Multiplayer Reach-Avoid Games via Pairwise Outcomes

Mo Chen, Zhengyuan Zhou and Claire J. Tomlin

***Abstract*—A multiplayer reach-avoid game is a differential game between two adversarial teams, each of N cooperative players playing on a compact domain with obstacles. The attacking team aims to send M of the N attackers to some target location, while the defending team aims to prevent that by capturing attackers or indefinitely delaying attackers from reaching the target. The analysis of this game plays an important role in collision avoidance, motion planning, aircraft control, among other applications involving cooperative agents. The optimal solution to this game is not only difficult to intuit, but also computationally intractable when $N > 1$ due to the high dimensionality of the joint state space of all players. In this paper, we present two different approaches for solving the reach-avoid game in the $N = 1$ case to determine pairwise outcomes, and a graph theoretic maximum matching approach that merges together these pairwise outcomes for an $N > 1$ solution that provides guarantees on the performance of the defending team. We will show that the four-dimensional Hamilton-Jacobi-Isaacs approach allows for real-time updates to the maximum matching in special cases, and that the two-dimensional "path defense" approach is in general considerably more scalable with the number of players and trades off optimality for complexity while maintaining defender performance guarantees.**

I. Introduction

Multiplayer reach-avoid games are differential games between two adversarial teams of cooperative players playing on a compact domain with obstacles. One team, called the attacking team, aims to send as many team members, called attackers, to some target set as quickly as possible. The other team, called the defending team, seeks to delay or prevent the attacking team from doing so by attempting to capture the attackers. Such differential games have been studied extensively [1], [2], and are not only useful in analyzing games such as Capture-the-Flag (CTF), which was explored most notably in the Cornell Roboflag competition [3], [4], [5], [6]. The multiplayer reach-avoid game that we present in this paper is also a powerful theoretical tool for analyzing realistic situations. Potential applications include autonomous robots maneuvering in a warehouse trying to reach preset destinations while avoiding other robots, and aircrafts trying to reach a target airport while avoiding other aircrafts in the vicinity, among other applications in robotics, aircraft control, security, and other domains [7], [8], [9].

The multiplayer reach-avoid game is difficult to analyze for several reasons. First, the two teams have conflicting and asymmetric goals while at the same time, complex cooperation among the players within each team may exist. The optimal control for each player is difficult to intuit and visualize even in the case of a 1 vs. 1 game, in which human players sometimes lose in situations in which an optimal winning strategy exists [1], [2]. Also, in the general multiplayer reach-avoid game, optimal solutions are impossible to compute using traditional dynamic programming approaches due to the intrinsic high dimensionality of the joint state space.

Multiplayer differential games have been previously addressed using various techniques. In [3], where a team of defenders assumes that the attackers move towards their target in straight lines, a mixed-integer linear programming approach was used. In [10], optimal defender strategies could also be determined using a mixed integer linear program, with the assumption that the attackers use a linear feedback control law; the mixed integer linear program was then relaxed and a linear program was solved instead. In complex pursuit-evasion games where players may change roles over time, a nonlinear model-predictive control [11] approach has been investigated. In this case, a cost function of the opponent was assumed, and opponent strategies were estimated based on an explicit prediction model. Approximate dynamic programming (ADP) [12] has also been used to analyze reach-avoid games. While in some cases ADP provides good estimates of the value function that would be obtained from dynamic programming, guarantees on the direction of conservatism typically cannot be made.

Although the above techniques provide some useful insight into multiplayer differential games, they only work well when strong assumptions are made or when accurate models of the opposing team can be obtained. Furthermore, the techniques cannot be easily adapted to solve a general multiplayer reach-avoid game when no prior information of the control strategies of each team is known. To solve general reach-avoid games, the Hamilton-Jacobi-Isaacs (HJI) approach [13] is ideal when the game is low-dimensional. The approach involves solving an HJI partial differential equation (PDE) on a grid to compute a reach-avoid set in the joint state space of the players. The reach-avoid set partitions the joint state space of the players into a winning region for the defending team and a winning region for the attacking team, assuming both sides play optimally. The optimal strategies can then be extracted from the gradient of the solution to the HJI PDE. This

This work has been supported in part by NSF under CPS:ActionWebs (CNS-931843), by ONR under the HUNT (N0014-08-0696) and SMARTS (N00014-09-1-1051) MURIs and by grant N00014-12-1-0609, by AFOSR under the CHASE MURI (FA9550-10-1-0567).

M. Chen, and C. J. Tomlin are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, USA {mochen72,tomlin}@eecs.berkeley.edu

Z. Zhou is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA zyzhou@stanford.edu

We thank Haomiao Huang for sharing MatLab code for 4D HJI calculations.

approach is particularly useful because of the many numerical tools [14], [15], [16] available to numerically solve the HJI PDE, and has been able to solve a variety of optimal control problems such as aircraft collision avoidance [14], automated in-flight refueling [17], and two-player reach-avoid games [2]. The advantage of the HJI approach is that it can be applied to a large variety of player dynamics and does not explicitly assume any control strategy or prediction models for the players. However, the approach cannot be directly applied to our multiplayer reach-avoid game because the complexity of solving a PDE on a grid scales exponentially with the number of players, making the approach only applicable to the two-player game. Even in the two-player case, the memory requirement is significant even for a relatively coarse grid. Therefore, when analyzing the multiplayer reach-avoid game, complexity-optimality trade-offs must be made.

[18] presented an open-loop approach to approximate the solution to the two-player reach-avoid game, in which the time for the attacker to reach the target was conservatively estimated by assuming that the defender first chooses an open-loop control strategy, after which the attacker chooses a control strategy in response. The open-loop approach utilizes a modified version of the fast marching method (FMM) [15], [18], and is extremely computationally efficient. Another advantage of this approach is that it is conservative towards the defender and provides guarantees on the defender's performance. However, the degree of conservatism can be very high because the defender is assumed to use an open-loop control strategy. The open-loop approach, therefore, can be considered an extreme trade off between the complexity of solving the HJI PDE and the optimality of the control strategy.

Our contributions in this paper are in the analysis of both the two-player reach-avoid game and the multiplayer reach-avoid game. For the two-player reach-avoid game, we present two methods for solving it to characterize pairwise outcomes between the players of opposite teams. First, we present the HJI solution to the two-player game [2], which computes a four-dimensional (4D) reach-avoid set that determines which player wins the game assuming both players use the closed-loop optimal control strategy. The 4D reach-avoid set can also provide the optimal controls, and be used to provide real-time updates to the pairwise outcomes at virtually no additional computation cost. Next, we present the new "path defense" solution, an efficient and conservative approximation to the HJI solution in which the defenders utilize a "semi-open-loop" control strategy. This path defense approach conservatively approximates two-dimensional (2D) slices of the reach-avoid sets given by the HJI solution by solving a series of 2D Eikonal equations using FMM, while maintaining guarantees on the defending team's performance. The computation complexity in the path defense approach is thus vastly reduced compared to the HJI approach, and the degree of conservatism is much lower than the degree of conservatism in the open-loop approach. Furthermore, the path defense approach is scales only linearly with the number of players on each team, as opposed to quadratically in the HJI approach.

For the multiplayer reach-avoid game, we propose a novel way of approximating its solution by merging together the N^2 pairwise outcomes in the game using the graph theoretic maximum matching, which can be efficiently computed by known algorithms [19], [20]. We investigate several nuances associated with applying maximum matching to the HJI solution and the path defense solution to the two-player game. In particular, we will show that in the case in which the players on each team have identical dynamics, only a *single* HJI PDE needs to be solved to characterize *all* pairwise outcomes. Furthermore, we show that in general, when applying maximum matching to the two-player path defense solution, the computational complexity scales linearly with the number of players, as opposed to quadratically in the HJI approach.

II. The Reach-Avoid Problem

A. The Multiplayer Reach-Avoid Game

In our multiplayer reach-avoid game, $2N$ players are partitioned into the set of N attackers, $\{P_{A_i}\}_{i=1}^N = \{P_{A_1}, P_{A_2}, \ldots, P_{A_N}\}$ and the set of N defenders, $\{P_{D_i}\}_{i=1}^N = \{P_{D_1}, \ldots, P_{D_N}\}$, whose states are confined in a bounded, open domain $\Omega \subset \mathbb{R}^2$. The domain Ω is partitioned into $\Omega = \Omega_{\text{free}} \cup \Omega_{\text{obs}}$, where Ω_{free} is a compact set representing the free space in which the $2N$ players can move, while $\Omega_{\text{obs}} = \Omega \setminus \Omega_{\text{free}}$ corresponds to obstacles in the domain.

Let $x_{A_i}, x_{D_j} \in \mathbb{R}^2$ denote the state of players P_{A_i} and P_{D_j}, respectively. Then given initial conditions $x_{A_i}^0, x_{D_i}^0 \in \Omega_{\text{free}}, i = 1, 2, \ldots, N$, we assume the dynamics of the players to be defined by the following decoupled system for $t \geq 0$:

$$\begin{aligned} \dot{x}_{A_i}(t) &= v_{A_i} a_i(t), \qquad x_{A_i}(0) = x_{A_i}^0, \\ \dot{x}_{D_i}(t) &= v_{D_i} d_i(t), \qquad x_{D_i}(0) = x_{D_i}^0, \\ & i = 1, 2, \ldots, N \end{aligned} \tag{1}$$

where v_{A_i}, v_{D_i} represent maximum speeds for P_{A_i} and P_{D_i} respectively, and a_i, d_i represent the controls of P_{A_i} and P_{D_i} respectively. Different players may have different maximum speeds, and we assume that a_i, d_i are drawn from the set $\Sigma = \{\sigma \colon [0, \infty) \to \overline{B}_n \mid \sigma \text{ is measurable}\}$, where $\overline{B}_n$ denotes the closed unit disk in $\mathbb{R}^2$. Furthermore, we constrain the controls of both players to be those which allow the player states to remain within the free space for all time. Denote the joint state of all players by $\mathbf{x} = (\mathbf{x}_A, \mathbf{x}_D)$ where $\mathbf{x}_A = (x_{A_1}, \ldots x_{A_N})$ is the joint state of the attackers $\{P_{A_i}\}_{i=1}^N$, and $\mathbf{x}_D = (x_{D_1}, \ldots, x_{D_N})$ is the joint state of the defenders $\{P_{D_i}\}_{i=1}^N$.

The winning conditions of our reach-avoid game are as follows. The attacking team wins whenever M of the N attackers reach some target set without being captured by the defenders; M is pre-specified with $0 < M \leq N$. The target set is denoted $\mathcal{T} \subset \Omega_{\text{free}}$ and is compact. The defending team wins if it can prevent the attacking team from winning by capturing or indefinitely delaying $N - M + 1$ attackers from reaching $\mathcal{T}$. An illustration of the game setup is shown in Figure 1.

The notion of capture is defined by the capture sets $\mathcal{C}_{ij} \subset \Omega^{2N}, i, j = 1, \ldots, N, i \neq j$, which specify the conditions under which P_{D_i} is captured by P_{A_j}. In general, $\mathcal{C}_{ij}$ can be any set that models capture, collision,

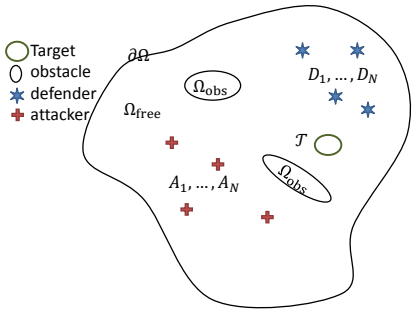


Fig. 1: An illustration of the components of a multiplayer reach-avoid game.

or any situation in which a defender takes an attacker out of the game. In this paper, we will define $\mathcal{C}_{ij}$ to be $\mathcal{C}_{ij} = \left\{\mathbf{x} \in \Omega^{2N} \mid \|x_{A_i} - x_{D_j}\|_2 \leq R_C\right\}$, the interpretation of which is that P_{A_i} is captured by P_{D_j} if P_{A_i}'s position is within a distance R_C of P_{D_j}'s position.

In this paper, we address the following the problems:

1) Given the joint initial states of all players $\mathbf{x^0}$, the target $\mathcal{T}$, and some fixed integer $M, 0 < M \leq N$, can the attacking team win?
2) More generally, given the joint initial states of all players $\mathbf{x^0}$ and the target $\mathcal{T}$, how many attackers can the defending team prevent from reaching the target?

B. The Two-Player Reach-Avoid Game

We will answer the above questions about the general $2N$-player reach-avoid game by using solution to the two-player game as a building block. Thus, it will convenient to specialize the above formulation to the two-player game. In the two-player game, we denote the attacker as P_A, the defender as P_D, their states as x_A, x_D, and their initial conditions as x_A^0, x_D^0. Their dynamics are

$$\begin{aligned} \dot{x}_A(t) &= v_A a(t), \quad x_A(0) = x_A^0, \\ \dot{x}_D(t) &= v_D d(t), \quad x_D(0) = x_D^0 \end{aligned} \tag{2}$$

The players' joint state becomes $\mathbf{x} = (x_A, x_D)$, and their joint initial condition becomes $\mathbf{x^0} = (x_A^0, x_D^0)$. The capture set becomes simply $\mathcal{C} = \left\{(x_A, x_D) \in \Omega^2 \mid \|x_A - x_D\|_2 \leq R_C\right\}$.

The attacker P_A wins if it reaches the target $\mathcal{T}$ without being captured by the defender P_D. The defender P_D wins if it can prevent the attacker from winning by capturing the attacker or indefinitely delaying the attacker from reaching $\mathcal{T}$.

For the two-player reach-avoid game, we seek to answer the following:

1) Given the joint initial state of the attacker and defender $\mathbf{x^0}$ and the target $\mathcal{T}$, can the defender be guaranteed to win?
2) More generally, given the joint initial state of the attacker x_A and the target $\mathcal{T}$, what is the set of initial defender positions in Ω such that the defender is guaranteed to win?

III. The Hamilton-Jacobi-Isaacs Solution of the Two-Player Game

In this section, we describe the HJI approach for solving differential games with arbitrary terrain, domain, obstacles, target set, and player velocities based on [2], [14], [21]. In particular, we will show how to compute the optimal joint control strategies for the attacker and the defender in a two-player reach-avoid game by solving a 4D HJI PDE. This solution allows us to determine whether the defender will win against the attacker in a 1 vs. 1 setting.

In general, to approximate the solution to the N vs. N game, we solve N^2 HJI PDEs corresponding to the N^2 attacker-defender pairs, and then piece together the pairwise solutions using the maximum matching approach described in Section V-A. In Section V-C, we will show that in the special case where players on each team have the same dynamics, only a *single* 4D HJI PDE needs to be solved. This is because the solution to the 4D HJI PDE completely characterizes the outcome of the game given *any* joint initial condition.

A. Hamilton-Jacobi-Isaacs Reachability

The multiplayer reach-avoid game is a differential game in which two teams have competing objectives [22]. The results of the HJI computations assume a closed-loop strategy for both players given previous information of the other players. The setup for using the HJI approach to solve differential games can be found in [2], [14], [21]. In summary, we are given the continuous dynamics of the system state:

$$\dot{\mathbf{x}} = f(\mathbf{x}, u, d), \mathbf{x}(0) = \mathbf{x^0} \tag{3}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the system state, $u \in \mathbb{U}$ is the joint control input of the attacking team, and $d \in \mathbb{D}$ is the joint control input of the defending team. In our reach-avoid game, $n = 4N$ where N is the number of players on each team. The sets $\mathbb{U}$ and $\mathbb{D}$ represent the sets of the joint admissible control inputs of the attacking team and the defending team, respectively. The attacking team selects a control input based on the past and the current joint states of all the players. The defending team then selects a control input based on the past and the present control inputs of the attacking team, in addition to the past and the current joint states. *A priori*, this information pattern is conservative towards the attackers, as defenders have more information available. However, in the case that the system (described by the function f) is decoupled, as in Equation (1), the Isaacs condition [13] holds and this information pattern yields the same optimal solutions for both the attackers and the defenders compared to the analagous information pattern that is conservative towards the defenders.

In the HJI approach, we specify the terminal set R as the attackers' winning condition, and propagate backwards this set for some time horizon, subject to the constraint imposing that the attackers be outside the capture regions and the obstacles. This constraint is described by the avoid set A. The result of the backwards propagation is a reach-avoid set that partitions the state space into two regions. All points inside the reach-avoid set represent the joint initial conditions from

which the attacking team is guaranteed to win within a certain time horizon, and all points outside represent the joint initial conditions from which the attacking team is guaranteed to not win within a certain time horizon.

More precisely, the HJI reachability calculation is as follows. First, given a set G, the level set representation of G is a function $\phi_G : \mathbb{R}^n \to \mathbb{R}$ such that $G = \{\mathbf{x} \in \mathbb{R}^n \mid \phi_G \leq 0\}$. In particular, the terminal set R and the avoid set A are represented by the functions $\phi_R(\mathbf{x})$ and $\phi_A(\mathbf{x})$ respectively.

Let $\Phi : \mathbb{R}^n \times [-T, 0] \to \mathbb{R}$ be the viscosity solution [23] to the constrained terminal value HJI PDE in the form of an HJI variational inequality:

$$\begin{aligned} \max\Big\{ \frac{\partial \Phi}{\partial t} + \min\left[0, H\left(\mathbf{x}, \nabla_{\mathbf{x}} \Phi(\mathbf{x})\right)\right], \\ -\phi_A(\mathbf{x}) - \Phi(\mathbf{x}, t) \Big\} = 0 \\ \Phi(\mathbf{x}, 0) = \phi_R(\mathbf{x}) \end{aligned} \tag{4}$$

where the optimal Hamiltonian is given by

$$H\left(\mathbf{x}, p\right) = \min_{u \in \mathbb{U}} \max_{d \in \mathbb{D}} p \cdot f(\mathbf{x}, u, d)$$

By the argument presented in [14], [24], [25], the set of initial conditions from which the attackers are guaranteed to win within time T is given by

$$\mathcal{RA}_T(R, A) := \{\mathbf{x} \in \mathbb{R}^n \mid \Phi(\mathbf{x}, -T) \leq 0\} \tag{5}$$

Hence, $\Phi(\mathbf{x}, -T)$ is the level set representation of $\mathcal{RA}_T(R, A)$.

The optimal control input for the attacking team is given by [2], [26], [27]:

$$u^*(\mathbf{x}, t) = \arg\min_{u \in \mathbb{U}} \max_{d \in \mathbb{D}} \nabla_{\mathbf{x}} \Phi(\mathbf{x}, t) \cdot f(\mathbf{x}, u, d),\ t \in [-T, 0] \tag{6}$$

Similarly, an initial player configuration outside $\mathcal{RA}_T(R, A)$ guarantees that the defenders will be able to delay the attackers from winning for time T by using the optimal control input

$$d^*(\mathbf{x}, t) = \arg\max_{d \in \mathbb{D}} \nabla_{\mathbf{x}} \Phi(\mathbf{x}, t) \cdot f(\mathbf{x}, u^*, d),\ t \in [-T, 0] \tag{7}$$

Taking $T \to \infty$, we obtain the set of initial conditions from which the attackers are guaranteed to win. We denote this set $\mathcal{RA}_\infty(R, A)$. The set of initial conditions from which the defenders are guaranteed to win is given by all points *not* in $\mathcal{RA}_\infty(R, A)$. For a two-player game on a 2D domain $\Omega \subset \mathbb{R}^2$, the reach-avoid set $\mathcal{RA}_\infty(R, A)$ is 4D.

A highly accurate numerical solution to Equation (4) can be computed using the Level Set Toolbox for MATLAB [21].

Next, we will describe the terminal set and the avoid set for the two player reach-avoid game.

B. Hamilton-Jacobi-Isaacs Reachability for the Two-Player Game

In the two-player reach-avoid game, the goal of the attacker is to reach the target set $\mathcal{T}$ while avoiding capture by the defender. This goal is represented by the attacker being inside $\mathcal{T}$. En route to $\mathcal{T}$, the attacker must avoid capture by the defender. This is represented by the set $\mathcal{C}$.

In addition, both players need to avoid the obstacles Ω_{obs}, which can be considered to be the locations in Ω where the players have zero velocity. In particular, the defender wins if the attacker is in Ω_{obs}, and vice versa. Therefore, we define the terminal set as

$$R = \left\{\mathbf{x} \in \Omega^2 \mid x_A \in \mathcal{T}\right\} \cup \left\{\mathbf{x} \in \Omega^2 \mid x_D \in \Omega_{\text{obs}}\right\} \tag{8}$$

Similarly, we define the avoid set as

$$\begin{aligned} A &= \left\{\mathbf{x} \in \Omega^2 \mid \|x_A - x_D\|_2 \leq R_C\right\} \cup \left\{\mathbf{x} \in \Omega^2 \mid x_A \in \Omega_{\text{obs}}\right\} \\ &= \mathcal{C} \cup \left\{\mathbf{x} \in \Omega^2 \mid x_A \in \Omega_{\text{obs}}\right\} \end{aligned} \tag{9}$$

Given these sets, we can define the corresponding level set representations ϕ_R, ϕ_A, and solve (4). Assuming $\Omega \subset \mathbb{R}^2$, the result is $\mathcal{RA}_\infty(R, A) \in \mathbb{R}^4$, a 4D reach-avoid set with the level set representation $\Phi(\mathbf{x}, -\infty)$. The attacker wins if and only if the joint initial condition is such that $(x_A^0, x_D^0) = \mathbf{x^0} \in \mathcal{RA}_\infty(R, A)$.

If $\mathbf{x^0} \in \mathcal{RA}_\infty(R, A)$, then the attacker is guaranteed to win the game by using the optimal control input given in (6). Applying Equation (6) to the two-player game, we have that the attacker winning strategy satisfies

$$a^*(x_A, x_D, t) = \arg\min_{a \in \mathbb{U}} \max_{d \in \mathbb{D}} p(x_A, x_D, t) \cdot f(x_A, x_D, a, d) \tag{10}$$

for $t \leq 0$. The explicit winning strategy satisfying (10) is given in [2] as

$$a^*(x_A, x_D, t) = -v_A \frac{p_u(x_A, x_D, t)}{\|p_u(x_A, x_D, t)\|_2} \tag{11}$$

where $p = (p_u, p_d) = \nabla_{\mathbf{x}} \Phi(x_A, x_D)$.

Similarly, if $\mathbf{x^0} \notin \mathcal{RA}_\infty(R, A)$, then the defender is guaranteed to win the game by using the optimal control input given in (7). Applying Equation (7) to the two-player game, we have that the defender winning strategy satisfies

$$d^*(x_A, x_D, t) = \arg\max_{d \in \mathbb{D}} p(x_A, x_D, t) \cdot f(x_A, x_D, a^*, d) \tag{12}$$

for $t \leq 0$. The explicit winning strategy satisfying (12) is given in [2] as

$$d^*(x_A, x_D, t) = v_D \frac{p_d(x_A, x_D, t)}{\|p_d(x_A, x_D, t)\|_2} \tag{13}$$

IV. The Path Defense Solution to the Two-Player Game

Solving a 4D HJI PDE involves discretizing a 4D space, and is thus time- and memory-intensive. Instead of computing the entire 4D reach-avoid set given by the HJI solution, we will approximate 2D slices of the reach-avoid set (or simply "2D slices") in the path defense approach. Each slice will be the 4D reach-avoid set sliced at an attacker position.

In this section, we first introduce the *path defense game*, a specific type of reach-avoid game, and then describe an efficient way to approximately compute the defender's winning region in this specific game. Next, we will use the solution to

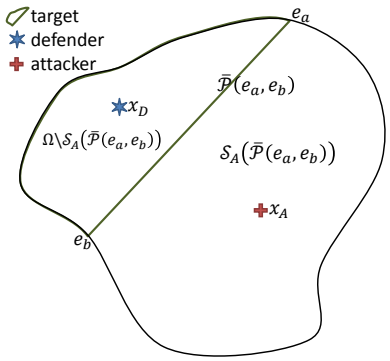


Fig. 2: An illustration of the components of a path defense game between two players.

the path defense game as a building block for approximating 2D slices of the general two-player reach-avoid game. For the path defense approach, we will assume that the defender is not slower than the attacker: $v_A \leq v_D$.

In Section V-D, we will show that when applying maximum matching to merge together pairwise outcomes obtained from the path defense approach as described in Section V-A, the computation complexity will scale linearly with N, the number of players on each team, as opposed to with N^2 in the case in which pairwise solutions are obtained from the HJI approach.

A. The Path Defense Game

The *Path Defense Game* is a two-player reach-avoid game in which the boundary of the target set is the shortest path between two points on $\partial\Omega$, and the target set is on one side of that shortest path. We denote the target set as $\mathcal{T} = \Omega \backslash \mathcal{S}_A\left(\bar{\mathcal{P}}(e_a, e_b)\right)$ for two given points on the boundary e_a, e_b. $\mathcal{S}_A\left(\bar{\mathcal{P}}(e_a, e_b)\right)$ and $\bar{\mathcal{P}}(e_a, e_b)$ are defined below.

Definition 1: **Path of defense**. A path of defense, $\bar{\mathcal{P}}(e_a, e_b)$, is the shortest path between two boundary points $e_a, e_b \in \partial\Omega$. e_a and e_b are referred to as the **anchor points** of path $\bar{\mathcal{P}}(e_a, e_b)$.

For convenience, we denote the shortest path between *any* two points $x, y \in \Omega_{\text{free}}$ to be $\mathcal{P}(x, y)$. The length of $\mathcal{P}(x, y)$ is denoted $\text{dist}(x, y)$, and the time it takes for the attacker and defender to traverse $\mathcal{P}(x, y)$ is denoted $t_A(x, y), t_D(x, y)$, respectively. Note the distinction between $\text{dist}(\cdot, \cdot)$, which denotes distance, and $d(\cdot)$, which denotes control function of the defender.

For convenience, we will also use $\text{dist}(\cdot, \cdot)$ with one or both arguments being sets in Ω. In this case, $\text{dist}(\cdot, \cdot)$ will denote the shortest distance between the objects in the two arguments.

Definition 2: **Attacker's side of the path**. A path of defense $\bar{\mathcal{P}}(e_a, e_b)$ partitions the domain Ω into two regions. Define $\mathcal{S}_A\left(\bar{\mathcal{P}}(e_a, e_b)\right)$ to be the region that contains the attacker, not including points on the path $\bar{\mathcal{P}}(e_a, e_b)$. The attacker seeks to reach the target set $\mathcal{T} = \Omega \backslash \mathcal{S}_A\left(\bar{\mathcal{P}}(e_a, e_b)\right)$.

The basic setup of the path defense game is illustrated in Figure 2.

B. Solving The Path Defense Game

A path defense game can be directly solved by computing a 4D reach-avoid set as described in Section III with $\mathcal{T} = \Omega \backslash \mathcal{S}_A\left(\bar{\mathcal{P}}(e_a, e_b)\right)$, which, as noted earlier, can be time- and memory- intensive. Hence, we propose an efficient way to approximate a 2D slice that is conservative towards the defender. We start with some definitions.

Definition 3: **Defendable path**. Given initial conditions $\mathbf{x^0} = (x_A^0, x_D^0)$, a path $\bar{\mathcal{P}}(e_a, e_b)$ is defendable if regardless of the attacker's actions, the defender has a control function $d(\cdot)$ to prevent the attacker from reaching $\bar{\mathcal{P}}(e_a, e_b)$ without being captured.

Definition 4: **Strongly defendable path**. Given initial conditions $\mathbf{x^0} = (x_A^0, x_D^0)$, a path $\bar{\mathcal{P}}(e_a, e_b)$ is *strongly* defendable if regardless of the attacker's actions, the defender has a control function $d(\cdot)$ to reach $\bar{\mathcal{P}}(e_a, e_b)$ after finite time and prevent the attacker from reaching $\bar{\mathcal{P}}(e_a, e_b)$ without being captured. Note that the defender is not explicitly required to stay on $\bar{\mathcal{P}}(e_a, e_b)$ after reaching it.

Remark 1: A path $\bar{\mathcal{P}}(e_a, e_b)$ is defendable if it is strongly defendable.

Checking whether a path $\bar{\mathcal{P}}(e_a, e_b)$ is defendable is exactly the path defense problem. Since solving the path defense problem involves a 4D reach-avoid set calculation, we instead consider the problem of checking whether a path $\bar{\mathcal{P}}(e_a, e_b)$ is strongly defendable. The following definitions lead to our first Lemma which describes how to determine strong defendability using 2D distance calculations; the definitions and Lemma are illustrated in Figure 3.

Definition 5: **Level set image of attacker**. Given attacker position $x_A(t)$, define the level set image of the attacker with respect to anchor point e_a to be $x_A\prime(t; e_a) = \{x \in \bar{\mathcal{P}}(e_a, e_b) : t_A(x, e_a) = t_A(x_A(t), e_a)\}$. $x_A\prime$ is the unique point on $\bar{\mathcal{P}}(e_a, e_b)$ such that $\text{dist}(x_A\prime, e_a) = \text{dist}(x_A, e_a)$. Define $x_A\prime(t; e_b)$ similarly by replacing e_a with e_b. For convenience, we will sometimes omit the time argument and write $x_A\prime(e_a)$.

Remark 2: $\text{dist}(x_A\prime(e_b), e_a) \leq \text{dist}(x_A\prime(e_a), e_a)$.

Proof: First note that

$$\begin{aligned}\text{dist}(e_a, e_b) &\leq \text{dist}(x_A, e_a) + \text{dist}(x_A, e_b)\\ &\quad \text{(Definition of shortest path between } e_a \text{ and } e_b)\\ &= \text{dist}(x_A\prime(e_a), e_a) + \text{dist}(x_A\prime(e_b), e_b)\\ &\quad \text{(Definition of level set image)}\end{aligned}$$

Then, since the left hand side is given by $\text{dist}(e_a, e_b) = \text{dist}(e_a, x_A\prime(e_b)) + \text{dist}(x_A\prime(e_b), e_b)$, the result follows.

Definition 6: **Capture set**: Define the capture set to be $D_C(y, t) = \{x \mid \|x - y(t)\|_2 \leq R_C\}$. Note that $y(t)$ is in general a time-dependent position, and this set moves with $y(t)$ as a function of time; however, for convenience, we will drop the second argument of D_C when y does not depend on time.

Remark 3: Given some path of defense $\bar{\mathcal{P}}(e_a, e_b)$, suppose the level set image of the attacker is within defender's capture set at some time s:

$$x_A\prime(s;e_a) \in D_C(x_D,s)$$
$$(\text{or } x_A\prime(s;e_b) \in D_C(x_D,s))$$

Then, there exists a control for the attacker to keep the level set image of the attacker within the capture radius of the defender thereafter:

$$x_A\prime(t;e_a) \in D_C(x_D,t)\forall t \geq s$$
$$(\text{or } x_A\prime(t;e_b) \in D_C(x_D,t)\forall t \geq s)$$

This is because the attacker's level set image can move at most as fast as the attacker, whose speed is at most the defender's speed, by assumption.

Definition 7: **Regions induced by point p on path**. Given a point $p \in \bar{\mathcal{P}}(e_a,e_b)$, define a region $\mathcal{R}_a(p)$ associated the point p and anchor point e_a as follows:

$$\mathcal{R}_a(p) = \{x : \text{dist}(x,e_a) \leq \text{dist}(D_C(p),e_a)\} \quad (14)$$

Define $\mathcal{R}_b(p)$ similarly by replacing e_a with e_b.

Now we are ready for our first Lemma, which gives sufficient conditions for strong path defense.

Lemma 1: Suppose that the defender is on some point p on the path $\bar{\mathcal{P}}(e_a,e_b)$, i.e. $x_D^0 = p$. Furthermore, assume that $v_A = v_D$. Then, $\bar{\mathcal{P}}(e_a,e_b)$ is strongly defendable if and only if the attacker's initial position x_A^0 is outside the region induced by p: $x_A^0 \in \Omega\backslash(\mathcal{R}_a \cup \mathcal{R}_b)$.

Proof: The proof is illustrated in Figure 3. We assume $x_A^0 \notin \mathcal{T} = \Omega\backslash\mathcal{S}_A\left(\bar{\mathcal{P}}(e_a,e_b)\right)$, otherwise the attacker would start inside the target set.

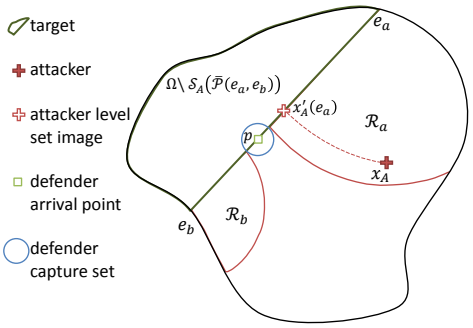
First, we show that if $x_A^0 \in \mathcal{R}_a \cup \mathcal{R}_b$, then the attacker can reach e_a or e_b and hence $\Omega\backslash\mathcal{S}_A\left(\bar{\mathcal{P}}(e_a,e_b)\right)$ without being captured.

Without loss of generality, suppose $x_A^0 \in \mathcal{R}_a$. To capture the attacker, the defender's capture set must necessarily contain $x_A\prime(e_a)$ or $x_A\prime(e_b)$ at some time t. By Definition 7, we have $\text{dist}(x_A^0,e_a) < \text{dist}(D_C(p),e_a)$, so $t_A(x_A\prime(e_a),e_a) < t_D(D_C(p),e_a)$. By Remark 2, we also have $\text{dist}(x_A\prime(e_b),e_a) \leq \text{dist}(x_A\prime(e_a),e_a)$, so it suffices to show that the defender's capture set never reaches $x_A\prime(e_a)$ before the attacker reaches e_a.

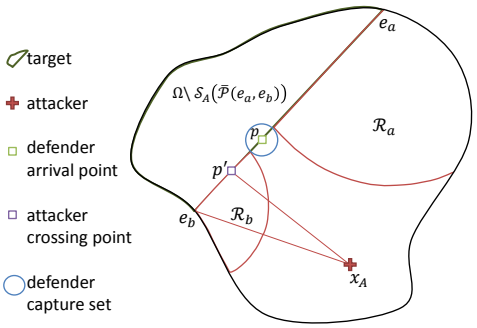
If the attacker moves towards e_a along $\mathcal{P}(x_A^0,e_a)$ with maximum speed, then $x_A\prime(e_a)$ will move towards e_a along $\mathcal{P}(x_A\prime(e_a),e_a)$ at the same speed. Since $t_A(x_A,e_a) = t_A(x_A\prime(e_a),e_a) < t_D(D_C(p),e_a)$, $x_A\prime(e_a)$ will reach e_a before the defender capture set $D_C(x_D,t)$ does. When $x_A\prime(e_a) = e_a$, we also have $x_A = e_a \in \Omega\backslash\mathcal{S}_A\left(\bar{\mathcal{P}}(e_a,e_b)\right)$. Thus, the defender never captures the attacker's level set image. Therefore, no matter what the defender does, the attacker can reach $\Omega\backslash\mathcal{S}_A\left(\bar{\mathcal{P}}(e_a,e_b)\right)$ by moving towards e_a at maximum speed along $\mathcal{P}(x_A^0,e_a)$.

Next, we show, by contradiction, that if $x_A \notin \mathcal{R}_a \cup \mathcal{R}_b$, then the attacker cannot reach $\Omega\backslash\mathcal{S}_A\left(\bar{\mathcal{P}}(e_a,e_b)\right)$ without being captured.

Suppose P_A will reach some point $p\prime$ before $D_C(P_D,t)$ does, i.e. $\text{dist}(x_A^0,p\prime) < \text{dist}(D_C(x_D^0),p\prime) = \text{dist}(D_C(p),p\prime)$. Without loss of generality, assume $p\prime \in \mathcal{P}(p,e_b)$, and note that



(a)



(b)

Fig. 3: (3a) Suppose the attacker is in $\mathcal{R}_a \cup \mathcal{R}_b$. If the attacker moves towards e_a, the defender cannot capture the attacker's level set image before the attacker reaches e_a. (3b) Suppose the attacker is not $\mathcal{R}_a \cup \mathcal{R}_b$, there is no point $p\prime$ on $\bar{\mathcal{P}}(e_a,e_b)$ that the attacker can reach without being captured.

$\text{dist}(D_C(p),e_b) < \text{dist}(x_A^0,e_b)$ since the attacker is not in $\mathcal{R}_b$. Starting with the definition of the shortest path, we have

$$\begin{aligned}
\text{dist}(x_A^0,e_b) &\leq \text{dist}(x_A^0,p\prime) + \text{dist}(p\prime,e_b) \\
&\quad \text{(definition of shortest path / triangle inequality)} \\
&< \text{dist}(D_C(p),p\prime) + \text{dist}(p\prime,e_b) \\
&\quad \text{(by assumption, } \text{dist}(x_A^0,p\prime) < \text{dist}(D_C(p),p\prime)) \\
&= \text{dist}(D_C(p),e_b) \\
\text{dist}(x_A^0,e_b) &< \text{dist}(x_A^0,e_b) \\
&\quad \text{(since } x_A^0 \notin \mathcal{R}_a) \qquad (15)
\end{aligned}$$

This is a contradiction. Therefore, the attacker cannot cross any point $p\prime$ on $\bar{\mathcal{P}}(e_a,e_b)$ without being captured. This proves Lemma 1. ■

Given $x_D^0 = p \in \bar{\mathcal{P}}(e_a,e_b)$, Lemma 1 partitions $\mathcal{S}_A\left(\bar{\mathcal{P}}(e_a,e_b)\right)$ into two regions, assuming $v_A = v_D$: if the attacker is initially in $\mathcal{R}_a \cup \mathcal{R}_b$, then $\bar{\mathcal{P}}(e_a,e_b)$ is not strongly defendable; otherwise, it is strongly defendable. In the case that $v_A < v_D$, the attacker being outside of $\mathcal{R}_a \cup \mathcal{R}_b$ becomes a sufficient condition (not necessary) for the strong defendability of $\bar{\mathcal{P}}(e_a,e_b)$.

In general, x_D^0 may not be on $\bar{\mathcal{P}}(e_a,e_b)$. In this case, to strongly defend $\bar{\mathcal{P}}(e_a,e_b)$, the defender needs to first arrive at some point $p \in \bar{\mathcal{P}}(e_a,e_b)$. If the defender can arrive at p before the attacker moves into $\mathcal{R}_a(p) \cup \mathcal{R}_b(p)$, then p is strongly defendable. Thus, given $\mathbf{x^0} = (x_A^0,x_D^0)$, we may

naively check whether a path $\bar{\mathcal{P}}(e_a, e_b)$ is strongly defendable as follows:

For all points on the path $p \in \bar{\mathcal{P}}(e_a, e_b)$,

1) Compute $t_D(x_D^0, p)$.
2) Compute $t_A\left(x_A^0, \mathcal{R}_a(p) \cup \mathcal{R}_b(p)\right)$.

If there exists some $p \in \bar{\mathcal{P}}(e_a, e_b)$ such that $t_D(x_D^0, p) \leq t_A\left(x_A^0, \mathcal{R}_a(p) \cup \mathcal{R}_b(p)\right)$, then $\bar{\mathcal{P}}(e_a, e_b)$ is strongly defendable.

The above procedure requires two distance computations for *every* point on the path $\bar{\mathcal{P}}(e_a, e_b)$. The next lemma shows that it is necessary and sufficient to perform the computations for only *one* special point denoted p^*. Before presenting the lemma, we first make a remark.

Remark 4: Given $p \in \bar{\mathcal{P}}(e_a, e_b)$, $\text{dist}\left(x_A^0, \mathcal{R}_a(p)\right) = \text{dist}(x_A^0, e_a) - \text{dist}(D_C(p), e_a)$. Similarly, $\text{dist}\left(x_A^0, \mathcal{R}_b(p)\right) = \text{dist}(x_A^0, e_b) - \text{dist}(D_C(p), e_b)$.

Lemma 2: Let a point p^* on the path $\bar{\mathcal{P}}(e_a, e_b)$ be such that $t_A(x_A^0, \mathcal{R}_a) = t_A(x_A^0, \mathcal{R}_b)$. Then, $\bar{\mathcal{P}}(e_a, e_b)$ is strongly defendable if and only if the defender can defend $\bar{\mathcal{P}}(e_a, e_b)$ by first going to p^*.

Proof: One direction is clear: if the defender can defend $\bar{\mathcal{P}}(e_a, e_b)$ by first going to p^*, then $\bar{\mathcal{P}}(e_a, e_b)$ is strongly defendable by definition.

We will show the other direction by showing its contrapositive: if the defender cannot defend $\bar{\mathcal{P}}(e_a, e_b)$ by first going to p^*, then $\bar{\mathcal{P}}(e_a, e_b)$ is not strongly defendable. Equivalently, we will show that if choosing p^* as the first point of entry does not allow the defender to defend $\bar{\mathcal{P}}(e_a, e_b)$, then no other choice of p as the first point of entry does.

Suppose that the defender cannot defend $\bar{\mathcal{P}}(e_a, e_b)$ by choosing p^* as the first point of entry, but can defend $\bar{\mathcal{P}}(e_a, e_b)$ by choosing another point $p\prime$ as the first point of entry. Without loss of generality, assume $\text{dist}(D_C(p^*), e_a) - \text{dist}(D_C(p\prime), e_a) > 0$. This assumption moves $p\prime$ further away from e_a than p^*, causing $\mathcal{R}_a$ to move closer to x_A^0. Starting with Remark 4, we have

$$\begin{aligned} \text{dist}\left(x_A^0, \mathcal{R}_a(p^*)\right) &= \text{dist}(x_A^0, e_a) - \text{dist}(D_C(p^*), e_a) \\ t_A\left(x_A^0, \mathcal{R}_a(p^*)\right) &= t_A(x_A^0, e_a) - t_A(D_C(p^*), e_a) \end{aligned} \tag{16}$$

Similarly, for the point $p\prime$, we have

$$\begin{aligned} \text{dist}\left(x_A^0, \mathcal{R}_a(p\prime)\right) &= \text{dist}(x_A^0, e_a) - \text{dist}(D_C(p\prime), e_a) \\ t_A\left(x_A^0, \mathcal{R}_a(p\prime)\right) &= t_A(x_A^0, e_a) - t_A(D_C(p\prime), e_a) \end{aligned} \tag{17}$$

Then, subtracting the above two equations, we derive that the attacker will be able to get to $\mathcal{R}_a$ sooner by the following amount of time:

$$\begin{aligned} t_A\left(x_A^0, \mathcal{R}_a(p^*)\right) - t_A\left(x_A^0, \mathcal{R}_a(p\prime)\right) &= t_A(D_C(p\prime), e_a) \\ &\quad - t_A(D_C(p^*), e_a) \\ &= t_A(p\prime, p^*) \\ &\geq t_D(p\prime, p^*) \end{aligned} \tag{18}$$

We now show that the defender can get to $p\prime$ sooner than to p^* by less than the amount $t_D(p\prime, p^*)$, and therefore the defender in effect "gains less time" than the attacker does by going to $p\prime$. We assume that $p\prime$ is closer to the defender than p^* is (otherwise the defend would actually "lose time" by going to $p\prime$, which trivially implies the the path cannot be strongly defended). Then, by the triangle inequality, we have

$$\begin{aligned} \text{dist}(x_D^0, p^*) &\leq \text{dist}(x_D^0, p\prime) + \text{dist}(p\prime, p^*) \\ \text{dist}(x_D^0, p^*) - \text{dist}(x_D^0, p\prime) &\leq \text{dist}(p\prime, p^*) \\ t_D(x_D^0, p^*) - t_D(x_D^0, p\prime) &\leq t_D(p\prime, p^*) \end{aligned} \tag{19}$$

This completes the proof. ■

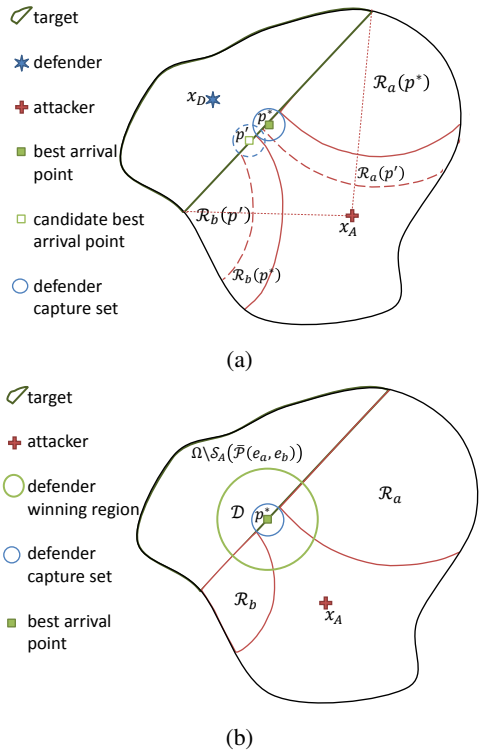


Fig. 4: (4a) If the defender cannot defend $\bar{\mathcal{P}}(e_a, e_b)$ by first going to p^*, then the defender cannot defend $\bar{\mathcal{P}}(e_a, e_b)$ by going to any other point p. (4b) Lemma 2 allows an easy computation of $\mathcal{D}$.

Lemmas 1 and 2 give a simple algorithm to compute, given x_A^0, the region that the defender must be in for a path of defense $\bar{\mathcal{P}}(e_a, e_b)$ to be strongly defendable:

1) Given e_a, e_b, x_A^0, compute p^* and $\mathcal{R}_a(p^*), \mathcal{R}_b(p^*)$.
2) If $v_A = v_D$, then $\bar{\mathcal{P}}(e_a, e_b)$ is strongly defendable if and only if $x_D^0 \in \mathcal{D}(e_a, e_b; x_A^0) = \{x : t_D(x, p^*) \leq t_A(x_A^0, \mathcal{R}_a \cup \mathcal{R}_b)\}$. If $v_A \leq v_D$, then $x_D^0 \in \mathcal{D}(e_a, e_b; x_A^0)$ becomes a sufficient condition for $\bar{\mathcal{P}}(e_a, e_b)$ being strongly defendable.

The computations in this algorithm can be efficiently computed by using FMM [15] on a 2D grid. Thus, we have conservatively solved the path defense problem, orginally a 4D problem, by solving a series of 2D Eikonal equations. Figure 4 illustrates the proof of Lemma 2 and the defender winning region $\mathcal{D}$.

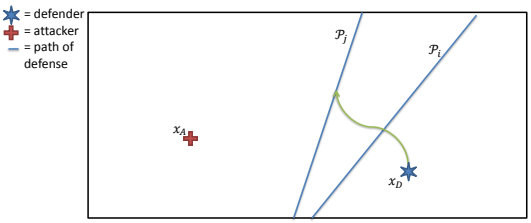


Fig. 5: $\bar{\mathcal{P}}_i$ is strongly defendable if $\bar{\mathcal{P}}_j$ is strongly defendable.

C. The Path Defense Solution to the Reach-Avoid Game

In section II-A, we formulated the two-player reach-avoid game, and in section III, we described how to solve the game by solving a 4D HJI PDE. We now present a more efficient way of solving the two-player reach-avoid game conservatively for the defender. This new approach is based on the path defense game from Section IV-A and only involves 2D distance calculations, which can be solved efficiently using FMM.

In Section IV-A, we described the path defense game as a reach-avoid game with the target set $\mathcal{T} = \Omega \backslash \mathcal{S}_A\left(\bar{\mathcal{P}}(e_a, e_b)\right)$ and computed a 2D slice. We now present the path defense solution to compute 2D slices for an arbitrary target set. The path defense approach leverages the idea described in Section IV-A: If the target set is enclosed by some defendable (in particular, strongly defendable) path $\bar{\mathcal{P}}(e_a, e_b)$ for some e_a, e_b, then the defender can win the game. Checking whether a path of defense is defendable is computationally expensive as it involves solving a 4D HJI PDE. Instead, we check for *strongly* defendable paths. This adds more conservatism towards the defender, but makes computation much more efficient.

Naively, one could fix e_a, then search all other anchor points $e_b \in \partial\Omega$ to find a defendable path. If a defendable path is found, then the defender wins the game; if not, try another e_a. However, we can reduce the number of paths that needs to be checked by only checking only paths of defense $\bar{\mathcal{P}}(e_a, e_b)$ that touch the target set. In a simply connected domain, this reduction in the number of paths checked does not introduce any additional conservatism.

To see this, observe that if a path is strongly defendable, any path "behind" it on the defender's side is also strongly defendable. This is illustrated in Figure 5. Suppose $\bar{\mathcal{P}}_i$ is strongly defendable, then it is defendable. This necessarily implies that $\bar{\mathcal{P}}_j$ is defendable, because if the defender defends $\bar{\mathcal{P}}_i$, then the attacker can never cross $\bar{\mathcal{P}}_i$ and thus never cross $\bar{\mathcal{P}}_j$. Since $\bar{\mathcal{P}}_i$ is strongly defendable, the defender must be able to move to some point on $\bar{\mathcal{P}}_i$, and then defend $\bar{\mathcal{P}}_i$. However, on the way to $\bar{\mathcal{P}}_i$, the defender would have crossed $\bar{\mathcal{P}}_j$. So, the defender is able to arrive at some point on $\bar{\mathcal{P}}_j$ and then defend it afterwards; therefore $\bar{\mathcal{P}}_j$ is strongly defendable. Thus, if we restrict the paths considered to be only those that touch the target set, then picking e_a determines e_b.

If some strongly defendable path $\bar{\mathcal{P}}(e_a, e_b)$ encloses the target set, then the defender's strategy would be to first go to $p^* \in \bar{\mathcal{P}}(e_a, e_b)$, then move towards $x_A\prime(e_a)$ or $x_A\prime(e_b)$ until the level set image is captured. Finally, the defender can simply track the captured level set image. This is a "semi-open-loop" strategy. The first part of the defender's control strategy is to move to p^* regardless of what the attacker does; this is an open-loop strategy. The second part of the defender's control strategy is to move towards and then track the attacker's level set image, which depends on how the attacker moves; this is a closed-loop strategy.

The following algorithm approximates a 2D slice conservatively towards the defender:

Algorithm 1: Given attacker position,

1) Choose some point $e_a \in \partial\Omega$, which defines e_b to create a path of defense $\bar{\mathcal{P}}(e_a, e_b)$ that touches the target $\mathcal{T}$.
2) Repeat step 1 for a desired set of points $e_a \in \partial\Omega$ to create a set of paths of defense.
3) For some particular $\bar{\mathcal{P}}(e_a, e_b)$, determine the defender winning region $\mathcal{D}(e_a, e_b; x_A^0)$.
4) Repeat step 3 for all the paths created in steps 1 and 2.
5) The union $\bigcup_{e_a} \mathcal{D}(e_a, e_b; x_A^0)$ gives the approximate 2D slice, representing the conservative winning region for the defender in the two-player reach-avoid game.

V. From Two-Player to Multiplayer

In principle, the multiplayer reach-avoid game can be analyzed by directly solving the corresponding HJI PDE. However, in our multiplayer reach-avoid game involving N players on each team, the joint state space of all $2N$ players is $4N$-dimensional ($4N$D). Numerical solutions to the HJI PDE are only feasible up to approximately five dimensions [1]; thus, solving the HJI PDE corresponding to a N vs. N game is computationally intractable for $N > 1$, and complexity-optimality trade-offs must be considered.

In this section, we assume that it is known whether or not each defender is guaranteed to win against each attacker in a 1 vs. 1 setting. This information can be obtained from, for example, the HJI solution described in Section III or the path defense solution described in Section IV. Given this knowledge of the outcome of each attacker-defender pair, we construct an approximation to the $4N$D HJI PDE solution using the graph-theoretic maximum matching. The approximation *guarantees* an upper bound on the number of attackers that are able to reach the target.

A. Maximum Matching

We piece together our assumed knowledge of the outcome of each attacker-defender pair using maximum matching to approximate the $4N$D HJI PDE solution as follows:

Algorithm 2:

1) Construct a bipartite graph with two sets of nodes $\{P_{A_i}\}_{i=1}^N, \{P_{D_i}\}_{i=1}^N$, where each node represents a player.
2) Form a bipartite graph: Draw an edge between P_{D_i} and P_{A_j} if P_{D_i} wins against P_{A_j} in a two-player reach-avoid game.
3) Run any matching algorithm to find a maximum matching in the graph. This can be done using, for example, a linear program [19], or the Hopcroft-Karp algorithm [20]. In this paper we use the linear program approach.

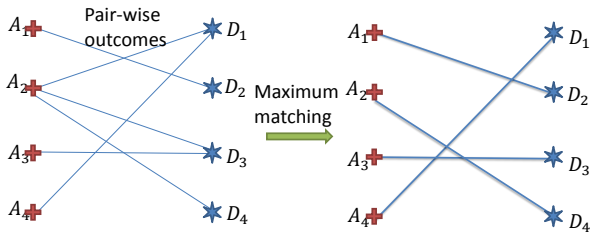


Fig. 6: An illustration of using maximum matching to conservatively approximate the multiplayer reach-avoid game. A bipartite graph is created based on pairwise outcomes. Then, a maximum matching of the bipartite graph is found to optimally assign defender-attacker pairs. A maximum matching of size m indicates that at most $N-m$ attackers will be able to reach the target.

After finding a maximum matching, we can guarantee an upper bound on the number of attackers that will be able to reach the target. If the maximum matching is of size m, then the defending team would be able to prevent *at least* m attackers from reaching the target. In other words, $N-m$ is an upper bound on the number of attackers that can reach the target.

For intuition, consider the following specific cases of m. If the size of the maximum matching is $m = N$, then the defenders can prevent *all* N attackers from reaching the target and thus no attacker will be able to reach the target. If $m = 0$, then there is no initial pairing that will prevent any attacker from reaching the target; however, the attackers are *not* guaranteed to all reach the target, as $N - m = N$ is only an upper bound on the number of attackers who can reach the target. Finally, if $m = N - M + 1$, then the attacking team would only be able to send at most $N - m = M - 1$ attackers to the target. Since M attackers need to reach the target for the attacking team to win, in this case the defending team would be guaranteed to win. The entire procedure of applying maximum matching to the pairwise outcomes is illustrated in Figure 6.

Our solution to the multiplayer reach-avoid game is an approximation to the optimal solution that would be obtained by directly solving the $4N$D HJI PDE; it is conservative for the defending team because by creating defender-attacker pairs, each defender restricts its attention to only one opposing player. Even if the size of maximum matching is zero, not all attackers are guaranteed to reach the target, as the defending team could potentially capture some attackers without using a strategy that creates defender-attacker pairs. Nevertheless, our solution is able to overcome numerical intractability to approximate a $4N$D calculation, and is useful in many game configurations.

B. Time-Varying Defender-Attacker Pairings

The procedure outlined in Section V-A assigns an attacker to each defender that is part of a maximum pairing in an open-loop manner: the assignment is done at the beginning of the game. However, the bipartite graph and its corresponding maximum matching can actually be updated as the players change positions during the game. This update can potentially be performed in real time by the following procedure:

Algorithm 3:

1) Given the position of each defender x_{D_i} and of each attacker x_{A_j}, determine whether x_{A_j} can win for all j.
2) Construct the bipartite graph and find its maximum matching to assign an attacker to each defender that is part of the maximum matching.
3) For a short, chosen duration Δ, apply a winning control input and compute the resulting trajectory for each defender that is part of the maximum matching. For the rest of the defenders and for all attackers, compute the trajectories assuming some (any) control function.
4) Update the player positions after the duration Δ and repeat steps 1 to 3 with the new player positions.

As $\Delta \to 0$, the above procedure computes a bipartite graph and its maximum matching as a function of time. Whenever the maximum matching is not unique, the defenders can choose a different maximum matching and still be guaranteed to prevent the same number of attackers from reaching the target. As long as each defender uses a winning control input against the paired-up attacker, the size of the maximum matching can never decrease as a function of time.

On the other hand, it is possible for the size of the maximum matching to increase as a function of time. This occurs if the joint configuration of the players becomes such that the resulting bipartite graph has a bigger maximum matching than before, which may happen since the size of the maximum matching only gives an upper bound on the number of attackers that are able to reach the target. Furthermore, there is currently no numerically tractable way to compute the joint optimal control input for the attacking team, so a suboptimal strategy from the attacking team can be expected, making an increase of maximum matching size likely. Determining defender control strategies that optimally promote an increase in the size of the maximum matching would be an important step towards the investigation of cooperation. Since the maximum matching of a bipartate is in general not unique, such strategies may involve intelligently choosing among several maximum matchings or adding weights to edges and performing weighted maximum matching. These considerations will be part of our future work.

C. Application to the Two-Player HJI Solution

Solving the 4D HJI PDE (4) with $T \to \infty$ as described in Section III-B gives us $\mathcal{RA}_\infty(R, A)$, which characterizes the pairwise outcome between an attacker-defender pair. In general, when the speed of every player on each team differs from those of all its teammates, solving N^2 4D HJI PDEs, each corresponding to a different pair of attacker speed and defender speed, gives us the pairwise outcomes between every attacker-defender pair. The computation time required is thus CN^2, where C is the time required to solve a single 4D HJI PDE. The pairwise outcomes can then be merged together to approximate the N vs. N game as described in Section V-A. In the special case where each team has a single maximum

speed, i.e. $v_{A_i} = v_A, v_{D_i} = v_D, \forall i$, solving a *single* 4D HJI PDE will characterize all pairwise outcomes.

The solution to the 4D HJI PDE not only characterizes the outcome between an attacker-defender pair given their initial positions, but also characterizes the outcome between any joint-positions of that pair. When using maximum matching to determine how many attackers are guaranteed to not reach the target set, the HJI approach allows for real-time updates of the maximum matching. As the players move on the game domain Ω to new positions, the pairwise outcome between P_{A_i}, P_{D_j} can be determined by simply checking whether (x_{A_i}, x_{D_j}) is in $\mathcal{RA}_\infty(R, A)$.

D. Application to the Two-Player Path Defense Solution

To use the pairwise outcomes determined by the path defense approach as building blocks for approximating the solution to the multiplayer game, we add the following step to Algorithm 1:

6) Repeat steps 3 to 5 for every attacker position.

For a given domain, set of obstacles, and target set, steps 1 and 2 in Algorithm 1 only need to be performed once, regardless of the number of players in the game. In step 3, the speeds of the defenders come in only through a single distance calculation from p^*, and this calculation only needs to be done once per attacker position. Therefore, steps 3 to 5 scale only with the number of attackers. Therefore, the total computation time required for the path defense solution to the multiplayer is on the order of $C_1 + C_2 N$, where C_1 is the time required for steps 1 and 2, C_2 is the time required for steps 3 to 5, and N is the number of players on each team.

VI. Comparison Between the HJI and Path Defense Approaches

In Sections III and IV, we presented two approaches for determining pairwise outcomes in our multiplayer reach-avoid game. The pairwise outcomes determined by either method can then be tied together to approximate the solution to the full N vs. N game using maximum matching according to Section V. We now briefly compare the two approaches.

The HJI approach holds an advantage over the path defense approach in terms of optimality. The HJI solution to the two-player reach-avoid game computes the entire 4D reach-avoid set and gives the optimal closed-loop control strategies for each attacker-defender pair; on the other hand, the path defense solution computes 2D slices of the 4D reach-avoid set sliced at the attacker position, and provides a more conservative semi-open-loop control strategy to the defenders. As a result, the defender winning regions computed in the path defense approach tend to be smaller than those computed in the HJI approach, as seen in Figures 14 and 15. The degree of conservatism is more pronounced when the attacker is slower than the defender and when the domain contains large obstacles, as seen in Figure 15. Furthermore, because the HJI approach computes entire 4D reach-avoid sets, very little additional computation overhead is required for updating pairwise outcomes as the players move during the course of the game.

The path defense approach trades off optimality for computation complexity for the case where $v_{A_i} \leq v_{D_j} \ \forall i, j$. Because the HJI approach involves solving an HJI PDE in a 4D space, the computation time tends to be long and the memory requirements are high. Typically, a single 4D HJI PDE takes approximately half an hour to solve on a relative coarse grid in each dimension. Computation of a 2D slice using the path defense approach can be done on much finer grids with the same amount of memory, and takes only a few seconds. Furthermore, when computing all pair-wise outcomes, the number of 2D slice computations required in path defense approach only scales with N, the number of attackers, whereas the number of 4D HJI PDEs that needs to be solved is N^2.

VII. Numerical Results

In the following subsections, we use a 4 vs. 4 example to illustrate our methods. The game is played on a square domain with obstacles. Defenders have a capture radius of 0.1 units, and all players have the same maximum speed. We first show simulation results for the HJI and path defense approaches for determining pairwise outcomes, and then compare the two approaches. Computations were done on a Lenovo T420s laptop with a Core i7-2640M processor with 4 gigabytes of memory.

A. HJI Formulation

Figures 7 to 9 show the results of solving the 4D HJI PDE in Equation (4) using the terminal set and avoid set in Equations (8) and (9), respectively. Computing the 4D reach-avoid set on a grid with 45 grid points in each dimension took approximately 30 minutes. In general, this makes the computation time required for computing N^2 pair-wise outcomes $CN^2 = 30N^2$ minutes. In our example, however, all players have the same maximum speed, so only a single 4D HJI PDE needed to be solved.

The solution of the 4D HJI PDE characterizes the 4D reach-avoid set $\mathcal{RA}_\infty(R, A)$. To visualize this set in 2D, we take 2D slices of the 4D reach-avoid set sliced at each player's position. Figure 7 shows the 4D reach-avoid set sliced at various *attacker positions*, which gives the all pairwise outcomes. For example, in the left top subplot, the attacker at $(-0.2, 0)$ is guaranteed to win a 1 vs. 1 reach-avoid game against each of the defenders at $(-0.3, 0.5)$ and at $(-0.3, -0.5)$, and guaranteed to lose against each of the defenders at $(0.3, 0.5)$ and at $(0.3, -0.5)$ if they play optimally.

Figure 8 shows the 4D reach-avoid set sliced at various *defender positions*, and characterizes the same pairwise outcomes. For example, in the right top subplot, the defender at $(-0.3, 0.5)$ is guaranteed to win a 1 vs. 1 reach-avoid game against each of the attackers at $(-0.5, 0)$ and at $(-0.2, 0.9)$, and guaranteed to lose against each of the attackers at $(0.7, -0.9)$ and at $(-0.2, 0)$ if they play optimally.

Figure 9 shows the bipartite graph and maximum matching resulting from the pairwise outcomes. In this case, the maximum matching is of size 4, a perfect matching. This guarantees that if each defender plays against the attacker matched by the maximum matching using the strategy in Equation (13), then *no* attacker will be able to reach the target.

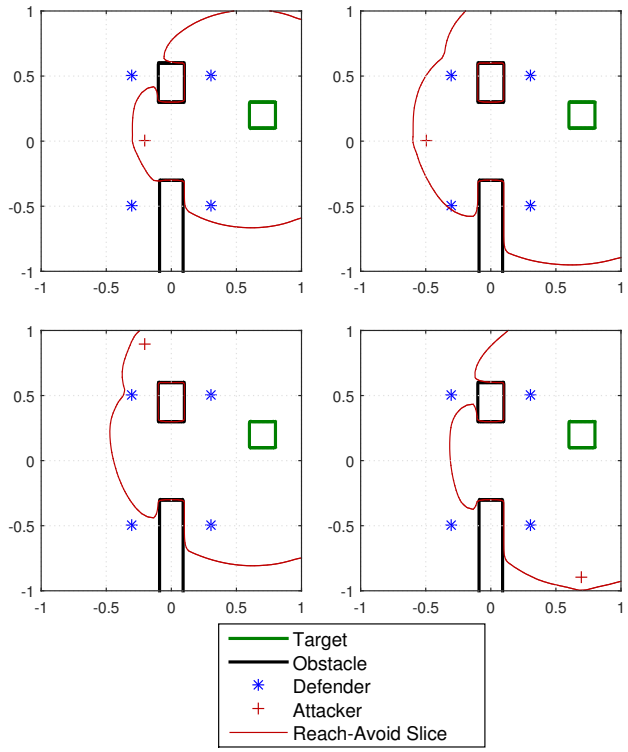


Fig. 7: The 4D reach-avoid set sliced at various attacker positions. In each subplot, defenders "inside" the reach-avoid slice boundary win against the plotted attacker in a 1 vs. 1 setting using the control in Equation (13); defenders "outside" lose.

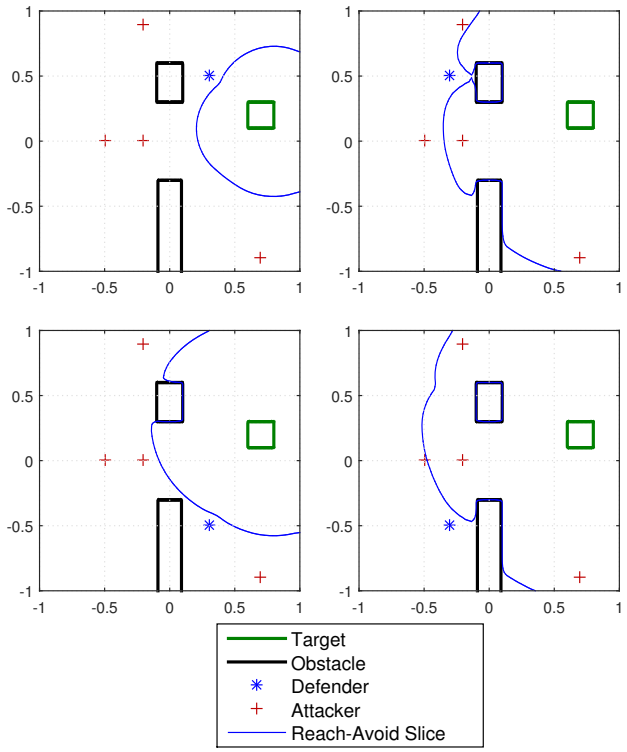


Fig. 8: The 4D reach-avoid set sliced at various defender positions. In each subplot, attackers "inside" the reach-avoid slice boundary win against the plotted defender in a 1 vs. 1 setting using the control in Equation (11); attackers "outside" lose.

B. Path Defense Formulation

Figures 10 to 12 show the results of using the path defense approach to compute conservative approximations of the 4D reach-avoid set sliced at various attacker positions.

Figure 10 shows the defender winning region for a given attacker position (red cross) and path of defense (blue line). If the defender can get to p^* (small light green square) before the attacker can get to $\mathcal{R}_a \cup \mathcal{R}_b$, then the defender will be able to strongly defend the path, and prevent the attacker from reaching any target set that is enclosed by the path (large dark green square). The region within which the defender is able to do this is the region $\mathcal{D}(e_a, e_b)$ (shown in light green).

As an example, refer to the bottom left subplot in Figure 10. Using the semi-open-loop strategy described in Section IV-C, the defenders at $(-0.3, 0.5)$ and at $(0.3, 0.5)$ are able to prevent the attacker at $(-0.2, 0.9)$ from reaching the target.

Figure 11 shows the bipartite graph and maximum matching resulting from the pairwise outcomes. In this case, the maximum matching is of size 3. This guarantees that if each defender plays against the attacker matched by the maximum matching using the semi-open-loop strategy, then *at most* 1 attacker will be able to reach the target.

Computations were done on a 200×200 grid, and 937 paths were used to compute the results in Figure 11. Computation time varies with the number of paths we chose in steps 1 and 2 in the algorithm in Section IV-C. Taking the union of the defender winning regions from more paths will give a less conservative result, but require more computation time. A summary of the performance of our algorithm is shown in Figure 13.

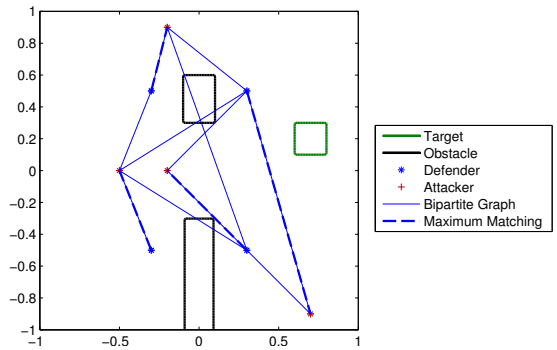


Fig. 9: Tying together pair-wise interactions through maximum matching. Here, a maximum matching of size 4 (a perfect matching) guarantees that *no* attacker will be able to reach the target without being captured if the defenders play optimally according to Equation (13) against their matched attackers.

With 937 paths, the computation of paths took approximately 60 seconds, and the computation of the 2D slice given the set of paths took approximately 30 seconds. However, very few paths are needed to approximate a 2D slice: Even with as few as 30 paths, the computed 2D slice covers more than 95% of the area of the 2D slice computed using 937 paths. This

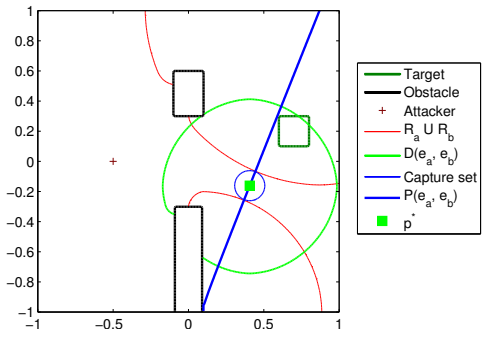


Fig. 10: Defense of a single path that encloses the target set.

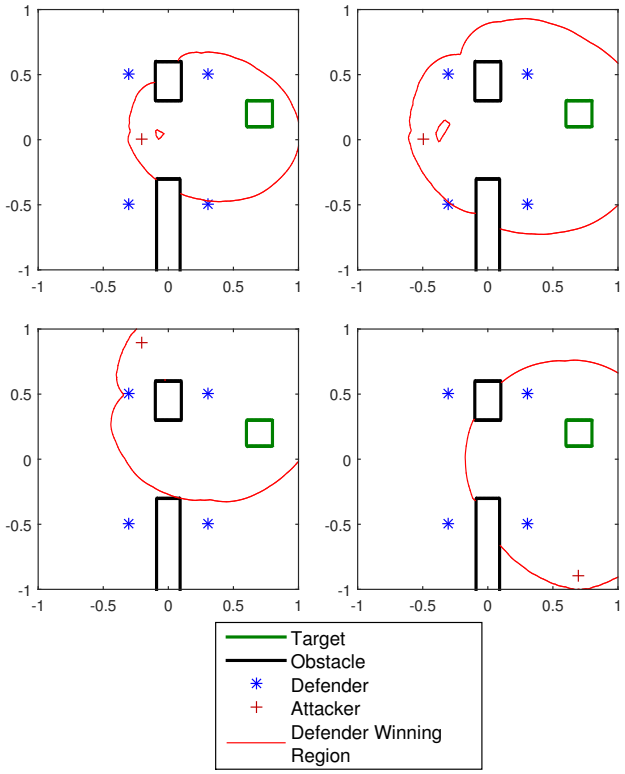


Fig. 11: The reach-avoid set sliced at various attacker positions. In each subplot, defenders "inside" the reach-avoid slice boundary are guaranteed to win against the plotted attacker in a 1 vs. 1 setting using the semi-open-loop control strategy described in Section IV-C.

reduces the computation time of the paths to 2.5 seconds, and the computation time of the 2D slices given the paths to 2.1 seconds. In terms of the complexity constants in Section IV-C, we have that the computation time required for computing all pairwise interactions is $C_1 + C_2 N = 2.5 + 2.1N$ seconds.

C. Comparison Between HJI and Path Defense Formulations

Each pairwise outcome computed with the HJI approach gives the optimal behavior assuming the players utilize a closed-loop strategy. In contrast, each pairwise outcome computed with the path defense approach assumes that the defender is using a semi-open-loop strategy as discussed in Section VI. Figures 14 and 15 compare the 2D slices computed

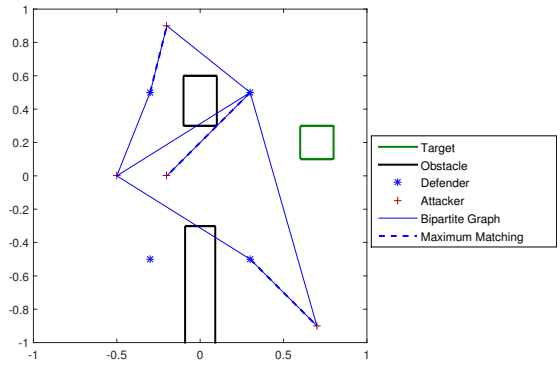


Fig. 12: Tying together pairwise interactions through maximum matching. Here, a maximum matching of size 3 guarantees that *at most* 1 attacker will be able to reach the target without being captured if the defenders use the semi-open-loop control strategy described in Section IV-C against their matched attackers.

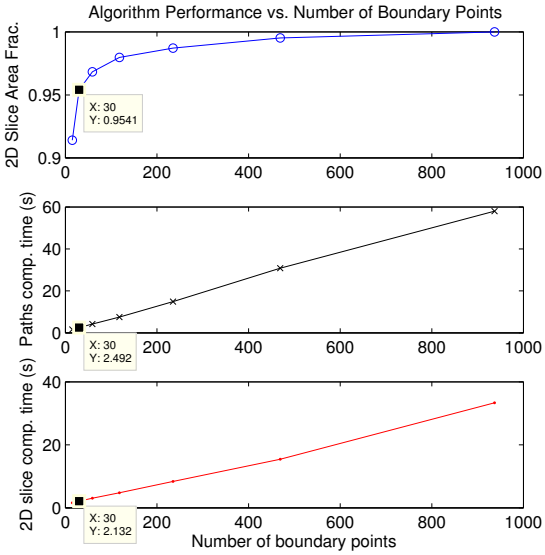


Fig. 13: Performance of the Path Defense Solution.

from the two different approaches.

Figure 14 shows the results for the 4 vs. 4 example in the preceding sections, where all players have the same maximum speed. Because the semi-open-loop strategy in the path defense approach is conservative towards the defender, the computed defender winning region is smaller in the path defense approach. The degree of conservatism depends on the domain, obstacles, target set, and attacker position. In this example, the defender winning regions computed using path defense approach on average covers approximately 75% of the area of the defender winning regions computed using the HJI approach.

The path defense approach becomes more conservative when the attacker is slower than the defender because in this case, the attacker being outside the region $\mathcal{R}_a \cup \mathcal{R}_b$ is not a necessary but only a sufficient condition for the defender to win using the semi-open-loop strategy, as discussed in the Lemmas in Section IV. The path defense approach also

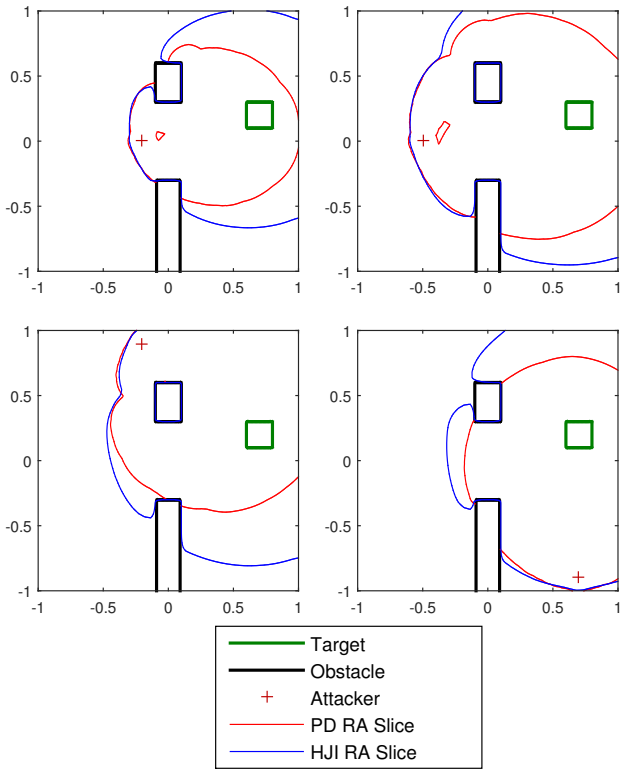


Fig. 14: Reach-avoid slices computed using the HJI approach and the path defense approach. Defenders and attackers have the same maximum speed.

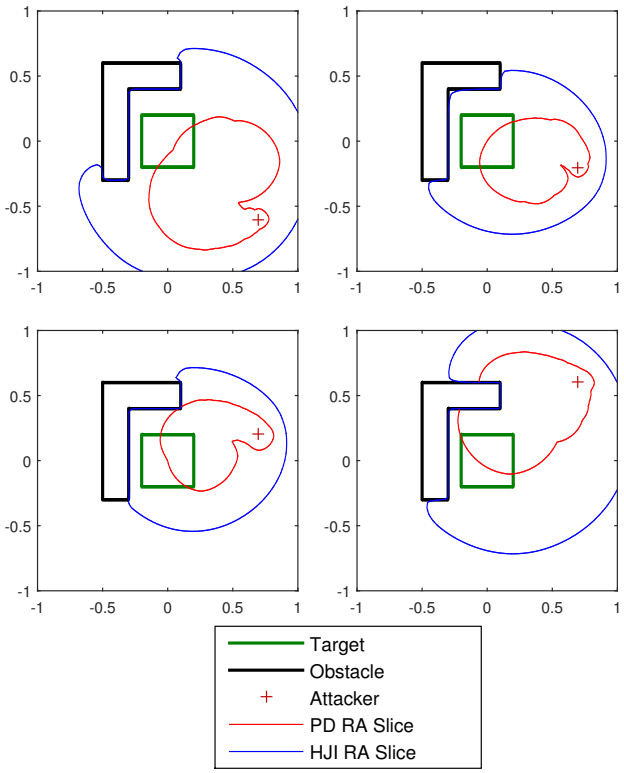


Fig. 15: Reach-avoid slices computed using the HJI approach and the path defense approach. Attackers' maximum speed is 80% of that of the defenders.

becomes more conservative when there are large obstacles in the domain because we only considered paths that touch the target, which introduces additional conservatism in a non-simply-connected free space, as discussed in IV-C. Figure 15 compares the 2D slices computed by the path defense approach and by the HJI approach in another 4 vs. 4 game where the attackers' maximum speed is 80% of the defenders' maximum speed (players on each team have the same maximum speed), and where there is a larger obstacle in the domain. In this case, the defender winning regions computed using path defense approach covers approximately 34% of the area of the defender winning regions computed using the HJI approach.

D. Real-Time Maximum Matching Updates

After determining all pairwise outcomes either by N^2 HJI PDEs in general or by solving a single 4D HJI PDE when all players on each team have the same maximum speed, pairwise outcomes of *any* joint state of the attacker-defender pair are characterized. Thus, the bipartite graph corresponding to the pairwise outcomes can be updated simply by checking whether the joint state of the attacker-defender pair is inside the appropriate 4D reach-avoid set. This allows for updates of the bipartite graph and its maximum matching as the players play out the game in real time.

Figure 16 shows the maximum matching at several time snapshots of a 4 vs. 4 game. Each defender that is part of a maximum matching plays optimally against the paired-up attacker according to Equation (13), and the remaining defender plays optimally against the closest attacker also according to Equation (13). The attackers' strategy is to move towards the target along the shortest path while steering clear of the obstacles by 0.125 units. The maximum matching is updated every $\Delta = 0.005$ seconds. At $t = 0$ and $t = 0.2$, the maximum matching is of size 3, which guarantees that at most one attacker will be able to reach the target. After $t = 0.4$, a perfect matching is found, which guarantees that no attacker will be able to reach the target.

VIII. Conclusion and Future Work

A general multiplayer reach-avoid game is numerically intractable to analyze by directly solving the corresponding high dimensional HJI PDE. To address this, we presented a way to tie together pairwise outcomes using maximum matching to approximate the solution to the full multiplayer game, guaranteeing an upper bound on the number of attackers that can reach the target. We also presented two approaches for determining the pairwise outcomes. The HJI approach is computationally more expensive compared to the path defense approach, produces the optimal closed-loop control strategy for each attacker-defender pair, and efficiently allows for real time maximum updates. The path defense approach is conservative towards the defender, performs computation on the state space of a single player as opposed to the joint state space, and scales only linearly in the number of players in the general case where each player has a different maximum speed.

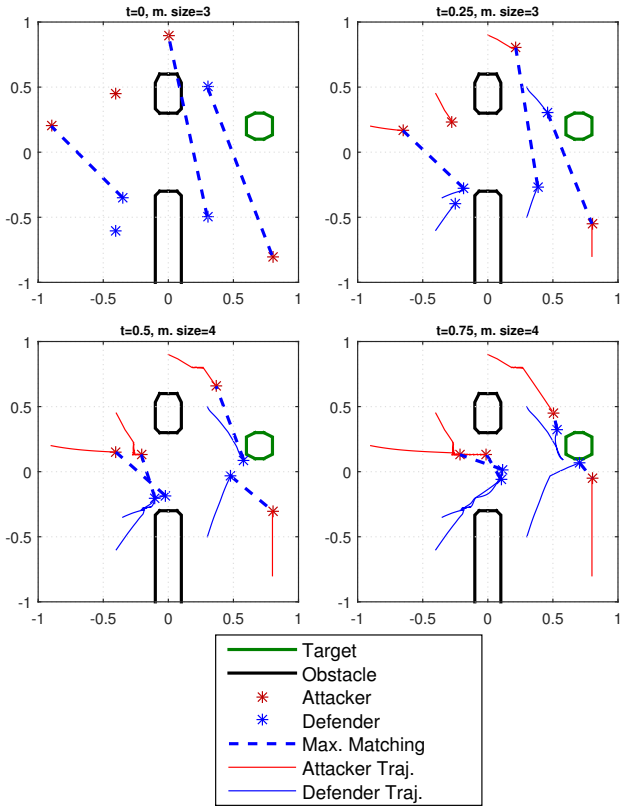


Fig. 16: An illustration of how the size of maximum matching can increase over time. Throughout the game, the defenders are updating the bipartite graph and maximum matching via the procedure described in Section V-B. Because the attacking team is not playing optimally, the defending team is able to find a perfect matching after $t = 0.4$ (bottom plots) and prevent *all* attackers from reaching the target.

The methods presented in this paper illustrate the power of using special properties of shortest paths and tying together low dimensional information to approximate a high dimensional problem. Our future work will focus on applying these themes to both similar problems such as a reach-avoid game where players have more complex dynamics, and other related problems such as cooperative multi-agent collision avoidance.

REFERENCES

[1] H. Huang, "Reachability-based control for human and autonomous agents in adversarial games," Ph.D. dissertation, Stanford University, 2012.

[2] H. Huang, J. Ding, W. Zhang, and C. Tomlin, "A differential game approach to planning in adversarial scenarios: A case study on capture-the-flag," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 1451–1456.

[3] M. Earl and R. D'Andrea, "A decomposition approach to multi-vehicle cooperative control," *Robotics and Autonomous Systems*, vol. 55, no. 4, pp. 276–291, 2007.

[4] M. Campbell, R. D'Andrea, D. Schneider, A. Chaudhry, S. Waydo, J. Sullivan, J. Veverka, and A. Klochko, "Roboflag games using systems based, hierarchical control," *Proceedings of the American Control Conference*, vol. 1, pp. 661–666, 2003.

[5] S. Waydo and R. Murray, "Vehicle motion planning using stream functions," *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA'03*, pp. 2484–2491, 2003.

[6] R. Parasuraman, S. Galster, P. Squire, H. Furukawa, and C. Miller, "A flexible delegation-type interface enhances system performance in human supervision of multiple robots: Empirical studies with roboflag," *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, vol. 35, no. 4, p. 481, 2005.

[7] Department of the Air Force, "United States Air Force unmanned aircraft systems flight plan 2009-2047," *oai.dtic.mil*, Jan 2009.

[8] H. Erzberger, "Automated conflict resolution for air traffic control," *25 th International Congress of the Aeronautical Sciences*, Jan 2006.

[9] A. Madrigal, "Autonomous robots invade retail warehouses," http://www.wired.com/wiredscience/2009/01/retailrobots/.

[10] G. Chasparis and J. Shamma, "Linear-programming-based multi-vehicle path planning with adversaries," *Proceedings of the American Control Conference*, pp. 1072–1077, 2005.

[11] J. Sprinkle, J. Eklund, H. Kim, and S. Sastry, "Encoding aerial pursuit/evasion games with fixed wing aircraft into a nonlinear model predictive tracking controller," *IEEE Conference on Decision and Control*, 2004.

[12] J. McGrew, J. How, L. Bush, B. Williams, and N. Roy, "Air combat strategy using approximate dynamic programming," *AIAA Guidance, Navigation, and Control Conference*, Aug 2008.

[13] R. Isaacs, *Differential Games*. New York: Wiley, 1967.

[14] I. Mitchell, A. Bayen, and C. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 947–957, July 2005.

[15] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996. [Online]. Available: http://www.pnas.org/content/93/4/1591.abstract

[16] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002, ISBN: 978-0-387-95482-0.

[17] J. Ding, J. Sprinkle, S. S. Sastry, and C. J. Tomlin, "Reachability calculations for automated aerial refueling," in *IEEE Conference on Decision and Control*, Cancun, Mexico, 2008.

[18] Z. Zhou, R. Takei, H. Huang, and C. Tomlin, "A general, open-loop formulation for reach-avoid games," in *IEEE Conference on Decision and Control*, 2012, pp. 6501–6506.

[19] A. Schrijver, *Combinatorial Optimization : Polyhedra and Efficiency (Algorithms and Combinatorics)*. Springer, Jul. 2004. [Online]. Available: http://www.worldcat.org/isbn/3540204563

[20] M. Karpinski and W. Rytter, *Fast parallel algorithms for graph matching problems*. New York, NY, USA: Oxford University Press, Inc., 1998.

[21] I. Mitchell, *A Toolbox of Level Set Methods*, 2009, http://people.cs.ubc.ca/ mitchell/ToolboxLS/index.html.

[22] T. Basar and G. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed. Philadelphia, PA: SIAM, 1999.

[23] M. G. Crandall and P.-L. Lions, "Viscosity solutions of Hamilton-Jacobi equations," *Transactions of the American Mathematical Society*, vol. 277, no. 1, pp. 1–42, 1983.

[24] I. Mitchell, "Application of level set methods to control and reachability problems in continuous and hybrid systems," Ph.D. dissertation, Stanford University, 2002.

[25] O. Bokanowski, N. Forcadel, and H. Zidani, "Reachability and minimal times for state constrained nonlinear problems without any controllability assumption," *SIAM Journal on Control and ...*, pp. 1–24, 2010.

[26] J. Lygeros, C. Tomlin, and S. Sastry, "Controllers for reachability specifications for hybrid systems," *Automatica*, vol. 35, no. 3, pp. 349 – 370, 1999.

[27] C. Tomlin, J. Lygeros, and S. Shankar Sastry, "A game theoretic approach to controller design for hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 949 –970, Jul 2000.