DOI: 10.7641/CTA.2014.40280

混合编码差分进化算法求解含邻域Dubins旅行商问题

辛 斌[1,2†], 陈 杰[1,2], 徐冬玲[3], 陈玉旺[3]

(1. 北京理工大学 自动化学院, 北京 100081; 2. 复杂系统智能控制与决策国家重点实验室, 北京 100081;
3. 曼彻斯特大学 商学院 认知与决策科学研究中心, 英国 曼彻斯特 M15 6PB)

摘要: 含邻域Dubins旅行商问题(DTSPN)是一个具有挑战性的混合变量优化问题, 它源于Dubins车的运动规划, 例如轨迹受曲率约束的高速飞行器. 本文在对DTSPN的相关研究进行综述的基础上, 提出两种混合编码差分进化算法来有效求解DTSPN, 这两种算法分别采用完整编码方案和部分编码方案. 完整编码差分进化算法在整个解空间中搜索最优的Dubins路径, 有利于充分探索搜索空间. 通过对Dubins车在相邻两点间移动时的终端朝向进行松弛, 本文提出一种部分编码差分进化算法, 在解的质量和计算时间方面实现了较好的权衡. 比较性计算实验包含两种差分进化算法以及现有文献中的两种先进DTSPN算法, 实验结果表明基于终端朝向松弛和部分编码的差分进化算法能够以较小的计算代价得到DTSPN的高质量解, 明显优于其他算法.

关键词: Dubins车; 路径规划; 曲率约束; 含邻域Dubins旅行商问题; 差分进化

中图分类号: TP18, O221 **文献标识码**: A

Hybrid encoding based differential evolution algorithms for Dubins traveling salesman problem with neighborhood

XIN Bin[1,2†], CHEN Jie[1,2], XU Dong-ling[3], CHEN Yu-wang[3]

(1. School of Automation, Beijing Institute of Technology, Beijing 100081, China;
2. State Key Laboratory of Intelligent Control and Decision of Complex Systems, Beijing 100081, China;
3. Decision and Cognitive Sciences Research Centre, Manchester Business School,
the University of Manchester, Manchester M15 6PB, UK)

Abstract: The Dubins traveling salesman problem with neighborhood (DTSPN) is a challenging mixed-variable optimization problem, stemming from the motion planning of a Dubins vehicle, e.g. an aircraft moving at a high speed, whose trajectory is restricted by curvature constraints. In this paper, a survey result of DTSPN is firstly provided; then, in order to solve DTSPN efficiently, we propose two hybrid encoding-based differential evolution (DE) algorithms, which adopt complete encoding scheme and partial encoding scheme, respectively. The DE algorithm with complete encoding searches for optimal Dubins tours in the entire solution space, in favor of a sufficient exploration of the search space. By relaxing the terminal heading of a Dubins vehicle when it moves from one point to another, a novel DE with partial encoding is proposed to achieve a better tradeoff between solution quality and computational time. Comparative experiments, involving the two DE algorithms and two state-of-the-art DTSPN algorithms identified in literature, show that the DE based on terminal heading relaxation and partial encoding can find high-quality solutions to DTSPN with lower computation cost, and has remarkable advantages over the other algorithms.

Key words: Dubins' vehicle; path planning; curvature constraint; Dubins traveling salesman problem with neighborhood; differential evolution

1 Introduction

Kinematic constraints of mobile agents such as unmanned aerial vehicles (UAVs) and autonomous underwater vehicles (AUVs) impose severe restrictions on the feasibility of their planned paths[1–3]. Therefore, in the motion planning of mobile agents, it is very important to take the kinematic constraints into account so that the planned paths can be implemented or followed. As a common approximate model for high-speed mobile agents (e.g., fixed-wing aerocrafts), a Dubins vehicle captures a nonholonomic planar vehicle, moving at a constant speed, whose trajectories are twice differentiable curves of bounded curvature. In the literature, the kinematic constraint of a Dubins vehicle is termed as maximal curvature constraint, minimal turning radius constraint, or yaw rate constraint for UAV[2,4–5]. The

Received 5 April 2014; accepted 17 May 2014.
†Corresponding author. E-mail: brucebin@bit.edu.cn; Tel.: +86 10-68912463.
This work is supported by the National Natural Science Foundation of China (No. 61304215), the Projects of Major International (Regional) Joint Research Program of NSFC (No. 61120106010); the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (No. 61321002), and the Beijing Outstanding Ph.D. Program Mentor (No. 20131000704).

dynamics of a Dubins vehicle can be described as follows:

$$\begin{cases} \dot{x} = v\cos\,\theta, \\ \dot{y} = v\sin\,\theta, \\ \dot{\theta} = \dfrac{v}{r}u,\ u \in [-1,1], \\ \dot{v} = 0, \end{cases} \tag{1}$$

where (x, y) and θ are the planar coordinate and the heading of the vehicle respectively, constituting the configuration of a Dubins vehicle (also termed Dubins state); v and r are the speed and the minimal turning radius of the vehicle respectively; and u is the available control.

It is well known that there are six possible shortest path patterns, termed Dubins paths, for any transition from one Dubins state to another: {LSL, RSR, LSR, RSL, LRL, RLR} where L means turning left with the minimal turning radius ($u = 1$), R means turning right with the minimal turning radius ($u = -1$), both L and R lead to circular arcs, and S means moving along a straight line ($u = 0$)[6]. Although it is argued that the bang-bang control with respect to the optimal path may be impractical due to instantaneous step changes of the control input, the issue can be alleviated by choosing a larger minimal turning radius to make vehicles follow the planned trajectories with only small excursions at transition points[7]. Recently, Sanfelice et al. extended the results and derived the necessary conditions for minimum-time curvature-constrained tours[8].

The Dubins traveling salesman problem with neighborhood[9–10] is a generalized variant of the traveling salesman problem for a Dubins vehicle (DTSP) which has been proved NP-hard[11–12]. In DTSPN, the vehicle is expected to pass multiple regions in a shortest time. Assume that the vehicle has n regions to visit, which are denoted by $\mathcal{R}_1$,$\mathcal{R}_2$,$\cdots$,$\mathcal{R}_n$, respectively. A general mathematical formulation of DTSPN is given as follows:

$$\begin{cases} \min\ D(\boldsymbol{\pi}, \boldsymbol{z}, \boldsymbol{h}) = \\ \displaystyle\sum_{i=0}^{n-1} d(z_{\pi_i}, z_{\pi_{i+1}}, h_{\pi_i}, h_{\pi_{i+1}}, r) + \\ d(z_{\pi_n}, z_0, h_{\pi_n}, h_0', r), \\ \boldsymbol{\pi} = [\pi_1\ \ \pi_2\ \ \cdots\ \ \pi_n],\ \pi_i \in \{1, 2, \cdots, n\}, \\ \boldsymbol{z} = [z_1\ \ z_2\ \ \cdots\ \ z_n],\ z_i \in \mathcal{R}_i,\ i = 1, 2, \cdots, n, \\ \boldsymbol{h} = [h_1\ \ h_2\ \ \cdots\ \ h_n;\ h_0'], \\ h_i \in [0, 2\pi),\ i = 1, 2, \cdots, n. \end{cases} \tag{2}$$

where $D(\boldsymbol{\pi}, \boldsymbol{z}, \boldsymbol{h})$ is the total length of a Dubins tour that needs to be minimized; $\boldsymbol{\pi}$ is a permutation of all regions, and $\pi_0 = 0$ corresponds to the starting point; z_i is a waypoint chosen from $\mathcal{R}_i (i = 1, 2, \cdots, n)$; h_i is the heading of the vehicle at the waypoint $z_i (i = 1, 2, \cdots, n)$; in some cases (e.g., in UAV's reconnaissance task), the heading of the vehicle may be fixed or specified within a small range for the purpose of observing targeted objects in an appropriate direction[13]; $[z_i, h_i]$ is the configuration of the vehicle; z_0 and h_0 are the initial position and the initial heading of the vehicle, respectively; h_0' is the heading of the vehicle when returning to the starting position; if it is not required to return to the origin, the final term of the objective function will be removed; besides, h_0' may be fixed in some scenarios even if the vehicle has to return to the origin[14]. In both cases, h_0' will not be included in the vector of decision variables $\boldsymbol{h}$; $d(z_{\pi_i}, z_{\pi_{i+1}}, h_{\pi_i}, h_{\pi_{i+1}}, r)$ is the Dubins distance between the two configurations $[z_{\pi_i}, h_{\pi_i}]$ and $[z_{\pi_{i+1}}, h_{\pi_{i+1}}]$, i.e., the length of the shortest Dubins path, which can be derived directly from the conclusions in the literature [6]; r is the minimal turning radius of the vehicle.

The difference between DTSP and DTSPN lies in that the vehicle is required to pass specified waypoints in DTSP rather than regions in DTSPN. The goal of DTSP is to find the shortest Dubins tour passing all waypoints by optimizing the visiting order $\boldsymbol{\pi}$ and the heading $\boldsymbol{h}$ of the vehicle at each waypoint, without considering the choice of the waypoints (i.e., the optimization w.r.t. $\boldsymbol{z}$). Generally, DTSPN combines the features of DTSP and traveling salesman problem with neighborhood (TSPN)[15–16]. From the perspective of problem solving, DTSPN poses more challenge for the motion planning of mobile agents. As compared with DTSP, very few studies regarding DTSPN are reported in the literature. In the following, we will review previous research about both DTSPN and DTSP since they share much similarity in model features and resolution methods.

The main difficulty of solving DTSP or DTSPN stems from the dependence of the point-to-point distance on the headings of the vehicle, which need to be optimized together with the visiting order of all waypoints[17–18]. In contrast, for most common traveling salesman problems (TSPs), e.g., the Euclidean TSP[19–20], the distance between any two waypoints can be determined prior to optimization, and the focus of finding the shortest path is on determining the best visiting order of waypoints. Generally, the resolution methods for DTSP can be grouped into three categories: decoupling methods (decomposition strategy), transformation methods, and direct search methods.

The decoupling methods separate the optimization of the visiting order of waypoints from vehicle headings. For example, the alternating algorithm, proposed by Savla et al., first identifies the visiting order by solving a Euclidean TSP which measures the distance

between any two waypoints by Euclidean metric, neglecting the curvature constraint[1,11,21]. At the second step of the alternating algorithm, the vehicle headings will be determined by a simple rule which retains all odd-numbered edges and replaces all even-numbered edges with minimum-length Dubins paths preserving the identified visiting order. Similar decoupling methods which determine the visiting order based on a Euclidean TSP were also adopted in the works[4,14,22]. However, these works vary with the idea of determining the vehicle headings and generating the final Dubins tour. For example, Rathinam et al. employed Christofides approximation algorithm to determine the visiting order, and the vehicle headings are derived from a constructive procedure which relaxes the terminal heading for any point-to-point movement[4]. Oberlin et al. applied a split dual algorithm to identify the visiting order first, and then employed dynamic programming to obtain the vehicle headings, finally producing a Dubins tour[22]. Macharet et al. proposed a three-stage decoupling method to solve DTSPN, which determines sequentially the position of waypoints, vehicle headings, and finally the order of visiting different regions[23]. Yazici constructs a Dubins tour by determining the visiting order to form a coarse tour which is then smoothed by use of Dubins path patterns[24].

One obvious drawback of the decoupling methods is that solving ETSP cannot guarantee the optimality of the visiting order, and the obtained order separated from vehicle headings may be far from desirable. The performance and effectiveness of the decoupling methods have a strong dependence on the similarity between DTSP and its ETSP counterpart, which requires a weak coupling between the visiting order and vehicle headings. Recently, Goaoc et al. tried to use convex optimization methods to derive the optimal Dubins tour with a given visiting order; however, it is required that the distance between consecutive waypoints is large enough w.r.t. the minimum turning radius of vehicles[17].

The transformation methods aim to find a solution to DTSP by solving a non-Euclidean TSP which is transformed from the DTSP. The distance between any two waypoints is measured by the length of the shortest Dubins path instead of the Euclidean metric. In order to derive the distance between any two waypoints, the vehicle heading at each waypoint has to be determined beforehand. Le Ny and Feron proposed two simple transformation methods to solve DTSP[18]. The first method fixes all vehicle headings at zero while the second method employs randomized headings. In both cases, all Dubins distances can be computed with predetermined headings, and the resulting TSP, an asymmetric TSP (ATSP) built on a directed graph, is solved by Helsgaun's implementation of the Lin-Kernighan heuristic[25]. It was demonstrated that the randomized version of Le Ny and Feron's method performs generally better than the alternating algorithm proposed by Savla et al.[21], especially in scenarios where the waypoints are densely distributed[2,18,26]. Le Ny also generalizes the two simple methods by discretizing the vehicle headings and formulating a one-in-a-cluster TSP, also known as generalized TSP (GTSP)[27], which is further transformed into an ATSP[2]. The same idea was also adopted by Oberlin et al.[19,28]. A common virtue of the transformation methods and decoupling methods is that efficient TSP solvers such as the Concorde TSP solver and the Lin-Kernighan heuristic (see [29]) can be used to provide a high-quality solution to the resulting TSP.

The direct search methods solve DTSP by optimizing the visiting order (permutation variables) and vehicle headings (continuous variables) simultaneously in the search space of mixed variables. Various optimization algorithms can be applied to achieve the direct search[30]. Obviously, this category of DTSP resolution methods does not sacrifice DTSP's model accuracy by approximation or transformation since the coupling between the visiting order and vehicle headings is taken into account. The challenge with direct search methods is how to carry out an efficient optimization in a larger search space of the mixed variables.

As for DTSPN, Obermeyer et al. proposed a genetic algorithm based direct search method for the path planning of a UAV performing reconnaissance of static ground targets which is formulated into a DTSPN[9]. In contrast, transformation methods were more frequently adopted to solve DTSPN. Obermeyer et al. proposed two sampling-based roadmap methods to solve a polygon-visiting Dubins traveling salesman problem which is de facto a DTSPN[31–32]. Both methods sample a finite set of candidate entry poses composed of the waypoints and the vehicle headings at the boundary of each region. The first method, named resolution complete method, transforms the DTSPN into a GTSP, and further uses Noon-Bean transformation to convert it into an ATSP. In contrast, the second method directly constructs an ATSP which treats each region as a graph vertex and uses the average distance of all candidate directed edges between two regions as an inter-region distance. The second method can be categorized as a decoupling method. Zhang et al. also adopted a sampling-based transformation method to construct a roadmap and then solve a resulting ATSP by heuristic search[33]. Isaacs et al. proposed a multi-step transformation method for DTSPN[10], which is similar to the first method proposed by Obermeyer et al.[31]. Han-

son et al. proposed a transformation method to solve a reduced DTSPN in which a UAV is required to visit multiple targets in an order specified a priori, and the field-of-view of sensors equipped on UAVs is considered to achieve target observations[13]. The DTSPN is transformed into a GTSP by discretizing sensors' footprint.

In this paper, we proposed two versions of random-key differential evolution algorithms to solve DTSPN. It is the first effort to use the differential evolution algorithm to solve DTSPN, though the algorithm has been proven very powerful in global optimization and evolutionary computation[34]. The first differential evolution algorithm is a direct search method without sacrificing DTSPN's model accuracy by approximation or transformation. The second one is proposed to reduce the computational cost by relaxing the vehicle headings and reducing the dimension of search space. Comparative experiments show that the second algorithm has excellent performance in terms of both solution quality and computational cost.

The paper is structured as follows. Section 2 gives a detailed DTSPN formulation which can describe the path-planning problem originating from UAV's multi-point operations (e.g., information collection or delivery). Section 3 presents two hybrid-encoding-based differential evolution algorithms to solve the DTSPN. One is a direct search method and the other is a heading-relaxation based method. Section 4 presents experimental results and performance comparisons. Section 5 concludes the paper.

2 Problem formulation

The DTSPN formulation shown in (2) is just a template. As for solving DTSPN, the neighborhood for waypoints has to be determined beforehand. For example, Obermeyer et al. define the neighborhood as a polygon for UAV's visual reconnaissance in urban terrain[9]. Hanson et al. define the neighborhood as a sector to fit the footprint of sensors equipped on UAVs[13]. In this paper, the neighborhood of each waypoint is defined as a disk centered at the waypoint. The disk-shaped neighborhood is especially suitable for capturing the effective communication range of static wireless platforms such as ground units carrying wireless communication devices in open areas. In particular, a specific scenario for the DTSPN can be described as follows. A Dubins vehicle is required to visit n disk-shaped regions in order to collect data from or/and deliver data to each static wireless platform located at the center of its corresponding region (neighborhood). Besides, the vehicle is also required to pass m specified points of interest in a given order after it visits all regions and before it returns to the origin. The specified points can be regarded as surveillance sites whose visiting precedence is designated a priori. The formulation for the specific DTSPN is presented as follows.

$$\begin{cases} \min\ D(\boldsymbol{\pi}, \boldsymbol{z}, \boldsymbol{h}) = \\ \sum\limits_{i=0}^{n-1} d(z_{\pi_i}, z_{\pi_{i+1}}, h_{\pi_i}, h_{\pi_{i+1}}, r) + \\ d(z_{\pi_n}, z'_1, h_{\pi_n}, h'_1, r) + \\ \sum\limits_{i=1}^{m-1} d(z'_i, z'_{i+1}, h'_i, h'_{i+1}, r) + \\ d(z'_m, z_0, h'_m, h'_0, r), \\ \boldsymbol{\pi} = [\pi_1\ \ \pi_2\ \cdots\ \pi_n],\ \pi_i \in \{1, 2, \cdots, n\}, \\ \boldsymbol{z} = [z_1\ \ z_2\ \cdots\ z_n],\ z_i = (x_i, y_i) \in \mathcal{R}_i, \\ \mathcal{R}_i = \{z | d(z, o_i) \leqslant R\},\ i = 1, 2, \cdots, n, \\ \boldsymbol{h} = [h_1\ \ h_2\ \cdots\ h_n;\ h'_1\ \ h'_2\ \cdots\ h'_m;\ h'_0], \\ h_i \in [0, 2\pi),\ i = 1, 2, \cdots, n. \end{cases} \tag{3}$$

where $z'_1, z'_2, \cdots, z'_m$ are m points of interest which will be visited in sequence by the vehicle, $o_i (i \in \{1, 2, \cdots, n\})$ is the center of the ith region (i.e., the position of the ith wireless platform), $d(z, o_i)$ is the Euclidean distance between z and o_i, R is the radius of each region, and the remaining denotations are the same as those in (2). Note that the decision variables include three parts: the permutation of n regions ($\boldsymbol{\pi}$), the visiting positions ($\boldsymbol{z}$), and the vehicle headings at all visiting positions and points of interest ($\boldsymbol{h}$). Obviously, the DTSPN is a mixed-variable optimization problem since $\boldsymbol{\pi}$ is a combinatorial variable while $\boldsymbol{z}$ and $\boldsymbol{h}$ are continuous-valued variables. On one hand, the combinatorial component disables applying traditional gradient-based optimization methods to solve the DTSPN. On the other hand, the optimization of continuous-valued components distinguishes the DTSPN from common TSPs, which makes it impossible to use prevailing TSP problem-solvers to resolve DTSPN directly. Therefore, for the simultaneous optimization of the different types of variables, new techniques and algorithms to deal with the mixed variables have to be designed.

3 Differential evolution for DTSPN

As a population-based optimizer, differential evolution has very good reputation in numerical optimization, having been widely applied in scientific and engineering research[35]. DE has shown excellent performance in many numerical optimization competitions including unconstrained single-objective optimization, constrained optimization, and multi-objective optimization[34]. Generally, DE includes three operations: differential mutation, crossover and selection, following the framework of genetic algorithms. The differential

mutation that is based on the difference of individuals (solutions) is the main novelty of DE which is inspired by the Nelder-Mead Simplex method[35]. One of the most successful DE variants is DE/rand/1/bin. The following section provides a detailed description of DE/rand/1/bin.

3.1 DE/rand/1/bin

In each generation of the evolution, each individual in the population sequentially goes through differential mutation, binominal crossover, and one-to-one tournament selection. As shown in (4), the differential mutation takes a randomly selected individual ($\boldsymbol{x}_{r1}$) as a base vector and uses two different individuals ($\boldsymbol{x}_{r2}$ and $\boldsymbol{x}_{r3}$, $r1 \neq r2 \neq r3$) from the current population to form a difference vector. A mutant individual ($\boldsymbol{v}_i$) is formed by the combination of the base vector and the difference vector.

$$\boldsymbol{v}_i = \underbrace{\boldsymbol{x}_{r1}}_{\text{base}} + F \cdot \underbrace{(\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3})}_{\text{difference}}, \tag{4}$$

where F is the so-called scaling factor, a general setting for this factor is $F \in (0, 2)$, and Storn and Price suggest $F \in [0.5, 1]$[35].

Then, the mutant individual $\boldsymbol{v}_i = [v_{i,1}\ \ v_{i,2} \cdots v_{i,D}]$ (D is the dimension of search space) will be combined with the original individual $\boldsymbol{x}_i = [x_{i,1}\ \ x_{i,2}\ \cdots\ x_{i,D}]$ to form a trial vector $\boldsymbol{u}_i = [u_{i,1}\ \ u_{i,2}\ \cdots\ u_{i,D}]$ as follows:

$$u_{i,d} = \begin{cases} v_{i,d}, \text{ if } \mathrm{rand}_i^d \leqslant \mathrm{CR} \text{ or } d = rn_i, \\ x_{i,d}, \text{ otherwise.} \end{cases} \tag{5}$$

where CR is the predefined crossover probability which is usually set to a fixed value in the interval (0,1) or changes dynamically within (0,1), rand_i^d is a random number generated within (0,1), and rn_i is a number randomly selected from the index set $\{1, 2, \cdots, D\}$ and used to ensure that the trial vector $\boldsymbol{u}_i$ is different from the original individual $\boldsymbol{x}_i$.

Finally, the trial vector will compete against the original vector to survive into the next generation [see (6)]. The one-to-one tournament selection guarantees that the fitness (objective value) of each individual does not deteriorate.

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{u}_i, \text{ if } f(\boldsymbol{u}_i) \leqslant f(\boldsymbol{x}_i), \\ \boldsymbol{x}_i, \text{ otherwise,} \end{cases} \tag{6}$$

here $f(\cdot)$ is the objective function to be minimized.

3.2 DE with complete encoding (DE–CE)

As a first step to adapt DE to solve DTSPN, one has to employ a suitable encoding scheme. The encoding should cover the selection of waypoints, the ordering of waypoints as well as the determination of vehicle headings at all waypoints. For DE to achieve direct search in the problem space of DTSPN, we employ a complete solution encoding $\boldsymbol{x} = [\boldsymbol{z}\ \ \boldsymbol{r}\ \ \boldsymbol{h}]$ where $\boldsymbol{z}$ and $\boldsymbol{h}$ are defined in (3) and $\boldsymbol{r} = [r_1\ \ r_2\ \cdots\ r_n]$ ($r_i \in [0, 1]$ for $\forall i \in \{1, 2, \cdots, n\}$) is a vector of random keys which is used to determine the visiting order of waypoints. The random key representation of a permutation originates from Bean's work on genetic algorithms to solve sequencing problems[36]. In order to translate (decode) a random key vector into a real permutation, the elements of the vector will be sorted and the indexes of the sorted elements in the vector will be used to generate a permutation. For example, the elements of the vector $\boldsymbol{r} = [0.3\ \ 0.1\ \ 0.2\ \ 0.7\ \ 0.5]$ will be sorted in a descending order as 0.7, 0.5, 0.3, 0.2, 0.1 whose indexes in the original vector are 4, 5, 1, 3 and 2, respectively. Therefore, the resulting permutation is $4-5-1-3-2$.

Since $\boldsymbol{z}$ and $\boldsymbol{h}$ are continuous-valued variables, they can be integrated into the solution encoding directly, without any decoding procedure. However, since each component of $\boldsymbol{z}$ can only be located within its corresponding region, i.e., $\mathcal{R}_i (i \in \{1, 2, \cdots, n\})$ in (3), and the components of $\boldsymbol{h}$ take values from $[0, 2\pi)$, extra operations are expected to guarantee that any new solutions generated in the evolutionary process of DE will not go beyond feasible regions. As the encoding scheme involves both direct encoding and random key encoding, we also term the encoding as a hybrid encoding.

Assume that DE maintains NP individuals in the population. Denote the ith individual by $\boldsymbol{x}_i = [\boldsymbol{z}_i\ \ \boldsymbol{r}_i\ \ \boldsymbol{h}_i]$ with

$$\begin{aligned} &\boldsymbol{z}_i = [z_{i,1}\ \ z_{i,2}\ \cdots\ z_{i,n}], \\ &z_{i,j} = (x_{i,j}, y_{i,j}),\ j = 1, 2, \cdots, n, \\ &\boldsymbol{r}_i = [r_{i,1}\ \ r_{i,2}\ \cdots\ r_{i,n}] \end{aligned}$$

and

$$\boldsymbol{h}_i = [h_{i,1} h_{i,2}\ \cdots\ h_{i,n};\ h'_{i,1}\ \ h'_{i,2}\ \cdots\ h'_{i,m};\ h'_{i,0}].$$

Then, the procedure of the proposed DE for DTSPN is presented as follows.

Step 1 Initialization. Randomly generate NP individuals satisfying $z_{i,j} \in \mathcal{R}_j$, $r_{i,j} \in [0, 1]$, $h_{i,j}$, $h'_{i,k} \in [0, 2\pi)$ for $\forall i, j, k$. Decode $\boldsymbol{r}_i$ into its corresponding permutation $\boldsymbol{\pi}_i (i = 1, 2, \cdots, NP)$. For the ith solution($i = 1, 2, \cdots, NP$) which is determined by $\boldsymbol{z}_i$, $\boldsymbol{\pi}_i$ and $\boldsymbol{h}_i$, evaluate its objective value, i.e. the total length of a Dubins tour.

Step 2 Main loop. For $i = 1$ to NP, apply the differential mutation [see (4)] to the ith individual $\boldsymbol{x}_i = [\boldsymbol{z}_i\ \ \boldsymbol{r}_i\ \ \boldsymbol{h}_i]$ to generate a mutant individual $\boldsymbol{v}_i = [\boldsymbol{z}_i^v\ \ \boldsymbol{r}_i^v\ \ \boldsymbol{h}_i^v]$ where

$$\begin{aligned} &\boldsymbol{z}_i^v = [z_{i,1}^v\ \ z_{i,2}^v\ \cdots\ z_{i,n}^v], \\ &\boldsymbol{r}_i^v = [r_{i,1}^v\ \ r_{i,2}^v\ \cdots\ r_{i,n}^v] \end{aligned}$$

and

$$\boldsymbol{h}_i^v = [h_{i,1}^v \ h_{i,2}^v \ \cdots \ h_{i,n}^v; \ h_{i,1}^{v'} \ h_{i,2}^{v'} \ \cdots \ h_{i,m}^{v'}; \ h_{i,0}^{v'}].$$

Then, the following three operations are implemented to ensure that $\boldsymbol{v}_i$ is a feasible individual.

$$z_{i,j}^v = o_j + \frac{R}{L}(z_{i,j}^v - o_j), \text{ if } L > R, \tag{7}$$

$$r_{i,j}^v = \max\{0, \min\{1, r_{i,j}^v\}\}, \tag{8}$$

$$h_{i,j}^v = h_{i,j}^v \bmod 2\pi, \tag{9}$$

$$h_{i,k}^{v'} = h_{i,k}^{v'} \bmod 2\pi, \tag{10}$$

where $j = 1, 2, \cdots, n, \ k = 0, 1, \cdots, m, \ L = d(z_{i,j}^v, o_j)$, and R is the radius of each region as defined in (3).

Apply the crossover [see (5)] to the mutant vector $\boldsymbol{v}_i$ corrected by (7)−(10) to generate a trial vector $\boldsymbol{u}_i$. Decode $\boldsymbol{u}_i$ into a real solution and evaluate its objective value. If $\boldsymbol{u}_i$ is no worse than $\boldsymbol{x}_i$, update $\boldsymbol{x}_i$ by $\boldsymbol{u}_i$[see (6)].

Step 3 If termination criteria are satisfied, output the discovered shortest path and related results. Otherwise, go to Step 2.

Remark 1 The DE based on complete encoding needs to optimize $4n + m + 1$ variables. Besides, the evaluation of objective function is achieved by computing the Dubins distances between any two neighboring waypoints according to the conclusions in Dubins' work. As demonstrated later, even given a smaller n, the time cost of the DE could be very high. In the following section, we propose another scheme to employ DE to solve DTSPN with obviously reduced time cost.

3.3 DE with heading relaxation and partial encoding (DE–HRPE)

As mentioned in the introduction, the difficulty of solving DTSPN mainly stems from the dependence of point-to-point distance on vehicle headings. In order to reduce the time complexity of DE to solve DTSPN, we relax the terminal vehicle heading for any point-to-point movement, and derive the shortest point-to-point path for a Dubins vehicle, with a given initial heading and free terminal heading, moving from one point to a specified destination.

For any point-to-point movement of a Dubins vehicle, denote by Z^0 the starting vehicle position and by Z^1 the terminal vehicle position. Build a coordinate system XOY with Z^0 being its origin ($O = Z^0$) and the initial vehicle heading being the direction of Y-axis. Denote by $Z = (Z_{\rm X}, Z_{\rm Y})$ the coordinate of the terminal position Z^1 in the coordinate system XOY. Denote by $O_{\rm L}$ the center coordinate of the vehicle's minimal left-turning circle w.r.t. O(see Fig.1). Denote by $O_{\rm R}$ the center coordinate of its minimal right-turning circle w.r.t. O. The optimal point-to-point Dubins path pattern to reach Z^1 is pointed out in the following theorems.

Theorem 1 If $Z_{\rm X} > 0$ and $d(Z, O_{\rm R}) > r$, then the optimal Dubins path is a circular arc plus a straight line segment. The vehicle first turns right along a minimal right-turning circle, and then moves along a tangent line to reach Z^1. In particular, the optimal Dubins path is reduced to a circular arc when $Z_{\rm X} > 0$ and $d(Z, O_{\rm R}) = r$.

Theorem 2 If $Z_{\rm X} < 0$ and $d(Z, O_{\rm L}) > r$, then the optimal Dubins path is a circular arc plus a straight line segment. The vehicle first turns left along a minimal left-turning circle, and then moves along a tangent line to reach Z^1. In particular, the optimal Dubins path is reduced to a circular arc when $Z_{\rm X} < 0$ and $d(Z, O_{\rm L}) = r$.

Theorem 3 If $Z_{\rm X} = 0$ and $Z_{\rm Y} \geqslant 0$, then the optimal Dubins path is a straight line segment stretching from Z^0 to Z^1.

Theorem 4 If $Z_{\rm X} = 0$ and $Z_{\rm Y} < 0$, then the optimal Dubins path is a circular arc plus a straight line segment. The vehicle first turns left or right along its minimal turning circle, and then moves along a tangent line to reach Z^1.

Theorem 5 If $d(Z, O_{\rm R}) < r$, then the optimal Dubins path is a combination of two circular arcs. The vehicle first turns left along a minimal left turning circle, and then turns right along a minimal right turning circle to reach Z^1.

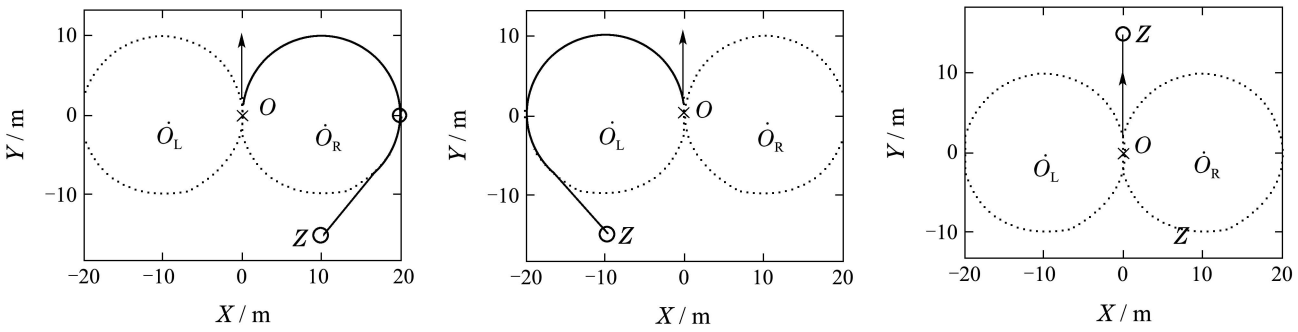
Theorem 6 If $d(Z, O_{\rm L}) < r$, then the optimal Dubins path is a combination of two circular arcs. The vehicle first turns right along a minimal right turning circle, and then turns left along a minimal left turning circle to reach Z^1.

Remark 2 For computing the optimal point-to-point Dubins paths, the above theorems can be reduced to four cases as described in Theorems 1, 2, 5 and 6. The optimal paths described in Theorems 3 and 4 can be regarded as special cases of Theorem 1 or 2. The proofs for Theorems 1, 2, 5 and 6 can be found in the reference [37].

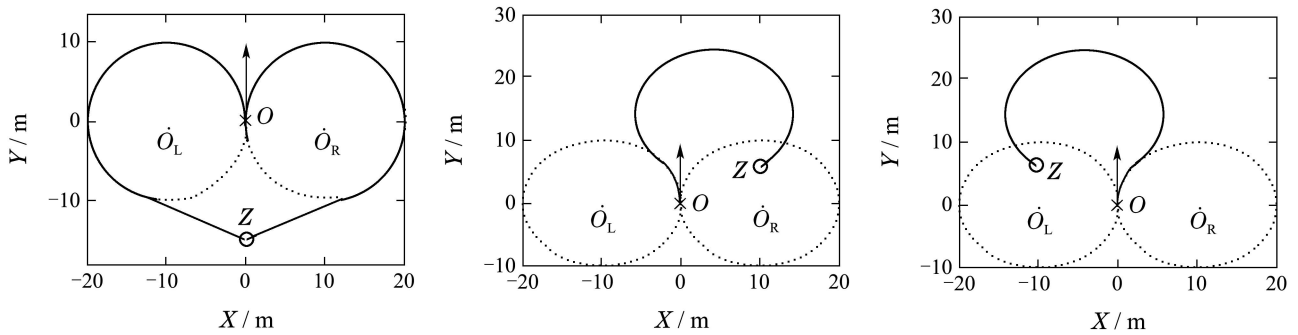
Figure 1 provides an illustration for different optimal Dubins paths. Theorems 1-6 provide a way of obtaining optimal point-to-point Dubins paths in the sense of terminal heading relaxation. Obviously, the terminal vehicle heading can be determined together with the corresponding optimal Dubins path (see Fig.1). The terminal heading of the current path will be taken as the initial heading of the next path. Once the positions and visiting order of waypoints and the vehicle heading at the origin are given, optimal point-to-point Dubins paths can be generated from Theorems 1−6. A piecewise optimal Dubins tour can be constructed by successively determining each optimal Dubins path be-

tween any two neighboring waypoints, in the sequence from the origin to the final waypoint. In this way, DE does not optimize vehicle headings but focuses on optimizing the positions of waypoints and their visiting order. Besides, each optimal Dubins path can be identified easily by checking the relative position of the terminal point with reference to the starting left-turning and right-turning circles, without the need to compute all candidate paths. Therefore, the terminal heading relaxation scheme not only reduces the search space but also simplifies the computation of point-to-point distance.

In Fig.1, The point marked with a cross is the vehicle's starting position. The point marked with a small circle is the terminal point. The arrows indicate the vehicle's initial heading, the dashed rings represent the turning circles, and solid curves show the optimal paths.The turning radius is 10 m. In (d), two optimal paths exist as stated in Theorem 4.



(a) $Z_X > 0$ and $d(Z, O_R) \geqslant r$ (b) $Z_X < 0$ and $d(Z, O_L) \geqslant r$ (c) $Z_X = 0$ and $Z_Y \geqslant 0$



(d) $Z_X = 0$ and $Z_Y < 0$ (e) $d(Z, O_R) < r$ (f) $d(Z, O_L) < r$

Fig. 1 Optimal point-to-point Dubins paths with terminal heading relaxation

Since terminal headings will not be optimized, the encoding scheme in the second DE algorithm can be described as $\boldsymbol{x} = [\boldsymbol{z} \ \ \boldsymbol{r}]$ where $\boldsymbol{z}$ and $\boldsymbol{r}$ are the same as in the complete encoding scheme explained in Section 3.2. DE in this case only needs to search in a $3n$-dimensional space. The procedure of the DE with partial encoding is the same as that of the DE with complete encoding except that vehicle headings are not optimized in the partial encoding based DE and the computation of a Dubins tour's length is obviously simplified. To save space, the procedure of the second DE algorithm is not presented. It is worth noting that terminal heading relaxation implicitly reduces the range of vehicle headings, resulting in a possibility of being unable to discover the real global optimal Dubins tour. However, it can be guaranteed that the Dubins tour finally obtained by the second DE is at least a piecewise optimal one.

4 Computational experiments

The following experiments provide a performance comparison of four DTSPN algorithms, including the two DE algorithms proposed in this paper (DE–CE and DE–HRPE), a variant of the genetic algorithm proposed in the reference [9] and a sampling-based transformation method similar to the one proposed in the reference [31]. The genetic algorithm (GA) for comparison adopts the complete encoding shown in Section 3.2, the same operators as those employed in the reference [9], and the same initialization method as the two DE algorithms. The sampling-based transformation method uniformly samples multiple points along the boundary of each region, and the corresponding vehicle heading at each sampling point is randomly generated. The operations in the sampling process are shown as follows:

$$\varphi_{i,0} = 2\pi \cdot \mathrm{rand}_{i,1}, \ \varphi_{i,j} = \varphi_{i,0} + \frac{2\pi j}{n_s}, \tag{11}$$

$$\theta_{i,j} = -\frac{\pi}{2} + \pi \cdot \mathrm{rand}_{i,j,2}, \tag{12}$$

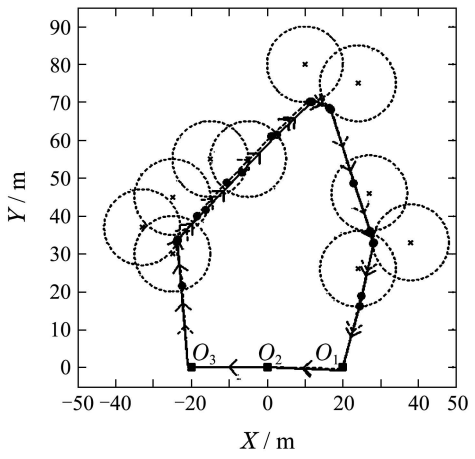
$$h_{i,j} = \varphi_{i,j} + \theta_{i,j} + \pi, \tag{13}$$

$$z_{i,j} = o_i + R \cdot [\cos\varphi_{i,j} \ \ \sin\varphi_{i,j}], \tag{14}$$

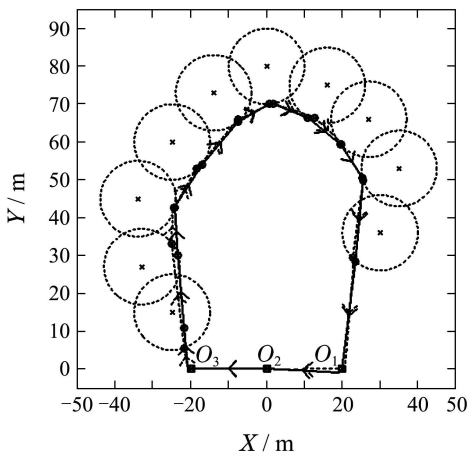
$$i = 1, 2, \cdots, n, \ j = 1, 2, \cdots, n_s,$$

where $\varphi_{i,0}$ is the reference phase angle corresponding to the region $\mathcal{R}_i$ w.r.t. its center o_i , $\varphi_{i,j}$ is the phase angle of the jth sampling point on the boundary of $\mathcal{R}_i$, $n_{\rm s}$ is the number of sampling points on the boundary of each region, $z_{i,j}$ is the jth sampling point on the boundary of $\mathcal{R}_i$, $\theta_{i,j}$ is the relative vehicle heading at the jth sampling point on the boundary of $\mathcal{R}_i$, $h_{i,j}$ is the vehicle heading at the jth sampling point on the boundary of $\mathcal{R}_i$, $\mathrm{rand}_{i,1}$ and $\mathrm{rand}_{i,j,2}$ are random numbers generated within (0,1), and the definition of R and o_i can be found in (3). All sampling points on the boundary of the same region constitute a cluster, and the Dubins distance between any two points belonging to different clusters can be computed according to the conclusions in Dubins' work. Thus, a GTSP can be formulated and further transformed into an ATSP by Noon-Bean transformation[31]. The resulting ATSP is solved by the state-of-the-art TSP solver LKH[25]. On the whole, the sampling-based transformation method shares the same idea with the resolution-complete algorithm proposed by Obermeyer et al.[31].

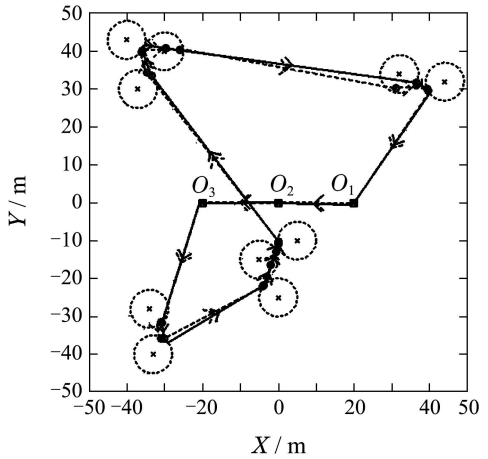
The comparative experiments are organized into two parts. The first part uses eight typical DTSPN instances with distinct features to test different algorithms (see Fig.2). The second part aims at carrying out a general comparison based on more randomly generated DTSPN instances. Each instance in the first part involves ten regions and three points of interest for a Dubins vehicle to visit. The effect of varying the vehicle's turning radius on path planning is also considered in the first part. In the second part, the region centers and the points of interest in all instances are randomly generated within the square $[-100, 100] \times [-100, 100]$ (distance unit: m), the number of regions (n) is randomly chosen from the integer set $\{10, 11, \cdots, 20\}$, and the number of points of interest (m) is randomly chosen from the integer set $\{3, 4, \cdots, 10\}$. Besides, the vehicle's minimum turning radius (r) and the radius of regions (R) are randomly generated from given intervals: $r \in [1, 10]$ and $R \in [5, 10]$. A total of fifteen DTSPN instances are randomly generated to constitute a test suite. Using these random instances, we also add a comparison between DE–HRPE and another DE variant recently proposed by Islam et al. (namely, MDE-pBX)[38]. MDE-pBX is one of the well-known state-of-the-art DE variants, showing outstanding performance against many other advanced optimizers. The purpose of this comparison is to identify the possibility of further improving the performance of the DTSPN algorithm by employing advanced optimizers.



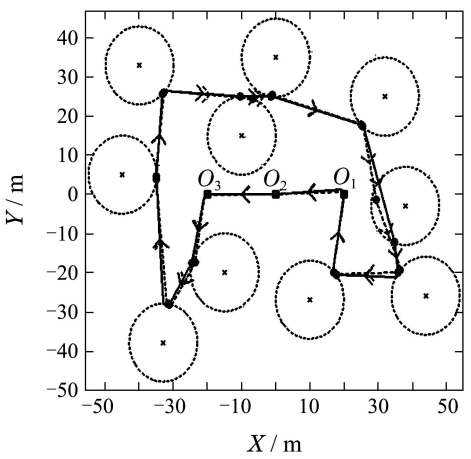
(a) Instance A1: overlapped, clustered ($r = 1$)



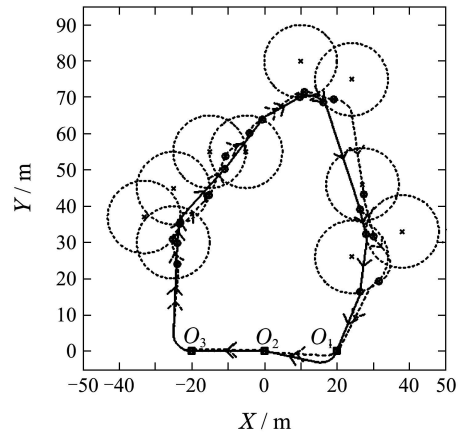
(b) Instance A2: overlapped, stringed ($r = 1$)



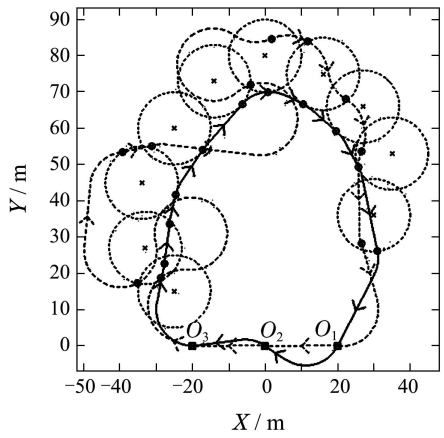
(c) Instance A3: non-overlapped, clustered ($r = 1$)



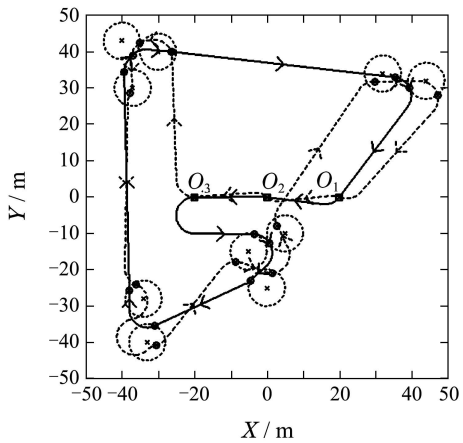
(d) Instance A4: evenly distributed ($r = 1$)



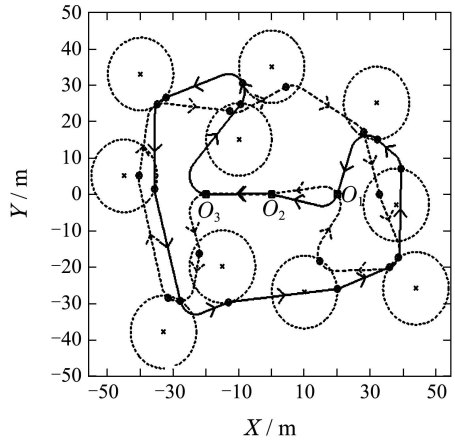
(e) Instance A5: overlapped, clustered ($r = 5$)



(f) Instance A6: overlapped, stringed ($r = 10$)



(g) Instance A7: non-overlapped, clustered ($r = 5$)



(h) Instance A8: evenly distributed ($r = 5$)

Fig. 2 Comparison of Dubins tours planned by two DE algorithms ($\mathrm{NFE} = 50,000$)[1]

In the two experiments, the three parameters for DE–HRPE and DE–CE were set as $F = 0.5$, $CR = 0.9$ and $NP = 10n$ according to the suggestion of Price et al.[35]. The GA for comparison maintains the same population size as DE in each case, and the other parameters for the algorithm follow the settings in the work of Obermeyer et al.[9]. The parameter setting for MDE-pBX follows the same rule proposed by its inventors (see [38]) except for $NP = 10n$.

DE–HRPE and MDE-pBX will be terminated when 50,000 function evaluations (i.e., calculations of 50,000 Dubins tours) have been implemented. In the first experiment, DE–CE and GA were allowed to carry out 500,000 function evaluations since they search in a larger space than DE–HRPE. The purpose of the long-term run of the two algorithms is to exhibit their limit behavior and obverse their performance difference. However, as the long-term run is very time-consuming, the maximal number of function evaluations for DE–CE and GA is also restricted to 50,000 in the second experiment, which also benefits a fair comparison with DE–HRPE. As for the sampling-based transformation method, 30 entry points ($n_{\mathrm{s}} = 30$) will be sampled along the boundary of each region, resulting in a $30n$ dimensional ATSP for each instance. All algorithms were implemented on a PC with Intel(R) Core (TM) 2 Duo CPU E8400 3.0 GHz, 2 GB RAM. Except for the TSP solver LKH (available from http://www.akira.ruc.dk/∼keld/research/LKH/), all programs including the derivation of the GTSP and ATSP formulations in the sampling-based transformation were compiled and executed in MATLAB R2009a. Regarding any DTSPN instance, each algorithm ran 30 times, and related results were statistically analyzed.

4.1 Experiment 1– typical DTSPN instances

The eight typical DTSPN instances in this experiment are illustrated in Fig.2. The four instances shown in Fig.2(a)−(d) involve a vehicle with a smaller minimal turning radius. In this case, the tours planned by DE–CE and DE–HRPE with the number of function evaluations $\mathrm{NFE} = 50,000$ are very similar. As the turning radius of the vehicle is increased, DE–CE and DE–HRPE show clear

[1]Dashed circles represent regions to be visited. O_1, O_2 and O_3 are three points of interest for the vehicle to visit in sequence. O_3 is also the starting position of the vehicle, and the initial heading angle is π (the vehicle points to the left at O_3). The arrows on planned tours indicate the moving direction of the vehicle. Dashed line with arrows: DE–CE; Solid line with arrows: DE–HRPE.

differences in the quality of planned tours [see Fig.2(e) vs. Fig.2(a); Fig.2(f) vs. Fig.2(b); Fig.2(g) vs. Fig.2(c); Fig.2(h) vs. Fig.2(d)]. As shown in Fig.2(e)−(f), both DE algorithms find the same visiting order of regions in instance A5 and instance A6; however, DE–CE has difficulty in fine-tuning the vehicle headings to reduce the tour length. As for instance A7 and instance A8, DE–CE and DE–HRPE even produced different visiting order of regions [see Fig.2(g)–(h)]. On the whole, with the same setting $\mathrm{NFE} = 50,000$, DE–HRPE performs better than DE–CE in terms of the quality of planned tours as well as the time cost (see also the statistical results in Table 1). Therefore, the turning radius (r) has a greater effect on DE–CE. In contrast, DE–HRPE is insensitive to the change of the turning radius, showing a stable and excellent performance in planning desirable Dubins tours for the vehicle.

Table 1 Statistical results about different algorithms in solving typical DTSPN instances[2]

Instance No.		Algorithms		
		DE–HRPE($\mathrm{NFE} = 50,000$) (avg,std,min,max)	DE–CE($\mathrm{NFE} = 50,000$) (avg,std,min,max)	DE–CE($\mathrm{NFE} = 500,000$) (avg,std,min,max)
A1	L	(202.1, **0.0**, 202.1, 202.2)	(203.1, 1.6, 201.6, 208.0)	**(201.6, 0.0, 201.6, 201.7)**
	T	(34.4, 0.3, 34.3, 36.0)	(265.2, 1.7, 262.7, 270.4)	(2850.3, 18.3, 2823.3, 2901.5)
A2	L	(205.1, **0.0**, 205.1, 205.1)	(206.7, 1.1, 204.8, 208.6)	**(204.6, 0.0, 204.6, 204.6)**
	T	(34.4, **0.1,** 34.3, 34.5)	(264.1, 0.6, 263.0, 266.1)	(2806.3, 2.3, 2802.4, 2813.6)
A3	L	(297.2, 0.4, 296.6, 297.5)	(296.1, 0.6, 295.4, 297.4)	**(295.1, 0.3, 294.4, 295.4)**
	T	(34.4, **0.1**, 34.2, 34.5)	(265.0, 1.3, 263.7, 269.4)	(2859.9, 30.7, 2835.3, 2923.9)
A4	L	(278.2, 9.3, 267.6, 296.2)	(273.4, 6.3, 265.4, 292.5)	(**273.3**, 8.1, **264.2, 289.4**)
	T	(34.4, **0.1**, 34.3, 34.6)	(263.3, 0.6, 262.7, 265.0)	(2799.7, 12.8, 2792.2, 2862.0)
A5	L	(209.5, 11.5, 207.3, 270.2)	(227.0, 10.8, 207.3, 244.9)	**(204.7, 0.0, 204.7, 204.7)**
	T	(34.7, **0.1**, 34.5, **35.0**)	(302.6, 3.7, 296.0, 309.3)	(3512.0, 20.7, 3465.5, 3551.9)
A6	L	(222.6, 24.8, 217.5, 354.1)	(421.0, 31.9, 329.1, 455.6)	**(212.0, 0.0, 212.0, 212.0)**
	T	(**35.5**, 0.2, **34.9, 35.8**)	(312.6, 1.9, 307.8, 316.4)	(3723.1, 34.6, 3639.4, 3787.4)
A7	L	(311.4, 5.2, 303.3, 319.8)	(338.2, 13.9, 310.7, 363.1)	**(307.4, 4.4, 301.8, 311.0)**
	T	(**31.8**, 0.3, **31.5, 32.8**)	(311.3, 4.6, 303.8, 321.1)	(3348.0, 29.9, 3297.1, 3390.9)
A8	L	(303.9, 12.4, 284.3, 319.9)	(312.5, 19.7, 283.6, 357.5)	**(281.2, 8.5, 273.0, 303.1)**
	T	(**32.5**, 1.6, **31.2**, 34.8)	(299.5, 4.2, 288.7, 305.8)	(3222.2, 29.0, 3155.7, 3280.0)

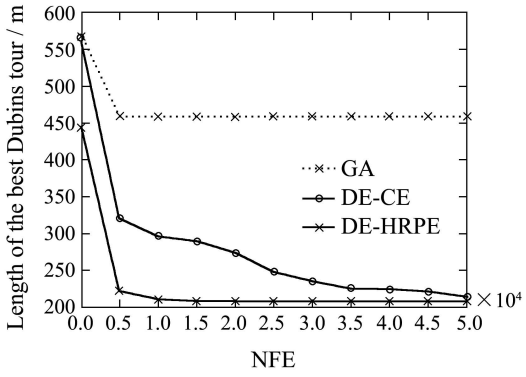
Instance No.		Algorithms	
		GA($\mathrm{NFE} = 500,000$) (avg,std,min,max)	Transformation method (avg,std,min,max)
A1	L	(291.6, 16.8, 256.0, 315.5)	(226.2, 4.8, 216.0, 234.7)
	T	(2843.4, 4.2, 2836.7, 2853.5)	**(32.7, 0.1, 32.6, 32.8)**
A2	L	(304.5, 12.6, 267.3, 328.7)	(231.7, 4.9, 221.2, 242.1)
	T	(2838.6, 1.7, 2836.1, 2843.3)	**(32.8, 0.1, 32.7, 33.1)**
A3	L	(320.2, 13.9, 278.7, 345.1)	(311.0, 5.2, 301.1, 326.4)
	T	(2840.0, 1.1, 2838.1, 2842.0)	(**33.0**, 0.3, **32.8, 34.0**)
A4	L	(350.2, 13.3, 315.8, 379.5)	(284.0, **5.2**, 271.9, 295.8)
	T	(2838.1, 6.3, 2834.1, 2856.9)	**(32.9, 0.1, 32.7, 33.0)**
A5	L	(412.4, 16.0, 373.4, 440.1)	(388.0, 24.4, 343.5, 436.9)
	T	(3004.3, 4.0, 2996.8, 3015.5)	(**34.1**, 0.2, **33.9**, 35.1)
A6	L	(619.0, 26.4, 560.0, 661.6)	(795.6, 49.8, 673.9, 907.2)
	T	(3152.0, 4.8, 3143.8, 3160.7)	(36.0, **0.0**, 36.0, 36.1)
A7	L	(463.3, 13.5, 432.6, 483.0)	(414.7, 33.8, 360.8, 490.9)
	T	(3008.3, 8.0, 2996.3, 3035.3)	(34.3, **0.1**, 34.1, 34.8)
A8	L	(468.8, 12.7, 438.0, 489.3)	(370.6, 19.9, 333.7, 410.3)
	T	(2948.2, 3.8, 2940.6, 2955.2)	(33.4, **0.2**, 33.1, **34.1**)

As shown in Table 1, the long-term run of DE–CE leads to the best Dubins tours in all test cases, providing a reference for measuring the quality of planned Dubins tours. However, the long-term run of DE–CE is very time-consuming (e.g. on average 2850.3 s in instance A1) and even unacceptable, especially in real-time path planning. Although DE–HRPE with $\mathrm{NFE} = 50,000$ produces slightly longer Dubins tours than DE–CE with $\mathrm{NFE} = 500,000$, it greatly reduces the time cost as compared to DE–CE. From Table 1, it can be observed that DE–HRPE in comparison with DE–CE adopting the same NFE, saves approximately 7/8 of the computation time, and meanwhile, the Dubins tours obtained by DE–HRPE in length is very close to or even much better than those by

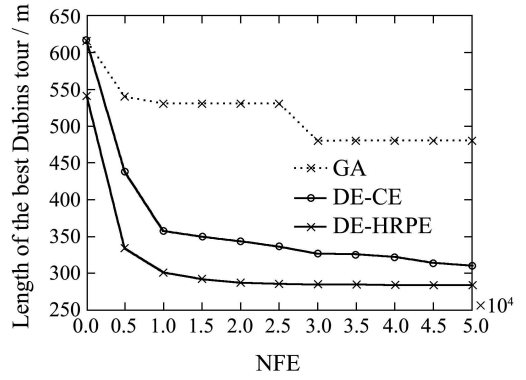
[2] L: Length of planned Dubins tour; T: time cost (unit: s); avg, std, min, max: average, standard deviation, minimum, and maximum of the results in 30 runs.

DE–CE.

Detailed statistical results about the two DE algorithms, GA and the sampling-based transformation method are presented in Table 1. The Dubins tours planned by GA with $\mathrm{NFE} = 500,000$ are obviously inferior to those by DE–CE ($\mathrm{NFE} = 500,000$) and DE–HRPE ($\mathrm{NFE} = 50,000$). In contrast to DE–CE and GA, the sampling-based transformation method has significant advantage in term of computation time, but it is inferior to DE–HRPE as the Dubins tours that it obtains in each case are too long. It was also observed that the time cost of the transformation method mainly stems from the computation of Dubins distances which are used as GTSP parameters and further transformed into ATSP parameters for LKH. By comparison, DE–HRPE on one hand saves a lot of time like the transformation method, and on the other hand can produce high-quality Dubins tours whose lengths are close to those obtained by DE–CE with a long-term run. The convergence process of DE–HRPE, DE–CE and GA with $\mathrm{NFE} = 50,000$ in instance A5 and instance A8, as examples, is illustrated in Fig.3. As compared with DE–CE and GA, DE–HRPE has a good start because the initial tour that DE–HRPE produces is a piecewise optimal one which is obviously shorter than the randomly generated ones in a larger search space for DE–CE and GA. Besides, DE–HRPE converges to high-quality solutions very fast, performing visibly better than DE–CE and GA.



(a) Instance A5



(b) Instance A8

Fig. 3 Convergence plots of GA, DE–CE and DE–HRPE ($\mathrm{NFE} = 50,000$)[3]

[3]NFE: the accumulated number of function evaluations

4.2 Experiment 2– a general test

This experiment is aimed at providing a general comparison of five DTSPN algorithms including MDE-pBX, since all instances are randomly generated instead of being elaborately designed as in the first experiment. The statistical results are presented in Table 2. In each case, the best result w.r.t. each index is highlighted in bold. On the whole, among the four algorithms involved in the first experiment, DE–HRPE is still the best as it leads to the best Dubins tours in most cases with a lower time cost. The Dubins tours planned by DE–HRPE are, on average, 14.54%, 32.01%, 16.11%, shorter than those by DE–CE, GA and the sampling-based transformation method, respectively. In term of computation time, DE–HRPE in contrast to DE–CE, GA and the transformation method yields, on average, about 83.83%, 83.80%, and 3.27% reductions, respectively. We also employed the statistics tool, Wilcoxon rank sum test, to analyze the experimental results. The hypothesis tests with a significance level 0.05 indicate that in term of solution quality, DE–HRPE significantly outperforms DE–CE in all instances except instances B3 and B15. Even in instances B3 and B15, the tours planned by DE–HRPE are just slightly longer than those by DE–CE, while DE–HRPE reduces the time cost greatly. The hypothesis tests also show that DE–HRPE outperforms GA in all cases regarding both solution quality and time cost, and with comparative time cost, the tours planned by DE–HRPE in all cases are significantly better than those by the transformation method.

As compared with DE–HRPE, MDE-pBX shows its advantages in finding high-quality Dubins tours. MDE-pBX outperforms DE–HRPE in term of the length of Dubins tours in almost all cases except for instances B2, B6 and B7 whose scales are a bit larger than the other ones. However, in all cases, MDE-pBX spent much longer time in obtaining better tours, which is due to its relatively complex operations (see [38] for technical details). On average, MDE-pBX reduces the tour length by 0.3% as compared with DE–HRPE while the time cost of MDE-pBX is about 30% larger than that of DE–HRPE. What's more, MDE-pBX is defeated by DE–HRPE in cases of B2, B6 and B7. It should be noted that the only difference between DE–HRPE and MDE-pBX lies in their operators and mechanisms for parameter tuning. Both of them adopt the same technique of heading relaxation and partial encoding. Therefore, the superiority of both DE–HRPE and MDE-pBX validates the advantages of direct search based on heading relaxation and partial encoding. Since a balance between solution quality and time cost is usually requisite from a pragmatic view, the algorithm DE–HRPE is still competent for solving DTSPN even as compared with MDE-pBX.

Table 2 Statistical results about different algorithms in solving random DTSPN instances

Ins No.	(n,m)	Perf. Index	DE–HRPE (avg,std,min,max)	MDE-pBX (avg,std,min,max)	DE–CE (avg,std,min,max)	GA (avg,std,min,max)	Transformation method (avg,std,min,max)
B1	(13,6)	L	(699.8, 31.7, 665.7, 804.9)	**(695.2, 20.2, 662.8, 749.2)**	(933.1, 26.4, 873.6, 986.7)	(1080.1, 20.4, 1048.0, 1130.3)	(907.0, 164.8, 704.8, 1221.5)
		T	(63.2, 0.1, 62.9, 63.4)	(82.2, 0.2, 81.5, 83.7)	(394.2, 0.8, 392.9, 395.7)	(397.3, 0.4, 396.7, 398.1)	**(57.2, 0.1, 57.1, 57.3)**
B2	(17,10)	L	**(837.9, 17.7, 793.8, 876.0)**	(843.5, 18.3, 804.2, 893.0)	(893.0, 18.0, 854.9, 936.8)	(1081.1, 20.9, 1026.1, 1122.9)	(893.5, 23.8, 835.5, 915.4)
		T	(89.2, 1.3, 87.0, 90.4)	(124.1, 2.0, 121.2, 128.5)	(530.7, 1.0, 529.1, 533.4)	(535.7, 0.5, 534.5, 536.4)	**(83.2, 0.2, 83.0, 83.6)**
B3	(10,7)	L	(566.4, 3.4, 563.2, 582.6)	**(564.2, 3.1, 561.4, 578.4)**	(557.3, 9.0, 543.7, 577.3)	(665.5, 14.3, 641.6, 691.0)	(600.4, 11.5, 585.2, 623.4)
		T	(57.0, 0.1, 56.8, 57.2)	(74.1, 0.3, 73.2, 76.1)	(339.2, 2.0, 334.1, 344.2)	(340.2, 1.0, 336.4, 341.0)	**(55.2, 0.1, 55.2, 55.4)**
B4	(13,8)	L	(610.8, 13.7, **592.2**, 659.3)	**(608.8, 12.2, 592.2, 624.1)**	(705.8, 17.6, 672.6, 739.2)	(903.8, 19.0, 862.8, 936.8)	(679.6, 28.8, 646.6, 737.0)
		T	**(70.5, 0.2, 70.2, 70.9)**	(91.7, 0.5, 89.4, 94.0)	(433.6, 1.3, 431.4, 436.3)	(431.5, 0.5, 430.6, 432.6)	(97.8, 2.2, 96.8, 102.2)
B5	(10,6)	L	(565.5, 14.0, **532.6**, 590.9)	**(564.1, 6.3, 532.6, 580.2)**	(735.5, 19.6, 688.5, 768.1)	(832.2, 20.6, 773.6, 860.6)	(822.7, 76.3, 698.8, 929.8)
		T	**(54.9, 0.3, 54.4, 55.5)**	(71.4, 0.5, 69.1, 73.6)	(347.1, 0.9, 345.4, 349.2)	(347.6, 0.4, 346.9, 348.4)	(59.6, 1.0, 59.2, 62.2)
B6	(13,10)	L	**(1031.6, 22.0, 988.1, 1069.3)**	(1036.8, 22.3, 992.7, 1075.5)	(1256.9, 29.9, 1201.5, 1314.7)	(1467.2, 23.0, 1387.7, 1502.5)	(1215.7, 135.9, 1074.1, 1476.8)
		T	**(78.0, 0.2, 77.6, 78.4)**	(101.4, 0.6, 98.8, 103.5)	(497.0, 1.4, 493.5, 500.2)	(494.2, 0.6, 492.9, 495.0)	(113.6, 0.1, 113.6, 113.8)
B7	(18,8)	L	**(933.9, 30.8, 867.6, 1005.4)**	(968.3, 32.3, 872.2, 1024.1)	(1208.7, 27.1, 1155.1, 1276.5)	(1399.5, 26.5, 1337.1, 1443.9)	(1135.2, 151.1, 885.7, 1309.8)
		T	**(88.7, 1.0, 88.1, 93.6)**	(115.3, 2.2, 107.1, 122.9)	(559.4, 2.4, 555.6, 569.0)	(560.2, 2.5, 557.0, 568.5)	(110.5, 0.1, 110.4, 110.6)
B8	(10,4)	L	(451.0, 3.7, 444.5, 457.9)	(**446.3, 1.3,** 444.5, **450.6**)	(459.1, 10.3, **444.0**, 487.0)	(574.9, 12.8, 531.3, 596.6)	(533.9, 39.1, 469.5, 578.1)
		T	**(47.7, 0.1, 47.5, 47.9)**	(62.4, 0.2, 61.6, 63.3)	(289.9, 0.8, 288.2, 291.4)	(290.6, 0.2, 290.1, 291.0)	(62.0, 0.1, 61.8, 62.0)
B9	(15,3)	L	(585.2, 36.2, **535.1**, 667.7)	**(576.3, 12.2, 535.1, 614.8)**	(873.0, 32.2, 808.4, 932.4)	(1098.0, 29.7, 1031.9, 1144.3)	(765.6, 54.0, 653.5, 850.3)
		T	**(62.2, 0.2, 61.7, 62.8)**	(80.9, 0.5, 79.7, 82.1)	(382.2, 1.5, 378.9, 384.3)	(381.5, 0.4, 380.7, 382.2)	(73.1, 0.1, 73.1, 73.2)
B10	(16,6)	L	(637.6, 24.1, **595.8**, 695.3)	**(632.9, 11.5, 595.8, 649.1)**	(754.6, 19.5, 713.4, 798.7)	(1011.8, 18.7, 979.6, 1059.1)	(706.0, 88.2, 604.3, 826.3)
		T	**(75.5, 0.2, 75.0, 75.8)**	(98.4, 0.9, 96.1, 100.2)	(455.6, 1.8, 451.9, 460.1)	(454.9, 0.5, 453.9, 455.9)	(82.7, 0.0, 82.6, 82.7)
B11	(11,3)	L	(362.4, 9.8, **355.1**, 408.2)	**(358.2, 3.7, 355.1, 367.7)**	(433.9, 14.7, 407.0, 475.8)	(658.5, 14.2, 621.3, 678.7)	(566.7, 56.2, 491.1, 687.6)
		T	(48.6, 0.1, 48.4, 48.8)	(63.5, 0.3, 62.0, 64.3)	(310.6, 1.6, 306.0, 313.6)	(300.2, 0.4, 299.2, 300.9)	**(39.2, 0.0, 39.2, 39.2)**
B12	(13,5)	L	(698.4, 15.3, **670.2**, 721.8)	**(691.2, 11.5, 670.2, 705.2)**	(874.9, 30.9, 829.3, 937.3)	(1099.1, 13.0, 1059.2, 1122.3)	(862.7, 47.5, 777.7, 941.2)
		T	(61.8, 0.5, 60.9, 62.6)	(80.4, 1.0, 77.7, 82.3)	(388.0, 1.7, 385.4, 391.2)	(383.9, 0.4, 382.9, 384.7)	**(55.1, 0.0, 55.0, 55.2)**
B13	(14,8)	L	(805.9, 23.7, **784.9**, 869.5)	**(793.1, 12.9, 784.9, 838.4)**	(933.6, 31.3, 885.3, 982.9)	(1196.6, 30.2, 1144.0, 1243.4)	(932.6, 26.0, 895.6, 972.6)
		T	(74.3, 0.2, 73.8, 74.6)	(97.1, 0.7, 95.2, 99.6)	(474.9, 2.0, 471.3, 479.2)	(470.9, 0.5, 469.9, 472.1)	**(65.0, 0.0, 64.9, 65.0)**
B14	(17,3)	L	(596.7, 34.0, **553.5**, 687.9)	**(590.3, 22.6, 553.5, 644.3)**	(676.3, 18.2, 643.3, 728.0)	(1009.8, 27.2, 914.3, 1074.2)	(683.2, 52.9, 588.5, 732.2)
		T	**(68.1, 0.2, 67.8, 68.5)**	(89.1, 0.8, 87.2, 92.4)	(421.1, 1.9, 417.6, 424.9)	(420.0, 0.2, 419.5, 420.5)	(92.8, 0.0, 92.7, 92.9)
B15	(12,7)	L	(530.2, 7.9, 518.8, 544.2)	(529.8, 7.8, 518.8, 543.7)	**(524.5, 7.5, 516.2, 543.2)**	(662.1, 18.5, 621.7, 684.3)	(544.2, 7.1, 533.1, 551.6)
		T	(64.1, 0.2, 63.6, 64.4)	(83.7, 0.8, 81.2, 85.3)	(385.0, 0.8, 383.6, 386.5)	(391.5, 0.3, 390.9, 391.9)	**(45.9, 0.5, 45.3, 46.2)**

4.3 Discussion

Global search with complete encoding like DE–CE is beneficial to a full exploration in the whole problem space of DTSPN. However, it is usually very time-consuming to achieve a global search for the sake of guaranteeing the optimality of the Dubins tour finally obtained. The partial encoding scheme based on terminal heading relaxation in DE–HRPE greatly reduces the search space without obvious deterioration of solution quality. On the whole, DE–HRPE achieves a desirable trade-off between solution quality and time cost. It should be noted that the terminal heading relaxation sacrifices the optimality of solutions to some extent since the restriction on terminal headings according to theorems 1-6 may miss real optimal Dubins tours. So, it is not surprising that the Dubins tours produced by DE–HRPE ($\mathrm{NFE} = 50,000$) in each case are slightly longer than those by DE–CE ($\mathrm{NFE} = 500,000$). However, any Dubins tour derived from the theoretical results in DE–HRPE is at least a piecewise optimal one, which is a main contribution to the excellent performance of DE–HRPE. In contrast, the sampling-based transformation method depends heavily on the number of sampling points. Generally, with more sampling points, the solution quality of this method may be further improved; however, its time cost will also be increased quickly.

5 Conclusion

In order to solve the challenging Dubins traveling salesman problem with neighborhood efficiently, we proposed two differential evolution algorithms which are based on a hybrid encoding scheme (a combination of direct encoding and random-key based indirect encoding). The first DE algorithm is expected to search for the optimal or near-optimal Dubins tour in the complete search space which involves the optimization of both continuous variables (positions of waypoints and vehicle headings) and discrete variables (visiting order of regions). This DE algorithm with sufficient iterations can produce high-quality solutions; however, relatively speaking, its time cost is undesirable. In order to achieve a better trade-off between solution quality and time cost, we proposed the second DE algorithm which adopts a terminal heading relaxation scheme to generate piecewise optimal Dubins tours without tuning vehicle headings. The second DE algorithm can find high-quality Dubins tours with obviously reduced computation time. Experimental results demonstrate that the DE algorithm based on terminal heading relaxation and partial encoding, namely DE–HRPE, has prominent advantages over the DE algorithm with complete encoding, a genetic algorithm, and a sampling-based transformation method. As an excellent DTSPN solver, DE–HRPE can gain ground in many applications, e.g. the path-planning of unmanned aerial vehicles.

References:

[1] SAVLA K D. Multi UAV systems with motion and communication constraints [D]. Santa Barbara, USA: University of California, 2007.

[2] LE NY J. Performance optimization for unmanned vehicle systems [D]. Cambridge, MA, USA: Massachusetts Institute of Technology, 2008.

[3] CHOW B. Assigning closely spaced targets to multiple autonomous underwater vehicles [D]. Waterloo, Ontario, Canada: University of Waterloo, 2009.

[4] RATHINAM S, SENGUPTA R, DARBHA S. A resource allocation algorithm for multivehicle systems with nonholonomic constraints [J]. *IEEE Transactions on Automation Science and Engineering*, 2007, 4(1): 98 – 104.

[5] RATHINAM S, SENGUPTA R. Algorithms for routing problems involving UAVs [M] //CHAHL J S, JAIN L C, MIZUTANI A, et al. *Studies in Computational Intelligence*. Berlin: Springer, 2007: 147 – 172.

[6] DUBINS L E. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents [J]. *American Journal of Mathematics*, 1957, 79(3): 497 – 516.

[7] SHIMA T, RASMUSSEN S, CROSS D. Assigning micro UAVs to task tours in an urban terrain [J]. *IEEE Transactions on Control System Technology*, 2007, 15(4): 601 – 612.

[8] SANFELICE R G, YONG S Z, FRAZZOLI E. On minimum-time paths of bounded curvature with position-dependent constraints [J]. *Automatica*, 2014, 50(2): 537 – 546.

[9] OBERMEYER K J, OBERLIN P, DARBHA S. Path planning for a UAV performing reconnaissance of static ground targets in terrain [C] //*Proceedings of the AIAA Guidance, Navigation, and Control Conference*. Reston: AIAA, 2009: 1 – 10.

[10] ISAACS J T, KLEIN D J, HESPANHA J P. Algorithms for the traveling salesman problem with neighborhoods involving a Durbins vehicle [C] //*Proceedings of the American Control Conference*. New York: IEEE, 2011: 1704 – 1709.

[11] SAVLA K, FRAZZOLI E, BULLO F. Traveling salesperson problems for the Dubins vehicle [J]. *IEEE Transactions on Automatic Control*, 2008, 53(6): 1378 – 1391.

[12] LE NY J, FERON E, FRAZZOLI E. On the Dubins traveling salesman problem [J]. *IEEE Transactions on Automatic Control*, 2012, 57(1): 265 – 270.

[13] HANSON C, RICHARDSON J, GIRARD A. Path planning of a Dubins vehicle for sequential target observation with ranged sensors [C] //*Proceedings of the American Control Conference*. New York: IEEE, 2011: 1698 – 1703.

[14] YADLAPALLI S, MALIK W A, DARBHA S, et al. A Lagrangian-based algorithm for a multiple depot, multiple traveling salesman problem [J]. *Nonlinear Analysis: Real World Applications*, 2009, 10(4): 1990 – 1999.

[15] DUMITRESCU A, MITCHELL J S B. Approximation algorithms for TSP with neighborhoods in the plane [J]. *Journal of Algorithms*, 2003, 48(1): 135 – 159.

[16] YUAN B, ORLOWSKA M, SADIQ S. On the optimal robot routing problem in wireless sensor networks [J]. *IEEE Transactions on Knowledge and Data Engineering*, 2007, 19(9): 1252 – 1261.

[17] GOAOC X, KIM H S, LAZARD S. Bounded-curvature shortest paths through a sequence of points using convex optimization [J]. *SIAM Journal on Computing*, 2013, 42(2): 662 – 684.

[18] LE NY J, FERON E. An approximation algorithm for the curvature-constrained traveling salesman problem [C] // *Proceedings of the 43rd Annual Allerton Conference on Communications, Control and Computing*. New York: Curran Associates, Inc., 2005: 1 – 10.

[19] OBERLIN P, RATHINAM S, DARBHA S. Today's traveling salesman problem: heterogeneous, multiple depot, multiple UAV routing problem [J]. *IEEE Robotics & Automation Magazine*, 2010, 17(4): 70 – 77.

[20] AL-MULHEM M, AL-MAGHRABI T. Efficient convex-elastic net algorithm to solve the Euclidean traveling salesman problem [J]. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 1998, 28(4): 618 – 620.

[21] SAVLA K, FRAZZOLI E, BULLO F. On the point-to-point and traveling salesperson problems for Dubins' vehicle [C] //*Proceedings of the American Control Conference*. New York: IEEE, 2005: 786 – 791.

[22] OBERLIN P, RATHINAM S, DARBHA S. Combinatorial motion planning for a Dubins vehicle with precedence constraints [C] //*Proceedings of the ASME Dynamic Systems and Control Conference*. New York: ASME, 2009: 1627 – 1634.

[23] MACHARET D G, NETO A A, NETO V F D, et al. An evolutionary approach for the Dubins' traveling salesman problem with neighborhood [C] //*Proceedings of the 14th International Conference on Genetic and Evolutionary Computation*. New York: ACM, 2012: 377 – 384.

[24] YAZICI A. A smooth tour construction approach for a mobile robot with kinematic constraints [J]. *International Journal of Advanced Robotic Systems*, 2013, 10: DOI:10.5772/56942.

[25] HELSGAUN K. An effective implementation of the Lin-Kernighan traveling salesman heuristic [J]. *European Journal of Operational Research*, 2000, 126(1): 106 – 130.

[26] LE NY J, FERON E, FRAZZOLI E. The curvature-constrained traveling salesman problem for high point density [C] //*Proceedings of the 46th IEEE Conference on Decision and Control*. New York: IEEE, 2007: 5985 – 5990.

[27] SNYDER L V, DASKIN M S. A random-key genetic algorithm for the generalized traveling salesman problem [J]. *European Journal of Operational Research*, 2006, 174(1): 38 – 53.

[28] OBERLIN P, RATHINAM S, DARBHA S. A transformation for a heterogeneous, multiple depot, multiple traveling salesman problem [C] //*Proceedings of the 2009 American Control Conference*. New York: IEEE, 2009: 1292 – 1297.

[29] REGO C, GAMBOA D, GLOVER F, et al. Traveling salesman problem heuristics: leading methods, implementations and latest advances [J]. *European Journal of Operational Research*, 2011, 211(3): 427 – 441.

[30] XIN B, CHEN J, ZHANG J, et al. Hybridizing differential evolution and particle swarm to design powerful optimizers: a review and taxonomy [J]. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 2012, 42(5): 744 – 767.

[31] OBERMEYER K J, OBERLIN P, DARBHA S. Sampling-based roadmap methods for a visual reconnaissance UAV [C] //*Proceedings of the AIAA Guidance, Navigation and Control Conference*. Reston: AIAA, 2010: 1 – 21.

[32] OBERMEYER K J, OBERLIN P, DARBHA S. Sampling-based path planning for a visual reconnaissance unmanned air vehicle [J]. *Journal of Guidance, Control, and Dynamics*, 2012, 35(2): 619 – 631.

[33] ZHANG Y, CHEN J, SHEN L C. Hybrid hierarchical trajectory planning for a fixed-wing UCAV performing air-to-surface multi-target attack [J]. *Journal of Systems Engineering and Electronics*, 2012, 23(4): 536 – 552.

[34] DAS S, SUGANTHAN P N. Differential evolution: a survey of the state-of-the-art [J]. *IEEE Transactions on Evolutionary Computation*, 2011, 15(1): 4 – 31.

[35] PRICE K V, STORN R M, LAMPINEN J A. *Differential Evolution-a Practical Approach to Global Optimization* [M]. Berlin: Springer, 2005.

[36] BEAN J C. Genetic algorithms and random keys for sequencing and optimization [J]. *ORSA Journal on Computing*, 1994, 6(2): 154 – 160.

[37] BOISSONNAT J D, BUI X N. *Accessibility region for a car that only moves forwards along optimal paths: research report* [R]. Sophia-Antipolis, France: INRIA, 1994.

[38] ISLAM S M, DAS S, GHOSH S, et al. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization [J]. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 2012, 42(2): 482 – 500.

作者简介:

辛 斌 (1982−), 男, 第18届(2012年)《关肇直奖》获奖论文作者, 讲师, 目前研究方向为智能优化与智能控制, E-mail: brucebin@bit.edu.cn;

陈 杰 (1965−), 男, 教授, 目前研究方向为复杂系统的多指标优化与控制, E-mail: chenjie@bit.edu.cn;

徐冬玲 (1962−), 女, 教授, 目前研究方向为多属性决策分析、决策论、最优化方法等, E-mail: ling.xu@mbs.ac.uk;

陈玉旺 (1982−), 男, 讲师, 目前研究方向为复杂系统的决策分析、建模与优化、计算智能, E-mail: yu-wang.chen@mbs.ac.uk.