Dynamic Adversarial Resource Allocation: The dDAB Game

Yue Guan[1*], Daigo Shishika[2*], Michael Dorothy[3], Jason R. Marden[4], Panagiotis Tsiotras[2], and Vijay Kumar[5]

***Abstract*—This work introduces the dynamic Defender-Attacker Blotto (dDAB) game, extending the classical static Blotto game to a dynamic resource allocation setting over graphs. In the dDAB game, a team of Defender robots is required to maintain numerical superiority against a team of Attacker robots across a set of key nodes in a connected graph. The engagement unfolds as a discrete-time game, where each team reallocates its robots in turn, with individual robots allowed to move at most one hop per time step. The primary goal is to determine the necessary and sufficient number of Defender robots required to guarantee sustained defense, along with the corresponding strategies. To address the central challenge arising from graph-constrained robot reallocation, we conduct a reachability analysis, starting with simplified settings where Attacker robots act as a single cohesive group. We then extend the framework to allow Attacker robots to split and merge arbitrarily, and construct Defender strategies using superposition principles. A set-based dynamic programming algorithm is developed to compute the optimal strategies, as well as the minimum number of Defender robots to ensure successful defense. The effectiveness of our approach is demonstrated through numerical simulations and hardware experiments on a multi-robot testbed.**

***Index Terms*—Multi-robot systems, dynamic games, adversarial games, resource allocation.**

I. Introduction

Deploying resources such as robots, sensors, or supplies to the right locations at the right time is a fundamental challenge in multi-agent systems, commonly studied as a multi-robot task allocation (MRTA) problem [1], [2]. In real-world scenarios, MRTA must account for dynamically changing environments. Much of the literature focuses on time-varying demands arising from stochastic or uncertain processes, enabling robust allocation strategies in domains such as wireless networks [3], ride-sharing [4], power grids [5], and cloud computing [6]. In contrast, dynamics driven by intelligent adversaries have received comparatively less attention, despite their prevalence in applications such as military operations [7], network cyber-attacks [8], and power grid defenses [9]. This gap motivates the study of *adversarial resource allocation*, where the decision-maker must anticipate and counteract strategic opponents.

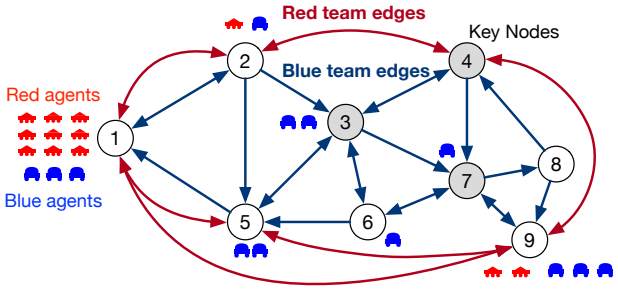

Fig. 1. Illustration of the adversarial resource allocation problem.

Adversarial resource allocation problems are often modeled as Colonel Blotto (CB) games, where two players (instantaneously) distribute limited resources across multiple locations/battlefields to maximize territorial control. The game outcome is determined by numerical superiority at each location. Classical CB games assume instantaneous allocation, which is unrealistic for embodied agents such as robots. Moreover, these games are typically modeled as one-shot interactions, rendering them insufficient for dynamic, real-world scenarios.

To address these limitations, we formulate the dynamic Defender-Attacker Blotto (dDAB) game, an adversarial resource allocation problem on graphs. Nodes of the graph represent locations, and edges define robot traversability, as illustrated in Fig. 1. The game is turn-based and is played between the Defender and the Attacker, who reallocate their robots at each step. Unlike the classical Blotto game, where resources (robots) can be distributed instantaneously, the dDAB formulation requires each robot to traverse through the graph, moving one hop per time step. Consequently, both players can update their allocations only through sequential motion rather than instantaneous deployment. The objective of the Defender is to defend a set of key nodes by maintaining its numerical superiority over the Attacker robots at these nodes. If the Attacker robots outnumber the Defender robots at any key node, the Defender loses the game. Consequently, adversarial presence imposes a *hard constraint* on the Defender's allocation, thereby requiring the Defender to dynamically reallocate its robots in real time. This framework also captures other safety-critical applications such as resilient power grid defense [10] and wildfire surveillance [11], which exhibit adversarial or rapidly changing demands.

A. Related Work

Population dynamics on graphs: Early work on distributed resource allocation over graphs developed stochastic control laws that drive a population of robots toward distributions

We gratefully acknowledge the support of ARL grant DCIST CRA W911NF-17-2-0181. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Army, Department of Defense, or the United States Government.

[1]Yue Guan and Panagiotis Tsiotras are with the School of Aerospace Engineering at Georgia Institute of Technology, Atlanta, GA, USA. {yguan44, tsiotras}@gatech.edu

[2]Daigo Shishika is with the Department of Mechanical Engineering at George Mason University, Fairfax, VA, USA. dshishik@gmu.edu

[3]Michael Dorothy is with DEVCOM Army Research Laboratory, Adelphi, MD, USA. michael.r.dorothy.civ@army.mil

[4]Jason R. Marden is with with the University of California, Santa Barbara, CA, USA. jrmarden@ece.ucsb.edu

[5]Vijay Kumar is with GRASP Laboratory at the University of Pennsylvania, Philadelphia, PA, USA. kumar@seas.upenn.edu

*The first two authors contributed equally as co-first authors.

satisfying *static* demands [12]. These results were later extended to accommodate heterogeneous teams with diverse task requirements [13], [14]. However, these efforts primarily target either fixed terminal allocations [12], [13] or predetermined time-varying demands [2], and do not address the transient behaviors required to respond to dynamically evolving conditions. In contrast, our framework introduces a dynamic feedback mechanism that updates allocations in real time in response to adversarial actions, albeit within a centralized framework. Conceptually, our approach functions as an *outer loop* that continuously revises the desired allocation based on the observed adversarial behavior. This high-level allocation can then be executed by an *inner loop* of distributed control laws, such as those proposed in [12].

Dynamic resource/task allocation: The dynamic aspects of resource allocation have been explored through a variety of approaches. Scheduling-based formulations consider tasks that must be completed in a prescribed sequence [15], [16]. Adaptation mechanisms, such as market-based optimizers, have been proposed to reassign tasks in response to robot failures [15]. In graph-based environments, distributed resource allocation has also been studied using local adaptation mechanisms [16], where population dynamics are modulated by individual agents adjusting their behavior based on local sensing. While these works enable scalable within-team coordination, their adaptation strategies are inherently reactive and do not account for the anticipation of future events or adversarial actions. In contrast, our work focuses on *inter-team* strategic interactions (Defender vs. Attacker), where both teams select their allocations based on the opponent's *anticipated* response.

Colonel Blotto Games: Colonel Blotto (CB) games provide a classical *static* model for competitive allocation across multiple locations [17]–[20]. Variants of the CB game include asymmetric budget [17], asymmetric information [21], etc. However, most existing formulations consider static games and assume instantaneous allocation, ignoring the dynamics of transporting resources. Although recent works extend CB games to dynamic settings [22]–[24], they focus on budget allocation over time and still assume instantaneous allocation at each time step.

In contrast, the dynamic Defender–Attacker Blotto (dDAB) game explicitly models sequential resource movement within an environment. The dDAB game was first introduced in [25], where a sampling-based algorithm was proposed to establish *sufficient* conditions for the Attacker's victory. This work advances this line of research by deriving *necessary and sufficient* conditions for successful defense via a set-based dynamic programming framework, together with additional theoretical insights and numerical tools described in the subsequent section on contributions.

B. Contributions

Our formulation yields feedback strategies that reallocate resources based on the evolving system state—namely, the current locations of the Attacker and the Defender robots. Unlike prior work that yields open-loop strategies against known demand [2], [12], we address the adversarial aspect of the proposed dDAB game by employing a novel reachability-based analysis. Such approach leads to feedback strategies that reallocate resources with all possible next allocations of the opposing team in mind, thus providing worst-case guarantees.

To handle the game's temporal structure, we develop a set-based dynamic programming algorithm that recursively computes *k-step safe sets*—Defender allocations that are *necessary and sufficient* to maintain defense for k time steps against any Attacker strategy. The proposed algorithm explicitly incorporates the traversability constraints of the robots, and exploits the geometric properties of the safe sets for computational efficiency. We further mitigate the curse of dimensionality by first analyzing no-splitting Attacker strategies, then generalizing to arbitrary strategies via a subteam superposition approach.

Our analysis leads to three key results:

1) Identification of the critical amount of Defender resources that is *necessary and sufficient* for guaranteed defense over a given graph.
2) Synthesis of *feedback strategies* that ensure successful defense against any Attacker strategy.
3) Formal proof that the Attacker gains no advantage by splitting its robots into subteams, along with Attacker strategies that guarantee its earliest victory when defense is infeasible.

These results provide practical guidance for designing and deploying Defender robotic systems capable of provably securing a graph against intelligent Attackers, with explicit guarantees on the required number of robots.

The remainder of the paper is organized as follows: Section II introduces the formulation of the dDAB game. Section III and Section IV characterize the Defender strategies against no-splitting Attacker team, leveraging the k-step safe sets (Q-sets). Section V generalizes the results to arbitrary Attacker strategies via subteam superposition, and shows that Attacker gains no benefit from splitting. Section VI presents a numerical algorithm for computing Q-sets and synthesizing strategies. Section VII validates our approach through numerical simulations and hardware implementations. Section VIII concludes and discusses future directions. For the sake of readability and conciseness all essential proofs are delegated in the appendices.

II. Problem Formulation

The dynamic Defender-Attacker Blotto (dDAB) game is played between two players: the Defender and the Attacker. The environment is modeled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the N nodes represent locations ($N = |\mathcal{V}|$), and the directed edges represent the traversability among those locations. We assume that $\mathcal{G}$ is strongly connected [12] to avoid degenerate cases. Strong connectivity implies that every node is reachable from any other node. For notational simplicity, we assume that the two players share the same graph, but our analysis easily extends to the case where the two players have different edge sets.

To capture the connectivity among the nodes, we define the graph adjacency matrix $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ as follows:

$$[A]_{ji} = \begin{cases} 1, & \text{if } (i,j) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases}$$

The *out-degree* of node i is denoted as $d_i = \sum_j [A]_{ji}$, and the set of *out-neighbors* is denoted as $\mathcal{N}_i = \{j \in \mathcal{V} | (i,j) \in \mathcal{E}\}$.

The total number of robots for the Defender and the Attacker are denoted by $X \in \mathbb{R}_{>0}$ and $Y \in \mathbb{R}_{>0}$, respectively. For some time horizon T, the allocation of the Defender's resources over the graph at time $t = 0, 1, \ldots, T$ is denoted by the state (allocation) vector $\mathbf{x}_t \in \mathbb{R}^{|\mathcal{V}|}$, which lies on a scaled simplex, such that $[\mathbf{x}_t]_i \geq 0$ and $\sum_{i \in \mathcal{V}} [\mathbf{x}_t]_i = X$. The state (allocation) vector $\mathbf{y}_t \in \mathbb{R}^{|\mathcal{V}|}$ for the Attacker is defined similarly. We use Δ_X and Δ_Y to denote the state space of the Defender and the Attacker respectively. Note that we consider divisible continuous resources in this work: i.e., $\mathbf{x}_t$ and $\mathbf{y}_t$ are continuous variables. Such an assumption on the state vector simplifies the presentation of our analysis. Nevertheless, we will later show that our algorithms naturally accommodate states that only take discrete values as in robotic systems.

A. Dynamics

The major difference from the original Colonel Blotto game is that the dDAB game is played over multiple time steps, and that the allocation states evolve according to the following discrete-time dynamics:

$$\mathbf{x}_{t+1} = K_t \mathbf{x}_t \quad \text{and} \quad \mathbf{y}_{t+1} = F_t \mathbf{y}_t, \tag{1}$$

where K_t and F_t represent the *resource transition matrices* for the Defender and the Attacker, respectively. These matrices represent the reallocation of robots executed by the players. For example, the entry $[K_t]_{ji}$ denotes the fraction of Defender robots on node i to be transferred to node j. An action K_t of the Defender is admissible if and only if it satisfies the following graph-induced *linear* constraints:

$$\mathbf{1}^\top K_t = \mathbf{1}, \tag{2}$$
$$[K_t]_{ij} \geq 0, \qquad \forall\, i, j \in \mathcal{V}, \tag{3}$$
$$[K_t]_{ij} = 0, \qquad \text{if } [A]_{ij} = 0. \tag{4}$$

We denote the admissible set for the matrices K_t as $\mathcal{K}$, which depends only on the underlying graph $\mathcal{G}$ and is time-invariant. The matrix F_t for the Attacker satisfies similar constraints, and we use $\mathcal{F}$ to denote its admissible set.[1]

B. Terminal conditions & Sequential Actions

Similar to the Colonel Blotto game [26], the engagement at each location is modeled solely based on the number of robots. However, we evaluate the game outcome on a subset of nodes $\mathcal{V}_{\text{key}} \subseteq \mathcal{V}$, which we refer to as the key nodes[2]. Specifically, the Defender successfully *guards* a key node by allocating at least as many robots as the Attacker does, whereas the Attacker *breaches* a key node by allocating more than what the Defender does. For the dDAB game, the Defender wants to prevent the Attacker from breaching any key node. In this work, we mainly focus on a finite horizon T. The game terminates with the Attacker's victory at the earliest time instance $t \in \{0, \ldots, T\}$ at which

$$\exists\, i \in \mathcal{V}_{\text{key}}, \text{s.t.} [\mathbf{y}_t]_i > [\mathbf{x}_t]_i. \tag{5}$$

The Defender wins the game if it can prevent the Attacker from achieving condition (5) for all $t \in \{0, \ldots, T\}$. If the Defender can prevent (5) for all time horizons $T \geq 1$, we say that the Defender can defend indefinitely.

The key node formulation provides a generalization to the prior work [25], in which the Defender needs to defend *all* nodes in the graph.

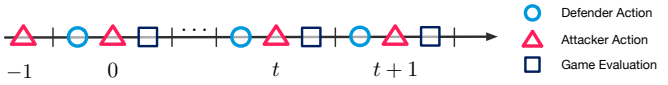

Fig. 2. Sequence of events at every time step of the dDAB game. The Defender first moves its resources based on the previous Attacker allocation. The Attacker then observes the new Defender allocation and reallocates. Finally, the game outcome at this time step is evaluated after the Attacker's move.

C. Information Structure

We assume that the players make decisions in sequence. Specifically, the Defender acts first; then the Attacker acts next, i.e., the Attacker selects its action after observing how the Defender allocated its resources. The game outcome is evaluated after the Attacker's move. To avoid the degenerate scenario where the Attacker wins immediately in the first time step, we let the Attacker specify its initial allocation $\mathbf{y}_{-1}$, then let the Defender freely pick its distribution $\mathbf{x}_0$ after observing $\mathbf{y}_{-1}$. The timeline of the dDAB game is presented in Fig. 2. In a realistic scenario where the two players make simultaneous actions, our problem formulation corresponds to a worst-case scenario for the Defender. Importantly, our setting accommodates state feedback strategies in contrast to previous results with constant action (transition) matrices [12], [13].

Finally, we consider centralized strategies in this work. Specifically, the Defender (Attacker) serves as a coordinator, who decides the next allocation for all of its robots. The allocation instructions, encoded as K_t (F_t), are then sent to the robots within each team to follow.

D. A Simple Example

We present a three-node example in Fig. 3, where all nodes are key nodes for simplicity, i.e., $\mathcal{V}_{\text{key}} = \mathcal{V}$. In (a), the Attacker starts with an initial allocation of $\mathbf{y}_{-1} = [0, 0, 2]$, while the Defender selects $\mathbf{x}_0 = [2, 2, 2]$ as its starting configuration. In (b), after observing $\mathbf{x}_0$, the Attacker employs the red matrix F_{-1} to update its allocation to $\mathbf{y}_0 = [0, 1, 1]$. In (c), the Defender redistributes its own resources via the blue matrix K_0. The states depicted in (c) are $\mathbf{x}_1 = [3, 2, 1]$ and $\mathbf{y}_0 = [0, 1, 1]$. Finally, in (d), the Attacker observes that $\mathbf{x}_3$ has only one blue robot at node 3 and moves its Robot 1 from node 2 to node 3 to breach the node. Consequently, the game terminates at time $t = 1$ with Attacker's victory, concluding with the states $\mathbf{x}_1 = [3, 2, 1]$ and $\mathbf{y}_1 = [0, 0, 2]$.

[1] When the two players share the same graph, we have $\mathcal{F} = \mathcal{K}$. For consistency, we still use $\mathcal{K}$ and $\mathcal{F}$ to denote the two action spaces.

[2] In a perimeter defense scenario, the key nodes can be the positions on the perimeter. When the Defender is defending a high-value asset, the key nodes can be the entrance points to the asset.

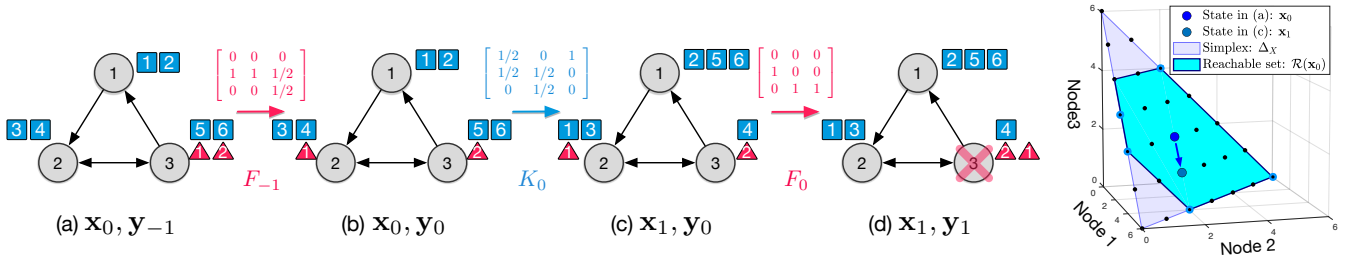

Fig. 3. An illustrative example of a three-node dDAB game, with Attacker's victory at time $t = 1$. Self-loop on each node is implied, and all nodes are key nodes. The agents are indexed to illustrate their movements. Right-most plot presents the Defender's reachable set from the allocation in (a). The black dots indicate the discrete states if the Defender's resources consists of indivisible units/robots.

E. Research Problems

Based on the discussion above, an instance of a dDAB game is defined by: (i) the available number of robots X and Y, (ii) the graph $\mathcal{G}$, and (iii) the required defense horizon. Given a graph, our goal is to identify the necessary and sufficient number of robots for the Defender to win the game, as well as its corresponding strategies. To formalize the above goal, we introduce the following multiplicative factor.

Definition 1 (Critical Resource Ratio)**.** *For a given graph $\mathcal{G}$ and a time horizon T, the Critical Resource Ratio (CRR), $\alpha_T \geq 1$, is the* smallest *positive number such that, if $X \geq \alpha_T Y$, then the Defender has a strategy to defend up to time step T against any admissible Attacker strategy that starts at any initial state $\mathbf{y}_{-1} \in \Delta_Y$. We use α_∞ to denote the CRR that enables the Defender to defend* indefinitely.

Notice that the CRR defined above is the *necessary and sufficient* number of Defender robots to guarantee defense over the given time horizon for the given graph.

The two main questions we address in this work are:

Problem 1. *Given a graph and a finite horizon T, what is the CRR α_T?*

Problem 2. *When $X \geq \alpha_T Y$, what is the Defender strategy that guarantees defense over T time steps? Given insufficient number of Defender robots, what is the optimal Attacker strategy to achieve the earliest possible breach?*

III. Single-Step Defense

In this section, we study the Defender's allocation configurations that guarantee defense at the current time step and introduce key concepts essential for the subsequent analysis. Several results build on prior work [25] and are included here for completeness.

A. Reachable Sets

To better predict and understand how the allocation of resources evolves over time, we focus on the possible states that the Defender and Attacker can reach at the next step, i.e., their reachable sets. Working with reachable sets offers two main advantages over working directly with the action spaces $\mathcal{K}$ and $\mathcal{F}$: (i) the dimensionality of the reachable sets is significantly lower than that of the edge sets ($|\mathcal{V}| \ll |\mathcal{E}|$), and (ii) the reachability analysis circumvents the non-uniqueness of actions that can produce a given transition from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$. Since the dynamics of the two players are symmetric, we restrict our analysis to the Defender's reachable sets and its action space $\mathcal{K}$.

Definition 2 (Reachable Set from a Point)**.** *The reachable set from a single point $\mathbf{x}_t$, denoted as $\mathcal{R}(\mathbf{x}_t)$, comprises all states the Defender can reach at the next time step with an admissible action. Formally,*

$$\mathcal{R}(\mathbf{x}_t) = \{\mathbf{x} \mid \exists K \in \mathcal{K} \text{ s.t. } \mathbf{x} = K\mathbf{x}_t\}. \tag{6}$$

Due to the linear dynamics in (1), we have the following result that characterizes the geometry of the reachable set.

Lemma 1. *Given a point $\mathbf{x}_t$, the reachable set $\mathcal{R}(\mathbf{x}_t)$ is given by the convex hull $\mathcal{R}(\mathbf{x}_t) = \mathrm{Co}\big(\{\hat{K}\mathbf{x}_t\}_{\hat{K}\in\hat{\mathcal{K}}}\big)$, where the extreme action set $\hat{\mathcal{K}}$ is defined as*

$$\hat{\mathcal{K}} = \big\{K \in \mathcal{K} \mid [K]_{ij} \in \{0, 1\}\big\}. \tag{7}$$

In words, $\hat{\mathcal{K}}$ contains all admissible *extreme* actions whose entries are either 0 or 1, and these extreme actions do not split resources on a node to multiple nodes. All generic actions $K \in \mathcal{K}$ can then be written as a convex combination of the extreme actions.

Denote by $\mathbf{v}_{t+1}^{(\ell)} = \hat{K}^{(\ell)}\mathbf{x}_t$ the state achieved by propagating $\mathbf{x}_t$ with extreme action $\hat{K}^{(\ell)} \in \hat{\mathcal{K}}$. Then, $\mathcal{R}(\mathbf{x}_t)$ is a polytope in Δ_X with (a subset of) $\{\mathbf{v}_{t+1}^{(\ell)}\}_\ell$ as its vertices. The right most plot in Fig. 3 presents an example of the reachable set for the aforementioned simple three-node example. The vertices of the reachable set are circled with blue. For discrete resources (robots), the Defender is able to achieve any discrete state (black dot) that is contained in the reachable set.

Using the same argument, we can construct the Attacker reachable set via $\mathcal{R}(\mathbf{y}_t) = \mathrm{Co}\big(\{\mathbf{w}_{t+1}^{(r)}\}_r\big)$, where the vertices are given by $\mathbf{w}_{t+1}^{(r)} = \hat{F}^{(r)}\mathbf{y}_t$ for $r = 1, 2, \ldots, |\hat{F}|$.

Since any state in $\mathcal{R}(\mathbf{x}_t)$ can be reached at the next time step from $\mathbf{x}_t$, we view, equivalently, this polytope as the action space for the Defender at state $\mathbf{x}_t$. This definition of the action space provides the two key advantages discussed earlier: reduced dimensionality and uniqueness.

The definition of reachable set of a single point can be extended to reachable set of a (potentially unbounded) set.

Definition 3 (Reachable Set from a Set)**.** *Given a set $P \subseteq \mathbb{R}^n_{\geq 0}$, the reachable set from P is defined as*

$$\mathcal{R}(P) = \{\mathbf{x} = K\mathbf{x}_t \mid K \in \mathcal{K},\ \mathbf{x}_t \in P\}. \tag{8}$$

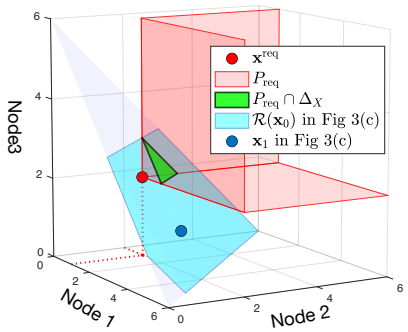

Fig. 4. Illustration of the required set $\mathcal{P}_{\text{req}}$ for the graph in Fig. 3 with $\mathbf{y}_1 = [0, 1, 1]^\top$ and $X = 6$. We conclude that $\mathbf{x}_1^{\text{req}} = [1, 2, 2]^\top$ and hence $X^{\text{req}} = 4$. The red surface shows the boundary of $\mathcal{P}_{\text{req}}$.

Lemma 2. *Given a polytope P, the reachable set $\mathcal{R}(P)$ is also a polytope.*

B. Required Set

In this section, we consider the Defender's selection of $\mathbf{x}_{t+1}$ after observing the Attacker's current allocation $\mathbf{y}_t$. The goal is to identify the set of the Defender's feasible states $\mathbf{x}_{t+1}$ such that the Attacker, upon observing $\mathbf{x}_{t+1}$, cannot select an action $\mathbf{y}_{t+1} \in \mathcal{R}(\mathbf{y}_t)$ that results in a successful breach of any key node.

For the Defender to defend every key node at time $t+1$, it is necessary and sufficient that the allocation vector $\mathbf{x}_{t+1}$ matches or outnumbers $\mathbf{y}_{t+1}$ at every key node $i \in \mathcal{V}_{\text{key}}$:

$$[\mathbf{x}_{t+1}]_i \geq [\mathbf{y}_{t+1}]_i \quad \forall i \in \mathcal{V}_{\text{key}}. \tag{9}$$

Since the Attacker takes its action after observing the Defender's allocation $\mathbf{x}_{t+1}$, the question is whether the Defender can select an allocation $\mathbf{x}_{t+1}$ such that (9) is true for all $\mathbf{y}_{t+1} \in \mathcal{R}(\mathbf{y}_t)$. This observation leads to the following condition for selecting $\mathbf{x}_{t+1}$ to guarantee defense at time $t+1$:

$$[\mathbf{x}_{t+1}]_i \geq \max_{\mathbf{y}_{t+1} \in \mathcal{R}(\mathbf{y}_t)} [\mathbf{y}_{t+1}]_i \quad \forall i \in \mathcal{V}_{\text{key}}. \tag{10}$$

Since $\mathcal{R}(\mathbf{y}_t)$ is a bounded polytope (Lemma 1), the optimization $\max_{\mathbf{y}_{t+1} \in \mathcal{R}(\mathbf{y}_t)} [\mathbf{y}_{t+1}]_i$ can be viewed as a linear program, whose optimum is attained at one of the vertices of $\mathcal{R}(\mathbf{y}_t)$. Consequently, we define the minimum required Defender resources at $t+1$ as $\mathbf{x}_{t+1}^{\text{req}}$, whose elements are

$$[\mathbf{x}_{t+1}^{\text{req}}]_i = \begin{cases} \max_r \left[\mathbf{w}_{t+1}^{(r)}\right]_i & \text{if } i \in \mathcal{V}_{\text{key}} \\ 0 & \text{otherwise} \end{cases}, \tag{11}$$

where $\left\{\mathbf{w}_{t+1}^{(r)}\right\}_r = \left\{\hat{F}^{(r)}\mathbf{y}_t\right\}_r$ are the vertices of $\mathcal{R}(\mathbf{y}_t)$. Then, the condition in (10) can be expressed in the following (component-wise) vector inequality form:

$$\mathbf{x}_{t+1} \geq \mathbf{x}_{t+1}^{\text{req}}. \tag{12}$$

We now claim that the Defender can guarantee defense at $t+1$ by selecting $\mathbf{x}_{t+1}$ inside the polytope $\mathcal{P}_{\text{req}}(\mathbf{y}_t)$, which is defined as follows.

Definition 4 (Required Set)**.** *Given the Attacker's allocation $\mathbf{y}_t$ at time t, the required set for the Defender at time $t+1$ is defined as:*

$$\mathcal{P}_{\text{req}}(\mathbf{y}_t) \triangleq \{\mathbf{x}_{t+1} \mid [\mathbf{x}_{t+1}]_i \geq [\mathbf{x}_{t+1}^{\text{req}}(\mathbf{y}_t)]_i,\ \forall\, i \in \mathcal{V}\}. \tag{13}$$

Proposition 1. *The condition $\mathbf{x}_{t+1} \in \mathcal{P}_{\text{req}}(\mathbf{y}_t)$ is necessary and sufficient for the Defender to defend time step $t+1$.*

Proof. From (10) and the definition of the Attacker's reachable set, the Defender can guarantee that key node i is defended against all feasible Attacker allocations at $t+1$ if it allocates at least $[\mathbf{x}_{t+1}^{\text{req}}]_i$ to that node. This establishes *sufficiency*.

Suppose the Defender allocates $[\mathbf{x}_{t+1}]i < [\mathbf{x}_{t+1}^{\text{req}}]_i$. Then, the condition (10) is violated, and there exists a vertex $\mathbf{w}_{t+1}^{(r)}$ of the Attacker's reachable set such that $[\mathbf{w}_{t+1}^{(r)}]_i > [\mathbf{x}_{t+1}]_i$. In this case, after observing the Defender's allocation, the Attacker can select a feasible action (e.g., $\hat{F}^{(r)}$) to reach $\mathbf{w}_{t+1}^{(r)} \in \mathcal{R}(\mathbf{y}_t)$ and breach key node i. This establishes *necessity*. □

In other words, as long as the Defender can reach an allocation within the required set $\mathcal{P}_{\text{req}}(\mathbf{y}_t)$, it is guaranteed to be safe at the next time step $t+1$. Conversely, if the Defender fails to achieve such an allocation, it will lose the game at $t+1$ against a rational Attacker.

There are three possible reasons the Defender may fail to reach the required set:

1) *Insufficient resources*: The total required resource, $X_{t+1}^{\text{req}} = \mathbf{1}^\top \mathbf{x}_{t+1}^{\text{req}}$, depends on the graph $\mathcal{G}$ and the current Attacker allocation $\mathbf{y}_t$. If $X_{t+1}^{\text{req}} > X$, then no Defender strategy can guarantee defense, regardless of the current allocation $\mathbf{x}_t$.
2) *Suboptimal strategy*: An allocation within the required set is feasible from the current Defender state $\mathbf{x}_t$, but the Defender selects a bad next allocation outside of $\mathcal{P}_{\text{req}}(\mathbf{y}_t)$.
3) Bad current allocation: The Defender has sufficient total resource ($X \geq X_{t+1}^{\text{req}}$), but its current state $\mathbf{x}_t$ does not allow it to reach any point in the required set. That is, $\mathcal{R}(\mathbf{x}_t) \cap \mathcal{P}_{\text{req}}(\mathbf{y}_t) = \varnothing$.

C. Example

For the Attacker configuration $\mathbf{y}_0$ in Fig. 3(c), the required Defender allocation at the next time step, denoted $\mathbf{x}_1^{\text{req}}$, can be determined as follows. At **node 1**, the Attacker can place at most one robot by moving Robot 2 from node 3. At **node 2**, the Attacker can place two robots by keeping Robot 1 at node 2 and moving Robot 2 there. Similarly, at **node 3**, two robots can be placed. Thus, we can conclude that $\mathbf{x}_1^{\text{req}} = [1, 2, 2]$. One can observe that the Defender allocation $\mathbf{x}_1 = [3, 2, 1]$ in (c) falls short for $\mathbf{x}_1^{\text{req}}$ at node 3, and thus is breached by the Attacker in (d).

Fig. 4 highlights in green the intersection between the required set $\mathcal{P}_{\text{req}}(\mathbf{y}_0)$ and the Defender's state space Δ_X. For the action sequence presented in Fig. 3, one can observe that the selected Defender state $\mathbf{x}_1$ lies outside the green intersection, leading to the Defender's defeat at time $t = 1$. This failure corresponds to case (2) suboptimal strategy, since the reachable set (cyan) does intersect the required set, but the Defender chooses a suboptimal allocation.

IV. Multi-Step Defense: No-Splitting Attacker

This section develops the tools required to construct multi-step feedback strategies for both the Defender and the Attacker. We begin by analyzing the case where the Attacker allocates its robots as a single concentrated group (a "blob") — a restriction we will lift in the next section. Note that, throughout this paper, we do not restrict the Defender's allocation strategies.

Let $\mathbf{e}_i \in \mathbb{R}^n$ be the unit vector with its i-th element equal to one and all others being zero. For notational convenience, we define $\mathbf{y}^{(i)} \triangleq Y\mathbf{e}_i$ to represent the Attacker allocation fully concentrated at node i.

Definition 5 (No-Splitting Attacker Strategy)**.** *A no-splitting Attacker strategy selects its action exclusively from the set of extreme actions, i.e., $F_t \in \hat{\mathcal{F}}$ for all t.*

Under a no-splitting Attacker strategy, if the initial Attacker allocation is fully concentrated, then the Attacker's state remains concentrated for all time steps, i.e., $\mathbf{y}_t = \mathbf{y}^{(i_t)} \triangleq Y\mathbf{e}_{i_t}$, where i_t denotes the location of the Attacker's concentrated resources.

A. k-step Safe Sets

The key challenge we address in this section is the fact that selecting a state in the required set does not imply that the Defender can do so again in the next time step.[3] As an example, for the Defender to guarantee defense over the next *two* time steps starting from the current allocations $\mathbf{x}_t$ and $\mathbf{y}_t$, the following condition is necessary:

$$\exists\, \mathbf{x}_{t+1} \in \mathcal{P}_{\mathrm{req}}(\mathbf{y}_t) \cap \mathcal{R}(\mathbf{x}_t) \text{ s.t.} \tag{14a}$$

$$\mathcal{P}_{\mathrm{req}}(\mathbf{y}_{t+1}) \cap \mathcal{R}(\mathbf{x}_{t+1}) \neq \varnothing,\ \forall\, \mathbf{y}_{t+1} \in \mathcal{R}(\mathbf{y}_t). \tag{14b}$$

In words, (14a) ensures that the Defender selects a reachable allocation $\mathbf{x}_{t+1}$ that guarantees defense at $t+1$, while (14b) ensures that the selected $\mathbf{x}_{t+1}$ allows transition to a state $\mathbf{x}_{t+2}$ that guarantees defense at $t+2$ against all potential Attacker allocation $\mathbf{y}_{t+1} \in \mathcal{R}(\mathbf{y}_t)$.

The need to account for all possible future Attacker actions quickly renders this formulation intractable beyond two steps. To overcome this, we introduce the notion of k-step safe sets, later formalized as Q-sets.

Definition 6 (k-step Safe Set)**.** *Let $\mathbf{y}_{t-1} = \mathbf{y}^{(i)}$ be the Attacker state concentrated at node i. The set $\mathcal{Q}_k^{(i)} \subseteq \mathbb{R}_{\geq 0}^{|\mathcal{V}|}$ is defined such that $\mathbf{x}_t \in \mathcal{Q}_k^{(i)}$ if and only if there exists a Defender strategy that can defend against any no-splitting Attacker strategy through time step $t+k$ (inclusive).*

While the above definition introduces the concept of multi-step safe sets, we next present a recursive formulation for constructing Q-sets in the following theorem.

Theorem 1. *The following recursive expression provides the k-step safe set:*

$$\mathcal{Q}_0^{(i)} = \mathcal{P}_{\mathrm{req}}(\mathbf{y}^{(i)}), \tag{15a}$$

$$\mathcal{Q}_k^{(i)} = \left\{\mathbf{x} \middle| \mathbf{x} \in \mathcal{P}_{\mathrm{req}}(\mathbf{y}^{(i)}) \wedge \mathcal{R}(\mathbf{x}) \cap \mathcal{Q}_{k-1}^{(j)} \neq \varnothing\ \forall j \in \mathcal{N}_i\right\}, \quad \forall\, k \geq 1, \tag{15b}$$

where $\mathcal{N}_i$ is the set of out-neighbors of node i.

Proof Sketch. The proof proceeds by induction. The base case in (15a) matches the required condition for single-step safety (cf. Proposition 1). For the inductive step, (15b) requires that: $\mathbf{x}$ defends against an Attacker concentrated at node i at the current step, i.e., $\mathbf{x} \in \mathcal{P}_{\mathrm{req}}(\mathbf{y}^{(i)})$; meanwhile, for every possible next Attacker node $j \in \mathcal{N}_i$, the Defender can reach the corresponding $(k-1)$-step safe set, i.e., $\mathcal{R}(\mathbf{x}) \cap \mathcal{Q}_{k-1}^{(j)} \neq \varnothing$. Formal proofs of sufficiency (Lemma 4) and necessity (Lemma 5) are given in the appendix. □

Next, we present two important properties of the Q-sets.

Remark 1. *For a fixed node $i \in \mathcal{V}$, $\left(\mathcal{Q}_k^{(i)}\right)_k$ is a decreasing sequence of sets. That is*

$$\mathcal{Q}_{k+1}^{(i)} \subseteq \mathcal{Q}_k^{(i)}, \forall k \geq 0, i \in \mathcal{V}. \tag{16}$$

The above remark follows directly from the definition of the Q-sets: if the Defender can defend $k+1$ steps from some state, then it can clearly defend k steps.

Theorem 2. *For all finite k and all nodes $i \in \mathcal{V}$, the Q-set $\mathcal{Q}_k^{(i)}$ is a polytope.*

The proof is deferred to Section VI, as the result follows directly from the numerical algorithm introduced therein, where the Q-sets are computed via intersections of polyhedra.

B. Q-Set Propagation

The Q-set propagation process is described in Algorithm 1, which takes three inputs: the graph environment $\mathcal{G}$, the Attacker total resource Y, and the horizon of the game T. We assume that the players do not consider their performance beyond T, and therefore, Q-sets are only computed up to this horizon. The algorithm applies the recursion in (15) to construct the Q-sets for each node. In practice, a numerically efficient implementation uses an equivalent but computationally friendly formulation (28) in Section VI.

The iterative construction terminates if the Q-sets converge, as checked in line 4 of the algorithm[4]. In this case, we can conclude that the Defender has a strategy to defend indefinitely against all no-splitting Attacker strategies. For the remainder of the paper, when Algorithm 1 converges, we refer to the converged Q-sets as $\{\mathcal{Q}_\infty^{(i)}\}_i$ for notational simplicity.

[3] See Fig. 4 of [25] for an example of a situation where single-step defense can be achieved, but the Attacker is able to breach after two steps.

[4] Since all Q-sets are polytopes, one can simply check the vertices (and extreme rays) for convergence.

Algorithm 1: Q-Prop

Inputs: $\mathcal{G}$, Y, T;

Set $k_\infty = \infty$ and set $\mathcal{Q}_0^{(i)} = \mathcal{P}_{\text{req}}(\mathbf{y}^{(i)})$ for all $i \in \mathcal{V}$;
for $k = 1$ *to* T **do**
  Construct $\mathcal{Q}_k^{(i)}$ using (15) or (28) for all $i \in \mathcal{V}$;
  if $\mathcal{Q}_k^{(i)} = \mathcal{Q}_{k-1}^{(i)}$ *for all* $i \in \mathcal{V}$ **then**
    $k_\infty = k - 1$;
    Break;
  end
end
Return: $\{\mathcal{Q}_k^{(i)}\}_{i\in\mathcal{V},k\in[T]}$, k_∞

C. Indefinite Defense

Since $\mathcal{Q}_k^{(i)} \subseteq (\mathbb{R}_{\geq 0})^{|\mathcal{V}|}$ for all $i \in \mathcal{V}$, the recursive definition in (15) can be interpreted as a set operator acting on powersets $\left(2^{\mathbb{R}_{\geq 0}^{|\mathcal{V}|}}\right)^{|\mathcal{V}|}$. From this perspective, (15) becomes an iterative algorithm, and its fixed points are therefore of great interest for indefinite defense.

Definition 7 (Indefinite Safe Set)**.** *We define the indefinite safe sets* $\mathcal{Q}_\infty^{(i)} \subseteq (\mathbb{R}_{\geq 0})^{|\mathcal{V}|}$ *for* $i \in \mathcal{V}$ *as follows:*

$$\mathcal{Q}_\infty^{(i)} = \lim_{k\to\infty} \mathcal{Q}_k^{(i)}. \tag{17}$$

The following result formalizes the intuitive claim that indefinite safe sets characterize the Defender's ability to defend indefinitely against any no-splitting Attacker strategy.

Theorem 3 (Indefinite Defense)**.** *If* $\mathcal{Q}_\infty^{(i)} \neq \varnothing$ *for some node* $i \in \mathcal{V}$*, then* $\mathbf{x}_t \in \mathcal{Q}_\infty^{(i)}$ *is necessary and sufficient for indefinite defense given that the Attacker is at* $\mathbf{y}_{t-1} = \mathbf{y}^{(i)}$.

With some additional technicalities, one can establish that the indefinite safe sets is a fixed-point of (15). Specifically, for every node $i \in \mathcal{V}$, the set $\mathcal{Q}_\infty^{(i)}$ satisfies

$$\mathcal{Q}_\infty^{(i)} = \left\{\mathbf{x} \middle| \mathbf{x} \in \mathcal{P}_{\text{req}}(\mathbf{y}^{(i)}) \wedge \mathcal{R}(\mathbf{x}) \cap \mathcal{Q}_\infty^{(j)} \neq \varnothing \ \forall j \in \mathcal{N}_i\right\}. \tag{18}$$

However, the existence and uniqueness of indefinite safe sets, as well as the convergence of the recursion (15) depend heavily on the graph structure. In particular, not all graphs admit nonempty fixed points. For example, for graphs with sink nodes where Defender resources get absorbed, infinite Defender resources is required for indefinite defense, and $\mathcal{Q}_\infty^{(i)}$ is empty for all i.

Empirically, we observe that for all strongly-connected and undirected graphs, the iteration in (15) converges within $|\mathcal{V}|$ steps. Establishing formal guarantees for convergence and characterizing the class of graphs that admit nonempty indefinite safe sets remain important directions for future research.

D. Restricted K-step Defender Strategies

The construction of Q-sets in Theorem 1 provides a guideline for designing the strategies of the Defender and the Attacker under the *no-splitting restriction*. We begin by summarizing the Defender strategy in the following two algorithms.

Algorithm 2 specifies how the Defender selects its initial allocation. Given the total resource budget X and the observed initial Attacker allocation $\mathbf{y}_{-1}$, the Defender chooses an initial state that maximizes the guaranteed defense horizon as in line 2.

Algorithm 2: Initial Defender Allocation (against No-Splitting Attacker)

Inputs: $\mathcal{G}$, X, Y, $\mathbf{y}_{-1} = \mathbf{y}^{(i_{-1})}$, T;
Construct Q-sets via Algorithm 1;
$k_{\max,0} \leftarrow$ ▷ find the longest defense time
  $\arg\max_k \left\{k \leq \min\{T, k_\infty\} | \Delta_X \cap \mathcal{Q}_k^{(i_{-1})} \neq \varnothing\right\}$;
$\mathbf{x}_0 \leftarrow$ any element in $\mathcal{Q}_{k_{\max,0}}^{(i_{-1})}$
Return: Initial allocation $\mathbf{x}_0$, defense time $k_{\max,0}$

Algorithm 3 outlines the feedback strategy for the Defender. Suppose that Algorithm 2 returns $k_{\max,0} = k_\infty$. In this case, the Defender has sufficient resources to guarantee indefinite defense. At each time step t, upon observing the Attacker's allocation $\mathbf{y}^{(i_t)}$, the Defender reallocates its resources to a feasible state $\mathbf{x}_{t+1}$ in the corresponding set $\mathcal{Q}_\infty^{(i_t)}$.

On the other hand, if Algorithm 2 returns $k_{\max,0} < k_\infty$, then the Defender is only guaranteed to defend up to time step $k_{\max,0}$. This is due to either (i) the Q-set iteration in Algorithm 1 did not converge within the considered horizon, or (ii) the Defender does not have sufficient resources to ensure indefinite defense. If we further have $k_{\max,0} < T$, then a rational Attacker will identify a winning strategy and breach the defense at time $t = k_{\max,0} + 1$. This scenario is analyzed explicitly in Algorithms 4 and 5. Under rational play from both sides, the value $k_{\max,t}$ decreases by 1 at each step, leading to termination with Attacker's victory at $t = k_{\max,0} + 1$. However, if the Attacker is not rational, the Defender may be able to delay the breach. The optimization performed in line 1 of Algorithm 3 ensures that the Defender exploits such opportunity.

Finally, if the intersection $\mathcal{R}(\mathbf{x}_t) \cap \mathcal{Q}_0^{(i_t)}$ is empty, then a breach is imminent at the next step (against a rational Attacker). In this case, the Defender selects any admissible allocation in hopes of an Attacker mistake. For practical purposes, we select the reachable state closest to the $\mathcal{Q}_0^{(i_t)}$ in line 5.

Algorithm 3: Feedback Defender Strategy (against No-Splitting Attacker)

Inputs: Q-sets, $\mathbf{x}_t$, $\mathbf{y}_t = \mathbf{y}^{(i_t)}$, T;
if $\mathcal{R}(\mathbf{x}_t) \cap \mathcal{Q}_0^{(i_t)} \neq \varnothing$ **then**
  $k_{\max,t+1} \leftarrow$ ▷ exploit Attacker's mistake
    $\arg\max_k \left\{k \leq \min\{T, k_\infty\} | \mathcal{R}(\mathbf{x}_t) \cap \mathcal{Q}_k^{(i_t)} \neq \varnothing\right\}$;
  $\mathbf{x}_{t+1} \leftarrow$ any element in $\mathcal{R}(\mathbf{x}_t) \cap \mathcal{Q}_{k_{\max,t}}^{(i_t)}$;
else ▷ breach imminent
  $\mathbf{x}_{t+1} \leftarrow$ element in $\mathcal{R}(\mathbf{x}_t)$ closest to $\mathcal{Q}_0^{(i_t)}$;
end
Return: Next allocation $\mathbf{x}_{t+1}$, defense time $k_{\max,t+1}$

E. K-step Attacker Strategies

The next two algorithms describe the Attacker's strategy under the no-splitting restriction. In particular, Algorithm 4 defines the initial allocation of the Attacker, while Algorithm 5 specifies the Attacker's feedback strategy for all subsequent time steps $t \geq 0$.

Algorithm 4: Attacker Initial Allocation

Inputs: $\mathcal{G}$, X, Y, T;

Construct Q-sets using Algorithm 1;

if $\exists k \leq \min\{T, k_\infty\}$, $i \in \mathcal{V}$ *s.t.* $\Delta_X \cap \mathcal{Q}_k^{(i)} = \varnothing$ **then**

  $k_{\min,-1} \leftarrow$ ▷ find the earliest breach time $\arg\min_k \left\{ k \leq \min\{T, k_\infty\} \mid \Delta_X \cap \mathcal{Q}_k^{(i)} = \varnothing \right\}$;

  $i^*_{-1} \leftarrow$ any element in $\{i \mid \Delta_X \cap \mathcal{Q}^{(i)}_{k_{\min,-1}} = \varnothing\}$;

else

  $k_{\min,-1} \leftarrow \infty$; ▷ no breach found

  $i^*_{-1} \leftarrow$ any element in node set $\mathcal{V}$;

end

Return: Initial allocation $\mathbf{y}_{-1} = \mathbf{y}^{(i^*_{-1})}$, guaranteed breach time $k_{\min,-1}$.

Algorithm 5: Feedback Attacker Strategy

Inputs: Q-sets, $\mathbf{x}_{t+1}$, i_t, T;

if $\exists k \leq \min\{T, k_\infty\}$ *and* $j \in \mathcal{N}_{i_t}$ *such that* $\mathbf{x}_{t+1} \notin \mathcal{Q}_k^{(j)}$ **then**

  $k_{\min,t+1} \leftarrow$ ▷ exploit Defender's mistake $\arg\min_k \left\{ k \leq \min\{T, k_\infty\} \mid \mathbf{x}_{t+1} \notin \mathcal{Q}_k^{(j)},\ j \in \mathcal{N}_{i_t} \right\}$;

  $i^*_{t+1} \leftarrow$ any element in $\{j \in \mathcal{N}_{i_t} \mid \mathbf{x}_{t+1} \notin \mathcal{Q}^{(j)}_{k_{\min,t+1}}\}$;

else

  $k_{\min,t+1} \leftarrow \infty$;

  $i^*_{t+1} \leftarrow$ any element in $\mathcal{N}_{i_t}$;

end

Return: Next allocation $\mathbf{y}_{t+1} = \mathbf{y}^{(i^*_{t+1})}$, guaranteed breach time $k_{\min,t+1}$.

The Attacker can win the game if and only if the Defender allocates resource outside the corresponding Q-sets. When the Defender maintains its allocation within $\mathcal{Q}_T^{(i)}$ or $\mathcal{Q}_{k_\infty}^{(i)}$, the Attacker cannot breach any key node, provided the Defender plays rationally. In this case, the Attacker has no preference among its feasible actions since we formulated the dDAB game as a *game of kind* without any performance metric. Therefore, the node selections in line 7 of Algorithm 4 and line 6 of Algorithm 5 are arbitrary[5].

As we show later in Corollary 1, the Attacker has no incentive to split, i.e., if the Attacker can win a dDAB game by splitting, it can also win the game without splitting. Consequently, the algorithms presented here are sufficient for the Attacker to play the dDAB game. However, the Defender strategies need to be generalized to counter potential splitting Attacker, which we will present in the next section.

[5]An extension of this work is to incorporate costs for the Defender's reallocation. This would convert the game into a game of degree, and could be modeled under the competitive convex body chasing framework [27].

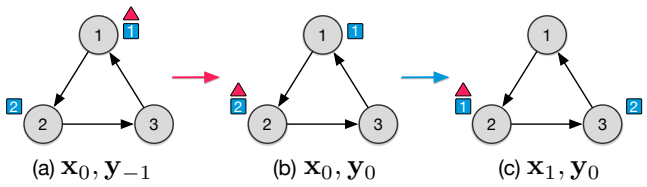

Fig. 5. An example of indefinite-defense on a ring graph with self-loops.

F. Ring-graph Example

We apply the proposed algorithms to an example that admits indefinite defense. Consider the (directed) ring graph with self-loops shown in Fig. 5. In this case, the Q-set propagation algorithm converges immediately with $k_\infty = 0$, implying that $\mathcal{Q}_\infty^{(i)} = \mathcal{P}_{\text{req}}(\mathbf{y}^{(i)})$ for all nodes $i \in \mathcal{V}$. The resulting strategy for the Defender is straightforward: upon observing the Attacker's allocation $\mathbf{y}_0$, the Defender places one robot at the Attacker's current node and one robot at the node immediately ahead in the ring. This allocation guarantees coverage of both the current and potential next positions of the Attacker robot. The Defender can maintain this pattern indefinitely, thereby ensuring indefinite defense.

Next, we demonstrate how the above Defender strategy is generated using Algorithm 3. Fig. 6 illustrates the computed indefinite Q-sets for nodes 1 and 2 in the ring graph; the Q-set for node 3 is similar, with its vertex located at $[1, 0, 1]$.

With the Attacker robot at node 1 and two Defender robots, the initial Defender allocation $\mathbf{x}_0$ is obtained by solving the feasibility problem $\Delta_X \cap \mathcal{Q}_\infty^{(1)}$, which yields the unique solution $\mathbf{x}_0 = [1, 1, 0]$, as shown in subplot (a). After the Attacker moves to node 2, the Defender computes its next allocation via $\mathcal{R}(\mathbf{x}_0) \cap \mathcal{Q}_\infty^{(2)}$. This again yields a unique solution, $\mathbf{x}_1 = [0, 1, 1]$, depicted in subplot (b). If the Attacker remains at node 2, the Defender's allocation remains unchanged. Otherwise, if the Attacker moves to node 3 (i.e., $\mathbf{y}_1 = [0, 0, 1]$), the same process can be repeated, yielding the next Defender allocation $\mathbf{x}_2 = [1, 0, 1]$. This behavior is consistent with Theorem 3 in the prior work [25], though here we derive it from an algorithmic framework capable of handling generic graphs.

Remark 2. *Although the Q-sets are constructed based on continuous resources, one can extract policies for indivisible robots by selecting the discrete states lying within the Q-sets (e.g., black dots in Fig. 6).*

V. Generalized Defense Strategies

This section generalizes the Defender strategy from the previous section to scenarios where the Attacker can split its resources to multiple nodes. The core idea leverages the superposition of Defender's reactions to each individual Attacker subteam's reallocation.

Definition 8 (Attacker Subteam)**.** *We refer to the Attacker robots allocated to each node as an Attacker subteam. The size of the i-th subteam (on node i) at time t is given by $[\mathbf{y}_t]_i$.*

Based on the notion of the Attacker subteam, we define the i-th Defender subteam as follows.

Definition 9 (Defender Subteam)**.** *The i-th (scaled) Defender subteam is defined as*

$$\mathbf{x}_t^{(i)} \triangleq \frac{[\mathbf{y}_{t-1}]_i}{Y}\hat{\mathbf{x}}_t^{(i)}, \text{ where } \hat{\mathbf{x}}_t^{(i)} \in \mathcal{Q}_k^{(i)}. \tag{19}$$

We refer to $\hat{\mathbf{x}}_t^{(i)}$ as the unscaled Defender subteam.

Note that the Q-sets in (15) are defined based on the total Attacker team size Y. Consequently, each Defender subteam is scaled according to the size of its corresponding Attacker subteam in (19). The defense condition for the Defender subteams is thus linked to the Q-sets through the unscaled Defender subteam $\hat{\mathbf{x}}$.

As illustrated in Fig. 7(a), the Attacker subteams at nodes 1 and 2 have sizes of 1 (red robot) and 2 (dark red and pink robots), respectively. The initial Defender subteams are $\mathbf{x}_0^{(1)} = [1,1,0]$ and $\mathbf{x}_{0,T}^{(2)} = [0,2,2]$. Specifically, the 1st Defender subteam consists of the two blue robots, while the 2nd Defender subteam comprises the dark blue and cyan robots. The dark blue Defender robots are assigned to defend against the dark red Attacker robot, and the cyan robots are designated to the pink robot.

Based on the results from the no-splitting Attacker scenario, if the Attacker subteams do not split further, each Defender subteam can successfully defend against its respective Attacker subteam for the next k steps, ensured by condition (19).

We now extend the above construction to scenarios where the Attacker subteams further split. This requires additional bookkeeping to track the splitting and merging of the Attacker's subteams. Consider a generic Attacker action F_{t-1} that transitions $\mathbf{y}_{t-1}$ to $\mathbf{y}_t$. Let $\boldsymbol{f}^{(i)}$ represent the i-th column of F_{t-1}, i.e., $F_{t-1} = [\boldsymbol{f}^{(1)}, \boldsymbol{f}^{(2)}, ..., \boldsymbol{f}^{(N)}]$. The vector $\boldsymbol{f}^{(i)}$ encodes the splitting action of the Attacker subteam on node i, where the fraction of this subteam relocating to node j is given by $[\boldsymbol{f}^{(i)}]_j$. For instance, in Fig. 7, the red action F_{-1} yields $[\boldsymbol{f}^{(2)}] = [0, 1/2, 1/2]^\top$, meaning that one of the two red robots on node 2 remains, while the other moves to node 3.

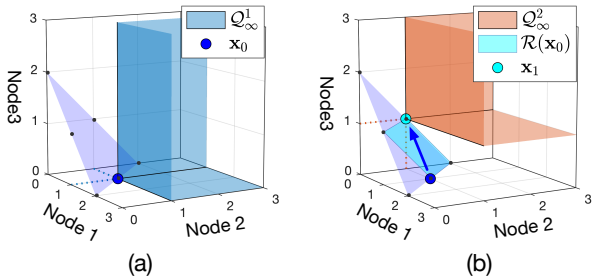

Fig. 6. An illustration of the indefinite Q-sets and the construction of the Defender strategy.

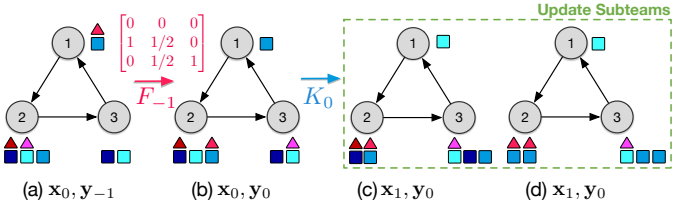

Fig. 7. An illustrative example with splitting Attacker over a ring graph.

The i-th Defender subteam should react to the splitting of the i-th Attacker subteam in the following manner.

1) The i-th Defender subteam is divided into "sub-subteams", according to the splitting action $\boldsymbol{f}^{(i)}$ of the i-th Attacker subteam.
2) Each j-th Defender sub-subteam of the i-th subteam counteracts the j-th Attacker sub-subteam that moves from node i to node j. Specifically, the sub-subteam is given by
$$\mathbf{x}_t^{(i\to j)} = [\boldsymbol{f}^{(i)}]_j \mathbf{x}_t^{(i)} = \frac{[\boldsymbol{f}^{(i)}]_j [\mathbf{y}_{t-1}]_i}{Y}\hat{\mathbf{x}}_t^{(i)}.$$
3) This counteraction is executed by applying the action $K^{(i\to j)}$, resulting in a new unscaled defender subteam:
$$\hat{\mathbf{x}}_{t+1}^{(i\to j)} = K^{(i\to j)}\hat{\mathbf{x}}_t^{(i)} \in \mathcal{Q}_{k-1}^{(j)}.$$
4) The new state of the Defender sub-subteam at the next time step is then
$$\mathbf{x}_{t+1}^{(i\to j)} = K^{(i\to j)}\mathbf{x}_t^{(i\to j)} = \frac{[\boldsymbol{f}^{(i)}]_j [\mathbf{y}_{t-1}]_i}{Y}\hat{\mathbf{x}}_{t+1}^{(i\to j)}. \tag{20}$$

In the example shown in Fig. 7, the new states of the sub-subteams are given by $\mathbf{x}_1^{(1\to 2)} = [0,1,1]$ (blue), $\mathbf{x}_1^{(2\to 2)} = [0,1,1]$ (dark blue), and $\mathbf{x}_1^{(2\to 3)} = [1,0,1]$ (cyan). Notably, $\mathbf{x}_{t+1}^{(i\to j)}$ contributes only a portion of the new j-th Defender subteam, originating from the previous i-th subteam.

To compute the new j-th Defender subteam, we group the Defender resources from different subteams that responded to the Attacker resources and ended up at node j. This grouping is expressed as:

$$\mathbf{x}_{t+1}^{(j)} = \sum_{i\in\mathcal{V}} \mathbf{x}_{t+1}^{(i\to j)} = \sum_{i\in\mathcal{V}} \frac{[\boldsymbol{f}^{(i)}]_j [\mathbf{y}_{t-1}]_i}{Y}\hat{\mathbf{x}}_{t+1}^{(i\to j)}. \tag{21}$$

This step is illustrated in Fig. 7(d), where the new Defender subteams are $\mathbf{x}_1^{(2)} = [0,2,2]$ (blue), and $\mathbf{x}_1^{(3)} = [1,0,1]$ (cyan). Meanwhile, two new Attacker subteams form at nodes 2 and 3, with sizes of 2 and 1, respectively.

The critical question is whether this j-th new Defender subteam can effectively defend against the new Attacker subteam at node j. Unscaling the new j-th Defender subteam in (21) gives

$$\hat{\mathbf{x}}_{t+1}^{(j)} = \frac{Y}{[\mathbf{y}_t]_j}\mathbf{x}_{t+1}^{(j)} = \sum_i \frac{[\boldsymbol{f}^{(i)}]_j [\mathbf{y}_{t-1}]_i}{[\mathbf{y}_t]_j}\hat{\mathbf{x}}_{t+1}^{(i\to j)}. \tag{22}$$

Note that the size of the new Attacker subteam on node j is $[\mathbf{y}_t]_j = \sum_i [\boldsymbol{f}^{(i)}]_j [\mathbf{y}_{t-1}]_i$. Thus, $\hat{\mathbf{x}}_{t+1}^{(j)}$ is a convex combination of the states $\hat{\mathbf{x}}_{t+1}^{(i\to j)}$, all of which are in the Q-set $\mathcal{Q}_{k-1}^{(j)}$ as constructed in (20). Given that the Q-sets are polytopes (Theorem 2), it follows that $\hat{\mathbf{x}}_{t+1}^{(j)} \in \mathcal{Q}_{k-1}^{(j)}$. Therefore, we can prove by induction that the Defender subteams at the next time step can maintain their defense for an additional $k-1$ steps.

Formally, we have the following theorem.

Theorem 4. *Given the Attacker's initial state $\mathbf{y}_{-1}$, the Defender can defend against any Attacker strategy until time T if the Defender's initial state $\mathbf{x}_0$ can be expressed as*

$$\mathbf{x}_0 = \sum_{i\in\mathcal{V}} \frac{[\mathbf{y}_{-1}]_i}{Y}\hat{\mathbf{x}}_0^{(i)}, \text{ for some } \hat{\mathbf{x}}_0^{(i)} \in \mathcal{Q}_T^{(i)}. \tag{23}$$

As a direct consequence of Theorem 4, the Defender strategy outlined in Section IV can be extended to scenarios where the Attacker distributes its resources across multiple nodes. The generalized Defender strategy is detailed in Algorithms 6 and 7. In particular, Lines 3–12 of Algorithm 6 are designed to improve Defender performance when the Attacker does not concentrate all of its robots at the most critical node that leads to the earliest breach.

Algorithm 6: Initial Defender Allocation

Inputs: $\mathcal{G}$, X, Y, $\mathbf{y}_{-1}$, T;
1 Construct Q-sets via Algorithm 1;
2 $\mathcal{I}_{-1} \leftarrow \{i \,|\, [\mathbf{y}_{-1}]_i > 0\}$;
3 **for** $k = 1, \dots, T$ **do**
4 **for** $i \in \mathcal{I}_{-1}$ **do**
5 $\hat{\mathbf{x}}_{0,k}^{(i)} \leftarrow \arg\min_{\mathbf{x}\in\Delta_X \cap \mathcal{Q}_i^{(k)}} \mathbf{1}^\top \mathbf{x}$;
6 **end**
7 $X_{\text{tot},k} = \sum_{i\in\mathcal{I}_{-1}} \frac{[\mathbf{y}_{-1}]_i}{Y} \mathbf{1}^\top \hat{\mathbf{x}}_0^{(i)}$;
8 **if** $X_{\text{tot},k} > X$ **then**
9 $k_{\max,0} = k-1$;
10 **Break**;
11 **end**
12 **end**
13 $X_{\text{tot}} \leftarrow X_{\text{tot},k_{\max,0}}$ and $\hat{\mathbf{x}}_0^{(i)} \leftarrow \frac{X}{X_{\text{tot}}} \hat{\mathbf{x}}_{k_{\max,0}}^{(i)}$
14 $\mathbf{x}_0 \leftarrow \sum \frac{[\mathbf{y}_{-1}]_i}{Y} \hat{\mathbf{x}}_0^{(i)}$;
15 **Return:** Initial allocation $\mathbf{x}_0$, initial subteams $\{\hat{\mathbf{x}}_0^{(i)}\}_i$, longest guaranteed defense time $k_{\max,0}$

A. The Critical Resource Ratio

We leverage the results in the previous sections to identify the critical resource ratio CRR that ensures k-step defense (see Definition 1). In Section IV, we have shown that being in $\mathcal{Q}_k^{(i)}$ is necessary and sufficient for the Defender to defend against any no-splitting Attacker strategy that starts from $\mathbf{y}_{-1} = \mathbf{y}^{(i)}$ for k steps. This leads to an intermediate version of the CRR defined for the case of no-splitting Attacker starting at node i:

$$\beta_k^{(i)} \triangleq \frac{1}{Y} \min_{\mathbf{x}\in\mathcal{Q}_k^{(i)}} \mathbf{1}^\top \mathbf{x}. \tag{24}$$

Given that the Attacker can freely select its initial state $\mathbf{y}_{-1}$, we define the k-step CRR against a no-splitting Attacker as

$$\beta_k \triangleq \max_{i\in\mathcal{V}} \beta_k^{(i)}. \tag{25}$$

The following result shows that the CRR against general splitting Attacker is identical to the one defined under no-splitting restriction.

[6]Note that if $[\mathbf{y}_t]_j = 0$, then there is no need to create a subteam designated for node j, and one can set $\hat{\mathbf{x}}_{t+1}^{(j)} = \mathbf{0}$.

Algorithm 7: Feedback Defender Strategy

Inputs: $\{\mathcal{Q}_k^{(i)}\}_{i,k}$, $\{\hat{\mathbf{x}}_t^{(i)}\}_i$, $\mathbf{y}_{t-1}$, $\mathbf{y}_t$;
1 $F_{t-1} \leftarrow$ Attacker action such that $\mathbf{y}_t = F_{t-1}\mathbf{y}_{t-1}$;
2 $k_{\max,t+1} \leftarrow$
 $\arg\max_k \left\{ k \mid \mathcal{R}(\hat{\mathbf{x}}_t^{(i)}) \cap \mathcal{Q}_k^{(i)} \neq \varnothing \ \forall\, i \in \mathcal{I}_t \right\}$;
 // Generate Defender subteams' actions
3 **for** $i \in \mathcal{V}$ **do**
4 $\boldsymbol{f}^{(i)} \leftarrow i$-th column of F_{t-1};
 // Reaction to Attacker subteam i splitting
5 **for** $j \in \mathcal{V}$ *s.t.* $[\boldsymbol{f}^{(i)}]_j \neq 0$ **do**
6 $\hat{\mathbf{x}}_{t+1}^{(i\to j)} \leftarrow$ element in $\mathcal{R}(\hat{\mathbf{x}}_t^{(i)}) \cap \mathcal{Q}_{k_{\max,t+1}}^{(j)}$;
7 **end**
8 **end**
 // Update subteams
9 **for** $j \in \mathcal{V}$ **do**
10 $\hat{\mathbf{x}}_{t+1}^{(j)} \leftarrow \sum_i \frac{[\mathbf{y}_{t-1}]_i [\boldsymbol{f}^{(i)}]_j}{[\mathbf{y}_t]_j} \hat{\mathbf{x}}_{t+1}^{(i\to j)}$; [6]
11 $\mathbf{x}_{t+1}^{(j)} = \frac{[\mathbf{y}_t]_j}{Y} \hat{\mathbf{x}}_{t+1}^{(j)}$;
12 **end**
13 $\mathbf{x}_{t+1} = \sum_{j\in\mathcal{V}} \mathbf{x}_{t+1}^{(j)}$;
14 **Return:** Next allocation $\mathbf{x}_{t+1}$, updated unscaled subteams $\{\hat{\mathbf{x}}_{t+1}^{(i)}\}_i$, guaranteed defense time $k_{\max,t+1}$.

Theorem 5 (Critical Resource Ratio)**.** *The necessary and sufficient resource ratio for the Defender to achieve k-step defense against any Attacker strategy is given by*

$$\alpha_k = \beta_k = \frac{1}{Y} \max_{i\in\mathcal{V}} \min_{\mathbf{x}\in\mathcal{Q}_k^{(i)}} \mathbf{1}^\top \mathbf{x}. \tag{26}$$

Proof. It is obvious that $\alpha_k \geq \beta_k$, since at least β_k is necessary against no-splitting strategies, and α_k considers all feasible Attacker strategies. Consequently, it suffices to show that $X = \beta_k Y$ is sufficient to guard against any admissible Attacker strategy, including the ones with splitting.

Using the result of Theorem 4, the minimum number of robots to achieve the allocation in (23) is given by

$$\begin{aligned}
\alpha_k &= \frac{1}{Y} \min_{\substack{(\hat{\mathbf{x}}_0^{(1)},\dots,\hat{\mathbf{x}}_0^{(|\mathcal{V}|)}) \\ \in \mathcal{Q}_k^{(1)}\times\cdots\times\mathcal{Q}_k^{(|\mathcal{V}|)}}} \mathbf{1}^\top \Big(\frac{1}{Y} \sum_i [\mathbf{y}_{-1}]_i \hat{\mathbf{x}}_0^{(i)} \Big) \\
&= \frac{1}{Y^2} \sum_i [\mathbf{y}_{-1}]_i \Big(\min_{\hat{\mathbf{x}}_0^{(i)}\in\mathcal{Q}_k^{(i)}} \mathbf{1}^\top \hat{\mathbf{x}}_0^{(i)} \Big) = \frac{1}{Y^2} \sum_i [\mathbf{y}_{-1}]_i \beta_k^{(i)} Y \\
&\leq \frac{1}{Y} \sum_i [\mathbf{y}_{-1}]_i \beta_k = \beta_k,
\end{aligned}$$

where equality is attained when the Attacker initial state $\mathbf{y}_{-1}$ places all its robots at the node $i^* \in \arg\max_{i\in\mathcal{V}} \beta_k^{(i)}$. Hence, we have $\alpha_k = \beta_k$. □

Corollary 1 (No Incentive to Split)**.** *For a given graph $\mathcal{G}$ and total resources X and Y, the Attacker has a strategy to win the dDAB game by time step k if and only if it can do so using a no-splitting winning strategy.*

Proof. The sufficiency is given trivially. The necessity comes as a direct consequence of Theorem 5. If the Attacker can win a dDAB game by time step k with some strategy, we have that

$Y \geq \alpha_k X$. From Theorem 5, we obtain $Y \geq \alpha_k X = \beta_k X$, which implies that the Attacker can also win by time step k with a no-splitting strategy. □

Corollary 1 shows that allowing arbitrary splitting neither enables a breach that cannot already be achieved by a no-splitting strategy nor leads to an earlier breach. Consequently, the Attacker has no incentive to split its resources.

VI. Algorithmic Solution

In this section, we develop algorithms to numerically construct Q-sets and extract optimal actions from them. The Q-set construction algorithm also provides the basis for proving Theorem 2, which asserts that all Q-sets are polytopes.

A. Numerical Q-set Construction

Recall the recursive definition of the Q-sets in (15b):

$$\mathcal{Q}_k^{(i)} = \Big\{ \mathbf{x} \;\Big|\; \mathbf{x} \in \mathcal{P}_{\mathrm{req}}(\mathbf{y}^{(i)}) \wedge \mathcal{R}(\mathbf{x}) \cap \mathcal{Q}_{k-1}^{(j)} \neq \varnothing \;\forall j \in \mathcal{N}_i \Big\}.$$

To construct the Q-sets numerically, we begin by analyzing the structure of the set $\{\mathbf{x} \mid \mathcal{R}(\mathbf{x}) \cap \mathcal{Q}_k^{(j)} \neq \varnothing\}$, which contains all states from which the Defender can reach $\mathcal{Q}_k^{(j)}$ in a single time step. We formalize this notion by introducing the concept of the inverse reachable set:

Definition 10 (Inverse Reachable Set)**.** *Given a set $P \subseteq \mathbb{R}^N_{\geq 0}$, the inverse reachable set of P is defined as*

$$\mathcal{R}^{-1}(P) = \left\{ \mathbf{x} \;\middle|\; \mathcal{R}(\mathbf{x}) \cap P \neq \varnothing \right\}. \tag{27}$$

With the notion of the inverse reachable set, we can simplify the recursive construction of the Q-sets as

$$\mathcal{Q}_k^{(i)} = \Big(\bigcap_{j \in \mathcal{N}_i} \mathcal{R}^{-1}(\mathcal{Q}_{k-1}^{(j)}) \Big) \cap \mathcal{P}_{\mathrm{req}}(\mathbf{y}^{(i)}),\ \forall\ k \geq 1.$$

Note that any admissible action $K \in \mathcal{K}$ is reversible: if an action maps a state $\mathbf{x}_t$ to a successor $\mathbf{x}_{t+1}$, then there exists a corresponding reverse action that recovers $\mathbf{x}_t$ from $\mathbf{x}_{t+1}$. This reversibility offers an alternative interpretation of the inverse reachable set: as a forward reachable set under a reversed graph, in which all edge directions are inverted.

Definition 11. *For a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with connectivity matrix A, its reversed graph $\widetilde{\mathcal{G}} = (\mathcal{V}, \widetilde{\mathcal{E}})$ is defined based on the connectivity matrix $\widetilde{A} = A^\top$.*

We denote $\widetilde{\mathcal{K}}$ as the admissible action set of $\widetilde{\mathcal{G}}$. The reachable set for the reversed graph is then defined as

$$\widetilde{\mathcal{R}}(\mathbf{x}) = \left\{ \mathbf{x}' \mid \exists \widetilde{K} \in \widetilde{\mathcal{K}} \text{ s.t. } \mathbf{x}' = \widetilde{K}\mathbf{x} \right\}.$$

The equivalence between the inverse reachable set under the original graph $\mathcal{G}$ and the reachable set under the reversed graph follows directly.

Lemma 3. *For any graph $\mathcal{G}$, we have*

$$\mathcal{R}^{-1}(P) = \widetilde{\mathcal{R}}(P), \qquad \forall P \subseteq \mathbb{R}^N_{\geq 0}.$$

Using Lemma 3, we arrive at a computationally friendly expression for constructing the Q-sets via the forward reachability in the reversed graph:

$$\mathcal{Q}_0^{(i)} = \mathcal{P}_{\mathrm{req}}(\mathbf{y}^{(i)}), \tag{28a}$$

$$\mathcal{Q}_k^{(i)} = \Big(\bigcap_{j \in \mathcal{N}_i} \widetilde{\mathcal{R}}(\mathcal{Q}_{k-1}^{(j)}) \Big) \cap \mathcal{P}_{\mathrm{req}}(\mathbf{y}^{(i)}), \quad \forall\ k \geq 1. \tag{28b}$$

Building on the above results, we can now establish that all Q-sets are polytopes.

Proof of Theorem 2. Recall from Lemma 2 that the reachable set of a polytope remains a polytope. Given the recursive construction of the Q-sets in (28), and noting that all operations (reachable set computations and intersections) preserve polyhedrality, the result follows by induction on k. □

As a direct consequence of the polyhedral structure of the Q-sets, we obtain the following corollary:

Corollary 2. *The k-step CRR, α_k, is attained at one of the vertices of the Q-set.*

Proof. Since the Q-sets are polytopes, the optimization defining $\beta_k^{(i)}$ in (24) is a linear program. Furthermore, since the CRR is bounded from below by zero, an optimal solution is attainable and occurs at one of the vertices. □

B. Action Extraction from Q-sets

Recall that we have treated the next state $\mathbf{x}_{t+1}$ in the reachable set as the action chosen by the Defender at step t. Executing such a strategy in practice requires identifying a feasible action matrix $K_t \in \mathcal{K}$ that satisfies $\mathbf{x}_{t+1} = K_t \mathbf{x}_t$.

By construction of the reachable set $\mathcal{R}(\mathbf{x}_t)$, the existence of such an action is guaranteed. To compute K_t, we solve a linear system derived from the convex representation of $\mathcal{R}(\mathbf{x}_t)$, as characterized in Lemma 1. Specifically, we solve for the coefficients $\boldsymbol{\lambda} = (\lambda^{(\ell)})_{\ell=1}^{|\hat{\mathcal{K}}|}$ that satisfy

$$\Phi \boldsymbol{\lambda} = \mathbf{x}_{t+1}, \quad \boldsymbol{\lambda} \geq 0 \text{ and } \sum_\ell \lambda^{(\ell)} = 1, \tag{29}$$

where the matrix $\Phi \in \mathbb{R}^{|\mathcal{V}| \times |\hat{\mathcal{K}}|}$ has columns given by $\left(\hat{K}^{(\ell)} \mathbf{x}_t\right)_{\ell=1}^{|\hat{\mathcal{K}}|}$. With the obtained $\boldsymbol{\lambda}$, the feasible action that brings the Defender from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$ is given by

$$K_t = \sum_\ell \lambda^{(\ell)} \hat{K}^{(\ell)}.$$

VII. Numerical Illustrations

This section provides numerical examples that illustrate the results developed in the previous sections.

A. Q-set Propagation

Figure 8 illustrates how the Q-sets, $\mathcal{Q}_k^{(i)}$, change with the horizon k. For the three-node graph selected for this example, the propagation in Algorithm 1 converges after four iterations, at which point the algorithm finds that $k_\infty = 4$. The CRR for this graph is $\alpha_\infty = 3$.

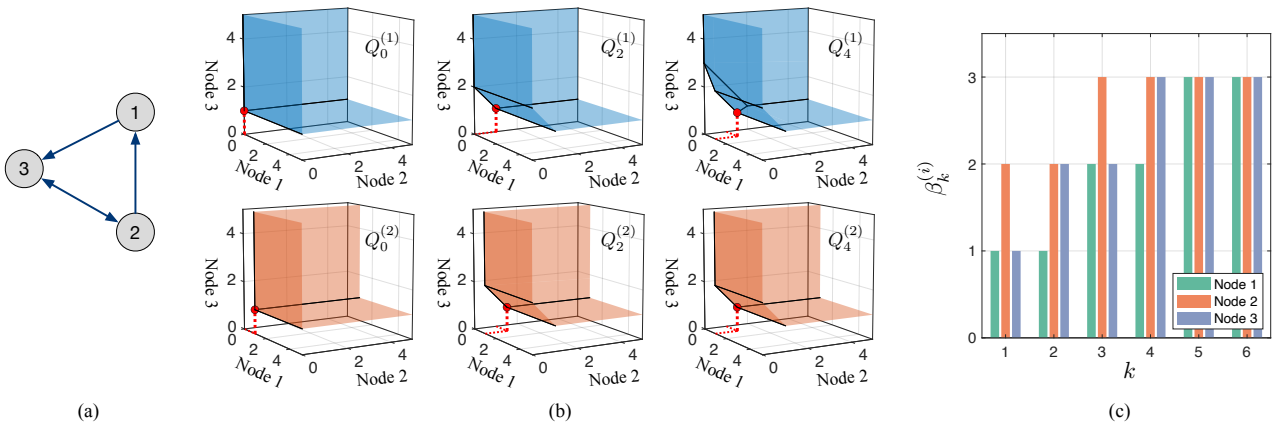

Fig. 8. Illustration of Q-sets evolution with different horizons k. (a) The three-node graph used for this example. (b) $\mathcal{Q}_k^{(i)}$ for nodes $i = \{1, 2\}$ and horizon $k = \{0, 2, 4\}$. The red dots indicate the element in the Q-set that achieves $\beta_k^{(i)}$. (c) The evolution of $\beta_k^{(i)}$ on each node, until they converge at $k_\infty = 6$.

It is worth noting that the Q-set for a given node may not change at every time step: e.g., $\mathcal{Q}_k^{(1)}$ changes only twice: once from $k = 1$ to 2, and then from $k = 3$ to 4.

We can verify the monotonicity of the Q-sets described in Remark 1 by observing how the Q-sets get "carved off" and become smaller as k increases. Specifically, some regions of the state space get excluded when k changes from 0 to 2 and similarly from 2 to 4. As an example, the state $\mathbf{x} = [0, 0, 1] \in \mathcal{Q}_0^{(1)}$ can guard against any immediate next action made by a unit Attacker at node 1 (i.e., $\mathbf{y}^{(1)}$). This is shown by the red dot in $\mathcal{Q}_0^{(1)}$ (top left subfigure in Fig. 8). However, this state is insufficient to defend over two time steps, and thus it is not included in $\mathcal{Q}_2^{(1)}$. The vertices of $\mathcal{Q}_4^{(1)}$ (top right subfigure) are $[0, 0, 3]$, $[1, 1, 1]$, $[1, 0, 2]$, and $[0, 2, 1]$. One can verify that any of these states, as well as any convex combination of these states are sufficient to guard against one Attacker robot indefinitely.

B. Effect of Edges on CRR

The relationship between the CRR and the graph structure is not straightforward. One might, for example, expect a positive correlation between the number of edges and the CRR, since an increase in the number of outgoing edges from a node increases the number of neighboring nodes that must be covered by the Defender. However, we show by a counter-example (found by the algorithm) that this is not the case.

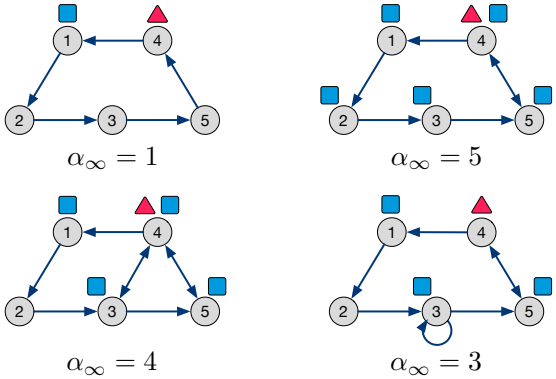

Fig. 9. Examples of how CRR changes with the graph structure (self-loops not assumed). The necessary and sufficient amount of blue robots are placed in the Q-set for the given red robot in each figure.

The following example illustrates how the addition of edges can drastically change the CRR. Fig. 9 provides examples of directed graphs with five nodes but with different edge sets. The corresponding indefinite-defense CRR, α_∞, for each graph is obtained using Algorithm 1.

In the simplest ring-graph instance, the Defender needs only a single robot to indefinitely defend against a single Attacker, consistent with prior results reported in [25]. Interestingly, if the edge between nodes 4 and 5 is made bidirectional, the CRR jumps to $\alpha_\infty = 5$, giving the Attacker a significant advantage. If we further add a bidirectional edge between nodes 3 and 4, the CRR decreases to 4, which benefits the Defender.

Finally, if instead of adding the edge between nodes 3 and 4 we introduce a self-loop at node 3, the CRR drops from 5 to 3. This observation highlights that some edges (e.g., the self-loop at node 3) have a larger impact on the game than others (e.g., the edge between nodes 3 and 4).

C. Non-integer Resource Ratio

Another natural conjecture regarding the CRR is that it is always integer-valued. However, the six-node dDAB example in Fig. 10 disproves this for *finite-horizon* dDAB games. Specifically, Algorithm 1 computes Q-sets that yield $\alpha_2 = 3.5$, indicating that while three Defender robots are insufficient for a two-step defense against a single Attacker, a (theoretical) allocation of 3.5 units of defensive effort suffices. In practice, one must deploy four Defender robots for a single Attacker. However, for two Attacker robots, only seven Defender robots are required.

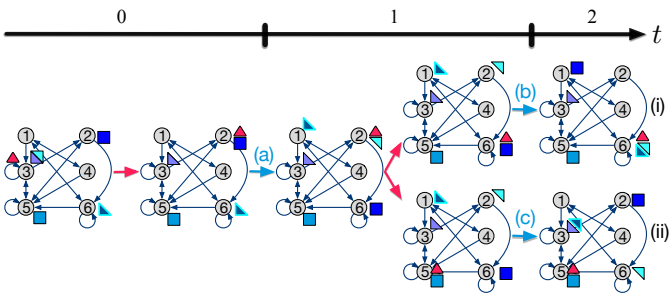

Fig. 10. The two time-step game tree over a six-node graph with explicit self-loops. The blue triangles represent half unit of Defender defense effort.

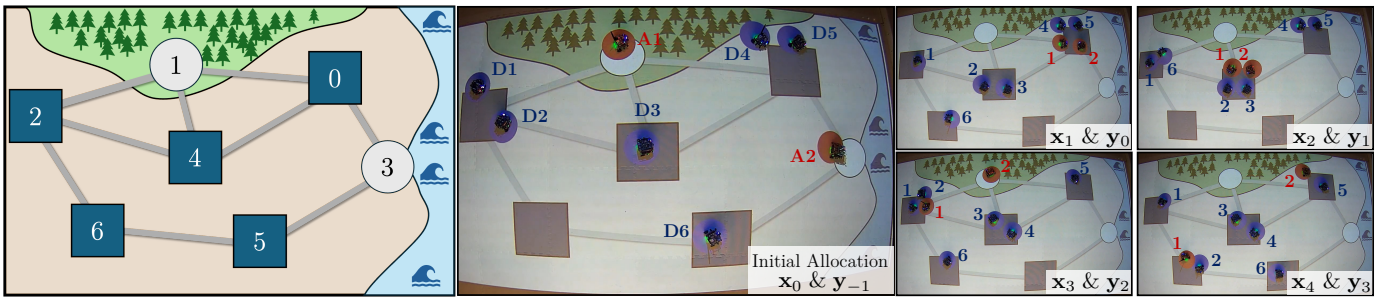

Fig. 11. Snapshots from hardware experiment for Scenario 1 with eight Defender robots against two Attacker robots. The first panel shows the underlying graph, and the remaining panels show the discrete allocations at the moments when the Attacker selects its next move. The Defender team consistently positions sufficient robots at nodes neighboring the Attacker robots' locations, preventing any potential breach after Attacker's next move.

To illustrate this behavior, we present a (partially shown) game tree where the Attacker starts at node 3, and the Defender's initial allocation is computed via Algorithm 6. Although the Attacker may initially move to either node 2 or node 5, we focus on the game branch where it moves to node 2.

After observing this Attacker move, the Defender applies action (a), which redistributes its fractional defensive effort at node 3, allocating half of that effort to node 1 while retaining the other half at node 3. This "fractional splitting" arises from the continuous Q-set analysis and represents idealized resource distribution rather than literal physical division of robots.

At $t = 1$, the Attacker can move to either node 5 or node 6. Suppose it moves to node 6, the Defender executes action (b), recombining the fractional allocations previously placed on nodes 1 and 3 to concentrate as sufficient defensive effort at node 6. This guarantees coverage regardless of whether the Attacker stays at node 6 or moves to node 5 next. A similar pattern of fractional splitting and regrouping appears in trajectory (ii).

By dynamically redistributing fractional defensive effort, the Defender succeeds using only an additional half unit of resource. This efficient strategy is generated automatically by the algorithms in Section IV, validating the proposed approach.

D. Experiments on a Multi-robot Testbed

We implement the proposed dDAB algorithm and the resulting Defender and Attacker strategies on a remotely accessible academic multi-robot testbed [28] to demonstrate the deployability on a physical multi-robot system. While the set-based dynamic program in (15) operates with continuous resources, the hardware experiments additionally demonstrate how the same algorithm can be used with discrete, embodied resources (mobile robots) via a simple discrete allocation strategy wrapper.

Specifically, at each time step, the Defender first selects a target Q-set $\mathcal{Q}_k^{(j)}$ based on the observed Attacker allocation, as described in line 6 of Algorithm 7. Given the current Defender *discrete* allocation $\mathbf{x}_t$, the wrapper first computes the intersection $\mathcal{R}(\mathbf{x}_t) \cap \mathcal{Q}_k^{(j)}$ and then selects a discrete point $\mathbf{x}_{t+1}$ (corresponding to X discrete robots) within the intersection. Based on the new discrete allocation, each Defender robot is assigned a target node to achieve the next discrete allocation $\mathbf{x}_{t+1}$, and the location of the assigned node is used as the robot's target waypoint. These waypoints are then sent to the control interface of the multi-robot testbed, where the built-in safety barrier certificates ensure collision-free execution.

We evaluate the implementation on two representative examples to highlight different operational scenarios of dDAB.

a) Scenario 1: This scenario emulates a broad-area outdoor defense task over a network of seven nodes. The five nodes marked with squares are the key nodes that the Defender needs to constantly maintain numerical advantage, while the the remaining two circular nodes are the Attacker's spawning nodes. The Attacker robots may appear from the forest or arrive from the sea at the two circular nodes and attempt to breach the defense at the square (key) nodes.

The Q-set computation indicates that four Defender robots are required to indefinitely hold off a single Attacker; accordingly, we first run an experiment with two Attacker robots (red) versus eight Defender robots (blue). Fig. 11 present snapshots of the experiment taken at the moments when the Attacker is about to select its next allocation. It can be observed that the neighboring nodes of those currently occupied by the Attacker robots consistently contain a sufficient number of Defender robots, ensuring successful defense regardless of the Attacker's next move.

Next, we repeat the experiment after removing one Defender robot from node 2. Under this reduced Defender team, Algorithm 5 predicts an earliest breach at $t = 2$ for the Attacker robot initiated from node 3. Figure 12 shows the movement sequence selected by the Attacker that leads to this breach at node 2, along with the corresponding Defender responses.

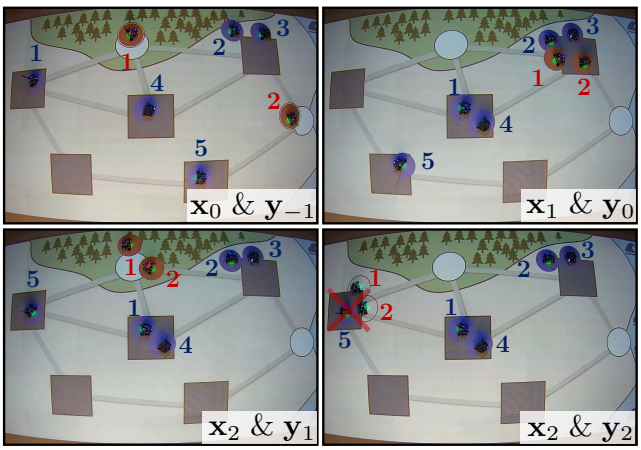

Fig. 12. Allocation sequence for Scenario 1 after removing one Defender robot. The Attacker robots achieve a breach at node 2 at time step $t = 3$.

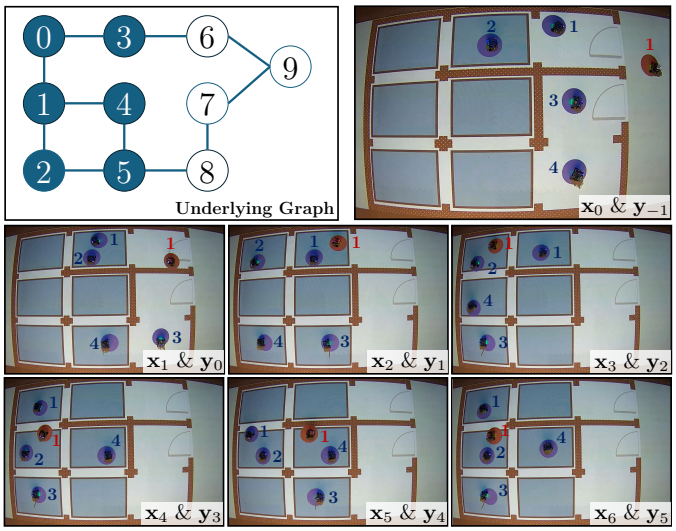

Fig. 13. Snapshots from the indoor surveillance experiment with four Defender robots against one Attacker. At each time step, the Defender robots maintain presence in the Attacker's room and its neighboring rooms, ensuring continuous surveillance.

b) Scenario 2: This scenario represents an indoor surveillance problem with nine rooms (nodes 0–8), of which six are key rooms (nodes 0–5). The Defender team must ensure that, at every time step, at least one defender is present in the same key room as the Attacker. We deploy four Defender robots to defend against a single Attacker robot—a configuration that, according to the Q-set analysis, guarantees indefinite defense. The Attacker is spawned outside the building at node 9, enters through node 6, and then explores the rooms at random. As shown in Fig. 13, at each time step there is always a Defender robot co-located with the Attacker and at least one Defender robot positioned in each neighboring room, thereby maintaining continuous surveillance throughout the experiment regardless of the Attacker's moves.

VIII. Conclusion

This work formulates a dynamic adversarial resource allocation problem by combining the Colonel Blotto game with population dynamics on graphs. Unlike instant allocations in traditional models, we require that players' resources traverse the graph edges. We develop an efficient reachable-set approach to predict state evolution and fully characterize the game by deriving necessary and sufficient conditions (Q-sets) for either player to win, along with the corresponding reactive strategies. Numerical simulations validate the efficacy of our approach. Future work will explore conditions for the convergence of the Q-prop algorithm, enabling guaranteed indefinite defense. Additionally, we aim to extend our framework to heterogeneous resources, as in [13], and decentralized decision-making using the common-information approach [29].

References

[1] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[2] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative Robots and Sensor Networks 2015*, pp. 31–51, 2015.

[3] Y. Xu, G. Gui, H. Gacanin, and F. Adachi, "A survey on resource allocation for 5G heterogeneous networks: Current research, future trends, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 668–695, 2021.

[4] X. Bei and S. Zhang, "Algorithms for trip-vehicle assignment in ride-sharing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[5] A. S. Nair, T. Hossen, M. Campion, D. F. Selvaraj, N. Goveas, N. Kaabouch, and P. Ranganathan, "Multi-agent systems for resource allocation and scheduling in a smart grid," *Technology and Economics of Smart Grids and Sustainable Energy*, vol. 3, no. 1, pp. 1–15, 2018.

[6] V. Anuradha and D. Sumathi, "A survey on resource allocation strategies in cloud computing," in *International Conference on Information Communication and Embedded Systems (ICICES2014)*, Chennai, India, 2014, pp. 1–7.

[7] D. Shishika, J. Paulos, and V. Kumar, "Cooperative team strategies for multi-player perimeter-defense games," *IEEE Robotics and Automation Letters.*, vol. 5, no. 2, pp. 2738–2745, 2020.

[8] M. R. Al-Eiadeh and M. Abdallah, "Genigraph: a genetic-based novel security defense resource allocation method for interdependent systems modeled by attack graphs," *Computers & Security*, vol. 144, p. 103927, 2024.

[9] W. Yuan, L. Zhao, and B. Zeng, "Optimal power grid protection through a defender–attacker–defender model," *Reliability Engineering & System Safety*, vol. 121, pp. 83–89, 2014.

[10] A. Ashok, M. Govindarasu, and J. Wang, "Cyber-physical attack-resilient wide-area monitoring, protection, and control for the power grid," *Proceedings of the IEEE*, vol. 105, no. 7, pp. 1389–1407, 2017.

[11] K. D. Julian and M. J. Kochenderfer, "Distributed wildfire surveillance with autonomous aircraft using deep reinforcement learning," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 8, pp. 1768–1778, 2019.

[12] S. Berman, A. Halász, M. A. Hsieh, and V. Kumar, "Optimized stochastic policies for task allocation in swarms of robots," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 927–937, 2009.

[13] A. Prorok, M. A. Hsieh, and V. Kumar, "The impact of diversity on optimal control policies for heterogeneous robot swarms," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 346–358, 2017.

[14] H. Ravichandar, K. Shaw, and S. Chernova, "STRATA: unified framework for task assignments in large teams of heterogeneous agents," *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 2, pp. 1–25, 2020.

[15] V. Tereshchuk, J. Stewart, N. Bykov, S. Pedigo, S. Devasia, and A. G. Banerjee, "An efficient scheduling algorithm for multi-robot task allocation in assembling aircraft structures," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3844–3851, 2019.

[16] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić, "Analysis of dynamic task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 225–241, 2006.

[17] B. Roberson, "The colonel Blotto game," *Economic Theory*, vol. 29, no. 1, pp. 1–24, 2006.

[18] R. Powell, "Sequential, nonzero-sum "Blotto": Allocating defensive resources prior to attack," *Games and Economic Behavior*, vol. 67, no. 2, pp. 611–615, 2009.

[19] R. Chandan, K. Paarporn, and J. R. Marden, "When showing your hand pays off: Announcing strategic intentions in colonel Blotto games," in *American Control Conference (ACC)*, Denver, CO, 2020, pp. 4632–4637.

[20] D. Kovenock and B. Roberson, "The optimal defense of networks of targets," *Economic Inquiry*, vol. 56, no. 4, pp. 2195–2211, 2018.

[21] K. Paarporn, R. Chandan, M. Alizadeh, and J. R. Marden, "Characterizing the interplay between information and strength in Blotto games," in *Conference on Decision and Control (CDC)*, Nice, France, 2019, pp. 5977–5982.

[22] K. A. Konrad, "Budget and effort choice in sequential colonel Blotto campaigns," *CESifo Economic Studies*, vol. 64, no. 4, pp. 555–576, 2018.

[23] M. Hajimirsaadeghi and N. B. Mandayam, "A dynamic colonel Blotto game model for spectrum sharing in wireless networks," in *Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2017, pp. 287–294.

[24] T. Klumpp, K. A. Konrad, and A. Solomon, "The dynamics of majoritarian Blotto games," *Games and Economic Behavior*, vol. 117, pp. 402–419, 2019.

[25] D. Shishika, Y. Guan, M. Dorothy, and V. Kumar, "Dynamic defender-attacker Blotto game," in *American Control Conference (ACC)*, Atlanta, GA, 2022, pp. 4422–4428.

[26] O. Gross and R. Wagner, "A continuous colonel Blotto game," RAND Corporation, Tech. Rep., 1950.

[27] Y. Guan, L. Pan, D. Shishika, and P. Tsiotras, "On the adversarial convex body chasing problem," in *2023 American Control Conference (ACC)*, San Diego, CA, 2023, pp. 435–440.
[28] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt, "The Robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *IEEE Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, 2020.
[29] Y. Guan, M. Afshari, and P. Tsiotras, "Zero-sum games between mean-field teams: Reachability-based analysis under mean-field sharing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024, pp. 15 930–15 937.

APPENDIX

Lemma 4 (Sufficiency of Q-sets)**.** *Let the Q-sets be defined in* (15)*, and suppose that the Attacker starts with* $\mathbf{y}_{t-1} = \mathbf{y}^{(i)}$*. Then, by selecting* $\mathbf{x}_t \in \mathcal{Q}_k^{(i)}$*, the Defender can defend at least until time step* $t + k$*.*

Proof. We provide a proof by induction.

Base Case: When $k = 0$, we have $\mathbf{x}_t \in \mathcal{Q}_0^{(i)} = \mathcal{P}_{\mathrm{req}}(\mathbf{y}^{(i)})$. From Proposition 1, defense is guaranteed at time t.

Inductive hypothesis: Suppose that for some $k \geq 1$, and for all $i \in \mathcal{V}$, the condition $\mathbf{x}_t \in \mathcal{Q}_k^{(i)}$ guarantees defense until time $t + k$ given that $\mathbf{y}_{t-1} = \mathbf{y}^{(i)}$.

Induction: Given $\mathbf{y}_{t-1} = \mathbf{y}^{(i)}$, we let $\mathbf{x}_t \in \mathcal{Q}_{k+1}^{(i)}$. Under the no-splitting strategy, suppose that the Attacker selects $\mathbf{y}_t = \mathbf{y}^{(j)}$, for some arbitrary $j \in \mathcal{N}_i$. The Attacker cannot immediately win with this (or any other) action since the Defender state $\mathbf{x}_t \in \mathcal{Q}_{k+1}^{(i)} \subseteq \mathcal{P}_{\mathrm{req}}(\mathbf{y}_{t-1})$ guarantees defense at time step t. After observing $\mathbf{y}_t = \mathbf{y}^{(j)}$, we let the Defender select its next state so that $\mathbf{x}_{t+1} \in \mathcal{R}(\mathbf{x}_t) \cap \mathcal{Q}_k^{(j)}$. This new state is reachable since $\mathbf{x}_t \in \mathcal{Q}_{k+1}^{(i)}$ ensures that $\mathcal{R}(\mathbf{x}_t) \cap \mathcal{Q}_k^{(j)} \neq \varnothing$ due to (15b). After the Defender's action, we are at a situation where $\mathbf{y}_t = \mathbf{y}^{(j)}$ and $\mathbf{x}_{t+1} \in \mathcal{Q}_k^{(j)}$. From the inductive hypothesis, the Defender can defend another k steps from this time on. The Defender can thus defend until time step $t+k+1$. □

Lemma 5 (Necessity of Q-sets)**.** *Let the Q-sets be defined in* (15)*, and suppose that the Attacker starts with* $\mathbf{y}_{t-1} = \mathbf{y}^{(i)}$*. If* $\mathbf{x}_t \notin \mathcal{Q}_k^{(i)}$*, the Attacker can win the game before, or at time step* $t + k$*.*

Proof. We prove this lemma via an inductive argument.

Base case: Suppose $\mathbf{x}_t \notin \mathcal{Q}_0^{(i)} = \mathcal{P}_{\mathrm{req}}(\mathbf{y}^{(i)})$. Then, by the construction of $\mathcal{P}_{\mathrm{req}}(\mathbf{y}^{(i)})$, there exists $j \in \mathcal{N}_i$ such that $\mathbf{y}_t = \mathbf{y}^{(j)}$ defeats $\mathbf{x}_t$ on node j, which corresponds to the Attacker's win at time t. If $\mathbf{x}_t \notin \mathcal{P}_{\mathrm{req}}(\mathbf{y}^{(i)})$, there exists at least one $\mathbf{y}_t \in \mathcal{R}(\mathbf{y}^{(i)})$ that breaches $\mathbf{x}_t$. Suppose this (potentially split) $\mathbf{y}_t$ defeats $\mathbf{x}_t$ on node j. Since we are starting from a concentrated state $\mathbf{y}_{t-1} = \mathbf{y}^{(i)}$, the Attacker can move all its resources to the same node j, and this concentrated state also breaches node j.

Inductive hypothesis: Suppose that, for all i, $\mathbf{x}_t \notin \mathcal{Q}_k^{(i)}$ implies that the Attacker with state $\mathbf{y}_{t-1} = \mathbf{y}^{(i)}$ can win the game before, or at time step $t + k$.

Induction: Let the Attacker start with $\mathbf{y}_{t-1} = \mathbf{y}^{(i)}$ and the Defender select $\mathbf{x}_t \notin \mathcal{Q}_{k+1}^{(i)}$. From the definition of $\mathcal{Q}_{k+1}^{(i)}$, we have either of the following two cases: (i) $\mathbf{x}_t \notin \mathcal{P}_{\mathrm{req}}(\mathbf{y}^{(i)})$, which leads to an immediate defeat at t; or (ii) there exists $j \in \mathcal{N}_i$, such that $\mathcal{R}(\mathbf{x}_t) \cap \mathcal{Q}_k^{(j)} = \varnothing$. In the latter case, the Attacker can move to $\mathbf{y}_t = \mathbf{y}^{(j)}$. Then, for all possible next Defender allocations $\mathbf{x}_{t+1} \in \mathcal{R}(\mathbf{x}_0)$, we have that $\mathbf{x}_{t+1} \notin \mathcal{Q}_k^{(j)}$. From the inductive hypothesis, the Attacker will win within k steps from this time $t + 1$. Thus, the Attacker can win the game before, or at time step $t + k + 1$. □