Discretization-Based and Look-Ahead Algorithms for the Dubins Traveling Salesperson Problem

Izack Cohen, Chen Epstein, Pantelis Isaiah, Saar Kuzi, and Tal Shima

***Abstract*— A new class of discretization-based look-ahead algorithms (DLAAs) for the Dubins traveling salesperson problem (DTSP) is presented that compares favorably with the existing algorithms from the literature. The discretization level and the length of the look-ahead horizon are the two parameters that uniquely determine a DLAA, and depending on the application in hand, their values can be easily modified to strike a balance between the execution time and the length of the resulting admissible tour. The time complexity of a DLAA is the sum of two terms, one linear in the number of targets (cities) and one that corresponds to the specification of an initial order for the targets. For instances of the DTSP with densely distributed targets, an algorithm that relies on clustering and leads to shorter tours than the DLAA is also presented.**

***Note to Practitioners*—The new class of algorithms, suggested in this paper, not only compares favorably with the existing ones, but is also highly adjustable and scalable. Extensive experiments demonstrate that execution times are reasonable for offline optimization procedures. We demonstrate the implementation of our ideas using a Pioneer D3-PX mobile robot that imitates a Dubins vehicle.**

***Index Terms*— Dubins vehicle, motion planning, traveling salesperson problem.**

I. Introduction

THE use of unmanned aerial vehicles (UAVs) in remote sensing, surveillance, search and rescue operations, minesweeping, reconnaissance and intelligence gathering [1], [2], and seafloor surveying [3] has led to a surge of interest in motion planning and co-operative control of multiple UAVs. A common problem in many application areas that involve UAVs is the generation of trajectories that pass through the sequences of targets and minimize the traveling distance or time. Although such problems strongly resemble the traveling salesperson problem, they tend to be more difficult to solve and require new algorithms because of the presence of kinematic constraints. In particular, because many UAVs are fixed-wing aircraft, their trajectories have bounded curvature; that is, arbitrarily sharp turns are not possible. A model for planar motion that is widely employed in this regard is the so-called *Dubins vehicle* [4]–[7]

$$\begin{aligned} \dot{x} &= \cos\theta \\ \dot{y} &= \sin\theta \\ \dot{\theta} &= \upsilon \end{aligned} \tag{1}$$

where $(x, y, \theta) \in \mathbb{R}^2 \times [0, 2\pi[$ and $\upsilon \in [-1/\rho, 1/\rho]$, with $\rho > 0$ being the minimum turn-radius of the vehicle. By a slight abuse of terminology, we refer both to (1) and to a vehicle modeled by (1) as Dubins vehicles. It follows from (1) that a Dubins vehicle moves at unit speed along planar paths, whose maximum curvature is equal to $1/\rho$. The Dubins traveling salesperson problem (DTSP) is a problem formulated in analogy with the Euclidean traveling salesperson problem (ETSP) [8] and offers a mathematical abstraction of scenarios, where a vehicle has to visit a set of *a priori* given locations in the plane. In the case of the DTSP, a vehicle modeled by (1) is envisaged playing the role of the traveling salesperson [9]–[15], and the unknowns consist of the order in which the targets should be visited and the heading of the vehicle at each target. Given a set T of $n > 1$ points in the plane, solving a DTSP consists in finding a closed curve $t \mapsto (x(t), y(t))$ of minimum length that passes through every point in T, with x and y being the first two components of a solution (x, y, θ) to (1). An admissible (i.e., not necessarily optimal) solution to the DTSP is a planar path that passes through every point in T and that can be traced out by a Dubins vehicle. For economy of language, an admissible solution $t \mapsto (x(t), y(t))$ is referred to as *Dubins tour*.

In the remainder of this section, some representative contributions from the literature on the DTSP are recalled. The prevalence of approximation and heuristic algorithms is due to the fact that similar to the ETSP [16], the DTSP is also NP-hard [17].

In [13], the *alternating algorithm* (AA) is introduced that constructs a Dubins tour by replacing the even-numbered edges of an ETSP tour with Dubins paths. The headings at each pair of targets connected by a Dubins path are the slopes of the adjacent edges, so that the overall tour is a continuously differentiable curve. For instances of the DTSP with a high density and a large number of targets, a second algorithm, called *bead-tiling algorithm* (BTA), is described in [13]. Unlike the AA, the BTA does not solve an ETSP to order the targets. Instead, the targets are visited by repeatedly sweeping an area in which the targets are contained.

Since a Dubins vehicle cannot perform arbitrarily sharp turns to instantly reorient itself toward the next target, a natural approach to the path-planning aspect of the DTSP is to employ a receding horizon or look-ahead principle. That is, to anticipate the location of more than one subsequent targets. The algorithms in [12] and [18] belong to this category; they order the targets by solving the corresponding ETSP and consider a short sequence of targets at a time (typically, up to three). The single-vehicle algorithm (SVA) algorithm presented in [12] looks one target ahead per iteration, according to the ETSP order, and is designed primarily for the cases when the Euclidean distance between any two targets is larger than twice the vehicle's

Manuscript received September 30, 2015; revised February 3, 2016 and June 14, 2016; accepted August 9, 2016. Date of publication September 20, 2016; date of current version January 4, 2017. This paper was recommended for publication by Associate Editor L. Tang and Editor J. Wen upon evaluation of the reviewers' comments. This work was supported by the Technion Autonomous Systems Program.

I. Cohen, C. Epstein, and S. Kuzi are with the Faculty of Industrial Engineering and Management, Technion–Israel Institute of Technology, Haifa 32000, Israel (e-mail: izik68@tx.technion.ac.il; chenep@tx.technion.ac.il; saarku@mail.technion.ac.il).

P. Isaiah and T. Shima are with the Faculty of Aerospace Engineering, Technion–Israel Institute of Technology, Haifa 32000, Israel (e-mail: pantelis.isaiah@gmail.com; tal.shima@technion.ac.il).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2016.2602385

minimum turn-radius. The current target, i.e., the one where the Dubins vehicle is located, is connected to the next one by a relaxed Dubins path, that is, the shortest path between a given initial position and orientation of the Dubins vehicle and a final position (the next target) with unconstrained final orientation [19]. The look-ahead algorithm (LAA) in [18] can be considered a natural extension of the SVA in the sense that the LAA looks two targets ahead of the current target. The continuity properties (or lack thereof) of the adjoint state and the dynamic programming principle are used in [18] to reduce the number of candidate optimal Dubins paths through a sequence of three targets.

When the minimum turn-radius ρ is comparable with the diameter of the area that contains the targets, ordering the targets by first solving the corresponding ETSP leads to Dubins tours that, in general, are far from optimal. Therefore, algorithms for the DTSP that break away from the dependence on a solution to the ETSP have been developed in [9], [10], [14], [20], and [21]. In [20], the k-step LAA (k-step LAA) also relies on a receding horizon principle; however, the combinatorial and motion planning aspects of the DTSP are integrated and reduced to a problem of constructing and searching a finite tree. In the form presented in [20], the k-step LAA has a computational complexity that restricts its applicability to instances of the DTSP with relatively a few targets (ten or less). To address this limitation, a randomized local-improvement algorithm, called 2-Opt k-step LAA, is introduced in [20]. The algorithms in [9], [10], and [17] dispense with the dependence on solving the ETSP by discretizing the DTSP. Specifically, it is assumed that, when the Dubins vehicle passes through a target, its heading θ can assume finitely many values in $[0, 2\pi[$. In [21], the resulting discretized DTSP (DDTSP) is formulated as a mathematical program that is solved to optimality by commercial solvers for scenarios with up to 30 targets and 2^5 possible values for the heading at each target. In the same reference, sharp bounds on the length of a DTSP tour are established that are independent of the distances between the targets and, also, imply that the tour length decreases as the discretization level increases—a fact also confirmed numerically. For larger instances of the DTSP, the DDTSP can be transformed into a generalized asymmetric traveling salesperson problem (GATSP) for which there exist standard effective algorithms [17, Sec. IV-B]. The problem that results from discretization and transformation of the DDTSP into the GATSP is denoted by DDTSP$'$ in the following.

The algorithms presented herein stem from a synthesis of discretization, as described in the previous paragraph, and a look-ahead principle. Given an instance of the DTSP, a solution (or an approximation thereof) to the corresponding ETSP is first constructed to order the targets. Then, starting from the first target and considering $L-1$ subsequent targets, L being the look-ahead horizon, the problem of finding an admissible Dubins path through the L targets is discretized, and the resulting mixed integer linear programming (MILP) problem is solved to optimality. The same procedure is iterated until a tour that passes through every target is constructed. It should be noted that solving the discretized subproblems may change the initial order of the targets, and therefore, the final order of the targets is not, in general, that of the ETSP solution. Moreover, considering L targets at a time is computationally more efficient than finding a global solution to the DDTSP or the DDTSP$'$. Moreover, the algorithms can be implemented in such a way that the look-ahead horizon and the discretization level can be easily altered.

II. Model, Notation, and Assumptions

Let $T = \{T_1, \ldots, T_n\}$ be a set of points in the plane that represents the targets to be visited by a Dubins vehicle initially located at T_1. To each target, T_i corresponds a pair of Cartesian coordinates (x_i, y_i). The vehicle's initial heading may be constrained (for example, an aircraft taking off from a runway or a marine vehicle leaving a mooring) or unconstrained. As noted in Section I, the discretization of the DTSP amounts to letting the heading of the Dubins vehicle takes only finitely many values in $[0, 2\pi[$ when the vehicle passes through a target $T_i \in T$, $i = 1, \ldots, n$. To this end, let $h = 2^{\hat{h}}$ be the *discretization level*, where $\hat{h} \in \{0, 1, 2, \ldots\}$, and let $H = \{\theta_k : \theta_k = 2\pi k/h, k = 0, 1, \ldots, h-1\}$ be the set of admissible values for θ when $(x, y) = (x_i, y_i)$. The heading θ is measured counterclockwise from the x-axis.

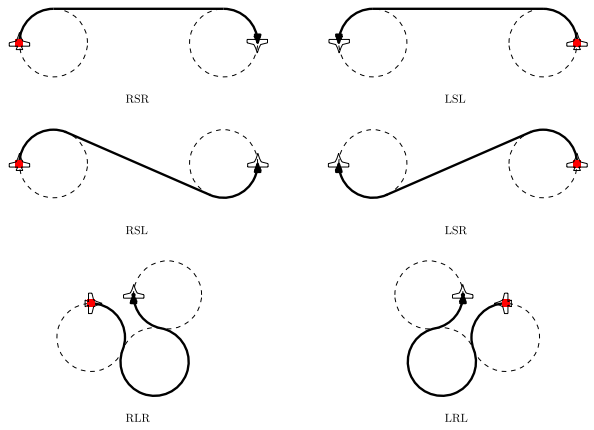


Fig. 1. Six Dubins paths that constitute a sufficient family of candidate optimal paths between any two given configurations.

The information associated with the DDTSP can be encoded to a graph $G = (V_T^H, E)$, where $V_T^H = \{(T_1, \theta_0), \ldots, (T_1, \theta_{h-1}), \ldots, (T_n, \theta_{h-1})\}$ is the set of nodes in the graph, and $E = \{(v_i^{\theta_k}, v_j^{\theta_m}) | v_i^{\theta_k}, v_j^{\theta_m} \in V_T^H, \forall i, \forall j, i \neq j, \forall m, \forall k = 0, 1, \ldots, h-1\}$ is the set of edges. Each node $v_i^{\theta_k} = (T_i, \theta_k) \in V_T^H$ is uniquely defined by the position of a target T_i and an admissible value θ_k for the heading, and there are $n \cdot h$ nodes in the graph. Each edge $(v_i^{\theta_k}, v_j^{\theta_m}) \in E$ links two nodes in V_T^H, thus representing the length of a path going from target (T_i, θ_k) to (T_j, θ_m). The weight of each edge $d_{(v_i^{\theta_k}, v_j^{\theta_m})}$ is defined to be the length of a shortest Dubins path that connects the two configurations (T_i, θ_k) and (T_j, θ_m) of the Dubins vehicle, and therefore, any other choice of path does not yield a shorter tour, if the order of the targets and the heading at each target is kept fixed. The value of $d_{(v_i^{\theta_k}, v_j^{\theta_m})}$ is found by searching for the shortest path in a sufficient family $\mathcal{F}$ of optimal paths between (T_i, θ_k) and (T_j, θ_m). If L and R denote the left and right turns along the circular arcs of radius ρ, and S denotes a straight line segment, then $\mathcal{F} = \{LSL, RSR, RSL, LSR, RLR, LRL\}$ (see Fig. 1) [5]–[7]. The objective is to minimize the total distance traveled by a Dubins vehicle; the analytic expression for the cost function is given in (2), where $X_{(v_i^{\theta_k}, v_j^{\theta_m})} \in \{0, 1\}$ are the binary decision variables that equal 1 if the vehicle moves from target T_i to target T_j with headings θ_k and θ_m, respectively, and zero otherwise

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=0}^{h-1} \sum_{m=0}^{h-1} X_{\left(v_i^{\theta_k}, v_j^{\theta_m}\right)} d_{\left(v_i^{\theta_k}, v_j^{\theta_m}\right)}. \tag{2}$$

Given a look-ahead horizon L, the algorithms described in Sections III and V use different approaches to choose from T a subset of L targets at a time, but they both solve Program 3 in the following, to find an admissible Dubins path through the L targets once they have been selected:

Program 3:

$$\min \sum_{i=1}^{L}\sum_{j=1}^{L}\sum_{k=0}^{h-1}\sum_{m=0}^{h-1} X_{\left(v_i^{\theta_k},v_j^{\theta_m}\right)} d_{\left(v_i^{\theta_k},v_j^{\theta_m}\right)} \tag{3a}$$

s.t.:

$$\sum_{i=1}^{L}\sum_{k=0}^{h-1}\sum_{m=0}^{h-1} X_{\left(v_i^{\theta_k},v_j^{\theta_m}\right)} = 1 \quad \forall j = 1,\ldots,L \tag{3b}$$

$$\sum_{j=1}^{L}\sum_{m=0}^{h-1}\sum_{k=0}^{h-1} X_{\left(v_i^{\theta_k},v_j^{\theta_m}\right)} = 1 \quad \forall i = 1,\ldots,L \tag{3c}$$

$$\sum_{i=1}^{L}\sum_{k=0}^{h-1} X_{\left(v_i^{\theta_k},v_j^{\theta_m}\right)} - \sum_{t=1}^{L}\sum_{s=0}^{h-1} X_{\left(v_j^{\theta_m},v_t^{\theta_s}\right)} = 0 \quad \forall j = 1,\ldots,L \quad \forall m = 0,\ldots,h-1 \tag{3d}$$

$$w_i - w_j + (L-1)\sum_{k=0}^{h-1}\sum_{m=0}^{h-1} X_{\left(v_i^{\theta_k},v_j^{\theta_m}\right)} + (L-3)\sum_{k=0}^{h-1}\sum_{m=0}^{h-1} X_{\left(v_j^{\theta_m}v_i^{\theta_k}\right)} \le L-2 \quad \forall i,j = 2,\ldots,L \tag{3e}$$

$$w_1 = 1; \quad 2 \le w_i \le L \ \forall i = 2,\ldots,L \tag{3f}$$

$$X_{\left(v_i^{\theta_k},v_j^{\theta_m}\right)} = 0 \quad \forall i = j$$

$$\text{otherwise } X_{\left(v_i^{\theta_k},v_j^{\theta_m}\right)} \in \{0,1\} \quad \forall i,j = 1,\ldots,L. \tag{3g}$$

Constraints (3b) and (3c) restrict the vehicle to one visit per target. Constraint (3d) ensures that for each target, the headings are identical when entering and exiting a target. Constraints (3e) and (3f) enforce a single continuous subpath through the L targets. The w_i's are nonnegative integers that represent the order of target T_i. These subpath elimination constraints are an improved version, described in [22], of the Miller–Tucker–Zemlin formulation [23].

The mathematical program includes $hL(hL-h)$ binary variables and $L-1$ continuous variables (w_1 is not taken into account), summing up to $h^2L^2 + (1-h^2)L - 1$ variables. Due to symmetry considerations, edges $(v_i^{\theta_k}, v_j^{\theta_m})$ and $(v_j^{\theta_m+\pi}, v_i^{\theta_k+\pi})$ represent the same Dubins path traversed in opposite directions and, hence, have the same weight. That is, $d_{(v_i^{\theta_k},v_j^{\theta_m})} = d_{(v_j^{\theta_m+\pi},v_i^{\theta_k+\pi})}$. This symmetry reduces the number of binary variables (or edges) by half [10].

III. Discretized Look-Ahead Algorithm

Suppose an instance of the DTSP is given and values for the look-ahead horizon L and the discretization level h are chosen. The discretization-based look-ahead algorithm (DLAA)—the first of the two algorithms introduced in this paper—consists of the following steps. First, the corresponding ETSP is solved to determine an initial order $(T_1,\ldots,T_n)$ in which the targets are to be visited. Depending on the number of targets, the solution to the ETSP can be either optimal or approximate. Then, the problem of finding a shortest Dubins path through the first L targets is discretized, and the resulting MILP (3a)–(3g) is solved. In accordance with the general philosophy of a look-ahead principle, the last segment of the computed path that connects T_{L-1} to T_L is discarded and a new path through targets $T_{L-1},\ldots,T_{2L-2}$ is computed. The same procedure is iterated until a Dubins tour through all the targets is obtained. The suggested algorithms are flexible in the sense that the look-ahead parameter, the discretization level, and the overlap between consecutive iterations can be easily changed. A path through L targets constructed during an iteration of the DLAA is called an open DDTSP path. When $L = n$, there is only one iteration and the resulting open DDTSP path is one final Dubins path that connects the last target to T_1a short of a complete solution to the DDTSP. Global solutions to the DDTSP, for a given level of discretization, are obtained and analyzed in [21].

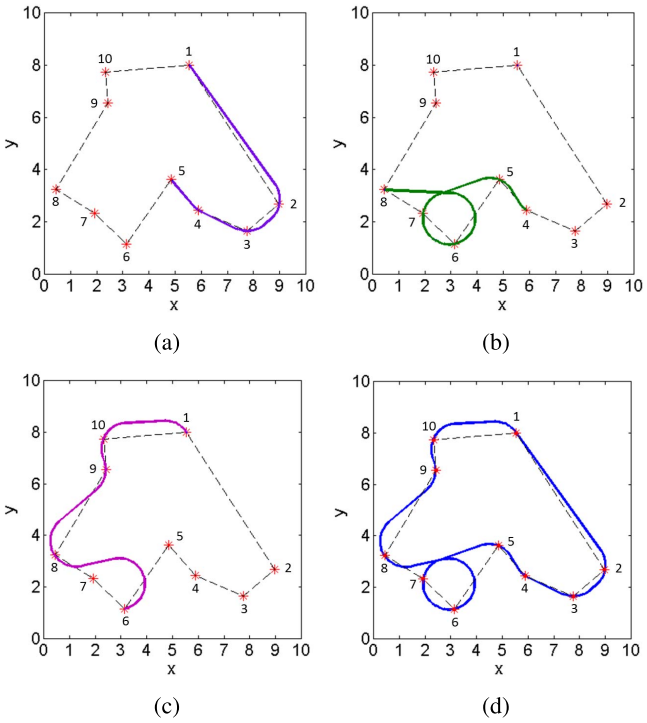


Fig. 2. Path planning by DLAA for a scenario with ten targets, a minimum turning radius of 1, $h = 32$, and $L = 5$. Solid lines mark the DLAA resulting path and dashed lines describe an ETSP path. (a)–(c) Correspond to iterations 1–3 of the algorithm, respectively. (d) Compares the overall DLAA path to an ETSP path.

Given an instance of the DTSP, the details of the DLAA are as follows.

1) Order the targets $\{T_1,\ldots,T_n\}$ by finding a feasible solution for the corresponding ETSP.
2) Choose values for $L \in \{3,\ldots,n\}$, $h \in H$, and ρ.
3) For iteration $m = 1,\ldots,M$, the following holds.
   a) If $m = 1$, calculate an open DDTSP path for the first L targets $(T_1,\ldots,T_L)$ to obtain $\{(T_i,\theta_i)\}_{i\in\{1,\ldots,L\}}$, where $\theta_i \in H$.
   b) If $2 \le m \le M-1$, calculate an open DDTSP path for the last two targets of iteration $m-1$ and the next $L-2$ targets from T, i.e., for the sequence $(T_{1+(m-1)\cdot(L-2)},\ldots,T_{m\cdot(L-2)+2})$ of targets, to obtain $\{(T_i,\theta_i)\}_{i\in\{1+(m-1)\cdot(L-2),\ldots,m\cdot(L-2)+2\}}$, where $\theta_i \in H$. During each iteration m, the configuration (position and heading) of the Dubins vehicle at the first target $T_{1+(m-1)\cdot(L-2)}$ of the sequence is the one already computed in the previous iteration, and the position of the last target $T_{m\cdot(L-2)+2}$ is determined according to the ETSP visiting order from Step 1).
   c) If $m = M$ such that $m\cdot(L-2)+1 \ge n$, set $(T_{n+1},\theta_{n+1}) := (T_1,\theta_1)$ and calculate an open DDTSP path for the last two targets of iteration $M-1$ and for the remaining targets from T, i.e., $(T_{1+(M-1)\cdot(L-2)},\ldots,T_{n+1})$, to obtain $\{(T_i,\theta_i)\}_{i\in\{1+(M-1)\cdot(L-2),\ldots,n+1\}}$, where $\theta_i \in H$.
4) Return the sequence of configurations $\{(T_i,\theta_i)\}_{i\in\{1,\ldots,n\}}$.

Fig. 2 shows the DLAA for a scenario with $n = 10$ targets, minimum turn-radius $\rho = 1$, discretization level $h = 32$, and

look-ahead horizon $L = 5$. It shows the three iterations that the algorithm performs and compares the output of the algorithm (a Dubins tour) to the solution to the ETSP. The order in which the targets are visited is different for the DTSP and the ETSP: the optimal ETSP tour passes through the targets T_i successively (in other words, the indexing coincides with the optimal order), whereas the order of the targets that corresponds to the DTSP tour is $(1, 2, 3, 4, 5, 7, 6, 8, 9, 10)$.

A. Lower and Upper Bounds on the Output of the DLAA

For a given set of targets, a lower bound on the output of the DLAA is the length of the ETSP, which is also a lower bound for the length of an optimal solution to the DTSP. Since the length of the Dubins tour constructed by the DLAA decreases as L and h increase, and because the tour is the same as the solution to the DDTSP [21] for the same discretization level when $L = n$, the following inequalities hold for $\tilde{h} \leq h$ and $L = 3, \ldots, n$:

$$l_{\mathrm{DLAA}(L,\tilde{h})} \geq l_{\mathrm{DLAA}(L,h)} \geq l_{\mathrm{DLAA}(L=n,h)} = l_{\mathrm{DDTSP}(h)}.$$

The method used in [21, Th. 2] to show that

$$\frac{l_{\mathrm{DDTSP}}}{l_{\mathrm{DTSP}}} \leq 1 + \frac{nF(1, \frac{2\pi}{h})}{2\pi}$$

where $F(1, ((2\pi)/h))$ is a transcendental function (for more details refer to [21]), provides an upper bound to $l_{\mathrm{DLAA}(L=n,h)}$, and can be used to find an upper bound on the output of the DLAA by applying it separately to each iteration of the DLAA.

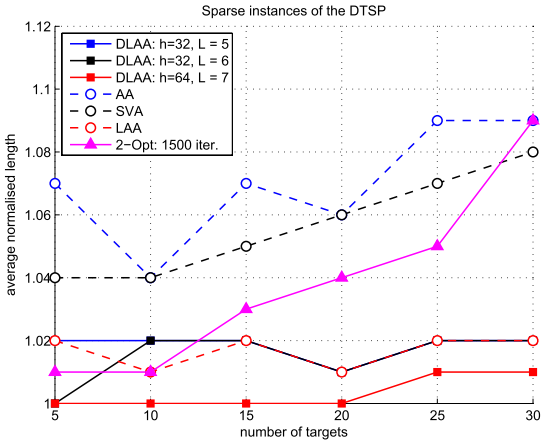
IV. Numerical Comparison of the DLAA With Other Algorithms

In this section, the DLAA is compared by means of simulations with the following algorithms from the literature:

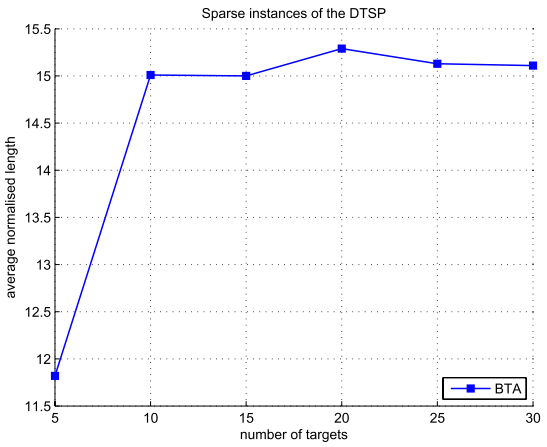
1) the AA [13];
2) the BTA, ibid.;
3) the SVA [12];
4) the LAA [18];
5) the 2-Opt two-step LAA [20];
6) the DDTSP′ [21].

The output of the DDTSP′ is used as a reference, so that the comparison between all algorithms is presented in terms of how longer are the tours they return than the tours constructed by the DDTSP′. The reason for this choice is that the DDTSP′ approximates global solutions to the DTSP. All instances of the DTSP used in the simulations were randomly generated and divided into three types depending on the density of the targets. Each instance was created by drawing points, randomly and uniformly, from a disk of radius $r \in \{1, 5, 25\}$ and is said to be *dense*, *intermediate*, or *sparse* when $r = 1$, 5, or 25, respectively. For each value of r, the number of targets n was varied from 5 to 30 in steps of 5, and for each $n \in \{5, 10, 15, 20, 25, 30\}$, 30 instances were generated. The results are presented as averages of ratios of tour lengths: each ratio equals the length of the tour returned by a given algorithm when applied to an instance of the DTSP over the length of the tour returned by the DDTSP′ for the same instance, and the average is taken over the 30 instances generated for each pair of values for r and n. For the DLAA, each instance was solved for discretization levels $h = 32, 64$ (which were found in [21] to lead to computationally tractable DDTSP′s), and for $L = 5, 6, 7$. In all cases, the minimum turn-radius ρ is equal to one.

Remark: The SVA is premised on the assumption that the Euclidean distance between any two targets is greater than 2ρ, a condition not satisfied by the dense instances of the DTSP. Nevertheless, the algorithm is still applicable.



(a)



(b)

Fig. 3. Comparison of algorithms (a) DLAA, AA, SVA, LAA and 2-opt, and (b) BTA, for sparse instances of the DTSP. The plots show the average values of tour lengths, normalized with respect to the output of the DDTSP' with $h = 32$, as the functions of the number of targets. Each average is taken over 30 scenarios.

A. Sparse Instances

Fig. 3 shows the results from applying algorithms 1)–5) listed at the beginning of this section to the sparse instances of the DTSP. The results for the BTA are plotted separately in Fig. 3(b) because of the difference in scale. Because the locations of the targets are chosen randomly and uniformly within a circle of radius $r = 25$ and the minimum turn-radius is $\rho = 1$, there is no substantial difference between the ETSP and the DTSP tours for most instances. In other words, when the targets are sufficiently far apart, the kinematic constraint (1) becomes less relevant. Apart from DDTSP′, the DLAA takes the longest to run compared with the other algorithms, but the running time remains reasonable for offline optimization procedures (Fig. 4).

B. Intermediate Instances

Fig. 5 shows the results from applying all algorithms to the intermediate instances of the DTSP. Similar to Fig. 3, Fig. 5 compares the average normalized (with respect to the DDTSP′) length of the tours constructed by each algorithm as a function of the number of targets. The ratios are higher than for sparse environments, indicating deteriorating performance. Performance also

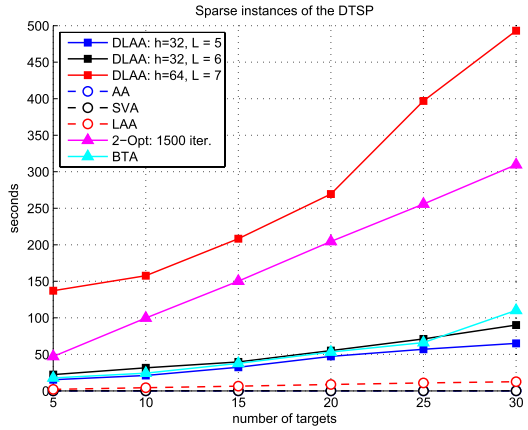


Fig. 4. Running times for sparse instances.

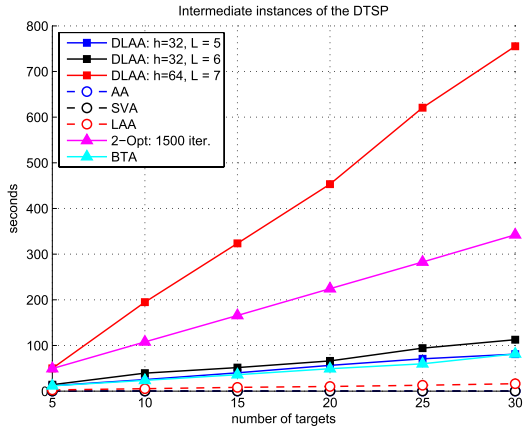


Fig. 6. Running times for intermediate instances.

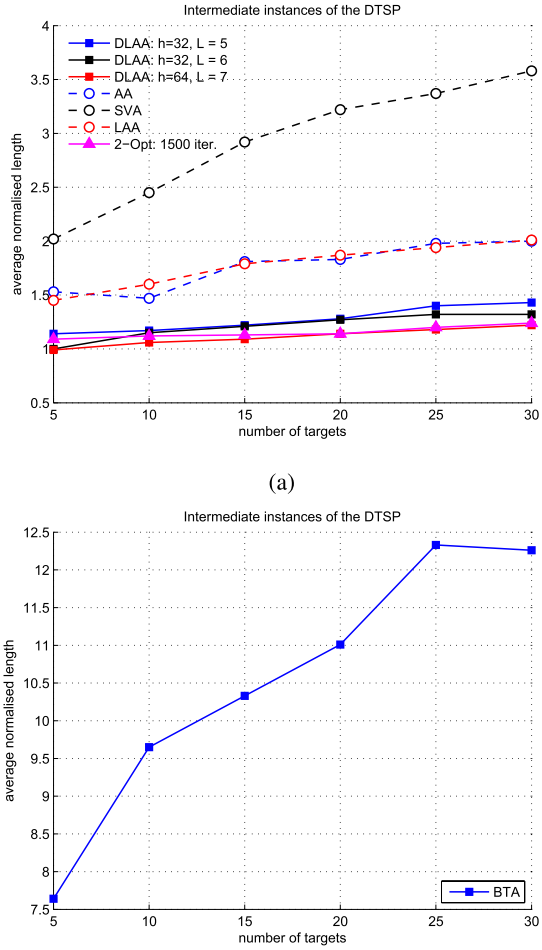


Fig. 5. Comparison of algorithms (a) DLAA, AA, SVA, LAA and 2-opt, and (b) BTA, for intermediate instances of the DTSP. The plot shows the average values of tour lengths normalized with respect to a tour length of the DDTSP' with $h = 32$ as the functions of the number of targets. Each average is taken over 30 scenarios.

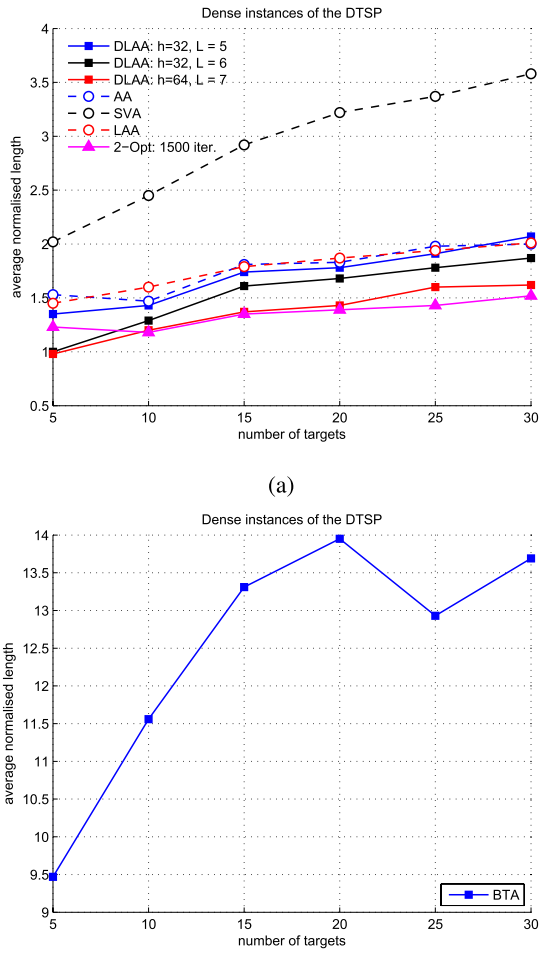


Fig. 7. Comparison of algorithms (a) DLAA, AA, SVA, LAA and 2-opt, and (b) BTA, for dense instances of the DTSP. The plot shows the average values of tour lengths normalized with respect to a tour length of the DDTSP' with $h = 32$ as the functions of the number of targets. Each average is taken over 30 scenarios.

deteriorates as n increases. Nevertheless, the DLAA with $h = 64$ and $L = 7$ performs best and for all values of n. Fig. 6 shows the running times as a function of the number of targets. While the running time of the DLAA is the longest, it is reasonable for offline computation.

C. Dense Instances

Fig. 7 shows the results from applying all algorithms to the dense instances of the DTSP. The ratios in Fig. 7(a) are higher compared with those in Fig. 5(a) for intermediate instances, indicating further deterioration of performance. Performance also deteriorates

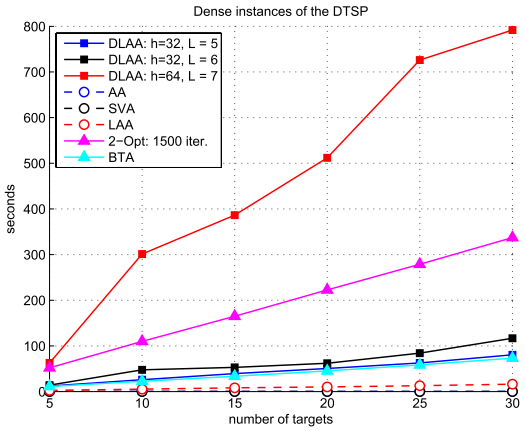


Fig. 8. Running times for dense instances.

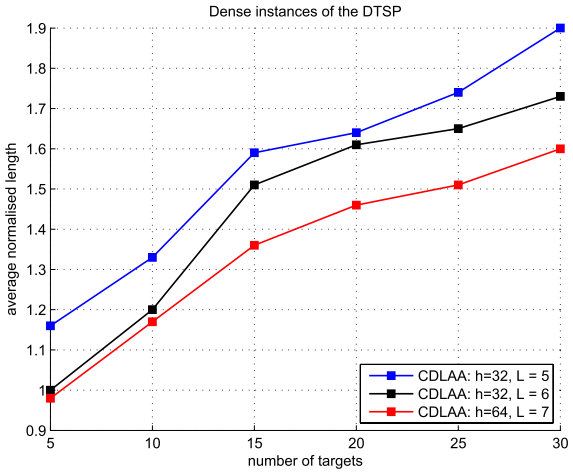


Fig. 9. Comparison of the CDLAA algorithm with different parameters for dense instances of the DTSP. The plot shows the average values of tour lengths normalized with respect to a tour length of the DDTSP′ with $h = 32$ as the functions of the number of targets. Each average is taken over 30 scenarios.

as n increases. It is interesting to note that the relative performance of the AA and LAA with respect to the other algorithms improves with n; for $n = 30$, the AA and LAA path lengths are similar to those of DLAA with $h = 32$ and $L = 5$. Increasing the DLAA look-ahead parameter to 6 leads to DLAA path lengths that are shorter than those generated by AA and LAA by 9.27% and 9.65%, respectively, for $n = 30$. The DLAA approximates an optimal solution to the DTSP as $h \to \infty$ and $L \to n$; thus, it is true, in general, that the performance of the DLAA improves as the values of h and L increase. For $n \geq 10$, the 2-Opt outperforms the other algorithms. The running time of the DLAA, although the longest, remains reasonable also for dense environments (Fig. 8).

Having compared the output and the running times of all the algorithms on the same instances of the DTSP, we cannot conclude unequivocally that one algorithm performs best for all possible application areas. The main advantage of the DLAA is that it is an easily tunable algorithm whose performance can be controlled according to the available computational budget. For example, if a DTSP tour returned by the DLAA is not sufficiently short, the algorithm can be rerun with a higher look-ahead horizon or a higher discretization level (or both). Three of the algorithms included in the comparison (AA, SVA, and LAA) do not leave any room for such improvement. At the same time, if the reduction in the tour length offered by the DLAA is marginal, and at a substantial computational cost, one of the faster algorithms is a better choice for the application in hand. The previous comments notwithstanding the numerical comparison in this and the previous sections are not definitive for the potential that each algorithm has. Specifically, the implementations that were used in the simulations do not incorporate any algorithmic of hardware optimizations that could improve their efficiency; rather, they are the proof-of-concept implementations. The algorithms that are compared vary in nature and their different characteristics offer different opportunities for optimization, such as parallelization (multithreading) and the use of suitable libraries. For example, the DLAA involves the solution of MILPs, the 2-Opt is a randomized algorithm, and the AA, the SVA, and the LAA require the solution of an ETSP and the calculation of optimal Dubins paths.

D. Time Complexity of DLAA

The simulations have shown that, at least for instances of the DTSP with up to 30 targets, the DLAA can be tuned to outperform other algorithms by choosing suitable values for the discretization level h and the look-ahead horizon L. Better performance comes at the cost of longer, but reasonable execution time. Although the largest instances used in the simulations consisted of 30 targets, such scenarios are by no means the largest that can be handled by the DLAA. In fact, if a solution to the corresponding ETSP is approximated by a polynomial time algorithm, e.g., by Christofides's algorithm [24], then the overall computational complexity of the DLAA is polynomial with respect to n and with constants that depend on h and L. To the cost of solving (approximately) the ETSP, a cost linear in n has to be added for the remaining steps of the DLAA, as explained now. Since at each iteration of the DLAA, an open Dubins path through L targets is constructed, the time complexity of one iteration is $O(h^L(L-1)!)$, which is a constant for fixed L and h. Therefore, the time complexity of Steps 2)–4) of the DLAA is $O((n/L)\cdot h^L(L-1)!)$, which is linear in n.

V. Clustered Discretized Look-Ahead Algorithm for Dense Instances of the DTSP

In this section, an algorithm for dense instances of the DTSP is presented that performs better than the DLAA with similar parameters. The algorithm, which is called cluster-based DDTSP LAA (CDLAA), relies on hierarchical agglomerative clustering with single linkage [25] to form clusters of targets. Given a set T of targets and a look-ahead parameter L, the targets are clustered into K clusters, where $K = \lceil((|T|)/L)\rceil$.[1] The geometric center of each cluster is computed and an ETSP path is constructed through all centers. This path is used to order the clusters, starting from the first cluster that includes the first (predefined) target. The order in which the targets are visited is determined by randomly choosing a target from each consecutive cluster according to the ETSP order of clusters until all the targets are numbered (the steps of the algorithm are described in detail in the following).

At each CDLAA iteration, an open Dubins path is constructed, where the first target is the second to last in the previous iteration of the algorithm. The last target, i.e., the Lth target, is fixed in the initial ordering of the targets. Consequently, at each iteration, the start and end targets are fixed, but otherwise, the initial order may change.

In the last iteration, the last target is what was originally the first one, so the final path is closed. The CDLAA consists of the following steps.

[1]If A is a set, $|A|$ denotes its cardinality. Given real number x, $\lceil x\rceil$ denotes the smallest integer greater than x.

Step 1) Implement hierarchical agglomerative clustering with single linkage to divide a target set T into $K = \lceil((|T|)/L)\rceil$ clusters.

Step 2) Set $(k_1, \ldots, k_K) :=$ cluster ordering (according to the ETSP path through the clusters' centers). Note that k_1 corresponds to T_1.

Step 3) Randomly choose a target T_i $(i = 2, \ldots n)$, each time from a different cluster k_j $(j = 1, \ldots K)$, starting with the first cluster and cycling through all the clusters until all targets are numbered.

Step 4) Set $(T_1, \ldots, T_n) :=$ target ordering.

Step 5) Choose $3 \leq L \leq n$, $h \in H$, and ρ.

Step 6) For iteration $m = 1, \ldots, M$, the following holds.

a) If $m = 1$, calculate an open DDTSP path for the first L targets from T, i.e., for the sequence $(T_1, \ldots, T_L)$, to obtain $\{(T_i, \theta_i)\}_{i\in\{1,\ldots,L\}}$, where $\theta_i \in H$.

b) If $2 \leq m \leq M-1$, calculate an open DDTSP path for the last two targets of iteration $m-1$ and for the other $L-2$ targets from T, i.e., for $(T_{1+(m-1)\cdot(L-2)}, \ldots, T_{m\cdot(L-2)+2})$, to obtain $\{(T_i, \theta_i)\}_{i\in\{1+(m-1)\cdot(L-2),\ldots,m\cdot(L-2)+2\}}$, where $\theta_i \in H$. At each iteration m, the configuration (position and heading) of the first target $T_{1+(m-1)\cdot(L-2)}$ is calculated in the previous $m-1$ iteration and the last target $T_{m\cdot(L-2)+2}$ is determined according to the ordering of the targets obtained in Step 4).

c) If $m = M$ such that $m \cdot (L-2) + 1 \geq n$, set $(T_{n+1}, \theta_{n+1}) := (T_1, \theta_1)$ and calculate an open DDTSP path for the last two targets of iteration $M-1$ and for the remaining targets from T, i.e., $(T_{1+(M-1)\cdot(L-2)}, \ldots, T_{n+1})$, to obtain $\{(T_i, \theta_i)\}_{i\in\{1+(M-1)\cdot(L-2),\ldots,n+1\}}$, where $\theta_i \in H$ and $T_{n+1} = T_1$.

Step 7) Return the sequence of configurations $\{(T_i, \theta_i)\}_{i\in\{1,\ldots,n\}}$.

VI. Comparison of the CDLAA With the Other Algorithms

Fig. 9 shows the results of applying the CDLAA to the same dense instances of the DTSP that were used in the simulations of Section IV. The output of the CDLAA was again normalized with respect to the DDTSP′, and each instance was solved for $h = 32$ and $L = 5, 6$ and $h = 64$ and $L = 7$. The CDLAA shows trends similar to those of the DLAA for dense environments, but the former achieved shorter tour lengths as the number of targets increased. For these larger instances of the DTSP, the average advantage of CDLAA over DLAA was between 6% and 31%.

The 2-Opt exhibits similar performance to the CDLAA with $h = 64$ and $L = 7$. Further tuning of the CDLAA enables it to outperform the 2-Opt, for the specific choice of parameters of the 2-Opt used in the simulations (1500 iterations and a two-step look-ahead horizon). For example, for $n = 30$, the CDLAA with $h = 32$ and $L = 14$ achieves tour lengths which are shorter, on average, by 12% than those of the 2-Opt. The CDLAA's running times, shown in Fig. 10, are of the same order of magnitude as those of the DLAA and are reasonable for offline optimization procedures.

A comparison between CDLAA and DLAA is shown in Fig. 11.

VII. Implementation

The algorithms presented in this paper have been implemented on a Pioneer D3-PX mobile robot whose motion resembles that of a Dubins vehicle. This is a differential drive robot with two main wheels, one for each motor, and a caster wheel. It is connected to a controller via a Lantronix WiBox that uses a wireless Ethernet-based serial bridge, since all processing and computation are done on a dedicated computer. The robot is controlled through the ARIA library suit. Motion planning algorithms, coded in MATLAB, run on a remote computer, and control signals are sent to the robot through the wireless link. Each of the main wheels is equipped with a high-resolution optical quadrature shaft encoder that counts the number of ticks per revolution. Based on the differences in consecutive counts, it is possible to calculate the distance traveled by each wheel. The robot averages the data from the two wheels.

A video demonstration in which a Dubins path is calculated for a scenario with six targets can be found `here`. The video is a superposition of the planned path, as shown on the computer screen (the solid blue line), and the actual movement of the robot through the targets. The (x, y) positions of the targets (in meters) are $(0, 0)$, $(0.2, 0.3)$, $(0.25, 0.75)$, $(0.5, 0.5)$, $(0.8, 0.2)$, and $(0.86, 1.14)$. The robot's minimum turning radius is set to 0.5 m and the discretization level is 32.

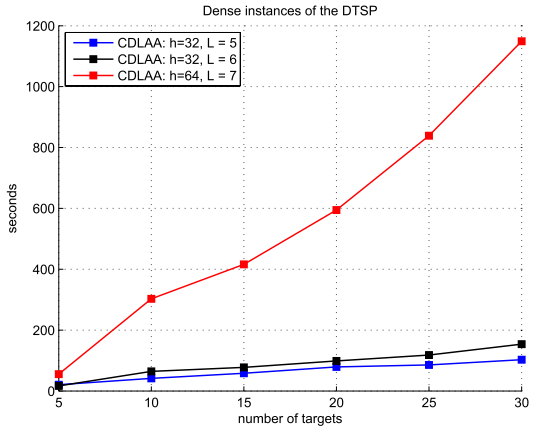


Fig. 10. Running times for the CDLAA.

length comparison for 30 targets					
sparse		intermediate		dense	
h=32, L=5	h=32, L=6	h=32, L=5	h=32, L=6	h=32, L=5	h=32, L=6
63%	59%	38%	37%	-6%	-6%
running time comparison for 30 targets					
sparse		intermediate		dense	
h=32, L=5	h=32, L=6	h=32, L=5	h=32, L=6	h=32, L=5	h=32, L=6
29%	28%	3%	5%	22%	24%

Fig. 11. Comparison between tour lengths and running times for CDLAA and DLAA, based on the results from the instances with the maximum number (30) of targets. The percentages shown are the ratios $(L_{\text{CDLAA}} - L_{\text{DLAA}})/L_{\text{CDLAA}}$ for the tour lengths and $(T_{\text{CDLAA}} - T_{\text{DLAA}})/T_{\text{CDLAA}}$ for the running times (L stands for look-ahead horizon).

VIII. Conclusion

Solving the DTSP to optimality for instances of realistic size is a computationally intractable problem, and this paper explores and demonstrates the efficacy of combining two natural approaches for obtaining satisfactory admissible tours: discretization and look-ahead policies. Although the resulting algorithms are of general applicability (i.e., no assumptions are made on the spatial distribution of the cities/targets), their usefulness becomes apparent when they are applied to instances of the DTSP, in which the distances between the targets are comparable with the minimum turn-radius of the Dubins vehicle. Effectively solving such instances requires that the kinematic constraints be integrated with the combinatorial aspect of the problem when designing an algorithm. To choose among various approaches

for performing such an integration, we have taken into account practical considerations, such as flexibility, ease of implementation, and availability of solvers, and as a result, the algorithms presented herein rely on the solution of tractable MILPs and can take the look-ahead horizon and the discretization level as input parameters. This flexibility allows for the calibration of the algorithms and the control of the tradeoff between computational cost and quality of solutions. This is in contrast with algorithms that use optimal Dubins paths as building blocks to construct DTSP tours, because increasing the look-ahead horizon for such algorithms leads quickly to impractical computational tasks. The idea of discretization and the use of a look-ahead policy are incorporated in the discretization-based look-ahead algorithm (DLAA). By adding a heuristic that preorders the targets using a clustering technique, the cluster-based DLAA (CDLAA) is obtained that performs even better than the DLAA.

A numerical comparison with algorithms from the literature shows the competitive performance of the algorithms. The execution times of the new algorithms are longer than other alternatives, but still reasonable for offline computation. For fixed discretization level and look-ahead horizon, the time complexity of the algorithms can be made polynomial with respect to the number of targets, if the initial order of the targets is determined by an approximation algorithm for the corresponding ETSP. As a result, the algorithms can be used to find feasible tours through large numbers of targets; the length of the resulting tours depends on the computational budget available.

REFERENCES

[1] V. Bertram, *Unmanned Surface Vehicles—A Survey*. Copenhagen, Denmark: Skibsteknisk Selskab, 2008.

[2] M. Caccia, M. Bibuli, R. Bono, and G. Bruzzone, "Basic navigation, guidance and control of an unmanned surface vehicle," *Auto. Robots*, vol. 25, no. 4, pp. 349–365, 2008.

[3] J. Ouellette. "Science goes exploring under the sea," *Ind. Phys.*, vol. 8, no. 4, pp. 20–23, 2002.

[4] R. Isaacs, *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization* (Dover Books on Mathematics). New York, NY, USA: Dover, 1999.

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

[6] J.-D. Boissonnat, A. Cérézo, and J. Leblond, "Shortest paths of bounded curvature in the plane," INRIA Sophia Antipolis, Rapport de Recherche, Tech. Rep. 1503, 1991.

[7] H. J. Sussmann and G. Tang, "Shortest paths for the Reeds–Shepp car: A worked out example of the use of geometric techniques in nonlinear optimal control," Rutgers Center Syst. Control, Tech. Rep. SYCON-91-10, 1991.

[8] D. L. Applegate *et al.*, *The Traveling Salesman Problem—A Computational Study*. Princeton, NJ, USA: Princeton Univ., 2006.

[9] M. S. Cons, T. Shima, and C. Domshlak, "Integrating task and motion planning for unmanned aerial vehicles," *Unmanned Syst.*, vol. 2, no. 1, pp. 19–38, 2014.

[10] E. Edison and T. Shima, "Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms," *Comput. Oper. Res.*, vol. 38, no. 1, pp. 340–356, 2011.

[11] T. G. McGee and J. K. Hedrick, "Path planning and control for multiple point surveillance by an unmanned aircraft in wind," in *Proc. Amer. Control Conf.*, Jun. 2006, pp. 4261–4266.

[12] S. Rathinam, R. Sengupta, and S. Darbha, "A resource allocation algorithm for multivehicle systems with nonholonomic constraints," *IEEE Trans. Autom. Sci. Eng.*, vol. 4, no. 1, pp. 98–104, Jan. 2007.

[13] K. Savla, E. Frazzoli, and F. Bullo, "Traveling salesperson problems for the Dubins vehicle," *IEEE Trans. Autom. Control*, vol. 53, no. 6, pp. 1378–1391, Jul. 2008.

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

[15] G. Yang and V. Kapila, "Optimal path planning for unmanned air vehicles with kinematic and tactical constraints," in *Proc. 41st IEEE Conf. Decision Control*, Dec. 2002, pp. 1301–1306.

[16] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity* (Dover Books on Computer Science). New York, NY, USA: Dover, 1998. [Online]. Available: http://books.google.co.il/books?id=u1RmDoJqkF4C

[17] J. Le Ny, E. Feron, and E. Frazzoli, "On the Dubins traveling salesman problem," *IEEE Trans. Autom. Control*, vol. 57, no. 1, pp. 265–270, Jan. 2012.

[18] X. Ma and D. A. Castañón, "Receding horizon planning for Dubins traveling salesman problems," in *Proc. 45th IEEE Conf. Decision Control*, Dec. 2006, pp. 5453–5458.

[19] J. D. Boissonnat and X. N. Bui, *Accessibility Region for a Car That Only Moves Forwards Along Optimal Paths*. France: INRIA, 1994.

[20] P. Isaiah and T. Shima, "Motion planning algorithms for the Dubins travelling salesperson problem," *Automatica*, vol. 53, pp. 247–255, Mar. 2015.

[21] I. Cohen, C. Epstein, and T. Shima, "On the discretized dubins traveling salesman problem," *IIE Trans.*, accepted for publication.

[22] M. Desrochers and G. Laporte, "Improvements and extensions to the Miller–Tucker–Zemlin subtour elimination constraints," *Oper. Res. Lett.*, vol. 10, no. 1, pp. 27–36, 1991.

[23] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *J. ACM*, vol. 7, no. 4, pp. 326–329, 1960.

[24] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," U.S. Dept. Commerce, Carnegie Mellon Univ., Pittsburgh, PA, USA, Manage. Sci. Res. Rep. 388, Feb. 1976.

[25] R. Sibson, "Slink: An optimally efficient algorithm for the single-link cluster method," *Comput. J.*, vol. 16, no. 1, pp. 30–34, 1973.