is guided by the K-Neighbor, which can change random segments freely in a tour. Double R-Possibility is helpful to escape from local minima. It accepts a bad solution with a probability p_1 and restore the history best with another probability p_2. All these strategies contribute to improving the performance of the DSTA.

Acknowledgements The work is supported by the National Science Found for Distinguished Young Scholars of China (Grant No. 61025015), Key Project of National Natural Science Funds (Grant No. 61134006), Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Grant No.61321003).

References

1. Gutin, G., Yeo, A.: Assignment problem based algorithms are impractical for the generalized TSP. Australas. J. Comb. **27**, 149–153 (2003)
2. Fischetti, M., Salazar Gonzlez, J.J., Toth, P.: The symmetric generalized travelling salesman polytope. Networks **26**, 113–123 (1995)
3. Fischetti, M., Salazar, G., J.J., Toth, P.: A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Oper. Res. **45**, 378–394 (1995)
4. Gutin, G., Karapetyan, D.: A memetic algorithm for the generalized travelling salesman problem. Nat. Comput. **9**, 47–60 (2010)
5. Tasgetiren, M.F., Suganthan, P.N.: A discrete particle swarm optimization algorithm for the generalized traveling salesman problem, networks. In: 9th Annual Conference on Genetic and Evolutionary Computation, pp. 158–167 (2007)
6. Skiscim, C.C., Golden, B.L.: Optimization by simulated annealing: a preliminary computational study for the TSP. In: 15th Conference on Winter Simulation, pp. 523–535. IEEE Press, Piscataway (1983)
7. Song, X., Li, B., Yang, H.: Improved ant colony algorithm and its applications in TSP. In: 6th International Conference on Intelligent Systems Design and Applications, pp. 1145–1148, IEEE Computer Society, Washington (2006)
8. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the Traveling-Salesman Problem. Oper. Res. **21**(2), 498–516 (1973)
9. Karapetyan, D., Gutin, G.: Lin-Kernighan heuristic adaptations for the generalized traveling salesman problem. Eur. J. Oper. Res. **208**, 221–232 (2011)
10. Bondou, B., Artigues, C., Feillet, D.: A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem. Comput. Oper. Res. **37**, 1844–1852 (2010)
11. Zhou, X.J., Yang, C.H., Gui, W.H.: Initial version of state transition algorithm. In: International Conference on Digital Manufacturing and Automation (ICDMA), pp. 644–647 (2011)
12. Zhou, X.J., Yang, C.H., Gui, W.H.: State transition algorithm. J. Ind. Manag. Optim. **8**(4), 1039–1056 (2012)
13. Zhou, X.J., Gao, D.Y., Yang, C.H.: Discrete state transition algorithm for unconstrained integer optimization problems. arXiv:1209.4199 [math.OC] (2012)
14. GTSP Instances Library. http://www.cs.rhul.ac.uk/home/zvero/GTSPLIB