2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Efficient Target Visiting Path Planning for Multiple Vehicles with Bounded Curvature

Douglas G. Macharet[1] Armando Alves Neto[2] Vilar F. da Camara Neto[3] Mario F. M. Campos[1]

***Abstract*— In this paper, we introduce the k-Dubins Traveling Salesman Problem with Neighborhoods (k-DTSPN), the problem of planning efficient paths among target regions for multiple robots with bounded curvature constraints (Dubins vehicles). This paper presents two approaches for the problem. Firstly, we present a heuristic that solves it in two steps, based on classical techniques found in the literature. Secondly, we employ a Memetic Algorithm to solve both combinatorial and continuous phases of the problem in a combined manner. We provide formal analysis about both proposed techniques, presenting upper bounds to the length of the longest tour. Numerous trials in simulated environments were executed, providing statistical examination of the final results.**

I. INTRODUCTION

Finding feasible paths for mobile agents that are either length or time optimized has been the goal of several research fields. In this context, the Traveling Salesman Problem (TSP) remains as one of the most studied problems, and several heuristics have been proposed to this NP-hard problem. However, for several real-world scenarios the mathematical formulation of the TSP may be either insufficient or too simplistic to be useful. To overcome some of these limitations, the TSP has been generalized in several ways:

- the k-Traveling Salesman Problem (k-TSP) employs k salesmen that start and end at a single city. The remaining cities must be visited once by one of the salesmen;
- in the Traveling Salesman Problem with Neighborhoods (TSPN), each city is represented by a region instead of a point. To visit a city, the salesman must reach any point inside its region;
- in the Dubins TSP (DTSP), the travel path must conform to a minimum curvature radius ρ at all points. This is called a nonholonomic constraint.

All of these variations were created to tackle some real-world problems such as precision agriculture, environmental monitoring and surveillance and exploration of regions. The k-TSP is the natural extension to multiple-agent setups. The TSPN is commonly employed in the study of Wireless Sensor Networks (WSNs), where a mobile agent travels through the network collecting data from the nodes. The visiting region of each node represents its communication range. Finally, the DTSP is convenient when the salesman is a nonholonomic vehicle that conforms to the Dubins constraint.

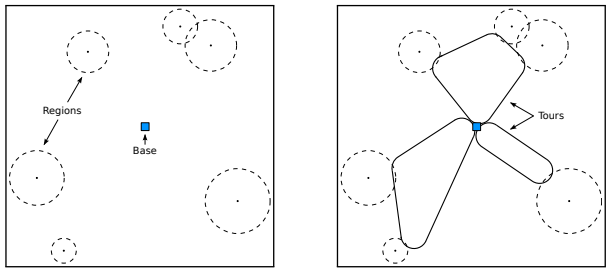


Fig. 1. Example of the k-Dubins Traveling Salesman Problem with Neighborhoods: (a) regions to be visited by a group of Dubins vehicles (dashed circles) and the base area (cyan square); (b) a possible set of tours, one for each robot, to reach each node's region and return to the base.

In this work we introduce the k-Dubins Traveling Salesman Problem with Neighborhoods (k-DTSPN), a three-fold generalization of the TSP suited to the scenario where a team of k Dubins-constrained vehicles are dispatched from a central base to visit some given regions of interest in the environment and then return to the base, accomplishing the mission as fast as possible. Figure 1 illustrates the problem.

This paper presents two approaches for the problem. Firstly, we present a heuristic that solves this problem in two steps, based on classical techniques found in the literature. Secondly, we propose a technique based on a Memetic Algorithm to solve both the combinatorial and the continuous steps of the problem in a combined manner. Though investigations concerning optimality issues are very complex in such cases, we provide formal analysis about both proposed techniques, presenting upper bounds to the length of the tours and showing statistical examination of the final results.

The remainder of this paper is structured as follows. A review of the literature is presented in Section II; in Section III, we provide the problem formalization, propose two algorithms to solve it and offer formal analysis for both; numerical results for different scenarios and statistical analysis are shown in Section IV; and in Section V, we draw the conclusions and discuss avenues for future investigation.

II. RELATED WORK

Several shortest path algorithms for vehicles with no constraints can be found in the literature. The Traveling Salesman Problem (TSP), for example, is a fundamental optimization problem and has been widely studied [1].

[1]D.G. Macharet and M.F.M. Campos are with the Computer Vision and Robotics Laboratory (VeRLab), Computer Science Department, Universidade Federal de Minas Gerais, MG, Brazil. E-mails: {doug,mario}@dcc.ufmg.br

[2]A. Alves Neto is with the Telecommunication and Mechatronic Department, Universidade Federal de São João del-Rei, MG, Brazil. E-mail: aaneto@ufsj.edu.br

[3]V.F. da Camara Neto is with the Fundação Centro de Análise, Pesquisa e Inovação Tecnológica (FUCAPI), Manaus, AM, Brazil. E-mail: vilar.neto@fucapi.br

The k-TSP aims to generate paths to k salesmen departing from a given city and returning to it, after each one of all cities have been visited by a single salesman. A more general instance of this problem is known as the Vehicle Routing Problem (VRP) [2], [3]. The k-TSP was also used to model Data Gathering in Wireless Sensor Networks, by applying holonomic multirobot teams [4], [5]. However, the kinematic constraints of the vehicles are not considered, resulting in paths that may not be feasible by real robots.

The challenge of generating a minimum length path through a set of waypoints for a single vehicle, subject to minimum curvature constraints and making use of Dubins curves, was initially introduced in [6]. This problem was named DTSP and was further studied in [7], [8], [9].

The generation of paths for multiple Dubins vehicles has been the focus of a great amount of research, specially for Unmanned Aerial Vehicles (UAVs) [10], [11]. However, in both works the problem is not modeled with a single start/end point (i.e. a base) for all vehicles, in contrast to the constraints imposed by the classical VRP.

The TSPN was initially introduced in [12]. In this problem, each city to be visited is represented as a geographic region, referred to as the neighborhood of the city. The salesman can meet the buyer at any point of its corresponding region. The TSPN can be stated as the problem of generating the shortest path that intersects every neighborhood at least once. The TSPN is often used to model problems related to the generation of tours to collect data from Wireless Sensor Networks [13], [14].

Recently, the constraints imposed by DTSP and TSPN began to be considered as part of the same challenge, which will be referred here as the Dubins Traveling Salesman Problem with Neighborhoods (DTSPN). In [15] a Genetic Algorithm is proposed to deal with this problem, while in [16], [17] a sampling based approach is used.

The DTSPN was also dealt with in our previous works. In [18] a simple three-step evolutionary algorithm was used, but the position and orientation of the waypoints were not considered in the optimization process. This restriction was lifted in [19], where a meta-heuristic was presented to optimize the position of the waypoints inside the regions and a heuristic was proposed to set the orientation of the waypoints.

The extension of the DTSPN to the multiple vehicle scenario (k-DTSPN) brings important benefits such as the reduction of the overall time to visit all the regions, but it also increases the challenge of evaluating efficient paths that consider all the problem constraints. To the best of our knowledge, this is the first work dealing with this problem.

Hence, in this paper we propose two approaches. The first one solves the combinatorial (sequence of visits) and continuous (position and orientation of waypoints) steps separately. The second approach jointly considers both steps. We show that the use of a Memetic Algorithm to solve both combinatorial and continuous optimization problems in a combined manner produces promising results.

III. METHODOLOGY

A. Theoretical formalization

Let us consider the scenario where a team of k robots must visit a set of regions of interest, such as nodes of a wireless network or targets of visual inspection (i.e., environmental surveillance). Let $\boldsymbol{\mathcal{H}} = \{\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_N\}$, where $\mathcal{H}_i \subset \mathbb{R}^2$, be a set of $N+1$ closed regions representing the neighborhood of each node i. The key to the underlying problem is to guarantee that every region $\mathcal{H}_i$ is visited by at least one robot. We call $\mathcal{H}_0$ the *base area*, where all robots are initially located and to which all must return at the end of the mission.

For simplicity reasons,[1] we consider that each region $\mathcal{H}_i$ is circular and centered at coordinates $\mathbf{n}_i = (x_i, y_i)$. Then, each region can be formally defined as:

$$\mathcal{H}_i = \{\mathbf{p} \in \mathbb{R}^2 : \|\mathbf{p} - \mathbf{n}_i\| \leq r_i\},$$

where r_i is the radius of the i-th node neighborhood.

A convenient and widely adopted way of dealing with the kinematic model of mobile robots in a two-dimensional space is to represent the instant configuration of the vehicle as a position $\mathbf{p}$ and an orientation vector ψ. We use the notation $\mathbf{q} = \langle \mathbf{p}, \psi \rangle$, where $\mathbf{q} \in \mathrm{SE}(2)$, to represent a *waypoint*, a configuration that a robot must eventually reach. A waypoint is always placed inside some region $\mathcal{H}_i$, i.e., $\mathbf{p} \in \bigcup_{i=1}^{N} \mathcal{H}_i$. When a robot assumes a configuration $\mathbf{q}$, we consider that the corresponding region $\mathcal{H}_i$ was *visited* by that robot. For future reference, the entire set of waypoints (excluding those at the base area) will be referenced as $\boldsymbol{\mathcal{Q}} = \{\mathbf{q}_1, \ldots, \mathbf{q}_P\}$.

In the case that some regions intersect, it is more convenient to define a single waypoint placed inside the common intersection area. In this article we select the desired intersections according to an heuristic proposed in [19], described as follows: Pick the set of intersections $(\mathcal{H}_i \cap \mathcal{H}_j \cap \ldots)$ that cover the largest amount of regions. If there is more than one set with the same number of regions, then we select the set whose common region has the largest area. Then place a waypoint at the centroid of the intersection and repeat the process with the remaining intersecting regions until all intersections have waypoints inside.

As for the nonholonomic motion constraints, in this work we adopt the classical Dubins vehicle model [20]. This model encompasses a large class of nonholonomic vehicles, such as Ackerman steering cars and fixed-wing airplanes flying at constant altitude. Under this model, the path is restricted to a maximum curvature κ. As far as the underlying physics of the system is concerned, the curvature is defined as a quantity directly proportional to the lateral acceleration of the robot in the plane, and κ is inversely proportional to the minimum curvature radius ρ the vehicle is able to perform.

Now, let $\boldsymbol{\mathcal{T}} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k\}$ be the set of tours $\mathcal{T}_j$ assigned to each robot j, where each tour by definition begins and ends at the waypoint lying on the base area. Formally:

$$\mathcal{T}_j = \langle \mathbf{q}_{\text{base}}^{(j)}, \mathbf{q}_1^{(j)}, \mathbf{q}_2^{(j)}, \ldots, \mathbf{q}_{\text{base}}^{(j)} \rangle,$$

[1] This is not a restriction of the method – just a convenience to keep reasoning, implementation, and results visualisation simple. The methodology proposed in this paper covers continuous, convex regions in general.

where the waypoints $\mathbf{q}_c^{(j)}$ are in one-to-one correspondence with the set $\boldsymbol{\mathcal{Q}}$, i.e.:

$$\mathbf{q}_c^{(j)} \in \boldsymbol{\mathcal{Q}} \qquad \text{subject to}$$
$$\mathbf{q}_c^{(j)} \neq \mathbf{q}_{c'}^{(j')} \qquad \forall c \neq c' \vee j \neq j'$$

and the base waypoints $\mathbf{q}_{\text{base}}^{(j)}$ are all inside the base region, $\mathcal{H}_0$. For simplicity reasons, we define each tour $\mathcal{T}_j$ as a sequence $\Sigma_j = \langle 0, a, b, \ldots, 0 \rangle$ of waypoint (configuration) indexes from the set $\boldsymbol{\mathcal{Q}}$.

The previous definitions do not necessarily set that all tours must contain more than the base area. In other words, a satisfactory solution to the problem of visit all regions $\boldsymbol{\mathcal{H}}$ may not use all the k robots. For example, if the goal is to spend the minimum of energy during the mission, using all robots may not be interesting. In the other hand, if one aims to reach all regions as fast as possible, using more agents may be better. As we will discuss later, our objective is to accomplish the task in minimum time.

Then, given a tour $\mathcal{T}_j$ for the j-th robot of the team, we must optimize the configurations $\mathbf{q}_c^{(j)} \in \mathcal{T}_j$ to minimize the total circuit length $\mathcal{L}_\rho^j$ composed of Dubins curves over such configurations, without breaking the geometric association between each waypoint and the corresponding region(s). In other words, we must find configuration sets $\boldsymbol{\mathcal{Q}}_{\Sigma_j}$ for $1 \leq j \leq k$ composed of $\boldsymbol{\mathcal{P}}_{\Sigma_j} = \langle \mathbf{p}_{\text{base}}^{(j)}, \mathbf{p}_1^{(j)}, \ldots, \mathbf{p}_{\text{base}}^{(j)} \rangle$ (a subset of positions) and $\Psi_{\Sigma_j} = \langle \psi_{\text{base}}^{(j)}, \psi_1^{(j)}, \ldots, \psi_{\text{base}}^{(j)} \rangle$ (a subset of corresponding orientations) that generate short Hamiltonian tours for each robot passing through the base area.

Problem 3.1: k-Dubins Traveling Salesman Problem with Neighborhoods: Let k be the number of robots available to execute the task of visit all $\boldsymbol{\mathcal{H}}$ regions. These robots are represented as Dubins vehicles. Let $\mathcal{F} : \text{SE}(2) \to \mathbb{R}^2$ be a projection function that transforms the robot's configuration $\mathbf{q}_c$ into the workspace. Also, let $\mathcal{G} : \text{SE}(2)^P \to \mathbb{R}_0^+$, where

$$\mathcal{G}(\boldsymbol{\mathcal{Q}}) = \max_j \left[\mathcal{L}_\rho^j(\boldsymbol{\mathcal{Q}}_{\Sigma_j}) \right], \; j = 1, \ldots, k, \tag{1}$$

and subject to

$$\mathcal{F}(\mathbf{q}_c) \in \mathcal{H}_i, i = 1, \ldots, P,$$

be the Dubins circuit length of the longest tour in $\boldsymbol{\mathcal{T}}$, calculated as

$$\mathcal{L}_\rho^j(\boldsymbol{\mathcal{Q}}_{\Sigma_j}) = \sum \mathcal{D}_\rho(\mathbf{q}_c^{(j)}, \mathbf{q}_{c+1}^{(j)}) \qquad \forall \mathbf{q}_c^{(j)} \in \mathcal{T}_j, \tag{2}$$

where $\mathcal{D}_\rho : \text{SE}(2) \times \text{SE}(2) \to \mathbb{R}_0^+$ is the length of the shortest Dubins path with minimum curvature radius ρ between two configurations of a tour.

Then, we must optimize the Dubins circuit length as:

$$\underset{\boldsymbol{\mathcal{T}}}{\text{minimize}} \; \mathcal{G}(\boldsymbol{\mathcal{Q}}). \tag{3}$$

By minimizing the longest tour applied to one robot and under the assumption that all robots navigate at the same constant speed, we expect to reduce the total time to visit all regions.

B. Algorithm 1

Our first algorithm solves the k-DTSPN in two steps based on classical techniques found in the literature. Initially, a TSP instance is solved considering Euclidean costs and then separated in k tours (k-TSP). The second step traces the Dubins curves between the positions in order to make the whole path feasible by vehicles with bounded curvature.

For solving the k-TSP we use a technique called k-SPLITOUR [21]. This technique is the first constant factor approximation algorithm for k-TSP, with a theoretical upper bound of $\epsilon + 1 - k$, where ϵ is the approximation ratio of the algorithm used for computing the TSP tour. The algorithm constructs k tours by splitting a TSP tour according to the following steps:

1) Find a 1-tour (TSP) $\mathcal{T}_0 = \langle \mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n, \mathbf{p}_0 \rangle$, where $\mathbf{p}_0$ is the initial vertex (base);
2) For each j, $1 \leq j < k$, find the last vertex $\mathbf{p}_{i(j)}$ such that the cost of the path from $\mathbf{p}_0$ to $\mathbf{p}_{i(j)}$ along $\mathcal{T}_0$ is no greater than $(j/k)(L - 2c_{\max}) + c_{\max}$, where L is the Euclidean length of the circuit found in Step 1 and
$$c_{\max} = \max_n \|\mathbf{p}_0 - \mathbf{p}_n\|; \tag{4}$$
3) Build the k tours as $\mathcal{T}_1 = \langle \mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{i(1)}, \mathbf{p}_0 \rangle$, $\mathcal{T}_2 = \langle \mathbf{p}_0, \mathbf{p}_{i(1)+1}, \ldots, \mathbf{p}_{i(2)}, \mathbf{p}_0 \rangle$, $\ldots$, $\mathcal{T}_k = \langle \mathbf{p}_0, \mathbf{p}_{i(k-1)}, \ldots, \mathbf{p}_n, \mathbf{p}_0 \rangle$.

The cost of any tour $\mathcal{T}_j$, for each $1 \leq j \leq k$, obtained by the k-SPLITOUR does not exceed

$$\frac{1}{k}(L - 2c_{\max}) + 2c_{\max}, \tag{5}$$

which is also the average cost of any tour [21].

The Dubins curves among the points in each tour are then estimated by using a technique called Alternating Algorithm (AA) [6], which is a very simple heuristic. In short, it builds the circuit in such a way that consecutive pairs of waypoints are linked together with straight line segments, and the remaining (non-straight line) links are built using Dubins curves based on the orientations previously assigned to the waypoints. Formally, the orientation of a waypoint i, expressed as ψ_i, is determined as follows:

$$\psi_i = \begin{cases} \text{dir}(\mathbf{p}_i, \mathbf{p}_{i+1}) & \text{if } i \text{ is odd} \\ \text{dir}(\mathbf{p}_{i-1}, \mathbf{p}_i) & \text{if } i \text{ is even,} \end{cases} \qquad 1 \leq i \leq P. \tag{6}$$

Figure 2 shows an example of the results given by each one of the steps of the first proposed algorithm.

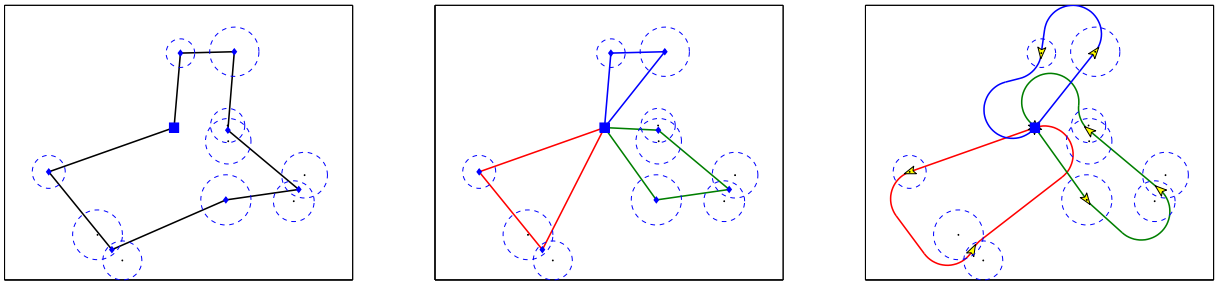
The upper bound for the length of the longest route obtained using Algorithm 1 can be obtained based on the bound of both techniques used (k-SPLITOUR and AA).

Theorem 3.4 from [22] demonstrates that the Dubins distance between two configurations is bounded by

$$\mathcal{D}_\rho(\mathbf{q}_c, \mathbf{q}_d) \leq \|\mathbf{p}_c - \mathbf{p}_d\| + \tau\rho\pi, \tag{7}$$

where $\tau \in [2.657, 2.658]$. Therefore, the upper bound for the length of a path obtained using AA is

$$\mathcal{L}_\rho(\boldsymbol{\mathcal{P}}, \Psi) \leq \text{ETSP}(\boldsymbol{\mathcal{P}}) + \left\lceil \frac{P}{2} \right\rceil \tau\rho\pi, \tag{8}$$



(a) (b) (c)

Fig. 2. Steps of Algorithm 1. (a) Circuit using TSP metrics; (b) resulting tours after executing k-SPLITOUR ; (c) Dubins paths obtained by the AA.

with $P \geq 2$ and $\rho > 0$.

According to Equations 5 and 8, a tour obtained by this technique has an upper bound in length given by

$$\mathcal{G} \leq \frac{1}{k}(L - 2c_{\max}) + 2c_{\max} + \left\lceil \frac{N}{2} \right\rceil \tau\rho\pi. \tag{9}$$

Regarding the time computational complexity, each of the steps (k-SPLITOUR and AA) runs in linear time $O(n)$ (excluding the complexity to solve the ETSP instances), where n is the number of waypoints ($P \leq N$).

C. Algorithm 2

Among the solutions for similar problems, evolutionary based ones have shown to be a good choice. Memetic Algorithms (MAs) (sometimes referred as Hybrid Genetic Algorithms) combine a population-based global technique to perform exploration and a local search method to perform exploitation [23], [24]. Figure 3 illustrates the basic steps of a MA.

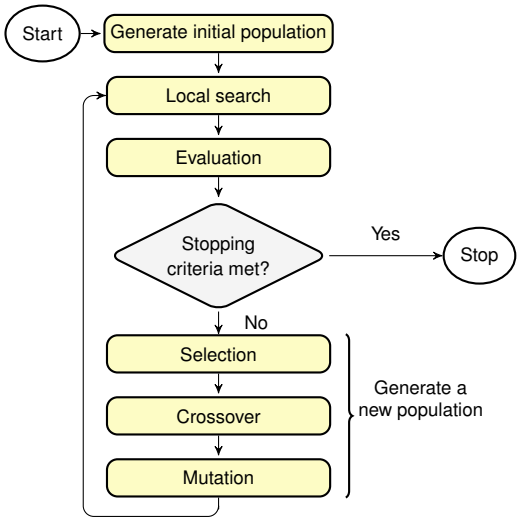


Fig. 3. Flowchart of a Memetic Algorithm.

One of the main steps in GAs is the choice of a good representation of the individual, which represents a valid candidate solution to the problem. Our chromosome is coded as a permutation of the waypoints $\mathbf{q}_c \in \boldsymbol{\mathcal{Q}}$ that must be visited, representing all possible tours ($\boldsymbol{\mathcal{T}}$). We represent the base as different waypoints for each tour, since the orientation at this specific point can be different among the tours.

Therefore, given the following example chromosome

$$\mathbf{q}_{\text{base}}^{(1)} - \mathbf{q}_1 - \mathbf{q}_2 - \mathbf{q}_3 - \mathbf{q}_{\text{base}}^{(2)} - \mathbf{q}_4 - \mathbf{q}_{\text{base}}^{(3)} - \mathbf{q}_5 - \mathbf{q}_6,$$

when decoded gives $\boldsymbol{\mathcal{T}} = \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}$ where

$$\begin{aligned}
\mathcal{T}_1 &= \{\mathbf{q}_{\text{base}}^{(1)}, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_{\text{base}}^{(1)}\},\\
\mathcal{T}_2 &= \{\mathbf{q}_{\text{base}}^{(2)}, \mathbf{q}_4, \mathbf{q}_{\text{base}}^{(2)}\},\\
\mathcal{T}_3 &= \{\mathbf{q}_{\text{base}}^{(3)}, \mathbf{q}_5, \mathbf{q}_6, \mathbf{q}_{\text{base}}^{(3)}\}.
\end{aligned}$$

The initial population is obtained by generating random individuals, i.e., different visiting sequences (leading to different tours). After decoding the chromosome into tours, we use the Alternating Algorithm to create initial feasible paths, since this method gives a good initial estimation.

Once the generation of the new population is complete, a local search procedure is applied in order to improve the fitness of each individual (as stated before, we aim to reduce the length of the longest tour. The local search is executed in two steps.

The first step searches for a placement of the waypoint inside a region that minimizes the Euclidean circuit formed by all configurations of a tour. The optimization process is based on the idea that a waypoint should be moved towards its neighbors as follows: For each waypoint i we define a target position $\mathbf{p}_{\text{dir}}$ as:

$$\mathbf{p}_{\text{dir}} = \begin{cases} \text{projection of } \mathbf{p}_i \text{ onto } \overline{\mathbf{p}_{i-1}\mathbf{p}_{i+1}} & \text{if } \alpha, \beta \leq 90^\circ \\ \text{middle point between } \mathbf{p}_{i-1} \text{ and } \mathbf{p}_{i+1} & \text{otherwise,} \end{cases}$$

where α and β are the angles presented at Figure 4. Formally, the direction of movement is given by:

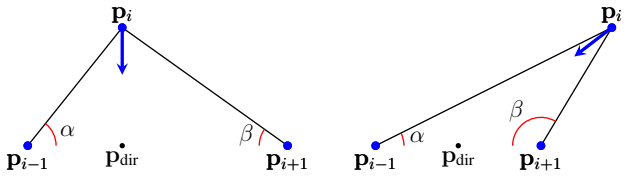
$$\eta = \text{dir}(\mathbf{p}_i, \mathbf{p}_{\text{dir}}), \tag{10}$$

where $\text{dir}(\mathbf{p}_m, \mathbf{p}_n)$ is defined as:

$$\text{dir}(\mathbf{p}_m, \mathbf{p}_n) \triangleq \frac{\mathbf{p}_n - \mathbf{p}_m}{\|\mathbf{p}_n - \mathbf{p}_m\|}. \tag{11}$$

The position of all waypoints are then optimized towards each $\mathbf{p}_{\text{dir}}$ in order to minimize $(\overline{\mathbf{p}_{i-1}\mathbf{p}_i} + \overline{\mathbf{p}_i\mathbf{p}_{i+1}})$, while still respecting the boundaries of the region (or a common region).

The second step of the local search is responsible for reassigning the orientation each waypoint in order to find a



(a) Projection along the height (b) Middle point

Fig. 4. Determination of the direction of movement for optimization of the position of waypoints. (a) Case where the height of the triangle formed by the neighbor points $\mathbf{p}_{i-1}, \mathbf{p}_i$ and $\mathbf{p}_{i+1}$ fall on the line segment $\overline{\mathbf{p}_{i-1}, \mathbf{p}_{i+1}}$, the direction of movement points to the point $\mathbf{p}_{\text{dir}}$ relative to the projection. (b) Case where the projection falls outside the triangle, the direction of movement points to the middle point $\mathbf{p}_{\text{dir}}$ of the segment $\overline{\mathbf{p}_{i-1}\mathbf{p}_{i+1}}$.

shorter Dubins curve connecting them. We make use of the Mean Angle algorithm proposed in [19], which states that the new orientation is evaluated as follows:

$$\psi_i = \frac{\mathbf{v}_{\text{prev}} + \mathbf{v}_{\text{next}}}{\|\mathbf{v}_{\text{prev}} + \mathbf{v}_{\text{next}}\|}, \tag{12}$$

where $\mathbf{v}_{\text{prev}}$ and $\mathbf{v}_{\text{next}}$ are direction vectors between neighbor waypoints (see Figure 5) and defined as follows:

$$\mathbf{v}_{\text{prev}} = \text{dir}(\mathbf{p}_{i-1}, \mathbf{p}_i) \quad \text{and} \quad \mathbf{v}_{\text{next}} = \text{dir}(\mathbf{p}_i, \mathbf{p}_{i+1}). \tag{13}$$

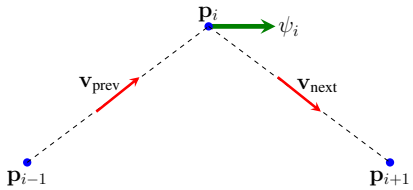


Fig. 5. Evaluation of a given waypoint's orientation based on the positions of neighbor waypoints. The green arrow is the resulting orientation for the waypoint $\mathbf{p}_i$, as shown in Equation (12).

It is important to remind that the results obtained during the local search phase will only be kept if they are better than the previous found solutions.

The steps related to the classical GA execution (e.g. Crossover and Mutation) are responsible for dealing with the combinatorial part of the problem (sequence of visiting). The hybrid step (local search) will be focusing on the continuous part of the problem (position and orientation of the waypoints).

Recalling that all robots travel at the same constant speed, the upper bound for the time spent to visit all nodes and return to the base is defined by the robot that traves the longest tour. The worst case is when a single robot is selected to visit all regions (which may happen due to the random nature of the initial population generation). Therefore, any tour obtained by this technique has the same upper bound in length given by Equation 8, which represents the length of a single tour given by the AA.

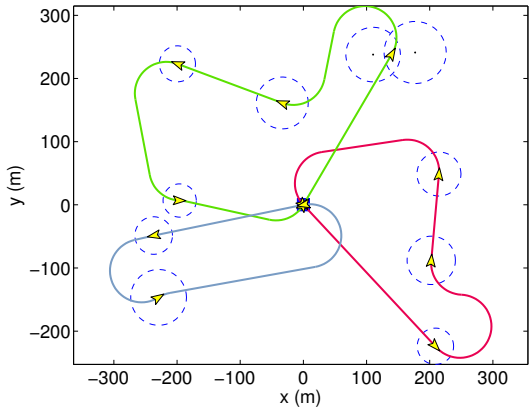
The method runs in polynomial time $O(nm)$, where n is the number of regions and m is the size of the population. Assuming a constant amount of individuals, the execution time cost is linear and depends only on the number of regions in the environment.

IV. EXPERIMENTS

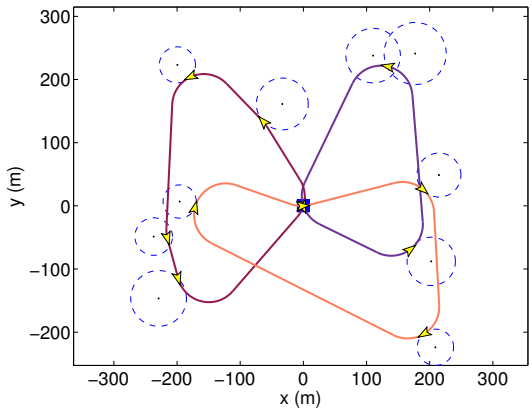
In this section, we describe our experiments and the corresponding statistical analysis. All ETSP problems instances were solved to optimality using the well-known TSP solver *Concorde* [25].

For better visualization purposes, we initially present an example of possible results given by both techniques for a small instance. In this first example we have 10 regions randomly placed in an environment with dimensions $500\,\text{m} \times 500\,\text{m}$. The regions are to be visited by 3 vehicles, each one with a minimum curvature radius of $\rho = 50\,\text{m}$. The radius of each region was randomly chosen from the interval $[25\,\text{m}, 50\,\text{m}]$.

Figures 6(a) and 6(b) present the resulting tours obtained by Algorithm 1 and Algorithm 2, respectively. One may clearly observe in Figure 6(b) the benefits of the local search in Algorithm 2, where the visiting waypoints (marked as yellow arrows) have changed.



(a) 1,337.16 m



(b) 1,075.15 m

Fig. 6. Example solutions obtained by (a) Algorithm 1 and (b) Algorithm 2. The length of the longest tour is presented. The base is at the blue square.

Since the proposed technique is probabilistic in nature, in order to perform a thorough statistical analysis we present next an overall analysis with a significant number of experiments. We ran 200 experiments in an environment with dimensions $1{,}500\,\text{m} \times 1{,}500\,\text{m}$ containing a total of 30 regions whose positions were uniformly randomly distributed.

Each region had a fixed radius chosen randomly from the interval $[50\,\mathrm{m}, 100\,\mathrm{m}]$. For each experiment, 3 vehicles with a minimum curvature radius of $\rho = 100\,\mathrm{m}$ are available to perform the task. Parameters related to the algorithm execution are presented in Table I.

TABLE I
PARAMETERS USED IN ALGORITHM 2.

Parameter	Value
Population size	100
Number of generations	50
Selection method	Tournament of size 2
Elitism	yes
Crossover probability	0.8
Crossover operator	Order-based (OX2)
Mutation probability	0.5
Mutation operator	Simple inversion

Figure 7 presents a histogram of the final result ratio obtained by Algorithm 1 compared to the results given by Algorithm 2. It is important to remember that ours cost function is the length of the longest tour (not the sum of all tours). We can see that Algorithm 1 produced paths that are approximately 20% in average longer than Algorithm 2 (std. dev. $\approx 8\%$) for about 97% of the instances. The mean ratio is significantly larger than zero, $t(199) = 26.64$, two-tail $p = 0$.

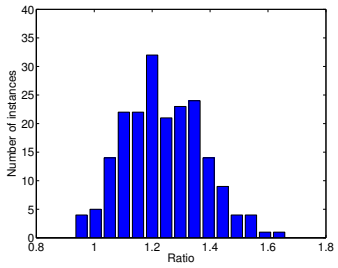
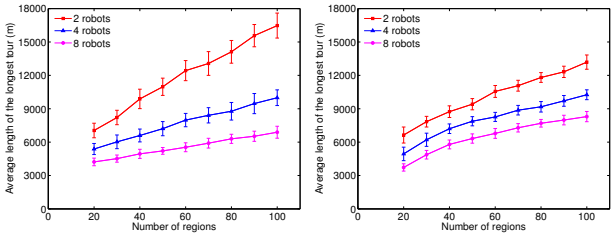


Fig. 7. Ratio between the results (length of the longest tour) obtained by Algorithm 1 vs. Algorithm 2.

The next experiment was conducted in order to empirically assess the asymptotic behavior of the quality of the results (length of the longest tour). Simulations were conducted with varying amounts of regions from the set $\{20, 30, ..., 100\}$, and for each value 30 random instances were generated. We used an environment with dimensions $2{,}500\,\mathrm{m} \times 2{,}500\,\mathrm{m}$ and the radius of each region was chosen randomly from the interval $[50\,\mathrm{m}, 100\,\mathrm{m}]$. For each instance the maximum number of robots to be used was selected from the set $\{2, 4, 8\}$, all having a minimum curvature radius of $\rho = 100\,\mathrm{m}$. The parameters used by Algorithm 2 were the same as shown in Table I, except for the size of the population, now composed by 50 individuals.

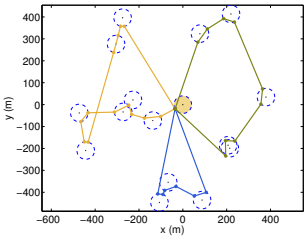
Figure 8 presents the results. With a smaller amount of regions and robots, Algorithm 2 obtains better results. However, with the increase in these amounts the results of Algorithm 1 become better. One reason for this is the use a low number of generations since a greater number of robots requires more iterations to find good sequences (permutations) of visiting for all routes.



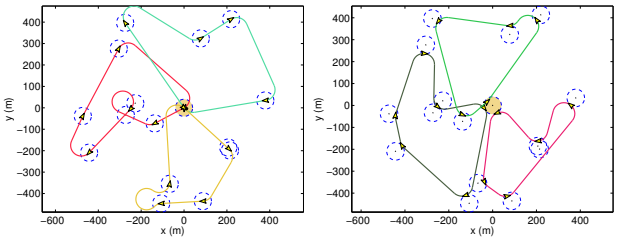
(a) Algorithm 1 (b) Algorithm 2

Fig. 8. Asymptotic behavior of the average length of the longest tour in accordance with the number of regions in the environment and number of robots used.

As mentioned before, no previous works addressing the problem presented here (k-DTSPN) have been found in literature. Thus, we chose to perform a comparison with a technique addressing a similar problem, the k-Traveling Salesman Problem with Neighborhoods (k-TSPN). The work of [5] presents an algorithm for data gathering in WSNs using mobile robots called *DGPTour*. The DGPTour is a constant factor $\gamma + 2 - 1/k$ away from the optimal tour, where γ is the approximation ratio of the algorithm used to find the TSPN tour. Figure 9 illustrates the paths obtained by the techniques for a given instance.



(a) DGPTour [5] (1,619.23 m)

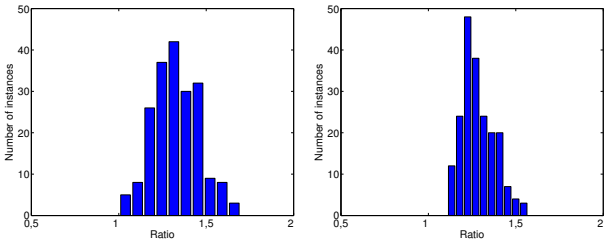


(b) Algorithm 1 (2,002.40 m) (c) Algorithm 2 (1,762.95 m)

Fig. 9. Example presenting the resulting paths obtained by the different techniques being compared. The center circle highlighted in orange represents the base position.

We performed 200 comparative experiments for an environment with dimensions $1{,}000\,\mathrm{m} \times 1{,}000\,\mathrm{m}$. For each experiment, 40 regions were distributed uniformly at random, all having a radius of 40 m. Four vehicles are available to solve the problem, each one with a minimum radius of curvature $\rho = 50\,\mathrm{m}$. The parameters used by Algorithm 2 are the same as shown in Table I. For each instance Algorithm 2 was executed 10 times, and the average values were used for comparison.

Figure 10 presents ratio of the length of the longest tour histograms found by the proposed methods (Algorithm 1 and Algorithm 2) relative to the technique used as the basis for comparison (DGPTour).



(a) Algorithm 1 vs. DGPTour (b) Algorithm 2 vs. DGPTour

Fig. 10. Ratio histograms between the length of the longest tour. (a) Algorithm 1 vs. DGPTour; (b) Algorithm 2 vs. DGPTour.

The mean ratio found by Algorithm 1 was about 1.33, while Algorithm 2 returned values 1.28 greater than the largest tour found by DGPTour on average. However, it is interesting to note that, although Algorithm 1 presents on average worse results than Algorithm 2, in some cases it found better tours, with a ratio slightly above 1.

V. CONCLUSIONS AND FUTURE WORK

This work introduced the k-Dubins Traveling Salesman Problem with Neighborhoods, which consists of planning efficient paths among regions for multiple vehicles with bounded curvature. Both proposed techniques presented good results, with Algorithm 2 producing paths 20% shorter in average than Algorithm 1. Both methods produced results on average 30% longer than the ones obtained for the k-TSPN using the DGPTour [5].

Future directions include the extension of the proposed methodology to groups of heterogeneous robots (i.e., with different curvature constraints and velocities) and to environments containing obstacles, a scenario that best represents real world cases. We also intend to improve the local search procedure combining both steps, tackling the problems of finding the best placement and orientation of waypoints jointly.

A known problem concerning the use of Dubins curves is the discontinuity of the curvature derivative, leading to abrupt lateral accelerations that makes paths that are not actually feasible by real robots. Therefore, we intend to study possible techniques for generating smoother variations of acceleration, e.g., the use of other types of curves such as clothoids.

ACKNOWLEDGMENTS

This work was developed with the support of the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) and Fundação Centro de Análise, Pesquisa e Inovação Tecnológica (FUCAPI).

REFERENCES

[1] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton, NJ, USA: Princeton University Press, 2007.

[2] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[3] P. Toth and D. Vigo, Eds., *The vehicle routing problem*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001.

[4] O. Tekdas, V. Isler, J. Lim, and A. Terzis, "Using mobile robots to harvest data from sensor fields," *WCI*, vol. 16, pp. 22–28, Feb. 2009.

[5] D. Bhadauria, O. Tekdas, and V. Isler, "Robotic data mules for collecting data over sparse sensor fields," *Journal of Field Robotics*, vol. 28, no. 3, pp. 388–404, 2011.

[6] K. Savla, E. Frazzoli, and F. Bullo, "On the point-to-point and traveling salesperson problems for Dubins' vehicle," in *Proc. of the IEEE ACC*, vol. 2, June 2005, pp. 786–791.

[7] Z. Tang and Ü. Özgüner, "Motion planning for multitarget surveillance with mobile sensor agents," *IEEE Trans. on Robotics*, vol. 21, no. 5, pp. 898–908, Oct. 2005.

[8] X. Ma and D. A. Castañón, "Receding horizon planning for Dubins traveling salesman problems," in *Proc. of IEEE CDC*, Dec. 2006, pp. 5453–5458.

[9] J. Le Ny, E. Frazzoli, and E. Feron, "The curvature-constrained traveling salesman problem for high point densities," in *Proc. of IEEE CDC*, Dec. 2007, pp. 5985–5990.

[10] S. Rathinam, R. Sengupta, and S. Darbha, "A resource allocation algorithm for multivehicle systems with nonholonomic constraints," *IEEE Trans. on Autom. Sci. and Eng.*, vol. 4, pp. 98–104, Jan. 2007.

[11] J. J. Enright, K. Savla, E. Frazzoli, and F. Bullo, "Stochastic and dynamic routing problems for multiple UAVs," *AIAA JGCD*, vol. 32, no. 4, pp. 1152–1166, 2009.

[12] E. M. Arkin and R. Hassin, "Approximation algorithms for the geometric covering salesman problem," *Discrete Applied Mathematics*, vol. 55, pp. 197–218, Dec. 1994.

[13] B. Yuan, M. Orlowska, and S. Sadiq, "On the optimal robot routing problem in wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 9, pp. 1252–1261, Sept. 2007.

[14] G. Comarela, K. Gonçalves, G. L. Pappa, J. Almeida, and V. Almeida, "Robot routing in sparse wireless sensor networks with continuous ant colony optimization," in *Proc. of GECCO*, New York, NY, USA, 2011, pp. 599–606.

[15] K. J. Obermeyer, "Path planning for a UAV performing reconnaissance of static ground targets in terrain," in *Proc. of AIAA CGNC*, Chicago, IL, USA, Aug. 2009.

[16] K. J. Obermeyer, P. Oberlin, and S. Darbha, "Sampling-based roadmap methods for a visual reconnaissance UAV," in *Proc. of AIAA CGNC*, Toronto, Canada, Aug. 2010.

[17] J. T. Isaacs, D. J. Klein, and J. P. Hespanha, "Algorithms for the traveling salesman problem with neighborhoods involving a Dubins vehicle," in *Proc. of IEEE ACC*, 2011, pp. 1704–1709.

[18] D. G. Macharet, A. Alves Neto, V. F. da Camara Neto, and M. F. M. Campos, "An evolutionary approach for the Dubins' traveling salesman problem with neighborhoods," in *Proceedings of the 21th Genetic and Evolutionary Computation Conference*, 2012.

[19] ——, "Data gathering tour optimization for Dubins' vehicles," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2012.

[20] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[21] G. N. Frederickson, M. S. Hecht, and C. E. Kim, "Approximation algorithms for some routing problems," *SIAM Journal on Computing*, vol. 7, no. 2, pp. 178–193, 1978.

[22] K. Savla, E. Frazzoli, and F. Bullo, "Traveling salesperson problems for the Dubins vehicle," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1378–1391, July 2008.

[23] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," Caltech Concurrent Computation Program, Tech. Rep. C3P Report 826, 1989.

[24] P. Moscato and C. Cotta, "A modern introduction to memetic algorithms," in *Hand. of Metaheuristics*, ser. Int. Series in Op. Research & Management Sci. Springer, 2010, vol. 146, pp. 141–183.

[25] "Concorde TSP solver," http://www.tsp.gatech.edu/concorde/index.html, 2012, [Online; accessed 19-October-2012].