# 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 an attacking team with $N_A$ attackers and a defending team with $N_D$ defenders playing on a compact domain with obstacles. The attacking team aims to send $M$ of the $N_A$ attackers to some target location, while the defending team aims to prevent that by capturing attackers or indefinitely delaying attackers from reaching the target. Although the analysis of this game plays an important role in many applications, the optimal solution to this game is computationally intractable when $N_A > 1$ or $N_D > 1$. In this paper, we present two approaches for the $N_A = N_D = 1$ case to determine pairwise outcomes, and a graph theoretic maximum matching approach to merge these pairwise outcomes for an $N_A, N_D > 1$ 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, and that the two-dimensional "path defense" approach is considerably more scalable with the number of players while maintaining defender performance guarantees.**

***Index Terms*—Author, please supply index terms/keywords for your paper. To download the IEEE Taxonomy go to http://www.ieee.org/documents/taxonomy_v101.pdf.**

AQ1

## I. Introduction

Multiplayer reach-avoid games are differential games between two adversarial teams of cooperative players playing on a compact domain with obstacles. The "attacking team" aims to send as many team members, called "attackers," to some target set as quickly as possible. 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 also powerful theoretical tools for analyzing realistic situations in robotics, aircraft control, security, and other domains [3]–[5].

The multiplayer reach-avoid game is difficult to analyze because the two teams have conflicting and asymmetric goals, while complex cooperation within each team may exist. In addition, optimal solutions are impossible to compute using traditional dynamic programming approaches due to the intrinsic high dimensionality of the joint state space. Previously, in [6], where a team of defenders assumes that the attackers move toward their target in straight lines, a mixed-integer linear programming approach was used. [7] assumes that the attackers use a linear feedback control law, and a mixed integer linear program was then relaxed into linear program. In complex pursuit-evasion games where players may change roles over time, a nonlinear model-predictive control [8] approach has been investigated. Approximate dynamic programming [9] has also been used to analyze reach-avoid games.

Although the above techniques provide some useful insight, they only work well when strong assumptions are made or when accurate models of the opposing team can be obtained. To solve general reach-avoid games, the Hamilton–Jacobi–Isaacs (HJI) approach [10] is ideal when the game is low-dimensional. The approach involves solving an HJI partial differential equation (PDE) in the joint state space of the players to compute a reach-avoid set, which partitions the players' joint state space into a winning region for the defending team and one for the attacking team. The optimal strategies can then be extracted from the gradient of the solution. This approach is particularly useful because of the numerical tools [11]–[13] available, and has been able to solve several practical problems [2], [11], [14]. The HJI approach 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 its complexity scales exponentially with the number of players, making the approach only tractable for the two-player game. Thus, complexity-optimality trade-offs must be made.

For the two-player reach-avoid game, we first present the two-player HJI solution [2], which computes a 4-D reach-avoid set that determines which player wins the game assuming both players use the closed-loop optimal control strategy. Next, we propose the "path defense" approximation to the HJI solution, in which the defenders utilize a "semi-open-loop" control strategy. Here, we approximate 2-D slices of the reach-avoid sets by solving 2-D Eikonal equations, and provide guarantees for the defending team's performance.

For the multiplayer reach-avoid game, we propose to merge the $N_A N_D$ pairwise outcomes using the graph theoretic maximum matching, which can be efficiently computed by known algorithms [15], [16]. The maximum matching process incorporates cooperation among defenders without introducing significant additional computation cost. When players on each team have identical dynamics, only a *single* HJI PDE needs to be solved to characterize *all* pairwise outcomes. Furthermore, when applying maximum matching to the two-player path defense solution, the computational complexity scales linearly with the number of attackers, as opposed to quadratically with the total number of players in the HJI approach.

## II. Reach-Avoid Problem

### A. Multiplayer Reach-Avoid Game

Consider $N_A + N_D$ players partitioned into the set of $N_A$ attackers, $\{P_{A_i}\}_{i=1}^{N_A} = \{P_{A_1}, P_{A_2}, \ldots, P_{A_{N_A}}\}$ and the set of $N_D$ defenders, $\{P_{D_i}\}_{i=1}^{N_D} = \{P_{D_1}, \ldots, P_{D_{N_D}}\}$, whose states are confined in a bounded, open domain $\Omega \subset \mathbb{R}^2$. The domain $\Omega$ is partitioned into $\Omega = \Omega_{\text{free}} \cup \Omega_{\text{obs}}$, where $\Omega_{\text{free}}$ is a compact set representing the free space in which the players can move, while $\Omega_{\text{obs}} = \Omega \setminus \Omega_{\text{free}}$ corresponds to obstacles in the domain.

Manuscript received September 21, 2015; revised April 29, 2016 and May 1, 2016; accepted May 27, 2016. This work was supported in part by the National Science Foundation under CPS:ActionWebs (CNS-931843), by the Office of Naval Research under the HUNT (N0014-08-0696) and SMARTS (N00014-09-1-1051) MURIs and by Grant N00014-12-1-0609, and by the Air Force Office for Scientific Research under the CHASE MURI (FA9550-10-1-0567). Recommended by Associate Editor X. XXXX.

AQ2

M. Chen and C. J. Tomlin are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720 USA (e-mail: mochen72@eecs.berkeley.edu; tomlin@eecs.berkeley.edu)).

Z. Zhou is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: zyzhou@stanford.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2016.2577619

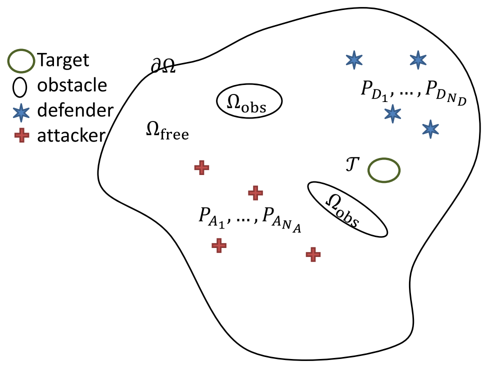


Fig. 1. Components of a multiplayer reach-avoid game.

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 $x^0_{A_i} \in \Omega_{\text{free}}, i = 1, 2, \ldots, N_A, x^0_{D_i} \in \Omega_{\text{free}}, i = 1, 2, \ldots, N_D$, we assume the dynamics of the players to be defined by the following decoupled system for $t \geq 0$:

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

where $v_{A_i}$, $v_{D_i}$ denote maximum speeds for $P_{A_i}$ and $P_{D_i}$ respectively, and $a_i$, $d_i$ denote controls of $P_{A_i}$ and $P_{D_i}$ respectively. We assume that $a_i$, $d_i$ are drawn from the set $\Sigma = \{\sigma : [0, \infty) \to \overline{B}_2 | \sigma \text{ is measurable}\}$, where $\overline{B}_2$ denotes the closed unit disk in $\mathbb{R}^2$. We also constrain the players to remain within $\Omega_{\text{free}}$ for all time. Denote the joint state of all players by $\mathbf{x} = (\mathbf{x}_A, \mathbf{x}_D)$ where $\mathbf{x}_A = (x_{A_1}, \ldots x_{A_{N_A}})$ is the attacker joint state $\{P_{A_i}\}_{i=1}^{N_A}$, and $\mathbf{x}_D = (x_{D_1}, \ldots, x_{D_{N_D}})$ is the defender joint state $\{P_{D_i}\}_{i=1}^{N_D}$.

The attacking team wins whenever $M$ of the $N_A$ attackers reach some target set without being captured by the defenders; $M$ is pre-specified with $0 < M \leq N_A$. 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 $N_A - M + 1$ attackers from reaching $\mathcal{T}$. An illustration of the game setup is shown in Fig. 1.

Let $\mathcal{C}_{ij} = \{\mathbf{x} \in \Omega^{N_A + N_D} | \|x_{A_i} - x_{D_j}\|_2 \leq R_C\}$ denote the capture set. $P_{A_i}$ 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 problems:

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

### B. Two-Player Reach-Avoid Game

We will answer the above questions about the $N_A$ versus $N_D$ reach-avoid game by using the solution to the two-player 1 versus 1 game as a building block. In the two-player game, we denote the attacker $P_A$, the defender $P_D$, their states $x_A$, $x_D$, and their initial conditions $x^0_A, x^0_D$. Their dynamics are

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

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

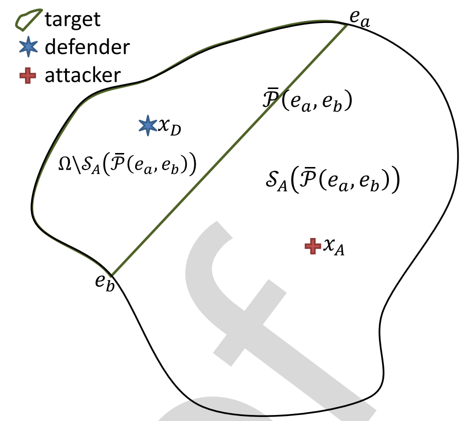


Fig. 2. Components of a path defense game.

$P_A$ wins if it reaches the target $\mathcal{T}$ without being captured by $P_D$. $P_D$ wins if it can prevent $P_A$ from winning by capturing $P_A$ or indefinitely delaying $P_A$ from reaching $\mathcal{T}$. For the two-player reach-avoid game, we seek to answer the following:

1) Given $\mathbf{x}^0$ and $\mathcal{T}$, is the defender guaranteed to win?
2) More generally, given $x_A$ and $\mathcal{T}$, what is the set of initial positions from which the defender is guaranteed to win?

## III. HJI Solution of the 1 Versus 1 Game

The HJI approach for solving differential games is outlined in [2], [11], and [17]. The optimal joint closed-loop control strategies for the attacker and the defender in a two-player reach-avoid game can be obtained by solving a 4-D HJI PDE. This solution allows us to determine whether the defender will win against the attacker in a 1 versus 1 setting.

In the two-player game, the attacker aims to reach $\mathcal{T}$ while avoiding $\mathcal{C}$. Both players also avoid $\Omega_{\text{obs}}$. In particular, the defender wins if the attacker is in $\Omega_{\text{obs}}$, and vice versa. Therefore, we define the terminal set and avoid set to be

$$\begin{aligned} R &= \{\mathbf{x} \in \Omega^2 | x_A \in \mathcal{T}\} \cup \{\mathbf{x} \in \Omega^2 | x_D \in \Omega_{\text{obs}}\} \\ A &= \mathcal{C} \cup \{\mathbf{x} \in \Omega^2 | x_A \in \Omega_{\text{obs}}\}. \end{aligned} \tag{3}$$

Given (3), we can define the corresponding implicit surface functions $\phi_R, \phi_A$ required for solving the HJI PDE. Since $\Omega \subset \mathbb{R}^2$, the result is $\mathcal{RA}_\infty(R, A) \in \mathbb{R}^4$, a 4-D reach-avoid set. If $\mathbf{x}^0 \in \mathcal{RA}_\infty(R, A)$, then the attacker is guaranteed to win the game by using the optimal control *even if* the defender is also using the optimal control; if $\mathbf{x}^0 \notin \mathcal{RA}_\infty(R, A)$, then the defender is guaranteed to win the game by using the optimal control *even if* the attacker is also using the optimal control.

## IV. Path Defense Solution to the 1 Versus 1 Game

We approximate 2-D slices of the 4-D reach-avoid set (or simply "2-D slices") in the path defense approach. Each slice will be taken at an attacker position. Here, we will assume that the defender is not slower than the attacker: $v_A \leq v_D$.

### A. 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 (Fig. 2). We

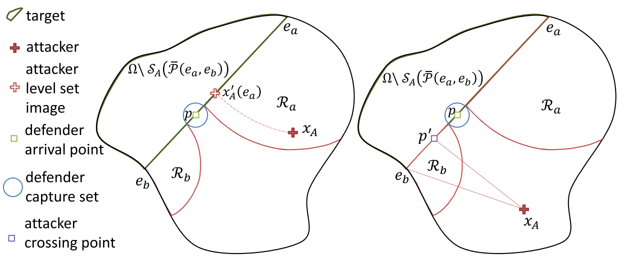


Fig. 3. (Left) If the attacker is in $\mathcal{R}_a \cup \mathcal{R}_b$ and moves toward $e_a$, then the attacker will be able to reach $e_a$ without being captured. (Right) If the attacker is not in $\mathcal{R}_a \cup \mathcal{R}_b$, there is no point on the path $p\prime \in \bar{\mathcal{P}}(e_a, e_b)$ that the can be reached without being captured.

denote the target set as $\mathcal{T} = \Omega \setminus \mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$ for two given points on the boundary $e_a$, $e_b$. $\mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$ 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 called the **anchor points** of path $\bar{\mathcal{P}}(e_a, e_b)$.

Denote the shortest path between **any** two points $x, y \in \Omega_{\text{free}}$ to be $\mathcal{P}(x, y)$, with length $\text{dist}\,(x, y)$, and requiring the attacker and defender durations of $t_A(x, y)$, $t_D(x, y)$ to traverse, respectively. We will also use $\text{dist}\,(\cdot, \cdot)$ with one or both arguments being sets in $\Omega$ to denote the shortest distance between the arguments.

*Definition 2—Attacker's Side of the Path:* A path of defense $\bar{\mathcal{P}}(e_a, e_b)$ partitions the domain $\Omega$ into two regions. Define $\mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$ 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 $\mathcal{T} = \Omega \setminus \mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$.

## B. Solving the Path Defense Game

A path defense game can be directly solved by computing a 4-D reach-avoid set. Since the direct solution is time- and memory-intensive, we propose an efficient approximation of 2-D slices that is conservative toward the defender.

*Definition 3—Defendable Path:* Given $\mathbf{x}^0 = (x_A^0, x_D^0)$, 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:* $\bar{\mathcal{P}}(e_a, e_b)$ 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 $\bar{\mathcal{P}}(e_a, e_b)$.

Checking whether a path $\bar{\mathcal{P}}(e_a, e_b)$ is defendable involves a 4-D reach-avoid set calculation, so instead we check 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 2-D distance calculations; the definitions and lemma are illustrated in Fig. 3.

*Definition 5—Attacker Level Set Image :* Given attacker position $x_A(t)$, define the attacker level set image with respect to anchor point $e_a$ to be $x'_A(t; e_a) = \{x \in \bar{\mathcal{P}}(e_a, e_b) : t_A(x, e_a) = t_A(x_A(t), e_a)\}$. $x'_A$ is the unique point on $\bar{\mathcal{P}}(e_a, e_b)$ such that $t_A(x'_A, e_a) = t_A(x_A, e_a)$. Define $x'_A(t; e_b)$ similarly by replacing $e_a$ with $e_b$. For convenience, we sometimes omit the time argument and write $x'_A(e_a)$.

*Proposition 1:* $\text{dist}\,(x'_A(e_b), e_a) \le \text{dist}\,(x'_A(e_a), e_a)$.

*Proof:* First note that

$$\begin{aligned}\text{dist}\,(e_a, e_b) &\le \text{dist}\,(x_A, e_a) + \text{dist}\,(x_A, e_b)\\ &= \text{dist}\,(x'_A(e_a), e_a) + \text{dist}\,(x'_A(e_b), e_b)\,.\end{aligned}$$

Then, since the left-hand side is given by $\text{dist}\,(e_a, e_b) = \text{dist}\,(e_a, x'_A(e_b)) + \text{dist}\,(x'_A(e_b), e_b)$, the result follows. ∎

*Definition 6—Capture Set:* Define the capture set to be $D_C(y, t) = \{x \| \|x - y(t)\|_2 \le R_C\}$. We will drop the second argument of $D_C$ when $y$ does not depend on time.

*Remark 1:* Given $\bar{\mathcal{P}}(e_a, e_b)$, suppose the attacker level set image is within defender's capture set at some time $s$

$$x'_A(s; e_a) \in D_C(x_D, s) \text{ (or } x'_A(s; e_b) \in D_C(x_D, s))\,.$$

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

$$\begin{gathered}x'_A(t; e_a) \in D_C(x_D, t) \forall\, t \ge s\\ (\text{or } x'_A(t; e_b) \in D_C(x_D, t) \forall\, t \ge s)\,.\end{gathered}$$

This is because the attacker level set image can move at most as fast as the attacker, who is not faster than the defender.

*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 with point $p$ and anchor point $e_a$ as follows:

$$\mathcal{R}_a(p) = \{x : \text{dist}\,(x, e_a) \le \text{dist}\,(D_C(p), e_a)\}\,. \tag{4}$$

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

*Lemma 1:* Suppose $x_D^0 = p \in \bar{\mathcal{P}}(e_a, e_b)$ and $v_A = v_D$. Then, $\bar{\mathcal{P}}(e_a, e_b)$ is strongly defendable if and only if $x_A^0$ is outside the region induced by $p$: $x_A^0 \in \Omega \backslash (\mathcal{R}_a \cup \mathcal{R}_b)$.

*Proof:* See Fig. 3. Assume $x_A^0 \notin \mathcal{T} = \Omega \setminus \mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$; 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 $\Omega \setminus \mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$ without being captured. Without loss of generality (WLOG), suppose $x_A^0 \in \mathcal{R}_a$. To capture the attacker, the defender's capture set must contain $x'_A(e_a)$ or $x'_A(e_b)$ at some time $t$. By Definition 7, we have $\text{dist}\,(x_A^0, e_a) < \text{dist}\,(D_C(p), e_a)$, so $t_A(x'_A(e_a), e_a) < t_D(D_C(p), e_a)$. By Proposition 1, $\text{dist}\,(x'_A(e_b), e_a) \le \text{dist}\,(x'_A(e_a), e_a)$, so it suffices to show that the defender's capture set cannot reach $x'_A(e_a)$ before the attacker reaches $e_a$.

If the attacker moves toward $e_a$ along $\mathcal{P}(x_A^0, e_a)$ with maximum speed, then $x'_A(e_a)$ will move toward $e_a$ along $\mathcal{P}(x'_A(e_a), e_a)$ at the same speed. Since $t_A(x_A, e_a) = t_A(x'_A(e_a), e_a) < t_D(D_C(p), e_a)$, $x_A$ will reach $e_a$ before the defender capture set $D_C(x_D, t)$ does.

Next we show, by contradiction, that if $x_A \notin \mathcal{R}_a \cup \mathcal{R}_b$, then the attacker cannot reach $\Omega \setminus \mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$ without being captured. Suppose $P_A$ will reach some point $p\prime$ before $D_C(P_D, t)$ does,i.e., $\text{dist}\,(x_A^0, p\prime) < \text{dist}\,(D_C(x_D^0), p\prime) = \text{dist}\,(D_C(p), p\prime)$. WLOG, assume $p\prime \in \mathcal{P}(p, e_b)$, and note that $\text{dist}\,(D_C(p), e_b) < \text{dist}\,(x_A^0, e_b)$, since the attacker is not in $\mathcal{R}_b$. Starting with the definition of the shortest path, we have

$$\begin{aligned}\text{dist}\,(x_A^0, e_b) &\le \text{dist}\,(x_A^0, p\prime) + \text{dist}\,(p\prime, e_b)\\ &< \text{dist}\,(D_C(p), p\prime) + \text{dist}\,(p\prime, e_b)\\ &= \text{dist}\,(D_C(p), e_b)\\ \text{dist}\,(x_A^0, e_b) &< \text{dist}\,(x_A^0, e_b) \text{ (since } x_A^0 \notin \mathcal{R}_a).\end{aligned} \tag{5}$$

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

If $v_A < v_D$, $P_A$ being outside of $\mathcal{R}_a \cup \mathcal{R}_b$ becomes a sufficient condition 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, if the defender can arrive at $p$ before the attacker moves into $\mathcal{R}_a(p) \cup \mathcal{R}_b(p)$, then $\bar{\mathcal{P}}(e_a, e_b)$ 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 by checking whether there exists some $p \in \bar{\mathcal{P}}(e_a, e_b)$ such that $t_D(x_D^0, p) \leq t_A(x_A^0, \mathcal{R}_a(p) \cup \mathcal{R}_b(p))$. If so, then $\bar{\mathcal{P}}(e_a, e_b)$ is strongly defendable. The next lemma shows that it is necessary and sufficient to check whether *one* special point, $p^* \in \bar{\mathcal{P}}(e_a, e_b)$, can be the first arrival point for strongly defending $\bar{\mathcal{P}}(e_a, e_b)$.

*Remark 2:* Given $p \in \bar{\mathcal{P}}(e_a, e_b)$, $\text{dist}\,(x_A^0, \mathcal{R}_a(p)) = \text{dist}\,(x_A^0, e_a) - \text{dist}\,(D_C(p), e_a)$. Similarly, $\text{dist}\,(x_A^0, \mathcal{R}_b(p)) = \text{dist}\,(x_A^0, e_b) - \text{dist}(D_C(p), e_b)$.

*Lemma 2:* Define $p^* \in \bar{\mathcal{P}}(e_a, e_b)$ 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 by definition.

We now show the other direction's 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 show that if choosing $p^*$ as the first entry point does not allow the defender to defend $\bar{\mathcal{P}}(e_a, e_b)$, then no other entry point does.

Suppose that the defender cannot defend $\bar{\mathcal{P}}(e_a, e_b)$ by choosing $p^*$ as the first entry point, but can defend $\bar{\mathcal{P}}(e_a, e_b)$ by choosing another entry point $p\prime$. WLOG, assume $\text{dist}\,(D_C(p^*), e_a) > \text{dist}\,(D_C(p'), e_a)$. 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 2, we have

$$\begin{aligned} \text{dist}\left(x_A^0, \mathcal{R}_a(p^*)\right) &= \text{dist}\left(x_A^0, e_a\right) - \text{dist}\left(D_C(p^*), e_a\right) \\ t_A\left(x_A^0, \mathcal{R}_a(p^*)\right) &= t_A\left(x_A^0, e_a\right) - t_A\left(D_C(p^*), e_a\right). \end{aligned} \tag{6}$$

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

$$\begin{aligned} \text{dist}\left(x_A^0, \mathcal{R}_a(p\prime)\right) &= \text{dist}\left(x_A^0, e_a\right) - \text{dist}\left(D_C(p\prime), e_a\right) \\ t_A\left(x_A^0, \mathcal{R}_a(p\prime)\right) &= t_A\left(x_A^0, e_a\right) - t_A\left(D_C(p\prime), e_a\right). \end{aligned} \tag{7}$$

Then, subtracting the above two equations, we see that the attacker can get to $\mathcal{R}_a$ sooner by the following amount:

$$\begin{aligned} & t_A\left(x_A^0, \mathcal{R}_a(p^*)\right) - t_A\left(x_A^0, \mathcal{R}_a(p\prime)\right) \\ &= t_A\left(D_C(p\prime), e_a\right) - t_A\left(D_C(p^*), e_a\right) \\ &= t_A\left(p\prime, p^*\right) \geq t_D(p\prime, p^*). \end{aligned} \tag{8}$$

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 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 defender "loses time" by going to $p\prime$). By the triangle inequality, we have

$$\begin{aligned} \text{dist}\left(x_D^0, p^*\right) &\leq \text{dist}\left(x_D^0, p\prime\right) + \text{dist}\left(p\prime, p^*\right) \\ \text{dist}\left(x_D^0, p^*\right) - \text{dist}\left(x_D^0, p\prime\right) &\leq \text{dist}\left(p\prime, p^*\right) \\ t_D\left(x_D^0, p^*\right) - t_D\left(x_D^0, p'\right) &\leq t_D(p\prime, p^*). \end{aligned} \tag{9}$$

■

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 $x_D^0 \in \mathcal{D}(e_a, e_b; x_A^0) = \{x : t_D(x, p^*) \leq t_A(x_A^0, \mathcal{R}_a \cup \mathcal{R}_b)\}$.

The computations in this algorithm can be efficiently done by solving a series of 2-D Eikonal equations by using a fast marching level set method (FMM) [12], reducing our 4-D problem to 2-D. Fig. 4 illustrates the proof of Lemma 2 and the defender winning region $\mathcal{D}$.

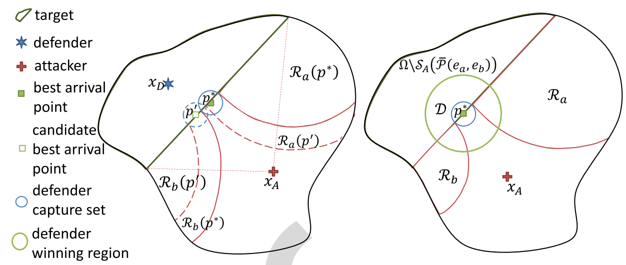


Fig. 4. (Left) If the defender cannot defend $\bar{\mathcal{P}}(e_a, e_b)$ by first going to $p^*$, then the attacker cannot defend $\bar{\mathcal{P}}(e_a, e_b)$ by going to any other point $p$. (Right) Defender winning region $\mathcal{D}$.

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

The central idea in using path defense is that if the target set is enclosed by some strongly defendable path for some $e_a, e_b$, then the defender can win the game using the semi-open-loop strategy outlined in this section, *even if* the attacker uses the optimal control. Checking for strongly defendable paths adds more conservatism toward 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. However, we can reduce the number of paths that needs to be checked by only checking 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.

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 $p^* \in \bar{\mathcal{P}}(e_a, e_b)$ (an open-loop strategy), then move toward $x_A'(e_a)$ or $x_A'(e_b)$ until the level set image is captured (a closed-loop strategy). Finally, the defender can simply track the captured level set image (a closed-loop strategy). This is a "semi-open-loop" strategy. The following algorithm approximates a 2-D slice conservatively toward 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 $e_a \in \partial\Omega$.
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 2-D slice, representing the conservative winning region for the defender in the two-player reach-avoid game.

## V. From Two-Player to Multiplayer

### A. Maximum Matching

We piece together the outcomes of all attacker–defender pairs using maximum matching as follows:

*Algorithm 2*:

1) Construct a bipartite graph with two sets of nodes $\{P_{A_i}\}_{i=1}^{N_A}$, $\{P_{D_i}\}_{i=1}^{N_D}$. Each node represents a player.
2) For all $i, j$, 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 (e.g., [15], [16]) to find a maximum matching in the graph.

After finding a maximum matching, we can guarantee an upper bound on the number of attackers that is be able to reach the target. If the maximum matching is of size $m$, then the defending team would be

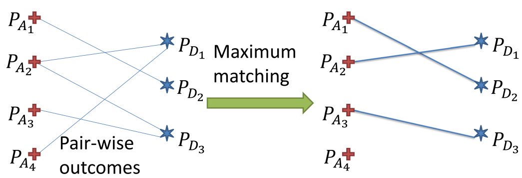


Fig. 5. Illustration of using maximum matching to conservatively approximate the multiplayer reach-avoid game.

able to prevent *at least* $m$ attackers from reaching the target, and thus $N_A - m$ is an upper bound on the number of attackers that can reach the target. The maximum matching approach is illustrated in Fig. 5.

### B. Time-Varying Defender-Attacker Pairings

With the next algorithm, the bipartite graph and its corresponding maximum matching can be updated, potentially in real time, as the players change positions during the game:

*Algorithm 3*:

1) Given each $x_{D_i}$ and each $x_{A_j}$, determine whether $P_{A_j}$ can win against $P_{D_i}$ for all $i, j$.
2) Assign a defender to each attacker that is part of a maximum matching.
3) For a short duration $\Delta$, 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 $\Delta$ and repeat steps 1 to 3 with the new player positions.

As $\Delta \to 0$, the above procedure continuously computes a bipartite graph and its maximum matching. As long as each defender uses a winning control input against the paired-up attacker, the size of maximum matching can never decrease.

### C. Application to the Two-Player HJI Solution

In general, solving $N_A N_D$ 4-D HJI PDEs gives us the pairwise outcomes between every attacker–defender pair. The computation time required is thus $CN_A N_D$, where $C$ is the time required to solve a single 4-D HJI PDE. The pairwise outcomes can then be merged together to approximate the $N_A$ versus $N_D$ game. In the case where each team has a single maximum speed, solving *one* 4-D HJI PDE would characterize all pairwise outcomes.

Since the solution to the 4-D HJI PDE characterizes pairwise outcomes based on any attacker–defender joint-state, it allows for real-time updates of the maximum matching. As players move to new positions, the pairwise outcome can be updated 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 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 step 3, the speeds of defenders come in only through a single distance calculation from $p^*$, which only needs to be done once per attacker position. Therefore, the total computation time required is on the order of $C_1 + C_2 N_A$, where $C_1$ is the time required for steps 1 and 2, $C_2$ is the time required for steps 3 to 5.

### E. Defender Cooperation

One of the strengths of the maximum matching approach is its simplicity in the way cooperation among the defenders is incorporated from pairwise outcomes. More specifically, cooperation is incorporated using the knowledge of the strategy of each teammate, and the knowledge of which attackers each teammate can win against in a 1 versus 1 setting.

The knowledge of the strategy of each teammate is incorporated in the following way. When the pairwise outcomes for each defender is computed, a particular defender strategy used. The strategy of each defender is then used to compute pairwise outcomes, which are used in the maximum matching process. Each defender may use the optimal closed-loop strategy given by the two-player HJI solution, the semi-open-loop strategy given by the two-player path defense solution, or even another strategy that is not described in this paper. In fact, different defenders may use a different strategy.

As already mentioned, all of the information about the strategy of each defender is used to compute the pairwise outcomes. Since each pairwise outcome specifies a winning region for the corresponding defender, each defender can be guaranteed to win against a set of attackers in a 1 versus 1 setting. The set of attackers against which each defender can win is then used to construct the bipartite graph on which maximum matching is performed. While executing the joint defense strategy as a team, each defender simply needs to execute its *pairwise* defense strategy against the attacker to which the defender is assigned.

The maximum matching process optimally combines the information about teammates' strategies and competence to derive a joint strategy to prevent as many attackers from reaching the target as possible. The size of the maximum matching then guarantees an upper bound on the number of attackers that can reach the target. To our knowledge, no other method can synthesize a joint defender control strategy that can provide such a guarantee in a multiplayer game.

## VI. Numerical Results

We use a 4 versus 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. Computations were done on a laptop with a Core i7-2640M processor with 4 GB of memory.

### A. HJI Formulation

Fig. 6 shows the results of solving the corresponding 4-D HJI PDE in blue. Computing the 4-D reach-avoid set on a grid with 45 points in each dimension took approximately 30 min. All players have the same maximum speed, so only a single 4-D HJI PDE needed to be solved. To visualize the 4-D reach-avoid set, we take 2-D slices of the 4-D reach-avoid set sliced at each attacker's position.

Fig. 7 shows the bipartite graph and maximum matching obtained from the pairwise outcomes. The maximum matching is of size 4. This guarantees that if each defender plays optimally against the attacker matched by the maximum matching, then *no* attacker will be able to reach the target.

### B. Path Defense Formulation

Fig. 6 shows, in red, the results of using the path defense approach to compute conservative approximations of the 4-D reach-avoid set sliced at various attacker positions.

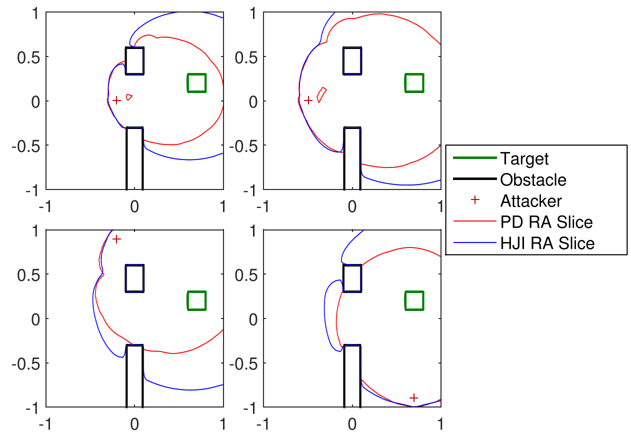


Fig. 6. Reach-avoid slices computed using the HJI approach and the path defense approach.

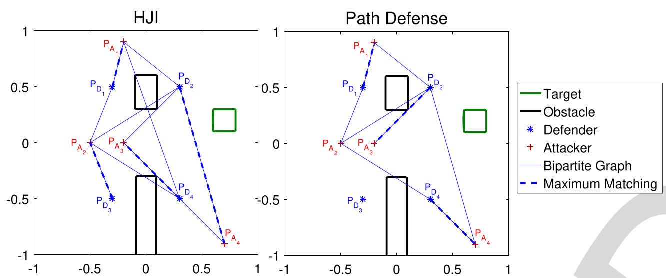


Fig. 7. Merging pairwise outcomes with maximum matching.

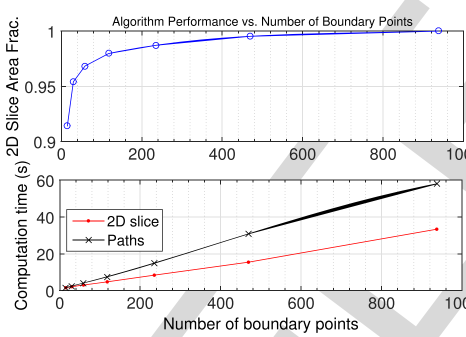


Fig. 8. Performance of the Path Defense Solution.

Fig. 7 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 $\times$ 200 grid, and 937 paths were used to compute the results in Fig. 6. Computation time varies with the number of paths we chose in steps 1 and 2 in Algorithm 1. Taking the union of the defender winning regions from more paths will give a less conservative result, but requires more computation time. A summary of the performance is shown in Fig. 8. With 937 paths, the computation of paths took approximately 60 s, and the computation of the 2-D slice given the set of paths took approximately 30 s. However, very few paths are needed to approximate a 2-D slice: Even with as few as 30 paths, the computed 2-D slice covers more than 95% of the area of the 2-D slice computed using 937 paths. This reduces the computation time of the paths to 2.5 s, and the computation time of the 2-D slices given the paths to 2.1 s.

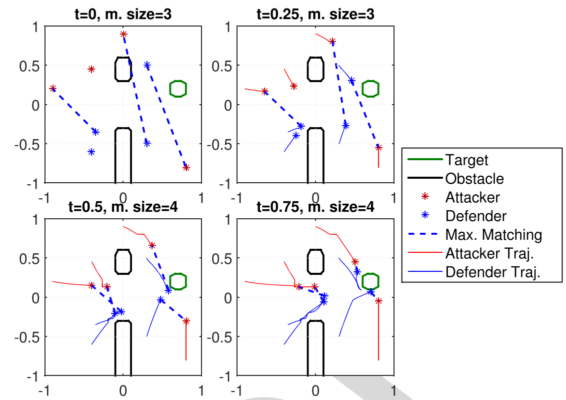


Fig. 9. Increasing maximum matching size over time.

### C. Defender Cooperation

To highlight the element of cooperation among the defenders, we examine Fig. 7 more closely. For the maximum matching results from the two-player HJI solutions (left subfigure), we see that the maximum matching creates the pairs $(P_{D_1}, P_{A_1})$, $(P_{D_2}, P_{A_4})$, $(P_{D_3}, P_{A_2})$, and $(P_{D_4}, P_{A_3})$. In general, such a pairing is not intuitively obvious. For example, it is not obvious, without knowledge of the pairwise outcomes, that $P_{D_1}$ can win against $P_{A_1}$ in a 1 versus 1 setting. If one were not sure whether $P_{D_1}$ can win against $P_{A_1}$, then $(P_{D_2}, P_{A_1})$ may seem like a reasonable pair. However, if $P_{D_2}$ defends against $P_{A_1}$, then $P_{D_1}$ would defend against $P_{A_2}$, leaving $P_{D_3}$ unable to find an attacker that $P_{D_3}$ can be guaranteed to win against to pair up with. The same observations can be made in many of the other pairings in both subfigures.

For the maximum matching results from the two-player path defense solutions (right subfigure), a semi-open-loop strategy is used. In this case, given the same initial conditions of the two teams, each defender may only be guaranteed to successfully defend against fewer attackers in a 1 versus 1 setting compared to when using the optimal closed-loop strategy from the two-player HJI solution. Regardless of the strategy that each defender uses, the maximum matching process can be applied to obtain optimal defender-attacker pairs given the strategy used and the set of attackers each defender can be guaranteed to win against in a 1 versus 1 setting.

### D. Real-Time Maximum Matching Updates

After determining all pairwise outcomes, pairwise outcomes of *any* joint state of the attacker–defender pair are characterized by the HJI approach. This allows for updates of the bipartite graph and its maximum matching as the players play out the game in real time. Fig. 9 shows the maximum matching at several time snapshots of a 4 versus 4 game. Each defender that is part of a maximum matching plays optimally against the paired-up attacker, and the remaining defender plays optimally against the closest attacker. The attackers' strategy is to move toward the target along the shortest path while steering clear of the obstacles by 0.125 units. The maximum matching is updated every $\Delta = 0.005$ s. 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 $t = 0.4$, the maximum matching size increases to 4, which guarantees that no attacker will be able to reach the target.

## VII. Conclusion and Future Work

A 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, 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 toward 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 attackers.

## Acknowledgment

AQ3 We thank Haomiao Huang for sharing MatLab code for 4-D HJI calculations.

## References

[1] H. Huang, "*Reachability-based control for human and autonomous agents in adversarial games*," Ph.D. dissertation, Stanford, CA, USA, Stanford Univ., 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 *Proc. IEEE ICRA*, 2011, pp. 1451–1456.

[3] Dept. Air Force, "United States Air Force Unmanned Aircraft Systems Flight Plan 2009-2047," oai.dtic.mil, Jan. 2009.

[4] H. Erzberger, "Automated conflict resolution for air traffic control," in *Proc. 25th Congr. Aeronaut. Sci.*, Jan. 2006, pp. 1–3.

[5] A. Madrigal, Autonomous Robots Invade Retail Warehouses. [Online]. Available: http://www.wired.com/wiredscience/2009/01/retailrobots/

[6] M. Earl and R. D'Andrea, "A decomposition approach to multi-vehicle cooperative control," *Robot. Auton. Syst.*, vol. 55, no. 4, pp. 276–291, 2007.

[7] G. Chasparis and J. Shamma, "Linear-programming-based multi-vehicle path planning with adversaries," in *Proc. Amer. Control Conf.*, 2005, pp. 1072–1077.

[8] 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," in *Proc. IEEE Conf. Decision Control*, 2004, pp. 2609–2614.

[9] J. McGrew, J. How, L. Bush, B. Williams, and N. Roy, "Air combat strategy using approximate dynamic programming," in *Proc. AIAA Guidance, Navig., Control Conf.*, Aug. 2008, pp. 1641–1654.

[10] R. Isaacs, *Differential Games*. Hoboken, NJ, USA: Wiley, 1967.

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

[12] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proc. National Acad. Sci.*, vol. 93, no. 4, pp. 1591–1595, 1996. [Online]. Available: http://www.pnas.org/content/93/4/1591.abstract

[13] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. Berlin, Germany: Springer-Verlag, 2002

[14] J. Ding, J. Sprinkle, S. S. Sastry, and C. J. Tomlin, "Reachability calculations for automated aerial refueling," in *Proc. IEEE Conf. Decision Control*, Cancun, Mexico, 2008, pp. 3706–3712.

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

[16] M. Karpinski and W. Rytter, *Fast Parallel Algorithms for Graph Matching Problems*. New York, NY, USA: Oxford Univ. Press, 1998.

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

# AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

AQ1 = Please provide keywords.
AQ2 = Please provide Associate Editor.
AQ3 = Please note that the last line in first footnote was captured as acknowledgement.

END OF ALL QUERIES

# 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 an attacking team with $N_A$ attackers and a defending team with $N_D$ defenders playing on a compact domain with obstacles. The attacking team aims to send $M$ of the $N_A$ attackers to some target location, while the defending team aims to prevent that by capturing attackers or indefinitely delaying attackers from reaching the target. Although the analysis of this game plays an important role in many applications, the optimal solution to this game is computationally intractable when $N_A > 1$ or $N_D > 1$. In this paper, we present two approaches for the $N_A = N_D = 1$ case to determine pairwise outcomes, and a graph theoretic maximum matching approach to merge these pairwise outcomes for an $N_A, N_D > 1$ 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, and that the two-dimensional "path defense" approach is considerably more scalable with the number of players while maintaining defender performance guarantees.**

***Index Terms*—Author, please supply index terms/keywords for your paper. To download the IEEE Taxonomy go to http://www.ieee.org/documents/taxonomy_v101.pdf.** AQ1

## I. Introduction

Multiplayer reach-avoid games are differential games between two adversarial teams of cooperative players playing on a compact domain with obstacles. The "attacking team" aims to send as many team members, called "attackers," to some target set as quickly as possible. 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 also powerful theoretical tools for analyzing realistic situations in robotics, aircraft control, security, and other domains [3]–[5].

The multiplayer reach-avoid game is difficult to analyze because the two teams have conflicting and asymmetric goals, while complex cooperation within each team may exist. In addition, optimal solutions are impossible to compute using traditional dynamic programming approaches due to the intrinsic high dimensionality of the joint state space. Previously, in [6], where a team of defenders assumes that the attackers move toward their target in straight lines, a mixed-integer linear programming approach was used. [7] assumes that the attackers use a linear feedback control law, and a mixed integer linear program was then relaxed into linear program. In complex pursuit-evasion games where players may change roles over time, a nonlinear model-predictive control [8] approach has been investigated. Approximate dynamic programming [9] has also been used to analyze reach-avoid games.

Although the above techniques provide some useful insight, they only work well when strong assumptions are made or when accurate models of the opposing team can be obtained. To solve general reach-avoid games, the Hamilton–Jacobi–Isaacs (HJI) approach [10] is ideal when the game is low-dimensional. The approach involves solving an HJI partial differential equation (PDE) in the joint state space of the players to compute a reach-avoid set, which partitions the players' joint state space into a winning region for the defending team and one for the attacking team. The optimal strategies can then be extracted from the gradient of the solution. This approach is particularly useful because of the numerical tools [11]–[13] available, and has been able to solve several practical problems [2], [11], [14]. The HJI approach 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 its complexity scales exponentially with the number of players, making the approach only tractable for the two-player game. Thus, complexity-optimality trade-offs must be made.

For the two-player reach-avoid game, we first present the two-player HJI solution [2], which computes a 4-D reach-avoid set that determines which player wins the game assuming both players use the closed-loop optimal control strategy. Next, we propose the "path defense" approximation to the HJI solution, in which the defenders utilize a "semi-open-loop" control strategy. Here, we approximate 2-D slices of the reach-avoid sets by solving 2-D Eikonal equations, and provide guarantees for the defending team's performance.

For the multiplayer reach-avoid game, we propose to merge the $N_A N_D$ pairwise outcomes using the graph theoretic maximum matching, which can be efficiently computed by known algorithms [15], [16]. The maximum matching process incorporates cooperation among defenders without introducing significant additional computation cost. When players on each team have identical dynamics, only a *single* HJI PDE needs to be solved to characterize *all* pairwise outcomes. Furthermore, when applying maximum matching to the two-player path defense solution, the computational complexity scales linearly with the number of attackers, as opposed to quadratically with the total number of players in the HJI approach.

## II. Reach-Avoid Problem

### A. Multiplayer Reach-Avoid Game

Consider $N_A + N_D$ players partitioned into the set of $N_A$ attackers, $\{P_{A_i}\}_{i=1}^{N_A} = \{P_{A_1}, P_{A_2}, \ldots, P_{A_{N_A}}\}$ and the set of $N_D$ defenders, $\{P_{D_i}\}_{i=1}^{N_D} = \{P_{D_1}, \ldots, P_{D_{N_D}}\}$, whose states are confined in a bounded, open domain $\Omega \subset \mathbb{R}^2$. The domain $\Omega$ is partitioned into $\Omega = \Omega_{\text{free}} \cup \Omega_{\text{obs}}$, where $\Omega_{\text{free}}$ is a compact set representing the free space in which the players can move, while $\Omega_{\text{obs}} = \Omega \setminus \Omega_{\text{free}}$ corresponds to obstacles in the domain.

Manuscript received September 21, 2015; revised April 29, 2016 and May 1, 2016; accepted May 27, 2016. This work was supported in part by the National Science Foundation under CPS:ActionWebs (CNS-931843), by the Office of Naval Research under the HUNT (N0014-08-0696) and SMARTS (N00014-09-1-1051) MURIs and by Grant N00014-12-1-0609, and by the Air Force Office for Scientific Research under the CHASE MURI (FA9550-10-1-0567). Recommended by Associate Editor X. XXXX.

AQ2 M. Chen and C. J. Tomlin are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720 USA (e-mail: mochen72@eecs.berkeley.edu; tomlin@eecs.berkeley.edu)).

Z. Zhou is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: zyzhou@stanford.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2016.2577619

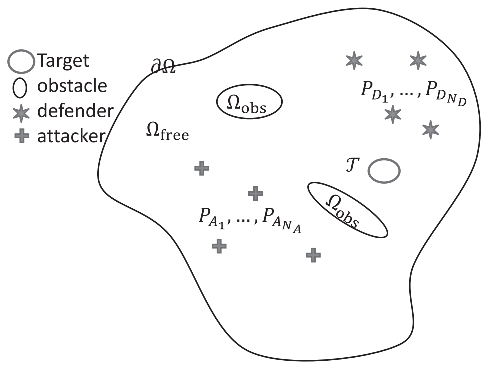


Fig. 1. Components of a multiplayer reach-avoid game.

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 $x^0_{A_i} \in \Omega_{\text{free}}, i = 1, 2, \ldots, N_A, x^0_{D_i} \in \Omega_{\text{free}}, i = 1, 2, \ldots, N_D$, we assume the dynamics of the players to be defined by the following decoupled system for $t \geq 0$:

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

where $v_{A_i}$, $v_{D_i}$ denote maximum speeds for $P_{A_i}$ and $P_{D_i}$ respectively, and $a_i$, $d_i$ denote controls of $P_{A_i}$ and $P_{D_i}$ respectively. We assume that $a_i$, $d_i$ are drawn from the set $\Sigma = \{\sigma : [0, \infty) \rightarrow \overline{B}_2 | \sigma \text{ is measurable}\}$, where $\overline{B}_2$ denotes the closed unit disk in $\mathbb{R}^2$. We also constrain the players to remain within $\Omega_{\text{free}}$ for all time. Denote the joint state of all players by $\mathbf{x} = (\mathbf{x}_A, \mathbf{x}_D)$ where $\mathbf{x}_A = (x_{A_1}, \ldots x_{A_{N_A}})$ is the attacker joint state $\{P_{A_i}\}_{i=1}^{N_A}$, and $\mathbf{x}_D = (x_{D_1}, \ldots, x_{D_{N_D}})$ is the defender joint state $\{P_{D_i}\}_{i=1}^{N_D}$.

The attacking team wins whenever $M$ of the $N_A$ attackers reach some target set without being captured by the defenders; $M$ is prespecified with $0 < M \leq N_A$. 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 $N_A - M + 1$ attackers from reaching $\mathcal{T}$. An illustration of the game setup is shown in Fig. 1.

Let $\mathcal{C}_{ij} = \{\mathbf{x} \in \Omega^{N_A + N_D} | \|x_{A_i} - x_{D_j}\|_2 \leq R_C\}$ denote the capture set. $P_{A_i}$ 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 problems:

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

### B. Two-Player Reach-Avoid Game

We will answer the above questions about the $N_A$ versus $N_D$ reach-avoid game by using the solution to the two-player 1 versus 1 game as a building block. In the two-player game, we denote the attacker $P_A$, the defender $P_D$, their states $x_A$, $x_D$, and their initial conditions $x^0_A, x^0_D$. Their dynamics are

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

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

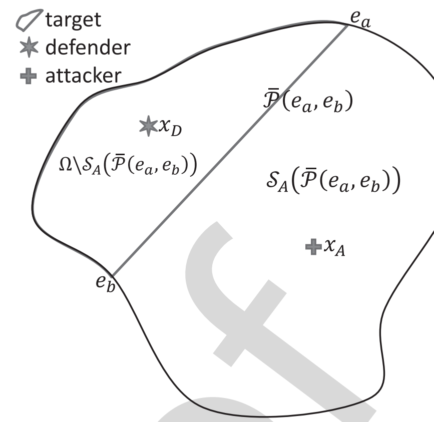


Fig. 2. Components of a path defense game.

$P_A$ wins if it reaches the target $\mathcal{T}$ without being captured by $P_D$. $P_D$ wins if it can prevent $P_A$ from winning by capturing $P_A$ or indefinitely delaying $P_A$ from reaching $\mathcal{T}$. For the two-player reach-avoid game, we seek to answer the following:

1) Given $\mathbf{x}^0$ and $\mathcal{T}$, is the defender guaranteed to win?
2) More generally, given $x_A$ and $\mathcal{T}$, what is the set of initial positions from which the defender is guaranteed to win?

## III. HJI Solution of the 1 Versus 1 Game

The HJI approach for solving differential games is outlined in [2], [11], and [17]. The optimal joint closed-loop control strategies for the attacker and the defender in a two-player reach-avoid game can be obtained by solving a 4-D HJI PDE. This solution allows us to determine whether the defender will win against the attacker in a 1 versus 1 setting.

In the two-player game, the attacker aims to reach $\mathcal{T}$ while avoiding $\mathcal{C}$. Both players also avoid $\Omega_{\text{obs}}$. In particular, the defender wins if the attacker is in $\Omega_{\text{obs}}$, and vice versa. Therefore, we define the terminal set and avoid set to be

$$\begin{aligned} R &= \{\mathbf{x} \in \Omega^2 | x_A \in \mathcal{T}\} \cup \{\mathbf{x} \in \Omega^2 | x_D \in \Omega_{\text{obs}}\} \\ A &= \mathcal{C} \cup \{\mathbf{x} \in \Omega^2 | x_A \in \Omega_{\text{obs}}\}. \end{aligned} \quad (3)$$

Given (3), we can define the corresponding implicit surface functions $\phi_R, \phi_A$ required for solving the HJI PDE. Since $\Omega \subset \mathbb{R}^2$, the result is $\mathcal{RA}_\infty(R, A) \in \mathbb{R}^4$, a 4-D reach-avoid set. If $\mathbf{x}^0 \in \mathcal{RA}_\infty(R, A)$, then the attacker is guaranteed to win the game by using the optimal control *even if* the defender is also using the optimal control; if $\mathbf{x}^0 \notin \mathcal{RA}_\infty(R, A)$, then the defender is guaranteed to win the game by using the optimal control *even if* the attacker is also using the optimal control.

## IV. Path Defense Solution to the 1 Versus 1 Game

We approximate 2-D slices of the 4-D reach-avoid set (or simply "2-D slices") in the path defense approach. Each slice will be taken at an attacker position. Here, we will assume that the defender is not slower than the attacker: $v_A \leq v_D$.

### A. 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 (Fig. 2). We

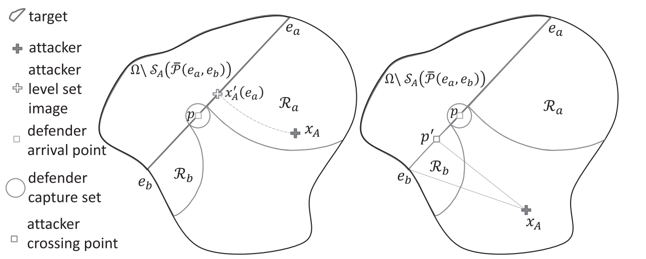


Fig. 3. (Left) If the attacker is in $\mathcal{R}_a \cup \mathcal{R}_b$ and moves toward $e_a$, then the attacker will be able to reach $e_a$ without being captured. (Right) If the attacker is not in $\mathcal{R}_a \cup \mathcal{R}_b$, there is no point on the path $p\prime \in \bar{\mathcal{P}}(e_a, e_b)$ that the can be reached without being captured.

denote the target set as $\mathcal{T} = \Omega \setminus \mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$ for two given points on the boundary $e_a$, $e_b$. $\mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$ 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 called the **anchor points** of path $\bar{\mathcal{P}}(e_a, e_b)$.

Denote the shortest path between **any** two points $x, y \in \Omega_{\text{free}}$ to be $\mathcal{P}(x, y)$, with length $\text{dist}\,(x, y)$, and requiring the attacker and defender durations of $t_A(x, y)$, $t_D(x, y)$ to traverse, respectively. We will also use $\text{dist}\,(\cdot, \cdot)$ with one or both arguments being sets in $\Omega$ to denote the shortest distance between the arguments.

*Definition 2—Attacker's Side of the Path:* A path of defense $\bar{\mathcal{P}}(e_a, e_b)$ partitions the domain $\Omega$ into two regions. Define $\mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$ 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 $\mathcal{T} = \Omega \setminus \mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$.

## B. Solving the Path Defense Game

A path defense game can be directly solved by computing a 4-D reach-avoid set. Since the direct solution is time- and memory-intensive, we propose an efficient approximation of 2-D slices that is conservative toward the defender.

*Definition 3—Defendable Path:* Given $\mathbf{x}^0 = (x_A^0, x_D^0)$, 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:* $\bar{\mathcal{P}}(e_a, e_b)$ 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 $\bar{\mathcal{P}}(e_a, e_b)$.

Checking whether a path $\bar{\mathcal{P}}(e_a, e_b)$ is defendable involves a 4-D reach-avoid set calculation, so instead we check 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 2-D distance calculations; the definitions and lemma are illustrated in Fig. 3.

*Definition 5—Attacker Level Set Image :* Given attacker position $x_A(t)$, define the attacker level set image with respect to anchor point $e_a$ to be $x'_A(t; e_a) = \{x \in \bar{\mathcal{P}}(e_a, e_b) : t_A(x, e_a) = t_A(x_A(t), e_a)\}$. $x'_A$ is the unique point on $\bar{\mathcal{P}}(e_a, e_b)$ such that $t_A(x'_A, e_a) = t_A(x_A, e_a)$. Define $x'_A(t; e_b)$ similarly by replacing $e_a$ with $e_b$. For convenience, we sometimes omit the time argument and write $x'_A(e_a)$.

*Proposition 1:* $\text{dist}\,(x'_A(e_b), e_a) \le \text{dist}\,(x'_A(e_a), e_a)$.

*Proof:* First note that

$$\begin{aligned}\text{dist}\,(e_a, e_b) &\le \text{dist}\,(x_A, e_a) + \text{dist}\,(x_A, e_b) \\ &= \text{dist}\,(x'_A(e_a), e_a) + \text{dist}\,(x'_A(e_b), e_b)\,.\end{aligned}$$

Then, since the left-hand side is given by $\text{dist}\,(e_a, e_b) = \text{dist}\,(e_a, x'_A(e_b)) + \text{dist}\,(x'_A(e_b), e_b)$, the result follows. ■

*Definition 6—Capture Set:* Define the capture set to be $D_C(y, t) = \{x \| \|x - y(t)\|_2 \le R_C\}$. We will drop the second argument of $D_C$ when $y$ does not depend on time.

*Remark 1:* Given $\bar{\mathcal{P}}(e_a, e_b)$, suppose the attacker level set image is within defender's capture set at some time $s$

$$x'_A(s; e_a) \in D_C(x_D, s) \;\; (\text{or } x'_A(s; e_b) \in D_C(x_D, s))\,.$$

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

$$\begin{gathered}x'_A(t; e_a) \in D_C(x_D, t) \forall\, t \ge s \\ (\text{or } x'_A(t; e_b) \in D_C(x_D, t) \forall\, t \ge s)\,.\end{gathered}$$

This is because the attacker level set image can move at most as fast as the attacker, who is not faster than the defender.

*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 with point $p$ and anchor point $e_a$ as follows:

$$\mathcal{R}_a(p) = \{x : \text{dist}\,(x, e_a) \le \text{dist}\,(D_C(p), e_a)\}\,. \tag{4}$$

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

*Lemma 1:* Suppose $x_D^0 = p \in \bar{\mathcal{P}}(e_a, e_b)$ and $v_A = v_D$. Then, $\bar{\mathcal{P}}(e_a, e_b)$ is strongly defendable if and only if $x_A^0$ is outside the region induced by $p$: $x_A^0 \in \Omega \backslash (\mathcal{R}_a \cup \mathcal{R}_b)$.

*Proof:* See Fig. 3. Assume $x_A^0 \notin \mathcal{T} = \Omega \setminus \mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$; 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 $\Omega \setminus \mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$ without being captured. Without loss of generality (WLOG), suppose $x_A^0 \in \mathcal{R}_a$. To capture the attacker, the defender's capture set must contain $x'_A(e_a)$ or $x'_A(e_b)$ at some time $t$. By Definition 7, we have $\text{dist}\,(x_A^0, e_a) < \text{dist}\,(D_C(p), e_a)$, so $t_A(x'_A(e_a), e_a) < t_D(D_C(p), e_a)$. By Proposition 1, $\text{dist}\,(x'_A(e_b), e_a) \le \text{dist}\,(x'_A(e_a), e_a)$, so it suffices to show that the defender's capture set cannot reach $x'_A(e_a)$ before the attacker reaches $e_a$.

If the attacker moves toward $e_a$ along $\mathcal{P}(x_A^0, e_a)$ with maximum speed, then $x'_A(e_a)$ will move toward $e_a$ along $\mathcal{P}(x'_A(e_a), e_a)$ at the same speed. Since $t_A(x_A, e_a) = t_A(x'_A(e_a), e_a) < t_D(D_C(p), e_a)$, $x_A$ will reach $e_a$ before the defender capture set $D_C(x_D, t)$ does.

Next we show, by contradiction, that if $x_A \notin \mathcal{R}_a \cup \mathcal{R}_b$, then the attacker cannot reach $\Omega \setminus \mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$ without being captured. Suppose $P_A$ will reach some point $p\prime$ before $D_C(P_D, t)$ does,i.e., $\text{dist}\,(x_A^0, p\prime) < \text{dist}\,(D_C(x_D^0), p\prime) = \text{dist}\,(D_C(p), p\prime)$. WLOG, assume $p\prime \in \mathcal{P}(p, e_b)$, and note that $\text{dist}\,(D_C(p), e_b) < \text{dist}\,(x_A^0, e_b)$, since the attacker is not in $\mathcal{R}_b$. Starting with the definition of the shortest path, we have

$$\begin{aligned}\text{dist}\left(x_A^0, e_b\right) &\le \text{dist}\left(x_A^0, p\prime\right) + \text{dist}\left(p\prime, e_b\right) \\ &< \text{dist}\left(D_C(p), p\prime\right) + \text{dist}\left(p\prime, e_b\right) \\ &= \text{dist}\left(D_C(p), e_b\right) \\ \text{dist}\left(x_A^0, e_b\right) &< \text{dist}\left(x_A^0, e_b\right) \;\; (\text{since } x_A^0 \notin \mathcal{R}_a).\end{aligned} \tag{5}$$

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

If $v_A < v_D$, $P_A$ being outside of $\mathcal{R}_a \cup \mathcal{R}_b$ becomes a sufficient condition 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, if the defender can arrive at $p$ before the attacker moves into $\mathcal{R}_a(p) \cup \mathcal{R}_b(p)$, then $\bar{\mathcal{P}}(e_a, e_b)$ 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 by checking whether there exists some $p \in \bar{\mathcal{P}}(e_a, e_b)$ such that $t_D(x_D^0, p) \le t_A(x_A^0, \mathcal{R}_a(p) \cup \mathcal{R}_b(p))$. If so, then $\bar{\mathcal{P}}(e_a, e_b)$ is strongly defendable. The next lemma shows that it is necessary and sufficient to check whether *one* special point, $p^* \in \bar{\mathcal{P}}(e_a, e_b)$, can be the first arrival point for strongly defending $\bar{\mathcal{P}}(e_a, e_b)$.

*Remark 2:* Given $p \in \bar{\mathcal{P}}(e_a, e_b)$, $\text{dist}\,(x_A^0, \mathcal{R}_a(p)) = \text{dist}\,(x_A^0, e_a) - \text{dist}\,(D_C(p), e_a)$. Similarly, $\text{dist}\,(x_A^0, \mathcal{R}_b(p)) = \text{dist}\,(x_A^0, e_b) - \text{dist}(D_C(p), e_b)$.

*Lemma 2:* Define $p^* \in \bar{\mathcal{P}}(e_a, e_b)$ 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 by definition.

We now show the other direction's 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 show that if choosing $p^*$ as the first entry point does not allow the defender to defend $\bar{\mathcal{P}}(e_a, e_b)$, then no other entry point does.

Suppose that the defender cannot defend $\bar{\mathcal{P}}(e_a, e_b)$ by choosing $p^*$ as the first entry point, but can defend $\bar{\mathcal{P}}(e_a, e_b)$ by choosing another entry point $p\prime$. WLOG, assume $\text{dist}\,(D_C(p^*), e_a) > \text{dist}\,(D_C(p'), e_a)$. 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 2, we have

$$\begin{aligned} \text{dist}\left(x_A^0, \mathcal{R}_a(p^*)\right) &= \text{dist}\left(x_A^0, e_a\right) - \text{dist}\left(D_C(p^*), e_a\right) \\ t_A\left(x_A^0, \mathcal{R}_a(p^*)\right) &= t_A\left(x_A^0, e_a\right) - t_A\left(D_C(p^*), e_a\right). \end{aligned} \tag{6}$$

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

$$\begin{aligned} \text{dist}\left(x_A^0, \mathcal{R}_a(p\prime)\right) &= \text{dist}\left(x_A^0, e_a\right) - \text{dist}\left(D_C(p\prime), e_a\right) \\ t_A\left(x_A^0, \mathcal{R}_a(p\prime)\right) &= t_A\left(x_A^0, e_a\right) - t_A\left(D_C(p\prime), e_a\right). \end{aligned} \tag{7}$$

Then, subtracting the above two equations, we see that the attacker can get to $\mathcal{R}_a$ sooner by the following amount:

$$\begin{aligned} & t_A\left(x_A^0, \mathcal{R}_a(p^*)\right) - t_A\left(x_A^0, \mathcal{R}_a(p\prime)\right) \\ &= t_A\left(D_C(p\prime), e_a\right) - t_A\left(D_C(p^*), e_a\right) \\ &= t_A\left(p\prime, p^*\right) \ge t_D(p\prime, p^*). \end{aligned} \tag{8}$$

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 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 defender "loses time" by going to $p\prime$). By the triangle inequality, we have

$$\begin{aligned} \text{dist}\left(x_D^0, p^*\right) &\le \text{dist}\left(x_D^0, p\prime\right) + \text{dist}\left(p\prime, p^*\right) \\ \text{dist}\left(x_D^0, p^*\right) - \text{dist}\left(x_D^0, p\prime\right) &\le \text{dist}\left(p\prime, p^*\right) \\ t_D\left(x_D^0, p^*\right) - t_D\left(x_D^0, p'\right) &\le t_D(p\prime, p^*). \end{aligned} \tag{9}$$

■

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 $x_D^0 \in \mathcal{D}(e_a, e_b; x_A^0) = \{x : t_D(x, p^*) \le t_A(x_A^0, \mathcal{R}_a \cup \mathcal{R}_b)\}$.

The computations in this algorithm can be efficiently done by solving a series of 2-D Eikonal equations by using a fast marching level set method (FMM) [12], reducing our 4-D problem to 2-D. Fig. 4 illustrates the proof of Lemma 2 and the defender winning region $\mathcal{D}$.

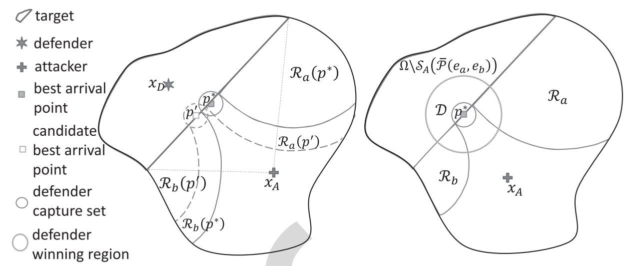


Fig. 4. (Left) If the defender cannot defend $\bar{\mathcal{P}}(e_a, e_b)$ by first going to $p^*$, then the attacker cannot defend $\bar{\mathcal{P}}(e_a, e_b)$ by going to any other point $p$. (Right) Defender winning region $\mathcal{D}$.

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

The central idea in using path defense is that if the target set is enclosed by some strongly defendable path for some $e_a, e_b$, then the defender can win the game using the semi-open-loop strategy outlined in this section, *even if* the attacker uses the optimal control. Checking for strongly defendable paths adds more conservatism toward 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. However, we can reduce the number of paths that needs to be checked by only checking 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.

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 $p^* \in \bar{\mathcal{P}}(e_a, e_b)$ (an open-loop strategy), then move toward $x'_A(e_a)$ or $x'_A(e_b)$ until the level set image is captured (a closed-loop strategy). Finally, the defender can simply track the captured level set image (a closed-loop strategy). This is a "semi-open-loop" strategy. The following algorithm approximates a 2-D slice conservatively toward 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 $e_a \in \partial\Omega$.
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 2-D slice, representing the conservative winning region for the defender in the two-player reach-avoid game.

## V. From Two-Player to Multiplayer

### A. Maximum Matching

We piece together the outcomes of all attacker–defender pairs using maximum matching as follows:

*Algorithm 2*:

1) Construct a bipartite graph with two sets of nodes $\{P_{A_i}\}_{i=1}^{N_A}$, $\{P_{D_i}\}_{i=1}^{N_D}$. Each node represents a player.
2) For all $i, j$, 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 (e.g., [15], [16]) to find a maximum matching in the graph.

After finding a maximum matching, we can guarantee an upper bound on the number of attackers that is be able to reach the target. If the maximum matching is of size $m$, then the defending team would be

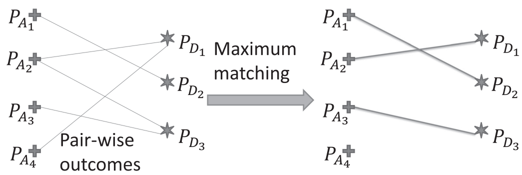


Fig. 5. Illustration of using maximum matching to conservatively approximate the multiplayer reach-avoid game.

able to prevent *at least* $m$ attackers from reaching the target, and thus $N_A - m$ is an upper bound on the number of attackers that can reach the target. The maximum matching approach is illustrated in Fig. 5.

### B. Time-Varying Defender-Attacker Pairings

With the next algorithm, the bipartite graph and its corresponding maximum matching can be updated, potentially in real time, as the players change positions during the game:

*Algorithm 3*:

1) Given each $x_{D_i}$ and each $x_{A_j}$, determine whether $P_{A_j}$ can win against $P_{D_i}$ for all $i, j$.
2) Assign a defender to each attacker that is part of a maximum matching.
3) For a short duration $\Delta$, 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 $\Delta$ and repeat steps 1 to 3 with the new player positions.

As $\Delta \to 0$, the above procedure continuously computes a bipartite graph and its maximum matching. As long as each defender uses a winning control input against the paired-up attacker, the size of maximum matching can never decrease.

### C. Application to the Two-Player HJI Solution

In general, solving $N_A N_D$ 4-D HJI PDEs gives us the pairwise outcomes between every attacker–defender pair. The computation time required is thus $CN_A N_D$, where $C$ is the time required to solve a single 4-D HJI PDE. The pairwise outcomes can then be merged together to approximate the $N_A$ versus $N_D$ game. In the case where each team has a single maximum speed, solving *one* 4-D HJI PDE would characterize all pairwise outcomes.

Since the solution to the 4-D HJI PDE characterizes pairwise outcomes based on any attacker–defender joint-state, it allows for real-time updates of the maximum matching. As players move to new positions, the pairwise outcome can be updated 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 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 step 3, the speeds of defenders come in only through a single distance calculation from $p^*$, which only needs to be done once per attacker position. Therefore, the total computation time required is on the order of $C_1 + C_2 N_A$, where $C_1$ is the time required for steps 1 and 2, $C_2$ is the time required for steps 3 to 5.

### E. Defender Cooperation

One of the strengths of the maximum matching approach is its simplicity in the way cooperation among the defenders is incorporated from pairwise outcomes. More specifically, cooperation is incorporated using the knowledge of the strategy of each teammate, and the knowledge of which attackers each teammate can win against in a 1 versus 1 setting.

The knowledge of the strategy of each teammate is incorporated in the following way. When the pairwise outcomes for each defender is computed, a particular defender strategy used. The strategy of each defender is then used to compute pairwise outcomes, which are used in the maximum matching process. Each defender may use the optimal closed-loop strategy given by the two-player HJI solution, the semi-open-loop strategy given by the two-player path defense solution, or even another strategy that is not described in this paper. In fact, different defenders may use a different strategy.

As already mentioned, all of the information about the strategy of each defender is used to compute the pairwise outcomes. Since each pairwise outcome specifies a winning region for the corresponding defender, each defender can be guaranteed to win against a set of attackers in a 1 versus 1 setting. The set of attackers against which each defender can win is then used to construct the bipartite graph on which maximum matching is performed. While executing the joint defense strategy as a team, each defender simply needs to execute its *pairwise* defense strategy against the attacker to which the defender is assigned.

The maximum matching process optimally combines the information about teammates' strategies and competence to derive a joint strategy to prevent as many attackers from reaching the target as possible. The size of the maximum matching then guarantees an upper bound on the number of attackers that can reach the target. To our knowledge, no other method can synthesize a joint defender control strategy that can provide such a guarantee in a multiplayer game.

## VI. Numerical Results

We use a 4 versus 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. Computations were done on a laptop with a Core i7-2640M processor with 4 GB of memory.

### A. HJI Formulation

Fig. 6 shows the results of solving the corresponding 4-D HJI PDE in blue. Computing the 4-D reach-avoid set on a grid with 45 points in each dimension took approximately 30 min. All players have the same maximum speed, so only a single 4-D HJI PDE needed to be solved. To visualize the 4-D reach-avoid set, we take 2-D slices of the 4-D reach-avoid set sliced at each attacker's position.

Fig. 7 shows the bipartite graph and maximum matching obtained from the pairwise outcomes. The maximum matching is of size 4. This guarantees that if each defender plays optimally against the attacker matched by the maximum matching, then *no* attacker will be able to reach the target.

### B. Path Defense Formulation

Fig. 6 shows, in red, the results of using the path defense approach to compute conservative approximations of the 4-D reach-avoid set sliced at various attacker positions.

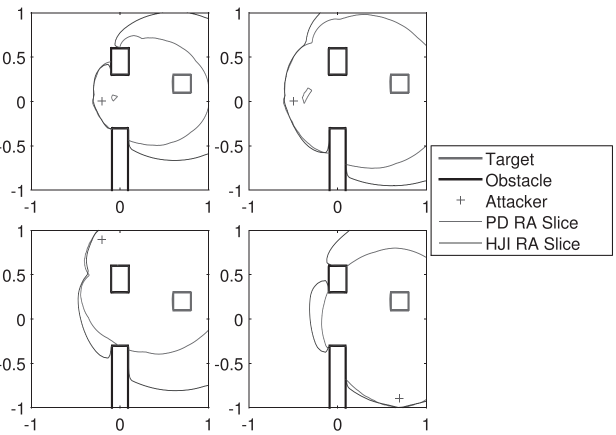


Fig. 6. Reach-avoid slices computed using the HJI approach and the path defense approach.

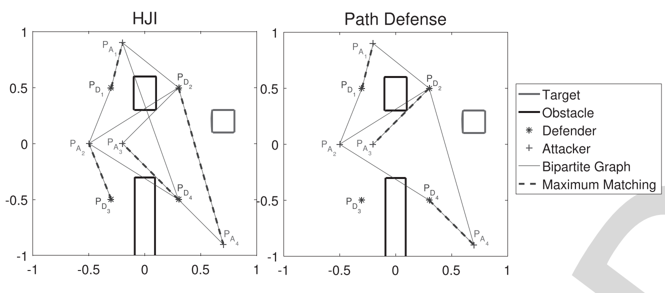


Fig. 7. Merging pairwise outcomes with maximum matching.

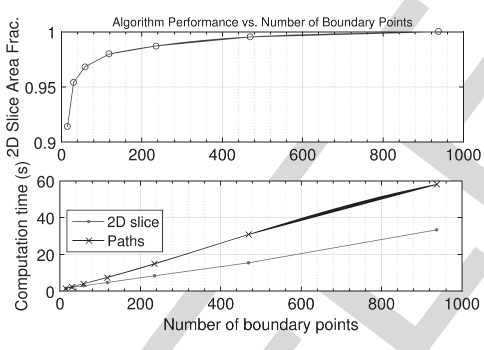


Fig. 8. Performance of the Path Defense Solution.

Fig. 7 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 $200 \times 200$ grid, and 937 paths were used to compute the results in Fig. 6. Computation time varies with the number of paths we chose in steps 1 and 2 in Algorithm 1. Taking the union of the defender winning regions from more paths will give a less conservative result, but requires more computation time. A summary of the performance is shown in Fig. 8. With 937 paths, the computation of paths took approximately 60 s, and the computation of the 2-D slice given the set of paths took approximately 30 s. However, very few paths are needed to approximate a 2-D slice: Even with as few as 30 paths, the computed 2-D slice covers more than 95% of the area of the 2-D slice computed using 937 paths. This reduces the computation time of the paths to 2.5 s, and the computation time of the 2-D slices given the paths to 2.1 s.

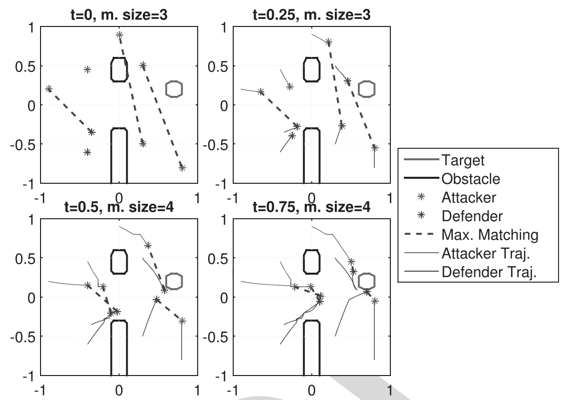


Fig. 9. Increasing maximum matching size over time.

### C. Defender Cooperation

To highlight the element of cooperation among the defenders, we examine Fig. 7 more closely. For the maximum matching results from the two-player HJI solutions (left subfigure), we see that the maximum matching creates the pairs $(P_{D_1}, P_{A_1})$, $(P_{D_2}, P_{A_4})$, $(P_{D_3}, P_{A_2})$, and $(P_{D_4}, P_{A_3})$. In general, such a pairing is not intuitively obvious. For example, it is not obvious, without knowledge of the pairwise outcomes, that $P_{D_1}$ can win against $P_{A_1}$ in a 1 versus 1 setting. If one were not sure whether $P_{D_1}$ can win against $P_{A_1}$, then $(P_{D_2}, P_{A_1})$ may seem like a reasonable pair. However, if $P_{D_2}$ defends against $P_{A_1}$, then $P_{D_1}$ would defend against $P_{A_2}$, leaving $P_{D_3}$ unable to find an attacker that $P_{D_3}$ can be guaranteed to win against to pair up with. The same observations can be made in many of the other pairings in both subfigures.

For the maximum matching results from the two-player path defense solutions (right subfigure), a semi-open-loop strategy is used. In this case, given the same initial conditions of the two teams, each defender may only be guaranteed to successfully defend against fewer attackers in a 1 versus 1 setting compared to when using the optimal closed-loop strategy from the two-player HJI solution. Regardless of the strategy that each defender uses, the maximum matching process can be applied to obtain optimal defender-attacker pairs given the strategy used and the set of attackers each defender can be guaranteed to win against in a 1 versus 1 setting.

### D. Real-Time Maximum Matching Updates

After determining all pairwise outcomes, pairwise outcomes of *any* joint state of the attacker–defender pair are characterized by the HJI approach. This allows for updates of the bipartite graph and its maximum matching as the players play out the game in real time. Fig. 9 shows the maximum matching at several time snapshots of a 4 versus 4 game. Each defender that is part of a maximum matching plays optimally against the paired-up attacker, and the remaining defender plays optimally against the closest attacker. The attackers' strategy is to move toward the target along the shortest path while steering clear of the obstacles by 0.125 units. The maximum matching is updated every $\Delta = 0.005$ s. 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 $t = 0.4$, the maximum matching size increases to 4, which guarantees that no attacker will be able to reach the target.

## VII. Conclusion and Future Work

A 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, 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 toward 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 attackers.

## Acknowledgment

AQ3 We thank Haomiao Huang for sharing MatLab code for 4-D HJI calculations.

## References

[1] H. Huang, "*Reachability-based control for human and autonomous agents in adversarial games*," Ph.D. dissertation, Stanford, CA, USA, Stanford Univ., 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 *Proc. IEEE ICRA*, 2011, pp. 1451–1456.

[3] Dept. Air Force, "United States Air Force Unmanned Aircraft Systems Flight Plan 2009-2047," oai.dtic.mil, Jan. 2009.

[4] H. Erzberger, "Automated conflict resolution for air traffic control," in *Proc. 25th Congr. Aeronaut. Sci.*, Jan. 2006, pp. 1–3.

[5] A. Madrigal, Autonomous Robots Invade Retail Warehouses. [Online]. Available: http://www.wired.com/wiredscience/2009/01/retailrobots/

[6] M. Earl and R. D'Andrea, "A decomposition approach to multi-vehicle cooperative control," *Robot. Auton. Syst.*, vol. 55, no. 4, pp. 276–291, 2007.

[7] G. Chasparis and J. Shamma, "Linear-programming-based multi-vehicle path planning with adversaries," in *Proc. Amer. Control Conf.*, 2005, pp. 1072–1077.

[8] 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," in *Proc. IEEE Conf. Decision Control*, 2004, pp. 2609–2614.

[9] J. McGrew, J. How, L. Bush, B. Williams, and N. Roy, "Air combat strategy using approximate dynamic programming," in *Proc. AIAA Guidance, Navig., Control Conf.*, Aug. 2008, pp. 1641–1654.

[10] R. Isaacs, *Differential Games*. Hoboken, NJ, USA: Wiley, 1967.

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

[12] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proc. National Acad. Sci.*, vol. 93, no. 4, pp. 1591–1595, 1996. [Online]. Available: http://www.pnas.org/content/93/4/1591.abstract

[13] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. Berlin, Germany: Springer-Verlag, 2002

[14] J. Ding, J. Sprinkle, S. S. Sastry, and C. J. Tomlin, "Reachability calculations for automated aerial refueling," in *Proc. IEEE Conf. Decision Control*, Cancun, Mexico, 2008, pp. 3706–3712.

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

[16] M. Karpinski and W. Rytter, *Fast Parallel Algorithms for Graph Matching Problems*. New York, NY, USA: Oxford Univ. Press, 1998.

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

# AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

AQ1 = Please provide keywords.
AQ2 = Please provide Associate Editor.
AQ3 = Please note that the last line in first footnote was captured as acknowledgement.

END OF ALL QUERIES