



GORTS: genetic algorithm based on one-by-one revision of two sides for dynamic travelling salesman problems

Xiaolong Xu¹ · Hao Yuan² · Peter Matthew³ · Jeffrey Ray³ · Ovidiu Bagdasar⁴ · Marcello Trovati³

© The Author(s) 2019

Abstract

The dynamic travelling salesman problem (DTSP) is a natural extension of the standard travelling salesman problem, and it has attracted significant interest in recent years due to its practical applications. In this article, we propose an efficient solution for DTSP, based on a genetic algorithm (GA), and on the one-by-one revision of two sides (GORTS). More specifically, GORTS combines the global search ability of GA with the fast convergence feature of the method of one-by-one revision of two sides, in order to find the optimal solution in a short time. An experimental platform was designed to evaluate the performance of GORTS with TSPLIB. The experimental results show that the efficiency of GORTS compares favourably against other popular heuristic algorithms for DTSP. In particular, a prototype logistics system based on GORTS for a supermarket with an online map was designed and implemented. It was shown that this can provide optimised goods distribution routes for delivery staff, while considering real-time traffic information.

Keywords DTSP · Path optimisation · One-by-one revision of two slides · Genetic algorithm

Communicated by V. Loia.

✉ Marcello Trovati
trovatim@edgehill.ac.uk

Xiaolong Xu
xuxl@njupt.edu.cn

Hao Yuan
1214043029@njupt.edu.cn

Peter Matthew
matthewp@edgehill.ac.uk

Jeffrey Ray
Rayj@edgehill.ac.uk

Ovidiu Bagdasar
o.bagdasar@derby.ac.uk

¹ Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, Nanjing University of Posts and Telecommunications, Nanjing, China

² School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China

³ Department of Computer Science, Edge Hill University, Ormskirk, UK

⁴ School of Electronics, Computing and Mathematics, University of Derby, Derby, UK

1 Introduction

The *Travelling Salesman Problem* (TSP), a classical example of NP (non-deterministic polynomial) problem, was first investigated in more detail in the 1950s (Garey and Johnson 1979). The classic TSP aims to determine the shortest path across a set of randomly located cities. (Each city is visited once and only once, except for the starting point.) In other words, TSP is a shortest route planning, whose solution is a minimum Hamiltonian circuit. However, many practical applications exhibit a dynamic behaviour, such as transportation planning (Niendorf and Kabamba 2015; Niendorf et al. 2016), communication design (Boryczka and Strak 2012), workload distribution (Mavrovouniotis and Yang 2013), IoT analysis and optimisation (Bessis et al. 2012; Xu et al. 2013), as well as disaster management (Reina et al. 2014). Hence, such TSP models need to be adaptive, and often real-time, in order to obtain the most appropriate description.

The *Dynamic Travelling Salesman Problem* (DTSP) is an extension of TSP, which has the following supplementary features (Kang et al. 2004; Li et al. 2006):

- *Real-time* The weights of the links between nodes may change probabilistically with time;

- *Robustness* Allows for unexpected situations which require timely response (e.g. nodes may randomly join or quit the system);
- *Efficiency* DSTP requires an optimal or sub-optimum solution in a finite time.

Typical algorithms for solving DTSP mainly include ant colony algorithms (ACA) (Mavrovouniotis and Yang 2013; Melo et al. 2013) and genetic algorithms (GA) (Falcon and Nayak 2010), which generally make appropriate adjustments based on changes in the environment, such as varying travelling cost between cities. However, in this case it is often computationally expensive to identify the optimal solution in a short time, due to the numerous sudden changes in parameter values.

In this article, we propose a new algorithm named *GORTS*, which is based on the integration of DTSP with a genetic algorithm defined by a one-by-one revision of two sides. Combining the global search ability of the GA and the fast convergence of the method of one-by-one revision of two sides, we show that this method can achieve accurate solutions in shorter time. In particular, the method of one-by-one revision of two sides is adopted to modify the multiple chromosomes initialised by the genetic algorithm, and the elitist strategy is used to select the optimal solution. Subsequently, the crossover and mutation operations identify the optimal solution.

The rest of the article is organised as follows. In Sect. 2, we discuss the current state-of-the-art approaches and methods for solving TSP and DTSP. Section 3 presents the details of the mathematical DTSP model. In Sects. 4 and 5, we introduce and discuss the *GORTS* algorithm, and in Sects. 6 and 7, the experimental results and the prototype logistics system for supermarket with *GORTS* are presented. Finally, Sect. 8 concludes the paper by summarising the main contributions of this work and commenting on future directions of investigation.

2 Related work

Significant research on the classical TSP model and versions with different constraint conditions has been carried out. This includes TSP models with time windows Li et al. (2015) and the minimum ratio TSP Cook (2011), which mainly focus on determining the minimal Hamiltonian circuit for a single travelling salesman. On the other hand, within the DTSP, the parameters related to each node may change dynamically and it can be seen as a combined series of TSPs Li (2013).

2.1 TSP algorithms

Currently, the algorithms for solving TSP can be divided into accurate and approximate algorithms. The former mainly

include the dynamic programming algorithm Mahfoudh et al. (2015), the branch and bound Ghadle and Muley (2014), and integer linear programming algorithm Singh and Mehta (2014). However, these algorithms can only address small-scale TSP, as their complexity increases exponentially with the number of nodes. The latter include heuristic algorithms such as the ant colony optimisation algorithm (ACO) Alves and Lopes (2015), the particle swarm optimisation algorithm (PSO) Avin et al. (2012), the genetic algorithm (GA) Yuan et al. (2013), which have been widely accepted for their suitability at identifying better solutions in a reasonable time. However, their efficiency tends not to match up with the abrupt rates of change present in the DTSP system parameters. Therefore, a variety of fusion algorithms have been proposed.

2.2 DTSP algorithms

Following the introduction of the DTSP with a single objective in 1988 (Mavrovouniotis et al. 2016), most of the research has mainly focused on the definition of the problem, algorithm design, performance measurement, or on the test platform construction. Melo et al. (2013) attempted to address the DTSP via the ant colony optimisation algorithm. This type of algorithm tends to quickly converge to obtain the optimal solution based on the pheromone produced by ants, where previous pheromone trajectories can accelerate the optimisation process. However, this scheme is only suitable for DTSP with small environmental changes.

If the number of nodes varies significantly, the optimisation process needs to be recalculated. Cheng and Yang (2009) proposed a genetic algorithm based on the elite immigration strategy. The best population in the previous generation produce individuals through mutation operations, who will replace the least suitable ones in the current generation. Compared with the original GA, it has better adaptability and can optimise the quality of the solution. However, if there are large-scale nodes, the algorithm often needs to iterate several times to converge and cannot meet the real-time requirement. Gharehchopogh et al. Gharehchopogh and Farahmandian (2012) proposed to combine ACO with GA to find the optimal solution, more suitable for severe variations. Nonetheless, its computational overhead is relatively high. Blackwell (2004) proposed a new mutant PSO approach, specifically designed for dynamic environments. This algorithm extends the diversity of particle swarm in a single population and can find a better solution in the case of environmental change. However, the diversity increases the cost of the algorithm and affects the usability in contexts involving real-time processes.

3 DTSP modelling

The objective of TSP is to find the shortest path across a set of randomly located cities, or in other words, to obtain the minimum Hamiltonian circuit. Regarding their different constraints and limitations, TSP models can be extended to include specific features of practical interest, such as the TSP versions with multiple salesmen, or TSP models with multiple objectives, DTSP, etc.

Let $G = G(V, E)$ be a graph, where $V = \{v_1, \dots, v_n\}$ represents the set of vertices and denote the set of edges by $E = \{d_{i,j} : d_{i,j} > 0, d_{i,i} = 0, i, j \in N\}$. In the TSP model, $i \in N \subset \{1, \dots, n\}$ represents the city number, and $d(v_i, v_j)$ the distance between cities i and j . If TSP is symmetric, then we have $d(v_i, v_j) = d(v_j, v_i)$.

The optimal solution of TSP comes in the form of a path $V = (v_1, v_2, \dots, v_n)$, minimising the value of the following objective function:

$$f(V) = \sum_{i=1}^{n-1} d(v_i, v_{i+1}) + d(v_n, v_1). \tag{1}$$

The decision variables are defined as

$$x_{i,j} = \begin{cases} 1 & \text{if } (i, j) \in L \\ 0 & \text{if } (i, j) \notin L \end{cases} \tag{2}$$

where L is the solution sequence. The model of TSP can be formulated as a linear programming in the form given below

$$\min D(L) = \sum_{i \neq j} d_{i,j} x_{i,j} \tag{3}$$

$$\sum_{i \in V, i \neq j} x_{i,j} = 1 \tag{4}$$

$$\sum_{j \in V, i \neq j} x_{i,j} = 1 \tag{5}$$

$$\sum_{i,j \in V} x_{i,j} = |L|, \quad L \subset E \tag{6}$$

The objective of Eq. 3 is to minimise the total distance. Equations 4 and 5 ensure that all the salesmen's itineraries must begin and end at the same point. Equation 6 ensures that the Hamilton circuit does not have any sub loop.

The main feature of DTSP is the time-varying matrix of road distance, which can be defined as:

$$D(t) = \{d_{ij}(t)\}_{n(t) \times n(t)}, \tag{7}$$

where t is the period of a dynamic change, $d_{ij}(t)$ is the distance between cities i and j , and $n(t)$ is number of cities at moment t .

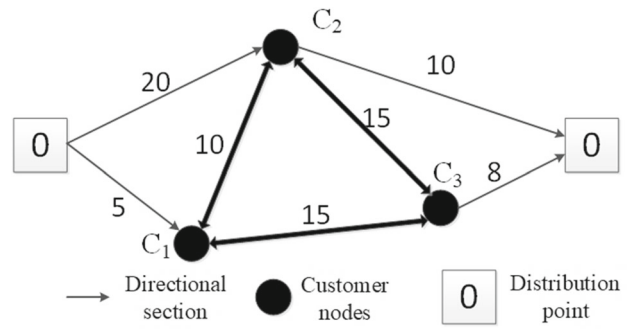


Fig. 1 Simple network

The difficulty of solving DTSP is proportional to the changes of $n(t)$ and $D(t)$, and inversely proportional to the time interval Δt of environmental change. The more significantly $n(t)$ changes, the weaker the inter-connection between TSPs in each time interval is; the higher performance of algorithm we require, and the greater the difficulty of solving the problem becomes.

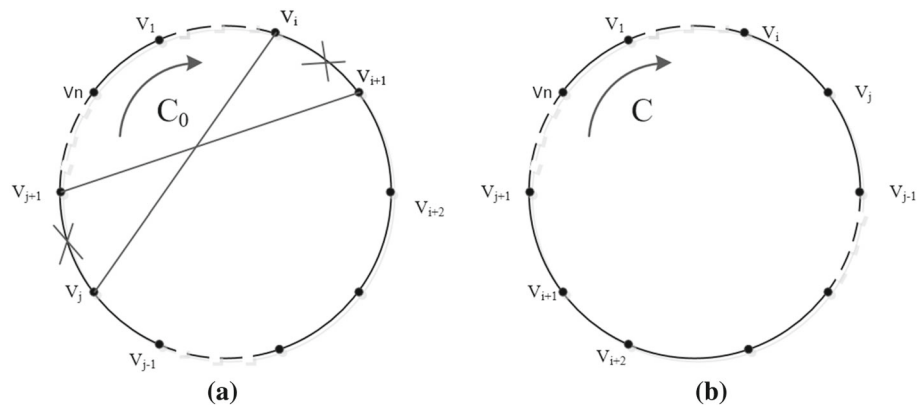
A natural question arises. How can we deal with the information changing continuously? This is addressed by defining a suitable sampling. Figure 1 shows a simple network with 3 nodes, as well as the cost of each directed edge. From the graph, we can easily get the initial optimal access order, i.e. $0 - c_1 - c_2 - c_3 - 0$, and the initial travel time is 38 min. If a salesman arrives at c_1 and a traffic congestion occurs on the way from c_1 to c_2 , this may increase the travelling time cost by 30 minutes, if the route is not changed. However, if the route gets real-time updates, the travelling time cost can be reduced significantly.

Suppose that: 1) the traffic information between cities can be known in advance, and 2) the scale of the city, the distance between cities, and other parameters are fixed during the sampling period. We define DTSP as a combination of different TSP in different sampling period, where

- $TSP(t)$ is used to represent TSP at different times;
- $d_{ij}(t)$ represents the distance between city i and city j ;
- $n(t)$ is the scale of TSP at time t ;
- $H(i)$ is the serial number of city;
- Δt is the sampling period;
- Δd_{ij} is the variation of $d_{ij}(t)$ in Δt , and Δn is the change of city number in Δt ;
- T is the time period, and
- N is the sampling times in each Δt .

The objective value of DTSP is the sum of objective values of all TSP within period T . Therefore, DTSP can be defined as

Fig. 2 a Initial H circuit b New H circuit



$$TSP(t) = \begin{cases} TSP_1 & 0 \leq t \leq \Delta t \\ TSP_2 & \Delta t < t \leq 2\Delta t \\ \dots & \\ TSP_N & (N-1)\Delta t < t \leq N\Delta t \end{cases} \quad (8)$$

$$\min d(T) = \sum_{i=1}^N \sum_{j=1}^{n(i\Delta t)} d_{H(j)H(j+1)}(i\Delta t) \quad (9)$$

$$\Delta t = \frac{T}{N} \quad (10)$$

$$\frac{\Delta d_{ij}}{\Delta t} = 0 \quad (11)$$

$$\frac{\Delta n}{\Delta t} = 0 \quad (12)$$

sides to correct chromosomes, which speeds up the process of convergence to meet the real-time requirements.

4.2 The method of one-by-one revision of two sides

The method of one-by-one revision of two sides is an approximate algorithm for obtaining the optimal Hamiltonian circuit, suitable for the case with a large number of vertices.

Suppose that $G = (V, E)$ is a connected undirected graph. The cycle, which joins each vertex of G exactly once, is called a Hamiltonian circuit of G . Suppose that $W(v_i, v_j)$ is the weight of the link between the vertices v_i and v_j , then

$$D(T) = \{d_{ij}(t)\}_{n(t) \times n(t)} \quad (13)$$

$$d_{ij} = d_{ji} = \begin{cases} W(v_i, v_j) & \text{if } (v_i, v_j) \in E \\ 0 & \text{if } i = j \end{cases} \quad (14)$$

The optimum H circuit is the cycle on G with the smallest total weight. The workflow of finding the optimum H circuit is as follows:

Step 1 Draw the initial Hamiltonian circuit randomly, as shown in Fig. 2,

$$C_0 = v_1, v_2, \dots, v_i, \dots, v_j, \dots, v_n, v_1; \quad (15)$$

Step 2 For all $i, j, 1 < i + 1 < j < n$, if

$$W(v_i, v_j) + W(v_{i+1}, v_{j+1}) < W(v_i, v_{j+1}) + W(v_j, v_{j+1}), \quad (16)$$

delete edges (v_i, v_{j+1}) and (v_j, v_{j+1}) in C_0 , and add edge (v_i, v_j) and $W(v_{i+1}, v_{j+1})$ to form a new Hamiltonian cycle, that is,

$$C = v_1, v_2, \dots, v_i, v_j, v_{j-1} \dots, v_{i+1}, v_{j+1}, \dots, v_n, v_1, \quad (17)$$

as shown in Fig. 2.

4 Description of the GORTS algorithm

4.1 Basic idea

The dynamics of DTSP is defined by the scale of the problem and the time variation of the cost matrix change. This dynamical aspect affects the performance and effectiveness of the algorithm and determines how to choose the appropriate algorithm. If the system exhibits a stable behaviour, a local optimisation algorithm is used to obtain an acceptable approximate solution. Otherwise, a global optimisation algorithm is necessary. The application of a GA often requires to iterate several times to converge to get the satisfactory solution for TSP. However, in DTSP it is generally required that the algorithm can converge to reach the optimal solution with fewer iterations. The method of one-by-one revision of two sides Yuan et al. (2013) aims to continuously optimise the solution via several iterations. Its process of calculation is fast. However, the algorithm is greatly affected by the initial solution and may fall easily into a local optimum.

In this article, we combine the method of one-by-one revision of two sides with GA to introduce the new GORTS algorithm. This algorithm inherits the global search ability of GA and uses the method of one-by-one revision of two

Fig. 3 a Complete graph b Updated initial circle c Better H circuit

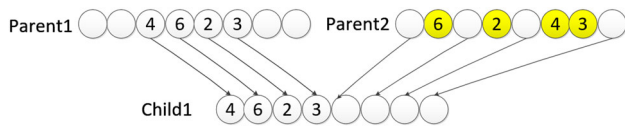
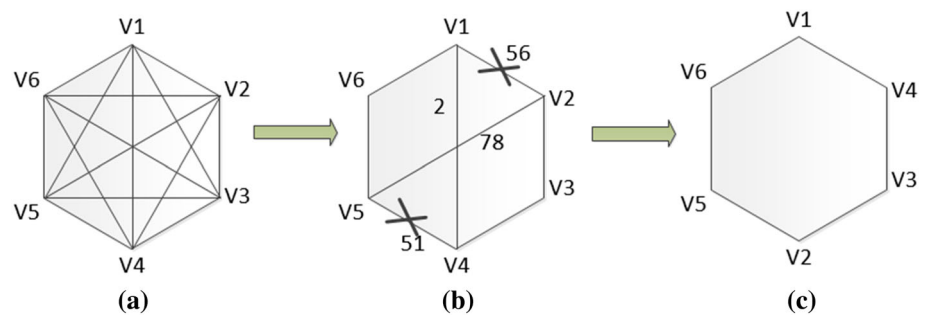


Fig. 4 Implementation process of crossover operation

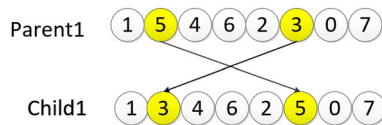


Fig. 5 Mutation operation

Step 3 Repeat the above steps until the best H circuit is achieved.

Consider the complete graph as is shown in Fig. 2. Suppose that the weight of the initial circle $C_0 = v_1, v_2, \dots, v_6, v_1$ is 237, and $w(1, 4) + w(2, 5) < w(1, 2) + w(4, 5)$, edges (v_1, v_2) and (v_4, v_5) need to be deleted to obtain a better H circuit C , that is, $C = v_1, v_4, v_3, v_2, v_5, v_6, v_1$.

5 Genetic algorithm improvement

The genetic algorithm (GA) is known for its good global search ability, high efficiency, and good scalability in solving TSP (Fig. 3). However, GA needs many iterations to obtain the optimal solution in high-dynamic systems. The integration of the GA with the method of one-by-one revision of two sides can accelerate the speed of finding the ideal solution, based on the following workflow:

- Step 1* Generate the initial population;
- Step 2* Optimise each chromosome by the method of one-by-one revision of two sides, and use the optimised individual to replace the original one;
- Step 3* Execute operations of selection, crossover, and mutation on individuals in the population;
- Step 4* Select the best individual in each generation.

Unsuitable implementations and various operations of encoding, crossover, or mutation may lead to different precision values, as well as the failure of the iterative process. Therefore, it is necessary to redesign the above operations.

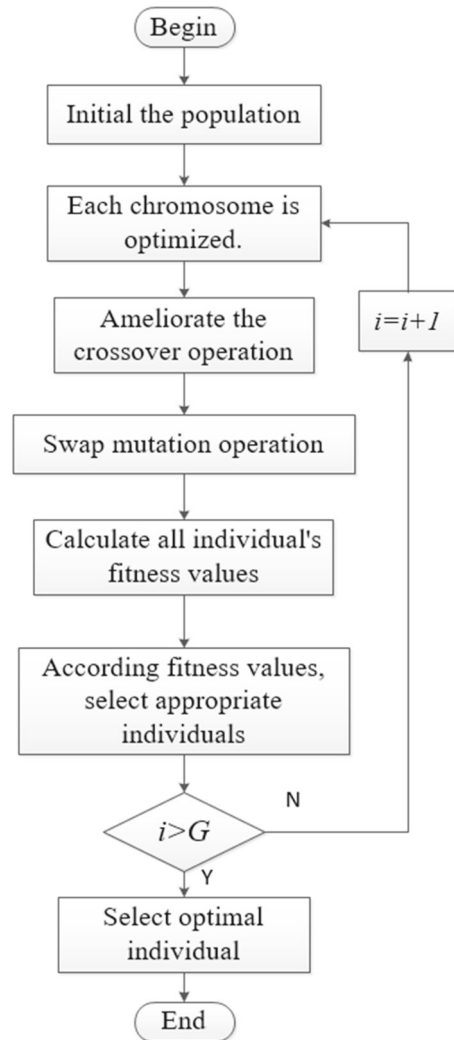


Fig. 6 The workflow of GORTS

5.1 Initial chromosome encoding

Suppose $X = (x_1, x_2, \dots, x_i, \dots, x_n)$, $1 \leq x_i \leq x_{n_i}$ where n_i is the maximum of gene i . This requires genes to be encoded with different values at different bits. The length of chromosome is determined by the scale of the DTSP. The initialisation of chromosome is automatically generated

Table 2 Performance of GORTS and other algorithms with changes of distance between cities

Problem instance	GORTS	MMAS	PACO	EIACO	EIGA-GAPX
$m \in [0, 0.25], f = 15s]$					
Berlin52.tsp	8012.5	7923.2	8055.6	7916.0	8347.5
eil101.tsp	667.6	655.7	671.4	663.2	703.4
d198.tsp	17087.7	16932.0	17679.6	17106.9	18203.2
lin318.tsp	46625.4	45182.1	48588.1	46426.1	52737.9
$m \in [0, 0.5] \text{ with } f = 15s]$					
Berlin52.tsp	8470.8	8313.2	8506.4	8327.6	8715.8
eil101.tsp	725.6	701.5	724.8	716.1	783.7
d198.tsp	18160.5	17783.6	18707.5	18134.6	19588.6
lin318.tsp	49618.4	47779.3	51670.3	48981.4	56478.4
$m \in [0, 1] \text{ with } f = 15s]$					
Berlin52.tsp	8806.2	8603.8	8856.4	8634.5	9144.2
eil101.tsp	781.3	755.2	784.8	768.1	843.2
d198.tsp	19234.6	18727.1	19774.5	19086.8	21127.5
lin318.tsp	52742.6	50976.1	54641.3	51141.6	61448.2

Table 1 Parameters and set values

Parameter	Value
F_L	0
F_U	2
G	2000
P_{size}	500
P_c	0.80
p_m	0.1

with some heuristic algorithm, and the encoding method is to directly arrange nodes randomly.

5.2 Fitness function

The fitness function is used to differentiate the individuals in a group. High value of the fitness evaluation implies that an individual has a high probability of being chosen. The selection operation based on the fitness value is important to GA, which means that the fitness function determines the performance of GA. The fitness function is defined as

$$F(x) = 1/D(x), \quad (18)$$

where x represents an individual of a population, and $D(x)$ is the evaluation value of x . Here, $D(x)$ is defined as the distance travelled by individual. In this article, we use the reciprocal of $D(x)$ as the fitness function.

5.3 Crossover operation

Two parent chromosomes, *parent 1* and *parent 2* are selected according to the crossover probability p_c , which generates

two intersection points. The segments Δp_1 and Δp_2 are subsequently selected according to the two intersection points. *Child 1* assigns Δp_1 as the initial gene, while the same parts of *parent 2*'s chromosome are ignored. Finally, the remaining component is added to *child 1*. Likewise, *child 2* is similarly generated. In this way, even if the two parents are identical, the new offspring can be generated iteratively to overcome the disadvantages of local optimisation and premature convergence.

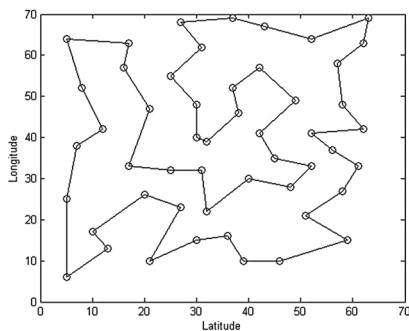
As shown in Fig. 4, assuming there are 8 cities, the integers $\{0, 1, 2, 3, 4, 5, 6, 7\}$ are randomly selected to represent the two parents chromosomes. The randomly selected gene segment (e.g. 4, 6, 2, 3) of *parent 1* is adopted as the initial gene of *child 1*, and the identical parts of *parent 2*'s chromosome are ignored. Finally, the remaining parts are added to *child 1*. *Child 2* is generated in a similar manner.

5.4 Mutation operation

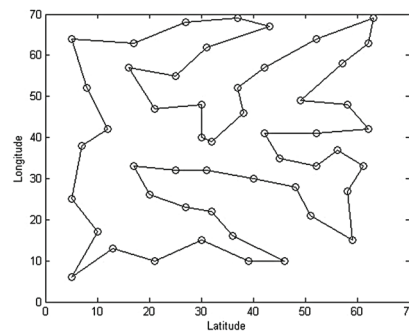
The mutation operation plays an important role in improving local search ability, maintaining variability of the population, and preventing the premature convergence of GA. The swapping mutation operation is based on the mutation probability. This will identify a mutated chromosome, and subsequently, two crossing points are randomly selected and swapped

As shown in Fig. 5, assuming again there are 8 cities, the sequence of integers corresponding to cities (1, 5, 4, 6, 2, 3, 0, 7) may represent a route scheme. Two selected swapping gene points are node pairs (3, 6), and (3, 5), which are exchanged to generate the offspring.

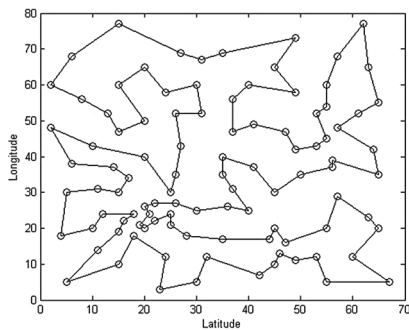
Fig. 7 Routes planned by GORTS with $G = 5$ or $G = 10$



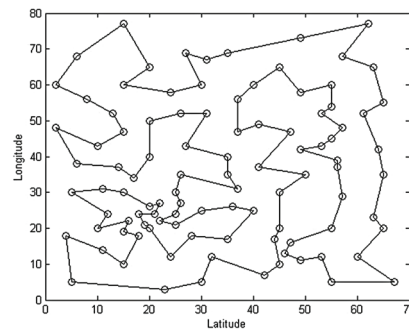
(a) $G=5$, Eil51



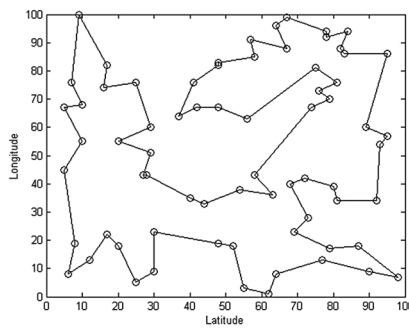
(b) $G=10$, Eil51



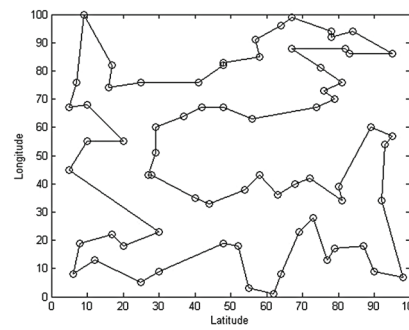
(c) $G=5$, Eil101



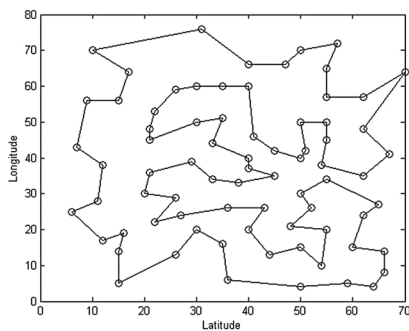
(d) $G=10$, Eil101



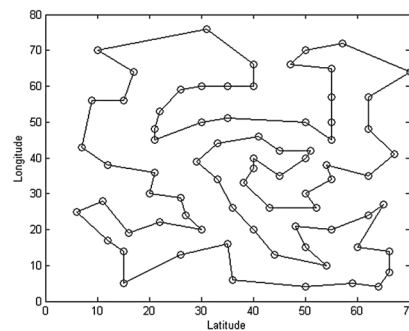
(e) $G=5$, St70



(f) $G=10$, St70



(g) $G=5$, Eil76



(h) $G=10$, Eil76

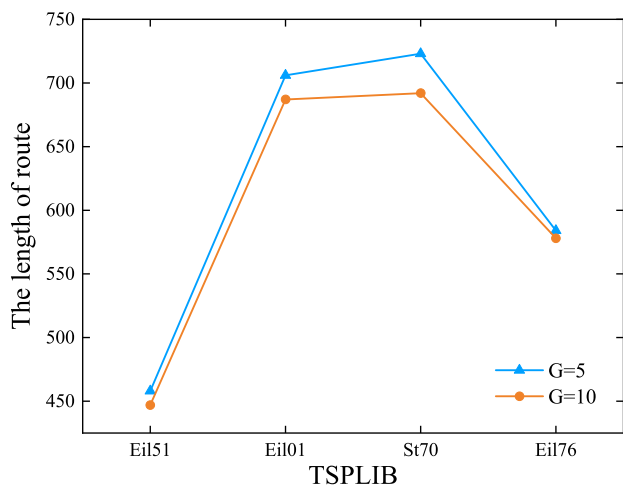


Fig. 8 Experimental results with $G = 5$ and $G = 10$

5.5 Selection strategy

A new selection strategy is adopted, and after crossover and mutation, the new individuals are put together with the original population. Subsequently, all the chromosomes are arranged from good to bad according to fitness values. The chromosomes, whose number is equal to the population size, are selected to the next iteration.

GORTS is designed as a global optimisation algorithm, suitable for dynamical environments with global changes, such as the change of distance between cities and the change of number of cities. In this article, DTSP is abstracted into a series of TSPs. This requires that GORTS has the strong convergence ability to get the better solution during the sampling period. The workflow of GORTS is shown in Fig. 6.

6 Experiments and protosystems

6.1 Performance criteria

According to Eq. 19, assume that the distance matrix of DTSP is a function of time. Therefore, a test case of the dynamical properties of DTSP can be generated by modifying the value of distance between two nodes

$$d'_{ij} \leftarrow d_{ij} \times t_{ij} \tag{19}$$

where t_{ij} represents the change on the edge between v_i and v_j .

$$t_{ij} = \begin{cases} t_{ij} \leftarrow 1 + r \in [F_L, F_U], & \text{if } q \leq m \\ t_{ij} \leftarrow 1, & \text{otherwise} \end{cases} \tag{20}$$

where r is a random variable uniformly distributed in $[F_L, F_U]$, q is a random variable uniformly distributed in

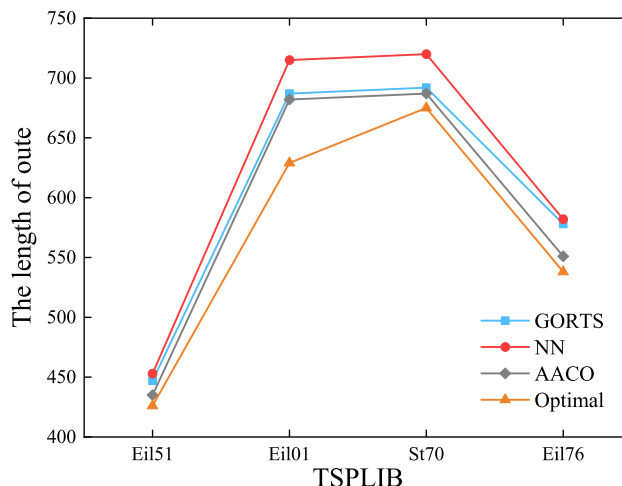
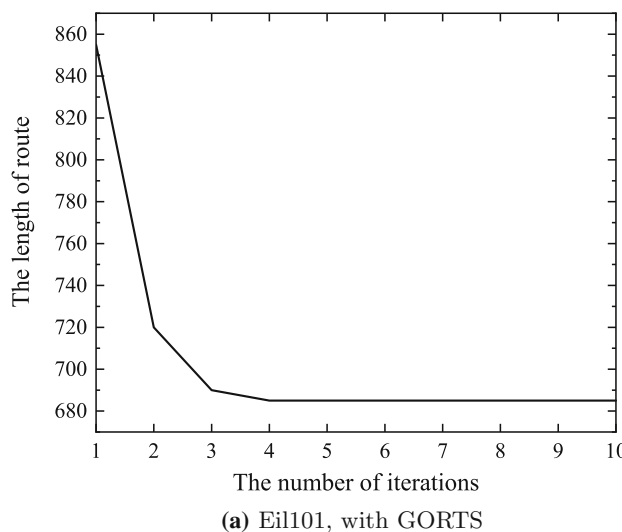
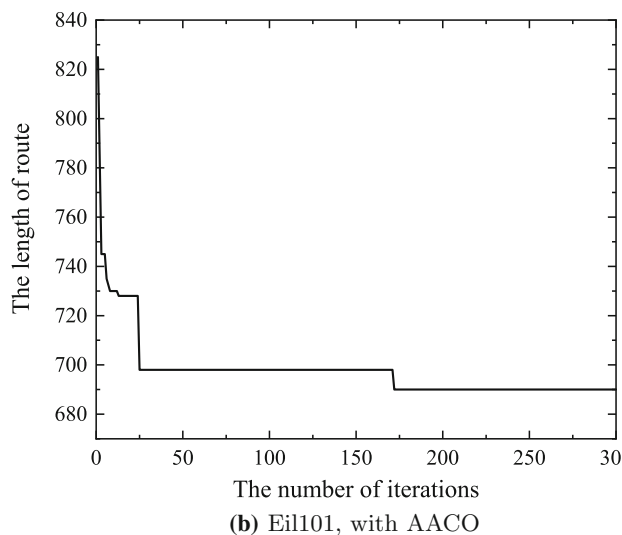


Fig. 9 Length of route planned with different algorithms



(a) Eil101, with GORTS



(b) Eil101, with AACO

Fig. 10 Iteration times

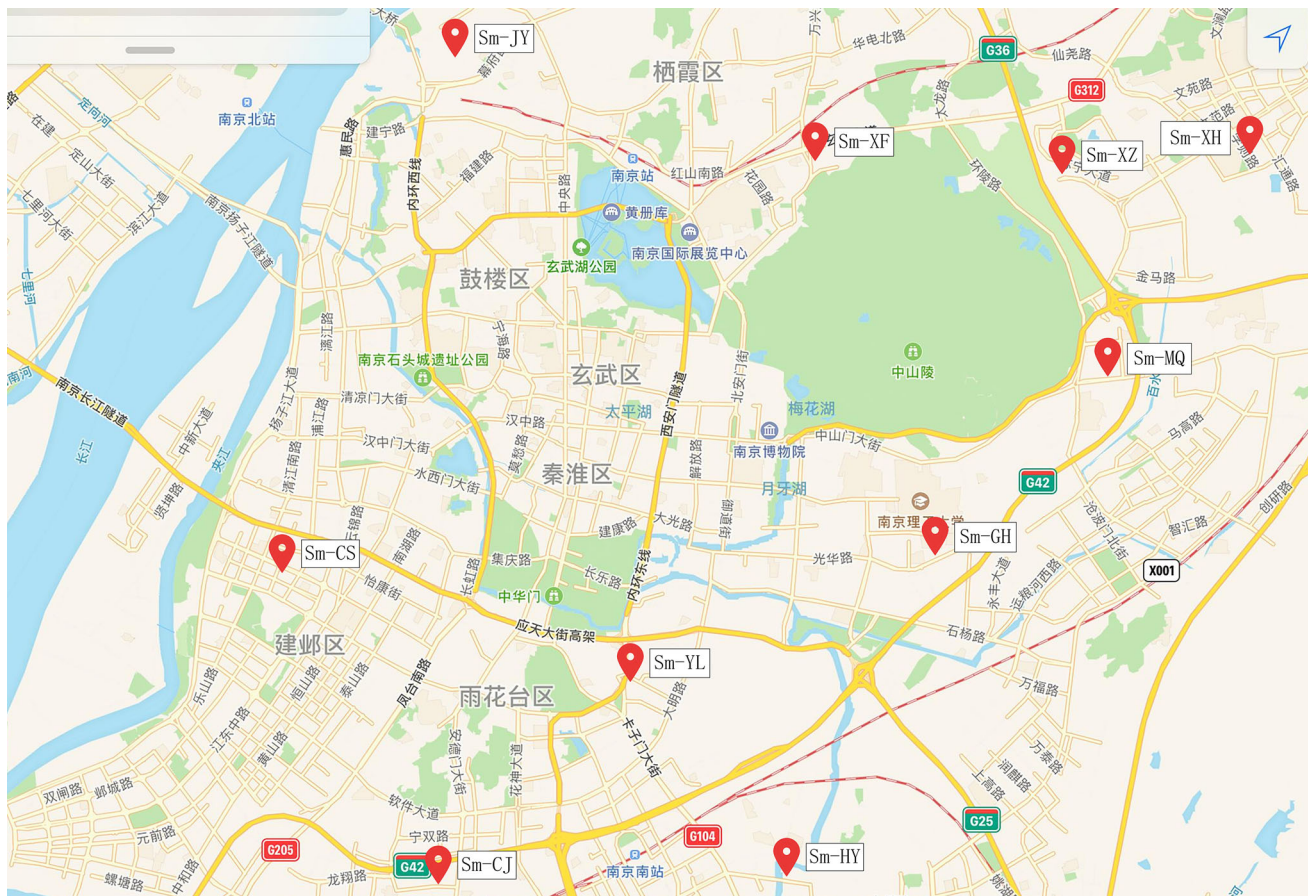


Fig. 11 The geographical sites of ten stores at the Gulou District

$[0,1]$, and m for $0 < m \leq 1$ defines the magnitude of the change. For every arc, a different r value is generated to embed real-world characteristics in the constructed DTSP. The offline performance Yang et al. (2009) is generally used as an important criteria in solving dynamic optimisation problems

$$\overline{P}_{\text{OFF}} = \frac{1}{I} \sum_{i=1}^I \left(\frac{1}{E} \sum_{j=1}^E P_{ij}^* \right) \quad (21)$$

where I is the total number of iterations, E is the number of independent executions, and P_{ij}^* is the best-so-far solution.

6.2 Experiments

We selected four representative datasets from TSPLIB Gerhard (2013) with different scales of cities, ranging from 52 to 318. The value of m is defined within $[0, 0.25]$, $[0, 0.5]$ or $[0, 1]$, to create different dynamic changes of the environment. Subsequently, GORTS is compared with typical heuristic algorithms, including the MAX-MIN ant system (MMAS) Stutzle and Hoos (1997), the population based on

ACO (PACO) Guntch and Middendorf (2002), the elitism-based immigrants ACO (EIACO) Mavrouniotis and Yang (2013), and EIGA-GAPX Tinos et al. (2014). The experimental parameter values are set as follows:

As shown in Table 1, In Table 2 only MMAS has slightly better performance than GORTS with changes of distance between cities.

In the rest of the section, the performance of GORTS compared with algorithms with changes in the number of cities will be discussed. We selected four representative datasets from TSPLIB, including Eil51, Eil101, St70, Eil76, and defined DTSP (t) as follows:

$$\text{DTSP}(t) = \begin{cases} \text{Eil51} & 0 \leq t \leq \Delta t \\ \text{Eil101} & \Delta t < t \leq 2\Delta t \\ \text{St70} & 2\Delta t < t \leq 3\Delta t \\ \text{Eil76} & 3\Delta t < t \leq 4\Delta t \end{cases} \quad (22)$$

We tested GORTS in different sample periods. Specific parameters are set as the following two groups:

1. If $\Delta t = 1s$, $G = 5$, $P_{\text{size}} = 10$, $p_c = 0.80$, and $p_m = 0.05$.



Fig. 12 The process of route adjusting

2. If $\Delta t = 2s$, $G = 10$, $P_{size} = 10$, $p_c = 0.80$, and $p_m = 0.05$.

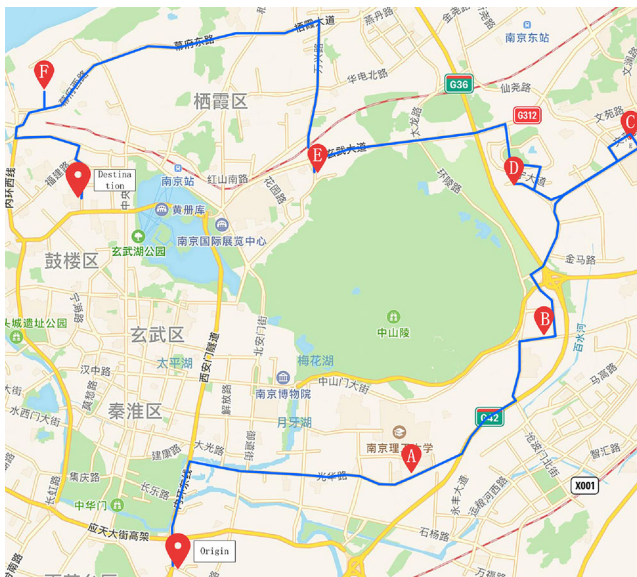
As shown in Fig. 7, there is no cross in any routes, which indicates that the solution quality of GORTS is good. With the increase in the number of iterations, we can find that the quality of the solution is improved, as shown in Fig. 8. However, with respect to DTSP, it is necessary to find the

optimal solution in a short time, which means reducing the number of iterations.

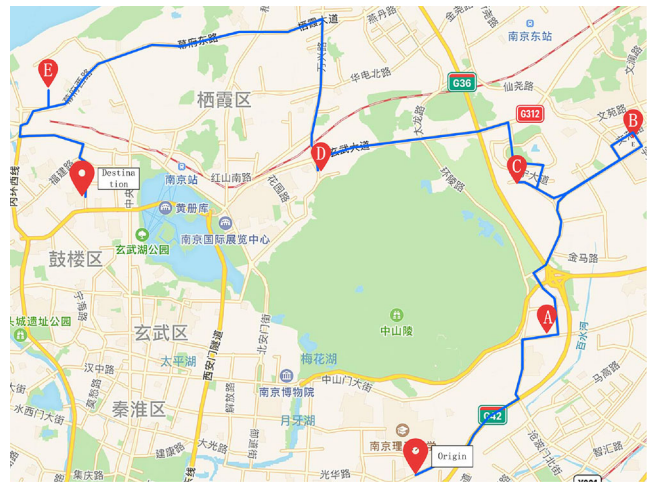
Furthermore, we compared the length of route planned with GORTS, the adaptive ACO algorithm (AACO) Liu et al. (2012), and the nearest neighbour method (NN) Wang (2014). As shown in Fig. 9, the length of route planned with GORTS is shorter than the length of route planned with NN



(a) Route 3



(b) Route 4



(c) Route 5

Fig. 13 The process of route adjusting

but slightly longer than the length of route planned with AACO.

The dynamic properties of DTSP require the algorithm to quickly obtain the optimal solution. GORTS can reach this after 10 iterations in 2s, while the length of route planned by AACO with 10 iterations is much longer.

As shown in Fig. 10, it can be seen that GORTS only needs 5 or 6 iterations to converge to satisfying results, while AACO needs to iterate 150 times to converge to satisfying results. The computation time increases with the increasing number of iterations.

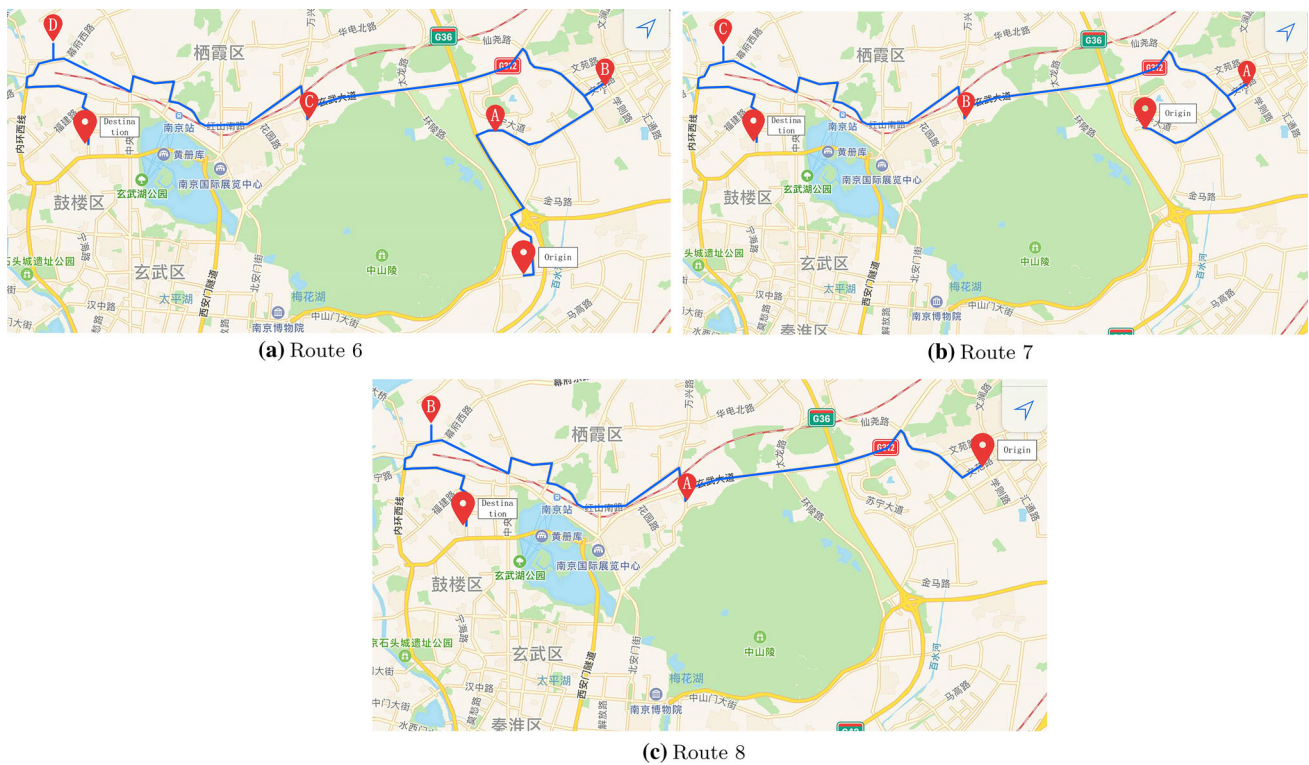


Fig. 14 The process of route adjusting

7 Prototype implementation and testing

We applied GORTS to the logistics distribution planning system for large supermarket chains with distribution centre. Low efficiency in the logistics distribution planning system is likely to cause a waste of transport capacity and the high distribution costs. We implemented the logistics distribution planning system with the Baidu Map SDK based on the Android 4.2 platform. We selected ten supermarket stores at the Gulou District. The detailed locations of these ten stores are shown in Fig. 11 and below the Table. The distribution centre of the supermarket at the Hehui road (Sm-HH) is the starting point of distribution.

No.	Name of stores	Abbreviation	Latitude and longitude
1	JinYan Store	Sm-TY	(118.768708, 32.110693)
2	Gulou Community Store	Sm-CS	(118.733738, 32.023282)
3	Chunjiang Store	Sm-CJ	(118.763309, 31.968957)
4	Yongle Store	Sm-YL	(118.804406, 32.004606)
5	Hongyun Store	Sm-HY	(118.835017, 31.970562)
6	Xinfen Store	Sm-XF	(118.840049, 32.092209)
7	Guanhua Store	Sm-GH	(118.861029, 32.023088)
8	Maqun Store	Sm-MQ	(118.901653, 32.056732)
9	Xueheng Store	Sm-XH	(118.921983, 32.100508)
10	Xuzhuang Store	Sm-XZ	(118.889726, 32.088439)

Note that 0 represents the distribution centre. The initial logistics distribution route is: $0 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 6 \rightarrow 1 \rightarrow 0$, as shown in Fig. 12a. The delivery of the goods depends on specific initial arrangement and the delivery route will be adjusted according to the real-time traffic. As shown in Fig. 12c, there is a traffic congestion along the route to Sm-YL (Fig. 13). Therefore, the rest route is recalculated, which is $3 \rightarrow 5 \rightarrow 4 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 1$. Finally, on the way to Sm-XH, the route is adjusted as: $8 \rightarrow 10 \rightarrow 9 \rightarrow 6 \rightarrow 1$, as shown in Fig. 14a. Compared with the initial route, the route planned with GORTS can be adaptive to condition of traffic and save the cost of time.

8 Conclusion

TSP is an NP combinatorial optimisation problem, with important theoretical value and many applications. DTSP is an adaptation with usability in a realistic dynamic environment. GA is an intelligent search algorithm for simulating biological evolution, which is widely used in solving TSP. In this article, based on the analysis of DTSP theory and mathematical model, a genetic algorithm based on the method of one-by-one revision of two sides (GORTS) is introduced.

This method integrates the better global search ability of genetic algorithm, while improving the convergence speed of

the algorithm by adding the method of one-by-one revision of two sides to correct the chromosome. Finally, GORTS was compared with other algorithms and the experimental results showed it can provide accurate solutions, while the reduced computational time ensures the algorithm is suitable for use for models involving dynamic environments.

Acknowledgements This work was jointly sponsored by the National Natural Science Foundation of China under Grants 61472192 and 91646116, the Scientific and Technological Support Project (Society) of Jiangsu Province under Grant BE2016776, the Talent Project in Six Fields of Jiangsu Province under Grant 2015-JNHB-012, the “333” Scientific Research program of Jiangsu Province under Grant BRA2017228, and the Jiangsu Key Laboratory of Big Data Security and Intelligent Processing at NJUPT.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

- Alves RMF, Lopes CR (2015) Using genetic algorithms to minimize the distance and balance the routes for the multiple traveling salesman problem. In: in Proceedings of CEC, Sendai. 1–8
- Avin R, Agin D, Adnan M (2012) Solving tsp using genetic algorithms—case of Kosovo, advances in. *Computer Sci* 12(11):256–260
- Bessis N, Sotiriadis S, Pop F, Cristea V (2012) Optimizing the energy efficiency of message exchanging for service distribution in interoperable infrastructures. pp 105–112. <https://doi.org/10.1109/iNCoS.2012.16>
- Blackwell BT, Branke J (2004) Multi-swarm optimization in dynamic environments. *Applications of Evolutionary Computing, Evoworkshops* 3005
- Boryczka, U, Strak Ł (2012) A hybrid discrete particle swarm optimization with pheromone for dynamic traveling salesman problem. In: *Computational Collective Intelligence. Technologies and Applications*, LNCS pp 503–512
- Cheng H, Yang SX (2009) Genetic algorithms with elitism-based immigrants for dynamic shortest path problem in mobile ad hoc networks. *IEEE Congress Evolut Comput (CEC) 2009(14)*:7–3135
- Cook WJ (2011) In pursuit of the traveling salesman: mathematics at the limits of computation. *J Classif* 12(5):19–21
- Falcon R, Nayak A (2010) The one-commodity traveling salesman problem with selective pickup and delivery: an ant colony approach. In: in Proceedings of CEC. pp 4326–4333. Barcelona, Spain
- Garey MR, Johnson DS (1979) *Computers and intractability: a guide to the theory of np-completeness*. Wh Freeman & Company, New York
- Gerhard R (2013) Tsplib. Website <https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>
- Ghadle KP, Muley YM (2014) An application of assignment problem in traveling salesman problem (tsp). *Journal of Engineering Research and Applications* pp 169–172
- Gharehchopogh FS, Farahmandian IMM (2012) New approach for solving dynamic travelling salesman problem with hybrid genetic algorithms and ant colony optimization. *Int J Comput Appl* 53(1):0975–8887
- Guntsch M, Middendorf M (2002) Applying population based aco to dynamic optimization problems. In: in *Ant Algorithms, LNCS*. pp 111–122
- Kang L, Zhou A, McKay B, Li Y, Kang Z (2004) Bench-marking algorithms for dynamic travelling salesman problems. In: in proceedings of 2004. *IEEE Congr Evol Comput* 2(2), 1286–1292
- Li J (2013) An improved dynamic programming algorithm for bitonic tsp. *J Classification* 15(6):24–27
- Li C, Yang M, Kang L (2006) A new approach to solving dynamic traveling salesman problem. In: in Proceedings of 6th International Conference on Simulated Evolution and Learning. pp 236–243
- Li ZY, Ma L, Zhang HZ (2015) Discrete bat algorithm for solving minimum ratio traveling salesman problem. *J Clas-sif Appl Res Comput* 32(2):356–359
- Liu Y, Shen, X, Chen H (2012) An adaptive ant colony algorithm based on common information for solving the traveling salesman problem. In: in Proceedings of ICSAI, Shandong .pp 763–766
- Mahfoudh, SS, Khaznaji W, Bellalouna M (2015) A branch and bound algorithm for the probabilistic traveling salesman problem. In: in Proceedings of SNPD, Takamatsu. pp 1–6
- Mavrouniotis M, Müller FM, Yang S (2016) Ant colony optimization with local search for dynamic traveling salesman problems. *IEEE Trans Cybern* 13(4):1–14
- Mavrouniotis M, Yang S (2013) Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors. *J Classif Appl* 13(10):4023–4037
- Melo L, Pereira F, Costa E (2013) ser LNCS: Multi-caste ant colony algorithm for the dynamic traveling salesperson problem. In: in *Adaptive and Natural Computing Algorithms*. pp 179–188
- Niendorf M, Kabamba PT (2015) Girard a r. stability of solutions to classes of traveling salesman problems. *Cybernetics IEEE Trans* 12(1):974–985
- Niendorf M, Kabamba PT, Girard AR (2016) Stability of solutions to classes of traveling salesman problems. *IEEE Trans Cybern* 46(4):973–985
- Reina DG, León-Coca JM, Toral SL, Asimakopoulou E, Barrero F, Norrington P, Bessis N (2014) Multi-objective performance optimization of a probabilistic similarity/dissimilarity-based broadcasting scheme for mobile ad hoc networks in disaster response scenarios. *Soft Comput* 18(9):1745–1756. <https://doi.org/10.1007/s00500-013-1207-3>
- Singh G, Mehta R (2014) Implementation of travelling salesman problem using ant colony optimization. *J Eng Res Appl* 6(3):385–389
- Stutzle T, Hoos H (1997) Max-min ant system and local search for the traveling salesman problem. *IEEE Int Conf Evol Comput* 309–314
- Tinos R, Whitley D, Howe A (2014) Use of explicit memory in the dynamic traveling salesman problem. In: in Proceedings of 2014 Annual Conference Genetic
- Wang Y (2014) A nearest neighbor method with a frequency graph for traveling salesman problem. In: in Proceedings of IHMSC. Hangzhou. pp 335–338
- Xu X, Bessis N, Cao J (2013) An autonomic agent trust model for iot systems. *Procedia Computer Science* 21, 107 – 113. <https://doi.org/10.1016/j.procs.2013.09.016>, <http://www.sciencedirect.com/>

[science/article/pii/S11877050913008090](https://doi.org/10.1007/s11877050913008090), the 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2013) and the 3rd International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH)

Yang XW, Chen ZJ (2009) Li: A 1, seeking the best hamilton cycle through matrix tuning. *J Logistical Eng Univ* 24(1):102–106

Yuan S, Skinner B, Huang S, Liu D (2013) A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. *Eur J Oper Res* 228(1):72–82

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.