

# Hierarchical Decompositions of Stochastic Pursuit-Evasion Games

Yue Guan<sup>1</sup> Mohammad Afshari<sup>2</sup> Qifan Zhang<sup>3</sup> Panagiotis Tsiotras<sup>4</sup>

**Abstract**—In this work we present a hierarchical framework for solving discrete stochastic pursuit-evasion games (PEGs) in large grid worlds. Given a partition of the grid world into superstates, the proposed approach creates a two-resolution decision-making process, which consists of a set of local PEGs at the original state level and an aggregated PEG at the superstate level. With much smaller state spaces, both the local games and the aggregated game can be easily solved to Nash equilibria. Through numerical simulations, we show that the proposed hierarchical framework significantly reduces the computation overhead, while still maintaining a satisfactory performance.

## I. INTRODUCTION

Pursuit-evasion games (PEGs) [1] are a special class of dynamic games introduced in the 60s. An abundance of literature exists on the topic [2]–[4], where many of the formulations are in a continuous setting. Often, some form of discretization is proposed to cast the problem on a finite space. A common approach to discretize these continuous stochastic games is via the Markov chain approximation method (MCAM) [5], [6], which ensures that the discrete solution converges to the optimal policy of the original continuous game, as the discretization size approaches zero.

To achieve a good approximation, MCAM requires a fine discretization, which leads to discrete PEGs with a prohibitively large state space. For the zero-sum case we consider in this paper, the classic value iteration algorithm for finding Nash equilibrium (NE) requires solving a matrix game (equivalent to a linear program) at *each* state and at *each* iteration [7]. Even though efficient algorithms exist for solving linear programs (LPs) [8], the sheer number of LPs to be solved makes the classic “flat” approach without hierarchy extremely expensive. To address this challenge, we propose a hierarchical framework to decompose a large PEG into multiple smaller ones.

In the single-agent domain, hierarchical decision-making has seen many successes [9]–[11]. One of the fundamental concepts in this area is the concept of options [9]. An option is a generalization of the concept of action, which, upon execution, selects primitive actions (actions from the original action space) to achieve certain subgoals. Options can be

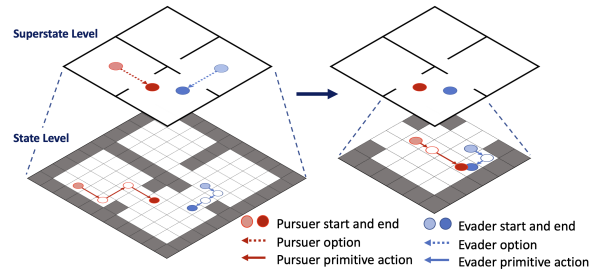


Fig. 1. A schematic of the proposed hierarchical approach. **Left** presents the aggregated PEG, where the game is played at the room level and the agents select options. **Right** is a local game restricted within a room. In this scenario, the PEG ends with a successful capture.

regarded as a set of useful behaviors that an agent can use directly off-the-shelf. By using options instead of primitive actions, an agent can shorten its planning horizon and break down complex tasks into multiple simpler subtasks.

Previous works [12]–[14] have extended the option framework and the hierarchical approach to the multi-agent domain. However, the majority of that research focuses on cooperative games [12] or team games [13], [14]; general results in the competitive setting are still lacking. The key challenge in the multi-agent scenario comes from the fact that the subgoal exists in the joint state space of all the agents. Consequently, a subgoal preferable to one agent may not be favorable to the others, especially in a competitive setting. As a result, it is not straightforward to extend single-agent hierarchical techniques to competitive multi-agent scenarios.

**Contributions:** The main contribution of this work is a hierarchical framework to solve large two-agent zero-sum PEGs. Under this framework, the grid world is first abstracted into multiple superstates (i.e., “rooms”). The proposed method then exploits the separable dynamics of the PEGs to construction *options* that navigate agents among the *individual* superstates. A competitive decision-making process is then constructed at two resolutions (see Fig. 1): **local PEGs** operating at the original resolution of the state space, each restricted within a superstate, and an **aggregated PEG** operating at the superstate-level, where the agents select options to navigate among the superstates. The games at different resolutions are connected through a value propagation scheme, where the Nash values of the local PEGs are used as the rewards for the aggregated game. The end result is a hierarchical policy that operates at two resolutions. Finally, the effectiveness of the proposed approach is demonstrated through numerical simulations, where the proposed hierarchical approach significantly reduces the computation overhead, while still achieving satisfactory performance when competing against flat Nash policies.

<sup>1</sup>Yue Guan is a PhD Candidate with the School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, USA. Email: yguan44@gatech.edu

<sup>2</sup>Mohammad Afshari is a Postdoctoral Fellow with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA, USA. Email: mafshari@gatech.edu

<sup>3</sup>Qifan Zhang is currently a Software Engineer at Google, Mountain View, CA, USA. This work was done while QZ was at Georgia Tech. Email: qzhang410@gatech.edu

<sup>4</sup>Panagiotis Tsiotras is the David & Andrew Lewis Chair and Professor with the School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, USA. Email: tsiotras@gatech.edu

## II. PROBLEM FORMULATION

We consider a discrete two-player pursuit-evasion game (PEG) over a grid world. An example is presented in Fig. 2, where a Pursuer (red) and an Evader (blue) inhabit a 11-by-11 grid world and the dark cells denote obstacles (e.g., walls). The Pursuer needs to capture the Evader by having the Evader within its capture zone (red dashed square), while the Evader tries to avoid the capture. The episode ends when the Evader is captured. To ensure capture, we assume that the Pursuer can move at most two cells in each of the four directions, while the Evader can move at most one cell. Together with the “no-move” action, the Pursuer has nine primitive actions, while the Evader has five. The reachable states for the two agents are shown in Fig. 2. If an agent intends to make a move that will hit the wall, it stays at its original position and no penalty is given.

Formally, we define a discrete pursuit-evasion game (PEG) as a tuple  $\mathcal{G} = \langle \mathcal{S}^1, \mathcal{S}^2, \mathcal{A}^1, \mathcal{A}^2, \mathcal{T}^1, \mathcal{T}^2, \mathcal{R}, \mathcal{F}, \beta \rangle$ . The discrete individual state space  $\mathcal{S}^i$  corresponds to the position of agent  $i$ , and the discrete individual action space  $\mathcal{A}^i$  denotes the actions agent  $i$  can take. We refer to these actions as the *primitive* actions. We use  $\mathcal{S} = \mathcal{S}^1 \times \mathcal{S}^2$  and  $\mathcal{A} = \mathcal{A}^1 \times \mathcal{A}^2$  to denote the joint state and action spaces. The terminal set  $\mathcal{F}$  consists of all joint states  $s = (s^1, s^2)$  that correspond to successful capture. The movement of the agents is characterized by the individual agent dynamics  $\mathcal{T}^i : \mathcal{S}^i \times \mathcal{A}^i \times \mathcal{S}^i \rightarrow [0, 1]$ , where  $\mathcal{T}^i(s^{i'}, |s^i, a^i)$  denotes the probability that agent  $i$  transitions from its current state (position)  $s^i$  to  $s^{i'}$  under the action  $a^i$ . The joint dynamics is given by  $\mathcal{T}(s'|s, a) = \mathcal{T}^1(s^{1'}|s^1, a^1)\mathcal{T}^2(s^{2'}|s^2, a^2)$  for all states  $s = (s^1, s^2) \notin \mathcal{F}$ , and  $\mathcal{T}(s|s, a) = 1$  for all joint action  $a \in \mathcal{A}$  if  $s \in \mathcal{F}$ . In other words, all terminal joint states in  $\mathcal{F}$  are absorbing. The function  $\mathcal{R} : \mathcal{S} \rightarrow [R_{\min}, R_{\max}]$  provides the reward at the joint state  $s \in \mathcal{S}$ . We assume a zero-sum reward structure and let the Pursuer be the maximizing Agent 1 and the Evader be the minimizing Agent 2. We further consider a sparse reward and assign  $\mathcal{R}(s) = +1$  for  $s \in \mathcal{F}$ , and  $\mathcal{R}(s) = 0$  otherwise. Finally,  $\beta \in (0, 1)$  is the discount factor.

Assuming full observation of the joint states, we first consider Markovian mixed policies for the two agents. Specifically, the policy  $\pi^i$  for agent  $i$  is a mapping  $\pi^i : \mathcal{A}^i \times \mathcal{S} \rightarrow [0, 1]$ , where  $\pi^i(a^i|s)$  gives the probability of choosing action  $a^i$  at the joint state  $s$ . Given a policy pair  $(\pi^1, \pi^2)$ , we denote the induced value at each state  $s \in \mathcal{S}$  as

$$\mathcal{V}^{\pi^1, \pi^2}(s) = \mathbb{E}^{\pi^1, \pi^2} \left[ \sum_{t=0}^{\infty} \beta^t \mathcal{R}(s_t) | s_0 = s \right], \quad (1)$$

where  $s_t = (s_t^1, s_t^2)$  is the joint state at time step  $t$ .

The Nash equilibrium (NE) is then defined as a policy pair  $(\pi^{1*}, \pi^{2*})$ , such that, at each joint state,  $s \in \mathcal{S}$ ,

$$\mathcal{V}^{\pi^1, \pi^{2*}}(s) \leq \mathcal{V}^{\pi^{1*}, \pi^{2*}}(s) \leq \mathcal{V}^{\pi^{1*}, \pi^2}(s), \quad (2)$$

for all admissible policies  $\pi^1$  and  $\pi^2$ . That is, there is no incentive for either agent to unilaterally deviate from a NE.

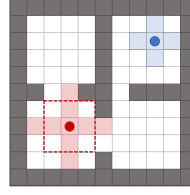


Fig. 2. A PEG example.

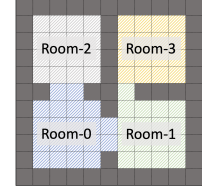


Fig. 3. The four “rooms”.

The Nash equilibrium can be solved via a value iteration algorithm [7] with the following update rules:

$$\begin{aligned} \mathcal{Q}_{k+1}(s, a^1, a^2) &= \mathcal{R}(s) + \beta \sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s, a^1, a^2) \text{Nash}(\mathcal{Q}_k(s')), \end{aligned} \quad (3)$$

where the subscript  $k$  indicates the iteration step and the Q-matrix  $\mathcal{Q}(s) \in \mathbb{R}^{|\mathcal{A}^1| \times |\mathcal{A}^2|}$  is defined as  $[\mathcal{Q}(s)]_{a^1, a^2} = \mathcal{Q}(s, a^1, a^2)$ . At each joint state  $s$ , the Nash value of the Q-matrix can be computed via the following linear program from Agent 1's perspective:

$$\begin{aligned} \max_{v, \pi^1(s)} \quad & v \\ \text{subject to} \quad & v\mathbf{1} - \pi^1(s)^\top \mathcal{Q}(s) \leq 0, \\ & \mathbf{1}^\top \pi^1(s) = 1, \quad \pi^1(s) \geq 0, \end{aligned} \quad (4)$$

where  $\pi^1(s) \in \mathbb{R}^{|\mathcal{A}^1|}$  is the policy of Agent 1 in vector form and  $\mathbf{1}$  is a column vector full of ones of compatible dimension. The solution  $v^*$  of (4) gives the Nash value, and  $\pi^{1*}(s)$  gives the Nash policy for Agent 1 at state  $s$ . The Nash policy for Agent 2 can be solved through a similar LP.

Although efficient algorithms exist to solve linear programs [8], there are still 4,624 states for the example shown in Fig. 2 (68 position states for each agent). That is, to compute the Nash equilibrium for this rather small problem one needs to solve 4,624 linear programs for *each* value iteration. The number of linear programs can easily get prohibitively large. For example, the last example in Fig. 7 requires almost one million LPs per iteration. Consequently, we seek an alternative approach that will allow us to decompose a large game into smaller pieces using a hierarchical approach.

The proposed hierarchical approach is motivated by the following intuitive observations: (i) when the two agents are far away, knowing the approximate position of the opponent is enough to make a decent decision, and at this stage, the agents do not need to perform fine maneuvers to achieve satisfactory performance; and (ii) when the two agents are close and a capture is imminent, the agents need the exact position to perform precise maneuvers to achieve (or avoid) capture. In this case, the agents can further restrict their attention to the local environment and ignore the rest of the grid world. Motivated by these two observations, we want to design intelligent agents that make decisions according to various resolutions of the state space, depending on the circumstances.

## III. THE HIERARCHICAL PEG

In this section, we will first partition the grid world domain (individual state space  $\mathcal{S}^i$ ) into smaller components, which

we refer to as the individual superstates (or “rooms,” see Fig. 3). A decision-making process is then constructed to operate at two different resolutions: (i) **superstate-level**, at which the two agents navigate among the “rooms” using options [9], and the Pursuer tries to be in the same room as the Evader; and (ii) **state-level**, where the two agents play a local PEG restricted within a superstate that contains a terminal state<sup>1</sup>. If capture occurs, the game terminates (visualized in Fig. 1); otherwise, the Evader escapes to another room, and the process reverses back to the aggregated game at the superstate level. The end product is a hierarchical policy, whose execution will be discussed in the next section.

### A. Superstates

We first formalize the definition of “rooms” through the concept of individual superstates. Let  $\Gamma^i = \{\gamma_1^i, \dots, \gamma_\ell^i\}$  ( $i = 1, 2$ ) be a partition of the *individual* state space  $\mathcal{S}^i$  for agent  $i$ , where each  $\gamma_k^i$ ,  $k \in \{1, 2, \dots, \ell\}$  is a subset of  $\mathcal{S}^i$ . We say that a partition  $\Gamma^i$  is an aggregated state space for agent  $i$  if  $\mathcal{S}^i = \bigcup_{k=1}^{\ell} \gamma_k^i$  and  $\gamma_j^i \cap \gamma_k^i = \emptyset$  for all  $j \neq k$ . We refer to the subset  $\gamma^i \in \Gamma^i$  as an individual *superstate*.

Next, we identify two important classes of individual states that serve as a medium through which agents transition from one individual superstate to another.

**Definition 1 (adopted from [15]):** The *periphery* and the *boundary* of an individual superstate  $\gamma^i \in \Gamma^i$  are defined as

$$\begin{aligned} \text{Peri}(\gamma^i) &= \{s^{i'} \notin \gamma^i \mid \exists s^i \in \gamma^i, a^i \in \mathcal{A}^i \text{ s.t. } \mathcal{T}^i(s^{i'}|s^i, a^i) > 0\}, \\ \text{Bndry}(\gamma^i) &= \{s^{i'} \in \gamma^i \mid \exists s^i \notin \gamma^i, a^i \in \mathcal{A}^i \text{ s.t. } \mathcal{T}^i(s^{i'}|s^i, a^i) > 0\}. \end{aligned}$$

The periphery and boundary for the Evader are visualized in Fig. 4, for the environment in Fig. 3.

**Definition 2:** An individual superstate  $\gamma^{i'}$  is *adjacent* to  $\gamma^i$ , denoted as  $\gamma^i \rightsquigarrow \gamma^{i'}$ , if  $\text{Peri}(\gamma^i) \cap \gamma^{i'} \neq \emptyset$ .

The joint aggregated state space is defined as the Cartesian product  $\Gamma = \Gamma^1 \times \Gamma^2$ , and the notions of periphery, boundary and adjacency extend naturally to the joint superstates.<sup>2</sup>

The joint aggregated state space will be used as the state space for the aggregated game as shown in Fig. 1.

### B. Options

To construct the corresponding macro-actions of the aggregated game that result in transitions among individual superstates we leverage the option framework [9]. The separable dynamics of PEGs allows us to generate options in the *individual* state space, which significantly alleviates the computational burden.

**Definition 3:** An *option* for agent  $i$  is a tuple

$$o^i = \langle \mathcal{S}_{o^i}^i, \mathcal{F}_{o^i}^i, \pi_{o^i}^i \rangle,$$

where  $\mathcal{S}_{o^i}^i \subset \mathcal{S}^i$  is the domain of the option;  $\mathcal{F}_{o^i}^i$  is the terminal set, and the option terminates once agent  $i$  reaches

<sup>1</sup>With the capture region in Fig. 2, a capture may happen when the two agents are in neighboring rooms. For conciseness, however, we visualize local games when the two agents are in the same room.

<sup>2</sup>For simplicity, we consider the case where  $\Gamma^1 = \Gamma^2$ . However, the proposed framework can be easily extended to the scenarios where different abstractions of the environment are used by the two agents.

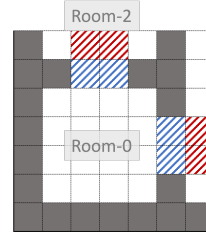


Fig. 4. Boundary (blue stripes) and Periphery (red stripes) of Room-0.

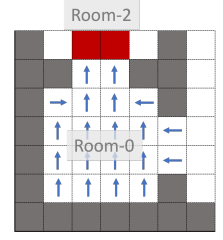


Fig. 5. The option for navigating from Room-0 to Room-2.

an individual state within  $\mathcal{F}_{o^i}^i$ ; finally,  $\pi_{o^i}^i : \mathcal{S}_{o^i}^i \rightarrow \mathcal{A}^i$  is the local policy, according to which agent  $i$  selects its primitive action given its state within the domain.

For each pair of adjacent individual superstates  $\gamma^i \rightsquigarrow \gamma^{i'}$ , we construct a Markov Decision Process (MDP) to generate the local policy of option  $o_{\gamma^i \rightsquigarrow \gamma^{i'}}$  that navigates agent  $i$  from individual states within  $\gamma^i$  to  $\gamma^{i'}$ . This local MDP is defined as  $\mathcal{M}_{\gamma^i \rightsquigarrow \gamma^{i'}} = \langle \mathcal{S}^i|_{\gamma^i}, \mathcal{A}^i, \mathcal{T}^i|_{\gamma^i}, \tilde{\mathcal{R}}_{\gamma^i \rightsquigarrow \gamma^{i'}}^i, \beta \rangle$ , where

- i)  $\mathcal{S}^i|_{\gamma^i} = \gamma^i \cup \text{Peri}(\gamma^i)$  is the restricted local state space;
- ii)  $\mathcal{A}^i$  is the original action space;
- iii)  $\mathcal{T}^i|_{\gamma^i}$  is the restricted transition kernel, such that, for all  $a^i \in \mathcal{A}^i$ ,

$$\begin{aligned} \mathcal{T}^i|_{\gamma^i}(s^{i'}|s^i, a^i) &= \mathcal{T}^i(s^{i'}|s^i, a^i), & \text{if } s^i \in \gamma^i, s^{i'} \in \mathcal{S}^i|_{\gamma^i} \\ \mathcal{T}^i|_{\gamma^i}(s^i|s^i, a^i) &= 1, & \text{if } s^i \in \text{Peri}(\gamma^i). \end{aligned}$$

- iv)  $\tilde{\mathcal{R}}_{\gamma^i \rightsquigarrow \gamma^{i'}}^i : \mathcal{S}^i|_{\gamma^i} \rightarrow \mathbb{R}$  is the local pseudo-reward, and

$$\begin{aligned} \tilde{\mathcal{R}}_{\gamma^i \rightsquigarrow \gamma^{i'}}^i(s^i) &= +1, & \text{if } s^i \in \text{Peri}(\gamma^i) \cap \gamma^{i'}, \\ \tilde{\mathcal{R}}_{\gamma^i \rightsquigarrow \gamma^{i'}}^i(s^i) &= 0, & \text{otherwise.} \end{aligned}$$

The local pseudo-reward provides an incentive for agent  $i$  to move to the periphery states in the target individual superstate  $\gamma^{i'}$ . The local MDP  $\mathcal{M}_{\gamma^i \rightsquigarrow \gamma^{i'}}$  is a *single-agent* problem and can be easily solved via value iterations or policy iterations<sup>3</sup>. The resulting optimal policy  $\pi^{i*} : \mathcal{S}^i|_{\gamma^i} \rightarrow \mathcal{A}^i$  is then assigned as the local policy for option  $o_{\gamma^i \rightsquigarrow \gamma^{i'}}$ , which operates on the individual state space.

We set the domain for option  $o_{\gamma^i \rightsquigarrow \gamma^{i'}}$  as  $\gamma^i \subset \mathcal{S}^i$ , and the terminal set as  $\text{Peri}(\gamma^i)$ . Consequently, agent  $i$  can only initiate option  $o_{\gamma^i \rightsquigarrow \gamma^{i'}}$  within superstate  $\gamma^i$  and the option automatically terminates when the agent leaves  $\gamma^i$ . An example of the Evader’s ‘Room-0 To Room-2’ option is presented in Fig 5. The arrows are the actions taken by the local policy at each specific cell within the room, and the red cells are the target periphery states within the individual superstate ‘Room-2’.

For each adjacent pair of individual superstates, we solve for all such options, and we use  $\mathcal{O}^i$  to denote the set of all these options for agent  $i$ . Furthermore, we let  $\mathcal{O}^i(\gamma)$  denote the options available to agent  $i$  at the joint superstate  $\gamma$ .

### C. Local Games

One key ingredient still missing for the aggregated game is the rewards. As the aggregated game transitions to a local game when the two agents are roughly in the same room<sup>1</sup>,

<sup>3</sup>Both agent maximizes the pseudo-reward to reach the target superstate.

we use the Nash value of the local PEGs restricted within a room to inform the decision-making at the superstate level.

Formally, we construct local games for each of the *joint* superstates  $\gamma$  such that  $\gamma \cap \mathcal{F} \neq \emptyset$ , where  $\mathcal{F}$  is the set terminal states (i.e., successful capture). A local game, restricted to a joint superstate  $\gamma$ , is defined as a tuple  $\mathcal{G}_\gamma = \langle \mathcal{S}|_\gamma, \mathcal{A}^1, \mathcal{A}^2, \mathcal{T}_\gamma, \mathcal{R}_\gamma, \beta \rangle$ , where

- i)  $\gamma \cap \mathcal{F} \neq \emptyset$ .
- ii)  $\mathcal{S}|_\gamma$  is the restricted local joint state space, defined as

$$\mathcal{S}|_\gamma = \gamma \cup \text{Peri}(\gamma).$$

- iii)  $\mathcal{A}^i$  ( $i = 1, 2$ ) are the original action spaces.
- iv)  $\mathcal{T}_\gamma$  is the joint transition restricted to the local state space, given by

$$\begin{aligned} \mathcal{T}_\gamma(s'|s, a^1, a^2) &= \mathcal{T}(s'|s, a^1, a^2), \text{ if } s \in \gamma, s' \in \mathcal{S}|_\gamma \\ \mathcal{T}_\gamma(s'|s, a^1, a^2) &= 1, \text{ if } s \in \text{Peri}(\gamma). \end{aligned}$$

- v)  $\mathcal{R}_\gamma$  is the local reward, and

$$\mathcal{R}_\gamma(s) = \mathcal{R}(s), \quad a^i \in \mathcal{A}^i, s \in \mathcal{S}|_\gamma.$$

A local game terminates<sup>4</sup> once the agents leave the joint superstate  $\gamma$ . Consequently, the local games have absorbing peripheries. With a much smaller state space, these local games can be easily solved via value iterations as in (3)-(4). We denote the local Nash value as  $\mathcal{V}_\gamma^*$  and the local Nash policies as  $\pi_\gamma^{i*}$ . The local Nash policy  $\pi_\gamma^{i*}$  governs the behaviors of agent  $i$  within  $\gamma$ . The Nash values will be used as the rewards for the aggregated game in the next subsection.

#### D. Aggregated Game

The aggregated game operates over the joint superstates. Instead of directly selecting a primitive action, the agent in the aggregated game selects an option based on the current joint superstate observation. First, we define the transition probabilities between the joint superstates resulting from the options introduced in Section III-B.

We leverage the discounted multi-step transition kernel [9] to properly address the different timescale on which the aggregated game operates. With the convention  $s = (s^1, s^2)$  and  $s' = (s'^1, s'^2)$ , the discounted multi-step transition probability between two joint superstates  $\gamma$  and  $\gamma'$  can be computed as

$$\tilde{\mathcal{T}}^\beta(\gamma'|\gamma, o^1, o^2) = \frac{1}{|\text{Bndry}(\gamma)|} \sum_{s \in \text{Bndry}(\gamma)} \phi(\gamma'|s, o^1, o^2), \quad (5)$$

where,

$$\begin{aligned} \phi(\gamma'|s, o^1, o^2) &= \beta \sum_{s' \in \gamma'} \mathcal{T}(s'|s, \pi_{o^1}^1(s^1), \pi_{o^2}^2(s^2)) \\ &\quad + \beta \sum_{s' \in \gamma} \mathcal{T}(s'|s, \pi_{o^1}^1(s^1), \pi_{o^2}^2(s^2)) \phi(\gamma'|s', o^1, o^2). \end{aligned} \quad (6)$$

<sup>4</sup>If a capture happens, the PEG terminates. Otherwise, the Evader escaped to another superstate, and we transition back to the aggregated game.

The computation performed in (5) is equivalent to: (i) set  $\text{Peri}(\gamma)$  as absorbing, (ii) start the Markov Chain induced by  $(\pi_{o^1}^1, \pi_{o^2}^2)$  with a uniform distribution over  $\text{Bndry}(\gamma)$ <sup>5</sup>, and (iii) compute the probability of the Markov Chain ending up in joint superstate  $\gamma'$  and properly discount the probability based on the arrival time. See the full version [16] for details.

With the multi-step transitions, we can define the aggregated game  $\mathcal{G}_\Gamma = \langle \Gamma^1, \Gamma^2, \mathcal{O}^1, \mathcal{O}^2, \tilde{\mathcal{T}}^\beta, \tilde{\mathcal{R}}, \beta \rangle$ , where

- i)  $\Gamma = \Gamma^1 \times \Gamma^2$  is the joint aggregated state space.
- ii)  $\mathcal{O}^i$  is the set of options constructed for agent  $i$  to navigate between its individual superstates.
- iii)  $\tilde{\mathcal{T}}^\beta$  is the discounted transition kernel in (5)-(6).
- iv)  $\tilde{\mathcal{R}}$  is the aggregated reward, given by

$$\begin{aligned} \tilde{\mathcal{R}}(\gamma) &= \frac{1}{|\text{Bndry}(\gamma)|} \sum_{s \in \text{Bndry}(\gamma)} \mathcal{V}_\gamma^*(s), \quad \text{if } \gamma \cap \mathcal{F} \neq \emptyset, \\ \tilde{\mathcal{R}}(\gamma) &= 0, \quad \text{if } \gamma \cap \mathcal{F} = \emptyset, \end{aligned}$$

where  $\mathcal{V}_\gamma^*$  is the Nash value of the local game within  $\gamma$ .

Again, the aggregated game can be solved via value iterations, and we denote the Nash policies of the aggregated game as  $\pi_\Gamma^{i*}$  ( $i = 1, 2$ ).

#### E. Proposed Algorithm

We summarize the hierarchical approach in Algorithm 1.

---

##### Algorithm 1: Hierarchical Decomposition Algorithm

---

- 1 **Inputs:** Game  $\mathcal{G}$ , Individual State Space Partition  $\Gamma^i$  ;  
/\* Superstates and options \*/
  - 2 Generate the superstates [Section III-A] ;
  - 3 Generate option  $o_{\gamma^i \rightsquigarrow \gamma'^i}$  for each agent  $i$  and each adjacent individual superstate pair [Section III-B] ;  
/\* Local games \*/
  - 4 Generate local games for joint superstate  $\gamma$  if  $\gamma \cap \mathcal{F} \neq \emptyset$  [Section III-C];
  - 5 Solve the Nash equilibrium of the local games;  
/\* Aggregated game \*/
  - 6 Construct the aggregated game with  $\mathcal{O}^i$  as the action space and  $\mathcal{V}_\gamma^*$  as the rewards [Section III-D];
  - 7 Solve the NE of the aggregated game;
  - 8 **Outputs:** Option sets  $\mathcal{O}^i$ ; Local game NE  $\pi_\gamma^{i*}$ ;  
Aggregated game NE  $\pi_\Gamma^{i*}$ .
- 

#### F. Complexity Analysis

Consider a grid world with  $M \times M$  rooms, each room with  $N \times N$  cells. The flat Nash approach needs to compute  $\mathcal{O}(M^4 N^4)$  matrix games for each value iteration. For the hierarchical approach, the aggregated game has  $M^2$  individual superstates, and there are at most  $5M^2$  local games to be solved<sup>1</sup>, each with  $N^4$  states. Consequently, the hierarchical approach solves a total of  $\mathcal{O}(M^2 N^4 + M^4)$  matrix games solved per iteration. For more details see the full version [16].

<sup>5</sup>Since the boundary states are the entry points to a superstate and no prior knowledge is given regarding how agents approach this superstate, a uniform distribution over the boundary is considered.

#### IV. THE HIERARCHICAL POLICY

The hierarchical policy constructed under the proposed framework consists of policies in two resolutions: (i) the aggregated Nash policy  $\pi_{\Gamma}^{i*}$  observes the *joint* superstate  $\gamma$  and selects an option  $o^i \in \mathcal{O}^i(\gamma)$ ; once an option  $o^i$  is selected, the local policy  $\pi_{o^i}^i(s^i)$  observes the *individual state*  $s^i$  and selects primitive actions; (ii) When the system transitions to a joint superstate  $\gamma$  that contains a local game, the local Nash policy  $\pi_{\gamma}^{i*}(s)$  is deployed, and the agents directly selects its primitive action based on state observation. If the system later leaves the superstate that contains a local game, the process reverse back to the aggregated game. Algorithm 2 presents the execution of the proposed hierarchical policy.

---

##### Algorithm 2: Execution of the Hierarchical Policy

---

```

1 Inputs: Option sets  $\mathcal{O}^i$ , Local game NEs  $\pi_{\gamma}^{i*}$ ,
   Aggregated game NE  $\pi_{\Gamma}^{i*}$ , ( $i = 1, 2$ );
2 while PEG not terminated do
3   Observe current superstate  $\gamma$ ;
4   if  $\gamma$  contains a local game then
5     while Local PEG not terminated do
6       Select  $a^i$  according to  $\pi_{\gamma}^{i*}(s)$ ;
7       Environment step forward with  $(a^1, a^2)$ ;
8     end
9   else
10    Select option  $o^i$  according to  $\pi_{\Gamma}^{i*}(\gamma)$ ;
11    while Not leaving joint  $\gamma$  do
12      Select  $a^i$  according to  $\pi_{o^i}^i(s^i)$ ;
13      Environment step forward with  $(a^1, a^2)$ 
14    end
15 end

```

---

The while loop in lines 5 to 9 makes the hierarchical policy non-Markovian, since the primitive actions used depend on the option the agent has selected when entering the superstate. At line 12, we let the agents terminate their old options when the joint superstate makes a transition<sup>6</sup>. Note that this “any” condition [14] is consistent with the definition of the aggregated transitions in (5).

An example of the execution of the proposed hierarchical policy is presented in Fig. 6. The agents first start with the aggregated game and select their options. The system then transitions to a local game, where the agents select primitive actions according to the local Nash. Finally, the Evader escaped the room, and the system transitions back to the aggregated game, and the process continues.

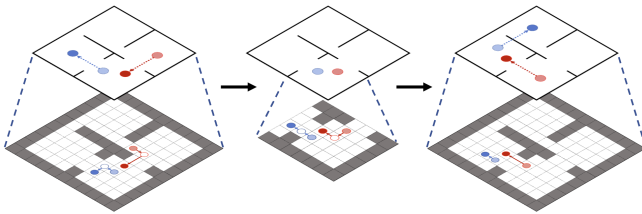


Fig. 6. An example of the execution of the proposed hierarchical policy.

<sup>6</sup>Equivalent to *any* one of the agents reaches the terminal set of its option.

#### V. NUMERICAL RESULTS

In this section, we verify the efficacy of the proposed algorithm through numerical simulations. The four PEGs we used are presented in Fig. 7.

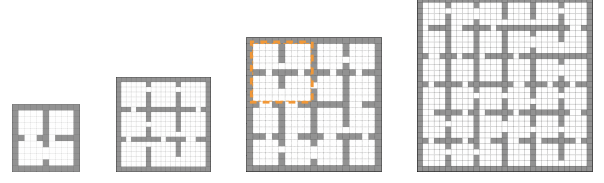


Fig. 7. The test grid worlds:  $2 \times 2$  (4.7K joint states),  $3 \times 3$  (56.6K joint states),  $4 \times 4$  (78.9K joint states), and  $6 \times 6$  (0.9M joint states).

Table I shows the computation time using the flat method and the proposed hierarchical approach. The computation time is significantly reduced through the hierarchical decomposition, while the majority of the computation time is spent on the computation of NEs of the local games. This result is expected, as we have multiple local games and they are all executed at the finest resolution. However, the local games can be easily solved in parallel, as the local games are independent from each other.

TABLE I  
COMPUTATION TIME COMPARISON [IN SECONDS]

| PEG Size        | $2 \times 2$ | $3 \times 3$ | $4 \times 4$ | $6 \times 6$ |
|-----------------|--------------|--------------|--------------|--------------|
| Flat            | 380.62       | 916.85       | 7068.72      | 37412.62     |
| Hierarchical    | 477.46       | 638.91       | 2095.70      | 4501.27      |
| - Option        | 0.92         | 2.07         | 4.70         | 9.70         |
| - Local Game    | 462.54       | 610.98       | 2034.33      | 4257.70      |
| - Abstract Game | 14.00        | 25.86        | 56.67        | 233.87       |

Fig. 8 illustrates the trend of computation time vs. state space size. The computation time of the hierarchical approach grows slower with respect to the size of the PEG than the flat approach, which confirms the motivation of this work.

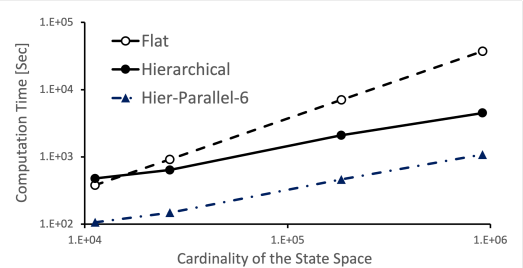


Fig. 8. The log-log plot of the computation time vs. state space size trend.

Table II compares the performance of the hierarchical policy versus the flat Nash policy. For each grid world, we ran 2,000 episodes, with randomly selected initial positions for the Pursuer and the Evader. We used the average number of transitions till capture as the performance metric. When the hierarchical pursuer competed against the Nash evader, it took about 15% more steps to capture compared to a Nash pursuer. This performance drop is expected, since the Nash pursuit policy is, by definition, the best response to the Nash evasion policy, but the relative performance drop decreases when the state space gets larger.

One of the causes for the sub-optimal performance of the hierarchical approach is the information loss due to agents making decisions based on the superstates. To better

TABLE II  
AVERAGE # STEPS TILL CAPTURE

| P vs. E       | 2 × 2       | 3 × 3        | 4 × 4        | 6 × 6        |
|---------------|-------------|--------------|--------------|--------------|
| Nash vs. Nash | 5.41        | 13.12        | 18.42        | 25.85        |
| Hier vs. Nash | 6.62 (+22%) | 15.55 (+18%) | 20.61 (+12%) | 28.18 (+9%)  |
| Nash vs. Hier | 4.47 (-18%) | 10.86 (-17%) | 16.26 (-12%) | 23.05 (-10%) |

illustrate this point, we present two trajectories in Fig. 9. The trajectories are truncated from the 4×4 grid world, with the four rooms corresponds to the rooms in orange in Fig. 7. The right subplot presents a trajectory of a Nash Pursuer against a Nash Evader. The Nash Pursuer utilizes its speed advantage and captures the Evader in seven steps.

The left subplot in Fig. 9 presents the trajectory of a hierarchical Pursuer against a Nash Evader. At timestep 3 (highlighted with stripes), given the speed advantage, the Pursuer should directly chase down the Evader through Room-B. However, at that moment, the hierarchical pursuer is still playing the aggregated game. Based only on the room information, the Pursuer does not know the exact location of the Evader within room-D (light orange area). Furthermore, if the Evader is at the bottom part of room-D, it could easily escape to one of the other rooms. Consequently, the Pursuer follows the sub-optimal route down to Room-C and then come to Room-D to defend. Such behavior is inevitable owing to the abstraction.

Finally, we examine the impact of the aggregation size. Instead of having one room as one individual superstate, we have a  $K \times K$  block of rooms as a single individual superstate. The performance for different aggregation size is presented in Table III. Note that when all rooms coalesce into a single superstate, the original game is recovered as a single local game, and we recover the same performance as the flat Nash policy. On the other extreme, where a single state is treated as a superstate, we can recover the original game as the aggregated game, under some additional assumptions.

When each superstate contains more states, the corresponding local games would contain a larger region of the state space and the resulting hierarchical agents are better informed in the local PEG. However, a larger superstate also leads to a lower resolution for the aggregated game, and the hierarchical agents ignore more information regarding the opponent’s position when selecting options, which leads to a degraded performance in the aggregated game. The trade-off between the performance of the local games and the aggregated game is not observed in this example. Future work will further investigate this trade-off.

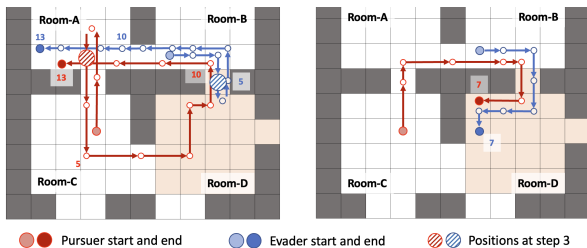


Fig. 9. A comparison of sample trajectories: **left** is Hierarchical Pursuer vs. Nash Evader, and **right** is Nash Pursuer vs. Nash Evader.

TABLE III  
PERFORMANCE WITH DIFFERENT AGGREGATIONS

| P vs. E       | each individual superstate contains |              |              |          |
|---------------|-------------------------------------|--------------|--------------|----------|
|               | 1 room                              | 4 rooms      | 9 rooms      | 36 rooms |
| Nash vs. Nash | 25.85                               | 25.85        | 25.85        | 25.85    |
| Hier vs. Nash | 28.18 (+9%)                         | 28.79 (+11%) | 30.54 (+18%) | 25.79    |
| Nash vs. Hier | 23.05 (-10%)                        | 21.55 (-16%) | 17.43 (-32%) | 25.82    |

## VI. CONCLUSION

In this work, we proposed a hierarchical framework to decompose a large pursuit-evasion game in a grid world. The proposed approach constructs a two-resolution decision making process, which consists of a set of local PEGs at the original state level and an aggregated PEG at the superstate level. With this hierarchy, the decomposed PEGs have much smaller state spaces and can be easily solved to Nash equilibria. Through numerical simulations, we showed that the proposed approach significantly reduced the computation overhead compared to the non-hierarchical approach, while still maintaining a good level of performance. Future work will investigate the theoretical bounds for the sub-optimality induced by the hierarchical decomposition.

## REFERENCES

- [1] R. Isaacs, *Differential Games*. John Wiley and Sons, 1965.
- [2] T. Başar, *Dynamic Noncooperative Game Theory*. SIAM, 1998.
- [3] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry, “Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation,” *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 662–669, 2002.
- [4] D. W. Oyler, P. T. Kabamba, and A. R. Girard, “Pursuit–evasion games in the presence of obstacles,” *Automatica*, vol. 65, pp. 1–11, 2016.
- [5] H. J. Kushner and P. G. Dupuis, *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer Science & Business Media, 2001, vol. 24.
- [6] Y. Guan, D. Maity, C. M. Kroninger, and P. Tsiotras, “Bounded-rational pursuit-evasion games,” in *American Control Conference*, New Orleans, LA, May, 25–28, 2021, pp. 3216–3221.
- [7] J. Filar and K. Vrieze, *Competitive Markov Decision Processes*. Springer, 2012.
- [8] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Athena Scientific Belmont, MA, 1997, vol. 6.
- [9] R. S. Sutton, D. Precup, and S. Singh, “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning,” *Artificial Intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.
- [10] T. G. Dietterich, “Hierarchical reinforcement learning with the MaxQ value function decomposition,” *Journal of Artificial Intelligence Research*, vol. 13, pp. 227–303, 2000.
- [11] A. Levy, G. Konidaris, R. Platt, and K. Saenko, “Learning multi-level hierarchies with hindsight,” *arXiv preprint arXiv:1712.00948*, 2017.
- [12] M. Ghavamzadeh, S. Mahadevan, and R. Makar, “Hierarchical multi-agent reinforcement learning,” *Autonomous Agents and Multi-Agent Systems*, vol. 13, no. 2, pp. 197–229, 2006.
- [13] J. Yang, I. Borovikov, and H. Zha, “Hierarchical cooperative multi-agent reinforcement learning with skill discovery,” *arXiv preprint arXiv:1912.03558*, 2019.
- [14] R. Makar, S. Mahadevan, and M. Ghavamzadeh, “Hierarchical multi-agent reinforcement learning,” in *Proceedings of the Fifth International Conference on Autonomous Agents*, 2001, pp. 246–253.
- [15] T. Dean and S.-H. Lin, “Decomposition techniques for planning in stochastic domains,” in *IJCAI*, vol. 2. Citeseer, 1995, p. 3.
- [16] Y. Guan, M. Afshari, Q. Zhang, and P. Tsiotras, “Hierarchical decompositions of stochastic pursuit-evasion games,” *arXiv preprint arXiv:2205.00885*, 2022.