C*: A New Bounding Approach for the Moving-Target Traveling Salesman Problem

Allen George Philip[1], Zhongqiang Ren[2], Sivakumar Rathinam[1], and Howie Choset[3]

***Abstract*—We introduce a new bounding approach called Continuity* (C^*), which provides optimality guarantees for the Moving-Target Traveling Salesman Problem (MT-TSP). Our approach relaxes the continuity constraints on the agent's tour by partitioning the targets' trajectories into smaller segments. This allows the agent to arrive at any point within a segment and depart from any point in the same segment when visiting each target. This formulation enables us to pose the bounding problem as a Generalized Traveling Salesman Problem (GTSP) on a graph, where the cost of traveling along an edge requires solving a new problem called the Shortest Feasible Travel (SFT). We present various methods for computing bounds for the SFT problem, leading to several variants of C^*. We first prove that the proposed algorithms provide valid lower-bounds for the MT-TSP. Additionally, we provide computational results to validate the performance of all C^* variants on instances with up to 15 targets. For the special case where targets move along straight lines, we compare our C^* variants with a mixed-integer Second Order Conic Program (SOCP) based method, the current state-of-the-art solver for the MT-TSP. While the SOCP-based method performs well on instances with 5 and 10 targets, C^* outperforms it on instances with 15 targets. For the general case, on average, our approaches find feasible solutions within approximately 4.5% of the lower-bounds for the tested instances.**

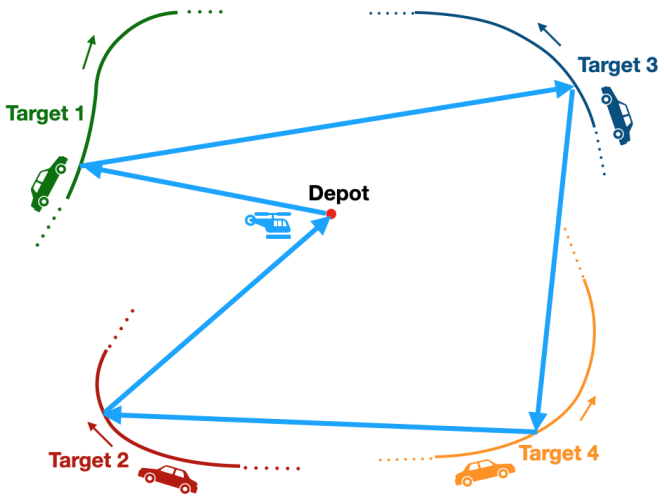


Fig. 1: A feasible solution for an instance of the MT-TSP with four targets. The blue lines shows the path of the vehicle. Also, the colored solid segments for each target indicates the part of its trajectory corresponding to its time-windows when the vehicle can visit the target.

I. Introduction

The Traveling Salesman Problem (TSP) is one of the most important problems in optimization with several applications including unmanned vehicle planning [1]–[4], transportation and delivery [5], monitoring and surveillance [6], [7], disaster management [8], precision agriculture [9], and search and rescue [10], [11]. Given a set of target locations (or targets) and the cost of traveling between any pair of targets, the TSP aims to find a shortest tour for a vehicle to visit each of the targets exactly once. In this paper, we consider a natural generalization of the TSP referred to as the Moving Target-TSP (MT-TSP) where the targets are mobile and traverse along known trajectories (refer to Fig. 1). Specifically, given a set of targets S, where each target $i \in S$ moves at a constant speed $v_i \geq 0$ along a known trajectory π_i, MT-TSP aims to find a trajectory for a vehicle that starts and ends at a depot and travels at a maximum speed v, such that it intercepts all the targets in the shortest possible time. This generalization is motivated by applications including monitoring and surveillance [12]–[15], missile defense [16]–[18], fishing [19], [20], human evacuation [21], and dynamic target tracking [22], where vehicles are required to visit or monitor a set of mobile targets. The focus of this paper is on the optimality guarantees for the MT-TSP.

Targets are typically assumed to move at a speed equal to or slower than the agent's speed since the agent is anticipated to visit or intercept each target [16]. When the speed of every target is equal to 0, the Moving-Target Traveling Salesman Problem (MT-TSP) simplifies to the standard Euclidean Traveling Salesman Problem (ETSP). Therefore, MT-TSP is a generalization of the ETSP and is NP-Hard. In addition, the MT-TSP considered in this paper includes time-window constraints for visiting the targets. These additional constraints make the problem even more challenging, as finding a feasible solution to the TSP with time-window constraints is NP-Complete, even for stationary targets [23]. Unlike the TSP which has been extensively studied, the current literature on MT-TSP is limited.

A. Literature Review

Exact and approximation algorithms are available in the literature for some special cases of the MT-TSP where the targets are assumed to move along straight lines with constant speeds on a 2D plane. In [24], Chalasani and Motvani propose a $(5(1+v)/3(1-v))$-approximation algorithm for the case

1. Allen George Philip and Sivakumar Rathinam are with Texas A&M University, College Station, TX 77843, USA.
2. Zhongqiang Ren is with Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai, China.
3. Howie Choset is with Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA.

where all the targets move in the same direction with the same speed v. Hammar and Nilsson [25], also present a $(1+(\epsilon/(1-v))$-approximation algorithm for the same case. If the targets can move at different speeds in arbitrary directions on a 2D plane, they also show that the MT-TSP cannot be approximated better than a factor of $2^{\Omega(\sqrt{n})}$ by a polynomial time algorithm (unless $P = NP$), where n is the number of targets. In [16], Helvig et al. develop an exact $O(n)$-time algorithm for the case when the moving targets are restricted to a single line (SL-MT-TSP). Also, for the case where most of the targets are static and at most $O(log(n)/log(log(n)))$ targets are moving out of the n targets, they propose a $(2+\epsilon)$-approximation algorithm. An exact algorithm is also presented [16] for the MT-TSP with resupply where the agent must return to the depot after visiting each target; here, the targets are assumed to be far away from the depot or slow, and move along lines through the depot, towards or away from it. In [17], Stieber, and Fügenschuh formulate the MT-TSP as a mixed-integer Second Order Conic Program (SOCP) by relying on the key assumption that the targets travel along straight lines. Also, multiple agents are allowed, agents are not required to return to the depot, and each target has to be visited exactly once within its visibility window by one of the agents. Optimal solutions to the MT-TSP are then found for this special case. Recently, we have also developed a new formulation [26] for a special case when the targets travel along straights lines and the objective is to minimize the distance traveled by the agent; this paper on the other hand deals with minimizing travel time for a generalization of the MT-TSP with multiple time-windows.

Apart from the above methods that provide optimality guarantees to some special cases of the MT-TSP, feasible solutions can be obtained using heuristics as shown in [12]–[14], [18], [19], [21], [22], [27]–[30]. However, these approaches do not show how far the feasible solutions are from the optimum.

A few variants of the MT-TSP and related problems have also been addressed in the literature. In [31], Hassoun et al. suggest a dynamic programming algorithm to find an optimal solution to a variant of the SL-MT-TSP where the targets move at the same speeds and may appear at different times. Masooki and Kallio in [15] address a bi-criteria variant of the MT-TSP where the number of targets vary with time and their motion is approximated using (discontinuous) step functions of time.

B. Our work and contributions

In this article, we consider a variant of the MT-TSP where each target moves along piecewise-linear segments or Dubins [32] curves. Each target may also be associated with time-windows during which the vehicle must visit the target. When targets move along generic trajectories, such as those considered in this paper, no algorithms currently exist for finding the optimum to the MT-TSP. When optimal solutions are difficult to obtain, one can develop algorithms that can find feasible solutions which provide upper bounds, and lower-bounding algorithms that provide tight underestimates to the optimum. Optimality guarantees can then be obtained by comparing the upper and lower-bounds.

One way to generate feasible solutions to this problem is to sample a discrete set of times from the (planning) time horizon, and then consider the corresponding set of locations for each target; given a pair of targets and their sampled times, one can readily check for feasibility of travel and compute the travel costs between the targets. A solution can then be obtained for the MT-TSP by posing it as a *Generalized* TSP (GTSP) [33] where the objective is to find an optimal TSP tour that visits exactly one (sampled) location for each target. While this approach can produce feasible solutions and upper-bounds, it may not find the optimum or lower-bounds for the MT-TSP. Therefore, we seek to develop methods that primarily focus on finding tight lower-bounds in this paper.

In the special case where each target travels along a straight line, the SOCP-based formulation in [17] can be used to find the optimum. For the general case where each target travels along a trajectory made of piecewise-linear segments or Dubins curves, currently, we do not know of any method in the literature that can find the optimum or provide tight lower-bounds to the optimum for the MT-TSP. In this article, we develop a new approach called Continuity* (C^*) to answer this question.

C^* relies on the following key ideas. First, we relax the continuity of the trajectory of the agent and allow it to be discontinuous whenever it reaches the trajectory of a target. We do this by partitioning the trajectory of each target into smaller intervals [1] and allow the agent to arrive at any point in an interval and depart from any point from the same interval. We then construct a graph $\mathcal{G}$ where all the nodes (intervals) corresponding to each target are grouped into a cluster, and any two nodes belonging to distinct clusters is connected by an edge. Next, the cost of traveling any edge is obtained by solving a Shortest Feasible Travel (SFT) problem between two intervals corresponding to distinct targets. Specifically, given two distinct targets i and j, and parts of their trajectories, $\bar{\pi}_i$ and $\bar{\pi}_j$, corresponding to two intervals, the SFT problem aims to find a time t_i to depart from $\bar{\pi}_i$ and feasibly reach $\bar{\pi}_j$ at time t_j such that $t_j - t_i$ is minimized.

Once all the travel costs are computed, we formulate a Generalized TSP (GTSP) [33] in $\mathcal{G}$ which aims to find a tour such that exactly one node from each cluster is visited by the tour and the sum of the costs of the edges in the tour is minimum. We then show that our approach provides lower-bounds for the MT-TSP. As the number of partitions or discretizations of each target's trajectory increases, the lower-bounds get better and converge to the optimum.

In addition to solving the SFT problem to optimality for targets moving along piecewise-linear segments, we also provide three simple and fast methods for computing bounds to the cost of an edge in $\mathcal{G}$ for more generic target trajectories. In this way, we develop several variants of C^*. We also show how feasible solutions can be constructed from the lower-bounds, though this may not be always possible[2] if challenging time-

[1] Since each target travels at a constant speed, distance traveled along the trajectory has one to one correspondence to the time elapsed.

[2] If tight time-window constraints are present, finding feasible solutions is difficult in any case, whether we use C^* or not.

window constraints are present.

We provide extensive computational results to corroborate the performance of the variants of C^* for instances with up to 15 targets. For the special case where targets travel along lines, we compare our C^* variants with the SOCP-based method, which is the current state-of-the-art solver for MT-TSP. While C^* provides similar bounds compared to the SOCP-based method for all cases, C^* is an order of magnitude faster relative to the SOCP-based method for instances with 15 targets. For the general case, on average, our approaches find feasible solutions within $\approx$ 4.5% of the lower-bounds for the tested instances.

II. Problem Definition

Let $S := \{s_1, s_2, \cdots, s_n\}$ be the set of targets. All the targets and the agent move in a 2D Euclidean plane. Each target $i \in S$ moves at a constant speed $v_i \geq 0$, and follows a *continuous trajectory* π_i. In this paper, π_i is piecewise-linear (made of a finite set of line-segments) or a Dubins curve (made of circular arcs of a given turning radius and line-segments). Consider an agent that moves at a speed no greater than v_{max} at any time instant. There are no other dynamic constraints placed on the motion of the agent. The agent starts and ends its path at a location referred to as the depot. We assume $v_{max} > v_i$ for all $i \in S$. Also, any target $i \in S$ is associated with a set of k time-windows $[\underline{t}_{i,1}, \bar{t}_{i,1}], \cdots, [\underline{t}_{i,k}, \bar{t}_{i,k}]$ during which times the agent can visit the target. The objective of the MT-TSP is to find a tour for the agent such that

- the agent starts and ends its tour at the depot d,
- the agent visits each target $i \in S$ exactly once within one of its specified time-windows $[\underline{t}_{i,1}, \bar{t}_{i,1}], \cdots, [\underline{t}_{i,k}, \bar{t}_{i,k}]$, and
- the travel time of the agent is minimized.

III. Notations and Definitions

A **trajectory-point** for a target $i \in S$ is denoted by $\pi_i(t)$ and represents the position occupied by target i at time t. A **trajectory-interval** for a target i is denoted by $\pi_i[\underline{t}_i, \bar{t}_i]$, and refers to the set of all the positions occupied by i over the time interval $[\underline{t}_i, \bar{t}_i]$, where $\underline{t}_i \leq \bar{t}_i$. In the special case when $\underline{t}_i = \bar{t}_i$, $\pi_i[\underline{t}_i, \bar{t}_i]$ reduces to a trajectory-point and can be written as just $\pi_i(\underline{t}_i)$. Suppose the time interval $[\underline{t}_i, \bar{t}_i]$ lies within another time interval $[\underline{t}_{i,1}, \bar{t}_{i,1}]$ (*i.e.* $[\underline{t}_i, \bar{t}_i] \subseteq [\underline{t}_{i,1}, \bar{t}_{i,1}]$). Then, $\pi_i[\underline{t}_i, \bar{t}_i]$ is said to be a **trajectory-sub-interval** that lies within the trajectory-interval $\pi_i[\underline{t}_{i,1}, \bar{t}_{i,1}]$.

A **travel** from $\pi_i(t)$ to $\pi_j(t')$ denotes the event where the agent departs from $\pi_i(t)$ at time t, and arrives at $\pi_j(t')$ at time t'. This is the same as saying the agent departs from target i at time t and arrives at target j at time t'. For travel from the depot d to $\pi_i(t)$, the agent departs from d at time $t_d = 0$, and for travel from $\pi_i(t)$ to d, the agent arrives at d at some time $t_d \geq t$.

A **travel is feasible** if the agent can complete it without exceeding its maximum speed v_{max}. Clearly, feasible travel requires the arrival time to be greater than or equal to the departure time. We define a feasible travel exists from some trajectory-interval $\pi_i[\underline{t}_i, \bar{t}_i]$ to another trajectory-interval $\pi_j[\underline{t}_j, \bar{t}_j]$, if there exists some $t \in [\underline{t}_i, \bar{t}_i]$ and some $t' \in [\underline{t}_j, \bar{t}_j]$ such that the travel from $\pi_i(t)$ to $\pi_j(t')$ is feasible. We also define a feasible travel exists from a trajectory-point $\pi_i(t_i)$ to a target trajectory π_j, if there exists some $t \geq 0$ such that travel from $\pi_i(t_i)$ to $\pi_j(t)$ is feasible. Feasible travels from a trajectory to a trajectory-point or feasible travels to and from a depot can be defined similarly.

Now, we define the following optimization problems between any two targets i and j which we need to solve as part of our approach to the MT-TSP:

- **Shortest Feasible Travel (SFT) problem from the trajectory-interval $\pi_i[\underline{t}_i, \bar{t}_i]$ to the trajectory-interval $\pi_j[\underline{t}_j, \bar{t}_j]$:**
  - $\min_{t,t'} \ (t' - t)$ where $t \in [\underline{t}_i, \bar{t}_i]$, $t' \in [\underline{t}_j, \bar{t}_j]$, and travel from $\pi_i(t)$ to $\pi_j(t')$ is feasible.

- **Earliest Feasible Arrival Time (EFAT) problem from the trajectory-point $\pi_i(t_i)$ to the trajectory π_j:**
  - $\min t$ such that $t \geq 0$ and the travel from $\pi_i(t_i)$ to $\pi_j(t)$ is feasible.

  The optimal time (also referred to as EFAT) to the above problem is denoted as $\boldsymbol{\mathcal{E}(t_i)}$. In other words, after departing from $\pi_i(t_i)$, the agent can reach target j at the earliest at the trajectory-point $\pi_j(\mathcal{E}(t_i))$.

- **Latest Feasible Departure Time (LFDT) problem from a trajectory π_i to a trajectory-point $\pi_j(t_j)$:**
  - $\max t$ such that $t \geq 0$ and the travel from $\pi_i(t)$ to $\pi_j(t_j)$ is feasible.

  The optimal time (also referred to as LFDT) to the above problem is denoted as $\boldsymbol{\mathcal{L}(t_j)}$. In other words, if the agent needs to arrive at $\pi_j(t_j)$, the latest position it must depart from on π_i is $\pi_i(\mathcal{L}(t_j))$.

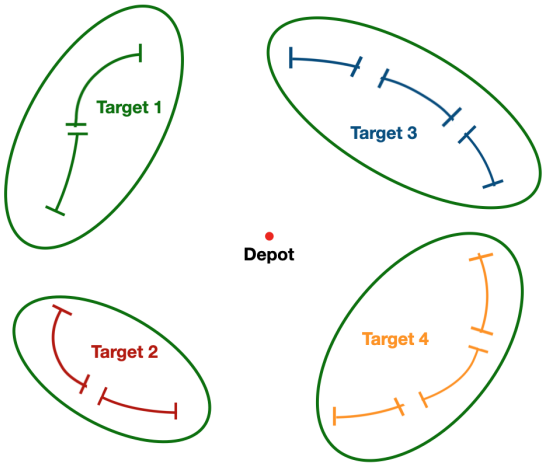
IV. C^* Algorithm

The following are the three key steps in the C^* Algorithm. These steps are also illustrated in Fig. 2.

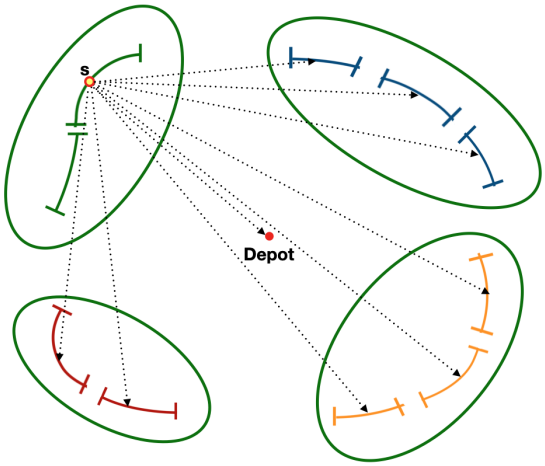
1) **Partition the time-windows corresponding to each target:** The time-windows corresponding to each target in S is first partitioned into smaller intervals[3] of a given size (say Δ). This partitioning step results in a cluster of trajectory-intervals corresponding to each target as shown in Fig. 2a. Henceforth, each trajectory-interval in any of these clusters is also referred to as a node. Also, the cluster of nodes corresponding to the target $i \in S$ is denoted as C_i. In the next step, we will construct a graph using these nodes and relax the continuity of the agent's path when it reaches any one of the nodes corresponding to a target.

2) **Construct a graph $\mathcal{G}$ with travel costs:** The graph $\mathcal{G}$ we construct is defined over the depot and the nodes in C_i, $i \in S$. Any two nodes p, q in $\mathcal{G}$ are

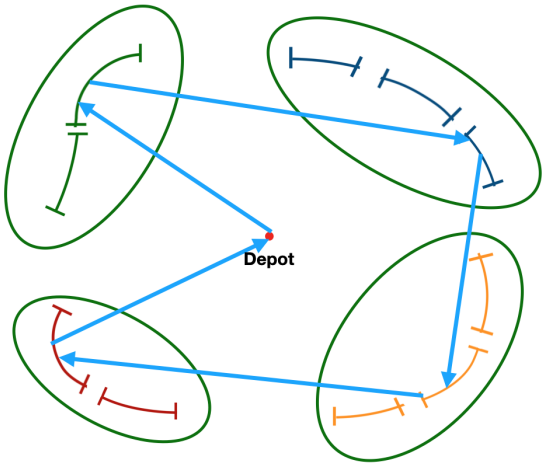
[3]Here, without loss of generality, we assume that each time-window corresponding to a target is an integer multiple of Δ.



(a) Partition the time-windows to create a cluster of trajectory-intervals for each target.



(b) Construct a graph built on the clusters of trajectory-intervals corresponding to the targets. In this figure, we only show the directed edges from trajectory-interval s to all the nodes outside the cluster containing s.



(c) Solve a GTSP on the graph to find an optimal solution to the GTSP. Note the discontinuities in the optimal solution when the agent's path reaches a trajectory-interval (a node) in each cluster.

Fig. 2: Illustration of the key steps in the C^* Algorithm.

connected by directed edges if they belong to distinct clusters. Also, the depot is connected to all the remaining nodes in $\mathcal{G}$ through directed edges. Formally, $\mathcal{G} := (V, E)$ where $V = \{d\} \cup \{p : p \in C_i, i \in S\}$ and $E := \{(p,q) : p \in C_i, q \in C_j, i,j \in S, i \neq j\} \cup \{(p,d),(d,p) : p \in V \setminus \{d\}\}$. Each edge (p,q) in $\mathcal{G}$ is associated with a non-negative, travel cost l_{pq} which is obtained by using any one of the algorithms presented in section V. Since, the aim of this paper is to generate tight bounds, given an edge (p,q), l_{pq} must be a lower-bound on the cost of the SFT from node p to node q.

3) **Solve the GTSP on $\mathcal{G}$ to find a bound**: The objective of the GTSP is to find a tour that starts and ends at the depot such that exactly one node is visited from each of the clusters corresponding to the targets and the sum of the travel costs is minimized. We will prove in Theorem 1 that the optimum to the GTSP provides a lower-bound on the optimal cost to the MT-TSP.

Theorem 1. *Suppose for any edge $(p,q) \in \mathcal{G}$, l_{pq} denotes a lower-bound on the cost of the SFT from p to q. Then, the optimal solution for the GTSP on $\mathcal{G}$ obtained in C^* provides a lower-bound on the optimum of the MT-TSP.*

Proof. Consider an optimal solution to the MT-TSP. Let the sequence of nodes in $\mathcal{G}$ visited by this solution be $S^* := (d, i_1, \cdots, i_n, d)$ and the corresponding arrival times be $(0, t_1, \cdots, t_n, t_{tour})$. In this solution, the agent travels from d to $\pi_{i_1}(t_1)$, then travels from $\pi_{i_k}(t_k)$ to $\pi_{i_{k+1}}(t_{k+1})$ for $k = 1, \cdots, n-1$ and finally travels from $\pi_{i_n}(t_n)$ to d at speed v_{max}. The time taken by the agent to complete the tour is $t_{tour} = t_1 + \sum_{k=1}^{n-1}(t_{k+1} - t_k) + (t_{tour} - t_n)$. For any pair of adjacent nodes (p,q) visited by the agent in this optimal solution, since l_{pq} is a lower-bound on the SFT cost from p to q, $l_{di_1} \leq t_1$, $l_{i_k i_{k+1}} \leq t_{k+1} - t_k$ for $k = 1, \cdots, n-1$ and $l_{i_n d} \leq t_{tour} - t_n$. Therefore, for the sequence of nodes S^* in the optimal solution, $l_{di_1} + \sum_{k=1}^{n-1} l_{i_k i_{k+1}} + l_{i_n d} \leq t_{tour}$. Since S^* is also a feasible solution to the GTSP, the optimal cost obtained by solving the GTSP must be at most equal to $l_{di_1} + \sum_{k=1}^{n-1} l_{i_k i_{k+1}} + l_{i_n d} \leq t_{tour}$. Hence proved. □

Remark 1. *As the number of partitions of the time-windows corresponding to the targets tend to infinity, the size of each time interval (Δ) tends to 0 and that of the discontinuities in the agent's path tend to disappear. Since for any partition of the time-windows, C^* provides a lower-bound, the optimal cost of the relaxed MT-TSP converges asymptotically to the optimal cost of the MT-TSP as $\Delta \to 0$.*

V. Algorithms for Computing Travel Costs

Given two trajectory-intervals (or nodes[4]) $p := \pi_i[\underline{t}_i, \bar{t}_i]$ and $q := \pi_j[\underline{t}_j, \bar{t}_j]$, this section presents four bounding algorithms to estimate the travel cost l_{pq} such that l_{pq} is at most equal to the optimal SFT cost from $\pi_i[\underline{t}_i, \bar{t}_i]$ to $\pi_j[\underline{t}_j, \bar{t}_j]$. Three of

[4]If a node is generated from a depot, the algorithms in section V can still be applied by treating the depot as a stationary target.

the algorithms can be applied to generic target trajectories while the fourth algorithm is specially optimized and tailored for piecewise-linear target trajectories. Before we present our algorithms, we introduce two conditions that, if satisfied, will address the trivial cases that do not require further optimization.

Theorem 2. *If the travel from $\pi_i(\underline{t}_i)$ to $\pi_j(\bar{t}_j)$ is not feasible, then the travel from $\pi_i[\underline{t}_i, \bar{t}_i]$ to $\pi_j[\underline{t}_j, \bar{t}_j]$ is not feasible, and vice-versa.*

Proof. We provide a proof by contraposition. If travel from $\pi_i(\underline{t}_i)$ to $\pi_j(\bar{t}_j)$ is feasible, then it readily follows that the travel from $\pi_i[\underline{t}_i, \bar{t}_i]$ to $\pi_j[\underline{t}_j, \bar{t}_j]$ is feasible. Now, let us show the other direction. If the travel from $\pi_i[\underline{t}_i, \bar{t}_i]$ to $\pi_j[\underline{t}_j, \bar{t}_j]$ is feasible, then there is a time $t_i^* \in [\underline{t}_i, \bar{t}_i]$ and a time $t_j^* \in [\underline{t}_j, \bar{t}_j]$ such that the travel from $\pi_i(t_i^*)$ to $\pi_j(t_j^*)$ is feasible. Let v^* denote the speed of the agent during this travel. Since the agent can match the speed of any target, it can travel along the trajectory of target i from $\pi_i(\underline{t}_i)$ to $\pi_i(t_i^*)$ at speed v_i, then travel from $\pi_i(t_i^*)$ to $\pi_j(t_j^*)$ at speed v^*, and finally travel along the trajectory of target j from $\pi_j(t_j^*)$ to $\pi_j(\bar{t}_j)$ at speed v_j. As a result, the travel from $\pi_i(\underline{t}_i)$ to $\pi_j(\bar{t}_j)$ is feasible. □

Theorem 3. *If travel from $\pi_i(\bar{t}_i)$ to $\pi_j(\underline{t}_j)$ is feasible, then $\underline{t}_j - \bar{t}_i$ is the optimal cost of SFT from $\pi_i[\underline{t}_i, \bar{t}_i]$ to $\pi_j[\underline{t}_j, \bar{t}_j]$.*

Proof. Note that the objective of SFT is to find a feasible solution such that $t_j - t_i$ is minimized subject to $t_j \in [\underline{t}_j, \bar{t}_j]$ and $t_i \in [\underline{t}_i, \bar{t}_i]$. Therefore, a trivial lower-bound on the optimal cost to this SFT problem is $\underline{t}_j - \bar{t}_i$. If travel from $\pi_i(\bar{t}_i)$ to $\pi_j(\underline{t}_j)$ is feasible, its travel cost will be equal to $\underline{t}_j - \bar{t}_i$ which matches the lower-bound. Hence, $\underline{t}_j - \bar{t}_i$ must be the optimal cost of the SFT from $\pi_i[\underline{t}_i, \bar{t}_i]$ to $\pi_j[\underline{t}_j, \bar{t}_j]$. □

Based on the above theorems, we first check if the travel from $\pi_i(\underline{t}_i)$ to $\pi_j(\bar{t}_j)$ is feasible. If this condition is not satisfied, then from Theorem 2, travel from $\pi_i[\underline{t}_i, \bar{t}_i]$ to $\pi_j[\underline{t}_j, \bar{t}_j]$ is also not feasible. In this case, each of the algorithms return a very large value for l_{pq} and no further computations are necessary. Next, if travel from $\pi_i(\bar{t}_i)$ to $\pi_j(\underline{t}_j)$ is feasible, each algorithm sets l_{pq} to be equal to $\underline{t}_j - \bar{t}_i$ and no further optimization is required (from Theorem 3). Therefore, we assume henceforth that travel from $\pi_i(\underline{t}_i)$ to $\pi_j(\bar{t}_j)$ is feasible, while travel from $\pi_i(\bar{t}_i)$ to $\pi_j(\underline{t}_j)$ is not feasible. Based on this assumption, we present our bounding algorithms. We obtain a variant of *C** based on the choice of the bounding algorithm used; consequently, each of the following subsections is named accordingly to match the corresponding *C** variant.

A. *C**-*Lite*

We derive a simple bound based solely on the timing constraints. Regardless of the feasibility of travel from $\pi_i(\bar{t}_i)$ to $\pi_j(\underline{t}_j)$, as discussed in the proof of Theorem 3, $\underline{t}_j - \bar{t}_i$ serves as a valid lower-bound for the SFT cost. Additionally, we know that the optimal SFT cost is always non-negative. Hence, in this variant of *C**, we set $l_{pq} := \max\{\underline{t}_j - \bar{t}_i, 0\}$.

B. *C**-*Geometric*

In this variant of *C**, we ignore the motion constraints of the target trajectories and only consider the shortest Euclidean distance between the points traveled in $\pi_i[\underline{t}_i, \bar{t}_i]$ and $\pi_j[\underline{t}_j, \bar{t}_j]$. Specifically, let all the points traveled in $\pi_i[\underline{t}_i, \bar{t}_i]$ and $\pi_j[\underline{t}_j, \bar{t}_j]$ be denoted as S_1 and S_2 respectively. Let the shortest Euclidean distance between the sets S_1 and S_2 be defined as $dist(S_1, S_2) := \min_{x_1 \in S_1, x_2 \in S_2} \|x_1 - x_2\|_2$. Here, we set $l_{pq} := \frac{dist(S_1, S_2)}{v_{max}}$ which is clearly a lower-bound on the SFT cost.

For the target trajectories considered in this paper, each segment of a trajectory is either a straight line or an arc with a given turning radius for the target. If $\pi_i[\underline{t}_i, \bar{t}_i]$ and $\pi_j[\underline{t}_j, \bar{t}_j]$ correspond to k_i and k_j segments, respectively, we compute $dist(S_1, S_2)$ by finding the shortest distance between any segment in S_1 and any segment in S_2, and then computing the minimum of all these optima. This requires $O(k_i k_j)$ computations.

C. *C**-*Sampling*

This variant of *C** first partitions $[\underline{t}_i, \bar{t}_i]$ into k uniform sub-intervals, where k is a sampling parameter. It then estimates the optimal SFT cost for departing from each sub-interval and selects the minimum. Formally, let the p^{th} sub-interval of $[\underline{t}_i, \bar{t}_i]$ be $[\underline{t}_{i,p}, \bar{t}_{i,p}]$. The optimal SFT cost is then given by $\min_{p=1}^{k} \min_{t \in [\underline{t}_{i,p}, \bar{t}_{i,p}]} (\mathcal{E}(t) - t)$.

Since $\mathcal{E}(t)$ monotonically increases with t (later proved in Theorem 4), the optimal SFT cost is at least equal to $l_{pq} := \min_{p=1}^{k} (\mathcal{E}(\underline{t}_{i,p}) - \bar{t}_{i,p})$. If travel is infeasible from $\pi_i(\underline{t}_{i,p})$ to $\pi_j(\bar{t}_j)$, then travel from $\pi_i(t)$ for any time $t \geq \underline{t}_{i,p}$ to π_j is also infeasible (from Theorem 2); in this case, $\mathcal{E}(\underline{t}_{i,p})$ is set to a very large value. Otherwise, if the trajectories consist of straight lines, $\mathcal{E}(\underline{t}_{i,p})$ is relatively easy to find using the formulae provided in the appendix. The number of steps required to compute l_{pq} in this case is $O(kk_j)$ where k_j is the number of line segments in π_j.

If the trajectories consist of more generic segments where $\mathcal{E}(\underline{t}_{i,p})$ is difficult to compute directly, we can bound $\mathcal{E}(\underline{t}_{i,p})$ using the following approach: We know that if the agent starts at $\pi_i(\underline{t}_{i,p})$, it cannot arrive at π_j at any time earlier than $\mathcal{E}(\underline{t}_{i,p})$. Therefore, if there is a sub-interval $[\mathcal{E}_{lp}, \mathcal{E}_{up}] \subseteq [\underline{t}_j, \bar{t}_j]$ such that the agent can start from $\pi_i(\underline{t}_{i,p})$ and reach $\pi_j(\mathcal{E}_{up})$ but cannot reach $\pi_j(\mathcal{E}_{lp})$, then $\mathcal{E}(\underline{t}_{i,p}) \in [\mathcal{E}_{lp}, \mathcal{E}_{up}]$. To refine this bound, we iteratively partition $[\underline{t}_j, \bar{t}_j]$ into smaller intervals using binary search, ensuring that the above condition is satisfied and $|\mathcal{E}_{up} - \mathcal{E}_{lp}| \leq \epsilon$ where ϵ is another sampling parameter that can be adjusted. Therefore, in this general case, we define $l_{pq} := \min_{p=1}^{k} (\mathcal{E}_{lp} - \bar{t}_{i,p})$. The sampling algorithm in this case will require $O(k \log_2 \frac{\Delta}{\epsilon})$ steps where Δ is the size of $[\underline{t}_j, \bar{t}_j]$.

D. *C**-*Linear*

This variant of *C** finds the optimum for SFT, and can be applied when both trajectory intervals, $\pi_i[\underline{t}_i, \bar{t}_i]$ and $\pi_j[\underline{t}_j, \bar{t}_j]$, correspond to piecewise-linear segments. If $\pi_i[\underline{t}_i, \bar{t}_i]$ and $\pi_j[\underline{t}_j, \bar{t}_j]$ each correspond to only one line segment, we

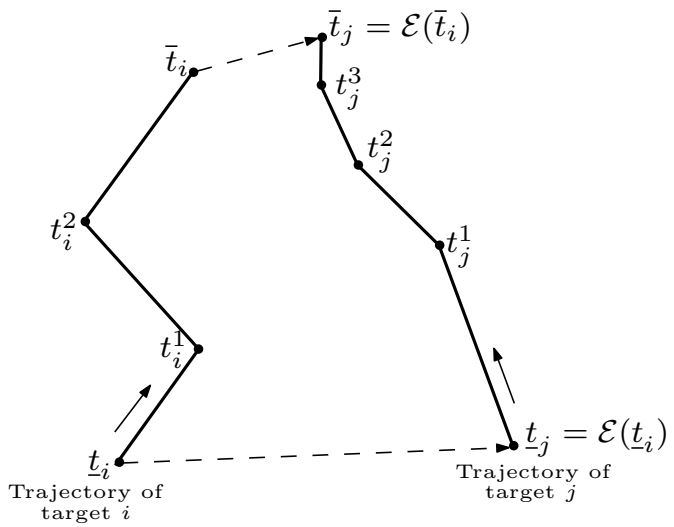


Fig. 3: Given target trajectories for *AlgoSFT*. In the trajectory of target i, there are 3 line-segments and 2 corner points. In the trajectory of target j, there are 4 line-segments and 3 corner points. Also, $T_i := (\underline{t}_i, t_i^1, t_i^2, \bar{t}_i)$ and $T_j := (\underline{t}_j, t_j^1, t_j^2, t_j^3, \bar{t}_j)$.

can simply identify the stationary or boundary points, verify feasibility, and select a solution that yields the optimal cost (derivations for finding the stationary points are provided in the appendix VIII-C). However, if either $\pi_i[\underline{t}_i, \bar{t}_i]$ or $\pi_j[\underline{t}_j, \bar{t}_j]$ corresponds to more than one line segment, considering all combinations of segments like in sub-section V-B will require $O(k_i k_j)$ steps. In this subsection, we present an efficient algorithm that reduces the complexity to $O(k_i + k_j)$ steps.

Before we present our algorithm, we mention why solutions to the EFAT and LFDT problems posed in Section III play a crucial role in our approach here. In a EFAT solution, $\mathcal{E}(t_i)$ denotes the earliest feasible time of arrival to the trajectory π_j from trajectory point $\pi_i(t_i)$. This implies that the agent cannot reach π_j at any time earlier than $\mathcal{E}(t_i)$; also, it is sub-optimal to reach π_j at any time later than $\mathcal{E}(t_i)$. Similar observations can also be deduced using a LFDT solution which provides the latest feasible time of departure from a trajectory to a trajectory-point. In the appendix, we show the calculations to solve the EFAT and LFDT problems. Solutions for these problems can be used to prune unnecessary parts of $\pi_i[\underline{t}_i, \bar{t}_i]$ and $\pi_j[\underline{t}_j, \bar{t}_j]$ that will never lead to an optimal solution. Specifically, we can ensure that $\mathcal{E}(\underline{t}_i) = \underline{t}_j$ (or $\underline{t}_i = \mathcal{L}(\underline{t}_j)$) and $\mathcal{E}(\bar{t}_i) = \bar{t}_j$ (or $\bar{t}_i = \mathcal{L}(\bar{t}_j)$). Henceforth, we will assume that the trajectory-intervals already satisfy these conditions. We will also assume that the trajectory-intervals don't intersect each other. That is there is no time t in $[\underline{t}_i, \bar{t}_i]$ and $[\underline{t}_j, \bar{t}_j]$ such that $\pi_i(t) = \pi_j(t)$; otherwise, this is a trivial case and the optimal SFT cost is 0.

Let T_i and T_j denote lists of selected times (or time instants) corresponding to targets i and j. The times in T_i and T_j are automatically sorted in ascending order as new times are added. T_i and T_j are first initialized with all the times corresponding to the corner[5] trajectory-points in $\pi_i[\underline{t}_i, \bar{t}_i]$ and $\pi_j[\underline{t}_j, \bar{t}_j]$ respectively. In addition, the boundary times $\underline{t}_i, \bar{t}_i$ are added to T_i, and similarly, $\underline{t}_j, \bar{t}_j$ are added to T_j (refer to Fig. 3). Also, let the k^{th} smallest time in T_i and T_j be referred to as $T_i(k)$ and $T_j(k)$ respectively. Our algorithm denoted as *AlgoSFT* is as follows:

1) For each time t in T_i corresponding to a corner-trajectory point in $\pi_i[\underline{t}_i, \bar{t}_i]$, find $\mathcal{E}(t)$ and add $\mathcal{E}(t)$ to T_j. Similarly, for each time t in T_j corresponding to a corner trajectory-point in $\pi_j[\underline{t}_j, \bar{t}_j]$, find $\mathcal{L}(t)$ and add $\mathcal{L}(t)$ to T_i. At the end of this step, $|T_i| = |T_j|$ (refer to Fig. 4).
2) For $k = 1, \cdots, |T_i| - 1$, do the following: Let π_{ik} denote the trajectory-sub-interval of $\pi_i[\underline{t}_i, \bar{t}_i]$ corresponding to $[T_i(k), T_i(k+1)]$. Similarly, let π_{jk} denote the trajectory-sub-interval of $\pi_j[\underline{t}_j, \bar{t}_j]$ corresponding to $[T_j(k), T_j(k+1)]$. Find the SFT cost from π_{ik} to π_{jk} (using the calculations in the appendix VIII-C). Let this cost be denoted as SFT_k.
3) Set $l_{pq} := \min_{k=1}^{|T_i|-1} SFT_k$.

We will now prove that *AlgoSFT* correctly computes the SFT cost from $\pi_i[\underline{t}_i, \bar{t}_i]$ to $\pi_j[\underline{t}_j, \bar{t}_j]$.

Theorem 4. *Consider any choice of times $t_1, t_2 \in [\underline{t}_i, \bar{t}_i]$ such that $t_1 < t_2$. Then, $\mathcal{E}(t_1) < \mathcal{E}(t_2)$.*

Proof. We prove this theorem by contradiction. Suppose $\mathcal{E}(t_1) \geq \mathcal{E}(t_2)$. Let the position occupied by target t_2 at time $\mathcal{E}(t_2)$ be denoted by $p^* = \pi_j(\mathcal{E}(t_2))$. The agent can travel from $\pi_i(t_1)$ to $\pi_i(t_2)$ at speed v_{max} and then travel from $\pi_i(t_2)$ to $\pi_j(\mathcal{E}(t_2))$ following the EFAT path. Since the agent can travel faster than target i, it will reach the location p^* sooner than target t_j. This will allow the agent to further travel along π_j to intercept t_j sooner than $\mathcal{E}(t_2)$. This implies that we have found a new arrival time for the agent that is less than $\mathcal{E}(t_2) \leq \mathcal{E}(t_1)$. As a result $\mathcal{E}(t_1)$ is not the earliest arrival time to visit t_j which is a contradiction. Hence proved. □

Theorem 5. *For any time $t \in [\underline{t}_i, \bar{t}_i]$, $\mathcal{L}(\mathcal{E}(t)) = t$. Similarly, for any time $t \in [\underline{t}_j, \bar{t}_j]$, $\mathcal{E}(\mathcal{L}(t)) = t$.*

Proof. We will prove that for any $t \in [\underline{t}_i, \bar{t}_i]$, $\mathcal{L}(\mathcal{E}(t)) = t$; the other result can be proved by similar arguments. $\mathcal{L}(\mathcal{E}(t))$ denotes the latest departure time available to leave π_i and arrive at $p^* = \pi_j(\mathcal{E}(t))$. Therefore, $\mathcal{L}(\mathcal{E}(t)) \geq t$. Now, suppose $\mathcal{L}(\mathcal{E}(t)) > t$. The agent can travel from $\pi_i(t)$ to $\pi_i(\mathcal{L}(\mathcal{E}(t)))$ at speed v_{max} and then travel from $\pi_i(\mathcal{L}(\mathcal{E}(t)))$ to $\pi_j(\mathcal{E}(t))$ following the EFAT path. Similar to the argument in Theorem 4, if $\mathcal{L}(\mathcal{E}(t)) > t$ is true, the agent will be able to visit target t_j sooner than $\mathcal{E}(t)$ which is not possible. Hence, the only possibility is that $\mathcal{L}(\mathcal{E}(t)) = t$. □

Theorem 6. *AlgoSFT correctly finds the optimal SFT cost from $\pi_i[\underline{t}_i, \bar{t}_i]$ to $\pi_j[\underline{t}_j, \bar{t}_j]$ in the order of $k_i + k_j$ steps where k_i and k_j denote the number of line segments in $\pi_i[\underline{t}_i, \bar{t}_i]$ and $\pi_j[\underline{t}_j, \bar{t}_j]$ respectively.*

Proof. Consider any time $\bar{t}$ in the list T_j at the end of step 1 of *AlgoSFT*. Either $\bar{t} = \mathcal{E}(t)$ for some $t \in T_i$ or there is a

[5]These are break-points where a target trajectory transitions from a line segment to another with a different slope.

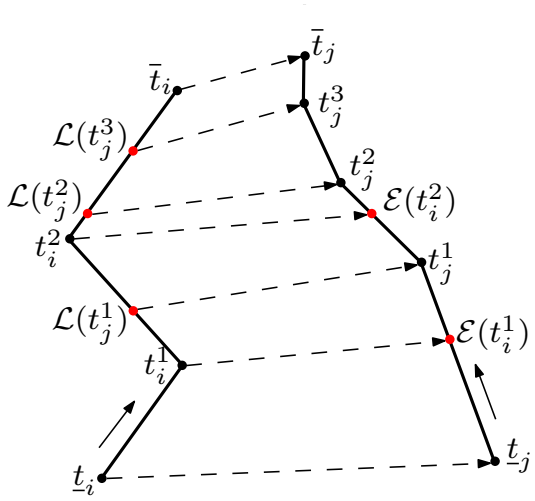


Fig. 4: Target trajectories from Fig. 3 showing the updated lists of times in T_i and T_j. At the end of the step 1 of *AlgoSFT*, $|T_i| = |T_j| = 7$.

time $t \in T_i$ such that $\mathcal{L}(\bar{t}) = t$ in which case $\mathcal{E}(t) = \bar{t}$ from Theorem 5. Therefore, for any time $\bar{t} \in T_j$, there is a $t \in T_i$ such that $\mathcal{E}(t) = \bar{t}$. Similarly, for any $t \in T_i$, there is a $\bar{t} \in T_j$ such that $\mathcal{E}(t) = \bar{t}$.

If $T_i(k)$ and $T_j(k)$ denote the k^{th} smallest time in T_i and T_j respectively, Theorem 4 implies that $T_j(k) = \mathcal{E}(T_i(k))$. Suppose the time to depart in an optimal SFT solution from $\pi_i[\underline{t}_i, \bar{t}_i]$ to $\pi_j[\underline{t}_j, \bar{t}_j]$ is $t^* \in [\underline{t}_i, \bar{t}_i]$. Then, for some $k^* \in 1, \cdots, |T_i| - 1$, $t^* \in [T_i(k^*), T_i(k^* + 1)]$. Applying Theorem 4, it then must follow that $\mathcal{E}(t^*) \in [T_j(k^*), T_j(k^* + 1)]$. Therefore, using the notations in step 2 of *AlgoSFT*, the optimal SFT cost from $\pi_i[\underline{t}_i, \bar{t}_i]$ to $\pi_j[\underline{t}_j, \bar{t}_j]$ must be equal $\min_{k=1}^{|T_i|-1} SFT_k$.

The computation of the optimal SFT cost essentially involves solving $|T_i| - 1$ optimization problems, each requiring the calculation of a fixed number of stationary or boundary points and checking their feasibility. Additionally, we can verify that $|T_i| = |T_j| = k_i + k_j$. Therefore, the number of steps required to implement *AlgoSFT* is in the order of $k_i + k_j$. Hence proved. □

VI. Numerical Results

A. Test Settings and Instance Generation

All the tests were run on a laptop with an Intel Core i7-7700HQ 2.80GHz CPU, and 16GB RAM. For all the C^* variants, relaxing MT-TSP and constructing the graph, including computing the edge costs, were all implemented using Python 3.11.6. An exact branch-and-cut solver, written in C++ and utilizing CPLEX 22.1, was used to solve an integer program for the GTSP, to optimality. For the special case where targets move along straight lines, the SOCP-based formulation by Stieber and Fügenschuh in [17], with the objective modified to minimize travel time, was used as our baseline. This formulation was implemented in CPLEX 22.1 IDE, which uses OPL. More details on this formulation can be found in the appendix (section VIII). The CPLEX parameter *EpGap*[6] was set to be 1e-04 and the CPLEX parameter *TiLim*[7] was set to 7200s for our algorithms as well as the baseline.

A total of 90 instances were generated, with 30 *simple* instances where targets move along lines, 30 *complex* instances where targets move along piecewise-linear paths, and 30 *generic* instances where targets move along Dubins curves made of straight lines and circular arcs. The generic instances are considered separately in section VI-G. For a given instance type, we generated three sets of 10 instances: one set with 5 targets, another set with 10 targets, and a third set with 15 targets. The instances were defined by the number of targets n, a square area of fixed size $100\,units$ containing the start locations of the targets, a fixed time horizon $T = 100\,secs$ over which the target trajectories are defined, the depot location fixed at the bottom-left corner of the square area with coordinates $(10, 10)$, a fixed maximum agent speed of $v_{max} = 4\,units/sec$, and a set of randomly generated trajectories for the n targets, where each target moves at a constant speed within $[0.5, 1]\,units/sec$. Each target was also assigned up to 2 time-windows, whose total duration adds up to $20\,secs$. Note that for the simple instances, we assigned only 1 time-window for each target. We also ensured that the paths traversed by the targets were all confined within the square area. This was so that the baseline SOCP, which relies on these assumptions, could be used.

The time-windows for any given instance were defined as follows. First, the time-window for each target was set to be the entire time horizon, and a feasible solution was found using the algorithm in section VI-B. Second, each target was assigned a primary time-window of duration $15\,secs$, which contains the time that target was visited by the agent in the feasible solution. Third, a secondary time-window of $5\ secs$ that does not intersect with the primary time-window, was randomly assigned to each of the targets as well. For the simple instances, only the primary time-window was assigned to each target, but with an increased duration of $20\,secs$.

Before proceeding further, note that when finding feasible solutions or when running the C^* variants, we partition the time-windows for each target into equal intervals of size $\Delta = 0.625\,secs$. Since for any target, the duration of each time-window is an integer multiple of $5\,secs$, and the total duration from all the time-windows sums to $20\,secs$, we get a total of 32 intervals per target. This is always true unless otherwise specified. Also, when using C^*-*Sampling*, the sampling parameter k is always set to 10, and the gap tolerance ϵ is set to 0.05.

B. Finding Feasible Solutions

We evaluate the quality of bounds from the C^* variants, based on how far, they deviate on average, from feasible solution costs. This section briefly discusses how feasible solutions for the MT-TSP can be obtained by first transforming

[6]Relative tolerance on the gap between the best solution objective and the best bound found by the solver.

[7]Time limit before which the solver terminates.

MT-TSP into a corresponding GTSP and then finding feasible solutions for the GTSP.

The time-windows for each target are first sampled into equally spaced time-instants. The trajectory-points corresponding to these time-instants are then found. A directed graph $\mathcal{G}$ is then constructed, with the vertex set defined as the depot and the set of all trajectory-points found. All the vertices corresponding to a given target are clustered together. If the agent can travel feasibly from a vertex belonging to a cluster to another vertex belonging to a different cluster, a directed edge is added, with the cost being the difference between the time-instants corresponding to the destination vertex and the start vertex. If the destination vertex is the depot, then the edge cost is the time taken by the agent to reach the depot from the start vertex by moving at its maximum speed. If travel between two vertices is not feasible, the cost of the edge between the vertices is set to a large value to denote infeasibility.

A feasible solution for the MT-TSP can now be obtained by finding a directed edge cycle which starts at the depot vertex, visits exactly one vertex from each cluster, and returns to the depot vertex. Note that this is simply, the problem of finding a feasible solution for the GTSP defined on graph $\mathcal{G}$. One way to solve this problem is to first transform the GTSP into an Asymmetric TSP (ATSP) using the transformation in [34], and then find feasible solutions for the ATSP using an LKH solver [35] or some other TSP heuristics. Note that one can also directly use heuristics for GTSP such as GLKH [36], and GLNS [37]. Since the target trajectories are continuous functions of time, the best arrival times for each target can be calculated, given the order in which they were visited in the GTSP solution, resulting in a feasible tour with improved travel time. Note that if the number of discrete time-instants are not sufficient, we may not obtain a feasible solution for the MT-TSP using this approach, even if one exists.

C. Evaluating the Bounds

In this section, we compare, for all the simple and complex instances, the feasible solution costs and the lower-bounding costs from the C^* variants. For simple instances, the optimum from the baseline SOCP is also added to the comparison[8].

The results are presented in Fig. 5. Here, (a), (b), (c) includes all the simple instances, and (d), (e), (f) includes all the complex instances. The instances are sorted from left to right in the order of increasing feasible solution costs. We observe C^*-*Linear* provides the tightest bounds, followed by C^*-*Geometric*, then C^*-*Sampling*, and finally C^*-*Lite*. The bounds from C^*-*Linear* are the strongest since it uses the optimum for SFT. However, these bounds are closely matched by those from C^*-*Geometric* and C^*-*Sampling*. Note that although C^*-*Geometric* relaxes the timing requirements for SFT, it returns slightly stronger bounds than C^*-*Sampling*. The bounds from C^*-*Sampling* however improves, and converges to those from C^*-*Linear* as the sampling parameter k approaches ∞ and the gap tolerance ϵ approaches 0. This however, comes at the expense of increased computational burden. Finally, C^*-*Lite* uses trivial lower-bounds for SFT, making its bounds the weakest. For simple instances, feasible solution costs are tightly bounded by the SOCP costs, with the lower-bounds not exceeding the SOCP costs. This shows that the approach to find feasible solutions is effective, and that the C^* variants indeed provide lower-bounds for the MT-TSP.

For instance-1 in (c), the SOCP cost slightly exceeds the feasible cost. This is because the problem becomes significantly more computationally expensive at 15 targets, and as a result, the CPLEX solver failed to converge to the optimum within the time limit for that instance. Hence, the best feasible cost found by the solver before exceeding the time limit was used for this instance. Similarly, in (c), the bounds from all C^* variants except C^*-*Lite* are weaker for instance-1, and the bounds from all C^* variants are weaker for instance-5. These too, were due to the increased computational complexity when considering 15 targets. Here, the CPLEX solver terminated due to insufficient memory, leaving the gap between the dual bound and the best objective value, not fully converged to be within the specified tolerance. In such cases, the best lower-bound found by the solver before termination was considered, as it still provides an underestimate. These *outlier* instances are illustrated using square markers, indicating that CPLEX solver terminated due to memory constraints.

D. Varying the Number of Targets

In this section, we see how varying the number of targets affects the tightness of bounds, as well as runtimes, for the C^* variants. The SOCP is also evaluated for simple instances. For a given instance, the $\%$ deviation is defined as $\frac{C_f - C_{lb}}{C_f} \times 100$ where C_f denotes the feasible solution cost and C_{lb} denotes the lower-bound. The lower-bound here represents the cost returned from a C^* variant, or the SOCP. The runtime (RT) for an instance is separated into the graph generation runtime (Graph Gen RT), which is the runtime for constructing graphs in the C^* variants, and the total runtime (Total RT), which is the time taken by the C^* variants for graph generation, and then solving GTSP on the generated graph. The runtime for SOCP is also referred to as Total RT. In Fig. 6, (a) considers all the simple instances and (b) considers all the complex instances. The average of the $\%$ deviation from all the instances (except outliers) corresponding to 5, 10, and 15 targets are presented in the figure. Fig. 7 follows the same layout as Fig. 6, and presents runtimes instead of $\%$ deviation.

From both Fig. 6 (a) and (b), we observe that $\%$ deviation for the C^* variants increases as the number of targets are increased. This is to be expected since each visited target in the relaxed MT-TSP incurs an additional discontinuity. As we saw previously, C^*-*Linear* gives the best bounds, with an average $\%$ deviation $\approx 4\%$ for 15 targets. This is followed by C^*-*Geometric*, C^*-*Sampling*, and C^*-*Lite*, respectively. We also observe that the growth in $\%$ deviation with the number of targets is greater in C^*-*Lite* as compared to the other C^* variants. Note how the $\%$ deviation for C^*-*Linear*, C^*-*Geometric*, and C^*-*Sampling* for 15 targets, are still smaller than that of C^*-*Lite* for 5 targets. Finally, we observe the

[8]The optimum is only known for the simple instances solved using the baseline SOCP.

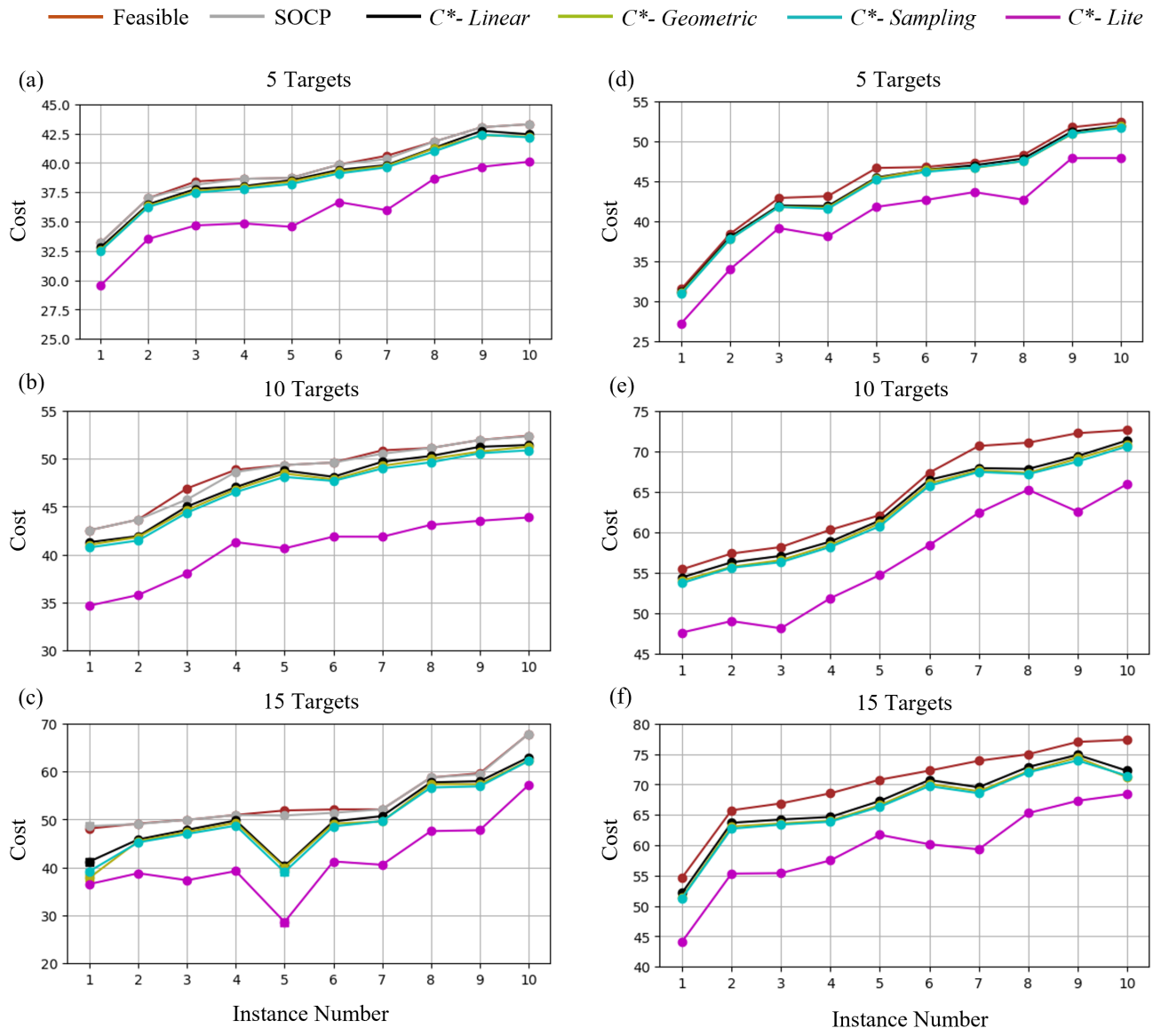


Fig. 5: Lower-bounds from the C^* variants as compared to the feasible costs, for the simple instances (left), and the complex instances (right). The SOCP costs are also included for the simple instances. In (c), square markers indicate the outlier cases for instances 1, and 5.

% deviation for SOCP to be very small, indicating that the feasible solution costs obtained were on average, very close to the optimal costs. Note that the SOCP costs does not depend on the number of targets as it aims to find the optimum.

From Fig. 7 (a) and (b) we see that Graph Gen RT is the smallest for C^*-*Lite*, followed by C^*-*Geometric*, then C^*-*Linear*, and finally, C^*-*Sampling*. This is because the lower-bounding algorithm for SFT in C^*-*Lite* is trivial, making it the fastest. This is followed by the one in C^*-*Geometric*, which solves an easier time-independent problem. C^*-*Linear* finds the optimum for the SFT making it slightly slower. However, it uses *AlgoSFT*, which is specifically tailored for piecewise-linear target trajectories, and involves an efficient search. Finally, for C^*-*Sampling*, the computational burden for finding lower-bounds for SFT increases with larger sampling parameter k, and smaller gap tolerance, ϵ. By setting k to 10 and ϵ to 0.05, C^*-*Sampling* incurred more computational burden than the other C^* variants. We also observe that the rate at which Graph Gen RT grows for the different C^* variants can be explained similarly. Note however, that Graph Gen RT in general contributes very little to the Total RT as compared to the time taken to solve the GTSP. We see a big jump in Total RT with increasing targets, especially between 10 targets and 15 targets. This can be attributed to the increasing complexity of solving GTSP on larger graphs. We observe that on average, the Total RT is the least for C^*-*Geometric* and C^*-*Sampling*, followed by C^*-*Linear*, and then C^*-*Lite*. It is likely that the relatively slower runtimes for C^*-*Lite* is due to its weaker

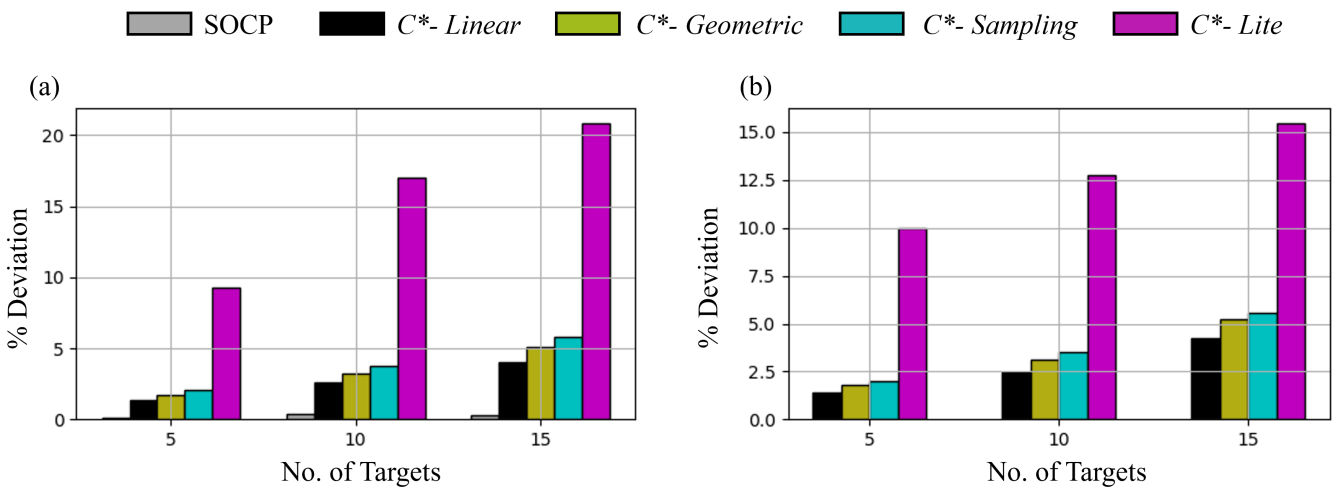


Fig. 6: Plots comparing the average % deviation for the C^* variants (and SOCP for simple instances), as the number of targets are varied. Note how the % deviation for C^*-*Linear* is $\approx 4\%$ for 15 targets, for both the simple (a) and complex (b) instances. Also, note how the average % deviation for SOCP is less than 1%.

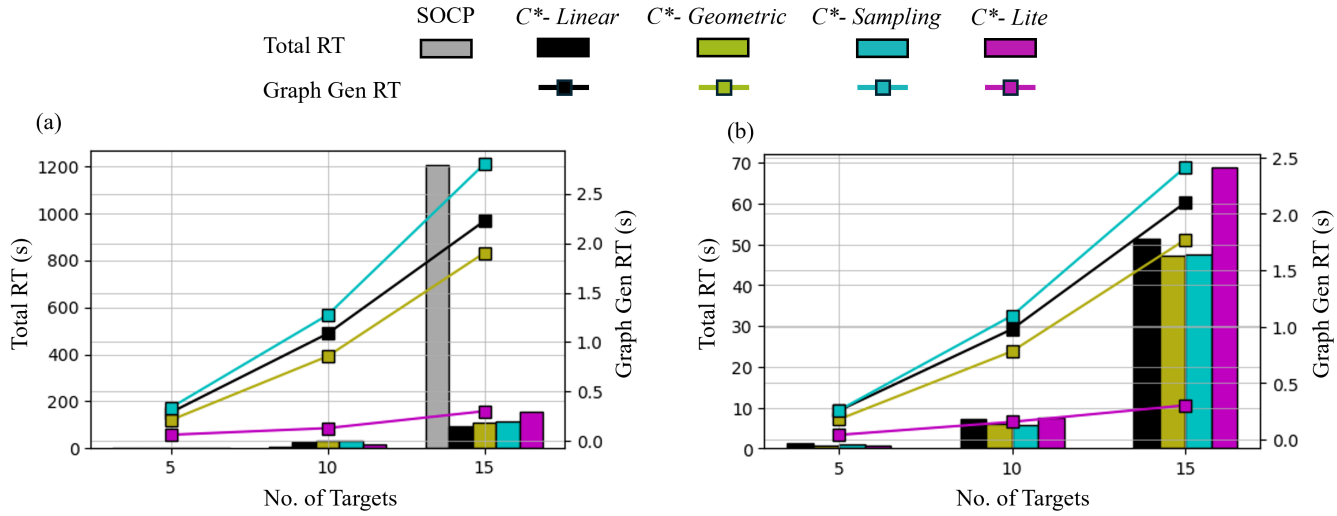


Fig. 7: Plots comparing the average runtimes for the C^* variants, as the number of targets are varied, for both the simple (a), and complex (b) instances. Note how for the simple instances, SOCP is faster than the C^* variants for up to 10 targets, but an order of magnitude slower for 15 targets.

relaxation for MT-TSP, making the underlying GTSP more difficult to solve. Finally, we observe from Fig. 7 (a), how the Total RT for SOCP is significantly smaller than the C^* variants for 5, and 10 targets, but becomes an order of magnitude larger for 15 targets.

E. Varying the Discretization

In this section, we show how varying the discretization levels for C^* variants affect their % deviation and runtimes. Recall that for any target, the duration of each time-window is an integer multiple of $5\ secs$, and the total duration from all the time-windows sums to $20\ secs$. Considering this, we define the discretization levels as follows. At `lvl-1`, the time-windows for each target are partitioned into equal intervals of duration of $5\ secs$. For every new level thereafter, the interval durations are halved, and the number of intervals are doubled. All of this is illustrated in Table. I.

Discretization Level	lvl-1	lvl-2	lvl-3	lvl-4
Intervals per Target	4	8	16	32
Interval Duration	5	2.5	1.25	0.625

TABLE I: Information about the discretization levels.

Fig. 8 (a) and (b) considers all the simple and complex instances, except for the outlier instances previously discussed. From all the instances considered, (a) illustrates the average % deviation, and (b) illustrates the average runtimes, for each discretization level. From (a), we observe how higher

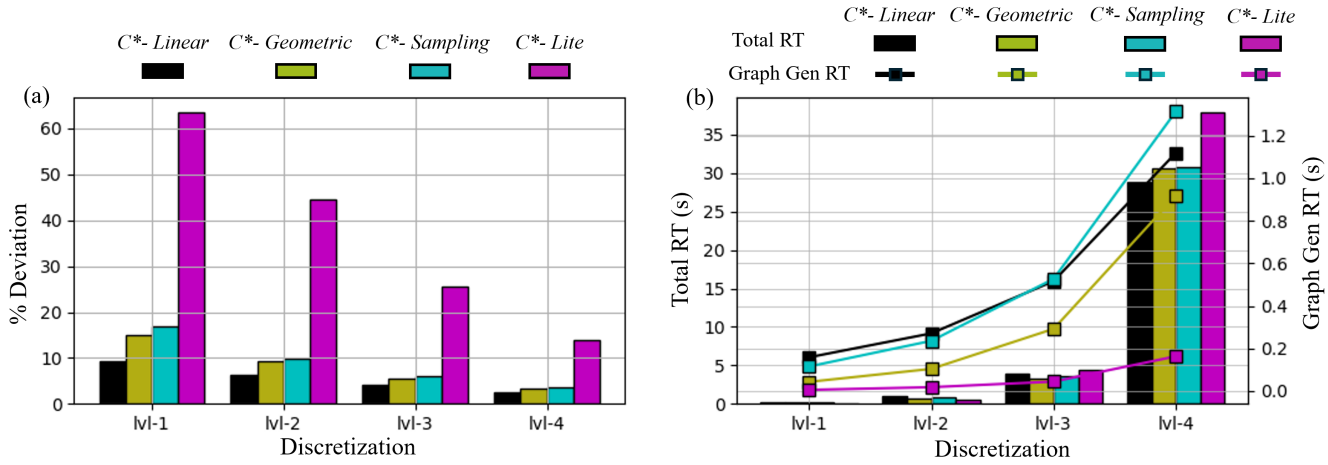


Fig. 8: Plots illustrating (a) the average % deviation for the C^* variants, and (b) the average runtimes for the C^* variants, for different levels of discretization. Note how in (a), C^*-*Linear* at `lvl-1` has lower % deviation than C^*-*Lite* at `lvl-4`. Also in (b), note how at `lvl-4`, the runtimes increase significantly for all approaches.

discretization produces tighter bounds for all the C^* variants as one would expect. Note that the % deviation improvement for C^*-*Lite* is more significant here, as compared to the rest of the variants. Also, note that the % deviation for C^*-*Lite* at `lvl-4` is still higher than it is for C^*-*Linear* at `lvl-1`. Finally, note that the rate at which % deviation improves, decreases at higher discretizations, for all the approaches. From (b), we observe how the runtimes increase with higher discretization. Note that the increase in Graph Gen RT is more significant for C^*-*Linear*, C^*-*Geometric*, and C^*-*Sampling*, than it is for C^*-*Lite* here. Finally, we observe how the growth in Total RT is significantly higher between `lvl-3`, and `lvl-4`, than it is between the previous levels. Like before, this too can be attributed to the increasing complexity of solving GTSP on larger graphs. Sample solutions for a *complex* instance are shown in Fig. 10.

F. Obtaining Feasible Solutions from C^* Variants

In this section, we attempt to construct feasible solutions for the MT-TSP, from lower-bounds obtained from the C^* variants. We also evaluate how good the costs are for these new solutions. Clearly, if the discretization parameter Δ goes to 0, the lower-bounds converge to the optimum. However, this is computationally infeasible and therefore, we will fix the discretization level at `lvl-4`.

To construct feasible solutions from lower-bounds, we first fix the order in which the targets are visited in the lower-bounding solution, and then find a minimum cost tour for the agent over that fixed order such that it a) visits each target within one of its time-windows and b) completes the tour without exceeding its maximum speed v_{max}. For the cases where such a tour cannot be constructed, we say a feasible solution cannot be constructed from a given lower-bound. Note that this is the same procedure we used in section VI-B to obtain feasible tours with improved travel times from the ones initially found by solving the GTSP.

For Table. II, we consider all the simple and complex instances, except for the outlier instances (same instances as in Fig. 8). Here, *Success Rate* illustrates the percentage of instances from which a feasible solution can be constructed from the lower-bound. For such instances, we compare the new feasible solutions with the original ones to determine if the order in which the targets are visited remains the same (or matches). Note that if the orders match, then the arrival times for the targets must also be the same, since we used the same procedure to reoptimize the original feasible solutions, as well as construct new feasible solutions from lower-bounds, as discussed earlier. The percentage of instances where the feasible solutions match is given by % Match in the table. Finally, from all the remaining instances where the solutions do not match, we average the costs of both the feasible solutions originally obtained as well as the newly constructed ones, denoted by C_f^{old} and C_f^{new} respectively, and find % Dev-Mismatch defined as $\frac{C_f^{new} - C_f^{old}}{C_f^{old}} \times 100$.

We observe that the success rate is more than 95% for all the C^* variants, with C^*-*Lite* giving the best rate of 98%. If a feasible solution cannot be constructed from one lower-bound, it might still be possible to construct one from another. Hence, for at least 98% of the instances, we were able to construct feasible solutions from various C^* lower-bounds. We also observe that the % match is similar for C^*-*Linear*, C^*-*Geometric*, and C^*-*Sampling*, and it is significantly higher than for C^*-*Lite*. Finally, we observe that the % dev-mismatch remains close to 0% for all the C^* variants. Specifically, C_f^{new} is marginally worse than C_f^{old} for C^*-*Linear* and C^*-*Lite*, while C_f^{new} is marginally better than C_f^{old} for C^*-*Geometric* and C^*-*Sampling*.

G. Evaluating C^* Variants for Generic Instances

In this section, we consider only the 30 generic instances, where targets move along Dubins curves. We evaluate C^*-

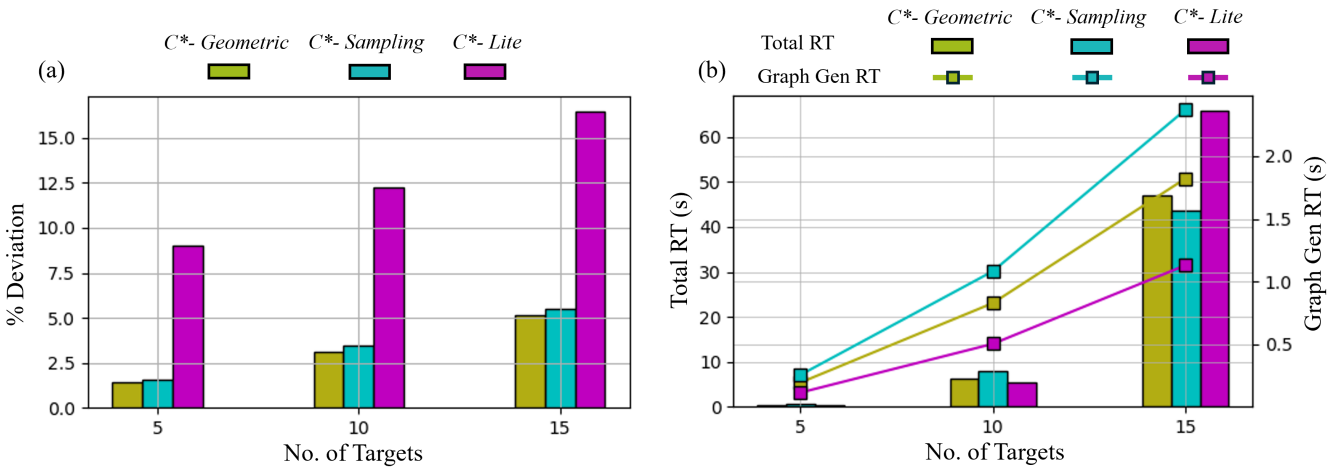


Fig. 9: Plots illustrating (a) the average % deviation, and (b) the average runtimes, for the C^* variants, as the number of targets are varied. Only the generic instances are considered for both the plots.

	Success Rate (%)	% Match	% Dev-Mismatch
C-Linear*	96.55	64.29	0.21
C-Geometric*	96.55	60.34	-0.026
C-Sampling*	96.55	62.07	-0.072
C-Lite*	98.28	47.37	0.27

TABLE II: Table illustrating the success rates for constructing feasible solutions from various C^* lower-bounds, and how these new feasible solutions compare with the originally found feasible solutions.

Geometric, *C*-Sampling*, and *C*-Lite* for these instances, and omit *C*-Linear* as it specifically caters to targets moving along piecewise-linear paths. Fig. 9 illustrates in (a), the average % deviation, and in (b), the average runtimes, for 5, 10, and 15 targets. Like for simple and complex instances in Fig. 6, the % deviation here grows with more targets, with *C*-Geometric* always giving the best bounds, followed by *C*-Sampling*, and then *C*-Lite*. Here too, the % deviation for both *C*-Geometric* and *C*-Sampling* at 15 targets are still smaller than it is for *C*-Lite* at 5 targets. Note that the % deviation for *C*-Geometric* at 15 targets is $\approx 5\%$. Averaging this with the % deviation of $\approx 4\%$ from complex instances, our approaches give on average, a % deviation of $\approx 4.5\%$ for general cases of MT-TSP. The runtime results are also similar to the ones presented for simple and complex instances in Fig. 7. Both Graph Gen RT and Total RT increases with more number of targets. The Graph Gen RT as well as its growth, are more significant for *C*-Geometric* and *C*-Sampling* than it is for *C*-Lite*. Finally, the Total RT grows significantly for all the C^* variants considered, as the number of targets are increased from 10 to 15. Sample solutions for the general case are shown in Fig. 11.

VII. Conclusion and Future Work

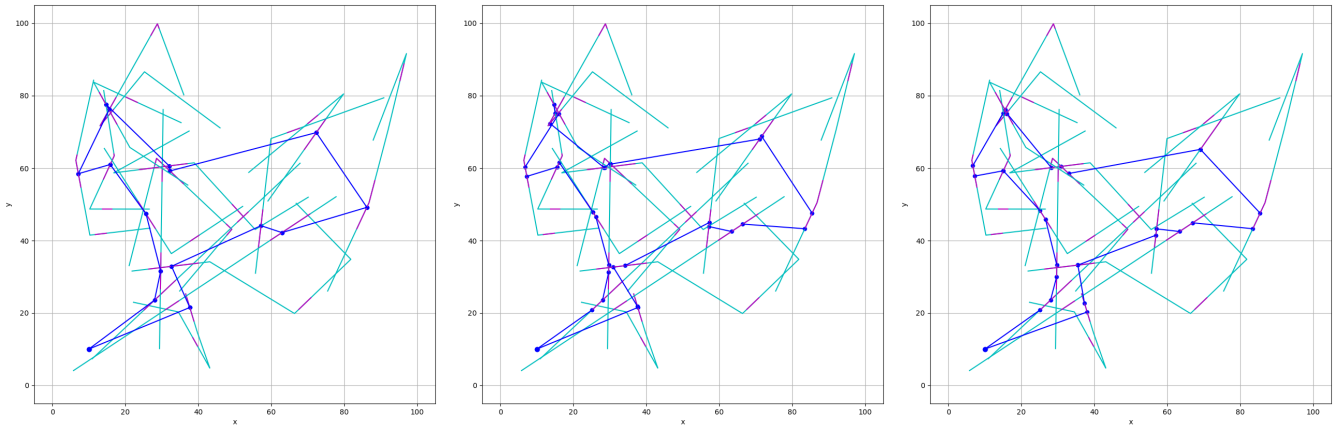
We presented C^*, an approach for finding lower-bounds for the MT-TSP with time-window constraints. Our method can handle generic target trajectories, including piecewise-linear segments and Dubins curves. We introduced several variants of C^*, providing a trade-off between the quality of guarantees and algorithm's running speed. Additionally, we proved that our approaches yield valid lower-bounds for the MT-TSP and presented extensive numerical results to demonstrate the performance of all the variants of C^*. Finally, we showed that feasible solutions can often be constructed from the lower-bounding solutions obtained using C^* variants.

One of the challenges in this paper was the computational burden associated with increasing the number of targets. This difficulty arises from solving the GTSP, where the computational complexity heavily depends on the number of nodes in the generated graph. Finding a way to overcome this challenge would enable us to compute tight bounds for a larger number of targets. In fact, one of our primary future goals is to develop efficient branch-and-cut implementations that leverage the specific features of the MT-TSP problem. For instance, unlike standard GTSP formulations that include subtour elimination constraints, we may be able to omit some of these constraints because the timing constraints in our problem inherently eliminate certain subtours (e.g., those that travel back in time). This represents a promising new research direction that warrants further investigation. Another direction of research can focus on developing approximation algorithms for simpler variants of the MT-TSP.

VIII. Appendix

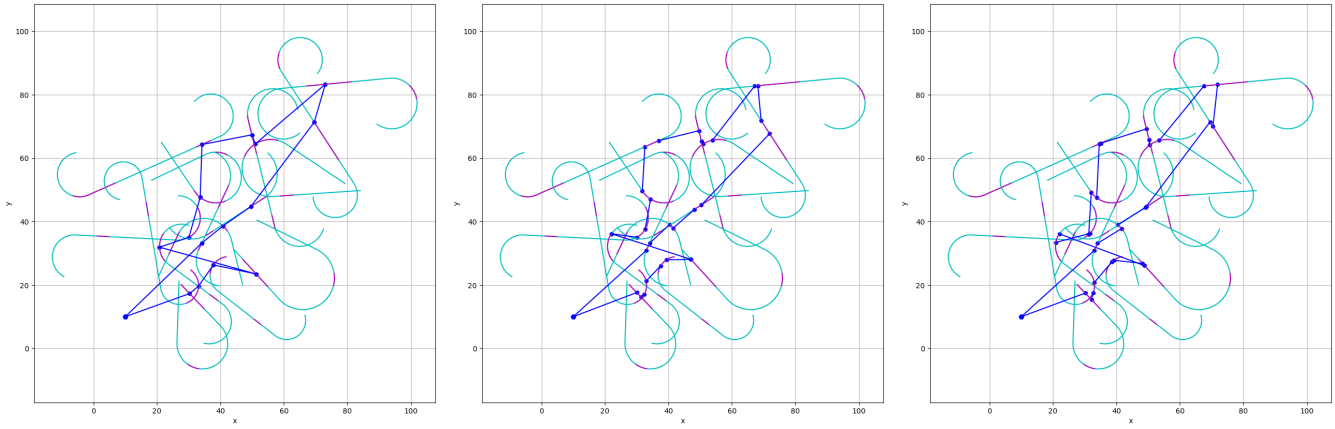
A. Finding Earliest Feasible Arrival Time

Let the trajectory-point $\pi_s(t)$ for some target s at time t be denoted by the tuple (x, y), where x and y are the coordinates of the position occupied by s at time t. Also, let $\dot{x}$ and $\dot{y}$ denote the time derivative of x and y respectively. Let i and j be two targets moving along trajectories π_i and π_j as shown in Fig 12. Note that $\pi_i(t_i) = (a_x, a_y)$ and $\pi_j(t) = (x, y)$. Let $\pi_i(t_1) = (a_{x_1}, a_{y_1})$ and $\pi_j(t_2) = (x_2, y_2)$. The following



(a) Feasible solution (b) C^*-*Linear* (c) C^*-*Geometric*

Fig. 10: A feasible solution (using the algorithm in section VI-B) and the two best bounding solutions for a complex instance with 15 targets moving along piecewise-linear paths.



(a) Feasible solution (b) C^*-*Geometric* (c) C^*-*Sampling*

Fig. 11: A feasible solution (using the algorithm in section VI-B) and the two best bounding solutions for a general instance with 15 targets moving along Dubins curves.

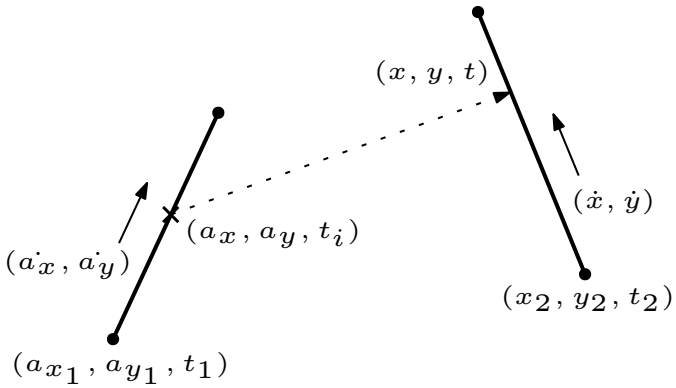


Fig. 12: Trajectories π_i and π_j for targets i and j.

equations then describe the motion of i and j from times t_1 and t_2 onward, respectively.

$$a_x = a_{x_1} + \dot{a_x}(t_i - t_1), \tag{1}$$
$$a_y = a_{y_1} + \dot{a_y}(t_i - t_1), \tag{2}$$
$$x = x_2 + \dot{x}(t - t_2), \tag{3}$$
$$y = y_2 + \dot{y}(t - t_2). \tag{4}$$

The distance between (a_x, a_y) and (x, y) is then given by $\sqrt{(x-a_x)^2 + (y-a_y)^2}$. Also, we want the agent to travel at speed v_{max} to obtain $e(t_i)$. Hence, we want t that satisfies

$$\sqrt{(x-a_x)^2 + (y-a_y)^2} = v_{max}(t - t_i). \tag{5}$$

Squaring both sides, we obtain

$$(x-a_x)^2 + (y-a_y)^2 = v_{max}^2(t-t_i)^2. \tag{6}$$

Substituting (1), (2), (3), (4) into (6), and rearranging the terms, we get

$$(\dot{x}t + C_1)^2 + (\dot{y}t + C_2)^2 = v_{max}^2(t-t_i)^2, \tag{7}$$

where

$$
\begin{aligned}
C_1 &= -\dot{a_x}t_i + C_1', \\
C_2 &= -\dot{a_y}t_i + C_2', \\
C_1' &= x_2 - \dot{x}t_2 - a_{x_1} + \dot{a_x}t_1, \\
C_2' &= y_2 - \dot{y}t_2 - a_{y_1} + \dot{a_y}t_1.
\end{aligned}
$$

After extensive algebra, we finally get the following.

$$
At^2 + B(t_i)t + C(t_i) = 0, \tag{8}
$$

where

$$
\begin{aligned}
A &= \dot{x}^2 + \dot{y}^2 - v_{max}^2, \\
B(t_i) &= 2B't_i + 2C', \\
C(t_i) &= A't_i^2 - D't_i + E', \\
B' &= -\dot{a_x}\dot{x} - \dot{a_y}\dot{y} + v_{max}^2, \\
C' &= C_1'\dot{x} + C_2'\dot{y}, \\
A' &= \dot{a_x}^2 + \dot{a_y}^2 - v_{max}^2, \\
D' &= 2\dot{a_x}C_1' + 2\dot{a_y}C_2', \\
E' &= C_1'^2 + C_2'^2.
\end{aligned}
$$

One of the two roots that satisfies (8) is then the EFAT $\mathcal{E}(t_i)$. These roots[9] can be obtained using the quadratic formula as shown below.

$$
t = \frac{-B(t_i) \pm \sqrt{B(t_i)^2 - 4AC(t_i)}}{2A}. \tag{9}
$$

B. Finding Latest Feasible Departure Time

From Theorem 5, we know that $t = \mathcal{E}(t_i) \iff \mathcal{L}(t) = t_i$. Hence, given a value of t, we seek to find t_i such that t satisfies (8). To solve this, note that (8) can be expanded as follows.

$$
At^2 + (2B't_i + 2C')t + (A't_i^2 - D't_i + E') = 0. \tag{10}
$$

By rearranging the terms in (10) we get the below equation.

$$
A't_i^2 + (2B't - D')t_i + (At^2 + 2C't + E') = 0. \tag{11}
$$

(11) can then be represented simply as

$$
A't_i^2 + H(t)t_i + I(t) = 0. \tag{12}
$$

Hence, one of the two roots that satisfies 12 is the LFDT $\mathcal{L}(t)$. Like previously shown, these roots can be obtained using the quadratic formula below.

$$
t_i = \frac{-H(t) \pm \sqrt{H(t)^2 - 4A'I(t)}}{2A'}. \tag{13}
$$

[9]We obtain two roots since (5) is squared to get (6). However, only one of the roots yield the EFAT. Similar reasoning can be used to explain the same occurrence when finding the LFDT.

C. Finding the Stationary Points for the SFT problem

Consider the same setup as in VIII-A. In the SFT problem, our aim is to find a feasible travel from t_i such that $\mathcal{E}(t_i) - t_i$ is minimized. While in the EFAT problem, t_i is given, here t_i is also a variable and the arrival time t will be a function of t_i. This function has already been derived in Equation (9). By differentiating t in (9) with respect to t_i, we can find an expression for $\frac{dt}{dt_i}$ as follows.

$$
\frac{dt}{dt_i} = \frac{1}{2A}\left[-\frac{d}{dt_i}B(t_i) \pm \frac{2B(t_i)\frac{d}{dt_i}B(t_i) - 4A\frac{d}{dt_i}C(t_i)}{2\sqrt{B(t_i)^2 - 4AC(t_i)}}\right] =
$$

$$
\frac{1}{2A}\left[-2B' \pm \frac{2(2B't_i + 2C')(2B') - 4A(2A't_i - D')}{2\sqrt{(2B't_i + 2C')^2 - 4A(A't_i^2 - D't_i + E')}}\right].
$$

After further simplification, we get

$$
\frac{dt}{dt_i} = \frac{1}{A}\left[-B' \pm \frac{2B'(B't_i + C') - A(2A't_i - D')}{2\sqrt{(B't_i + C')^2 - A(A't_i^2 - D't_i + E')}}\right]. \tag{14}
$$

To find the stationary points of $t - t_i$, we set $\frac{d}{dt_i}(t - t_i) = 0$ which then gives us

$$
\frac{dt}{dt_i} - 1 = 0 \implies \frac{dt}{dt_i} = 1. \tag{15}
$$

Substituting (14) into (15), we get the following.

$$
\frac{1}{A}\left[-B' \pm \frac{2B'(B't_i + C') - A(2A't_i - D')}{2\sqrt{(B't_i + C')^2 - A(A't_i^2 - D't_i + E')}}\right] = 1,
$$

which can be rearranged as

$$
(A + B') = \pm\frac{2B'(B't_i + C') - A(2A't_i - D')}{2\sqrt{(B't_i + C')^2 - A(A't_i^2 - D't_i + E')}}. \tag{16}
$$

Note that $(B't_i + C')^2 - A(A't_i^2 - D't_i + E') \geq 0$ for all t_i. If $(B't_i + C')^2 - A(A't_i^2 - D't_i + E') = 0$ for some t_i, then it means both s_1 and s_2 occupies the same position at time t_i. In this case, the function $t - t_i$ becomes 0 and takes a sharp turn ($\frac{d}{dt_i}(t - t_i)$ becomes undefined) at t_i. However, if $(B't_i + C')^2 - A(A't_i^2 - D't_i + E') > 0$, we get the following by squaring both sides of (16) and multiplying the denominator on both sides.

$$
\begin{aligned}
&4(A + B')^2((B't_i + C')^2 - A(A't_i^2 - D't_i + E')) \\
=&(2B'(B't_i + C') - A(2A't_i - D'))^2.
\end{aligned}
$$

After extensive algebra, we finally get the following.

$$
Pt_i^2 + Qt_i + R = 0, \tag{17}
$$

where

$$\begin{aligned}P &= 4(A+B')^2(B'^2-AA') - 4B'^4 \\ &\quad - (4A^2A'^2 - 8AA'B'^2), \\ Q &= 4(A+B')^2(2B'C'+AD') - 8B'^3C' \\ &\quad - (4AB'^2D' - 8AA'B'C' - 4A^2A'D'), \\ R &= 4(A+B')^2(C'^2-AE') - 4B'^2C'^2 \\ &\quad - (4AB'C'D' + A^2D'^2).\end{aligned}$$

The values of t_i that satisfies the two equations in (16) can be obtained by solving for the two roots that satisfies (17). These roots can once again, be obtained using the quadratic formula given below.

$$t_i = \frac{-Q \pm \sqrt{Q^2 - 4PR}}{2P}. \tag{18}$$

D. Second Order Cone Program (SOCP) Formulation

In this section, we explain the SOCP formulation for the special case of the MT-TSP where targets follow linear trajectories. Our formulation is very similar to the one presented in [17], with a few changes made to accommodate the new objective which is to minimize the time taken by the agent to complete the tour, as opposed to minimizing the length of the path traversed by the agent.

To ensure that the trajectory of the agent starts and ends at the depot, we do the following: Given the depot d and a set of $n-1$ targets $\{1, \cdots, n-1\}$, we define a new stationary target n which acts as a copy of d. This is achieved by fixing n at the same position as d. We then define constraints so that the agent's trajectory starts from d and ends at n. To find the optimal solution to the MT-TSP, we then seek to minimize the time at which n is visited by the agent.

Let $S := \{1, \cdots, n\}$ and $S_d := \{d\} \cup S$. We use the same family of decision variables as in [17] where $x_{i,j} \in \{0,1\}$ indicates the decision of sending the agent from target i to target j ($x_{i,j} = 1$ if yes. No otherwise), and $t_i \in \mathbb{R}$ describes the arrival time of the agent at target i (or depot).

Our objective is to minimize the time at which the agent arrives at target n as shown below.

$$\min t_n. \tag{19}$$

Each target j must be visited once by the agent:

$$\sum_{i \in S_d : i \neq j} x_{i,j} = 1, \ \forall\, j \in S. \tag{20}$$

The agent can start only once from the depot:

$$\sum_{j \in S} x_{d,j} \leq 1. \tag{21}$$

Flow conservation is ensured by:

$$\sum_{i \in S_d : i \neq j} x_{i,j} \geq \sum_{i \in S : i \neq j} x_{j,i}, \ \forall\, j \in S. \tag{22}$$

The agent must visit each target within its assigned time-window. Note that, for the depot d and the target n, we assign the time-window to be the entire time-horizon T:

$$t_l^j \leq t_j \leq t_u^j, \ \forall\, j \in S_d. \tag{23}$$

As shown in [17], real auxiliary variables $c_{i,j}^x$ and $c_{i,j}^y$ for the $x-$ and $y-$ components of the Euclidean distance are introduced as follows:

$$\begin{aligned}&c_{d,j}^x - \left(\left(x_l^j + t_j \frac{\Delta x_j}{\Delta t_j} - t_l^j \frac{\Delta x_j}{\Delta t_j}\right) - d_x\right) = 0, \\ &\forall\, j \in S,\end{aligned} \tag{24}$$

$$\begin{aligned}&c_{d,j}^y - \left(\left(y_l^j + t_j \frac{\Delta y_j}{\Delta t_j} - t_l^j \frac{\Delta y_j}{\Delta t_j}\right) - d_y\right) = 0, \\ &\forall\, j \in S,\end{aligned} \tag{25}$$

$$\begin{aligned}&c_{i,j}^x - \left(\left(x_l^j + t_j \frac{\Delta x_j}{\Delta t_j} - t_l^j \frac{\Delta x_j}{\Delta t_j}\right) - \left(x_l^i + t_i \frac{\Delta x_i}{\Delta t_i} - t_l^i \frac{\Delta x_i}{\Delta t_i}\right)\right) \\ &= 0, \ \forall\, i \in S, j \in S : i \neq j,\end{aligned} \tag{26}$$

$$\begin{aligned}&c_{i,j}^y - \left(\left(y_l^j + t_j \frac{\Delta y_j}{\Delta t_j} - t_l^j \frac{\Delta y_j}{\Delta t_j}\right) - \left(y_l^i + t_i \frac{\Delta y_i}{\Delta t_i} - t_l^i \frac{\Delta y_i}{\Delta t_i}\right)\right) \\ &= 0, \ \forall\, i \in S, j \in S : i \neq j.\end{aligned} \tag{27}$$

Here, for some target i, (x_l^i, y_l^i) represents the coordinates of i at time t_l^i and (x_u^i, y_u^i) represents the coordinates of i at time t_u^i. Also, $\Delta x_i = x_u^i - x_l^i$, $\Delta y_i = y_u^i - y_l^i$, and $\Delta t_i = t_u^i - t_l^i$. Finally, (d_x, d_y) denotes the coordinates of the depot d.

The following conditions requires that if the agent travels between any two targets or the depot and a target, this travel must be feasible:

$$\begin{aligned}&a_{i,j} \leq v_{max}(t_j - t_i + T(1 - x_{i,j})), \\ &\forall\, i \in S_d, j \in S : i \neq j,\end{aligned} \tag{28}$$

$$a_{i,j} \geq 0, \ \forall\, i \in S_d, j \in S : i \neq j. \tag{29}$$

The below conditions are needed to formulate the cone constraints:

$$\overline{a}_{i,j} = a_{i,j} + R(1 - x_{i,j}), \ \forall\, i \in S_d, j \in S : i \neq j. \tag{30}$$

Where given the square area with fixed side length L that contains all the moving targets and the depot, $R = \sqrt{2}L$ is the length of the square's diagonal.

Finally, the cone constraints are given as:

$$(c_{i,j}^x)^2 + (c_{i,j}^y)^2 \leq (\overline{a}_{i,j})^2, \ \forall\, i \in S_d, j \in S : i \neq j. \tag{31}$$

The agent must visit n only after visiting all the other targets:

$$t_n \geq t_j, \ \forall\, j \in S_d. \tag{32}$$

Remark 2. *Although* (28) *prevents subtours in most cases, they can still arise very rarely when two or more target trajectories intersect at a time common to their time-windows.*

The constraints defined by (33) *prevents subtours for the two target case. However for more than two targets, we will need additional subtour elimination constraints:*

$$x_{i,j} + x_{j,i} \leq 1, \ \forall\, i \in S, j \in S : i \neq j. \tag{33}$$

REFERENCES

[1] P. Oberlin, S. Rathinam, and S. Darbha, "Today's traveling salesman problem," *IEEE robotics & automation magazine*, vol. 17, no. 4, pp. 70–77, 2010.

[2] Y. Liu and R. Bucknall, "Efficient multi-task allocation and path planning for unmanned surface vehicle in support of ocean operations," *Neurocomputing*, vol. 275, pp. 1550–1566, 2018.

[3] J. L. Ryan, T. G. Bailey, J. T. Moore, and W. B. Carlton, "Reactive tabu search in unmanned aerial reconnaissance simulations," in *1998 Winter Simulation Conference. Proceedings (Cat. No. 98CH36274)*, vol. 1. IEEE, 1998, pp. 873–879.

[4] Z. Yu, L. Jinhai, G. Guochang, Z. Rubo, and Y. Haiyan, "An implementation of evolutionary computation for path planning of cooperative mobile robots," in *Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No. 02EX527)*, vol. 3. IEEE, 2002, pp. 1798–1802.

[5] A. M. Ham, "Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming," *Transportation Research Part C: Emerging Technologies*, vol. 91, pp. 1–14, 2018.

[6] S. Venkatachalam, K. Sundar, and S. Rathinam, "A two-stage approach for routing multiple unmanned aerial vehicles with stochastic fuel consumption," *Sensors*, vol. 18, no. 11, p. 3756, 2018.

[7] H. A. Saleh and R. Chelouah, "The design of the global navigation satellite system surveying networks using genetic algorithms," *Engineering Applications of Artificial Intelligence*, vol. 17, no. 1, pp. 111–122, 2004.

[8] O. Cheikhrouhou, A. Koubâa, and A. Zarrad, "A cloud based disaster management system," *Journal of Sensor and Actuator Networks*, vol. 9, no. 1, p. 6, 2020.

[9] J. Conesa-Muñoz, G. Pajares, and A. Ribeiro, "Mix-opt: A new route operator for optimal coverage path planning for a fleet in an agricultural environment," *Expert Systems with Applications*, vol. 54, pp. 364–378, 2016.

[10] W. Zhao, Q. Meng, and P. W. Chung, "A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario," *IEEE transactions on cybernetics*, vol. 46, no. 4, pp. 902–915, 2015.

[11] B. L. Brumitt and A. Stentz, "Dynamic mission planning for multiple mobile robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3. IEEE, 1996, pp. 2396–2401.

[12] R. S. de Moraes and E. P. de Freitas, "Experimental analysis of heuristic solutions for the moving target traveling salesman problem applied to a moving targets monitoring system," *Expert Systems with Applications*, vol. 136, pp. 392–409, 2019.

[13] Y. Wang and N. Wang, "Moving-target travelling salesman problem for a helicopter patrolling suspicious boats in antipiracy escort operations," *Expert Systems with Applications*, vol. 213, p. 118986, 2023.

[14] D. Marlow, P. Kilby, and G. Mercer, "The travelling salesman problem in maritime surveillance–techniques, algorithms and analysis," in *Proceedings of the international congress on modelling and simulation*, 2007, pp. 684–690.

[15] A. Maskooki and M. Kallio, "A bi-criteria moving-target travelling salesman problem under uncertainty," *European Journal of Operational Research*, 2023.

[16] C. S. Helvig, G. Robins, and A. Zelikovsky, "The moving-target traveling salesman problem," *Journal of Algorithms*, vol. 49, no. 1, pp. 153–174, 2003.

[17] A. Stieber and A. Fügenschuh, "Dealing with time in the multiple traveling salespersons problem with moving targets," *Central European Journal of Operations Research*, vol. 30, no. 3, pp. 991–1017, 2022.

[18] C. D. Smith, *Assessment of genetic algorithm based assignment strategies for unmanned systems using the multiple traveling salesman problem with moving targets*. University of Missouri-Kansas City, 2021.

[19] C. Groba, A. Sartal, and X. H. Vázquez, "Solving the dynamic traveling salesman problem using a genetic algorithm with trajectory prediction: An application to fish aggregating devices," *Computers & Operations Research*, vol. 56, pp. 22–32, 2015.

[20] I. Granado, L. Hernando, Z. Uriondo, and J. A. Fernandes-Salvador, "A fishing route optimization decision support system: The case of the tuna purse seiner," *European Journal of Operational Research*, vol. 312, no. 2, pp. 718–732, 2024.

[21] K. Sriniketh, A. V. Le, R. E. Mohan, B. J. Sheu, V. D. Tung, P. Van Duc, and M. B. Vu, "Robot-aided human evacuation optimal path planning for fire drill in buildings," *Journal of Building Engineering*, vol. 72, p. 106512, 2023.

[22] B. Englot, T. Sahai, and I. Cohen, "Efficient tracking and pursuit of moving targets by heuristic solution of the traveling salesman problem," in *52nd ieee conference on decision and control*. IEEE, 2013, pp. 3433–3438.

[23] M. W. Savelsbergh, "Local search in routing problems with time windows," *Annals of Operations research*, vol. 4, pp. 285–305, 1985.

[24] P. Chalasani and R. Motwani, "Approximating capacitated routing and delivery problems," *SIAM Journal on Computing*, vol. 28, no. 6, pp. 2133–2149, 1999.

[25] M. Hammar and B. J. Nilsson, "Approximation results for kinetic variants of tsp," in *Automata, Languages and Programming: 26th International Colloquium, ICALP'99 Prague, Czech Republic, July 11–15, 1999 Proceedings 26*. Springer, 1999, pp. 392–401.

[26] A. G. Philip, Z. Ren, S. Rathinam, and H. Choset, "A mixed-integer conic program for the moving-target traveling salesman problem based on a graph of convex sets," *arXiv preprint arXiv:2403.04917*, 2024.

[27] J.-M. Bourjolly, O. Gurtuna, and A. Lyngvi, "On-orbit servicing: a time-dependent, moving-target traveling salesman problem," *International Transactions in Operational Research*, vol. 13, no. 5, pp. 461–481, 2006.

[28] N. S. Choubey, "Moving target travelling salesman problem using genetic algorithm," *International Journal of Computer Applications*, vol. 70, no. 2, 2013.

[29] Q. Jiang, R. Sarker, and H. Abbass, "Tracking moving targets and the non-stationary traveling salesman problem," *Complexity International*, vol. 11, no. 2005, pp. 171–179, 2005.

[30] U. Ucar and S. K. Işleyen, "A meta-heuristic solution approach for the destruction of moving targets through air operations." *International Journal of Industrial Engineering*, vol. 26, no. 6, 2019.

[31] M. Hassoun, S. Shoval, E. Simchon, and L. Yedidsion, "The single line moving target traveling salesman problem with release times," *Annals of Operations Research*, vol. 289, pp. 449–458, 2020.

[32] 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.

[33] G. Laporte, H. Mercure, and Y. Nobert, "Generalized travelling salesman problem through n sets of nodes: the asymmetrical case," *Discrete Applied Mathematics*, vol. 18, no. 2, pp. 185–197, 1987.

[34] C. E. Noon and J. C. Bean, "An efficient transformation of the generalized traveling salesman problem," *INFOR: Information Systems and Operational Research*, vol. 31, no. 1, pp. 39–44, 1993.

[35] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *European journal of operational research*, vol. 126, no. 1, pp. 106–130, 2000.

[36] ——, "Solving the equality generalized traveling salesman problem using the lin–kernighan–helsgaun algorithm," *Mathematical Programming Computation*, vol. 7, pp. 269–287, 2015.

[37] S. L. Smith and F. Imeson, "Glns: An effective large neighborhood search heuristic for the generalized traveling salesman problem," *Computers & Operations Research*, vol. 87, pp. 1–19, 2017.

Allen George Philip (Student Member, IEEE) received the B.S. degree in Mechanical Engineering with a Mathematics Minor from Wichita State University, Wichita, KS, USA, in 2021. He is currently a Ph.D. candidate in the Mechanical Engineering Department at Texas A&M University, College Station, TX, USA, and a graduate research assistant at the university's Autonomous Systems Laboratory.

Zhongqiang (Richard) Ren (Member, IEEE) received the dual B.E. degree from Tongji University, Shanghai, China, and FH Aachen University of Applied Sciences, Aachen, Germany, and the M.S. and Ph.D. degrees from Carnegie Mellon University, Pittsburgh, PA, USA. He is currently an assistant Professor at the UM-SJTU Joint Institute and the Department of Automation, Shanghai Jiao Tong University, Shanghai, China.

Sivakumar Rathinam (Senior Member, IEEE) received the Ph.D. degree from the University of California at Berkeley in 2007. He is currently a Professor with the Mechanical Engineering Department, Texas A&M University. His research interests include motion planning and control of autonomous vehicles, collaborative decision making, combinatorial optimization, vision-based control, and air traffic control.

Howie Choset (Fellow, IEEE) received the undergraduate degrees in computer science and business from the University of Pennsylvania, Philadelphia, PA, USA, and the M.S. and Ph.D. degrees in mechanical engineering from Caltech, Pasadena, CA, USA. He is a Professor in the Robotics Institute, Carnegie Mellon, Pittsburgh, PA, USA.