# Multi-agent cross-trajectory planning using graph theory

Ziqi Yang and Roberto Tron

***Abstract*—This paper addresses the critical issue of security in multi-robot systems (MRS) when facing threats from adversaries attempting to compromise the robots' control. Such compromises could lead to unauthorized access to forbidden areas, potentially causing harm or disrupting the mission. To tackle this problem, we propose a novel multi-robot planning algorithm that leverages mutual observations between robots as an additional security measure. Our approach guarantees that even in the face of adversarial movement by attackers, compromised robots cannot reach forbidden regions without missing scheduled co-observations. We achieve this by introducing *reachability* constraints into the previously developed continuous trajectory planning algorithm. These constraints ensure an empty intersection between the sets of locations that the agents could potentially reach and the forbidden regions. Furthermore, to enhance system resilience, we introduce the concept of *sub-teams*, where multiple robots form cohesive units along each route, replacing individual robot assignments. This redundancy enables robots to deviate from their assigned sub-teams and join others to perform cross-trajectory co-observations, thereby securing multiple trajectories simultaneously. To efficiently plan cross-trajectory co-observations that maintain security against plan-deviation attacks, we formulate the multi-flow problem on unsecured MRS trajectories. We demonstrate the effectiveness and robustness of our proposed algorithm, which significantly strengthens the security of multi-robot systems in the face of adversarial threats.**

## I. Introduction

Multi-robot systems (MRS) have witnessed increasing adoption in diverse fields, such as warehouse organization, surveillance, forest fire monitoring, and precision agriculture, owing to their capacity for complex task execution through cooperation and coordination [1], [2]. The use of MRS offers numerous advantages, but it also introduces new cybersecurity challenges, including unauthorized access, malicious attacks, and data manipulation [**?**]. Given their distributed nature and reliance on network communication, MRS are particularly vulnerable to cyber threats. In this paper, we address the specific scenario where attackers compromise robots and direct them to restricted regions, potentially leading to the exploitation of confidential information, physical property damages, or even human injuries [**?**]. We refer to these restricted regions as *forbidden regions*, which might contain security-sensitive equipment or human workers.

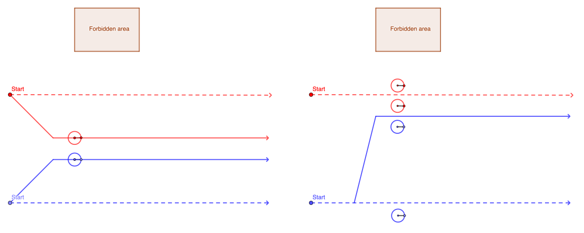
These deliberate deviations, termed *plan-deviation attacks*, have been previously identified and tackled in the literature [3] **zyang** add most recent articles. The work of [3] provide exact solution for grid-world configuration. Using the onboard sensing capabilities, robots can perform inter-robot observations as an additional security measure. Each robot' self-report includes not only its own status, but also additional observations of other robots' status. For example, robot $i$'s self-report may include its observation of robot $j$, "$i$ report its arrival at $v$ and $i$ observes $j$ at location $w$". Similarly, robot $j$ also reports its observations of robot $i$ to the CE. This solution provides paths with an observation schedule such that any attempts by a compromised robot to violate the safety constraints would necessarily break the observation plan and be detected.

When transforming the solution to continuous configuration spaces, additional analysis is needed on the reachability of robots in order for the security of co-observation schedule to hold. More broadly, the problem of solving a path optimization problem with constraints based on the sets of locations that the agents could *potentially* reach, which we call *reachability regions*, has not received attention in the literature. This constraint is formulated using an ellipsoidal bound of the reachability region, and the idea of using an ellipsoidal bound to limit the search space is inspired by the *heuristic sampling domain* introduced by [**?**] in the context of the `RRT`$^*$ path planning algorithm.

We formulate a way to enforce an empty intersection between forbidden regions and ellipsoidal reachability region while optimizing the trajectory, such that if an attacker takes control of the robots, they cannot perform an undetected attack without breaking the co-observation schedule. The idea is inspired by the *heuristic sampling domain* introduced by [**?**] in the context of the `RRT`$^*$ path planning algorithm; in that case, an ellipsoidal bound is used to limit the search space with the initial and goal states fixed. In our case, we use a similar bound to optimize the location of the two states to exclude the forbidden region from the ellipsoidal region. We propose a mathematical formulation of reachability regions that is compatible with the solver from **zyang** cite ADMM as a spatio-temporal constraint.

However, two main challenges have been identified in [**?**] **zyang** previous works. Firstly, finding a co-observation and reachability secured plan may not always be feasible when robots are separated from others by obstacles or forbidden regions, or are located far away from each other. In such case, it is impossible for other robots to get close enough to establish co-observation schedules or find a reachability-secured path. Agent 3 in the Figure **zyang** ref fig:ReachabilitySimulation is a good example, as it required additional security check-point to create secured reachability areas. Secondly, security requirements may come at the cost of overall system performance, as illustrated by the comparison of Figure **zyang** ref fig:trajectories-more-constraint and **zyang** ref fig:ReachabilitySimulation where, after the introduction of security constraints, the top left corner is never explored by any robots. This is particularly concerning given that system performance is a key factor in the decision to use multi-agent systems. These challenges are addressed in Chapter **??**, in

This project is supported by the National Science Foundation grant "CPS: Medium: Collaborative Research: Multiagent Physical Cognition and Control Synthesis Against Cyber Attacks" (Award number 1932162).



(a) Co-observation (b) Cross-trajectory co-observation

Fig. 1: 1a Limited by the co-observation requirement, both red and blue robot follow the co-observation secured routes (solid lines) and abandon the optimal ones (dashed line). 1b Through cross-trajectory co-observations, the blue team sends one robot to follow the red team (solid blue line) and performs co-observation while having the rest of the robots following the optimal trajectory.

order to ensure the effectiveness of planning with reachability and co-observation in securing multi-agent systems.

We propose to form *sub-teams* on each route, and setup additional co-observations both within the sub-team and across different sub-teams. Notice that co-observations across sub-teams (named *cross-trajectory co-observation*) are always preferred when feasible, since they provide better security in potential situations where the entire sub-team is compromised. In this chapter, we consider only the high-level path planning, while the assignment of the duties in the team will be performed dynamically as introduced later in Chapter **??**), making it more difficult for attackers to successfully plan and execute attacks. Cross-trajectory co-observations allows sub-teams to preserve the optimal unsecured trajectories (as shown in Figure 1), as it does not require the entire *sub-team* to maintain a close distance with other teams when co-observation occurs.

*a) Paper contributions:* In this paper, we formulate a way to enforce an empty intersection between forbidden region and reachability regions, such that if an attacker takes control of the robots, they cannot perform an undetected attack by entering forbidden regions and meeting co-observation schedules at the same time. The constraints are formulated using an ellipsoidal bound of the reachability region.

We propose to incorporate redundancy in the form of *sub-teams* of multiple robots in place of individual robots along each route. This allows robots to deviate from their assigned sub-team, and join others to perform cross-trajectory co-observations, thereby securing multiple trajectories. We propose a formulation of the multi-flow problem on unsecured MRS trajectories to plan the cross-trajectory co-observations that can preserve the security against plan-deviation attacks.

## II. Preliminaries

In this section, we review various mathematical concepts that will provide the foundations and context for our novel constraint.

### A. Ellipsoidal reachability constraints

The reachability region is defined as the set of locations $x(t)$ that a robot can reach between two given fixed positions:

**Definition 1.** *The* reachability region *for two waypoints $x(t_1) = x_1$, $x(t_2) = x_2$ is defined as the sets of points $x'$ in the workspace such that there exist a trajectory $x(t)$ where $x(t') = x'$, $t_1 \leq t' \leq t_2$ and $x(t)$ satisfies the velocity constraint $d(x(t), x(t+1)) \leq v_{max}$.*

This region can be analytically bound via an ellipsoid:

**Definition 2.** *The* reachability ellipsoid *is the region $\mathcal{E}(x_1, x_2) = \{\tilde{x} \in \mathbb{R}^n : d(x_1, \tilde{x}) + d(\tilde{x} + x_2) < 2a\}$, where $a = \frac{v_{max}}{2}(t_2 - t_1)$.*

The region $\mathcal{E}(x_1, x_2)$ is an ellipsoid with foci at $x_1, x_2$, center $o_{\mathcal{E}} = \frac{1}{2}(x_1 + x_2)$, and the major radius equal to $a$. Let $c_{\mathcal{E}} = \frac{1}{2}\|x_1 - x_2\| = \|o_{\mathcal{E}} - x_1\|$ be the distance from the center to a foci.

The reachability ellipsoid is an over-approximation of the exact reachability region; the difference between the two is due to the discretization of the trajectory, and the fact that $\mathcal{E}$ does not consider the presence of obstacles.

### B. Rapidly-exploring Random Trees

We use RRT* as a component of the solution [**?**], which is an optimized variant of the rapidly-exploring Random Trees (RRT). As a optimal path planning algorithm, RRT* returns the shortest paths between an initial location and multiple potential goals. We assume that the generated paths can be travelled in both directions (this is used later in our analysis). The basic RRT* algorithm is shown in Algorithm 1. For a given tree $G = (V, E)$, key functions called by the algorithm are

**`Sample`**$(i)$ This function returns independent identically distributed samples from the free configuration space.

**`Nearest`**$(x)$ This function returns the vertex $v \in V$ that is closest in Euclidean distance to the input point $x$.

**`Steer`**$(x, y)$ This function returns a point $z$ that minimize $\|z - y\|$ subject to $\|z - x\| \leq \eta$ for a prespecified $\eta > 0$.

**`ObstacleFree`**$(x, y)$ This function returns whether the line segment between $x$ and $y$ is free of collision.

**`Cost`**$(v)$ This function assigns a non-negative cost (total travel distance in our application) to the unique path from the initial position to $v$ .

**`Parent`**$(v)$ This is a function that maps the vertex $v$ to $v' \in V$ such that $(v', v) \in E$

## III. Problem overview

Assume we have a total of $N$ robots in the system. We define a partition $\mathcal{I} = \cup_p \mathcal{I}_p$ of the robots such that robots in *sub-team* $\mathcal{I}_p$ have the same nominal trajectory. We use the planner introduced in Chapter **??** to generate multi-robot trajectories with $N_p < N$ routes $\{q_p\}_{p=1}^{N_p}$ without reachability and co-observation constraints, where the state $q_p(t), p \in \{0, \dots, N_p\}$ represents the reference position of the $i$-th sub-team at time $t \in \{0, \dots, t_e\}$. Within each sub-team, at least one robot is required to follow the reference trajectory to fulfill the

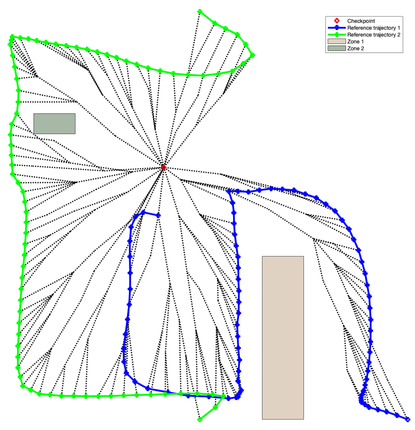


Fig. 2: RRT* used to find trajectory from a point to different trajectories

**Algorithm 1** RRT*($x_{init}, \{x_{goal}\}$)

```
1:  $V_{goal} \leftarrow \{x_{goal}\}$
2:  $V \leftarrow x_{init} \cup V_{goal}$, $E \leftarrow 0$
3:  $i \leftarrow 0$
4:  while $i < N$ do
5:      $G \leftarrow (V, E)$
6:      $x_{rand} \leftarrow \texttt{Sample}(i)$; $i = i + 1$;
7:      $x_{nearest} \leftarrow \texttt{Nearest}(G = (V, E), x_{rand})$;
8:      $x_{new} \leftarrow \texttt{Steer}(x_{nearest}, x_{rand})$;
9:      if $\texttt{ObstacleFree}(x_{nearest}, x_{new})$ then
10:         $V \leftarrow V \cup \{x_{new}\}$;
11:         $x_{min} \leftarrow x_{nearest}$;
12:         for all $x_{near} \in X_{near}$ do
13:             if $\texttt{ObstacleFree}(x_{near}, x_{new})$ then
14:                 $c' \leftarrow \texttt{Cost}(x_{near}) + c(\texttt{Line}(x_{near}, x_{new}))$;
15:                 if $c' < \texttt{Cost}(x_{new})$ then
16:                     $x_{min} \leftarrow x_{near}$;
17:                 end if
18:             end if
19:         end for
20:         for all $x_{near} \in X_{near} \backslash x_{min}$ do
21:             if $\texttt{ObstacleFree}(x_{new}, x_{near})$ and $\texttt{Cost}(x_{near}) > \texttt{Cost}(x_{new} + c(\texttt{Line}(x_{new}, x_{near})))$ then
22:                 $x_{parent} \leftarrow \texttt{Parent}(x_{near})$;
23:                 $E' \leftarrow E' \backslash \{(x_{parent}, x_{near})\}$;
24:                 $E' \leftarrow E' \cup \{(x_{new}, x_{near})\}$
25:             end if
26:         end for
27:     end if
28: end while
29: return $V_{goal}$;
```

required tasks. Meanwhile, the additional robots can switch from one reference trajectory to another in order to perform co-observation with different sub-teams. We refer to the new type of co-observations as *cross-trajectory co-observations*. The goal is to find the cross trajectory co-observation plan to secure an unsecured MRS trajectory as with a minimal number of additional robots.

To solve this problem, we model the planned trajectory $\{q_p\}_{p=1}^{N_p}$ as a directed *checkpoint graph* $G_q = (V_q, E_q)$. Vertices $V_q$ are a key locations in $\{q_p\}$ that are used to perform co-observations. Part of these vertices are *checkpoints* which are selected to guarantee that, if robots deviate and reach forbidden regions, they will miss the scheduled co-observations. Other vertices are selected to connect checkpoints on different trajectories. The set $E_q = E_t \cup E_c$ consists of directed edges of two types, trajectory edges $E_t$ and cross-trajectory edges $E_c$, representing paths which robots can take. The trajectory edges $E_t$ represent the reference trajectories, where each edge connects two waypoints on the trajectory of the same sub-team. The cross-trajectory edges $E_c$ connect vertices on different reference trajectories, with at least one of the vertices being a checkpoint. These edges represent the potential paths that robots can take to travel and perform co-observations between trajectories of different sub-teams. Similar to the work done by [**?**], we show that additional robots in the sub-team can be formulated as flows in the checkpoint graph, transforming the co-observation planning problem into a network multi-flow problem that can be solved using general linear programming techniques as specialized solver.

In this chapter, we present a solution to the cross-trajectory co-observation planning problem. This approach consists of two components: constructing the checkpoint graph based on unsecured multi-robot trajectories, which includes the location of security checkpoints and cross-trajectory paths, and the formulation of the network flow problem to solve for the co-observation plan.

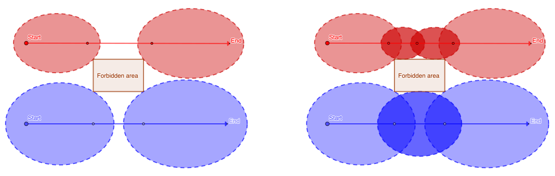
## IV. Checkpoint Graph Construction

In this section, we describe in detail how we define and search for security checkpoints and use the checkpoints to find cross-trajectory edges and construct a directed security graph $G_q = (V_q, E_q)$.

### A. *Checkpoints*

Let $q \in \mathbb{R}^{nmT}$ be the MRS trajectory found in Chapter **??** for $m$ sub-teams with time horizon $T = t_e$, with $q_p(t)$ be the reference trajectory waypoint for sub-team $p$ at time $t$. A checkpoint $v_i = (q_i, t_i)$ is a pair composed of a location $q_i = q_p(t_i)$ and a time $t_i$ representing a co-observation between a sub-team $\mathcal{I}_p$'s member and robots either from the same sub-team or different ones. For convenience, let $\mathcal{I}_{v_i}$ represent the corresponding sub-team $v_i$ belongs to.

Similar to the requirements of co-observations introduced in Chapter **??**, checkpoints $V_p = \{v_{p_1}, \ldots, v_{p_e}\} \subset V_q$ for a reference trajectory $\{q_p(t)\}$ of sub-team $p$ need to guarantee that the reachability region between consecutive checkpoints $\mathcal{E}(q_{p_i}, q_{p_{i+1}}, t_{p_i}, t_{p_{i+1}})$ does not intersect with any of the



(a) (b)

Fig. 3: Checkpoint generation example

forbidden areas. For convenience, we denote the union of all forbidden areas as $F$; then the requirement can be written as $\mathcal{E}(q_{p_i}, q_{p_{i+1}}, t_{p_i}, t_{p_{i+1}}) \cap F = \emptyset$, for every $i$.

We provide a heuristic approach to locate the checkpoints on given trajectories; an optimal solution would likely be NP-hard, while the approach below works well enough for our purpose. We begin by adding the start and end locations $q_p(t_0)$, $q_p(t_e)$ to $V_q$, where $t_0 = 0$, and $t_e$ is the time horizon. Then, we search for the waypoint $q_p(t_1)$ with the largest $t_1 \in \{t_0, \dots, t_2\}$ such that the reachability ellipsoid between $q_p(t_0)$ and $q_p(t_1)$ has no overlap with any of the forbidden areas, i.e. $\mathcal{E}(q_p(t_0), q_p(t_1), t_0, t_1) \cap F = \emptyset$. Simultaneously, we search backward to find $q_p(t_3)$ with the smallest $t_3 \in \{t_1, \dots, t_2\}$ that meets the reachability requirement $\mathcal{E}(q_p(t_2), q_p(t_3), t_2, t_3) \cap F = \emptyset$. $q_p(t_1)$ and $q_p(t_3)$ are then added to $V_q$. Afterwards, we set $t_0 = t_1$, $t_2 = t_3$ and repeat the same process. The search is stopped when $t_0 > t_2$, or when $\mathcal{E}(q_p(t_0), q_p(t_2), t_0, t_2) \cap F = \emptyset$. A toy example is shown in Figure **??**.

## B. *Cross-trajectory edges*

After locating checkpoints $\cup_p V_p$ for all sub-teams, we can search for cross-trajectory edges to connected all checkpoints to trajectories. Cross-trajectory edges represents feasible paths between two references trajectories that allow robots to reach a different reference trajectories and perform co-observations with the corresponding sub-teams. Thus, at least one end of the edges must be a security checkpoint $v_p \in \cup_p V_p$. Additionally, assuming that only one robot is sent at a time between trajectories, the cross-trajectory edges must satisfy the reachability constraints from Chapter **??** to ensure that no deviation to forbidden areas can be performed while switching trajectories.

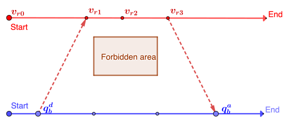


Fig. 4: For checkpoints of the red trajectory, latest departure node $q_b^d$ found for $v_{r1}$ and earliest arrival node $q_b^a$ found for $v_{r3}$.

To find cross-trajectory edges, we base our approach on RRT*. First, we search for feasible paths between each security checkpoint in each reference trajectory and all other trajectories, as shown in Figure 2. Since paths returned by RRT* are optimal and bidirectional, we can analyze the travel time and reachability for each candidate cross-trajectories, and add these edges to the checkpoint graph if they are feasible.

More vigorously, After a fixed iterations of RRT*, we find all possible and close to optimal paths between one security checkpoint $v_p = (q_p, t_p) \in V_p$ for sub-team $\mathcal{I}_p$, and $V_r = \{(q_{r_i}, t_{r_i})\}$ where $q_{r_i} = q_r(t_{r_i})$ are the entire waypoints on a reference trajectories for a different sub-team $\mathcal{I}_r$ for a fixed number of RRT* iterations. We can calculate the minimal travel time $t = \texttt{Cost}(q)/v_{max}$ for a robot to traverse each path. We call all nodes $v_{r_i} \in V_r$ *arrival nodes* if there exist a path between $v_{r_i}$ and $v_p$ with $t_p + t < t_{r_i}$, and the reachability ellipsoid $\mathcal{E}(q_r(t_{r_i}), q_p, t_{r_i}, t_p) \cap F = \emptyset$. Similarly, *departure nodes* are defined for $v_{r_i}$ with $t_p > t_{r_i} + t$ and $\mathcal{E}(q_p, q_r(t_{r_i}), t_p, t_{r_i}) \cap F = \emptyset$. Given a point $v_p \in V_p$, we define the *latest departure node* $q_r(t_r^d)$, and the *earliest arrival node* $q_r(t_r^a)$ as follows:

**Latest departure node** $q_r(t_r^d)$ : is the waypoint with the latest time $t_r^d$ such that a robot from sub-team $\mathcal{I}_r$ can meet with robots in sub-team $\mathcal{I}_p$ at $v_p$.

**Earliest arrival node** $q_r(t_r^a)$ : is waypoint with the earliest time $t_r^a$ such that a robot from sub-team $\mathcal{I}_p$ deviating from $v_p$ can meet with sub-team $\mathcal{I}_r$.

For $v_p \in V_p$, latest departure node $v_d = (q_r(t_r^d), t_r^d)$ and earliest arrival node $v_a = (q_r(t_r^a), t_r^a)$, if feasible, are added to $V_q$. Cross-trajectory edges $(v_p \rightarrow v_a)$ and $(v_d \rightarrow v_p)$ are added to $E_q$. Examples are shown in Figure **??**. Full algorithm is shown in Algorithm **??**.

## C. *In-trajectory edges*

All checkpoints $V_q$ found in Section **??** and **??**, can be divided by the corresponding sub-team and sorted in ascending order of time into $N_p$ subsets $V_p = \{v_0^p, v_1^p, \dots, v_T^p\}$ where $p \in \{1, \dots, N_p\}$. Consecutive edges in the same subset are added to $E_q$ as in-trajectory edges $(v_i^p \rightarrow v_{i+1}^p)$. Example are shown in Figure **??**.

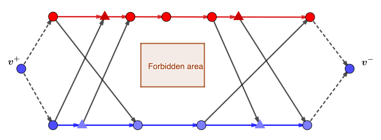


Fig. 5: Example of a two trajectory security graph. Round vertices are checkpoint generated through the heuristic search and triangle vertices are added with the cross-trajectory edges. Additional virtual source $v^+$ and sink $v^-$ is used later in planning problem.

**Algorithm 2** Cross-trajectory edges generation

```
procedure CROSSTRAJECTORYEDGES($V$)
    $V_q \leftarrow V$; $E_q \leftarrow 0$;
    for all $v \in V$ do
        for all $\mathbf{q}_r$ that $\mathcal{I}_r \neq \mathcal{I}_v$ do
            $V_q, E_q \leftarrow$ ADDEDGES($q_v, \mathbf{q}_r, V_q, E_q$);
        end for
    end for
    return $V_q, E_q$
end procedure
procedure ADDEDGES($q_v, \mathbf{q}_r, V_q, E_q$)
    $t_r^d \leftarrow -\infty$; $t_r^a \leftarrow \infty$;
    $\mathcal{V}_{goal} = RRT^*(q_v, \mathbf{q}_r)$;
    for all $q \in \mathcal{V}_{goal}$ that Parent($q$) $\neq \emptyset$ do
        if Cost($q$)$/v_{max} < t_q - t_v$ and $\mathcal{E}(q, v_p, t_q, t_{v_p}) \cup F = \emptyset$ then
            if $t_q < t_r^a$ then
                $t_r^a \leftarrow t_q$;
            end if
        else if Cost($q$)$/v_{max} < t_v - t_q$ and $\mathcal{E}(q, v_p, t_q, t_{v_p}) \cup F = \emptyset$ then
            if $t_q > t_r^d$ then
                $t_r^d \leftarrow t_q$;
            end if
        end if
    end for
    if $t_r^d \neq -\infty$ and $t_r^a \neq \infty$ then
        $v_a \leftarrow (q(t_r^a), t_r^a)$; $v_d \leftarrow (q(t_r^d), t_r^d)$;
        $V_q \leftarrow V_q \cup v_a$; $V_q \leftarrow V_q \cup v_d$;
        $E_q \leftarrow E_q \cup (v_p \rightarrow v_a)$; $E_q \leftarrow E_q \cup (v_d \rightarrow v_p)$
    end if
    return $V_q, E_q$
end procedure
```

## V. Co-observation planning problem

In this section, we describe in detail how to formulate the cross-trajectory planning problem as a network flow problem, and solve it using mixed-integer linear programs (MILP). We assume that there are always *reference robot*s in each sub-team following the reference trajectory. We want to plan the routes for additional *cross-trajectory robots* to perform co-observation with the reference robots.

To formulate the planning problem as a network multi flow problem, we augment the checkpoint graph $G_q$ to a flow graph $G = (V, E)$. The vertices of the new graph are defined as $V = V_q \cup \{v^+, v^-\}$, a *virtual source* $v^+$ and a *virtual sink* node $v^-$ are added to $V$. The edges of the new graph are defined as $E = E_q \cup \{(v_p^e, v_-)\}_p \cup \{(v_+, v_p^0)\}_p$, where directed edges are added to $E$ from $v^+$ to all the start vertices, and from all end vertices to $v^-$, with $v_p^0$ and $v_p^e$ representing the start and end vertices of sub-team $\mathcal{I}_p$.

Assume that all the robots starting from the virtual source $v^+$ and end in the virtual sink $v^-$. Each robot may move from vertex $v_i$ to $v_j$ if $(v_i, v_j) \in E_q$. The path of a robot $k$ can be represented as a flow vector $\mathbf{f}^k = \{f_{ij}^k\}$, where $f_{ij}^k \in \{1, 0\}$ is an indicator variable representing whether robot $k$'s path contains the edge $(v_i, v_j)$. As required by the security guarantee, reference robots must be seen by at least one surveillance robot at every checkpoint. Thus, the planning problem can then be formulated as a path cover problem on $G_q$, i.e., as finding a set of paths $F = [\mathbf{f}_1, \dots, \mathbf{f}_\mathcal{K}]$ for cross-trajectory robots such that every checkpoint in $V_p$ is included in at least one path in $F$. Additionally, to encourage the robots' exchange between different sub-teams, cross-trajectory co-observations are preferred compared to co-observation within the same team. This is achieved with the weights for edges $(v_i, v_j) \in E$ defined as:

$$w_{i,j} = \begin{cases} -w_t & \mathcal{I}_{v_i} = \mathcal{I}_{v_j}, (v_i, v_j) \in E_q \\ w_c & \mathcal{I}_{v_i} \neq \mathcal{I}_{v_j}, (v_i, v_j) \in E_q \\ 0 & (v_i, v_j) \in E / E_q \end{cases} \tag{1}$$

where $w_c > w_t$.

With the formulation above, the planning problem can be written as an optimization problem, where we want to balance between the co-observation performance and total number of flows (cross-trajectory robots) needed. For convenience, we use $(ij)$ to represent the edge $(v_i, v_j)$, and $(+i)$ to represent the edge $(v^+, v_i)$.

$$\min \quad \sum_{k}^{\mathcal{K}} \sum_{(+i) \in E} f_{+i}^k - \rho \sum_{k}^{\mathcal{K}} \sum_{(ij) \in E} w_{ij} f_{ij}^k \tag{2a}$$

$$\text{subject to} \quad \sum_{\{h:(hi) \in E\}} f_{hi}^k = \sum_{\{j:(ij) \in E\}} f_{ij}^k, \qquad \forall k, \forall v \in V_q / \{v^+, v^- \tag{2b}$$

$$\sum_{k}^{\mathcal{K}} \sum_{\{i:(ij) \in E\}} f_{ij}^k \geq 1 \qquad \forall v_j \in \{V_p^s \tag{2c}$$

$$f_{ij}^k \in \{0, 1\} \qquad \forall (ij) \in E \tag{2d}$$

The first term in **??** is total outgoing flow from the source $v^+$ while the second term is the total cost of all flows which represents the overall co-observation performance (defined as the total number of cross-trajectory edges taken by all flows beyond the regular trajectory edges). The constant $\rho$ is a penalty parameter manually elected to balance between two terms in the cost function. The constraint (**??**) is the flow conservation constraint, which ensures that the amount of flow entering and leaving a given node $v$ is equal (except for $v^+$ and $v^-$). The constratin (**??**) is the flow coverage constraints, which ensures that all security checkpoints $\{V_p^s\}$ have been visited by at least one flow (robot). Since the security graph $G_q$ is acyclic, this problem is in complexity class $P$, and can be solved in polynominal time [**?**].

## VI. Co-observation performance

Notice that problem (**??**) is guaranteed to have a solution for $\mathcal{K} = N_p$ where all $\mathcal{K}$ robots follows the reference trajectory ($f_{ij}^k = 1, \forall \mathcal{I}_{v_i} = \mathcal{I}_{v_j} = k$). Additionally, for a fixed number of robots $\mathcal{K} \geq N_p$, it is possible to have a subset of flows $F_\mathcal{E} \in F$ that is empty, i.e. $f_{ij} = 0, \forall (v_i, v_j) \in E$, $f \in F_\mathcal{E}$,

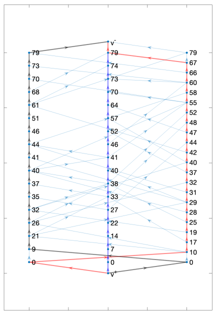


Fig. 6: Security Graph and result for 3 flows

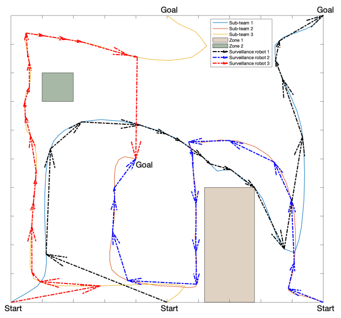


Fig. 7: Result of 3 surveillance agents' plan in workspace

these flow will not increase the cost and will not have any impact on the solution. The trade-off between the number of surveillance robots used, and the overall security performance is a critical consideration in this case. Increasing the number of robots generally contributes to improved security by enabling more cross-trajectory co-observations. However, adding more robots can lead to diminishing returns, as the benefits gained from additional robots may be counterbalanced by increased complexity and coordination challenges. To capture this trade-off, the penalty parameter $\rho$ in the cost function **??** allows us to fine-tune the balance between the number of robots employed and the desired task performance. By optimizing this parameter, we can strike a suitable equilibrium between security enhancement and operational efficiency.

To find such tradeoff, we employ an iterative approach. We start with $\mathcal{K} = 1$ and gradually increase it until $F$ contains a empty flow. This indicates that the performance can no longer be improved by adding more robots. To ensure that the value of $\mathcal{K}$ does not grow unbounded, penalty parameter $\rho$ need to ensure that the second term for a single flow in (**??**) $\rho \sum_{(ij)\in E} w_{ij} f_{ij}^{k}$ is always smaller than 1.

## VII. Result and simulation

In this section, we test the proposed method starting from the results in Figure **??**, where three trajectories are provided for the map exploration task with no security related constraints (co-observation schedule and reachability). We use the same environment and dynamic setup, where we have a $10m \times 10m$ task space, two forbidden regions (two rectangles in **??**) and robots with a max velocity of $0.5m/dt$.

We start by adding a total of $\mathcal{K} = 3$ surveillance robots into the workspace using the parameters $w_c = 10$, $w_t = 1$ and $\rho = 0.01$. The three trajectories are transformed into the graph $G$ shown in Figure **??**, where the number on each vertex $v_i$ represents the corresponding time $t_i$. Vertices in each column belong to the same trajectory, and edges across different columns are cross-trajectory edges.

The flows derived from the solution of the optimization problem (**??**) are highlighted in the graph. The planning result in the workspace is shown in **??** as dash-dotted arrows with the same color used for each flow in **??**. Notice that flows represent the robot co-observation plan instead of actual robots, such that the red flow, for example, shows that the sub-team 2 is expecting a robot from sub-team 1, departing at $t = 0$ and arriving at $t = 10$, and needs to send a robot (not necessarily the same one received from sub-team 1) to sub-team 3, departing at $t = 67$ and arriving at $t = 79$. This plan is not ideal in terms of our security criteria since co-observations happens between the same pair of surveillance robot and sub-team for the majority of the time with only three cross-trajectory edges that are covered.

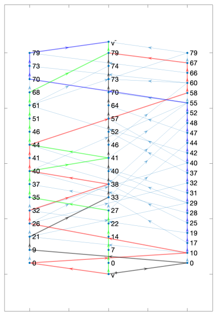


Fig. 8: Security Graph and result for 4 flows.

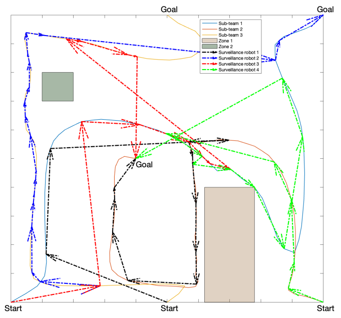


Fig. 9: Result of 3 surveillance agents' plan in work space.

For the case where $\mathcal{K} = 4$, the results are shown in Figure **??**-**??**. With one additional robot added, there are significant increase in the total cross-trajectory edges covered, and sub-teams performs co-observation with different robots during the task period.

If we further increase $\mathcal{K} > 4$, we do not get a better result; instead, the planner will generate 4 flows with the rest $\mathcal{K} - 4$ flows empty.

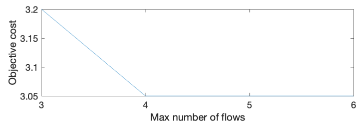


Fig. 10: Plan reach optimality when $\mathcal{K} = 4$, further increase of $\mathcal{K}$ does not increase the cost of objectives.

## VIII. Summary

In this chapter, we provide a method to enhance the security of a multi-robot system without sacrificing the performance. This is done by introducing additional robots to perform cross-trajectory co-observations by traveling between different trajectories. We model the unsecured multi-robot trajectories as a checkpoint graph by identifying checkpoints that requires observation and cross-trajectory paths that can safely access the checkpoints from different trajectories. We have shown that the co-observation planning problem across different trajectories can be transformed into a multi flow problem on the graph, and that we can find the minimal number of robot needed to finish the co-observation task.

## References

[1] Gonzalo Pajares. Overview and current status of remote sensing applications based on unmanned aerial vehicles (uavs). *Photogrammetric Engineering & Remote Sensing*, 81(4):281–330, 2015.

[2] Brian J Julian, Michael Angermann, Mac Schwager, and Daniela Rus. Distributed robotic sensor networks: An information-theoretic approach. *The International Journal of Robotics Research*, 31(10):1134–1154, 2012.

[3] Kacper Wardega, Roberto Tron, and Wenchao Li. Resilience of multi-robot systems to physical masquerade attacks. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 120–125. IEEE, 2019.

[4] Ziqi Yang and Roberto Tron. Multi-agent trajectory optimization against plan-deviation attacks using co-observations and reachability constraints. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 241–247. IEEE, 2021.

[5] Ziqi Yang and Roberto Tron. Multi-agent path planning under observation schedule constraints. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6990–6997. IEEE, 2020.

[6] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. Multi-robot task allocation: A review of the state-of-the-art. *Cooperative robots and sensor networks 2015*, pages 31–51, 2015.

[7] Brian Coltin and Manuela Veloso. Mobile robot task allocation in hybrid wireless sensor networks. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2932–2937. IEEE, 2010.

[8] Chun Liu and Andreas Kroll. A centralized multi-robot task allocation for industrial plant inspection by using a* and genetic algorithms. In *Artificial Intelligence and Soft Computing: 11th International Conference, ICAISC 2012, Zakopane, Poland, April 29-May 3, 2012, Proceedings, Part II 11*, pages 466–474. Springer, 2012.

[9] Yan Jin, Ali A Minai, and Marios M Polycarpou. Cooperative real-time search and task allocation in uav teams. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 1, pages 7–12. IEEE, 2003.

[10] Félix Quinton, Christophe Grand, and Charles Lesire. Market approaches to the multi-robot task allocation problem: a survey. *Journal of Intelligent & Robotic Systems*, 107(2):29, 2023.

[11] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438, 2012.

[12] Mehdi Alighanbari and Jonathan P How. Decentralized task assignment for unmanned aerial vehicles. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 5668–5673. IEEE, 2005.

[13] Dany Dionne and Camille A Rabbath. Multi-uav decentralized task allocation with intermittent communications: The dtc algorithm. In *2007 American Control Conference*, pages 5406–5411. IEEE, 2007.

[14] Sanem Sariel and Tucker Balch. Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments. In *Proceedings of the AAAI-05 Workshop on Integrating Planning into Scheduling*, pages 27–33. sn, 2005.

[15] William E Walsh and Michael P Wellman. A market protocol for decentralized task allocation. In *Proceedings International Conference on Multi Agent Systems (Cat. No. 98EX160)*, pages 325–332. IEEE, 1998.

[16] PB Sujit and Randy Beard. Distributed sequential auctions for multiple uav task allocation. In *2007 American Control Conference*, pages 3955–3960. IEEE, 2007.

[17] Han-Lim Choi, Luc Brunet, and Jonathan P How. Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, 25(4):912–926, 2009.

[18] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[19] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. 3(1):1–122, 2011.

[20] Dimitri P Bertsekas and John N Tsitsiklis. Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization*, 10(3):627–642, 2000.

[21] Xiangru Xu, Paulo Tabuada, Jessy W Grizzle, and Aaron D Ames. Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 48(27):54–61, 2015.

[22] Lars Lindemann and Dimos V Dimarogonas. Control barrier functions for signal temporal logic tasks. *IEEE control systems letters*, 3:96–101, 2018.