Mixed-Integer Motion Planning for Multiple Traveling Salesmen Problems with Connected Visitation

Xinyuan Wang[1], Lekang Zhou[1], Runhui Long[1], Zhitong Tang[1], Zhongqiang Ren[1]

***Abstract*—This paper considers a multiple traveling salesman problem (MTSP) with connected visitation (CV) that seeks shortest paths for multiple robots to visit a set of target locations, with each target visited exactly once, while achieving global connectivity among all robots when any robot visits any target, for the purpose of information sharing among the robots. While both MTSP and connectivity maintenance have been extensively studied, there is little work on MTSP-CV. This problem is challenging as it not only inherits the fundamental difficulties of target allocation and sequencing from MTSP, but also requires global connectivity maintenance, where the robots need to plan when and where to reach to get connected with others. To address the difficulties, we develop two new exact mixed-integer programs, which allow joint optimization of agents' paths and the locations and timing for connection. The first is a bilinear program, which is straightforward yet relatively slow to solve due to the loose convex relaxation. The second reformulates this bilinear program using perspective techniques, resulting in a mixed-integer second-order conic program with tight convex relaxation. Experimental results show that both our programs outperform an existing mixed-integer linear approach adapted from the literature, and our conic program can reduce the median runtime by up to 86.2% and the average optimality gap by up to 99.97% compared with the bilinear program. We showcase the use of our method on multiple real drones in a lab setting.**

***Index Terms*—Optimization, Mixed-Integer Programming, Traveling Salesman Problem, Motion Planning**

I. Introduction

Given a set of static target locations, the traveling salesman problem (TSP) seeks a shortest path for a robot to visit each of those targets once and return to the starting location (i.e., depot). TSP is a well-known NP-hard problem [1], [2], and intuitively, the challenge lies in finding an optimal sequence (i.e., visit order) from all permutations of the targets. Multiple traveling salesman problem (MTSP) [3], [4] generalizes TSP from using a single robot to multiple robots, which is harder than TSP as it further involves allocating the targets among the robots.

This paper considers MTSP with connected visitation (MTSP-CV), a generalization of MTSP with global connectivity constraints. Here, a pair of robots is connected if the distance between them is within a prespecified communication range R_C, and all robots are globally connected if any pair of robots can communicate with each other either directly or via other robots as relays. Specifically, we consider two variants of MTSP-CV. The first variant is with Agent Connectivity (MTSP-CV-AC), which requires all robots to be globally connected at any time instant when any robot visits any target. The second variant is with Depot Connectivity (MTSP-CV-DC), which requires a robot to be connected with the depot at the time instant when it visits a target, either directly or via other agents as relays. When the communication range R_C is infinitely large, both variants of MTSP-CV reduce to the standard MTSP, and are therefore NP-hard.

All authors are with Global College, Shanghai Jiao Tong University. Corresponding author: Zhongqiang Ren. zhongqiang.ren@sjtu.edu.cn

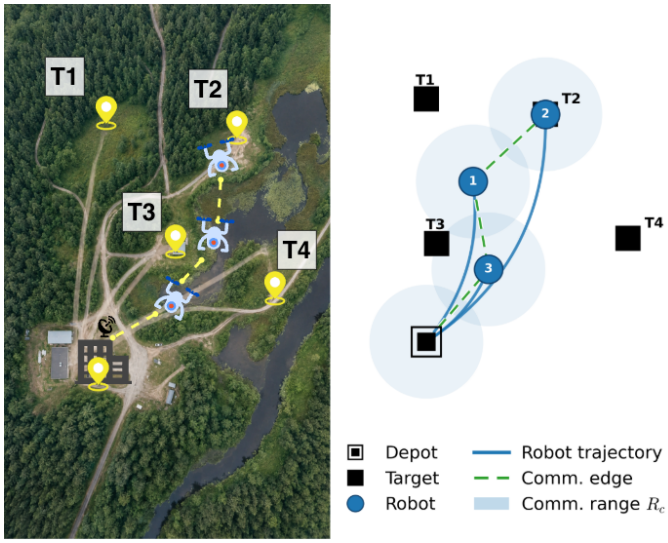


Fig. 1. A motivating example of MTSP-CV-DC, where multiple drones need to collectively visit several target locations while maintaining communication with the workstation at the depot when visiting each target. Each drone has only a limited communication range and some of them serve as relays.

MTSP-CV is a fundamental problem that naturally arises in applications such as surveillance and environmental monitoring [5]–[8], where, with only pairwise local communication, a robot needs to share the collected information with all other robots, or send the information back to the depot, when a target is visited (Fig. 1). MTSP-CV is challenging as it not only inherits the fundamental difficulties of target allocation and sequencing from MTSP, but is subject to connectivity maintenance constraints as well, which requires intelligently determining when and where the robots should reach to be connected with other robots. Solving MTSP-CV to optimality requires addressing all these difficulties simultaneously, which leads to planning in a large and constrained state space.

This paper is interested in exact methods for MTSP-CV that can solve the problem with solution optimality guarantees.

While TSP and MTSP have been investigated a lot using integer programming approaches with good scalability [1], [3], the only prior work that considers a problem similar to ours is a mixed-integer linear programming (MILP) approach [9]. However, we find that this MILP approach is computationally inefficient due to the frequent use of big-M constraints, which leads to loose relaxation and slows down the computation, and the inefficient formulation of visit order of targets and inter-robot connectivity.

To address the difficulties, we develop two new mixed-integer programs for the problem, which are based on the common formulations for MTSP in the literature, and use single-commodity flow to ensure connectivity among the robots. The first program is a bilinear program that relies on binary variables and bilinear terms to encode the connectivity maintenance constraints among the robots, which is straightforward and relatively easy to follow. While our bilinear formulation converges faster than the adapted MILP approach [9] from the literature, it still suffers from relatively low computational efficiency due to the non-convex relaxation of the program caused by the bilinear terms. Our second program reformulates the first bilinear program by using perspective techniques to bypass the bilinear terms, which results in a mixed-integer second-order cone program (MICP). We prove that our MICP is equivalent to the bilinear program, as they have the same optimal solution costs and an optimal solution for one program can be recovered from the other.

Experimental results show that our MICP reduces the median runtime by up to 86.2% and the average optimality gap by up to 99.97% compared with the bilinear formulation, and that both of our formulations outperform an MILP baseline adapted from the literature [9], whose average gap grows to 92.5% on instances where ours remain below 6.9%. Finally, we deploy our planner on real drones in a lab setting for verification.

For the rest of the article, Sec. II reviews the related work, and Sec. III defines MTSP-CV. Sec. IV presents our bilinear formulation, and Sec. V develops the MICP reformulation and proves its equivalence to the bilinear program. Sec. VI reports the experimental results, including comparisons with the MILP baseline and the real-robot demonstrations, and Sec. VII concludes the article.

II. Related Work

A. Traveling Salesman Problems and Perspective Techniques

TSP is one of the most well-known NP-hard problems with a long history [1]. To solve TSP, a variety of methods have been developed ranging from exact approaches (such as branch-and-bound, branch-and-price) [1] to heuristics [10] and approximation algorithms [11], with the latter methods trading solution optimality for runtime efficiency. As a generalization of TSP, MTSP further considers assigning targets to multiple agents [3], [4], [12], and is the basis for a variety of problems in logistics [13] and surveillance [14], [15]. Vehicle routing problems [16] and team orienteering problems [17] are both closely related to MTSP with additional constraints or different objective functions.

To solve these TSP and its variants to optimality, a popular approach is to formulate them as integer or mixed-integer programs and solve them using optimization software such as Gurobi or CPLEX. These formulations usually involve a lot of big-M constraints, which can lead to loose convex relaxation and slow down the computation. To obtain tight relaxations and speed up the computation, perspective techniques were used in the seminal work about the shortest path problem on graphs of convex sets [18], [19]. Later, perspective techniques were introduced to moving-target TSP [12], [20]–[23], where one or multiple agents need to visit targets moving along known trajectories, to provide tight relaxation of the mixed-integer conic program to speed up the solution process [12], [24]. Besides, recent work also developed control perspective techniques for multi-agent motion planning [25] for tight relaxations and computational speedup.

While the idea of using perspective is common in those works, their specific formulations are drastically different from one another. A major challenge when leveraging perspective is to identify the "right" formulation for specific problems, since improper formulation can break the underlying structure of the problem and even slow down the solution process. These recent studies inspire us to leverage perspective and find the "right" formulations with tight convex relaxation to speed up our bilinear program for the MTSP-CV problem.

B. Connectivity Maintenance

Connectivity maintenance has been extensively studied in multi-robot systems with different requirements on connectivity. Continual connectivity requires all robots to stay connected at all times globally [26]–[28] or locally [29], which is suitable for communication-critical missions. Periodic connectivity [30], [31] requires all robots to regain connectivity at pre-defined fixed intervals. Intermittent connectivity [32]–[35] enables the robots to meet at some locations intermittently over time without enforcing a fixed time interval between two subsequent connections, demonstrating higher flexibility. Our work requires global intermittent connectivity: we impose connectivity only at target-visit events in the MTSP problem, as is motivated by applications in which information must be exchanged when a target is visited, rather than continuously throughout the entire mission.

Different approaches were proposed to maintain intermittent connectivity. Some methods focus on selecting connection times or locations, such as searching other robots for connection [36], choosing rendezvous locations [37], planning reconnection paths with minimum travel cost [38], and determining connection times and locations using integer linear programming [39]. Other work seeks to first determine the connection schedule and then plan the trajectories based on the schedules [32]–[34]. Our formulation embeds connectivity requirements into the same mixed-integer program that determines target assignment, global visiting order, visit times, agent positions, and travel feasibility simultaneously, which can therefore provide solution quality guarantees.

III. Problem Definition

We consider MTSP-CV in a 2D Euclidean plane. Let $I_a := \{1, 2, \ldots, N\}$ denote the set of agents, and $I_k :=$

$\{1, 2, \ldots, K\}$ denote the set of targets. Each target $k \in I_k$ is located at a fixed position $p_k := (p_{k,x}, p_{k,y})$ and is associated with a time window $[\underline{t}_k, \bar{t}_k]$. We introduce two depot nodes: a start depot indexed by 0, and an end depot indexed by $K+1$, which can either share the same position or be different. Define the set of all nodes as $I_{k0} := \{0\} \cup I_k \cup \{K+1\}$. As a special case, the depots do not have time windows, and the positions are denoted by $(p_{0,x}, p_{0,y})$ and $(p_{K+1,x}, p_{K+1,y})$. All agents share a common maximum speed $v_{\max}$, start from the depot 0 at time 0, and must reach the end depot $K+1$ before a given time horizon $t_{\max}$. Each target $k \in I_k$ must be visited exactly once, and visiting a target means the position of an agent $i \in I_a$ coincides with the target position p_k at a time point t_k within the target's time window $[\underline{t}_k, \bar{t}_k]$.

In addition, let R_c denote the communication range between any pair of agents. We only consider range-based pairwise communication in a sense that as long as the distance between a pair of agents is within R_c, the agents are *connected* and can communicate freely, without considering other communication factors such as bandwidth and delay. The agents are required to satisfy connectivity constraints at each time point when a target is visited. We consider the following two types of connectivity constraints.

- Agent Connectivity (AC): At any time when a target is visited, all agents must stay connected with each other either directly or via other agents as relays. In other words, consider that when two agents are within distance R_c from each other, we draw a connection edge between them. Then, all these edges should contain a spanning tree among all agents so that every pair of agents is connected within this spanning tree.
- Depot Connectivity (DC): At any time when a target is visited by an agent $i \in I_a$, the agent i must stay connected with the depot either directly or via other agents as relays. In this case, it does not require all agents to stay connected. Instead, only the agent that visits the target must stay connected with the depot.

Note that we only require connectivity when a target is visited, which is motivated by the fact that the agent may need to share information with the workstation at the depot (depot connectivity), or share with all other agents (agent connectivity). During the motion of the agents from one target to another, connectivity is not required since the agents may delay or deviate from their ideal paths during motion execution, and all agents can still get globally connected and exchange information when the next target is visited by some agent as planned.

The goal of the MTSP-CV problem is to find a set of paths, one for each agent, so that all targets are visited by exactly one agent while satisfying the connectivity constraints and minimizing the total path length. We refer to the MTSP-CV with Agent Connectivity as MTSP-CV-AC, and the MTSP-CV with Depot Connectivity as MTSP-CV-DC, from now on.

IV. Mixed-Integer Bilinear Program

This section presents our bilinear formulation for MTSP-CV, which is relatively straightforward, and the next section presents our conic formulation for MTSP-CV that leverages perspective techniques to bypass some of the bilinear terms in the bilinear formulation, which provides tighter convex relaxation. This section first presents the common formulation for MTSP-CV-AC and MTSP-CV-DC and then discusses the constraints that differ.

A. Decision Variables

To better describe agents' transitions between targets, we define a directed graph $G = (V, E)$ with node set $V := I_{k0}$, where I_{k0} is the set containing the start depot, all targets, and the end depot. The directed edge set is $E := \{(u, v) : u, v \in V,\ u \neq v\}$. For each node $u \in V$, we define $E_u^{\text{out}} := \{(u, v) \in E\}$ and $E_u^{\text{in}} := \{(v, u) \in E\}$ as the sets of outgoing and incoming edges, respectively. Throughout the formulation, a directed edge is denoted as $e = (u, v)$.

Now we define the decision variables. For each target $k \in I_k$ and agent $i \in I_a$, let $a_k^i \in \{0, 1\}$ indicate whether agent i visits target k. Our formulation will use a global target-visit sequence shared by all agents, rather than individual route variables for each agent as explained later. For each directed edge $e = (u, v) \in E$, let $b_e \in \{0, 1\}$ be the routing variable, indicating whether node v immediately follows node u in the global target-visit sequence. For each node $k \in I_{k0}$ and agent $i \in I_a$, the vector $p_k^i \in \mathbb{R}^2$ denotes the position of agent $i \in I_a$ when node k is visited, and t_k denotes the corresponding visit time point. In other words, there are K target-visit time points in total, one for each target, in addition to the start-depot and end-depot times. For each of those time points t_k, each agent has a corresponding position vector p_k^i. These position vectors of the agents will be used to formulate the connectivity constraints and the travel distance later.

To activate position and time constraints only on selected edges, we introduce auxiliary product variables. For each edge $e = (u, v) \in E$ and agent $i \in I_a$, let $z_e^{p,i} = b_e p_u^i$ and $z_e^{\prime p,i} = b_e p_v^i$. Similarly, for each edge $e = (u, v) \in E$, let $z_e^t = b_e t_u$ and $z_e^{\prime t} = b_e t_v$. For each edge $e \in E$ and agent $i \in I_a$, we define $\ell_e^i \in \mathbb{R}^2$ as the displacement of agent i along edge e, and $\bar{\ell}_e^i \in \mathbb{R}_{\geq 0}$ as its Euclidean norm. The complete bilinear formulation is presented below.

B. Common Constraints

The following Constraint (1) ensures that each target is visited by exactly one agent.

$$\sum_{i \in I_a} a_k^i = 1, \quad \forall k \in I_k \tag{1}$$

The following flow constraints of depots and target nodes are common in TSP literature.

$$\sum_{e \in E_0^{\text{out}}} b_e = 1, \qquad \sum_{e \in E_0^{\text{in}}} b_e = 0 \tag{2}$$

$$\sum_{e \in E_{K+1}^{\text{in}}} b_e = 1, \qquad \sum_{e \in E_{K+1}^{\text{out}}} b_e = 0 \tag{3}$$

Constraints (2) and (3) select one outgoing edge from the start depot and one incoming edge to the end depot,

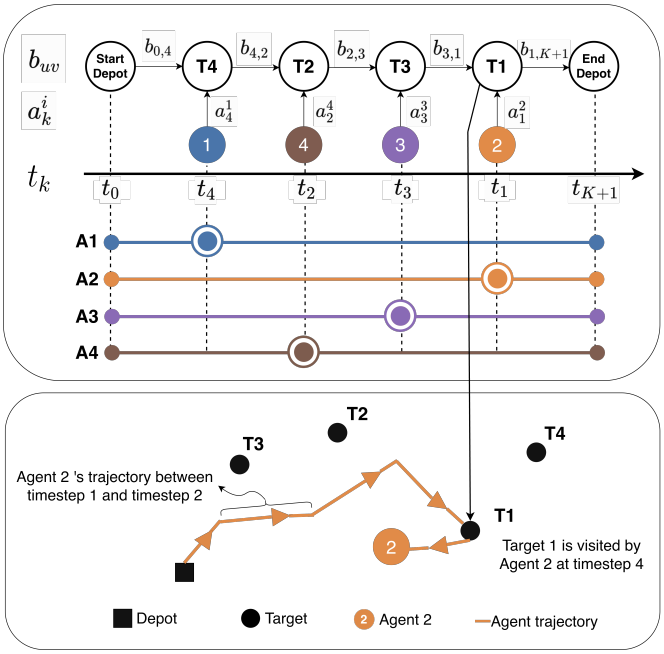


Fig. 2. Illustration of the global target-visit sequence and the corresponding agent assignment and motion. The upper panel shows the selected routing variables b_{uv} defining the global order from the start depot through targets $T4$, $T2$, $T3$, and $T1$ to the end depot. The assignment variables a_k^i identify the agent that visits each target. The colored horizontal lines for A1, A2, A3, and A4 represent the agents along the shared sequence of target-visit events. The lower panel illustrates the trajectory of Agent 2, which visits target T_1 at target 1's visit time, denoted by t_1.

respectively, and forbid any edge into the start depot or out of the end depot.

$$\sum_{e \in E_k^{\text{in}}} b_e = 1, \quad \forall k \in I_k \tag{4}$$

$$\sum_{e \in E_k^{\text{out}}} b_e = 1, \quad \forall k \in I_k \tag{5}$$

Constraints (4) and (5) require each target to have exactly one predecessor and one successor. Recall that the variables b_e define a single global target-visit sequence shared by all agents in which all targets are visited (Fig. 2).

$$p_0^i = p_0, \quad \forall i \in I_a, \tag{6}$$

$$p_{K+1}^i = p_{K+1}, \quad \forall i \in I_a, \tag{7}$$

$$p_k^i \le p_k + (1 - a_k^i) M_{\text{pos}}, \quad \forall i \in I_a,\ \forall k \in I_k, \tag{8}$$

$$p_k^i \ge p_k - (1 - a_k^i) M_{\text{pos}}, \quad \forall i \in I_a,\ \forall k \in I_k, \tag{9}$$

These constraints bind the agent positions to the target positions. Constraints (6) and (7) fix the agent positions at the start and end depots. Constraints (8) and (9) bind p_k^i to the fixed target position p_k if agent i visits target k; otherwise, the bounds are relaxed by the big-M term M_{pos}, which is an upper bound on the maximum distance between any pair of points in the workspace.

$$t_0 = 0 \tag{10}$$

$$t_{K+1} \le t_{\max} \tag{11}$$

$$\underline{t}_k \le t_k \le \bar{t}_k, \quad \forall k \in I_k \tag{12}$$

$$t_u + \epsilon \le t_v + M_{\text{time}}(1 - b_e), \quad \forall e = (u, v) \in E \tag{13}$$

The target visit times obey the time-window and time-ordering constraints. Constraints (10) and (12) fix the start time at zero and require each target to be visited within its required time window, and Constraint (11) requires all agents to reach the end depot no later than the time horizon $t_{\max}$. Constraint (13) enforces temporal progression: if $e = (u, v)$ is selected ($b_e = 1$), then node v must be visited after node u, where M_{time} is an upper bound on the maximum possible visit time, and $\epsilon > 0$ is a small constant:

Remark 1. *Here, the constant ϵ helps eliminate subtours. If ϵ were omitted, a selected edge would only impose $t_u \le t_v$. Then a directed subtour could satisfy all ordering constraints by assigning the same visit time to every node in the subtour. With ϵ, however, every selected edge requires a strict increase in time. An alternative way to eliminate subtours is to adapt MTZ or DFJ formulations from the TSP literature [40] to the problem here. We choose to use this small ϵ due to its simplicity. Finally, note that we are able to use this small ϵ to eliminate subtours, because the visit time t_k of each target is an explicit variable in our formulation, while in the classic MTZ or DFJ formulations for TSP, there is no such notion of visit time of targets.*

The following two constraints define the product variables as aforementioned. Note that the use of these product variables introduces bilinear terms, which makes the entire formulation bilinear and thus non-convex.

$$z_e^{p,i} = b_e\, p_u^i, \quad z_e'^{p,i} = b_e\, p_v^i, \ \forall i \in I_a, \ \forall e = (u, v) \in E \tag{14}$$

$$z_e^t = b_e\, t_u, \qquad z_e'^t = b_e\, t_v, \quad \forall e = (u, v) \in E \tag{15}$$

These auxiliary product variables couple the routing variable b_e with the agent positions and visit times, and enter the kinematics constraints (16) and (18) below. The displacement of each agent along an edge, its norm, and the speed limit are enforced by:

$$\ell_e^i = z_e'^{p,i} - z_e^{p,i}, \quad \forall i \in I_a, \ \forall e \in E \tag{16}$$

$$\|\ell_e^i\|_2 \le \bar{\ell}_e^i, \quad \forall i \in I_a, \ \forall e \in E \tag{17}$$

$$\bar{\ell}_e^i \le v_{\max}\big(z_e'^t - z_e^t\big), \quad \forall i \in I_a, \ \forall e \in E \tag{18}$$

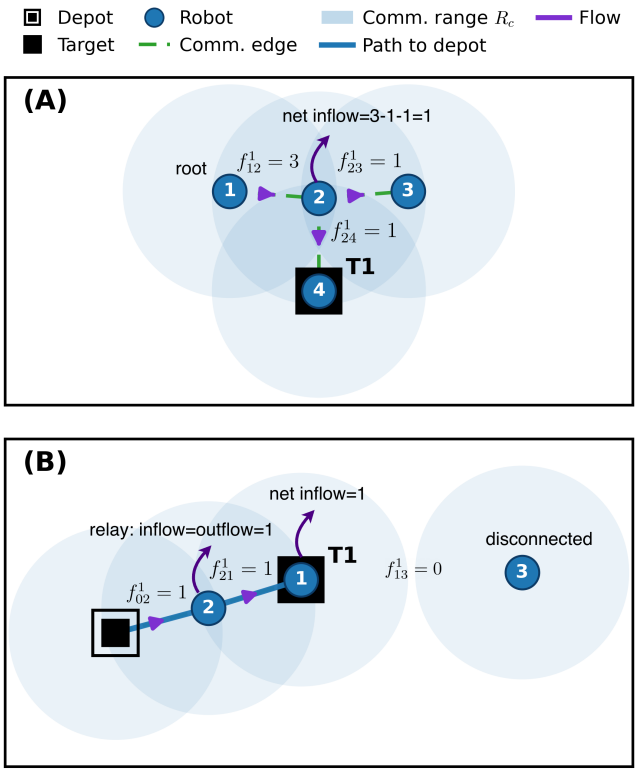


Fig. 3. Illustration of the flow connectivity method when visiting target $T1$. In the agent-connectivity setting (A), Agent 1 is designated as the root and sends $N-1=3$ units of single-commodity flow. Agent 2 receives three units, forwards one unit to Agent 3 and one unit to Agent 4, and therefore has one unit of net inflow. Consequently, every non-root agent receives one unit of net flow, certifying global connectivity. In the depot-connectivity setting (B), Agent 1 visits the target and receives one unit of flow from the depot through relay Agent 2. Agent 2 has equal inflow and outflow, while agents not used as relays (such as Agent 3) are not required to remain connected.

Constraints (16) and (17) define the displacement along each edge. If edge e is selected, the auxiliary product variables recover the endpoint positions, so ℓ_e^i is the displacement of agent i along e; if e is not selected, the product variables vanish, and ℓ_e^i and $\bar{\ell}_e^i$ are zero in an optimal solution. Constraint (18) guarantees speed feasibility by requiring the travel distance on each selected edge to be no greater than the distance that can be traveled within the corresponding time difference.

C. Constraints for Agent Connectivity

For MTSP-CV-AC, our formulation is inspired by the single-commodity flow formulation for TSP in [41]. When a target k is visited, at that time point, all agents should form a "communication graph" whose vertices are the agents, and edges indicate whether the pair of agents are close enough for communication. An undirected communication edge can exist between two agents only if their distance is at most R_c. Since the flow variables are directed, we allow the flow to traverse each undirected communication edge in either direction. For each target $k \in I_k$ and each unordered pair of distinct agents $i, j \in I_a$, let $z_{ij}^k \in \{0,1\}$ indicate whether the undirected communication edge between agents i and j is selected when target k is visited, with the convention $z_{ij}^k = z_{ji}^k$. We also introduce the continuous flow variable $f_{ij}^k \in [0, N-1]$ to represent the amount of connectivity flow sent from agent i to agent j when target k is visited. The idea is to use flow as a certificate of connectivity. An arbitrary fixed "root" agent in this communication graph sends out $N-1$ units of flow, and each non-root agent must receive one unit of net flow. Since flow is allowed to go through only those selected communication edges, when all non-root agents receive one unit of net flow, it implies that every non-root agent is reachable from the root. Therefore, all agents are ensured to be connected in the communication graph (Fig. 3). Finally, let $\ell_{ij,\text{conn}}^k \in \mathbb{R}^2$ and $\bar{\ell}_{ij,\text{conn}}^k \in \mathbb{R}_{\geq 0}$ denote the relative displacement and distance between agents i and j when target k is visited. Agent connectivity constraints are as follows:

$$\ell_{ij,\text{conn}}^k = p_k^i - p_k^j, \quad \forall i < j,\ \forall k \in I_k \tag{19}$$

$$\left\|\ell_{ij,\text{conn}}^k\right\|_2 \leq \bar{\ell}_{ij,\text{conn}}^k, \quad \forall i < j,\ \forall k \in I_k \tag{20}$$

$$\bar{\ell}_{ij,\text{conn}}^k \leq R_c + M_{\text{conn}}(1 - z_{ij}^k), \quad \forall i < j,\ \forall k \in I_k \tag{21}$$

$$\sum_{j \in I_a \setminus \{r\}} f_{rj}^k = N-1, \quad \forall k \in I_k \tag{22}$$

$$\sum_{j \in I_a \setminus \{i\}} f_{ji}^k - \sum_{j \in I_a \setminus \{i\}} f_{ij}^k = 1, \quad \forall i \in I_a \setminus \{r\},\ \forall k \in I_k. \tag{23}$$

$$0 \leq f_{ij}^k \leq \begin{cases} (N-1)\, z_{ij}^k, & i = r, \\ (N-2)\, z_{ij}^k, & i \in I_a \setminus \{r\}, \end{cases} \quad \forall i \in I_a,\ \forall j \in I_a \setminus \{i\},\ \forall k \in I_k \tag{24}$$

First, Constraints (19)–(21) ensure that every selected communication edge connects two agents within communication range. Constraint (19) defines the relative displacement between agents i and j when target k is visited, and Constraint (20) upper-bounds its Euclidean norm by $\bar{\ell}_{ij,\text{conn}}^k$. Constraint (21) uses M_{conn} to activate the communication-range constraint: if $z_{ij}^k = 1$, agents i and j are within range R_c when target k is visited; otherwise, the constraint is relaxed. In other words, z_{ij}^k indicates whether the undirected communication edge between agents i and j is selected for the connectivity certificate. After the selected communication edges are defined, we ensure that these edges connect all agents.

Constraints (22)–(24) enforce connectivity using a single-commodity flow formulation: a designated root agent r sends $N-1$ units of flow to the other agents through active edges. Constraint (22) sets the total flow sent by the root agent, Constraint (23) requires each non-root agent to receive one unit of net flow, and Constraint (24) allows flow only on active communication edges by upper-bounding f_{ij}^k with z_{ij}^k. Inspired by [41], we introduce the strengthened single-commodity-flow with an origin-dependent capacity in Constraint (24). The root outgoing edges have capacity $N-1$, whereas non-root outgoing edges have capacity $N-2$ because a non-root

edge never needs to carry flow for either the root or its own tail node. This preserves all feasible integer solutions while strengthening the continuous relaxation, particularly for the conic formulation as explained later.

D. Constraints for Depot Connectivity

For MTSP-CV-DC, the agent visiting a target must be connected to the depot, either directly or through other agents within communication range acting as relays. We treat the depot and the agents as communication nodes $I_c := \{0\} \cup I_a$. For each target $k \in I_k$, we define $q_0^k := p_0$ and $q_i^k := p_k^i$ for $i \in I_a$, where q_u^k denotes the position of communication node $u \in I_c$ when target k is visited. We define the following additional decision variables for depot connectivity. For each $k \in I_k$, $u \in I_c$ and $v \in I_a$ with $u \neq v$, let $z_{uv}^k \in \{0,1\}$ indicate whether the directed communication edge from node u to node v is selected, let continuous $f_{uv}^k \in [0,1]$ be the flow sent from u to agent v, and let $\ell_{uv,\text{conn}}^k \in \mathbb{R}^2$ and $\bar{\ell}_{uv,\text{conn}}^k \in \mathbb{R}_{\geq 0}$ denote their relative displacement and distance, all when target k is visited. Depot connectivity constraints are as follows:

$$\ell_{uv,\text{conn}}^k = q_u^k - q_v^k, \forall u \in I_c,\ \forall v \in I_a,\ u \neq v,\ \forall k \in I_k \quad (25)$$

$$\|\ell_{uv,\text{conn}}^k\|_2 \leq \bar{\ell}_{uv,\text{conn}}^k, \forall u \in I_c,\ \forall v \in I_a,\ u \neq v,\ \forall k \in I_k \quad (26)$$

$$\bar{\ell}_{uv,\text{conn}}^k \leq R_c + M_{\text{conn}}(1 - z_{uv}^k), \quad \forall u \in I_c,\ \forall v \in I_a,\ u \neq v,\ \forall k \in I_k \quad (27)$$

$$\sum_{v \in I_a} f_{0v}^k = 1, \quad \forall k \in I_k \quad (28)$$

$$\sum_{u \in I_c \setminus \{i\}} f_{ui}^k - \sum_{v \in I_a \setminus \{i\}} f_{iv}^k = a_k^i, \quad \forall i \in I_a,\ \forall k \in I_k \quad (29)$$

$$0 \leq f_{uv}^k \leq z_{uv}^k, \quad \forall u \in I_c,\ \forall v \in I_a,\ u \neq v,\ \forall k \in I_k \quad (30)$$

Constraints (25)–(27) are similar to the agent-connectivity constraints over the communication nodes: (25) is the relative displacement between nodes u and v, (26) bounds its Euclidean norm, and (27) activates the edge variable z_{uv}^k when the two nodes lie within range R_c. Connectivity to the depot is then enforced by a single-commodity flow rooted at the depot. Constraint (28) injects one unit of flow at the depot. Constraint (29) enforces that the agent visiting target k receives one unit of net flow from the depot, since $a_k^i = 1$ only for that agent. For all other agents, $a_k^i = 0$, so their inflow must equal their outflow. In other words, they may relay the flow but cannot consume it. Constraint (30) permits flow only over active edges. Together, these constraints guarantee a path in the communication graph from the depot to the visiting agent when a target is visited.

E. Summary and Discussion

The objective function to be minimized for both AC and DC variants is the total travel distance of all agents:

$$\min \sum_{i \in I_a} \sum_{e \in E} \bar{\ell}_e^i \quad (31)$$

Let $\mathcal{X}_{\text{com}} := \{a_k^i, b_e, p_k^i, t_k, z_e^{p,i}, z_e'^{p,i}, z_e^t, z_e'^t, \ell_e^i, \bar{\ell}_e^i\}$ denote all common decision variables shared by AC and DC. Let $\mathcal{X}_{\text{AC}} := \mathcal{X}_{\text{com}} \cup \{z_{ij}^k, f_{ij}^k, \ell_{ij,\text{conn}}^k, \bar{\ell}_{ij,\text{conn}}^k\}$ and $\mathcal{X}_{\text{DC}} := \mathcal{X}_{\text{com}} \cup \{z_{uv}^k, f_{uv}^k, \ell_{uv,\text{conn}}^k, \bar{\ell}_{uv,\text{conn}}^k\}$ denote the decision variables for the AC and DC variants, respectively.

The entire bilinear program for MTSP-CV-AC is as follows:

$$\min_{\mathcal{X}_{\text{AC}}} \sum_{i \in I_a} \sum_{e \in E} \bar{\ell}_e^i \quad (32)$$

$$\text{s.t. } (1) - (18), (19) - (24). \quad (33)$$

The entire bilinear program for MTSP-CV-DC is as follows:

$$\min_{\mathcal{X}_{\text{DC}}} \sum_{i \in I_a} \sum_{e \in E} \bar{\ell}_e^i \quad (34)$$

$$\text{s.t. } (1) - (18),\ (25) - (30). \quad (35)$$

An optimal solution can be recovered from the selected edges b_e, assignments a_k^i, visit times t_k, and agent positions p_k^i. However, the products between binary and continuous variables make the formulation mixed-integer, whose continuous relaxation is non-convex, and difficult to solve in practice. In the next section, we reformulate these bilinear terms into conic constraints, whose continuous relaxation is then convex.

V. Mixed-Integer Conic Program

A. Perspective and Cone Constraints

In this section, we reformulate the bilinear formulation as a mixed-integer conic program (MICP). The conic formulation keeps the same decision variables a_k^i, b_e, p_k^i, t_k, ℓ_e^i, $\bar{\ell}_e^i$, $z_e^{p,i}$, $z_e'^{p,i}$, z_e^t, and $z_e'^t$ as in the bilinear formulation, as well as z_{ij}^k, f_{ij}^k, $\ell_{ij,\text{conn}}^k$, $\bar{\ell}_{ij,\text{conn}}^k$ for AC and z_{uv}^k, f_{uv}^k, $\ell_{uv,\text{conn}}^k$, $\bar{\ell}_{uv,\text{conn}}^k$ for DC, respectively. The main source of nonconvexity in the bilinear formulation is the product between the binary routing variable b_e and the continuous position and time variables, namely $z_e^{p,i} = b_e p_u^i$, $z_e'^{p,i} = b_e p_v^i$, $z_e^t = b_e t_u$, and $z_e'^t = b_e t_v$ for all $e = (u,v) \in E$.

The key idea of the MICP formulation is to replace those bilinear terms by perspective constraints over convex sets. For each node $u \in V$, define the convex state set in $\mathbb{R}^2 \times \mathbb{R}$:

$$X_u = \{(q,\tau) \mid \|q\|_\infty \leq M_{\text{pos}},\ \underline{t}_u \leq \tau \leq \bar{t}_u\}. \quad (36)$$

Here, $q \in \mathbb{R}^2$ represents an agent position corresponding to node u, and $\tau \in \mathbb{R}$ represents the visit time of node u. For the depot nodes, we use the time bounds $[\underline{t}_0, \bar{t}_0] = [0,0]$ and $[\underline{t}_{K+1}, \bar{t}_{K+1}] = [0, t_{\max}]$. Since all position and time bounds are finite, X_u is also closed and bounded, and hence compact. The perspective of a compact convex set $X \subset \mathbb{R}^n$ is:

$$\widetilde{X} := \{(\hat{x}, \lambda) \in \mathbb{R}^n \times \mathbb{R}_{\geq 0} \mid \hat{x} \in \lambda X\}. \quad (37)$$

where $\hat{x}$ denotes the scaled state and $\lambda \geq 0$ is the scaling variable. Thus, for each node $u \in V$, the perspective of X_u is

$$\widetilde{X}_u := \left\{(\hat{q}, \hat{\tau}, \lambda) \in \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R}_{\geq 0} \;\middle|\; (\hat{q}, \hat{\tau}) \in \lambda X_u \right\}. \quad (38)$$

For each edge $e = (u, v) \in E$ and agent $i \in I_a$, the perspective representations are $(z_e^{p,i}, z_e^t, b_e) \in \widetilde{X}_u$ and $(z_e'^{p,i}, z_e'^t, b_e) \in \widetilde{X}_v$. Expanding the definitions of $\widetilde{X}_u$ and $\widetilde{X}_v$ in these representations yields the inequalities, which are referred to as the perspective constraints. We refer to $z_e^{p,i}$, $z_e'^{p,i}$, z_e^t, and $z_e'^t$ as the perspective auxiliary variables in the conic formulation because they represent the scaled states at the tail and head nodes of edge e. Therefore, when $b_e = 1$, we have $(z_e^{p,i}, z_e^t) \in X_u$ and $(z_e'^{p,i}, z_e'^t) \in X_v$, and when $b_e = 0$, we have $z_e^{p,i} = z_e'^{p,i} = 0$ and $z_e^t = z_e'^t = 0$. The nonconvex bilinear equalities are replaced by perspective constraints.

The routing constraints are restated with an additional constraint which explicitly excludes the direct edge from the start depot to the end depot. It is redundant but strengthens the continuous relaxation for the conic formulation.

$$b_{0,K+1} = 0 \quad (39)$$

$$\sum_{e \in E_0^{\text{out}}} b_e = 1, \qquad \sum_{e \in E_0^{\text{in}}} b_e = 0 \quad (40)$$

$$\sum_{e \in E_{K+1}^{\text{in}}} b_e = 1, \qquad \sum_{e \in E_{K+1}^{\text{out}}} b_e = 0 \quad (41)$$

$$\sum_{e \in E_k^{\text{in}}} b_e = 1, \quad \forall k \in I_k \quad (42)$$

$$\sum_{e \in E_k^{\text{out}}} b_e = 1, \quad \forall k \in I_k \quad (43)$$

The position variables are linked to the routing through perspective constraints rather than the bilinear products (14):

$$\|z_e^{p,i}\|_\infty \leq M_{\text{pos}} b_e, \quad \forall i \in I_a, \ \forall e \in E \quad (44)$$

$$\|z_e'^{p,i}\|_\infty \leq M_{\text{pos}} b_e, \quad \forall i \in I_a, \ \forall e \in E \quad (45)$$

$$\sum_{e \in E_k^{\text{in}}} z_e'^{p,i} = \sum_{e \in E_k^{\text{out}}} z_e^{p,i}, \quad \forall i \in I_a, \ \forall k \in I_k \quad (46)$$

$$p_k^i = \sum_{e \in E_k^{\text{out}}} z_e^{p,i}, \quad \forall i \in I_a, \ \forall k \in I_k \quad (47)$$

$$\sum_{e \in E_0^{\text{out}}} z_e^{p,i} = p_0, \qquad \sum_{e \in E_0^{\text{in}}} z_e'^{p,i} = 0, \quad \forall i \in I_a \quad (48)$$

$$\sum_{e \in E_{K+1}^{\text{in}}} z_e'^{p,i} = p_{K+1}, \qquad \sum_{e \in E_{K+1}^{\text{out}}} z_e^{p,i} = 0, \quad \forall i \in I_a \quad (49)$$

Constraints (44) and (45) activate the perspective auxiliary position variables. When $b_e = 0$ they force the variables to zero, and when $b_e = 1$ they allow the perspective variables to represent feasible position states at the endpoints of edge e. Constraint (46) enforces consistency between the incoming and outgoing states at each target, requiring the position on the selected incoming edge to equal the position on the selected outgoing edge for every agent. Constraint (47) recovers the agent position p_k^i from the selected outgoing edge of target k, and Constraints (48) and (49) impose the corresponding recovery at the start and end depots. Finally, the recovered position p_k^i is bound to the visited target's position by the reused Constraints (8) and (9).

The visit times are handled with the following time perspective and ordering constraints:

$$\underline{t}_u \, b_e \leq z_e^t \leq \bar{t}_u \, b_e, \quad \forall e = (u, v) \in E \quad (50)$$

$$\underline{t}_v \, b_e \leq z_e'^t \leq \bar{t}_v \, b_e, \quad \forall e = (u, v) \in E \quad (51)$$

$$z_e'^t \geq z_e^t + \epsilon \, b_e, \quad \forall e \in E \quad (52)$$

$$\sum_{e \in E_k^{\text{in}}} z_e'^t = \sum_{e \in E_k^{\text{out}}} z_e^t, \quad \forall k \in I_k \quad (53)$$

$$t_k = \sum_{e \in E_k^{\text{out}}} z_e^t, \quad \forall k \in I_k \quad (54)$$

$$\sum_{e \in E_0^{\text{out}}} z_e^t = t_0, \qquad t_0 = 0 \quad (55)$$

$$t_{K+1} = \sum_{e \in E_{K+1}^{\text{in}}} z_e'^t \quad (56)$$

Constraints (50) and (51) activate the perspective auxiliary time variables, forcing them to zero when $b_e = 0$ and bounding them within the node's time window when $b_e = 1$. Constraint (52) enforces strict temporal progression on each selected edge in perspective form. Constraint (53) requires the arrival and departure times at each target to coincide, and Constraint (54) recovers the visit time t_k from the selected outgoing edge. Constraints (55) and (56) impose the corresponding recovery at the start and end depots, with $t_0 = 0$. The connectivity constraints remain the same, namely, (19)–(24) for global connectivity, and (25)–(30) for depot connectivity. In the reused kinematics and connectivity constraints, the auxiliary variables $z_e^{p,i}, z_e'^{p,i}, z_e^t, z_e'^t$ are now defined by the perspective constraints above instead of the bilinear products (14)–(15).

From an optimal solution to the MICP formulation, we can recover an optimal solution to the bilinear formulation as follows. For each target $k \in I_k$ and each agent $i \in I_a$, recover $(p_k^i, t_k) = \sum_{e \in E_k^{\text{out}}} (z_e^{p,i}, z_e^t)$. For the depot nodes, use $(p_0^i, t_0) = \sum_{e \in E_0^{\text{out}}} (z_e^{p,i}, z_e^t)$ for all $i \in I_a$, and $(p_{K+1}^i, t_{K+1}) = \sum_{e \in E_{K+1}^{\text{in}}} (z_e'^{p,i}, z_e'^t)$ for all $i \in I_a$. These recovery relations correspond to Constraints (47), (48), (49), (54), (55), and (56).

B. Summary of Formulations

The objective function to be minimized for both AC and DC variants is still the total travel distance of all agents. Let $\mathcal{X}_{\text{com}}^{\text{conic}} := \{a_k^i, b_e, p_u^i, t_u, z_e^{p,i}, z_e'^{p,i}, z_e^t, z_e'^t, \ell_e^i, \bar{\ell}_e^i\}$ denote all common decision variables shared by AC and DC. Let $\mathcal{X}_{\text{AC}}^{\text{conic}} := \mathcal{X}_{\text{com}}^{\text{conic}} \cup \{z_{ij}^k, f_{ij}^k, \ell_{ij,\text{conn}}^k, \bar{\ell}_{ij,\text{conn}}^k\}$ and $\mathcal{X}_{\text{DC}}^{\text{conic}} := \mathcal{X}_{\text{com}}^{\text{conic}} \cup \{z_{uv}^k, f_{uv}^k, \ell_{uv,\text{conn}}^k, \bar{\ell}_{uv,\text{conn}}^k\}$ denote the decision variables for the AC and DC variants, respectively.

The entire conic program for MTSP-CV-AC is as follows:

$$\min_{\mathcal{X}_{\text{AC}}^{\text{conic}}} \quad \sum_{i\in I_a}\sum_{e\in E} \bar{\ell}_e^i$$
$$\text{s.t.} \quad (1),\ (8)-(9),\ (16)-(18),\ (39)-(56),\ (19)-(24).$$

The entire conic program for MTSP-CV-DC is as follows:

$$\min_{\mathcal{X}_{\text{DC}}^{\text{conic}}} \quad \sum_{i\in I_a}\sum_{e\in E} \bar{\ell}_e^i$$
$$\text{s.t.} \quad (1),\ (8)-(9),\ (16)-(18),\ (39)-(56),\ (25)-(30).$$

C. Proof of Validity

In this section, we show the correctness of the MICP formulation by proving the following theorem.

Theorem 1. *The optimum of the MICP formulation is equal to that of the nonconvex bilinear formulation, and an optimal solution to the bilinear formulation can be recovered from an optimal solution to the MICP formulation and vice versa.*

Proof. Let $E_{\text{tour}} := \{e \in E : b_e = 1\}$ be the set of selected edges obtained from a solution to either formulation. The routing constraints in both formulations require the edges in E_{tour} to form a global visit order where each target is visited exactly once. Thus, each target $k \in I_k$ has exactly one selected incoming edge and one selected outgoing edge. The additional constraint (39) in the MICP formulation is redundant. Since every target requires a selected incoming and outgoing edge, the direct edge from the start depot to the end depot can never be selected in any feasible solution of either formulation. Hence both formulations share the same set of feasible tours.

First, consider any inactive edge $e \notin E_{\text{tour}}$. In the bilinear formulation, $b_e = 0$ implies $z_e^{p,i} = z_e'^{p,i} = 0$ and $z_e^t = z_e'^t = 0$. In the MICP formulation, the same result follows from the perspective constraints (44), (45), (50), and (51). Therefore, for inactive edges, both formulations have zero edge states. Consequently, the displacement and cost contribution of inactive edges are zero in both formulations.

Now consider any selected edge $e = (u, v) \in E_{\text{tour}}$. In the bilinear formulation, since $b_e = 1$, $z_e^{p,i} = p_u^i$, $z_e'^{p,i} = p_v^i$, $z_e^t = t_u$, and $z_e'^t = t_v$. In the MICP formulation, Constraints (47), (48), (49), (54), (55), and (56), together with the consistency constraints (46) and (53), ensure that the auxiliary variables on the selected edge recover the same states at nodes u and v, namely $z_e^{p,i} = p_u^i$, $z_e'^{p,i} = p_v^i$, $z_e^t = t_u$, and $z_e'^t = t_v$. Hence, on every selected edge, the MICP formulation satisfies the same relations as the bilinear product constraints.

Since the auxiliary variables take the same value in the two formulations on selected edges and vanish on inactive edges, the displacement constraints $\ell_e^i = z_e'^{p,i} - z_e^{p,i}$ define the same edge displacements in both formulations. The connectivity constraints are unchanged. Thus, any feasible solution of one formulation can be mapped to a feasible solution of the other formulation with the same objective value $\sum_{i\in I_a}\sum_{e\in E}\bar{\ell}_e^i$. □

VI. Experimental Results

We compare our approach against the most relevant baseline method [9] and evaluate our bilinear and conic formulations of the MTSP-CV on its two connectivity scenarios of AC and DC. We first describe the baseline method, test settings, instance generation, and evaluation metrics that are shared by all experiments, and then discuss the results.

A. Mixed-Integer Linear Program (MILP) Baseline

1) Summary of Existing MILP Formulation: The formulation closest to our MTSP-CV-AC variant is a MILP formulation for the so-called Connected Vehicle Routing Problem (Connected VRP) [9]. We summarize this MILP as follows. Instead of using binary variables b_e as in our approach, MILP uses binary variables $X_e^i \in \{0, 1\}$ for every edge $e = (u, v) \in E$ and every agent $i \in I_a$, meaning that edge e is selected and agent i is the visitor of the destination node of that edge. Visit times are formulated by big-M constraints using a "group travel time" $d_e^{\max}$ over all agents for each edge e, an auxiliary variable defined as follows:

$$d_e^{\max} \geq \frac{\|p_u^i - p_v^i\|_1}{v_{\max}} - M\Big(1 - \sum_{i'\in I_a} X_e^{i'}\Big), \qquad (57)$$
$$\forall i \in I_a,\ \forall e = (u, v) \in E.$$

In other words, for every agent, $d_e^{\max}$ is lower-bounded by that agent's travel time on the selected transition, and it equals the maximum single-agent travel time among all agents for a specific edge e.

Then, visit times are formulated by big-M constraints:

$$t_v \geq t_u + d_e^{\max} - M\Big(1 - \sum_{i\in I_a} X_e^i\Big), \quad \forall e = (u, v) \in E. \quad (58)$$

This constraint means that, when transition e is selected, $d_e^{\max}$ makes the next visit wait until every agent can complete its motion between the two consecutive visits of targets. The assigned agent is pinned to the target location by big-M constraints.

The baseline uses Manhattan distances (ℓ_1) and minimizes the makespan $\min t_{K+1}$, i.e., the finish time of all agents. At every target $k \in I_k$, connectivity is enforced by building a directed spanning tree of communication links that connects every agent to one designated sink agent. For every ordered agent pair $i \neq j$, an adjacency binary A_{ij}^k is activated by the big-M range constraint

$$\|p_k^i - p_k^j\|_1 \leq R_c\sqrt{2} + M\big(1 - A_{ij}^k\big), \qquad (59)$$

a tree-link binary $P_{ij}^k \leq A_{ij}^k$ selects one outgoing link per non-sink agent, and flow variables $F_{ij}^k \leq (N-1)P_{ij}^k$ carry one unit from every agent to the sink. Note that the ℓ_1 threshold $R_c\sqrt{2}$

in (59) is a relaxation of the Euclidean communication disk of radius R_c as required in our approach. The baseline has no notion of depot connectivity, so all comparisons are restricted to MTSP-CV-AC when this MILP baseline is involved.

2) Adaption to MTSP-CV-AC: We generate two variants of this MILP method for our MTSP-CV-AC problem. The first variant (MILP-1) keeps the original makespan objective and ℓ_1 distance. Its objective value based on ℓ_1 is not comparable with our distance objective, but its runtime and optimality gap are. The second variant (MILP-2) shares the same objective as ours and uses the Euclidean distance (same as ours) while keeping the rest the same as the original MILP. In MILP-2, the objective becomes the total travel distance of all agents by adding the constraint $c_e^i \geq \text{len}_e^i - M(1 - \sum_j X_e^j)$, where $\text{len}_e^i = \|p_u^i - p_v^i\|_2$ is the Euclidean distance agent i travels on transition $e = (u, v)$ and c_e^i is its counted per-edge cost, so that only selected transitions are counted. The range constraint (59) is replaced by the exact Euclidean disk of Constraint (21), and target time windows are added as bounds on t_k. Actually, due to the use of Euclidean distance, the formulation is no more mixed-integer linear. We still use "MILP-2" to denote this baseline to make the notation consistent with MILP-1.

Remark 2. *The MILP formulation in [9] can admit zero-time subtours. When the travel time on a selected edge is zero, the timing constraint only enforces $t_v \geq t_u$, so it does not force the sequence to make strict progress, which can allow a cycle of targets that is disconnected from the depot route. We fix this issue by requiring a small positive time increase ϵ on every selected edge: $t_v \geq t_u + \epsilon - M_{\text{time}}(1 - \sum_i X_e^i)$, consistent with Constraint* (13) *in our formulations.*

B. Common Test Setup

1) Hardware and Test Settings: All tests were run on a laptop with an Apple M2 Pro CPU (12 cores) and 16 GB RAM. The implementation was in Python 3.10.9, and all formulations were solved using the Gurobi 12.0.1 optimizer with identical Gurobi parameters. The parameters `TimeLimit`, `IntFeasTol`, and `NonConvex` were set to 300 seconds, 10^{-9}, and 2, respectively, while all other parameters were left at their default values.[1]

2) Instance Generation: The instances were defined on a 2D square workspace of fixed size $L_x = L_y = 100$ units, within range $[-50, 50] \times [-50, 50]$, with the start and end depot fixed at its center $(0, 0)$. The time horizon was $t_{\max} = 100$ seconds and all agents shared a maximum speed of $v_{\max} = 10$ units/sec. For each instance, the K targets were assigned fixed positions p_k drawn uniformly at random within the workspace. Each target was given the full time window $[0, t_{\max}]$ by default. In the time-window experiments, each target was assigned a window of a prescribed duration placed uniformly at random within $[0, t_{\max}]$. The communication range R_c was set to 30.

Big-M constraints prefer small M, which leads to tighter continuous relaxations [2]. We therefore set every big-M constant to a tight and valid geometric or temporal bound of the problem. The communication-range constant M_{conn} is set to the length of the diagonal of the bounding box of all nodes, which upper-bounds the distance between any two nodes in the plane. The position constant M_{pos} is set to the larger side of that bounding box, which upper-bounds the difference between any two node coordinates along each axis. The time-ordering constant M_{time} is set to $t_{\max} + 10$, just above the time horizon that upper-bounds every visit-time variable.

The test instances were generated as follows. Starting from a fixed nominal setting, we varied one parameter at a time among the number of targets K, the number of agents N, the communication range R_c, and the time-window duration. The number of targets K and the number of agents N were varied among 3, 4, 5, 6 and 7, separately, the communication range R_c was varied among 10, 20, 30, 45, and 60, and the time-window duration was varied among 15, 30, 50, 75, and 100 seconds. When either K or N was varied, the other parameter was fixed at 4, which covers agent-abundant, balanced, and agent-scarce regimes. For each parameter combination, 10 instances were generated and solved by both formulations, and every experiment was run for both connectivity scenarios. Only feasible and nontrivial instances were retained. Any generated instance that was determined to be infeasible was discarded and regenerated. In addition, an instance admitting a feasible solution with objective value zero was considered trivial and was likewise discarded and regenerated.

3) Evaluation Metrics: To compare the formulations, we report Runtime and Average Gap (%). The same evaluation procedure was applied to both the AC and DC settings. Given each parameter combination and formulation, all 10 instances were solved by the solver. The optimality gap value returned by the solver for an instance is defined as $\frac{|z_P - z_D|}{|z_P|} \times 100\%$, where z_P is the primal, or best feasible, objective function value, and z_D is the dual, or lower-bound, of the objective function value. Runtime denotes the time needed for a formulation to first reduce the optimality gap below 5%. Instances that do not reach 5% within the 300-second time limit are recorded as 300 seconds. Average Gap (%) denotes the average optimality gap remaining after both formulations are run for the full 300-second time limit. For each parameter combination, Runtime is reported as the median of the 10 per-instance runtimes, and Average Gap (%) is reported as the mean of the 10 per-instance gaps.

C. Comparison with the MILP Baseline

We compare our formulations against the two variants (MILP-1 and MILP-2) of the MILP baseline method on MTSP-CV-AC, varying $N \in \{3, 4, 5, 6, 7\}$ with $K = 4$ targets and 10 instances per agent count. We first present the results, and then explain the differences between the baseline and our formulations that cause the observed performance gap.

1) Overall Results: We first compare our bilinear and conic formulations with MILP-2 on MTSP-CV-AC. On every instance that is solved to optimality, MILP-2 attains the same optimal value as our formulations, which confirms that all formulations solve the same problem (Fig. 4). However, MILP-2 is dramatically harder to solve. It proves optimality only on

[1] `NonConvex` is set to 2 to solve the bilinear program. More detail can be found in the documentation of Gurobi.

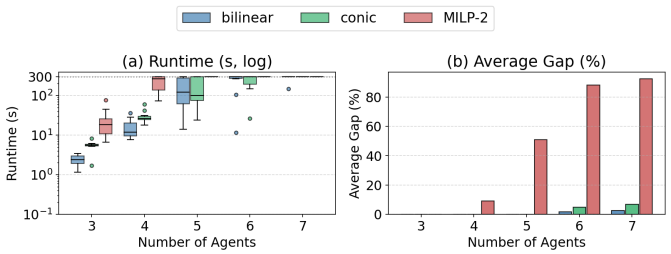


Fig. 4. Results of MTSP-CV-AC comparing the Runtime and Average Gap (%) of our bilinear and conic formulations against a baseline method MILP-2 when varying the number of agents. MILP-2 stops proving optimality beyond $N = 4$ and its average gap grows to 92.5% at $N = 7$, while both of our formulations remain below 6.9%.

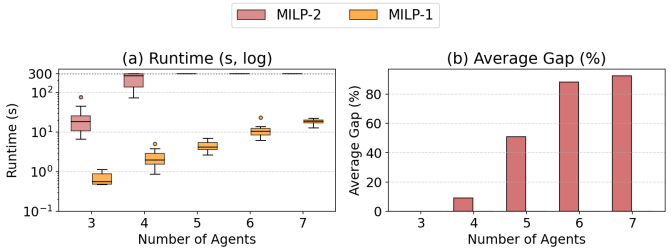


Fig. 5. The two MILP baseline methods on the same instances. MILP-1 (using makespan objective, ℓ_1 communication) solves every instance to optimality within 19 seconds, while MILP-2 (using our total-distance objective and Euclidean communication) times out from $N = 5$ on.

the smallest instances. From $N = 5$ on, every run reaches the 300-second time limit, and the remaining average gap grows from 50.8% at $N = 5$ to 92.5% at $N = 7$, against at most 6.9% for our formulations.

Then, we compare the two baseline variants, as shown in Fig. 5. MILP-1 keeps the original makespan objective and the ℓ_1 distance, so all of its constraints stay linear, and it solves every instance to optimality within 19 seconds, which is not a surprise as the entire formulation is mixed-integer linear.

This observation motivates the two following analyses. First, we evaluate the solution quality of MILP-1 under the original MTSP-CV-AC model, since MILP-1 is fast but uses modified distance and ℓ_1 distance for communication range. Second, we investigate the sources of difficulty in MILP-2, since MILP-2 solves the same problem but is slower than ours.

2) Solution Quality of MILP-1: As aforementioned, MILP-1 minimizes the makespan. Here, we further introduce a variant of MILP-1, denoted by MILP-1', that minimizes the total traveled ℓ_1 distance instead of makespan, with all else being the same. The use of the ℓ_1 distance in the connectivity constraints allows MILP-1 and MILP-1' to accept, under the relaxed threshold of Constraint (59), not only all truly feasible communication links but also links whose true Euclidean length exceeds R_c. We evaluate the effect of the use of ℓ_1 distance on identical instances with $N \in \{3, 4\}$, $K = 4$, and 10 instances per size, and Fig. 6 summarizes the results.

Under the relaxed ℓ_1 communication threshold (Constraint 59), some selected communication edges have true Euclidean length larger than R_c. These edges are accepted by MILP-1 but are invalid for the original MTSP-CV-AC. After

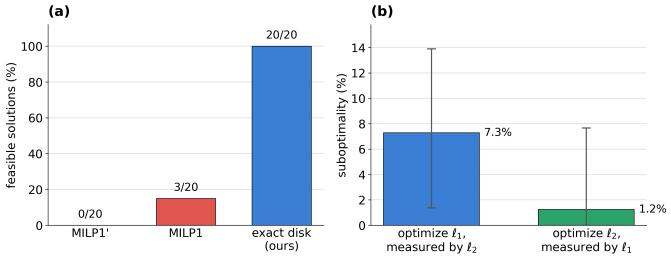


Fig. 6. Solution quality under MILP-1's two metric substitutions. (a) Feasibility: solutions accepted by the relaxed ℓ_1 communication threshold contain links whose true Euclidean length exceeds R_c by up to 41%; after removing such links, only 0 of 20 (distance objective) and 3 of 20 (makespan objective) solutions remain connected (i.e., feasible), whereas our method with exact-disk model is always feasible by construction. (b) Suboptimality: optimizing the ℓ_1 distance produces routes on average 7.3% (up to 13.9%) longer than the Euclidean optimum, whereas optimizing the Euclidean distance stays within 1.2% of the ℓ_1 optimum.

we remove these invalid edges and recheck connectivity under the true Euclidean communication disk, the solution becomes disconnected in 20 of 20 instances for MILP-1' and in 17 of 20 instances for MILP-1 as shown in Fig. 6a. These solutions are therefore not feasible for MTSP-CV-AC.

We then further examine the effect of the distance objective when using ℓ_1 and ℓ_2 norms. We first optimize the routes using the ℓ_1 distance, as in MILP-1', and then evaluate the same solution routes using the true Euclidean distance (ℓ_2 distance). These routes are on average 7.3% longer than the Euclidean optimum, and the worst case is 13.9% longer. In contrast, when we optimize the Euclidean distance and then evaluate the solution using the ℓ_1 distance, the loss is only 1.2% on average. The comparison can be seen in Fig. 6b. This shows that the ℓ_1 objective used by MILP-1' can produce worse routes for the original problem that minimizes the Euclidean distance.

In short, MILP-1 is fast because it solves a relaxed mixed-integer linear problem. Its solutions are usually infeasible under the true communication model based on Euclidean distance, and its ℓ_1 objective can also produce longer routes under the true Euclidean distance.

3) Formulation Differences Between MILP and Ours: MILP-2 and our formulations solve the same problem, and the performance gap comes from how the problem is formulated, which differs in three parts.

The first difference is how the geometry is coupled to the discrete decisions. MILP uses big-M gates to activate timing, motion, target-pinning, and communication constraints such as (58) and (59). Such gates lead to loose continuous relaxation, which makes the lower bound improve slowly during the branch and bound solution process, and eventually leads to a large optimality gap when the runtime budget depletes. Our formulations couple selection and geometry by product variables instead: $\ell_e^i = b_e(p_u^i - p_v^i)$ with the constraint $\bar{\ell}_e^i \geq \|\ell_e^i\|_2$. We refer to this replacement as the *product-bound objective* (obj. for short).

The second difference is the routing representation. MILP-2 duplicates the transition binaries per agent (X_e^i), so the number of routing binaries grows with the number of agents. Our formulations instead use one global visiting sequence shared

TABLE I
ABLATION STUDY OF THE THREE TECHNIQUES (5 INSTANCES PER SIZE, 120-SECOND BUDGET). EACH ROW ADDS THE CORRESPONDING TECHNIQUES TO MILP-2. RT IS MEDIAN RUNTIME (S), GAP MEANS THE REMAINING GAP (%) AT TERMINATION, OPT. MEANS THE INSTANCES SOLVED TO OPTIMALITY, AND # BIN. MEANS THE NUMBER OF BINARY VARIABLES AT $N = 5$.

		$N = 4$			$N = 5$		
Variant	# Bin.	RT	Gap	Opt.	RT	Gap	Opt.
MILP-2	260	120.0	4.2	2/5	120.0	92.5	0/5
+ obj.	260	120.0	9.7	1/5	120.0	18.8	0/5
+ tour	200	120.0	4.0	2/5	120.0	47.0	0/5
+ conn.	140	32.1	0.0	5/5	120.0	37.1	0/5
+ obj. + tour	200	49.0	0.0	5/5	120.0	8.4	0/5
+ obj. + conn.	140	31.3	0.0	5/5	120.0	3.0	1/5
+ tour + conn.	80	36.4	0.0	5/5	120.0	1.0	2/5
+ all three	80	11.3	0.0	5/5	120.0	0.3	2/5

by all agents, with a single binary b_e per edge. We refer to this difference as the *shared tour* (tour for short).

The third difference is the connectivity constraints. At every target, the MILP baseline builds a directed spanning tree over the agents, using two binaries per ordered agent pair (A_{ij}^k and P_{ij}^k, i.e., $2N(N-1)$ binaries per target), which is redundant since there are many equivalent solutions with this formulation. Our formulations use a single undirected link binary z_{ij}^k per unordered pair ($N(N-1)/2$ per target) and certify connectivity with continuous flow variables, which reduces the total number of binary variables. We refer to this replacement as the *undirected connectivity* (conn. for short).

4) Ablation Study: To find out which of the three differences causes the performance gap, we combine our three techniques into MILP-2 one at a time, in pairs, and all together, on identical instances ($N \in \{4, 5\}$, $K = 4$, 5 instances per size, 120-second budget). Table I reports the results.

According to the results, we observe that no single technique explains the gap, since each of the three alone leaves the mean gap at $N = 5$ at 18.8% or more. When using a pair of techniques, the combinations of (obj.+conn.) and (tour+conn.) perform relatively better, both of which include the (conn.) technique. When $N = 5$, (obj.+conn.) reduces the gap down to 3.0%, and (tour+conn.) reduces the gap down to 1.0%, compared with 8.4% for the only pair without the conn. technique. With all three techniques combined, the gap nearly closes to 0.3%, matching our bilinear formulation, while using $3.2\times$ fewer binary variables than MILP-2.

In summary, MILP-2 is slow because several weaknesses take effect at the same time: loose big-M relaxations, routing binaries duplicated across agents, and a connectivity encoding with many equivalent solutions. The gap closes when our three techniques are applied in pairs or all together.

D. Varying the Number of Targets

In this section, we varied the number of targets. We fixed the number of agents at $N = 4$, the communication range at $R_c = 30$, and the time window at $[0, t_{\max}]$, and solved all instances with the number of targets $K = 3, 4, 5, 6$, and 7. The results for this experiment are illustrated in Figs. 7 and 8.

Increasing the number of targets makes the problem more difficult for both formulations. The runtime generally increases as K grows, and the final optimality gap becomes larger for harder instances. As shown in Fig. 7, for MTSP-CV-AC, the conic formulation scales better than the bilinear formulation as the number of targets increases. For $K = 3$ and $K = 4$, both formulations solve the instances quickly, with the conic formulation being faster. When $K = 5$, the conic formulation reduces the median Runtime obviously by 38.7% and the Average Gap by 95.5%. For $K = 6$, both formulations often hit the 300-second limit before reaching the 5% gap threshold, but the conic formulation leaves a much smaller final gap.

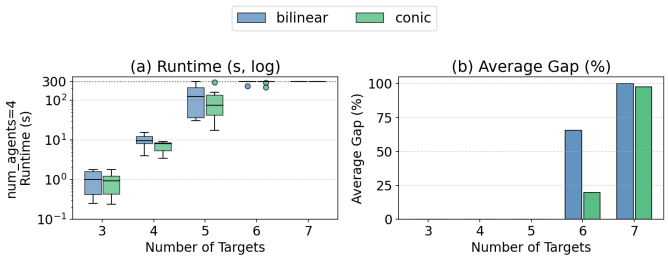


Fig. 7. Results of MTSP-CV-AC comparing the Runtime and Average Gap (%) of the bilinear and conic formulations when varying the number of targets. The Runtime and final optimality gap generally increase with K, while the conic formulation scales more effectively as the instances become larger. It reduces the median Runtime and Average Gap by up to 38.7% and 95.5%, respectively, with the clearest advantage observed for $K \geq 5$.

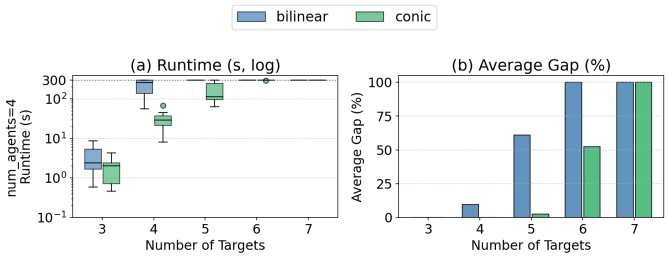


Fig. 8. Results of MTSP-CV-DC comparing the Runtime and Average Gap (%) of the bilinear and conic formulations when varying the number of targets. Both formulations become more difficult to solve as K increases, but the conic formulation scales substantially better. It reduces the median Runtime and Average Gap by up to 86.2% and 99.9%, respectively, with the advantage being most pronounced for $K = 4$ and $K = 5$.

As shown in Fig. 8, the improvement is more pronounced for MTSP-CV-DC. In this setting, the conic formulation reduces the median Runtime by up to 86.2% and the Average Gap by up to 99.9%. The strongest runtime improvement occurs at $K = 4$, where the bilinear formulation already becomes difficult to solve, while the conic formulation still solves all tested instances efficiently. At $K = 5$, the bilinear formulation fails to prove optimality on all tested instances within the time limit, whereas the conic formulation solves 90% of the instances.

E. Varying the Number of Agents

We then varied the number of agents. We fixed the number of targets at $K = 4$, the communication range at $R_c = 30$, and the time window at $[0, t_{\max}]$, and solved all instances with $N = 3, 4, 5, 6$, and 7. The results for this experiment are illustrated in Figs. 9 and 10.

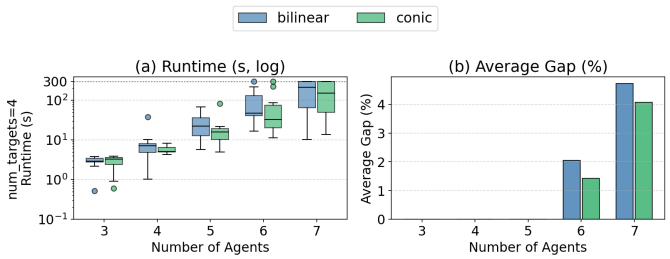


Fig. 9. Results of MTSP-CV-AC comparing the Runtime and Average Gap (%) of the bilinear and conic formulations when varying the number of agents. The Runtime generally increases with N, while the conic formulation exhibits more stable performance as the number of agents grows. It reduces the median Runtime and Average Gap by up to 30.9% and 30.2%, respectively.

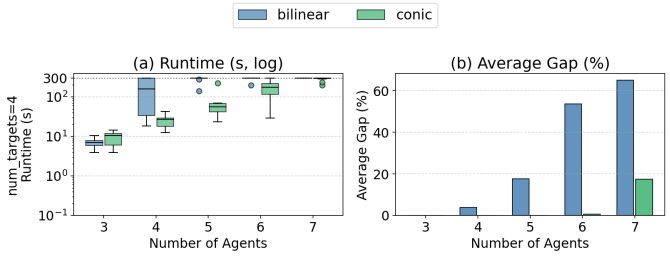


Fig. 10. Results of MTSP-CV-DC comparing the Runtime and Average Gap (%) of the bilinear and conic formulations when varying the number of agents. The problem becomes increasingly difficult as N grows, particularly for the bilinear formulation. The conic formulation reduces the median Runtime and Average Gap by up to 80.8% and 99.97%, respectively, with the largest advantage observed for $N \geq 5$.

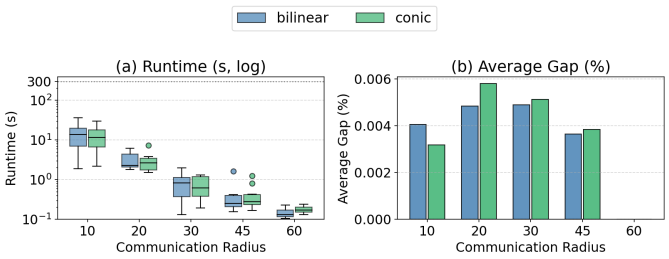


Fig. 11. Results of MTSP-CV-AC comparing the Runtime and Average Gap (%) of the bilinear and conic formulations when varying the communication range. Both formulations become easier to solve as R_c increases and more communication links become feasible. The conic formulation reduces the median Runtime by up to 25.0%, with the largest improvement occurring at $R_c = 30$.

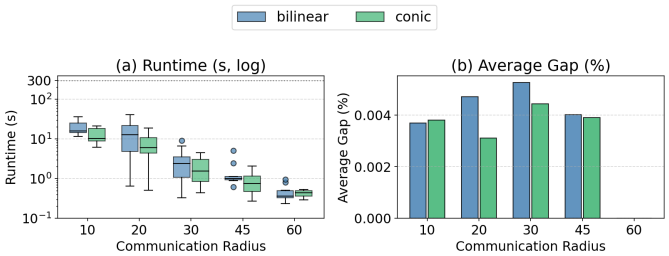


Fig. 12. Results of MTSP-CV-DC comparing the Runtime and Average Gap (%) of the bilinear and conic formulations when varying the communication range. The Runtime generally decreases as R_c increases because the connectivity constraints become less restrictive. The conic formulation consistently reaches the target gap faster, reducing the median Runtime by up to 53.8%, with the largest improvement occurring at $R_c = 20$.

Similarly, the Runtime increases as the number of agents increases for both formulations. For MTSP-CV-AC, the conic formulation shows more stable performance as the number of agents increases. It reduces the median Runtime by up to 30.9% and the Average Gap by up to 30.2%.

For MTSP-CV-DC, the advantage of the conic formulation is much stronger. The conic formulation reduces the median Runtime by up to 80.8% and the Average Gap by up to 99.97%. The biggest improvement occurs at $N = 5$, where the bilinear formulation solves only 20% of the instances to optimality within the time limit, while the conic formulation solves all tested instances. For $N = 6$ and $N = 7$, bilinear formulation can only achieve the Average Gaps of 53.7% and 65.1% respectively within 300s, while the conic formulation is able to achieve 1% and 18%, which is close to the optimal value, showing its ability to solve larger instances.

F. Varying the Communication Range

We then varied the communication range. We fixed the number of agents at $N = 4$, the number of targets at $K = 3$, and the time window at $[0, t_{\max}]$, and solved all instances with $R_c = 10, 20, 30, 45$, and 60. The results for this experiment are illustrated in Figs. 11 and 12.

We can observe that the Runtime decreases as the communication range increases. The communication range controls the tightness of the connectivity constraints. When R_c is small, only nearby agents can connect directly, so the agents must coordinate their positions more carefully at each target-visit event. As R_c increases, more communication edges can become active, and the connectivity constraints become easier to satisfy. In the limit of a very large communication range, the problem approaches a regular MTSP. In this experiment, both formulations have very small optimality gaps ($< 0.01\%$), and the ensuing discussion thus focuses on the runtime.

For MTSP-CV-AC, both formulations solve most instances quickly once the communication range is sufficiently large. The conic formulation reduces the median Runtime by up to 25.0%, with the largest improvement occurring at $R_c = 30$.

For MTSP-CV-DC, the conic formulation reduces the median Runtime by up to 53.8%, with the largest improvement occurring at $R_c = 20$. We observe that the conic formulation has a larger improvement when connectivity constraints are more restrictive. In these settings, the perspective-based conic formulation provides a stronger relaxation and helps the solver close the optimality gap faster.

G. Varying the Time-Window Duration

This section varied the time-window duration. We fixed the number of agents at $N = 4$, the number of targets at $K = 3$, and the communication range at $R_c = 30$, and solved all instances with time-window durations of 15, 30, 50, 75, and 100 seconds. The results are illustrated in Figs. 13 and 14.

The time-window duration affects the size of the continuous feasible region. Shorter time windows restrict the possible visit times more strongly and can reduce the number of feasible

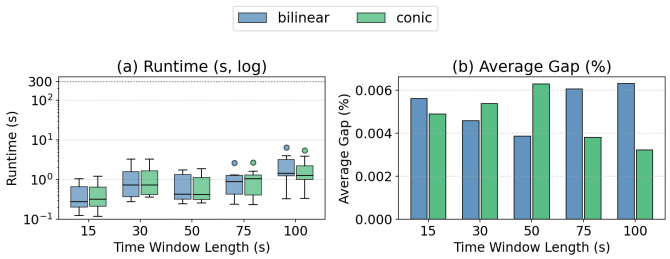


Fig. 13. Results of MTSP-CV-AC comparing the Runtime and Average Gap (%) of the bilinear and conic formulations when varying the time-window duration. The Runtime is only moderately affected by the window duration, and both formulations solve all tested instances with essentially zero final gaps. The conic formulation reduces the median Runtime by up to 14.3%.

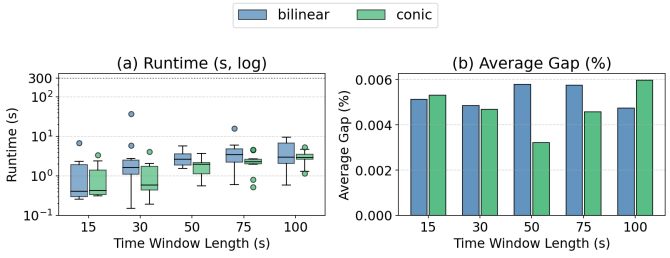


Fig. 14. Results of MTSP-CV-DC comparing the Runtime and Average Gap (%) of the bilinear and conic formulations when varying the time-window duration. The Runtime does not vary monotonically with the window duration because wider windows provide greater scheduling flexibility while enlarging the continuous search space. Both formulations achieve almost zero final gaps, while the conic formulation reduces the median Runtime by up to 62.5%.

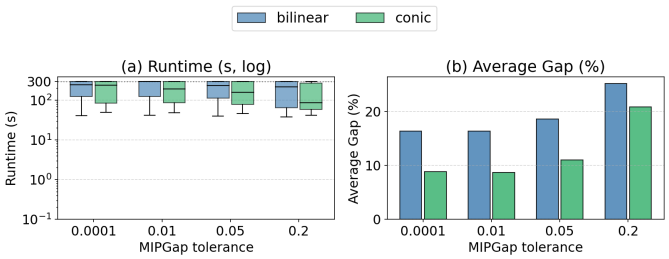


Fig. 15. Results of MTSP-CV-AC comparing the Runtime and Average Gap of the bilinear and conic formulations when varying the target MIP gap tolerance. Relaxing the tolerance generally decreases the Runtime and the Average Gap. The conic formulation reduces the median Runtime by up to 60.6% and the Average Gap by up to 47%

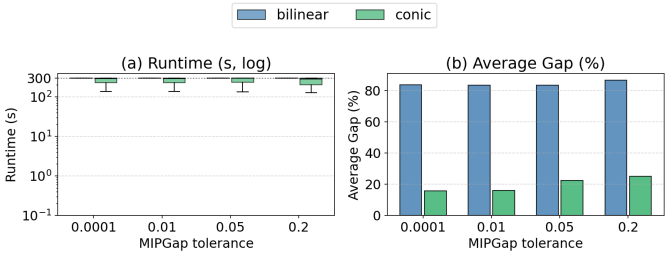


Fig. 16. Results of MTSP-CV-DC comparing the Runtime and Average Gap of the bilinear and conic formulations when varying the target MIP gap tolerance. Within the 300-second time limit, the bilinear formulation cannot reach any of the tested tolerances and retains a large final Average Gap above 83%, whereas the conic formulation attains a much smaller Average Gap. The conic formulation reduces the median Runtime by up to 5.2% and the Average Gap by up to 81%.

temporal orderings. Longer time windows give the solver more flexibility, but they also enlarge the continuous search space and can make it harder to prove optimality. Therefore, larger time windows do not necessarily make the mixed-integer problem easier, even though they relax the feasibility constraints.

Under MTSP-CV-AC, the instances in this experiment are relatively easy for both formulations, and the conic formulation reduces the median Runtime by up to 14.3%. Under MTSP-CV-DC, the conic formulation reduces the median Runtime by up to 62.5%. Under both connectivity scenarios, both formulations solve all tested instances to optimality, resulting in essentially zero final optimality gaps.

H. Varying the MIP Gap Tolerance

In this section, we consider the experiments where the target gap was varied. We selected a challenging instance class with $N = 5$ agents, $K = 5$ targets, and communication range $R_c = 30$, and solved the instances to see how fast these formulations can achieve a gap of 10^{-4}, 1%, 5%, and 20%. The results for this experiment are illustrated in Figs. 15 and 16.

This experiment evaluates how quickly each formulation can reduce the MIP gap below a specified threshold. A smaller MIP gap provides evidence that the best feasible solution found is close to optimal. A loose gap tolerance allows the solver to terminate once the gap reaches a moderate level, whereas a tight tolerance requires further improvement of the feasible solution, the lower bound, or both. Therefore, this experiment shows how quickly each formulation can reach practically useful MIP gap levels.

The conic formulation reaches the specified MIP gap tolerances consistently faster than the bilinear formulation. Under MTSP-CV-AC, it reduces the median Runtime by up to 60.6% and the average final gap by up to 47% (from about 16% to about 9%). The largest runtime improvement occurs under the loosest tolerance, indicating that the tighter relaxation of the conic formulation allows it to reduce the MIP gap to a moderate level much earlier, even when closing the remaining gap is substantially more difficult.

The MTSP-CV-DC instances are more difficult. The bilinear formulation does not reach any of the four target tolerances for any of the ten tested instances within the 300-second time limit, and its average final gap remains above 83%. In contrast, the conic formulation drives the average gap down to about 16% at the 10^{-4} and 1 % tolerances. Because at least half of the runs remain close to or reach the time limit, the reduction in median Runtime is relatively modest, reaching at most 5.2%. However, the reduction of the average gap by up to about 81% (from above 83% to about 16%) demonstrates that the conic formulation can provide meaningful optimality guarantees on instances for which the bilinear formulation is unable to reach even the loosest tested tolerance.

I. Real-Robot Demonstrations

To verify our approach with real robots, we set up a multi-drone system with motion capture and Crazyflie 2.1 micro-drones. Each of the drones is equipped with a downward-pointing LED that projects a light disk indicating the communication range. Several red cones representing the targets are placed on the ground. We executed the solutions of both connectivity scenarios. The paths were generated offline using our approach with Gurobi solver and a post-processing procedure, and then tracked by the robots using a PID controller. The actual positions of the drones are estimated by the motion capture system and used as the input to the PID controller. The entire system uses ROS2 for information exchange, including localization and control commands.

1) Post-Processing for Collision Avoidance: In practice, the robots are often deployed in a large area and there is no need to worry about inter-robot collision avoidance as the robots are wide spread in the workspace. In our experiment, however, due to the small lab space, we have to apply post-processing steps to ensure that the drones are never too close to each other for safety, without changing the visiting order, visit times, or relay structure. Note that these post-processing steps are simply engineering tricks to ensure safety in our experiments, and other more sophisticated trajectory optimization techniques can be applied to intelligently adapt the robot trajectories.

Our post-processing tricks work as follows (Fig. 17). First, note that all robots start and end at the same depot location. For real robots, we placed five take-off and landing pads on a regular pentagon of radius 0.45 m around the depot location and redrew each trajectory within a 0.42 m neighborhood of the depot to start from and return to its assigned pad. Second, since the solver treats drones as point agents, some planned trajectories may pass too close to each other for physical flight. We check the distance between every pair of drones every 0.1 s. Whenever two drones are closer than 0.42 m, which is more than twice the 0.2 m safety radius of the platform, we add equal and opposite smooth offsets to move them apart. Each offset is applied over a ± 1.2 s time window using a cosine bump and has maximum magnitude 0.28 m. We repeat this check-and-separate procedure until no pair is closer than 0.42 m. After these post-processing steps, the minimum inter-drone distance is 0.42 m over each mission, and no reference point moves by more than 0.33 m. If this separation step moves a visiting drone slightly away from its target, we add a local correction of at most 0.30 m within ± 1.2 s of the visit to restore the exact target position.

2) Experimental Settings and Results: We demonstrated one MTSP-CV-AC instance and one MTSP-CV-DC instance, both with $N = 5$ agents and $K = 6$ targets. The solver workspace is $[-60, 60] \times [-60, 60]$, with the depot at the center, $v_{\max} = 5$ units/s, $t_{\max} = 110$ s, and $R_c = 15$ units. Each target has a 6-second time window, and consecutive windows are separated by 14 s. The targets are placed near the workspace boundary and alternate between opposite sides, so the team must repeatedly move across the workspace while maintaining connectivity. Since R_c is small relative to the workspace, the visiting agent cannot connect to the depot or all other agents by a single direct link, making multi-hop relay necessary throughout the mission. Both instances were solved offline by the conic formulation using Gurobi, and the solutions were linearly interpolated into reference trajectories sampled every 0.5 s. For deployment (Fig. 18), we scaled the 120×120-unit workspace by $1/30$ to a 4×4 m flight arena, which maps R_c to 0.5 m, shifted the depot to the arena center, set the flight altitude to 0.8 m, and compressed time by a factor of 5. After this mapping, each run lasts about 22 s, each time window is 1.2 s wide, and the maximum speed is about 0.83 m/s.

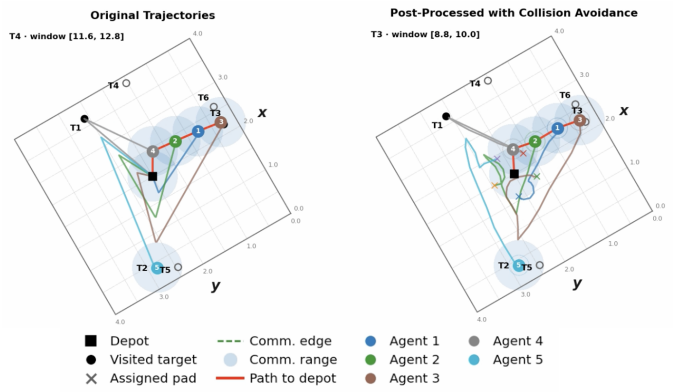


Fig. 17. Reference trajectories of the MTSP-CV-DC demonstration before and after post-processing. The left panel shows the original trajectories from the conic solution, and the right panel shows the post-processed trajectories after assigning separate take-off pads and adding collision-avoidance offsets. The blue disks indicate the communication range, and the red edges indicate the path to depot at the shown target-visit event.

This experiment confirms that the generated trajectories are executable with global connectivity maintained. The LED disks on the floor make the communication range easy to see. In the AC mission, the five drones gather into a connected formation at every visit event, and their disks form an unbroken chain that spans all agents. In the DC mission, only the visiting agent and its relay chain toward the depot need to be in range, and the other agents are free to fly ahead toward their next targets.

VII. Conclusion and Future Work

In this paper, we presented mixed-integer optimization methods for a multi-agent traveling salesman problem, where a team of robots must collectively visit a set of targets while maintaining global connectivity when a target is visited. Regarding the global connectivity, we considered both all-agent connectivity and depot-connectivity variants. We first introduced a bilinear formulation and then developed a conic reformulation based on the auxiliary perspective variables, which removes the nonconvex binary-continuous products and leads to tighter convex relaxation. Numerical results show that our conic formulation leads to tighter convex relaxation and is thus faster to solve or can lead to smaller optimality gaps at termination. Real robot experiments showcase the usage of our method in a lab setting.

For future work, one can further consider other communication factors such as bandwidth and delay. One can also

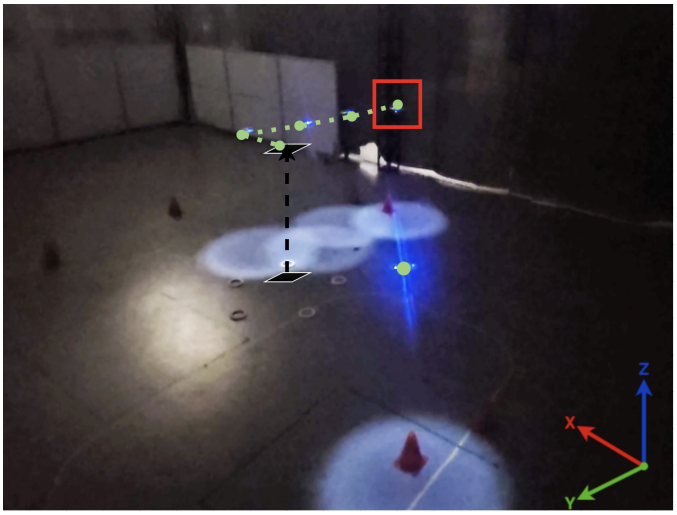


Fig. 18. Snapshot of the physical multi-drone flight for the MTSP-CV-DC demonstration at a target-visit event. The red box highlights the target-visiting drone. The green dots mark the drone positions, and the green dashed lines show the communication links. The black dashed line shows the position of the depot marked on the floor. The LED disks on the floor overlap into a relay chain, illustrating that the drones maintain depot connectivity, while the non-relay drone is free to execute its own mission.

generalize MTSP-CV to include obstacle avoidance or moving targets. Besides, the proposed agent connectivity and depot connectivity and their formulations may inspire research in other MTSP variants, such as persistent monitoring, VRP and team orienteering problems, with connectivity constraints.

References

[1] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, "The traveling salesman problem," in *The Traveling Salesman Problem*. Princeton university press, 2011.

[2] L. A. Wolsey, *Integer Programming*, 2nd ed. Wiley, 2020.

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

[4] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[5] 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.

[6] M. Ostertag, N. Atanasov, and T. Rosing, "Trajectory planning and optimization for minimizing uncertainty in persistent monitoring applications," *Journal of Intelligent & Robotic Systems*, vol. 106, no. 1, p. 2, 2022.

[7] 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.

[8] 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.

[9] S. Kaplan, "Vehicle routing problem with all-time connectivity constraints for the operations of mobile ad hoc networks," in *POMS 27th Annual Conference*, 2016.

[10] K. Helsgaun, "General k-opt submoves for the lin–kernighan tsp heuristic," *Mathematical Programming Computation*, vol. 1, no. 2, pp. 119–163, 2009.

[11] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, Tech. Rep., 1976.

[12] A. G. Philip, Z. Ren, S. Rathinam, and H. Choset, "A mixed-integer conic program for the multi-agent moving-target traveling salesman problem," 2025.

[13] Z. Ren, S. Rathinam, and H. Choset, "CBSS: A new approach for multiagent combinatorial path finding," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2669–2683, 2023.

[14] D. Thakur, M. Likhachev, J. Keller, V. Kumar, V. Dobrokhodov, K. Jones, J. Wurz, and I. Kaminer, "Planning for opportunistic surveillance with multiple robots," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 5750–5757.

[15] J. Keller, D. Thakur, M. Likhachev, J. Gallier, and V. Kumar, "Coordinated path planning for fixed-wing uas conducting persistent surveillance missions," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 17–24, 2016.

[16] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Computers & Industrial Engineering*, vol. 99, pp. 300–313, 2016.

[17] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

[18] T. Marcucci, J. Umenberger, P. Parrilo, and R. Tedrake, "Shortest paths in graphs of convex sets," *SIAM Journal on Optimization*, vol. 34, no. 1, pp. 507–532, 2024.

[19] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science robotics*, vol. 8, no. 84, p. eadf7843, 2023.

[20] 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.

[21] A. Bhat, G. Gutow, B. Vundurthy, Z. Ren, S. Rathinam, and H. Choset, "A complete and bounded-suboptimal algorithm for a moving target traveling salesman problem with obstacles in 3d," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 6132–6138.

[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*, 2013, pp. 3433–3438.

[23] A. Bhat, G. Gutow, B. Vundurthy, Z. Ren, S. Rathinam, and H. Choset, "Parallel, asymptotically optimal algorithms for moving target traveling salesman problems," *IEEE Transactions on Robotics*, 2026.

[24] 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," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 8847–8853.

[25] Z. Ren, A. G. Philip, S. Zhao, S. Rathinam, and H. Choset, "Cp-milp: Mixed integer linear programming for multi-agent motion planning with linear dynamics," *IEEE Robotics and Automation Letters*, vol. 10, no. 12, pp. 12 573–12 579, 2025.

[26] M. M. Zavlanos and G. J. Pappas, "Distributed connectivity control of mobile networks," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1416–1428, 2008.

[27] L. Sabattini, N. Chopra, and C. Secchi, "Decentralized connectivity maintenance for cooperative control of mobile robotic systems," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1411–1423, 2013.

[28] W. Luo, S. Yi, and K. Sycara, "Behavior mixing with minimum global and subgroup connectivity maintenance for large-scale multi-robot systems," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9845–9851.

[29] M. A. Hsieh, A. Cowley, V. Kumar, and C. J. Taylor, "Maintaining network connectivity and performance in robot teams," *Journal of field robotics*, vol. 25, no. 1-2, pp. 111–131, 2008.

[30] G. A. Hollinger and S. Singh, "Multirobot coordination with periodic connectivity: Theory and experiments," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 967–973, 2012.

[31] Y. Liu and Z. Ren, "Multi-robot ergodic trajectory optimization with relaxed periodic connectivity," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 8652–8659.

[32] R. Khodayi-mehr, Y. Kantaros, and M. M. Zavlanos, "Distributed state estimation using intermittently connected robot networks," *IEEE transactions on robotics*, vol. 35, no. 3, pp. 709–724, 2019.

[33] A. R. Da Silva, L. Chaimowicz, T. C. Silva, and M. A. Hsieh, "Communication-constrained multi-robot exploration with intermittent rendezvous," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 3490–3497.

[34] Y. Kantaros, M. Guo, and M. M. Zavlanos, "Temporal logic task planning and intermittent connectivity control of mobile robot networks," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4105–4120, 2019.

[35] Y. Liu and Z. Ren, "A Probabilistic Measure of Multi-Robot Connectivity and Ergodic Optimal Control," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, Jun. 2025.

[36] F. Cladera, Z. Ravichandran, I. D. Miller, M. A. Hsieh, C. Taylor, and V. Kumar, "Enabling large-scale heterogeneous collaboration with opportunistic communications," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 2610–2616.

[37] B. L. Wellman, S. Dawson, J. de Hoog, and M. Anderson, "Using rendezvous to overcome communication limitations in multirobot exploration," in *2011 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2011, pp. 2401–2406.

[38] J. Banfi, N. Basilico, and F. Amigoni, "Multirobot reconnection on graphs: Problem, complexity, and algorithms," *IEEE Transactions on Robotics*, vol. 34, no. 5, pp. 1299–1314, 2018.

[39] M. Guo and M. M. Zavlanos, "Distributed data gathering with buffer constraints and intermittent communication," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 279–284.

[40] T. "Oncan, I. K. Altınel, and G. Laporte, "A comparative analysis of several asymmetric traveling salesman problem formulations," *Computers & Operations Research*, vol. 36, no. 3, pp. 637–654, 2009.

[41] A. J. Orman and H. P. Williams, "A survey of different integer programming formulations of the travelling salesman problem," in *Optimisation, Econometric and Financial Analysis*, ser. Advances in Computational Management Science, E. J. Kontoghiorghes and C. Gatu, Eds. Berlin, Heidelberg: Springer, 2007, vol. 9, pp. 91–104.