Security-Aware Synthesis Using Delayed-Action Games[*]

Mahmoud Elfar[0000−0002−5579−1255], Yu Wang[0000−0002−0431−1039], and Miroslav Pajic[0000−0002−5357−0117]

Duke University, Durham NC 27708, USA
{mahmoud.elfar,yu.wang094,miroslav.pajic}@duke.edu

Abstract. Stochastic multiplayer games (SMGs) have gained attention in the field of strategy synthesis for multi-agent reactive systems. However, standard SMGs are limited to modeling systems where all agents have full knowledge of the state of the game. In this paper, we introduce delayed-action games (DAGs) formalism that simulates hidden-information games (HIGs) as SMGs, where hidden information is captured by delaying a player's actions. The elimination of private variables enables the usage of SMG off-the-shelf model checkers to implement HIGs. Furthermore, we demonstrate how a DAG can be decomposed into subgames that can be independently explored, utilizing parallel computation to reduce the model checking time, while alleviating the state space explosion problem that SMGs are notorious for. In addition, we propose a DAG-based framework for strategy synthesis and analysis. Finally, we demonstrate applicability of the DAG-based synthesis framework on a case study of a human-on-the-loop unmanned-aerial vehicle system under stealthy attacks, where the proposed framework is used to formally model, analyze and synthesize security-aware strategies for the system.

1 Introduction

Stochastic multiplayer games (SMGs) are used to model reactive systems where nondeterministic decisions are made by multiple players [4,13,23]. SMGs extend probabilistic automata by assigning a player to each choice to be made in the game. This extension enables modeling of complex systems where the behavior of players is unknown at design time. The *strategy synthesis* problem aims to find a *winning strategy*, i.e., a strategy that guarantees that a set of objectives (or winning conditions) is satisfied [6,21]. Algorithms for synthesis include, for instance, value iteration and strategy iteration techniques, where multiple reward-based objectives are satisfied [2,9,17]. To tackle the state-space explosion problem, [29] presents an *assume-guarantee* synthesis framework that relies on synthesizing strategies on the component level first, before composing them into a global winning strategy. Mean-payoffs and ratio rewards are further investigated in [3]

[*] This work was supported by the NSF CNS-1652544 grant, as well as the ONR N00014-17-1-2012 and N00014-17-1-2504, and AFOSR FA9550-19-1-0169 awards.

to synthesize ε-optimal strategies. Formal tools that support strategy synthesis via SMGs include PRISM-games [7,19] and Uppaal Stratego [10].

SMGs are classified based on the number of players that can make choices at each state. In *concurrent* games, more than one player is allowed to concurrently make choices at a given state. Conversely, *turn-based* games assign one player at most to each state. Another classification considers the information available to different players across the game [27]. *Complete-information* games (also known as *perfect-information* games [5]) grant all players complete access to the information within the game. In *symmetric* games, some information is equally hidden from all players. On the contrary, *asymmetric* games allow some players to have access to more information than the others [27].

This work is motivated by security-aware systems in which stealthy adversarial actions are potentially hidden from the system, where the latter can probabilistically and intermittently gain full knowledge about the current state. While hidden-information games (HIGs) can be used to model such systems by using private variables to capture hidden information [5], standard model checkers can only synthesize strategies for (full-information) SMGs; thus, demanding for alternative representations. The equivalence between turn-based semi-perfect information games and concurrent perfect-information games was shown [5]. Since a player's strategy mainly rely on full knowledge of the game state [9], using SMGs for synthesis produces strategies that may violate synthesis specifications in cases where required information is hidden from the player. *Partially-observable* stochastic games (POSGs) allow agents to have different belief states by incorporating uncertainty about both the current state and adversarial plans [15]. Techniques such as active sensing for online replanning [14] and grid-based abstractions of belief spaces [24] were proposed to mitigate synthesis complexity arising from partial observability. The notion of *delaying actions* has been studied as means for gaining information about a game to improve future strategies [18,30], but was not deployed as means for hiding information.

To this end, we introduce delayed-action games (DAGs) — a new class of games that simulate HIGs, where information is hidden from one player by delaying the actions of the others. The omission of private variables enables the use of off-the-shelf tools to implement and analyze DAG-based models. We show how DAGs (under some mild and practical assumptions) can be decomposed into subgames that can be independently explored, reducing the time required for synthesis by employing parallel computation. Moreover, we propose a DAG-based framework for strategy synthesis and analysis of security-aware systems. Finally, we demonstrate the framework's applicability through a case study of security-aware planning for an unmanned-aerial vehicle (UAV) system prone to stealthy cyber attacks, where we develop a DAG-based system model and further synthesize strategies with strong probabilistic security guarantees.

The paper is organized as follows. Sec. 2 presents SMGs, HIGs, and problem formulation. In Sec. 3, we introduce DAGs and show that they can simulate HIGs. Sec. 4 proposes a DAG-based synthesis framework, which we use for security-aware planning for UAVs in Sec. 5, before concluding the paper in Sec. 6.

2 Stochastic Games

In this section, we present turn-based stochastic games, which assume that all players have full information about the game state. We then introduce hidden-information games and their private-variable semantics.

Notation. We use $\mathbb{N}_0$ to denote the set of non-negative integers. $\mathcal{P}(A)$ denotes the powerset of A (i.e., 2^A). A variable v has a set of valuations $Ev(v)$, where $\eta(v) \in Ev(v)$ denotes one. We use Σ^* to denote the set of all finite words over alphabet Σ, including the empty word ϵ. The mapping $Eff : \Sigma^* \times Ev(v) \rightarrow Ev(v)$ indicates the effect of a finite word on $\eta(v)$. Finally, for general indexing, we use s_i or $s^{(i)}$, for $i \in \mathbb{N}_0$, while PL_γ denotes *Player* γ.

Turn-Based Stochastic Games (SMGs). SMGs can be used to model reactive systems that undergo both stochastic and nondeterministic transitions from one state to another. In a *turn-based* game,[1] actions can be taken at any state by at most one player. Formally, an SMG can be defined as follows [1,28,29].

Definition 1 (Turn-Based Stochastic Game). *A* turn-based game (SMG) *with players* $\Gamma = \{\mathrm{I}, \mathrm{II}, \bigcirc\}$ *is a tuple* $\mathcal{G} = \langle S, (S_\mathrm{I}, S_\mathrm{II}, S_\bigcirc), A, s_0, \delta\rangle$, *where*

- S *is a finite set of states, partitioned into* S_I, S_II *and* $S_\bigcirc$*;*
- $A = A_\mathrm{I} \cup A_\mathrm{II} \cup \{\tau\}$ *is a finite set of actions where* τ *is an empty action;*
- $s_0 \in S_\mathrm{II}$ *is the initial state; and*
- $\delta : S \times A \times S \rightarrow [0,1]$ *is a transition function, such that* $\delta(s, a, s') \in \{1, 0\}$, $\forall s \in S_\mathrm{I} \cup S_\mathrm{II}, a \in A$ *and* $s' \in S$, *and* $\delta(s, \tau, s') \in [0,1]$, $\forall s \in S_\bigcirc$ *and* $s' \in S_\mathrm{I} \cup S_\mathrm{II}$, *where* $\sum_{s' \in S_\mathrm{I} \cup S_\mathrm{II}} \delta(s, \tau, s') = 1$ *holds.*

For all $s \in S_\mathrm{I} \cup S_\mathrm{II}$ and $a \in A_\mathrm{I} \cup A_\mathrm{II}$, we write $s \xrightarrow{a} s'$ if $\delta(s, a, s') = 1$. Similarly, for all $s \in S_\bigcirc$ we write $s \xrightarrow{p} s'$ if s' is randomly sampled with probability $p = \delta(s, \tau, s')$.

Hidden-Information Games. SMGs assume that all players have full knowledge of the current state, and hence provide *perfect-information* models [5]. In many applications, however, this assumption may not hold. A great example are security-aware models where stealthy adversarial actions can be hidden from the system; e.g., the system may not even be aware that it is under attack. On the other hand, *hidden-information* games (HIGs) refer to games where one player does not have complete access to (or knowledge of) the current state. The notion of hidden information can be formalized with the use of *private variables* (PVs) [5]. Specifically, a game state can be encoded using variables $v_\mathcal{T}$ and $v_\mathcal{B}$, representing the true information, which is only known to PL_I, and PL_II belief, respectively.

[1] The term *turn-based* indicates that at any state only one player can play an action. It does not necessarily imply that players take fair turns.

Definition 2 (Hidden-Information Game). *A* hidden-information stochastic game (HIG) *with players* $\Gamma = \{\mathrm{I}, \mathrm{II}, \bigcirc\}$ *over a set of variables* $V = \{v_{\mathcal{T}}, v_{\mathcal{B}}\}$ *is a tuple* $\mathcal{G}_{\mathsf{H}} = \langle S, (S_{\mathrm{I}}, S_{\mathrm{II}}, S_{\bigcirc}), A, s_0, \beta, \delta\rangle$*, where*

- *set of states* $S \subseteq Ev(v_{\mathcal{T}}) \times Ev(v_{\mathcal{B}}) \times \mathcal{P}\left(Ev(v_{\mathcal{T}})\right) \times \Gamma$*, partitioned in* $S_{\mathrm{I}}, S_{\mathrm{II}}, S_{\bigcirc}$*;*
- $A = A_{\mathrm{I}} \cup A_{\mathrm{II}} \cup \{\tau, \theta\}$ *is a finite set of actions, where* τ *denotes an empty action, and* θ *is the action capturing PL*$_{\mathrm{II}}$ *attempt to reveal the true value* $v_{\mathcal{T}}$*;*
- $s_0 \in S_{\mathrm{II}}$ *is the initial state;*
- $\beta\colon A_{\mathrm{II}} \to \mathcal{P}(A_{\mathrm{I}})$ *is a function that defines the set of available PL*$_{\mathrm{I}}$ *actions, based on PL*$_{\mathrm{II}}$ *action; and*
- $\delta\colon S \times A \times S \to [0,1]$ *is a transition function such that* $\delta(s_{\mathrm{I}}, a, s_{\bigcirc}) = \delta(s_{\bigcirc}, a, s_{\mathrm{I}}) = 0$*, and* $\delta(s_{\mathrm{II}}, \theta, s_{\bigcirc})$*,* $\delta(s_{\mathrm{II}}, a, s_{\mathrm{I}})$*,* $\delta(s_{\mathrm{I}}, a, s_{\mathrm{II}}) \in \{0,1\}$ *for all* $s_{\mathrm{I}} \in S_{\mathrm{I}}$*,* $s_{\mathrm{II}} \in S_{\mathrm{II}}$*,* $s_{\bigcirc} \in S_{\bigcirc}$ *and* $a \in A$*, where* $\sum_{s' \in S_{\mathrm{II}}} \delta(s_{\bigcirc}, \tau, s') = 1$*.*

In the above definition, δ only allows transitions s_{I} to s_{II}, s_{II} to s_{I} or $s_{\bigcirc}$, with s_{II} to $s_{\bigcirc}$ conditioned by action θ, and probabilistic transitions $s_{\bigcirc}$ to s_{II}. A game state can be written as $s = (t, u, \Omega, \gamma)$, but to simplify notation we use $s_{\gamma}\,(t, u, \Omega)$ instead, where $t \in Ev(v_{\mathcal{T}})$ is the *true* value of the game, $u \in Ev(v_{\mathcal{B}})$ is PL$_{\mathrm{II}}$ current *belief*, $\Omega \in \mathcal{P}(Ev(v_{\mathcal{T}})) \setminus \{\emptyset\}$ is PL$_{\mathrm{II}}$ *belief space*, and $\gamma \in \Gamma$ is the current player's index. When the truth is hidden from PL$_{\mathrm{II}}$, the belief space Ω is the *information set* [27], capturing PL$_{\mathrm{II}}$ knowledge about the possible true values.

Example 1 (Belief vs. True Value). Our motivating example is a system that consists of a UAV and a human operator. For localization, the UAV mainly relies on a GPS sensor that can be compromised to effectively steer the UAV away from its original path. While aggressive attacks can be detected, some may remain stealthy by introducing only bounded errors at each step [20,26,22,16]. For example, Fig. 1 shows a UAV (PL$_{\mathrm{II}}$) occupying zone A and flying north (N). An adversary (PL$_{\mathrm{I}}$) can launch a stealthy attack targeting its GPS, introducing a bounded error (NE, NW) to remain stealthy. The set of stealthy actions available to the attacker depends on the preceding UAV action, which is captured by the function β, where $\beta(\mathsf{N}) = \{\mathsf{NE}, \mathsf{N}, \mathsf{NW}\}$. Being unaware of the attack, the UAV believes that it is entering zone C, while the true new location is D due to the attack (NE). Initially, $\eta(v_{\mathcal{T}}) = \eta(v_{\mathcal{B}}) = z_A$, and $\Omega = \{z_A\}$ as the UAV is certain it is in zone z_A. In s_2, $\eta(v_{\mathcal{B}}) = z_C$, yet $\eta(v_{\mathcal{T}}) = z_D$. Although $v_{\mathcal{T}}$ is hidden, PL$_{\mathrm{II}}$ is aware that $\eta(v_{\mathcal{T}})$ is in $\Omega = \{z_B, z_C, z_D\}$.

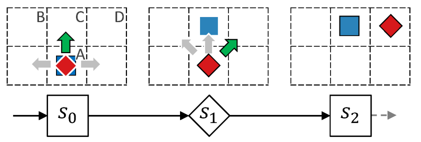


Fig. 1. The UAV belief (solid square) vs. the true value (solid diamond) of its location.

HIG Semantics. $\mathcal{G}_{\mathsf{H}}$ semantics is described using the rules shown in Fig. 2, where H2 and H3 capture PL$_{\mathrm{II}}$ and PL$_{\mathrm{I}}$ moves, respectively. The rule H4 specifies that a PL$_{\mathrm{II}}$ attempt θ to reveal the true value can succeed with probability p_i where PL$_{\mathrm{II}}$ belief is updated (i.e., $u'{=}t$), and remains unchanged otherwise.

Example 2 (HIG Semantics). Continuing Example 1, let us assume that the set of actions $A_{\mathrm{I}} = A_{\mathrm{II}} = \{\mathsf{N}, \mathsf{S}, \mathsf{E}, \mathsf{W}, \mathsf{NE}, \mathsf{NW}, \mathsf{SE}, \mathsf{SW}\}$, and that $\theta = \mathsf{GT}$ is a geolo-

$$
\begin{array}{lll}
\mathsf{H1}: & s_0 = s_{\mathrm{II}}(t_0, u_0, \Omega_0) & \text{if } t_0 = u_0,\ \Omega_0 = \{t_0\} \\
\mathsf{H2}: & s_{\mathrm{II}}\,(t,u,\Omega) \xrightarrow{a_i} s_{\mathrm{I}}\,(t',u',\Omega') & \text{if } a_i \in A_{\mathrm{II}},\ t'=t,\ u'=\mathit{Eff}(a_i,u)\,, \\
 & & \quad \Omega' = \{t' \mid t' = \mathit{Eff}(b_i,t)\ \forall b_i \in \beta(a_i), t \in \Omega\} \\
 & \qquad\qquad\quad \xrightarrow{\theta} s_{\bigcirc}\,(t',u',\Omega') & \text{if } t'=t,\ u'=u,\ \Omega'=\Omega \\
\mathsf{H3}: & s_{\mathrm{I}}\,(t,u,\Omega) \xrightarrow{b_i} s_{\mathrm{II}}\,(t',u',\Omega') & \text{if } b_i \in \beta\,(a_i)\,,\ t'=\mathit{Eff}(b_i,t)\,,\ u'=u,\ \Omega'=\Omega \\
\mathsf{H4}: & s_{\bigcirc}\,(t,u,\Omega) \xrightarrow{p_i} s_{\mathrm{II}}\,(t',u',\Omega') & \text{if } t'=t,\ u'=t,\ \Omega'=\{t\},\ p_i=\delta(s_{\bigcirc},\tau,s_{\mathrm{II}}) \\
 & \qquad\qquad\quad \xrightarrow{1-p_i} s_{\mathrm{II}}\,(t',u',\Omega') & \text{if } t'=t,\ u'=u,\ \Omega'=\Omega,\ 1-p_i=\delta(s_{\bigcirc},\tau,s_{\mathrm{II}})
\end{array}
$$

Fig. 2. Semantic rules for an HIG.

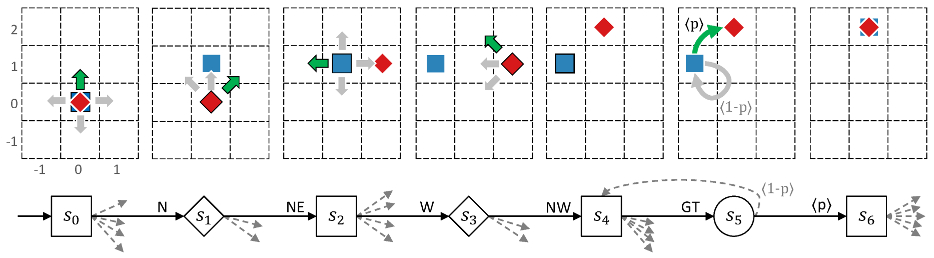


Fig. 3. An example of the UAV motion in a 2D-grid map, modeled as an HIG. Solid squares represent the UAV belief, while solid diamonds represent the ground truth. The UAV action GT denotes performing a geolocation task.

cation task that attempts to reveal the true value of the game.[2] Now, consider the scenario illustrated in Fig. 3. At the initial state s_0, the UAV attempts to move north (N), progressing the game to the state s_1, where the adversary takes her turn by selecting an action from the set $\beta(\mathsf{N}) = \{\mathsf{NE}, \mathsf{N}, \mathsf{NW}\}$. The players take turns until the UAV performs a geolocation task GT, moving from the state s_4 to s_5. With probability $p = \delta(s_5, \tau, s_6)$, the UAV detects its true location and updates its belief accordingly (i.e., to s_6). Otherwise, the belief remains the same (i.e., equal to s_4).

Problem Formulation. Following the system described in Example 2, we now consider the composed HIG $\mathcal{G}_{\mathsf{H}} = \mathcal{M}_{\mathrm{adv}} \| \mathcal{M}_{\mathrm{uav}} \| \mathcal{M}_{\mathrm{as}}$ shown in Fig. 4; the HIG-based model incorporates standard models of a UAV ($\mathcal{M}_{\mathrm{uav}}$), an adversary ($\mathcal{M}_{\mathrm{adv}}$), and a geolocation-task advisory system ($\mathcal{M}_{\mathrm{as}}$) (e.g., as introduced in [11,12]). Here, the probability of a successful detection $p(v_{\mathcal{T}}, v_{\mathcal{B}})$ is a function of both the location the UAV believes to be its current location ($v_{\mathcal{B}}$) as well as the ground truth location that the UAV actually occupies ($v_{\mathcal{T}}$). Reasoning about the flight plan using such model becomes problematic since the ground truth $v_{\mathcal{T}}$ is inherently unknown to the UAV (i.e., $\mathrm{PL}_{\mathrm{II}}$), and thus so is $p(v_{\mathcal{T}}, v_{\mathcal{B}})$. Furthermore, such representation, where some information is hidden, is not supported by off-the-shelf SMG model checkers. Consequently, for such HIGs, *our goal is to find an alternative representation that is suitable for strategy synthesis using off-the-shelf SMG model-checkers.*

[2] A geolocation task is an attempt to localize the UAV by examining its camera feed.

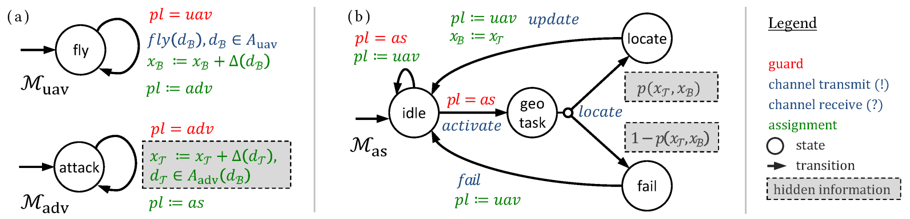

(a)
fly
$\mathcal{M}_{\text{uav}}$
$pl = uav$
$fly(d_{\mathcal{B}}), d_{\mathcal{B}} \in A_{\text{uav}}$
$x_{\mathcal{B}} := x_{\mathcal{B}} + \Delta(d_{\mathcal{B}})$
$pl := adv$
attack
$\mathcal{M}_{\text{adv}}$
$pl = adv$
$x_{\mathcal{T}} := x_{\mathcal{T}} + \Delta(d_{\mathcal{T}})$, $d_{\mathcal{T}} \in A_{\text{adv}}(d_{\mathcal{B}})$
$pl := as$
(b)
$pl = as$
$pl := uav$
$pl := uav$ update
$x_{\mathcal{B}} := x_{\mathcal{T}}$
locate
idle
$pl = as$
activate
geo task
locate
$p(x_{\mathcal{T}}, x_{\mathcal{B}})$
$1 - p(x_{\mathcal{T}}, x_{\mathcal{B}})$
fail
$pl := uav$
fail
$\mathcal{M}_{\text{as}}$
Legend
guard
channel transmit (!)
channel receive (?)
assignment
state
transition
hidden information

Fig. 4. An example of an HIG-based system model comprised of the UAV ($\mathcal{M}_{\text{uav}}$), the adversary ($\mathcal{M}_{\text{adv}}$), and the AS ($\mathcal{M}_{\text{as}}$). Framed information is hidden from the UAV-AS.

3 Delayed-Action Games

In this section, we provide an alternative representation of HIGs that eliminates the use of private variables — we introduce Delayed-Action Games (DAGs) that exploit the notion of *delayed actions*. Furthermore, we show that for any HIG, a DAG that simulates the former can be constructed.

Delayed Actions. Informally, a DAG reconstructs an HIG such that actions of PL_{I} (the player with access to perfect information) follow the actions of PL_{II}, i.e., PL_{I} actions are *delayed*. This rearrangement of the players' actions provides a means to hide information from PL_{II} without the use of private variables, since in this case, at PL_{II} states, PL_{I} actions have not occurred yet. In this way, PL_{II} can act as though she has complete information at the moment she makes her decision, as the future state has not yet happened and so cannot be known. In essence, the formalism can be seen as a partial ordering of the players' actions, exploiting the (partial) superposition property that a wide class of physical systems exhibit. To demonstrate this notion, let us consider DAG modeling on our running example.

Example 3 (Delaying Actions). Fig. 5 depicts the (HIG-based) scenario from Fig. 3, but in the corresponding DAG, where the UAV actions are performed first (in $\hat{s}_0, \hat{s}_1, \hat{s}_2$), followed by the adversary delayed actions (in $\hat{s}_3, \hat{s}_4$). Note that, in the DAG model, at the time the UAV executed its actions ($\hat{s}_0, \hat{s}_1, \hat{s}_2$) the adversary actions had not occurred (yet). Moreover, $\hat{s}_0$ and $\hat{s}_6$ (Fig. 5) share the same belief and true values as s_0 and s_6 (Fig. 3), respectively, though the transient states do not exactly match. This will be used to show the relationship between the games.

The advantage of this approach is twofold. First, the elimination of private variables enables simulation of an HIG using a full-information game. Thus, the formulation of the strategy synthesis problem using off-the-shelf SMG-based tools becomes feasible. In particular, a PL_{II} synthesized strategy becomes dependent on the knowledge of PL_{I} behavior (possible actions), rather than the specific (hidden) actions. We formalize a DAG as follows.

Definition 3 (Delayed-Action Game). *A* DAG *of an HIG* $\mathcal{G}_{\mathsf{H}} = \langle S, (S_{\text{I}}, S_{\text{II}}, S_{\bigcirc}), A, s_0, \beta, \delta \rangle$, *with players* $\Gamma = \{\text{I}, \text{II}, \bigcirc\}$ *over a set of variables* $V = \{v_{\mathcal{T}}, v_{\mathcal{B}}\}$ *is a tuple* $\mathcal{G}_{\mathsf{D}} = \langle \hat{S}, (\hat{S}_{\text{I}}, \hat{S}_{\text{II}}, \hat{S}_{\bigcirc}), A, \hat{s}_0, \beta, \hat{\delta} \rangle$ *where*

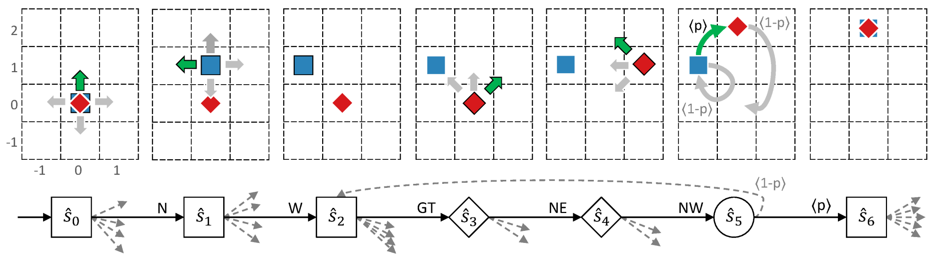


Fig. 5. The same scenario as in Fig. 3, modeled as a DAG. Solid squares represent UAV belief, while solid diamonds represent the ground truth. The UAV action GT denotes performing a geolocation task.

$$
\begin{array}{lll}
\mathsf{D1}: & \hat{s}_0 = \hat{s}_{\mathrm{II}}(\hat{t}_0, \hat{u}_0, w_0, 0) & \text{if } \hat{t}_0 = \hat{u}_0,\ w_0 = \epsilon \\
\mathsf{D2}: & \hat{s}_{\mathrm{II}}\left(\hat{t}, \hat{u}, w, 0\right) \xrightarrow{a_i} \hat{s}_{\mathrm{II}}\left(\hat{t}', \hat{u}', w', 0\right) & \text{if } a_i \in A_{\mathrm{II}},\ \hat{t}' = \hat{t},\ \hat{u}' = \mathit{Eff}(a_i, \hat{u}),\ w' = w a_i \\
 & \qquad\qquad\quad \xrightarrow{\theta} \hat{s}_{\mathrm{I}}\left(\hat{t}', \hat{u}', w', 0\right) & \text{if } \hat{t}' = \hat{t},\ \hat{u}' = \hat{u},\ w' = w \\
\mathsf{D3}: & \hat{s}_{\mathrm{I}}\left(\hat{t}, \hat{u}, w, j\right) \xrightarrow{b_i} \hat{s}_{\mathrm{I}}\left(\hat{t}', \hat{u}', w', j+1\right) & \text{if } b_i \in \beta(w_j),\ \hat{t}' = \mathit{Eff}\left(b_i, \hat{t}\right),\ \hat{u}' = \hat{u},\ w' = w,\ j < |w| - 1 \\
 & \qquad\qquad\quad \xrightarrow{b_i} \hat{s}_{\bigcirc}\left(\hat{t}', \hat{u}', w', j\right) & \text{if } b_i \in \beta(w_j),\ \hat{t}' = \mathit{Eff}\left(b_i, \hat{t}\right),\ \hat{u}' = \hat{u},\ w' = w,\ j = |w| - 1 \\
\mathsf{D4}: & \hat{s}_{\bigcirc}\left(\hat{t}, \hat{u}, w, j\right) \xrightarrow{p_i} \hat{s}_{\mathrm{II}}\left(\hat{t}', \hat{u}', w', 0\right) & \text{if } \hat{t}' = \hat{t},\ \hat{u}' = \hat{t},\ w' = \epsilon,\ p_i = \hat{\delta}(\hat{s}_{\bigcirc}, \hat{s}_{\mathrm{II}}) \\
 & \qquad\qquad\quad \xrightarrow{1-p_i} \hat{s}_{\mathrm{II}}\left(\hat{t}', \hat{u}', w', 0\right) & \text{if } \hat{t}' = \hat{t}_0,\ \hat{u}' = \hat{u},\ w' = w,\ q_i = \hat{\delta}(\hat{s}_{\bigcirc}, \hat{s}_{\mathrm{II}})
\end{array}
$$

Fig. 6. Semantic rules for DAGs.

- $\hat{S} \subseteq Ev(v_{\mathcal{T}}) \times Ev(v_{\mathcal{B}}) \times A_{\mathrm{II}}^{*} \times \mathbb{N}_0 \times \Gamma$ *is the set of states, partitioned into* $\hat{S}_{\mathrm{I}}, \hat{S}_{\mathrm{II}}$ *and* $\hat{S}_{\bigcirc}$*;*
- $\hat{s}_0 \in \hat{S}_{\mathrm{II}}$ *is the initial state; and*
- $\hat{\delta}\colon \hat{S} \times A \times \hat{S} \to [0,1]$ *is a transition function such that* $\hat{\delta}(\hat{s}_{\mathrm{II}}, a, \hat{s}_{\bigcirc}) = \hat{\delta}(\hat{s}_{\mathrm{I}}, a, \hat{s}_{\mathrm{II}}) = \hat{\delta}(\hat{s}_{\bigcirc}, a, \hat{s}_{\mathrm{I}}) = 0$*, and* $\hat{\delta}(\hat{s}_{\mathrm{II}}, a, \hat{s}_{\mathrm{II}}) \in \{0,1\}$*,* $\hat{\delta}(\hat{s}_{\mathrm{II}}, \theta, \hat{s}_{\mathrm{I}}) \in \{0,1\}$*,* $\hat{\delta}(\hat{s}_{\mathrm{I}}, a, \hat{s}_{\mathrm{I}}) \in \{0,1\}$*,* $\hat{\delta}(\hat{s}_{\mathrm{I}}, a, \hat{s}_{\bigcirc}) \in \{0,1\}$*, for all* $\hat{s}_{\mathrm{I}} \in \hat{S}_{\mathrm{I}}$*,* $\hat{s}_{\mathrm{II}} \in \hat{S}_{\mathrm{II}}$*,* $\hat{s}_{\bigcirc} \in \hat{S}_{\bigcirc}$ *and* $a \in A$*, where* $\sum_{\hat{s}' \in \hat{S}_{\mathrm{II}}} \delta(\hat{s}_{\bigcirc}, a, \hat{s}') = 1$.

Note that, in contrast to transition function δ in HIG $\mathcal{G}_{\mathsf{H}}$, $\hat{\delta}$ in DAG $\mathcal{G}_{\mathsf{D}}$ only allows transitions $\hat{s}_{\mathrm{II}}$ to $\hat{s}_{\mathrm{II}}$ or $\hat{s}_{\mathrm{I}}$, as well as $\hat{s}_{\mathrm{I}}$ to $\hat{s}_{\mathrm{I}}$ or $\hat{s}_{\bigcirc}$, and probabilistic transitions $\hat{s}_{\bigcirc}$ to $\hat{s}_{\mathrm{II}}$; also note that $\hat{s}_{\mathrm{II}}$ to $\hat{s}_{\mathrm{I}}$ is conditioned by the action θ.

DAG Semantics. A DAG state is a tuple $\hat{s} = \left(\hat{t}, \hat{u}, w, j, \gamma\right)$, which for simplicity we shorthand as $\hat{s}_{\gamma}\left(\hat{t}, \hat{u}, w, j\right)$, where $\hat{t} \in Ev(v_{\mathcal{T}})$ is the last known true value, $\hat{u} \in Ev(v_{\mathcal{B}})$ is $\mathrm{PL}_{\mathrm{II}}$ belief, $w \in A_{\mathrm{II}}^{*}$ captures $\mathrm{PL}_{\mathrm{II}}$ actions taken since the last known true value, $j \in \mathbb{N}_0$ is an index on w, and $\gamma \in \Gamma$ is the current player index. The game transitions are defined using the semantic rules from Fig. 6. Note that $\mathrm{PL}_{\mathrm{II}}$ can execute multiple moves (i.e., actions) before executing θ to attempt to reveal the true value (D2), moving to a PL_{I} state where PL_{I} executes all her delayed actions before reaching a 'revealing' state $\hat{s}_{\bigcirc}$ (D3). Finally, the revealing attempt can succeed with probability p_i where $\mathrm{PL}_{\mathrm{II}}$ belief is updated (i.e., $\hat{u}' = \hat{t}$), or otherwise remains unchanged (D4).

In both $\mathcal{G}_\mathsf{H}$ and $\mathcal{G}_\mathsf{D}$, we label states where all players have full knowledge of the current state as *proper*. We also say that two states are similar if they agree on the belief, and equivalent if they agree on both the belief and ground truth.

Definition 4 (States). *Let $s_\gamma(t, u, \Omega) \in S$ and $\hat{s}_{\hat{\gamma}}(\hat{t}, \hat{u}, w, j) \in \hat{S}$. We say:*

- *s_γ is* proper *iff $\Omega = \{t\}$, denoted by $s_\gamma \in \mathrm{Prop}(\mathcal{G}_\mathsf{H})$.*
- *$\hat{s}_{\hat{\gamma}}$ is* proper *iff $w = \epsilon$, denoted by $\hat{s}_{\hat{\gamma}} \in \mathrm{Prop}(\mathcal{G}_\mathsf{D})$.*
- *s_γ and $\hat{s}_{\hat{\gamma}}$ are* similar *iff $\hat{u} = u$, $\hat{t} \in \Omega$, and $\gamma = \hat{\gamma}$, denoted by $s_\gamma \sim \hat{s}_{\hat{\gamma}}$.*
- *s_γ, $\hat{s}_{\hat{\gamma}}$ are* equivalent *iff $t = \hat{t}$, $u = \hat{u}$, $w = \epsilon$, and $\gamma = \hat{\gamma}$, denoted by $s_\gamma \simeq \hat{s}_{\hat{\gamma}}$.*

From the above definition, we have that $s \simeq \hat{s} \implies s \in \mathrm{Prop}(\mathcal{G}_\mathsf{H}), \hat{s} \in \mathrm{Prop}(\mathcal{G}_\mathsf{D})$. We now define *execution fragments*, possible progressions from a state to another.

Definition 5 (Execution Fragment). *An* execution fragment *(of either an SMG, DAG or HIG) is a finite sequence of states, actions and probabilities $\varrho = s_0 a_1 p_1 s_1 a_2 p_2 s_2 \ldots a_n p_n s_n$ such that $(s_i \xrightarrow{a_{i+1}} s_{i+1}) \vee (s_i \xrightarrow{\langle p_{i+1} \rangle} s_{i+1}), \forall i \geq 0$.*[3]

We use *first*(ϱ) and *last*(ϱ) to refer to the first and last states of ϱ, respectively. If both states are proper, we say that ϱ is *proper* as well, denoted by $\varrho \in \mathrm{Prop}(\mathcal{G}_\mathsf{H})$.[4] Moreover, ϱ is *deterministic* if no probabilities appear in the sequence.

Definition 6 (Move). *A* move *m_γ of an execution ϱ from state $s \in \varrho$, denoted by $move_\gamma(s, \varrho)$, is a sequence of actions $a_1 a_2 \ldots a_i \in A^*_\gamma$ that player γ performs in ϱ starting from s.*

By omitting the player index we refer to the moves of all players. To simplify notation, we use *move*(ϱ) as a short notation for *move*(*first*$(\varrho), \varrho)$. We write $(m)(\mathit{first}(\varrho)) = \mathit{last}(\varrho)$ to denote that the execution of move m from the *first*(ϱ) leads to the *last*(ϱ). This allows us to now define the *delay operator* as follows.

Definition 7 (Delay Operator). *For an $\mathcal{G}_\mathsf{H}$, let $m = move(\varrho) = a_1 b_1 \ldots a_n b_n \theta$ be a move for some deterministic $\varrho \in \mathrm{TS}(\mathcal{G}_\mathsf{H})$, where $a_1 ... a_n \in A^*_{\mathrm{II}}, b_1 ... b_n \in A^*_{\mathrm{I}}$. The delay operator, denoted by $\overline{m}$, is defined by the rule $\overline{m} = a_1 \ldots a_n \theta b_1 \ldots b_n$.*

Intuitively, the delay operator shifts PL_I actions to the right of PL_II actions up until the next probabilistic state. For example,

$$\begin{tikzcd}
\text{if} & \rho = & s^{(0)}_{\mathrm{II}} \arrow[rr, "a_1"] & & s^{(1)}_{\mathrm{I}} \arrow[rr, "b_2"] & & s^{(2)}_{\mathrm{II}} \arrow[rr, "\theta"] & & s^{(3)}_{\bigcirc} \arrow[rr, "p_3"] & & s^{(4)}_{\mathrm{II}} \arrow[rr, "a_4"] & & s^{(5)}_{\mathrm{I}} \arrow[rr, "b_5"] & & s^{(6)}_{\mathrm{II}} \arrow[rr, "a_6"] & & s^{(7)}_{\mathrm{I}} \arrow[rr, "b_7"] & & s^{(8)}_{\mathrm{II}} \\
\text{then} & m = & & a_1 & & b_2 \arrow[drr] & & \theta \arrow[dll] & & \tau & & a_4 & & b_5 \arrow[drr] & & a_6 \arrow[dll] & & b_7, & \\
\text{and} & \overline{m} = & & a_1 & & \theta & & b_2 & & \tau & & a_4 & & a_6 & & b_5 & & b_7. &
\end{tikzcd}$$

Simulation Relation. Given an HIG $\mathcal{G}_\mathsf{H}$, we first define the corresponding DAG $\mathcal{G}_\mathsf{D}$.

Definition 8 (Correspondence). *Given an HIG $\mathcal{G}_\mathsf{H}$, a corresponding DAG $\mathcal{G}_\mathsf{D} = \mathfrak{D}[\mathcal{G}_\mathsf{H}]$ is a DAG that follows the semantic rules displayed in Fig. 7.*

$$\begin{array}{lll}
s_0 = s_{\text{II}}(t_0, u_0, \Omega_0) & \Longrightarrow & \hat{s}_0 = \hat{s}_{\text{II}}(\hat{t}_0, \hat{u}_0, w_0, 0) \text{ s.t. } \hat{t}_0 = t_0,\ \hat{u}_0 = u_0 \\
s_{\text{II}}(t, u, \Omega) \xrightarrow{a_i} s_{\text{I}}(t', u', \Omega') & \Longrightarrow & \hat{s}_{\text{II}}\left(\hat{t}, \hat{u}, w, 0\right) \xrightarrow{a_i} \hat{s}_{\text{II}}\left(\hat{t}', \hat{u}', w', 0\right) \text{ s.t. } \hat{u} = u \\
s_{\text{II}}(t, u, \Omega) \xrightarrow{\theta_i} s_{\bigcirc}(t', u', \Omega') & \Longrightarrow & \hat{s}_{\text{II}}\left(\hat{t}, \hat{u}, w, 0\right) \xrightarrow{\theta_i} s_{\text{I}}\left(\hat{t}', \hat{u}', w', 0\right) \text{ s.t. } \hat{u} = u \\
s_{\text{I}}(t, u, \Omega) \xrightarrow{b_i} s_{\text{II}}(t', u', \Omega') & \Longrightarrow & \hat{s}_{\text{I}}\left(\hat{t}, \hat{u}, w, j\right) \xrightarrow{b_i} s_{\text{I}}\left(\hat{t}', \hat{u}', w', j{+}1\right) \text{ s.t. } \hat{t} = t, j < |w| \\
s_{\text{I}}(t, u, \Omega) \xrightarrow{b_i} s_{\text{II}}(t', u', \Omega') & \Longrightarrow & \hat{s}_{\text{I}}\left(\hat{t}, \hat{u}, w, j\right) \xrightarrow{b_i} s_{\bigcirc}\left(\hat{t}', \hat{u}', w', j\right) \text{ s.t. } \hat{t} = t, j = |w| \\
s_{\bigcirc}(t, u, \Omega) \xrightarrow{p_i} s_{\text{II}}(t', u', \Omega') & \Longrightarrow & \hat{s}_{\bigcirc}\left(\hat{t}, \hat{u}, w, j\right) \xrightarrow{p_i} \hat{s}_{\text{II}}\left(\hat{t}', \hat{u}', w', 0\right) \text{ s.t. } \hat{t} = t,\ \hat{u} = u \\
s_{\bigcirc}(t, u, \Omega) \xrightarrow{1-p_i} s_{\text{II}}(t', u', \Omega') & \Longrightarrow & \hat{s}_{\bigcirc}\left(\hat{t}, \hat{u}, w, j\right) \xrightarrow{1-p_i} \hat{s}_{\text{II}}\left(\hat{t}', \hat{u}', w', 0\right) \text{ s.t. } \hat{t} = t,\ \hat{u} = u
\end{array}$$

Fig. 7. Semantic rules for HIG-to-DAG transformation.

For the rest of this section, we consider $\mathcal{G}_{\mathsf{D}} = \mathfrak{D}[\mathcal{G}_{\mathsf{H}}]$, and use $\varrho \in \mathrm{TS}(\mathcal{G}_{\mathsf{H}})$ and $\hat{\varrho} \in \mathrm{TS}(\mathcal{G}_{\mathsf{D}})$ to denote two execution fragments of the HIG and DAG, respectively. We say that ϱ and $\hat{\varrho}$ are *similar*, denoted by $\varrho \sim \hat{\varrho}$, iff $\mathit{first}(\varrho) \simeq \mathit{first}(\hat{\varrho})$, $\mathit{last}(\varrho) \sim \mathit{last}(\hat{\varrho})$, and $\overline{\mathit{move}(\varrho)} = \mathit{move}(\hat{\varrho})$.

Definition 9 (Game Proper Simulation). *A game $\mathcal{G}_{\mathsf{D}}$ properly simulates $\mathcal{G}_{\mathsf{H}}$, denoted by $\mathcal{G}_{\mathsf{D}} \rightsquigarrow \mathcal{G}_{\mathsf{H}}$, iff $\forall \varrho \in \mathrm{Prop}(\mathcal{G}_{\mathsf{H}})$, $\exists \hat{\varrho} \in \mathrm{Prop}(\mathcal{G}_{\mathsf{D}})$ such that $\varrho \sim \hat{\varrho}$.*

Before proving the existence of the simulation relation, we first show that if a move is executed on two equivalent states, then the terminal states are similar.

Lemma 1 (Terminal States Similarity). *For any $s_0 \simeq \hat{s}_0$ and a deterministic $\varrho \in \mathrm{TS}(\mathcal{G}_{\mathsf{H}})$ where $\mathit{first}(\varrho) = s_0$, $\mathit{last}(\varrho) \in S_{\text{II}}$, then $\mathit{last}(\varrho) \sim \left(\overline{\mathit{move}(\varrho)}\right)(\hat{s}_0)$ holds.*

Proof. Let $\mathit{last}(\varrho_i) = s_{\gamma_i}^{(i)}(t_i, u_i, \Omega_i)$ and $\left(\overline{\mathit{move}(\varrho_i)}\right)(\hat{s}_0) = \hat{s}_{\hat{\gamma}_i}^{(i)}(\hat{t}_i, \hat{u}_i, w_i, j_i)$, where $\mathit{move}(\varrho_i) = a_1 b_1 ... a_i b_i \theta$. We then write $\overline{\mathit{move}(\varrho)} = a_1 ... a_i \theta b_1 ... b_i$. We use induction over i as follows:

- Base ($i = 0$): $\varrho_0 = s_0 \implies s^{(0)} \simeq \hat{s}^{(0)}$ where $u_0 = \hat{u}_0$ and $t_0 = \hat{t}_0$.
- Induction ($i > 0$): Assume that the claim holds for $\mathit{move}(\varrho_{i-1}) = a_1 b_1 ... a_{i-1} b_{i-1} \theta$, i.e., $u_{i-1} = \hat{u}_{i-1}$ and $\hat{t}_{i-1} \in \Omega_{i-1}$. For ϱ_i we have that $u_i = \mathit{Eff}(a_i, u_{i-1})$ and $\hat{u}_i = \mathit{Eff}(a_i, \hat{u}_{i-1})$. Also, $t_i = \mathit{Eff}(b_i, t_{i-1}) \in \Omega_i$ and $\hat{t}_i = \mathit{Eff}\big(b_i, \hat{t}_{i-1}\big)$. Hence, $u_i = \hat{u}_i$, $\hat{t}_i \in \Omega_i$ and $\hat{\gamma}_i = \gamma_i = \bigcirc$. Thus, $s^{(i)} \sim \hat{s}^{(i)}$ holds. The same can be shown for $\mathit{move}(\varrho) = a_1 b_1 ... a_i b_i$ where no θ occurs. $\square$

Theorem 1 (Probabilistic Simulation). *For any $s_0 \simeq \hat{s}_0$ and $\varrho \in \mathrm{Prop}(\mathcal{G}_{\mathsf{H}})$ where $\mathit{first}(\varrho) = s_0$, it holds that*

$$\Pr\left[\mathit{last}(\varrho) = s'\right] = \Pr\left[\left(\overline{\mathit{move}(\varrho)}\right)(\hat{s}_0) = \hat{s}'\right] \quad \forall s', \hat{s}' \ \ s.t.\ \ s' \simeq \hat{s}'.$$

Proof. We can rewrite ϱ as $\varrho = \varrho_0 \xrightarrow{p_1} \varrho_1 \cdots \varrho_{n-1} \xrightarrow{p_n} s_{\text{II}}^{(n)}$, where $\varrho_0, \varrho_1, \ldots, \varrho_{n-1}$ are deterministic. Let $\mathit{first}(\varrho_i) = s_{\text{II}}^{(i)}(t_i, u_i, \Omega_i)$, $\mathit{last}(\varrho_i) = s_{\bigcirc}^{(i)}(t_i', u_i', \Omega_i')$, and $\left(\overline{\mathit{move}(\varrho)}\right)(\hat{s}_0) = \hat{s}^{(n)}(\hat{t}_n, \hat{u}_n, w_n, j_n)$. We use induction over n as follows:

[3] For deterministic transitions, $p = 1$, hence omitted from ϱ for readability.

[4] An execution fragment lives in the transition system (TS), i.e., $\varrho \in \mathrm{Prop}(\mathrm{TS}(\mathcal{G}))$. We omit TS for readability.

- Base ($n = 0$): for ϱ to be deterministic and proper, $\varrho = \varrho_0 = s^{(0)}$ holds.
- Case ($n = 1$): $p_1 = p(t'_0, u'_0)$. From Lemma 1, $\hat{u}_1 = u_1$ and $\hat{t}_1 = t_1$. Hence, $\Pr\left[\mathit{last}(\varrho) = s_{\mathrm{II}}^{(1)}\right] = \Pr\left[\left(\overline{\mathit{move}(\varrho)}\right)(\hat{s}_0) = \hat{s}_{\mathrm{II}}^{(1)}\right] = p(t'_0, u'_0)$ and $s_{\mathrm{II}}^{(1)} \simeq \hat{s}_{\mathrm{II}}^{(1)}$.
- Induction ($n > 1$): It is straightforward to infer that $p_n = p\left(t'_{n-1}, u'_{n-1}\right)$, hence $\Pr\left[\mathit{last}(\varrho) = s_{\mathrm{II}}^{(n)}\right] = \Pr\left[\left(\overline{\mathit{move}(\varrho)}\right)\left(\hat{s}^{(0)}\right) = \hat{s}^{(n)}\right] = P$, and $s_{\mathrm{II}}^{(n)} \simeq \hat{s}_{\mathrm{II}}^{(n)}$. □

Note that in case of multiple θ attempts, the above probability P satisfies

$$P = \prod_{i=1}^{n} \sum_{j=1}^{m_i} p_i\left(t'_{i-1}, u'_{i-1}\right)\left(1 - p_{i-1}\left(t'_{i-1}, u'_{i-1}\right)\right)^{(j-1)},$$

where m_i is the number of θ attempts at stage i. Finally, since Theorem 1 imposes no constraints on $\mathit{move}(\varrho)$, a DAG can simulate all proper executions that exist in the corresponding HIG.

Theorem 2 (DAG-HIG Simulation). *For any HIG $\mathcal{G}_{\mathsf{H}}$ there exists a DAG $\mathcal{G}_{\mathsf{D}} = \mathfrak{D}[\mathcal{G}_{\mathsf{H}}]$ such that $\mathcal{G}_{\mathsf{D}} \rightsquigarrow \mathcal{G}_{\mathsf{H}}$ (as defined in Def. 9).*

4 Properties of DAG and DAG-based Synthesis

We here discuss DAG features, including how it can be decomposed into subgames by restricting the simulation to finite executions, and the preservation of safety properties, before proposing a DAG-based synthesis framework.

Transitions. In DAGs, nondeterministic actions of different players underline different semantics. Specifically, PL_{I} nondeterminism captures what is known about the adversarial behavior, rather than exact actions, where PL_{I} actions are constrained by the earlier $\mathrm{PL}_{\mathrm{II}}$ action. Conversely, $\mathrm{PL}_{\mathrm{II}}$ nondeterminism abstracts the player's decisions. This distinction reflects how DAGs can be used for strategy synthesis under hidden information. To illustrate this, suppose that a strategy π_{II} is to be obtained based on a worst-case scenario. In that case, the game is explored for all possible adversarial behaviors. Yet, if a strategy π_{I} is known about PL_{I}, a counter strategy π_{II} can be found by constructing $\mathcal{G}_{\mathsf{D}}^{\pi_{\mathrm{I}}}$.

Probabilistic behaviors in DAGs are captured by $\mathrm{PL}_{\bigcirc}$, which is characterized by the transition function $\hat{\delta}\colon \hat{S}_{\bigcirc} \times \hat{S}_{\mathrm{II}} \to [0,1]$. The specific definition of $\hat{\delta}$ depends on the modeled system. For instance, if the transition function (i.e., the probability) is state-independent, i.e., $\hat{\delta}(\hat{s}_{\bigcirc}, \hat{s}_{\mathrm{II}}) = c, c \in [0,1]$, the obtained model becomes trivial. Yet, with a state-dependent transition function, i.e., $\hat{\delta}(\hat{s}_{\bigcirc}, \hat{s}_{\mathrm{II}}) = p(\hat{t}, \hat{u})$, the probability that $\mathrm{PL}_{\mathrm{II}}$ successfully reveals the true value depends on both the belief and the true value, and the transition function can then be realized since $\hat{s}_{\bigcirc}$ holds both $\hat{t}$ and $\hat{u}$.

Decomposition. Consider an execution $\hat{\varrho}^* = \hat{s}_0 a_1 \hat{s}_1 a_2 \hat{s}_2 \ldots$ that describes a scenario where $\mathrm{PL}_{\mathrm{II}}$ performs infinitely many actions with no attempt to reveal the true value. To simulate $\hat{\varrho}^*$, the word w needs to infinitely grow. Since we are interested in finite executions, we impose *stopping criteria* on the DAG, such that the game is *trapped* whenever $|w| = h_{\max}$ is true, where $h_{\max} \in \mathbb{N}$ is an *upper horizon.* We formalize the stopping criteria as a deterministic finite automaton (DFA) that, when composed with the DAG, traps the game whenever the stopping criteria hold. Note that imposing an upper horizon by itself is not a sufficient criterion for a DAG to be considered a stopping game [8]. Conversely, consider a proper (and hence finite) execution $\hat{\varrho} = \hat{s}_0 a_1 \ldots \hat{s}'$, where $\hat{s}_0, \hat{s}' \in \mathrm{Prop}(\mathcal{G}_{\mathsf{D}})$. From Definition 9, it follows that a DAG initial state is strictly proper, i.e., $\hat{s}_0 \in \mathrm{Prop}(\mathcal{G}_{\mathsf{D}})$. Hence, when $\hat{s}'$ is reached, the game can be seen as if it is *repeated* with a new initial state $\hat{s}'$. Consequently, a DAG game (complemented with stopping criteria) can be decomposed into a (possibly infinite) countable set of *subgames* that have the same structure yet different initial states.

Definition 10 (DAG Subgames). *The* subgames *of a* $\mathcal{G}_{\mathsf{D}}$ *are defined by the set* $\left\{\hat{\mathcal{G}}_i \;\middle|\; \hat{\mathcal{G}}_i = \left\langle \hat{S}^{(i)}, (\hat{S}^{(i)}_{\mathrm{I}}, \hat{S}^{(i)}_{\mathrm{II}}, \hat{S}^{(i)}_{\bigcirc}), A, \hat{s}^{(i)}_0, \hat{\delta}^{(i)} \right\rangle, i \in \mathbb{N}_0 \right\}$, *where* $\hat{S} = \bigcup_i \hat{S}^{(i)}$*;* $\hat{S}_\gamma = \bigcup_i \hat{S}^{(i)}_\gamma \; \forall \gamma \in \Gamma$*; and* $\hat{s}^{(i)}_0 = \hat{s}^{(i)}_{\mathrm{II}}$ *s.t.* $\hat{s}^{(i)}_{\mathrm{II}} \in \mathrm{Prop}(\mathcal{G}^{(i)}_{\mathsf{D}})$, $\hat{s}^{(i)}_{\mathrm{II}} \neq \hat{s}^{(j)}_{\mathrm{II}} \; \forall i, j \in \mathbb{N}_0$.

Intuitively, each subgame either reaches a proper state (representing the initial state of another subgame) or terminates by an upper horizon. This decomposition allows for the independent (and parallel) analysis of individual subgames, drastically reducing both the time required for synthesis and the explored state space, and hence improving scalability. An example of this decompositional approach is provided in Sec. 5.

Preservation of safety properties. In DAGs, the action θ denotes a transition from $\mathrm{PL}_{\mathrm{II}}$ to PL_{I} states and thus the execution of any delayed actions. While this action can simply describe a revealing attempt, it can also serve as a *what-if* analysis of how the true value may evolve at stage i of a subgame. We refer to an execution of the second type as a *hypothetical branch*, where $\mathrm{Hyp}(\hat{\varrho}, h)$ denotes the set of hypothetical branches from $\hat{\varrho}$ at stage $h \in \{1, \ldots, n\}$. Let $L_{\mathrm{safe}}(s)$ be a labeling function denoting if a state is safe. The formula $\Phi_{\mathrm{safe}} := [\mathsf{G}\, \mathit{safe}]$ is satisfied by an execution ϱ in HIG iff all $s(t, u, \Omega) \in \varrho$ are safe.

Now, consider $\hat{\varrho}$ of the DAG, with $\hat{\varrho} \sim \varrho$. We identify the following three cases:
(a) $L_{\mathrm{safe}}(s)$ depends only on the belief u, then $\varrho \models \Phi_{\mathrm{safe}}$ iff all $\hat{s}_{\mathrm{II}} \in \hat{\varrho}$ are safe;
(b) $L_{\mathrm{safe}}(s)$ depends only on the true value t, then $\varrho \models \Phi_{\mathrm{safe}}$ iff all $\hat{s}_{\mathrm{I}} \in \mathrm{Hyp}(\hat{\varrho}, n)$ are safe; and
(c) $L_{\mathrm{safe}}(s)$ depends on both the true value t and belief u, then $\varrho \models \Phi_{\mathrm{safe}}$ iff $\mathit{last}(\hat{\varrho}_h)$ is safe for all $\hat{\varrho}_h \in \mathrm{Hyp}(\hat{\varrho}, h), h \in \{1, ..., n\}$, where n is the number of $\mathrm{PL}_{\mathrm{II}}$ actions.
Taking into account such relations, both safety (e.g., never encounter a hazard) and distance-based requirements (e.g., never exceed a subgame horizon) can be specified when using DAGs for synthesis, to ensure their satisfaction in the original model. This can be generalized to other reward-based synthesis objectives, which will be part of our future efforts that we discuss in Sec. 6.

Synthesis Framework. We here propose a framework for strategy synthesis using DAGs, which is summarized in Fig. 8. We start by formulating the automata $\mathcal{M}_{\mathrm{I}}$, $\mathcal{M}_{\mathrm{II}}$ and $\mathcal{M}_{\bigcirc}$, representing PL_{I}, $\mathrm{PL}_{\mathrm{II}}$ and $\mathrm{PL}_{\bigcirc}$ abstract behaviors,

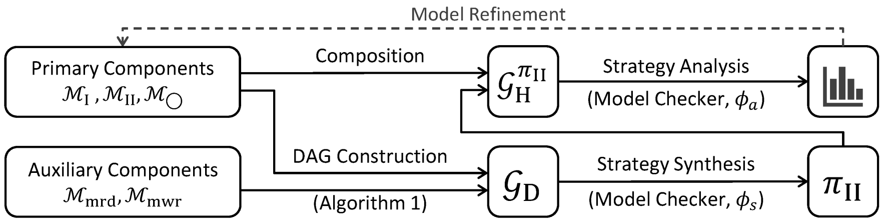

Model Refinement
Primary Components $\mathcal{M}_{\mathrm{I}}, \mathcal{M}_{\mathrm{II}}, \mathcal{M}_{\bigcirc}$
Composition
$\mathcal{G}_{\mathsf{H}}^{\pi_{\mathrm{II}}}$
Strategy Analysis
(Model Checker, ϕ_a)
Auxiliary Components $\mathcal{M}_{\mathrm{mrd}}, \mathcal{M}_{\mathrm{mwr}}$
DAG Construction
(Algorithm 1)
$\mathcal{G}_{\mathsf{D}}$
Strategy Synthesis
(Model Checker, ϕ_s)
π_{II}

Fig. 8. Synthesis and analysis framework based on the use of DAGs.

Algorithm 1: Procedure for DAG construction

```
Input: Components $\mathcal{M}_{\mathrm{I}}, \mathcal{M}_{\mathrm{II}}, \mathcal{M}_{\bigcirc}, \mathcal{M}_{\mathrm{mwr}}, \mathcal{M}_{\mathrm{mrd}}$; initial state $\hat{s}_0$
Result: DAG $\mathcal{G}_{\mathsf{D}}$
1 while $\neg$(end criterion) do
2     while $a \neq \theta$ do                                ▷ PL$_{\mathrm{II}}$ plays until a revealing attempt
3         $\mathcal{M}_{\mathrm{II}}.v_{\mathcal{B}} \leftarrow Eff(a, v_{\mathcal{B}})$, $\mathcal{M}_{\mathrm{mwr}}.write(a, {+}{+}wr)$
4     while $rd \leqslant wr$ do                              ▷ PL$_{\mathrm{I}}$ plays all delayed actions
5         $\mathcal{M}_{\mathrm{mrd}}.read(a, {+}{+}rd)$, $\mathcal{M}_{\mathrm{I}}.v_{\mathcal{T}} \leftarrow Eff(\beta(a), v_{\mathcal{T}})$
6     if draw $x \sim Brn(p(v_{\mathcal{T}}, v_{\mathcal{B}}))$ then       ▷ PL$_{\bigcirc}$ plays successful attempt
7         $\mathcal{M}_{\mathrm{II}}.v_{\mathcal{B}} \leftarrow \mathcal{M}_{\mathrm{I}}.v_{\mathcal{T}}$, $wr \leftarrow 0$, $rd \leftarrow 0$
8     else $rd \leftarrow 0$                                  ▷ Unsuccessful attempt, forget PL$_{\mathrm{I}}$ actions
```

respectively. Next, a FIFO memory stack $(m_i)_{i=1}^n \in A_{\mathrm{II}}^n$ is implemented using two automata $\mathcal{M}_{\mathrm{mrd}}$ and $\mathcal{M}_{\mathrm{mwr}}$ to perform reading and writing operations, respectively.[5] The DAG $\mathcal{G}_{\mathsf{D}}$ is constructed by following Algorithm 1. The game starts with PL$_{\mathrm{II}}$ moves until she executes a revealing attempt θ, allowing PL$_{\mathrm{I}}$ to play her delayed actions. Once an end criterion is met, the game terminates, resembling conditions such as 'running out of fuel' or 'reaching map boundaries'.

Algorithm 2 describes the procedure for strategy synthesis based on the DAG $\mathcal{G}_{\mathsf{D}}$, and an rPATL [6] synthesis query ϕ_{syn} that captures, for example, a safety requirement. Starting with the initial location, the procedure checks whether ϕ_{syn} is satisfied if action θ is performed at stage h, and updates the set of feasible strategies Π_i for subgame $\hat{\mathcal{G}}_i$ until $h_{\max}$ is reached or ϕ_{syn} is not satisfied.[6] Next, the set Π_i is used to update the list of reachable end locations ℓ with new initial locations of reachable subgames that should be explored. Finally, the composition of both $\mathcal{G}_{\mathsf{H}}$ and Π_{II}^* resolves PL$_{\mathrm{II}}$ nondeterminism, where the resulting model $\mathcal{G}_{\mathsf{H}}^{\Pi_{\mathrm{II}}^*}$ is a Markov Decision Process (MDP) of complete information that can be easily used for further analysis.

5 Case Study

In this section, we consider a case study where a human operator supervises a UAV prone to stealthy attacks on its GPS sensor. The UAV mission is to

[5] Specific implementation details are described in Sec. 5.
[6] Failing to find a strategy at stage i implies the same for all horizons of size $j > i$.

Algorithm 2: Procedure for strategy synthesis

```
    Input: Initial location $(x_0, y_0)$, synthesis query $\phi_{\text{syn}}$
    Output: $\text{PL}_{\text{II}}$ strategies $\Pi^*_{\text{II}}$
 1  $\ell \leftarrow [(x_0, y_0)]$ , $i \leftarrow 0$
 2  while $i < |\ell|$ do                                          ▷ Explore all reachable subgames
 3      $\hat{s}_0 \leftarrow (\ell[i], \ell[i], \epsilon, 0, \text{II})$, $h \leftarrow 1, stop \leftarrow \bot$      ▷ Construct initial state
 4      while $h \leqslant h_{\max} \wedge \neg stop$ do                   ▷ Explore subgame till upper horizon
 5          $(\pi_{\text{II}}, \varphi) \leftarrow \mathsf{Synth}\left(\hat{\mathcal{G}}^{\pi_h}_{\hat{s}_0},\ \phi_{\text{syn}}\right)$          ▷ Synthesize strategy for horizon h
 6          if $\pi_{\text{II}} \neq \emptyset$ then
 7              $\Pi_i \leftarrow \Pi_i \cup (\pi_{\text{II}}, \pi_h, \varphi)$, $h$++           ▷ Save synthesized strategy
 8          else $stop \leftarrow \top$
 9      $\mathsf{Prune}\,(\Pi_t)$, $\Pi^*_{\text{II}} \leftarrow \Pi^*_{\text{II}} \cup \Pi_t$                    ▷ Prune subgame strategies
10      $\ell \leftarrow \ell \cdot (\mathsf{Reachable}\,(\Pi_t) \setminus \ell)$, $i$++                  ▷ update reachability
```

visit a number of targets after being airborne from a known base (initial state), while avoiding hazard zones that are known a priori. Moreover, the presence of adversarial stealthy attacks via GPS spoofing is assumed. We use the DAG framework to synthesize strategies for both the UAV and an operator advisory system (AS) that schedules geolocation tasks for the operator.

Modeling. We model the system as a delayed-action game $\mathcal{G}_{\mathsf{D}}$, where PL_{I} and PL_{II} represent the adversary and the UAV-AS coalition, respectively. Fig. 9 shows the model primary and auxiliary components. In the UAV model $\mathcal{M}_{\text{uav}}$, $x_{\mathcal{B}} = (\mathsf{x}_{\mathcal{B}}, \mathsf{y}_{\mathcal{B}})$ encodes the UAV belief, and $A_{\text{uav}} = \{\mathsf{N}, \mathsf{S}, \mathsf{E}, \mathsf{W}, \mathsf{NE}, \mathsf{NW}, \mathsf{SE}, \mathsf{SW}\}$ is the set of available movements. The AS can trigger the action *activate* to initiate a geolocation task, attempting to confirm the current location. The adversary behavior is abstracted by $\mathcal{M}_{\text{adv}}$ where $x_{\mathcal{T}} = (\mathsf{x}_{\mathcal{T}}, \mathsf{y}_{\mathcal{T}})$ encodes the UAV true location. The adversarial actions are limited to one directional increment at most.[7] If, for example, the UAV is heading N, then the adversary set of actions is $\beta(\mathsf{N}) = \{\mathsf{N}, \mathsf{NE}, \mathsf{NW}\}$. The auxiliary components $\mathcal{M}_{\text{mwr}}$ and $\mathcal{M}_{\text{mrd}}$ manage a FIFO memory stack $(m_i)_{i=0}^{n-1} \in A^n_{\text{uav}}$. The last UAV movement is saved in m_i by synchronizing $\mathcal{M}_{\text{mwr}}$ with $\mathcal{M}_{\text{uav}}$ via *write*, while $\mathcal{M}_{\text{mrd}}$ synchronizes with $\mathcal{M}_{\text{adv}}$ via *read* to read the next UAV action from m_j. The subgame terminates whenever action *write* is attempted and $\mathcal{M}_{\text{mwr}}$ is at state n (i.e., out of memory).

The goal is to find strategies for the UAV-AS coalition based on the following:

– *Target reachability.* To overcome cases where targets are unreachable due to hazard zones, the label *reach* is assigned to the set of states with acceptable checkpoint locations (including the target) to render the objective incrementally feasible. The objective for all encountered subgames is then formalized as $\text{Pr}_{\max}\,[\mathsf{F}\ reach] \geqslant p_{\min}$ for some bound $p_{\min}$.
– *Hazard Avoidance.* Similar to target reachability, the label *hazard* is assigned to states corresponding to hazard zones. The objective $\text{Pr}_{\max}\,[\mathsf{G}\ \neg hazard] \geqslant p_{\min}$ is then specified for all encountered subgames.

[7] To detect aggressive attacks, techniques from literature (e.g., [25,26,16]) can be used.

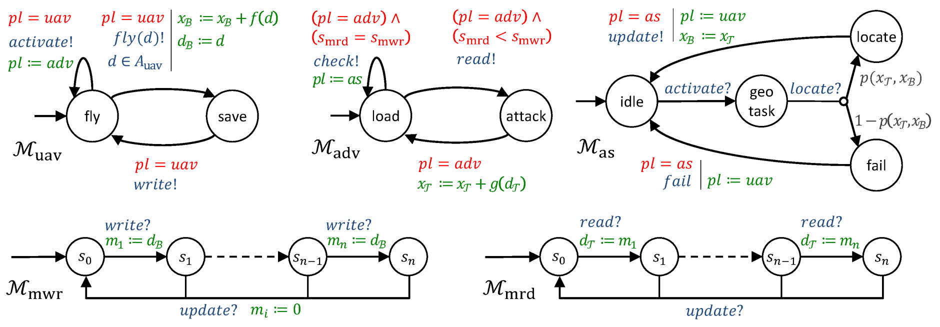


Fig. 9. Primary DAG components: UAV ($\mathcal{M}_{\text{uav}}$), adversary ($\mathcal{M}_{\text{adv}}$), and AS ($\mathcal{M}_{\text{as}}$). Auxiliary DAG components: memory write ($\mathcal{M}_{\text{mwr}}$) and memory read ($\mathcal{M}_{\text{mrd}}$) models, capturing the DAG representation. At stage i, the next memory location to write/read is m_i.

By refining the aforementioned objectives, synthesis queries are used for both the subgames and the supergame. Specifically, the query

$$\phi_{\text{syn}}(k) \coloneqq \langle\!\langle \text{uav} \rangle\!\rangle \text{Pr}_{\max=?} \left[\neg \textit{hazard} \; \mathsf{U}^{\leqslant k} \; (\textit{locate} \wedge \textit{reach}) \right] \tag{1}$$

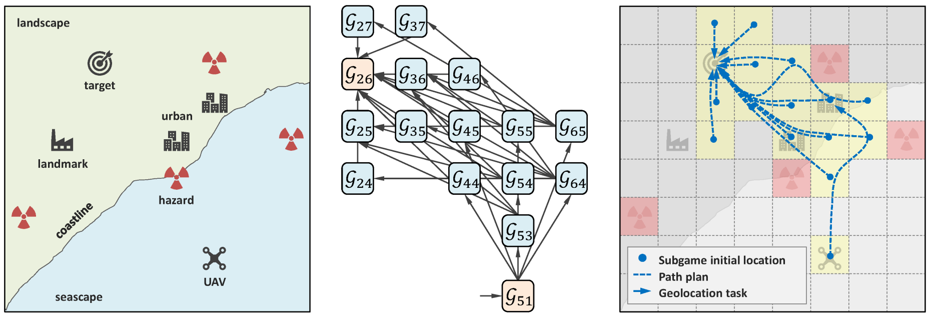
is specified for each encountered subgame $\hat{\mathcal{G}}_i$, where *locate* indicates a successful geolocation task. By following Algorithm 2 for a q number of reachable subgames, the supergame is reduced to an MDP $\mathcal{G}_{\mathsf{D}}^{\{\pi_i\}_{i=1}^{q}}$ (whose states are the reachable subgames), which is checked against the query

$$\phi_{\text{ana}}(n) \coloneqq \langle\!\langle \text{adv} \rangle\!\rangle \text{Pr}_{\min,\max=?} \left[\mathsf{F}^{\leqslant n} \; \textit{target} \right] \tag{2}$$

to find the bounds on the probability that the target is reached under a maximum number of geolocation tasks n.

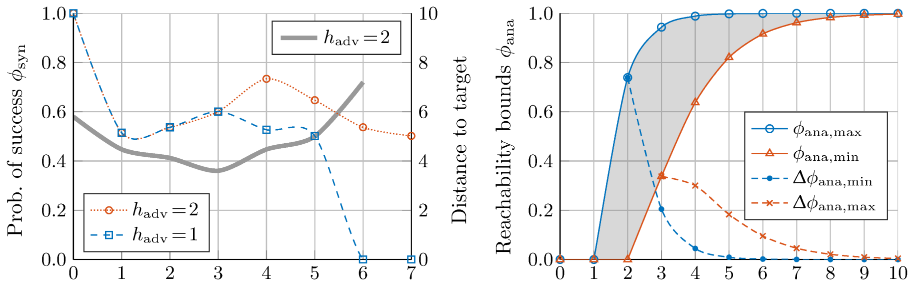
Experimental Results. Fig. 10(a) shows the map setting used for implementation. The UAV's ability to actively detect an attack depends on both its belief and the ground truth. Specifically, the probability of success in a geolocation task mainly relies on the disparity between the belief and true locations, captured by $f_{\text{dis}} \colon Ev(x_{\mathcal{B}}) \times Ev(x_{\mathcal{T}}) \to [0,1]$, obtained by assigning probabilities for each pair of locations according to their features (e.g., landmarks) and smoothed using a Gaussian 2D filter. A thorough experimental analysis where probabilities are extracted from experiments with human operators is described in [11]. The set of hazard zones include the map boundaries to prevent the UAV from reaching boundary values. Also, the adversary is prohibited from launching attacks for at least the first step, a practical assumption to prevent the UAV model from infinitely bouncing around the target location.

We implemented the model in PRISM-games [7,19] and performed the experiments on an Intel Core i7 4.0 GHz CPU, with 10GB RAM dedicated to the tool. Fig. 10(b) shows the supergame obtained by following the procedure in Algorithm 2. A vertex $\hat{\mathcal{G}}_{\mathtt{xy}}$ represents a subgame (composed with its strategy) that starts at location $(\mathtt{x}, \mathtt{y})$, while the outgoing edges points to subgames reachable from the current one. Note that each edge represents a probabilistic transition.


landscape
target
urban
landmark
hazard
coastline
UAV
seascape
$\mathcal{G}_{27}$ $\mathcal{G}_{37}$ $\mathcal{G}_{26}$ $\mathcal{G}_{36}$ $\mathcal{G}_{46}$ $\mathcal{G}_{25}$ $\mathcal{G}_{35}$ $\mathcal{G}_{45}$ $\mathcal{G}_{55}$ $\mathcal{G}_{65}$ $\mathcal{G}_{24}$ $\mathcal{G}_{44}$ $\mathcal{G}_{54}$ $\mathcal{G}_{64}$ $\mathcal{G}_{53}$ $\mathcal{G}_{51}$
Subgame initial location
Path plan
Geolocation task

(a) Environment setup. (b) Supergame $\mathcal{G}_\mathsf{D}$. (c) Protocols.

Fig. 10. (a) The environment setup used for the case study; (b) the induced supergame MDP, where the subgames form its states; and (c) the synthesized protocols.



(a) Geolocation task at stage h (b) Max. no. of geolocation tasks n

Fig. 11. Analysis results for (a) subgame $\hat{\mathcal{G}}_{51}$ and (b) supergame $\mathcal{G}_\mathsf{D}$.

Subgames with more than one outgoing transition imply nondeterminism that is resolved by the adversary actions. Hence, the directed graph depicts an MDP.

The synthesized strategy for (h_{adv}=2, h=4) is demonstrated in Fig. 10(c). For the initial subgame, Fig. 11(a) shows the maximum probability of a successful geolocation task if performed at stage h, and the remaining distance to target. Assuming the adversary can launch attacks after stage $h_{\mathrm{adv}}=2$, the detection probability is maximized by performing the geolocation task at step 4, and hazard areas can still be avoided up till h=6. For h_{adv}=1, however, h=3 has the highest probability of success, which diminishes at h=6 as no possible flight plan exists without encountering a hazard zone. The effect of the maximum number of geolocation tasks (n) on target reachability is studied by analyzing the supergame against ϕ_{ana} as shown in Fig. 11(b). The minimum number of geolocation tasks to guarantee a non-zero probability of reaching the target (regardless of the adversary strategy) is 3 with probability bounds of $(33.7\%, 94.4\%)$.

The experimental data obtained for this case study are listed in Table 1. For the same grid size, more complex maps require more time for synthesis while the state space size remains unaffected. The state space grows exponentially with the explored horizon size, i.e., $\mathcal{O}\left((|A_{\mathrm{uav}}||A_{\mathrm{adv}}|)^h\right)$, and is typically slowed by,

Table 1. Results for strategy synthesis using queries ϕ_{syn} and ϕ_{ana}.

Subgame $\hat{\mathcal{G}}_{51}$			Model Size			Time (sec)		
Map	t_{adv}	k	States	Transitions	Choices	Model	ϕ_{syn}	ϕ_{ana}
8×8	1	4	11,608	17,397	15,950	2.810	0.072	–
		5	57,129	87,865	83,267	14.729	0.602	–
		6	236,714	366,749	359,234	62.582	1.293	–
		7	876,550	1,365,478	1,355,932	231.741	6.021	–
	2	4	6,678	9,230	8,394	2.381	0.042	–
		5	33,904	48,545	45,354	10.251	0.367	–
		6	141,622	204,551	198,640	37.192	1.839	–
		7	524,942	763,144	754,984	145.407	8.850	–
Supergame $\mathcal{G}_{\text{D}}$			6,212	8,306	6,660	2.216	–	2.490

e.g., the presence of hazard areas, since the branches of the game transitions are trimmed upon encountering such areas. Interestingly, for h=6 and h=7, while the model construction time (size) for h_{adv}=1 is almost twice (quadruple) as those for h_{adv}=2, the time for checking ϕ_{syn} declines in comparison. This reflects the fact that, in case of h_{adv}=1 compared to h_{adv}=2, the UAV has higher chances to reach a hazard zone for the same k, leading to a shorter time for model checking.

6 Discussion and Conclusion

In this paper, we introduced DAGs and showed how they can simulate HIGs by delaying players' actions. We also derived a DAG-based framework for strategy synthesis and analysis using off-the-shelf SMG model checkers. Under some practical assumptions, we showed that DAGs can be decomposed into independent subgames, utilizing parallel computation to reduce the time needed for model analysis, as well as the size of the state space. We further demonstrated the applicability of the proposed framework on a case study focused on synthesis and analysis of active attack detection strategies for UAVs prone to cyber attacks.

DAGs come at the cost of increasing the total state space size as $\mathcal{M}_{\text{mrd}}$ and $\mathcal{M}_{\text{mwr}}$ are introduced. This does not present a significant limitation due to the compositional approach towards strategy synthesis using subgames. However, the synthesis is still limited to model sizes that off-the-shelf tools can handle.

The concept of delaying actions implicitly assumes that the adversary knows the UAV actions a priori. This does not present a concern in the presented case study as an abstract (i.e., nondeterministic) adversary model is analogous to synthesizing against the worst-case attacking scenario. Nevertheless, strategies synthesized using DAGs (and SMGs in general) are inherently conservative. Depending on the considered system, this can easily lead to no feasible solution.

The proposed synthesis framework ensures preservation of safety properties. Yet, general reward-based strategy synthesis is to be approached with care. For example, rewards dependent on the belief can appear in any state, and exploring hypothetical branches is not required. However, rewards dependent on a state's true value should only appear in proper states, and all hypothetical branches are to be explored. A detailed investigation of how various properties are preserved by DAGs, along with multi-objective synthesis, is a direction for future work.

References

1. Baier, C., Brazdil, T., Grosser, M., Kucera, A.: Stochastic game logic. In: Quantitative Evaluation of Systems, 2007. QEST 2007. Fourth International Conference on the. pp. 227–236. IEEE (2007)
2. Basset, N., Kwiatkowska, M., Topcu, U., Wiltsche, C.: Strategy synthesis for stochastic games with multiple long-run objectives. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 256–271. Springer (2015)
3. Basset, N., Kwiatkowska, M., Wiltsche, C.: Compositional strategy synthesis for stochastic games with multiple objectives. Information and Computation (2017)
4. Brázdil, T., Chatterjee, K., Křetínskỳ, J., Toman, V.: Strategy representation by decision trees in reactive synthesis. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 385–407. Springer (2018)
5. Chatterjee, K., Henzinger, T.A.: Semiperfect-information games. In: International Conference on Foundations of Software Technology and Theoretical Computer Science. pp. 1–18. Springer (2005)
6. Chen, T., Forejt, V., Kwiatkowska, M., Parker, D., Simaitis, A.: Automatic verification of competitive stochastic systems. Formal Methods in System Design **43**(1), 61–92 (2013)
7. Chen, T., Forejt, V., Kwiatkowska, M., Parker, D., Simaitis, A.: Prism-games: A model checker for stochastic multi-player games. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 185–191. Springer (2013)
8. Chen, T., Forejt, V., Kwiatkowska, M., Simaitis, A., Wiltsche, C.: On stochastic games with multiple objectives. In: International Symposium on Mathematical Foundations of Computer Science. pp. 266–277. Springer (2013)
9. Chen, T., Kwiatkowska, M., Simaitis, A., Wiltsche, C.: Synthesis for multi-objective stochastic games: An application to autonomous urban driving. In: International Conference on Quantitative Evaluation of Systems. pp. 322–337. Springer (2013)
10. David, A., Jensen, P.G., Larsen, K.G., Mikučionis, M., Taankvist, J.H.: Uppaal stratego. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 206–211. Springer (2015)
11. Elfar, M., Zhu, H., Cummings, M.L., Pajic, M.: Security-aware synthesis of human-uav protocols. In: Proceedings of 2019 IEEE International Conference on Robotics and Automation (ICRA). IEEE (2019)
12. Feng, L., Wiltsche, C., Humphrey, L., Topcu, U.: Synthesis of human-in-the-loop control protocols for autonomous systems. IEEE Transactions on Automation Science and Engineering **13**(2), 450–462 (2016)
13. Fremont, D.J., Seshia, S.A.: Reactive control improvisation. In: International conference on computer aided verification. pp. 307–326. Springer (2018)
14. Fu, J., Topcu, U.: Integrating active sensing into reactive synthesis with temporal logic constraints under partial observations. In: 2015 American Control Conference (ACC). pp. 2408–2413. IEEE (2015)
15. Hansen, E.A., Bernstein, D.S., Zilberstein, S.: Dynamic programming for partially observable stochastic games. In: AAAI. vol. 4, pp. 709–715 (2004)
16. Jovanov, I., Pajic, M.: Relaxing integrity requirements for attack-resilient cyber-physical systems. IEEE Transactions on Automatic Control (2019)

17. Kelmendi, E., Krämer, J., Kretinsky, J., Weininger, M.: Value iteration for simple stochastic games: Stopping criterion and learning algorithm. In: International conference on computer aided verification. pp. 623–642. Springer (2018)
18. Klein, F., Zimmermann, M.: How much lookahead is needed to win infinite games? In: International Colloquium on Automata, Languages, and Programming. pp. 452–463. Springer (2015)
19. Kwiatkowska, M., Parker, D., Wiltsche, C.: Prism-games: verification and strategy synthesis for stochastic multi-player games with multiple objectives. International Journal on Software Tools for Technology Transfer **20**(2), 195–210 (2018)
20. Lesi, V., Jovanov, I., Pajic, M.: Security-aware scheduling of embedded control tasks. ACM Transactions on Embedded Computing Systems (TECS) **16**(5s), 188:1–188:21 (2017). https://doi.org/10.1145/3126518
21. Li, W., Sadigh, D., Sastry, S.S., Seshia, S.A.: Synthesis for human-in-the-loop control systems. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 470–484. Springer (2014)
22. Mo, Y., Sinopoli, B.: On the performance degradation of cyber-physical systems under stealthy integrity attacks. IEEE Transactions on Automatic Control **61**(9), 2618–2624 (2016)
23. Neider, D., Topcu, U.: An automaton learning approach to solving safety games over infinite graphs. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 204–221. Springer (2016)
24. Norman, G., Parker, D., Zou, X.: Verification and control of partially observable probabilistic real-time systems. In: International Conference on Formal Modeling and Analysis of Timed Systems. pp. 240–255. Springer (2015)
25. Pajic, M., Lee, I., Pappas, G.J.: Attack-resilient state estimation for noisy dynamical systems. IEEE Transactions on Control of Network Systems **4**(1), 82–92 (March 2017). https://doi.org/10.1109/TCNS.2016.2607420
26. Pajic, M., Weimer, J., Bezzo, N., Sokolsky, O., Pappas, G.J., Lee, I.: Design and implementation of attack-resilient cyberphysical systems: With a focus on attack-resilient state estimators. IEEE Control Systems **37**(2), 66–81 (April 2017). https://doi.org/10.1109/MCS.2016.2643239
27. Rasmusen, E., Blackwell, B.: Games and information. Cambridge, MA **15** (1994)
28. Svoreňová, M., Kwiatkowska, M.: Quantitative verification and strategy synthesis for stochastic games. European Journal of Control **30**, 15–30 (2016)
29. Wiltsche, C.: Assume-Guarantee Strategy Synthesis for Stochastic Games. Ph.D. thesis, Ph. D. dissertation, Department of Computer Science, University of Oxford (2015)
30. Zimmermann, M.: Delay games with wmso+ u winning conditions. RAIRO-Theoretical Informatics and Applications **50**(2), 145–165 (2016)