Journal of Combinatorial Optimization
https://doi.org/10.1007/s10878-021-00759-5

Angular bisector insertion algorithm for solving small-scale symmetric and asymmetric traveling salesman problem

Jian Lin[1] · Xiangfei Zeng[1] · Jianxun Liu[1] · Keqin Li[2]

Accepted: 15 May 2021

Abstract
Different algorithmic performances are required in different engineering fields for solving both the symmetric and asymmetric traveling salesman problem (STSP and ATSP), both of which are commonly referred to as TSP. In the background of small-scale TSP, according to the principle of the optimal Hamiltonian loop, this paper describes an angular bisector insertion algorithm (ABIA) that can solve TSP. The main processes of ABIA are as follows. First, the angular bisector of the point group is constructed. Second, the farthest vertex perpendicular to the angular bisector is identified as the search criterion. Finally, for both STSP and ATSP, initial loop formation rules and vertex insertion rules are constructed. Experiments were conducted for all STSP and ATSP instances with up to 100 points in the TSPLIB database. The performance of ABIA was compared with that of the 2-point farthest insertion algorithm, convex hull insertion algorithm, branch-and-bound algorithm, and a genetic algorithm. The experimental results show that, for small-scale TSP (up to 40 points), the runtime of ABIA is second only to the convex hull insertion algorithm, and the gap between ABIA and the optimal solution is second only to the exact algorithm. ABIA offers good overall performance in solving small-scale TSP.

Keywords Angle bisector insertion algorithm · Asymmetric traveling salesman problem (ATSP) · Constructive heuristic algorithm · Hamiltonian cycle · Traveling salesman problem (TSP)

1 Introduction

A traveling salesman travels from one city to another to several cities (one city at a time) to sell goods and return to the city where he starts his journey. One of the problems he has to address is to find the shortest journey. If the journey from City

✉ Xiangfei Zeng
18010104015@mail.hnust.edu.cn

Extended author information available on the last page of the article

Published online: 31 May 2021

A to City B is equal to that from City B to City A, this problem is called symmetric traveling salesman problem (STSP). Otherwise, it is called asymmetric traveling salesman problem (ATSP). STSP and ATSP are collectively called TSP.

TSP can be expressed in terms of finding the optimum Hamiltonian cycle in a weighted undirected/directed graph (STSP/ATSP), which is an NP-hard combinatorial optimization problem. TSP widely exists across network wiring, mechanical drilling, artificial intelligence, distribution logistics, and other engineering fields (Yang et al. 2018). However, different engineering fields have different algorithm performance requirements. For example, in solving a large-scale vehicle routing problem (VRP), it is necessary to repeatedly compare the ATSP paths of different delivering store combinations. That is to say, the frequency required to solve the ATSP is high, but due to the rated loading capacity of vehicles involved, the number of combined stores is usually restricted to not more than 30, and the stability of results are also required. A further example related to distribution networks, is to solve 1000 small scale (around 40 cities) TSP within 30 seconds, that is, high speed algorithms are required (Sakurai et al. 2006). At present, the attention of academics has been focused on large-scale meta-heuristic algorithms. But for small-scale TSP applications that require high speed and stability, theoretically, meta-heuristic algorithms are usually inferior to constructive heuristic algorithms in terms of computational speed and stability (Rao and Jin 2012; Ismail 2019). Therefore, it has greater application value in respect to the study of constructive heuristic algorithms with high-precision, high speed and stability for STSP and ATSP.

The algorithms available to solve TSP can be categorized into exact algorithms and heuristic algorithms. Although exact algorithms can be used to obtain an optimal solution, it requires a lot of computation time. For example, a STSP of 20 points will have about 1.22×10^{17} feasible solutions (Kanda et al. 2016). As the number of points increases, the computation time will increase exponentially. While heuristic algorithms can be adopted to obtain an approximately optimal solution, heuristic algorithms can be categorized into constructive algorithms and meta-heuristic algorithms (the latter sometimes referred to as an improved heuristic algorithm). Constructive heuristic algorithms begin with one city, iteratively expanding sub-loops, one city at a time through determined search rules. The main constructive heuristic algorithms are 'greedy' algorithms (Reinelt 1994), insertion algorithms (Hore et al. 2018), and convex hull insertion algorithms (Golden et al. 1980). Because the search rules are definitive, constructive heuristic algorithms are typically very fast and can provide stable calculation results. But they can be easily trapped around local optima (Rosenkrantz et al. 1977) and are difficult to use in solving large-scale STSP. Current research is mainly focused on how to establish global heuristic rules (Ursani and Corne 2016) and segment solution algorithms for large-scale TSP (Xiang et al. 2015). In contrast, meta-heuristic algorithms begin with a complete journey and then make rearrangements in order to improve it. The representative meta-heuristic algorithms include genetic algorithms, simulated annealing algorithms (Yang et al. 2019), and particle swarm optimization (Zhong et al. 2018). Compared with constructive heuristic algorithms, meta-heuristic algorithms generally have higher precision when they are used to solve TSP. However, the results are unstable with

slow processing speeds. The current research is mainly focused on 1) how to construct random operators to avoid falling into local optimum in solving large-scale TSP (Pan et al. 2016; Zhu et al. 2017), 2) how to take the advantage of different types of algorithms available for comprehensive application (Hore et al. 2018; Zhang et al. 2018, 3) and how to design large-scale parallel computing algorithms (Siemiński and Kopel 2019).

In this paper, a new insertion algorithm (IA), angular bisector insertion algorithm (ABIA), is proposed to solve small-scale TSP. According to Hamiltonian loop theory, the main idea can be put forward that any vertex can be regarded as the starting point, the angle bisector of point group is made, and a 2-vertex initial loop is constructed. Then the vertices farthest from the angle bisector are inserted into the loop in turn to solve STSP and ATSP. In ABIA, the bisector reflects spatial distribution of point group, and the bisector is always taken as the baseline in vertex search algorithm. Compared with other insertion algorithms, ABIA can avoid local optimum in a better way.

The remainder of this paper is organized as follows. In Sect. 2, the correlative theory of Hamiltonian loops and the IA are introduced. The proposed angular bisector insertion algorithm for STSP and ATSP is described and analyzed in Sect. 3. In Sect. 4, the experiments that compare ABIA with the 2-point farthest insertion algorithm (FIA-2), convex hull insertion algorithm (CHIA), genetic algorithm (GA), and branch-and-bound algorithm (B&B) are evaluated. Finally, Sect. 5 presents our conclusions and ideas for future research.

2 Hamiltonian loop and inserting constructive heuristic algorithms

2.1 Correlative theory of Hamiltonian

In a graph $G(V,E)$, V and E denote the vertex set and the edge set of graph, respectively. Let $v_i(i \in V)$ be vertex i and let $e_{ij}(i,j \in V)$ be the distance between vertex i and vertex j.

Definition 1 If loop C passes through all vertices in G only once, it is called a Hamiltonian sloop (H-loop for short) of G. A graph with a H-loop is called a Hamiltonian graph.

Theorem 1 (Bondy and Murty 1976) *If G has a H-loop, then for any non-empty subset, the number of connected components of graph G\S is at most |S|* (*Fig*. 1).

Theorem 2 (Ore 1960) **Let** $n(G)$ **be the number of vertices of graph** G, $n(G) \geq 3$. *Let* $\delta(v_u)$ *be the degree of* v_u. *If any two non-adjacent vertices* v_u *and* v_v *in graph G have* $\delta(v_u) + \delta(v_v) \geq n(G)$, *then G is a Hamiltonian graph.*

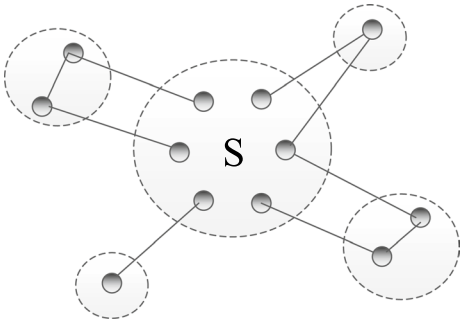


Fig. 1 Connected components of graph G\S

Corollary 1 (Dirac 1952) **If** *G is a simple graph with* $n(G) \geq 3$ *and the degree of each vertex is at least* $n(G)/2$, *then G is a Hamiltonian graph.*

A Hamiltonian graph usually has a large number of edges. If there are certain conditions to ensure that the edges in the graph are "scattered," particularly to avoid circuit crossing, then graphs with only a few edges may contain an H-loop.

Definition 2 The weighted graph $G\ (V, E, W)$ is a Hamiltonian graph. The weight of edge $e_{ij} \in E$ is $w_{ij} \in W(i, j \in V)$. Let H be the H-loop set of G; $h \in H$ and $d(h) = \sum_{\forall eij \in h} w_{ij}$. If $d_{\min}(c) = \min(d(h)|h \in H)$ for some $c \in H$, then c is the optimal H-loop.

The number of Hamiltonian loops in graph G is the number of combination of edges constrained by Hamiltonian rules. According to theorem 1, the number of vertices of non-empty subset $V(S)$ is related to the number of connected branches. If G is incomplete graph, the number of connected branches of $V(S)$ is also related to the vertex degree of graph G, that is, to the spatial distribution of vertices.

2.2 Insertion algorithm

The basic idea of the IA is to form an initial loop with fewer vertices and then insert the remaining vertices into the loop to minimize the loop path. The basic processes of the insertion heuristic algorithm can be summed up as follows:

(1) Arbitrarily select an initial loop with k vertices, $v_1, v_2, \cdots, v_k (k \geq 1)$. Let the set S be $V \backslash \{v_1, v_2, \cdots, v_k\}$.
(2) As long as $S \neq \emptyset$, perform the following steps:

(2.1) Select vertex $j (j \in S)$ according to the same search rule.
(2.2) Insert vertex j into the loop, so that $S = S \backslash \{v_j\}$.

In the IA, there can be one or more initial loop vertices, and various search rules can be used to select a vertex. Under the basic idea of the IA, different initial loop

generation rules or vertex insertion search rules are formulated to produce different IAs. The existing IAs include the nearest insertion algorithm, farthest insertion algorithm, least-cost insertion algorithm, arbitrary insertion algorithm and CHIA. Among them, the nearest insertion algorithm, farthest insertion algorithm, and CHIA are most commonly used.

Let the vertex set of the initial loop of IA be $V(R) = \{v_1, v_2, \cdots, v_k\}$ $(k \geq 1)$, and $V(R) \subseteq V(G)$. The optimal Hamiltonian loop of subgraph R is $c_R = v_1 v_2 \cdots v_k v_1$. The set of directed edges formed in the same clockwise or counterclockwise direction is $E(c_R) = \{< v_1, v_2 >, < v_2, v_k >, < v_1, v_k >, \cdots, < v_k, v_1 >\}$, where $< v_1, v_k > \neq < v_k, v_1 >$. The optimal Hamiltonian loop of G is $c_G = v_1 v_2 \cdots v_i v_k \cdots v_j v_1$. The set of directed edges formed in the same clockwise or counterclockwise direction is $E(C_G)$. For IA, it can be characterized as follows:

(1) If $E(C_R) \subseteq E(C_G)$, that is, the order of vertices in $E(C_R)$ is the same as that of $E(C_G)$, when a new vertex is inserted into the initial loop, let the set of directed edges formed by all vertices in the loop be $E(C_I)$,and $E(C_R) \subseteq E(C_I)$. If $E(C_I) \not\subset E(C_G)$, $E(C_G)$ cannot be obtained, it means that the optimal Hamiltonian loop cannot be obtained. But when all vertices are completely inserted into the initial loop, $E(C_I) \subseteq E(C_G)$,that is $E(C_G)$ can be obtained. (2) If $E(C_R) \not\subset E(C_G)$, no matter what search rule is adopted, $E(C_G)$ cannot be obtained.

Based on the above analysis, when the insertion algorithm is designed, it is beneficial to obtain $E(C_G)$ according to the following design principles:

Principle 1 Select the global initial vertex set by the vertex spatial distribution characteristics of graph G.
Principle 2 $E(C_R) \subseteq E(C_G)$ requires that the initial loop is a part of the globally optimal Hamiltonian loop.
Principle 3 The search rules should be global, and crossing of edges should be avoided as far as possible when new vertex is inserted into the loop.

2.2.1 2-Point farthest insertion algorithm

In 2-Point farthest insertion algorithm (FIA-2), any two vertices are taken to form an initial loop, with vertex (not in the loop) farthest from each vertex (in the loop) selected to insert into the loop in turn. The shortest distance from vertex j (not in the loop) to vertex i (in the loop) is defined as $j \in S, d_{\min}(j) = \min(w_{ij} | i \in V \backslash S)$.The basic process of FIA-2 can be described as follows:

(1) Arbitrarily select an initial loop with two vertices v_1, v_2, and let set $S = V \backslash \{v_1, v_2\}$.
(2) As long as $S \neq \emptyset$, perform the following steps:

(2.1) Select vertex $j \in S$ to satisfy $d_{\min}(j) = \max(d_{\min}(m) | m \in S)$.
(2.2) Insert vertex j into the loop, and then $S = S \backslash \{v_j\}$.

In the process of FIA-2, the initial vertex is selected arbitrarily, and the selection of vertex exerts an impact on the TSP result (not in conformity with Principle 1).

The number of initial vertices is 2, which satisfies $E(C_R) \subseteq E(C_G)$ (in conformity with Principle 2). According to FIA-2 search rule, with the new vertex inserted into the loop, the reference points for finding the farthest vertex changes, which means that the currently inserted vertex has an influence on the vertices inserted later. Once $E(C_I) \not\subset E(C_G)$ appears, it will have a great influence on vertex insertion in the future (not in conformity with Principle 3).

2.2.2 Convex hull insertion algorithm

According to the basic idea of convex hull insertion algorithm (CHIA): If k vertices $v_1, v_2 \cdots, v_k (k \geq 3)$ can form a maximum convex hull in which the remaining vertices are contained, it will be the initial loop. And then the vertex that minimizes the cost of the loop is inserted into the loop. When vertex j is inserted into loop, its adjacent vertices u and v are defined as: where $j \in S$ u and v satisfy $d_{\min}(u,j,v) = \min\left(w_{uj} + w_{jv} - w_{uv} | u, v \in V \backslash S\right)$.

The basic flow of the algorithm is presented as follows:

(1) Form the initial loop of convex hull with k vertices $v_1, v_2, \cdots, v_k (k \geq 3)$, then set $S = V \backslash \{v_1, v_2, \cdots, v_k\}$.
(2) As long as $S \neq \emptyset$, perform the following steps:

(2.1) Select vertex $j \in S$ satisfying $d_{\min}(u,j,v) = \min\left(d_{\min}(u,m,v) | m \in S\right)$.
(2.2) Insert vertex j into the loop, and then $S = S \backslash \{v_j\}$.

CHIA is difficult to guarantee that $E(C_R) \subseteq E(C_G)$ (not in line with Principle 2). In addition, like FIA-2, the step of vertex selection changes with the vertices in the loop. But initial loop of CHIA is determined according to the spatial distribution of vertices. Meanwhile, the edges of the loop can be dispersed as much as possible by inserting the vertex with the least cost into the loop formed by convex hull. (in conformity with to principle 1 and partially in conformity with principle 3).

3 Angular bisector insertion algorithm

In angular bisector insertion algorithm (ABIA), any vertex is taken as starting point to make an angular bisector of point group, with the vertex farthest from the starting point selected as initial path point. If there is no other vertex on the angular bisector, a projection is made from the farthest point to the bisector so as to form a projective point, with the vertex closest to the projective point selected as the initial path point. And it will form an initial loop with the starting point and the initial path point. The remaining vertices are inserted into the loop with the maximum vertical distance from the angle bisector.

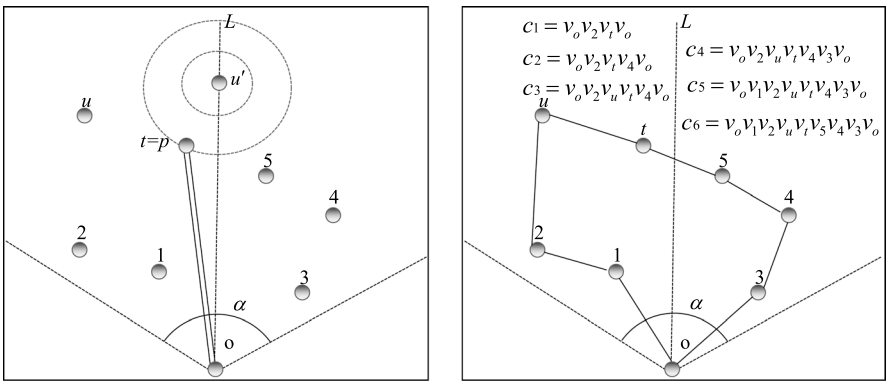
Let $d_{ij} = w_{ij}$ be the distance between $i \in V$ and $j \in V$, and $d(i, L)$ be the vertical distance from vertex i to the bisector line L. ABIA uses two vertices (the initial vertex and the initial path point) to form the initial loop. The basic rules can be summed up as follows:

(1) Generation rule for angular bisector: select any initial vertex $o \in V$ in the graph and form an angle $\angle\alpha$ that ensures all vertices are in the interior. On this basis, create a bisector line L of $\angle\alpha$ (see Fig. 2a).
(2) Initial path point selection rule: let $t \in V$ be the initial path point, with three steps as follows:

(2.1) Select the farthest point $u \in V$ from vertex o, i.e., $d_{ou} = \max\left(w_{oi} | i \in V\right)$.
(2.2) If vertex u is on the angular bisector L, then t=u.
(2.3) Else, let u' be an auxiliary point on L and set $d_{ou} = d_{ou'}$. Select the nearest point $p \in V$ from u', i.e., $d_{u'p} = \min\left(w_{u'i} | i \in V\right)$, t=p (See Fig. 2b).

(3) Vertex insertion search rule: select vertex $j \in S$ such that $d(j, L) = \max(d(m, L) | m \in S)$.
(4) Vertex insertion rule: if u, v satisfy $\min\left(w_{uj} + w_{jv} - w_{uv} | u, v \in V \backslash S\right)$, insert vertex j into e_{uv}.

At present, the existing IA is based on an undirected graph without involving the directed graph. In other words, it concerns STSP rules only and is not applicable to ATSP. In this section, the STSP and ATSP rules are formulated respectively to form the ABIA for STSP and ATSP.



(a) Initial loop **(b)** Order of vertices added to loop

Fig. 2 ABIA-STSP initial loop and the order of vertices added to loop schematic diagram

3.1 Initial loop generation and vertex insertion rules for STSP

(1) Determine the initial loop according to step1 (1) and (2) above. Connect the vertices to form the loop $c = v_o v_t v_o$ so that the set of vertices that are not in the loop is $S = V \backslash \{v_o, v_t\}$.
(2) As long as $S \neq \emptyset$, perform the following steps:

(2.1) Select the vertex to be inserted according to step (3) above.
(2.2) Insert vertex i into the loop c, so that $S = V \backslash \{v_i\}$.

When S=∅, i.e., all vertices of graph G are in loop c, the solution of the STSP is c (Fig. 2b).

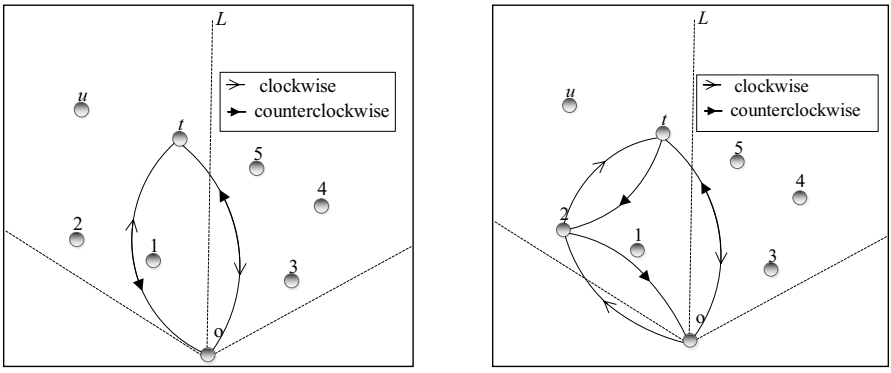
3.2 Initial loop generation and vertex insertion rules for ATSP

Let *G(V, E, W)* be a directed weighted graph with $w_{ij} \neq w_{ji}$. The clockwise and counterclockwise loops of vertex *i, j* are $c = v_i \rightarrow v_j \rightarrow v_i$, $c' = v_i \leftarrow v_j \leftarrow v_i$, respectively.

(1) Determine the initial loop vertex according to Step (1) and (2) in Sect. 3. Connect vertices to form the clockwise loop $c = v_o \rightarrow v_t \rightarrow v_o$, counterclockwise loop $c' = v_o \leftarrow v_t \leftarrow v_o$, and the set of vertices that are not in the loop $S = V \backslash \{v_o, v_t\}$ (Fig. 3a).
(2) As long as $S \neq \emptyset$, perform the following steps:

(2.1) Select the vertex to be inserted according to Step (3) in Sect. 3.
(2.2) Insert vertex i into the clockwise loop c and counterclockwise loop c', so that $S = V \backslash \{v_i\}$ (Fig. 3b).

When S=∅, $\min(c, c')$ is the solution of the ATSP.



(a) Initial loop **(b)** Order of vertices added to loop

Fig. 3 Schematic Diagram for ABIA-ATSP initial loop and the order of vertices added to loop

3.3 Characteristics and complexity of ABIA

The initial loop of ABIA is a 2-vertex loop. There is $E(C_R) \subseteq E(C_G)$ for STSP, $E(C_R) \subseteq E(C_G)$ for the directed edge of ATSP clockwise loop c, and $E(C'_R) \subseteq E(C'_G)$ for counterclockwise loop c'. Therefore, the initial loop of ABIA is the optimal Hamiltonian loop. The benchmark of ABIA is the bisector that can reflect the spatial distribution characteristics of point groups. In vertex search, the bisector is always taken as the reference line. When $E(C_I) \not\subset E(C_G)$, the new insertion point has little influence on the subsequent loop, so the search rule of ABIA is global. At the same time, when the vertex furthest from the angle bisector is inserted into the loop, the edges of the loop will be dispersed as much as possible. However, ABIA is also affected by the selection of initial vertices as FIA-2. Therefore, ABIA complies with principles 2 and 3 and partially complies with principle 1.

Hamiltonian loop is a combination of edges of graph. For STSP, the number of Hamiltonian loops is $(n-1)!/2$. The time complexity of exact algorithm is $O((n-1)!/2)$, and the time complexity of CHIA is $O(n \log n)$ (Jünger et al. 1995). In order to eliminate the influence of the selection of initial vertex set on ABIA and FIA-2, all vertices are taken as starting point and the best loop is selected as the algorithm result. The time complexity of FIA-2 to obtain two-point initial loop is $O\left(\frac{n(n-1)}{2}\right)$, and the total time complexity is $O\left(\frac{n(n-1)\times(n-2)^2}{2}\right)$. When the two-point initial loop of ABIA determines a vertex, the other vertex does not need to be combined with other n-1 vertices like FIA-2. In addition, the vertex selection only needs to be on one side of the angular bisector. If the number of vertices on both sides of the bisector is balanced, The time complexity of ABIA is $O\left(3n^2 + \frac{2n^3-8n^2+8n}{2}\right)$. The time complexity of GA is $O(Tn_0n^2)$. While n_0 is the initial size of population and T is the number of outer iterations (Hui 2012). In conclusion, CHIA has the lowest time complexity. When the scale of points is small, FIA-2 does not need to preprocess points like ABIA, so FIA-2 is faster than ABIA. On the contrary, when the scale of points is larger, ABIA is faster. GA is affected by iteration times and initial population. For small-scale STSP, the operation time of GA is higher than ABIA.

For ATSP, the number of Hamiltonian loops is (n-1)!. The STSP operation is required for ABIA once in clockwise and counterclockwise directions respectively. Its time complexity is twice that of STSP. In terms of GA, the edges are combined to obtain the optimal solution from the combinations according to the rules of genetic algorithm, and its time complexity is $O(Tn_0(2n)^2)$. Compared with STSP, ATSP-GA is more time-consuming than ATSP-ABIA.

4 Experimental analysis

Experiments were conducted to compare the performance of ABIA against that of other insertion, exact, and meta-heuristic algorithms. The other insertion algorithms were selected according to their similarity to the heuristic rules of ABIA. Therefore, FIA-2 and CHIA were implemented by us for comparison. To solve the problem

of initial vertex selection, all 2 vertex combinations will be calculated for FIA-2 to take the optimal as the solution to the algorithm. Meanwhile, all vertices are calculated for ABIA to take the optimal as the solution. Because these algorithms have no ATSP rules, only STSP experiments were performed.

Exact algorithms and meta-heuristic algorithms have been intensively studied. There are various improved algorithms based on the basic methods. For the comparison experiments, the basic B&B and GA were applied to STSP and ATSP scenarios.

The experimental data consisted of instances with up to 100 vertices from TSPLIB (Reinelt 1991), the standard instance database for STSP and ATSP. The experiments were conducted using a computer with 8 GB memory and an Intel$^{(R)}$ Core$^{(TM)}$ i7-4710HQ 2.50 GHz CPU. The experiments were coded in the JAVA programming language.

4.1 STSP experiment by using ABIA and other insertion algorithms

According to the principle of similarity of heuristic rules, FIA-2 has the same number of vertices as ABIA's initial loop. The initial loop in CHIA uses the strategy that disperse the edges of loop as much as possible and avoid intersections within the loop. It is a similar construction strategy to ABIA. The results for the STSP instances calculated by ABIA and other insertion algorithms are presented in Table 1. The deviation of the optimal solution and algorithm runtime are given in the table.

To analyze the data trends, the computation time and deviation of ABIA and the other insertion algorithms with respect to the STSP scale are shown in Fig. 4. If there are multiple instances with the same number of points, the average value of each instance is taken. Figure 4a shows the number of points (Point Num) against the cumulative deviation (Cum Dev) and Fig. 4b shows Point Num against time.

The following conclusions can be drawn from Table 1 and Fig. 4:

(1) The total deviation between the optimal solution and ABIA, FIA-2 and CHIA is 48.6%, 60.0%, and 78.6% respectively. Meanwhile, with the increasing number of vertex, the deviation between FIA-2 and CHIA increases faster than that of ABIA.
(2) Total runtime of ABIA, FIA-2 and CHIA is 10823.5 ms, 125,103.3 ms and 12.6 ms respectively. The operation speed of CHIA is the fastest. When there are less than 24 vertices, the operation efficiency of FIA-2 is higher than that of ABIA, and ABIA runs faster than FIA-2 as the number of vertices increases.

4.2 STSP experiment using ABIA, B&B, and GA

As an accurate algorithm, B&B is often used to solve mini-scale TSP, with the initial lower bound obtained by using a greedy algorithm. GAs are meta-heuristic algorithms that have been widely studied in solving TSP, allowing a series of improved GAs to be developed. In this experiment, the traditional GA is adopted and the

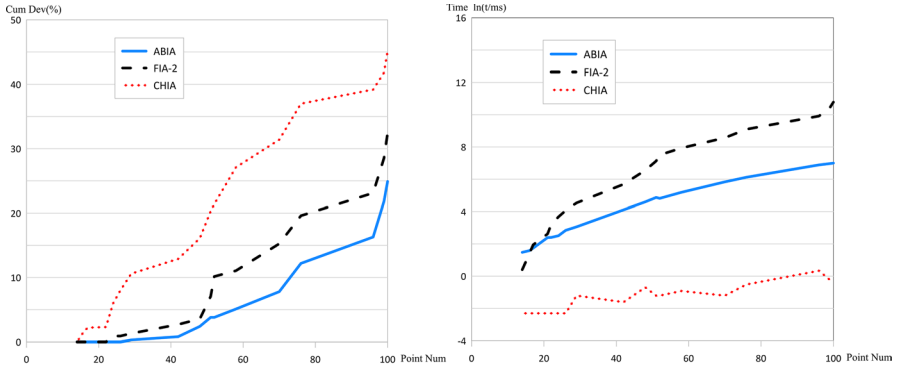
Table 1 STSP instances calculated by ABIA and other insertion algorithms

Instance		ABIA		FIA-2		CHIA	
Name	Best	Time(ms)	Dev (%)	Time(ms)	Dev (%)	Time(ms)	Dev (%)
burma14	3323	4.4	(0) 0.0	1.5	(0) 0.0	0.1	(0) 0.0
ulysses16	6859	4.9	(0) 0.0	2.6	(0) 0.0	0.1	(113) 1.7
gr17	2085	5.7	(0) 0.0	2.9	(0) 0.0	0.1	(10) 0.5
gr21	2707	11.0	(0) 0.0	6.8	(0) 0.0	0.1	(2) 0.1
ulysses22	7013	11.1	(0) 0.0	11.1	(0) 0.0	0.1	(30) 0.0
gr24	1272	12.3	(0) 0.0	14.7	(7) 0.9	0.1	(47) 3.7
fri26	937	17.2	(0) 0.0	19.3	(0) 0.0	0.1	(19) 2.0
bays29	2020	21.6	(6) 0.3	34.4	(6) 0.3	0.4	(75) 3.7
bayg29	1610	21.1	(5) 0.3	33.8	(8) 0.5	0.2	(22) 1.4
dantzig42	699	63.0	(6) 0.9	206.8	(13) 1.9	0.3	(25) 3.6
swiss42	1273	59.2	(0) 0.0	198.5	(12) 0.9	0.1	(13) 1.0
gr48	5046	99.3	(64) 1.3	393.7	(22) 0.4	0.2	(229) 4.5
att48	10,628	98.2	(224) 2.1	443.3	(66) 0.6	1.1	(323) 3.0
hk48	11,461	100.2	(168) 1.5	379.0	(182) 1.6	0.1	(250) 2.2
eilon51	426	130.2	(6) 1.4	544.1	(15) 3.5	0.3	(18) 4.2
berlin52	7542	123.1	(0) 0.0	574.1	(233) 3.1	0.3	(81) 1.1
brazil58	25,395	179.1	(333) 1.3	970.5	(231) 0.9	0.4	(1452) 5.7
st70	675	342.1	(18) 2.7	2483.2	(28) 4.2	0.3	(29) 4.3
eil76	538	457.6	(28) 5.2	3705.6	(29) 5.3	0.7	(38) 7.1
pr76	108,159	454.4	(3888) 3.6	3683.5	(3647) 3.4	0.4	(4319) 4.0
gr96	55,209	979.4	(2309) 4.1	11,301.4	(1934) 3.5	1.4	(1205) 2.2
rat99	1211	1060.8	(67) 5.5	13,446.5	(65) 5.4	0.8	(32) 2.6
kroA100	21,282	1108.3	(534) 2.5	14,606.2	(1025) 4.8	0.9	(772) 3.6
kroB100	22,141	1069.4	(857) 3.9	14,306.6	(1161) 5.2	0.8	(557) 2.5
kroC100	20,749	1109.1	(331) 1.6	14,668.7	(619) 2.9	0.8	(527) 2.5
kroD100	21,294	1100.6	(479) 2.3	14,565.2	(476) 2.2	0.7	(498) 2.3
kroE100	22,068	1105.0	(591) 2.7	14,202.3	(1134) 5.1	0.8	(836) 3.8
rd100	7910	1075.2	(331) 5.4	14,297.0	(272) 3.4	0.9	(415) 5.3
Total	–	10,823.5	(10,341) 48.6	125,103.3	(11,185) 60.0	12.6	11,937 (78.6)

parameters are set according to the relevant literature ((De Jong and De Jong 1975; Davis 1985; Ezugwu and Adewumi 2017); see Table 2).

Among these parameters, OX denotes order crossover, ES denotes elitist selection, and RWS denotes roulette wheel selection. The maximum number of generations is determined by a preliminary experiment using instance KroA100. The performance index of GA is shown in Fig. 5.

The GA is repeated 20 times to give an average solution deviation (ASD) and the optimal solution deviation (BSD). The runtime of GA is the average of these 20 executions. The calculation results for ABIA, B&B, and GA are presented in Table 3.



(a) Point Num–Cum Dev curve **(b)** Point Num–Time logarithmic curve

Fig. 4 Computation time and deviation between ABIA and other insertion algorithms with respect to STSP scale

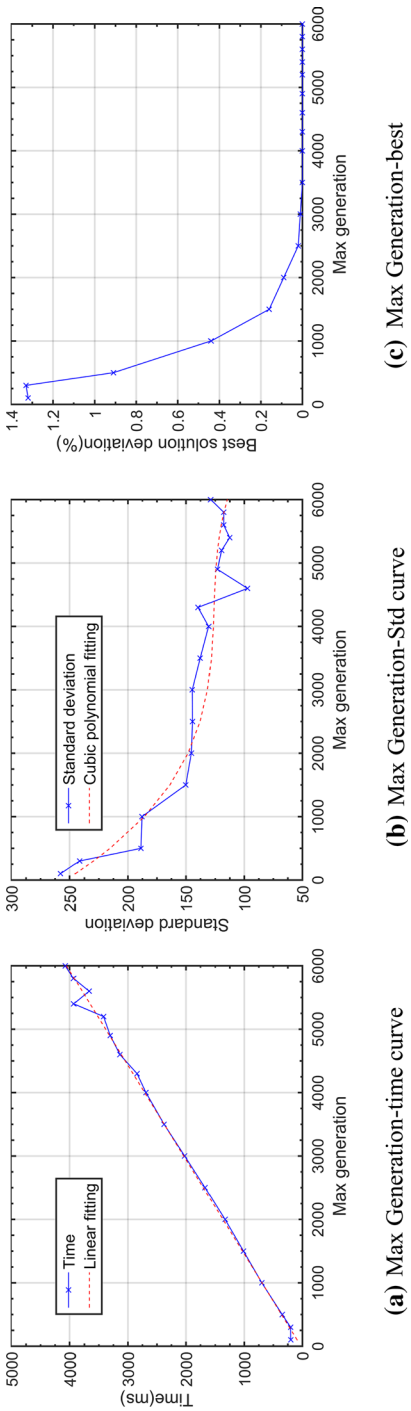
It can be seen from Table 3 that, when the number of points is insignificant, the runtime of B&B is much longer than that of ABIA. The worst time complexity of B&B is $O(n!)$. The B&B runtime is clearly reflected through the three instances indicated in Table 3, so the other instances are not taken into account. Figure 6 shows the trends in runtime and deviation of ABIA and GA with respect to the STSP scale. Meanwhile, Fig. 6a shows Point Num against Cum Dev and Fig. 6b shows Point Num against time.

The following conclusions can be drawn from Table 3 and Fig. 6:

(1) On account of the same results of ABIA no matter how many times it is calculated, ABIA is more stable than GA.
(2) Where there are not more than 40 points, the total deviation between the optimal solution and ABIA is 0.6%, while the deviation between the optimal solution and GA-ASD is 0.71%. When there are not more than 100 points, the total deviation between the optimal solution and ABIA is 48.6%, while the deviation between the optimal deviation and GA-ASD is17.8%. With the increasing number of vertices, the deviation growth rate of ABIA is higher than that of GA.
(3) When there are not more than 100 points, the total runtime of ABIA-STSP is only 3.6% of that of GA algorithm. But with the increase of STSP scale, the runtime of ABIA-STSP will increase faster than that of GA.
(4) The runtime of B&B is much longer than that of ABIA. When there are more than 16 points, B&B takes 10^5 times longer than ABIA.

4.3 ATSP experiment using GA, B&B, and ABIA

Without coordinate data, The ATSP instances in the TSPLIB database contain only the bidirectional distances between vertices. However, as ABIA requires coordinate data, the corresponding coordinates are obtained from the original paper (Gower 1966). In solving ATSP, B&B takes each vertex as the root node. The parameters of



(a) Max Generation-time curve **(b)** Max Generation-Std curve **(c)** Max Generation-best

Fig. 5 Testing GA performance indicators using instance KroA100

Table 2 GA parameters

Name	Parameter
Population size	50
Crossover operator	OX
Mutation operator	Insertion
Crossover probability	0.95
Mutation probability	0.25
Selection operator	ES&RWS
Max generation	3500

GA are similar to those for STSP. The results for ATSP given by ABIA, B&B, and GA are presented in Table 4.

From Table 4, it can be seen that the runtime of B&B for ATSP is similar to that for STSP—much longer than the other algorithms. Figure 7 shows the trends in runtime and deviation for ABIA and GA with respect to ATSP scale.

The following conclusions can be drawn from Table 4 and Fig. 7:

(1) On account of the same results of ABIA no matter how many times it is calculated, ABIA is more stable than GA for ATSP.
(2) The total deviation between ABIA and the optimal solution is 33.4%, and GA-ASD is 145.3%. With the increasing number of vertices, the accuracy of GA will get worse, which is caused by insufficient iterations when vertices increase (the parameters consistent with STSP are adopted for comparison).
(3) The total runtime of ABIA is 1491.8 ms, while that of GA is 11708.0 ms. With the increasing number of vertices, the runtime of ABIA increases faster than that of GA, and the runtime of B&B is much longer than that of ABIA and GA.

5 Conclusion

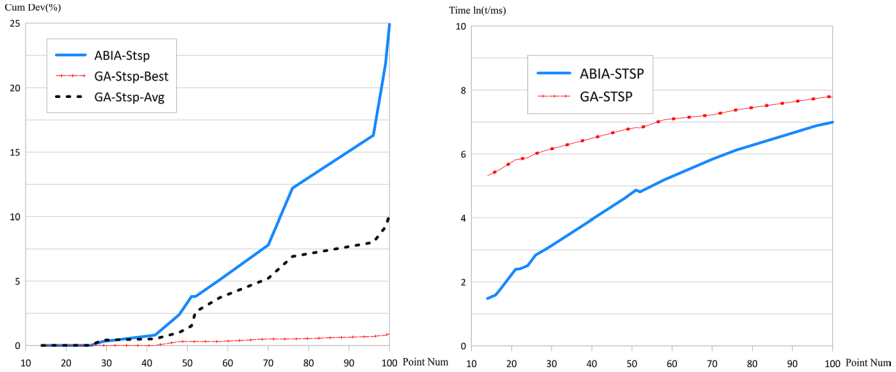
On the basis of the fundamental principle of optimal Hamiltonian loops, ABIA applies an angular bisector to the point group to disperse the edges of loop as much as possible and avoid loop intersections. Based on this angular bisector, the initial loop formation rules and vertex insertion rules for STSP and ATSP are formulated respectively. The angular bisector reflects the global spatial distribution of the point groups. The vertex search strategy based on the angular bisector is a global search, which is helpful to avoid local optima. Comparison experiments with similar insertion algorithms show that ABIA has good overall performance for solving

Table 3 STSP instances calculated by ABIA, B&B, and GA

Instances		ABIA		GA			B&B
Name	Best	Time(ms)	Dev (%)	ASD	BSD (%)	Time(ms)	Time(ms)
burma14	3323	4.4	(0) 0.0	(0) 0.0	(0) 0.0	204.1	10,246.0
ulysses16	6859	4.9	(0) 0.0	(1) 0.01	(0) 0.0	232.7	3,930,341.0
gr17	2085	5.7	(0) 0.0	(0) 0.0	(0) 0.0	246.9	3,455,931.0
gr21	2707	11.0	(0) 0.0	(0) 0.0	(0) 0.0	337.2	–
ulysses22	7013	11.1	(0) 0.0	(0) 0.0	(0) 0.0	343.4	–
gr24	1272	12.3	(0) 0.0	(0) 0.0	(0) 0.0	358.8	–
fri26	937	17.2	(0) 0.0	(0) 0.0	(0) 0.0	408.4	–
bays29	2020	21.6	(6) 0.3	(8) 0.4	(0) 0.0	456.4	–
bayg29	1610	21.1	(5) 0.3	(5) 0.3	(0) 0.0	461.6	–
dantzig42	699	63.0	(6) 0.9	(1) 0.1	(0) 0.0	706.1	–
swiss42	1273	59.2	(0) 0.0	(0) 0.0	(0) 0.0	709.6	–
gr48	5046	99.3	(64) 1.3	(61) 1.2	(40) 0.8	841.4	–
att48	10,628	98.2	(224) 2.1	(43) 0.4	(0) 0.0	854.4	–
hk48	11,461	100.2	(168) 1.5	(5) 0.0	(0) 0.0	882.9	–
eilon51	426	130.2	(6) 1.4	(2) 0.5	(0) 0.0	921.9	–
berlin52	7542	123.1	(0) 0.0	(83) 1.1	(0) 0.0	911.6	–
brazil58	25,395	179.1	(333) 1.3	(279) 1.1	(0) 0.0	1179.2	–
st70	675	342.1	(18) 2.7	(10) 1.5	(1) 0.2	1374.3	–
eil76	538	457.6	(28) 5.2	(8) 1.4	(0) 0.0	1575.0	–
pr76	108,159	454.4	(3888) 3.6	(2163) 2.0	(0) 0.0	1640.1	–
gr96	55,209	979.4	(2309) 4.1	(607) 1.1	(110)0.2	2300.8	–
rat99	1211	1060.8	(67) 5.5	(15) 1.2	(1) 0.1	2428.3	–
kroA100	21,282	1108.3	(534) 2.5	(149) 0.7	(0) 0.0	2400.8	–
kroB100	22,141	1069.4	(857) 3.9	(266) 1.2	(44) 0.2	2441.3	–
kroC100	20,749	1109.1	(331) 1.6	(104) 0.5	(0) 0.0	2400.8	–
kroD100	21,294	1100.6	(479) 2.3	(319) 1.5	(106) 0.5	2433.7	–
kroE100	22,068	1105.0	(591) 2.7	(132) 0.6	(44) 0.2	2473.8	–
rd100	7910	1075.2	(331) 5.4	(71) 0.9	(0) 0.0	2396.6	–
Total within 40 points	–	109.3	(11) 0.6	(14) 0.71	(0) 0.0	16.9	–
Total	–	10,823.5	(10,341) 48.6	(4332) 17.8	(346) 2.2	2824	–

small-scale TSP. It is especially suitable for small-scale applications that require stable and rapid results.

As a new constructive heuristic insertion algorithm, the following aspects of ABIA require further study: (1) The angular bisector reflects the distribution characteristics of point groups, and so ABIA is affected by the distribution of point groups; (2) When the scale of STSP and ATSP is large, it is difficult for a single angular bisector to fully reflect the distribution characteristics of point groups.

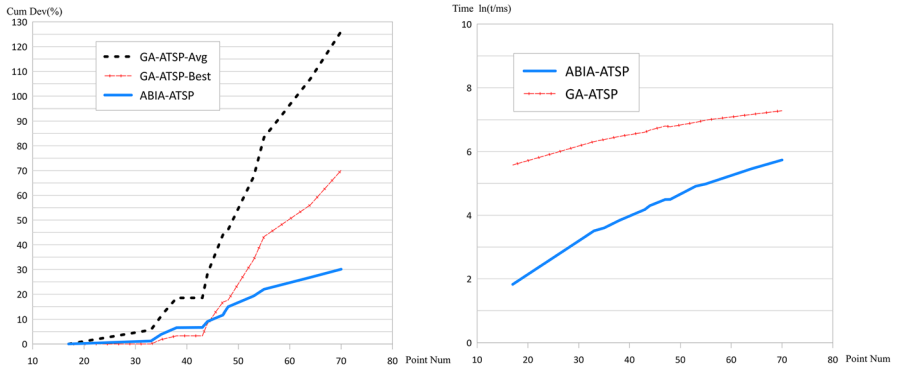


(a) Point Num–Cum Dev curve **(b)** Point Num–Time logarithmic curve

Fig. 6 Computation time and deviation of ABIA and GA with respect to STSP scale

Table 4 ATSP instances calculated by ABIA, B&B, and GA

Instances		ABIA		GA			B&B
Name	Best	Time (ms)	Dev. (%)	ASD	BSD (%)	Time (ms)	Times (ms)
br17	39	6.2	(0) 0.0	(0) 0.0	(0) 0.0	264.8	3,949,598.0
ftv33	1286	33.4	(15) 1.2	(73) 5.7	(0) 0.0	553.1	–
ftv35	1473	36.6	(40) 2.7	(82) 5.6	(27) 1.8	591.6	–
ftv38	1530	46.4	(41) 2.7	(110) 7.2	(23) 1.5	651.2	–
p43	5620	65.3	(6) 0.1	(2) 0.0	(0) 0.0	744.6	–
ftv44	1613	73.8	(39) 2.4	(158) 9.8	(81) 5.0	793.0	–
ftv47	1776	89.5	(46) 2.6	(279) 15.7	(151) 8.5	901.3	–
ry48p	14,422	89.8	(462) 3.2	(288) 2.0	(115) 0.8	881.3	–
ft53	6905	135.9	(304) 4.4	(1491) 21.6	(1139) 16.5	1012.7	–
ftv55	1608	146.0	(43) 2.7	(254) 15.8	(148) 9.2	1087.7	–
ftv64	1839	235.1	(88) 4.8	(432) 23.5	(235) 12.8	1305.9	–
ftv70	1950	305.2	(53) 2.7	(503) 25.8	(343) 17.6	1481.7	–
ft70	38,673	314.5	(1508) 3.9	(4873) 12.6	(3983) 10.3	1439.1	–
Total	–	1491.8	(2645) 33.4	(8545) 145.3	(6245) 84.0	1491.8	–



(a) Point Num–Cum Dev curve **(b)** Point Num–Time logarithmic curve

Fig. 7 Trend of ABIA and GA with respect to ATSP scale

References

Bondy JA, Murty USR (1976) Graph theory with applications. The Macmillan Press Ltd, New York

Davis L (1985) Applying adaptive algorithms to epistatic domains. In: Proceedings of the 9th international joint conference on artificial intelligence. Los Angeles, CA, USA. 162–164

De Jong K, De Jong KA (1975) An analysis of the behavior of a class of genetic adaptive systems. University of Michigan, Michigan

Dirac GA (1952) Some theorems on abstract graphs. Proc London Math Soc s3-2:69–81

Ezugwu AES, Adewumi AO (2017) Discrete symbiotic organisms search algorithm for travelling salesman problem. Expert Syst Appl 87:70–78. https://doi.org/10.1016/j.eswa.2017.06.007

Golden B, Bodin L, Doyle T, Stewart W (1980) Approximate traveling salesman algorithms. Oper Res 28:694–711

Gower JC (1966) Some distance properties of latent root and vector methods used in multivariate analysis. Biometrika 53:325–338

Hore S, Chatterjee A, Dewanji A (2018) Improving variable neighborhood search to solve the traveling salesman problem. Appl Soft Comput J 68:83–91. https://doi.org/10.1016/j.asoc.2018.03.048

Hui W (2012) Comparison of several intelligent algorithms for solving TSP problem in industrial engineering. Syst Eng Proc 4:226–235. https://doi.org/10.1016/j.sepro.2011.11.070

Ismail AH (2019) Domino algorithm: a novel constructive heuristics for traveling salesman problem. IOP Conf Ser Mater Sci Eng. https://doi.org/10.1088/1757-899X/528/1/012043

Jünger M, Reinelt G, Rinaldi G (1995) The traveling salesman problem. Handbooks Oper Res Manag Sci 7:225–330

Kanda J, de Carvalho A, Hruschka E et al (2016) Meta-learning to select the best meta-heuristic for the traveling salesman problem: a comparison of meta-features. Neurocomputing 205:393–406. https://doi.org/10.1016/j.neucom.2016.04.027

Ore O (1960) Note on hamilton circuits. Am Math Mon 67:55

Pan G, Li K, Ouyang A, Li K (2016) Hybrid immune algorithm based on greedy algorithm and delete-cross operator for solving TSP. Soft Comput 20:555–566. https://doi.org/10.1007/s00500-014-1522-3

Rao W, Jin C (2012) An improved greedy heuristic based on solving traveling salesman problem. Oper Res Manag Sci 6:1–9

Reinelt G (1991) TSPLIB—a traveling salesman problem library. ORSA J Comput 3:376–384

Reinelt G (1994) The traveling salesman: computational solutions for TSP applications. Springer-Verlag, Berlin, Heidelberg

Rosenkrantz DJ, Stearns RE, Lewis Ii PM et al (1977) An analysis of several heuristics for the traveling salesman problem * under the title approximate algorithms for the traveling salesperson problem. SIAM J Comput 6:563–581

Sakurai Y, Onoyama T, Kubota S et al (2006) A Multi-world intelligent genetic algorithm to interactively optimize large-scale TSP. Proc 2006 IEEE Int Conf Inf Reuse Integr IRI-2006. https://doi.org/10.1109/IRI.2006.252421

Siemiński A, Kopel M (2019) Solving dynamic TSP by parallel and adaptive ant colony communities. J Intell Fuzzy Syst 37:7607–7618. https://doi.org/10.3233/JIFS-179366

Ursani Z, Corne DW (2016) Introducing complexity curtailing techniques for the tour construction heuristics for the travelling salesperson problem. J Optim 2016:1–15. https://doi.org/10.1155/2016/4786268

Xiang Z, Chen Z, Gao X et al (2015) Solving large-scale TSP using a fast wedging insertion partitioning approach. Math Probl Eng. https://doi.org/10.1155/2015/854218

Yang Z, Xiao MQ, Ge YW et al (2018) A double-loop hybrid algorithm for the traveling salesman problem with arbitrary neighbourhoods. Eur J Oper Res 265:65–80. https://doi.org/10.1016/j.ejor.2017.07.024

Yang L, Hu X, Li K, et al (2019) Nested simulated annealing algorithm to solve large-scale TSP problem. In: International symposium on intelligence computation and applications. Springer, pp 473–487

Zhang ZC, Han W, Mao B (2018) Adaptive discrete cuckoo algorithm based on simulated annealing for solving TSP. Acta Electron Sin 46:1849–1857

Zhong Y, Lin J, Wang L, Hui Z (2018) Discrete comprehensive learning particle swarm optimization algorithm with Metropolis acceptance criterion for traveling salesman problem. Swarm Evol Comput S2210650216304680

Zhu J, Huai L, Cui R (2017) Research and Application of Hybrid Random Selection Genetic Algorithm. In: 2017 10th International Symposium on Computational Intelligence and Design (ISCID). pp 330–333

Authors and Affiliations

Jian Lin[1] · Xiangfei Zeng[1] · Jianxun Liu[1] · Keqin Li[2]

Jian Lin
694586970@qq.com

Jianxun Liu
ljx529@gmail.com

Keqin Li
lik@newpaltz.edu

[1] Key Laboratory Knowledge Processing and Networked Manufacturing, Hunan University of Science and Technology, Xiangtan 411201, China

[2] Department of Computer Science, State University of New York New Paltz, New York 12561, USA