A Path Defense Approach to The Multiplayer Reach-Avoid Game

Mo Chen, Zhengyuan Zhou, and Claire J. Tomlin

***Abstract*— We consider a multiplayer reach-avoid game played between N attackers and N defenders moving with simple dynamics on a general two-dimensional domain. The attackers attempt to win the game by sending at least M of them ($1 \leq M \leq N$) to a target location while the defenders try to prevent the attackers from doing so by capturing them. The analysis of this game plays an important role in collision avoidance, motion planning, and aircraft control, among other applications involving cooperative agents. The high dimensionality of the game makes computing an optimal solution for either side intractable when $N > 1$. To address this issue, we present an efficient, approximate solution to the 1 vs. 1 problem. We call the approximate solution the "path defense solution", which is conservative towards the defenders. This serves as a building block for an approximate solution of the multiplayer game. Compared to the classical Hamilton-Jacobi-Isaacs approach for solving the 1 vs. 1 game, our new method is orders of magnitude faster, and scales much better with the number of players.**

I. Introduction

Differential games have been studied extensively, and are powerful theoretical tools in robotics, aircraft control, security, and other domains [1], [2], [3]. The multiplayer reach-avoid game (to be defined precisely in Section II) is a differential game between two adversarial teams of cooperating players, where one team attempts to reach a certain target quickly while the other team aims to prevent the opposing team from reaching the target. One example of a reach-avoid game is the popular game Capture-the-Flag (CTF) [4], [5]. In robotics and automation, CTF has been explored most notably in the Cornell Roboflag competition, where two opposing teams of mobile robots are directed by human players to play the game [6]. Several results related to motion planning and human-robot interactions have been reported from the competition [7], [8], [9], [10].

A multiplayer reach-avoid game is a complex game due to both the conflicting goals of the two teams and the cooperation among the players within each team, rendering the optimal solution for each team nontrivial to obtain and visualize. In [4], [5], the authors showed that even in a 1 vs. 1 scenario capture the flag game, human players sometimes lose in situations in which an optimal winning strategy exists. General multiplayer reach-avoid games are even more difficult to intuit, and computing optimal solutions becomes computationally intractable due to the intrinsic high dimensionality of the joint state space. Multiplayer differential games have been previously addressed using various techniques. In [7], 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 [11], optimal defender strategies are determined using a linear program, with the assumption that the attackers use a linear feedback control law. In complex pursuit-evasion games where players may change roles over time, nonlinear model-predictive control [12] and approximate dynamic programming [13] approaches have been investigated. In both cases, opponent strategies are estimated based on explicit prediction models.

One framework for a general multiplayer reach-avoid game is the classical Hamilton-Jacobi-Isaacs (HJI) approach [14], in which an HJI partial differential equation (PDE) is solved to obtain optimal strategies for both teams. Modern numerical tools such as [15], [16], [17] are able to solve the HJI PDE when the dimensionality of the problem is low. The solution to the PDE partitions the joint state space into two regions: a set of initial joint states from which one team wins, and a set of initial joint states from which the other team wins. Furthermore, the associated optimal joint controls can be extracted as well. Despite the power that the HJI framework and the numerical tools have to offer, solving a general multiplayer reach-avoid game is computationally intractable due to the high dimensionality of the game: the joint state space of two N-player teams in a two-dimensional (2D) domain is $4N$-dimensional ($4N$D). Numerically, when the state space is discretized, the number of grid points scales exponentially with the number of dimensions. As a result, the inherent trade-off between optimality of the solutions and computational complexity must be considered.

[18] considered defender-attacker pairs and computed the optimal solutions for each pair using the HJI framework. The pair-wise interactions are then tied together by using the graph-theoretical maximum matching algorithm [19], [20] to determine optimal pairings. This procedure approximates the HJI solution to the multiplayer game by combining the solutions from the N^2 two-player games, and reduces the computation complexity from solving a $4N$D HJI PDE to solving N^2 4-dimensional (4D) HJI PDEs. This is a substantial complexity reduction; however, solving 4D HJI PDEs is still very time- and memory-intensive, especially when the number of PDEs that need to be solved is N^2. To address this issue, we present the path defense solution, which approximates the solution to the 4D HJI PDEs using

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

Preprint submitted to 53rd IEEE Conference on Decision and Control.
Received March 20, 2014.

a series of two-dimensional (2D) calculations. Our new approach is conservative towards the defender, and provide sufficient conditions for the defender to win.

Our contributions can be stated as follows. First, our path defense approach provides a computationally efficient approximation to the solution proposed in [5] and [18] for the two-player reach-avoid game. Having computed this approximation, the same extension of the two-player game to the multiplayer game as in [18] can be easily implemented at almost no additional computational cost. Furthermore, we will demonstrate that the computation complexity of the path defense approach will scale with N as opposed to N^2. Hence, our approach provides an appealing solution, especially when the number of players becomes large. In addition, some cooperation is taken into account by the maximum matching process.

II. Problem Formulation

We consider a multiplayer reach-avoid game between a team of N attackers, $\{P_{A_i}\}_{i=1}^N = \{P_{A_1}, P_{A_2}, \dots, P_{A_N}\}$ and a team of N defenders, $\{P_{D_i}\}_{i=1}^N = \{P_{D_1}, \dots, P_{D_N}\}$. Each player is confined in a bounded, open domain $\Omega \subset \mathbb{R}^2$, which can be partitioned as follows: $\Omega = \Omega_{free} \cup \Omega_{obs}$. Ω_{free} is a compact set representing the free space in which the players can move, while $\Omega_{obs} = \Omega \setminus \Omega_{free}$ represents the obstacles that obstruct movement in the domain. Let $x_{A_i}, x_{D_j} \in \mathbb{R}^2$ denote the states of the players P_{A_i} and P_{D_j} respectively. Initial conditions of the players are denoted by $x_{A_i}^0, x_{D_i}^0 \in \Omega_{free}, i = 1, 2, \dots, N$. We assume that the dynamics of the players are defined by the following decoupled system for $t \geq 0$:

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

where $a_i(\cdot), d_i(\cdot)$ represent the control functions of P_{A_i} and $P_{D_i}, i = 1, 2, \dots, N$ respectively. v_{A_i}, v_{D_i} represent the maximum speeds of the i^{th} attacker and i^{th} defender, respectively; each player can have a different maximum speed. We assume that every defender is no slower than all attackers: $\forall i, v_{D_i} \geq v_{A_j} \forall j$. We further assume that the control functions $a_i(\cdot), d_i(\cdot)$ 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 circle in $\mathbb{R}^2$.

As a clarification on the notation and terminology, the control functions (with a dot notation, e.g. $a_i(\cdot), d_i(\cdot)$ etc.) which are the entire control trajectories, are distinguished from the control inputs (such as $a_i(t), d_i(t)$ etc.) which are the instantaneous control inputs. Furthermore, given $x_{A_i}^0 \in \Omega_{free}$, we define the admissible control function set for P_{A_i} to be the the set of all control functions such that $x_{A_i}(t) \in \Omega_{free}, \forall i, \forall t \geq 0$. The admissible control function set for each defender $P_{D_i}, i = 1, 2, \dots, N$ is defined similarly, given that $x_{D_i}^0 \in \Omega_{free}$. The joint state of all the players is denoted by $\mathbf{x} = (x_{A_1}, \dots x_{A_N}, x_{D_1}, \dots, x_{D_N})$. The joint initial condition is denoted by $\mathbf{x^0} = (x_{A_1}^0, \dots, x_{A_N}^0, x_{D_1}^0, \dots, x_{D_N}^0)$.

In this reach-avoid game, the attacking team aims to reach the target $\mathcal{T} \subset \Omega_{free}$, a compact subset of the domain, without getting captured by the defenders. The capture conditions are formally described by the capture sets $\mathcal{C}_{ij} \subset \Omega^{2N}$ for the pairs of the players $(P_{A_i}, P_{D_j}), i, j = 1, \dots, N$. In general, $\mathcal{C}_{ij}$ can be an arbitrary compact subset of Ω^{2N}, which represents the set of the joint player states $\mathbf{x}$ at which P_{A_i} is captured by P_{D_j}. Hence, in the general case, the interpretation of capture is given by the sets $\mathcal{C}_{ij}$, which in turn depend on the specific situation one wishes to model. In this paper, we define the capture sets to be $\mathcal{C}_{ij} = \{\mathbf{x} \in \Omega^{2N} \mid x_{A_i} = x_{D_j}\}$, the interpretation of which is that P_{A_i} is captured by P_{D_j} if P_{A_i}'s position coincides with P_{D_j}'s position. The lack of capture radius allows for clearer presentation of our analysis, which can be extended to the case in which there is a positive capture radius.

In our multiplayer reach-avoid game, the attackers win when at least M attackers get to the target, and the defenders win if they can delay at least $N - M + 1$ attackers from reaching the target indefinitely. An illustration of the game setup is shown in Figure 1.

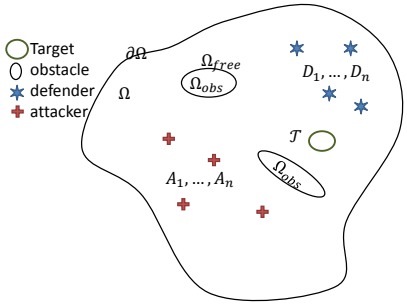

Fig. 1: An illustration of the components of a multiplayer reach-avoid game.

As an example of the multiplayer reach-avoid game, consider the special case in which each team only has one player. The solution to this special case will be a building block for the solution to the full multiplayer 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

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

We assume $v_D \geq v_A$. The players' joint state and joint initial condition become $\mathbf{x} = (x_A, x_D), \mathbf{x^0} = (x_A^0, x_D^0)$ respectively. The capture set for P_A is then simply

$$\mathcal{C} = \{(x_A, x_D) \in \Omega^2 \mid x_A = x_D\}. \tag{3}$$

The attacker P_A wins when it reaches the target $\mathcal{T}$ without being captured by the defender P_D. If the defender P_D can delay P_A from reaching $\mathcal{T}$ indefinitely, the defender wins.

This two-player reach-avoid game and its variant were first introduced and studied in [5] and [21]. In this paper, we provide a computationally efficient approximation to the HJI

Preprint submitted to 53rd IEEE Conference on Decision and Control.
Received March 20, 2014.

framework, and utilize the results in [18] to deal with the multiplayer reach-avoid game.

III. Hamilton-Jacobi-Isaacs Reachability

The multiplayer reach-avoid game is a differential game in which two teams have competing objectives [22]. By specifying the winning conditions, we can determine the winning regions of the attacker and the defender by numerically computing the HJI reach-avoid sets. These computations allow us to consider games with arbitrary terrain, domain, obstacles, target set, and player velocities. Furthermore, the results of HJI computation assume a closed-loop strategy for both players given previous information of the other players.

The setup for using HJI reachability to solve differential games can be found in [15], [23], [5], [18]. In summary, we are given the continuous dynamics of the system state:

$$\dot{\mathbf{x}} = f(\mathbf{x}, u, d), \mathbf{x}(0) = \mathbf{x_0}, \tag{4}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the system state, $u \in \mathbb{U}$ and $d \in \mathbb{D}$ are the joint control inputs of the attackers and defenders, respectively. The sets $\mathbb{U}$ and $\mathbb{D}$ represent the sets of the joint admissible control inputs of the attackers and the defenders, 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, which is true in our reach-avoid game defined in Equation (1), the Isaacs condition [14] holds and the two information patterns yield the same optimal solutions for both the attackers and the defenders.

To use the HJI reachability framework, we specify the terminal set as the attackers' winning condition, and propagate backwards this set subject to the constraint imposing that the attackers be outside the capture regions and the obstacles. This constraint is described by the avoid set. The result is a reach-avoid set that partitions the joint state space into a region that represents the joint initial conditions from which the attacking team is guaranteed to win, and a region that represents the joint initial conditions from which the defending team is guaranteed to win.

The HJI reachability framework involves solving the HJI PDE on a grid in the joint state space of all players. The full N vs. N game would involve gridding up a $4N$D state space, making the HJI PDE infeasible to solve numerically. [18] showed that maximum matching can piece together pairwise interactions between players, so that instead of solving the $4N$D HJI PDE, one could instead solve N^2 4D HJI PDEs that describe all the pair-wise interactions between the N players on each team. In the following sections, we will present the path defense solution to the two-player game. Our new method approximates 2D slices of the 4D reach-avoid set obtained from solving the HJI PDE, and involves a series of 2D distance calculations which can be done very efficiently by solving the 2D Eikonal equation using the fast marching method (FMM) [16].

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"). These slices will be the 4D reach-avoid set sliced at desired attacker positions.

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 solve it. The solution to the path defense game will be a building block for approximating 2D slices of the general two-player reach-avoid game, which will be presented in the next section.

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 particular points on $\partial\Omega$, and the target set is on one side of the 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. $\mathcal{S}_A\left(\bar{\mathcal{P}}(e_a, e_b)\right), \bar{\mathcal{P}}(e_a, e_b), e_a, e_b$ are defined below.

Definition 1: **Path of defense**. Denote the shortest path between two points x, y 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.

A path of defense, $\bar{\mathcal{P}}(e_a, e_b)$, is the shortest path between points e_a and e_b located on $\partial\Omega$. e_a and e_b are referred to as the **anchor points** of path $\bar{\mathcal{P}}(e_a, e_b)$.

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.

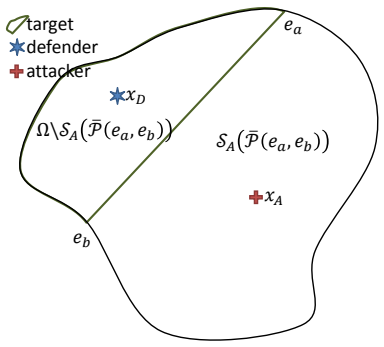

Fig. 2: An illustration of the components of a path defense game between two players.

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, is time- and memory- intensive. Hence, we propose an efficient way to approximate a 2D slice, which is conservative towards the defender.

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 unique. 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.

Remark 3: Given some path of defense $\bar{\mathcal{P}}(e_a, e_b)$, if the defender's position coincides with the level set image of the attacker, i.e. $x_D(s) = x_A\prime(s; e_a)$ (or $x_A\prime(s; e_b)$) at some time s, then there exists a control for the defender to always remain on the level set image of the attacker thereafter, i.e. $x_D(t) = x_A\prime(t; e_a)$ (or $x_A\prime(t; e_b)$) $\forall t \geq s$. This is because the attacker's image can move at most as fast as the attacker, whose speed is at most the defender's speed by assumption.

Definition 6: **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:

$$\mathcal{R}_a(p) = \{x : \text{dist}(x, e_a) \leq \text{dist}(p, e_a)\} \tag{5}$$

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 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 of Lemma 1: 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. The proof is illustrated in Figure 3.

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 must necessarily be on $x_A\prime(e_a)$ or $x_A\prime(e_b)$ at some time t. By definition 6, we have $\text{dist}(x_A^0, e_a) < \text{dist}(p, e_a)$, so $t_A(x_A\prime(e_a), e_a) < t_D(p, e_a)$. 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 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 $t_A(x_A, e_a) = t_A(x_A\prime(e_a), e_a) < t_D(p, e_a)$, $x_A\prime(e_a)$ will reach e_a before x_D 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 P_D does, i.e. $\text{dist}(x_A^0, p\prime) < \text{dist}(x_D^0, p\prime) = \text{dist}(p, p\prime)$. Without loss of generality, assume $p\prime \in \mathcal{P}(p, e_b)$, and note that $\text{dist}(p, e_b) < \text{dist}(x_A^0, e_b)$ since the attacker is not in $\mathcal{R}_b$. Starting with the definition of the shortest path, we have

$$\begin{aligned}\text{dist}(x_A^0, e_b) &\leq \text{dist}(x_A^0, p\prime) + \text{dist}(p\prime, e_b)\\ &\quad \text{(definition of shortest path / triangle inequality)}\\ &< \text{dist}(p, p\prime) + \text{dist}(p\prime, e_b)\\ &\quad \text{(by assumption, } \text{dist}(x_A^0, p\prime) < \text{dist}(p, p\prime))\\ &= \text{dist}(p, e_b)\\ &< \text{dist}(x_A^0, e_b)\\ &\quad \text{(since } x_A^0 \notin \mathcal{R}_a)\end{aligned} \tag{6}$$

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 Ω 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

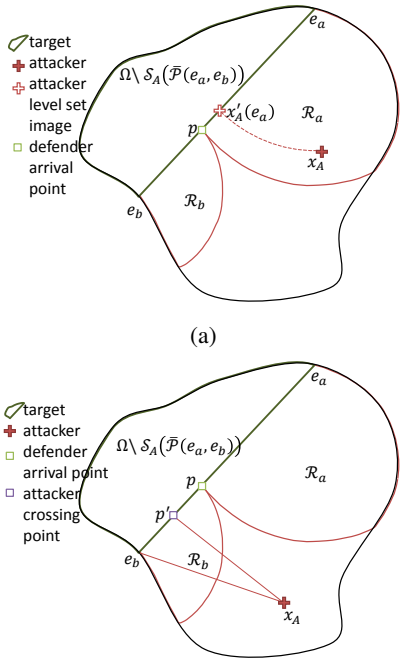

Fig. 3: (3a) Suppose the attacker is in $\mathcal{R}_a \cup \mathcal{R}_b$. If the attacker moves towards e_a, the defender cannot capture the attacker's level set image before the attacker reaches e_a. (3b) Suppose the attacker is not $\mathcal{R}_a \cup \mathcal{R}_b$, there is no point $p\prime$ on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

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 can 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 $t_A(x_A^0), \mathcal{R}_a(p) \cup \mathcal{R}_b(p)$.

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 following lemma shows that it is necessary and sufficient to perform the computations for only one special point denoted p^*.

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

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 of Lemma 2: 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 $\text{dist}(p^*, e_a) - \text{dist}(p\prime, e_a) > 0$. 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

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

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

$$\begin{aligned} \text{dist}\left(x_A^0, \mathcal{R}_a(p\prime)\right) &= \text{dist}(x_A^0, e_a) - \text{dist}(p\prime, e_a) \\ t_A\left(x_A^0, \mathcal{R}_a(p\prime)\right) &= t_A(x_A^0, e_a) - t_A(p\prime, e_a) \end{aligned} \tag{8}$$

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:

$$\begin{aligned} t_A\left(x_A^0, \mathcal{R}_a(p^*)\right) - t_A\left(x_A^0, \mathcal{R}_a(p\prime)\right) &= t_A(p\prime, e_a) - t_A(p^*, e_a) \\ &= t_A(p\prime, p^*) \\ &\geq t_D(p\prime, p^*) \end{aligned} \tag{9}$$

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. Then, by the triangle inequality, we have

$$\begin{aligned} \text{dist}(x_D^0, p^*) &\leq \text{dist}(x_D^0, p\prime) + \text{dist}(p\prime, p^*) \\ \text{dist}(x_D^0, p^*) - \text{dist}(x_D^0, p\prime) &\leq \text{dist}(p\prime, p^*) \\ t_D(x_D^0, p^*) - t_D(x_D^0, p\prime) &\leq t_D(p\prime, p^*) \end{aligned} \tag{10}$$

This completes the proof.

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) $\bar{\mathcal{P}}(e_a, e_b)$ is strongly defendable if $t_D(x_D^0, p^*) \leq t_A\left(x_A^0, \mathcal{R}_a(p^*) \cup \mathcal{R}_b(p^*)\right)$. Therefore, if $x_D^0 \in \mathcal{D}(e_a, e_b; x_A^0) = \{x : t_D(x, p^*) \leq t_A(x_A^0, \mathcal{R}_a \cup \mathcal{R}_b)\}$, then $\bar{\mathcal{P}}(e_a, e_b)$ is strongly defendable.

The computations in this algorithm can be efficiently computed by using FMM [16] on a 2D grid. Thus, we have conservatively solved the path defense problem, orginally a 4D problem, using a series of 2D fast marching calculations.

Figure 4 illustrates the proof of Lemma 2 and $\mathcal{D}$.

Preprint submitted to 53rd IEEE Conference on Decision and Control.
Received March 20, 2014.

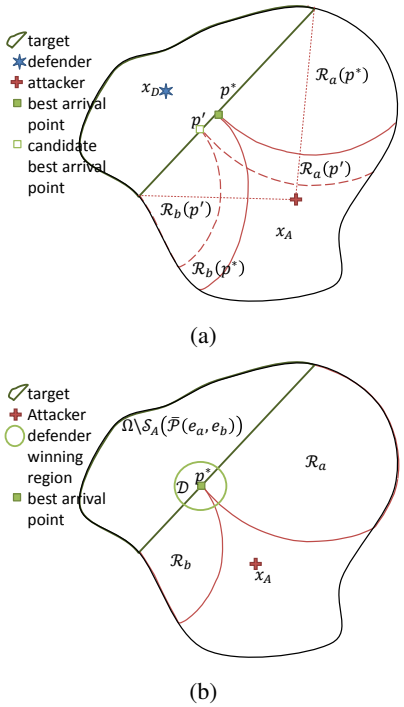

Fig. 4: (4a) If the defender cannot defend $\bar{\mathcal{P}}(e_a, e_b)$ by first going to p^*, then the defender cannot defend $\bar{\mathcal{P}}(e_a, e_b)$ by going to any other point p. (4b) Lemma 2 allows an easy computation of $\mathcal{G}(e_a, e_b)$.

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

In section II, 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 method is based on the path defense game from section IV and only involves 2D distance calculations, which can be solved efficiently using FMM.

In section IV, we described the path defense game as a reach-avoid game with the target set $\mathcal{T} = \Omega \backslash \mathcal{S}_A(\bar{\mathcal{P}}(e_a, e_b))$ and computed a 2D slice. We now present the path defense solution to compute a 2D slice in for an arbitrary target set. The path defense solution leverages the idea described in section IV. 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.

Naively, one could fix e_a, then search all other points on $e_b \in \partial\Omega$ 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. Therefore, picking e_a determines e_b.

As noted in section IV, 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 tractable.

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 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:

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 step 1.
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.

We will be using the solution to the two player game as a building block for approximating the solution to the multiplayer game. To calculate the solution to many defender-attacker pairs, we add the following step to the algorithm:

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

For a given domain, set of obstacles, and target set, steps 1 and 2 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 can needs to be done once per attacker position. Therefore, steps 3 to 5 scales 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.

Compare this to the HJI approach in [18], whose computation time is $C_3 N^2$, where C_3 is the time required to solve a 4D HJI PDE. Our new path defense solution is not only scales linearly with the number of players as opposed to quadratically, but as we will show in section VII, the constants C_1, C_2 are orders of magnitude smaller than C_3.

VI. Maximum Matching

We can determine whether the attacker can win the multiplayer reach-avoid game by combining the solution to the two player game and maximum matching [19], [20] from graph theory as follows:

1) Construct a bipartite graph with two sets of nodes $\{P_{A_i}\}_{i=1}^N, \{P_{D_i}\}_{i=1}^N$. Each node represents a player.

2) For each P_{D_i}, determine whether P_{D_i} can win against P_{A_j}, for all j using strong path defense.
3) Form a bipartite graph: Draw an edge between P_{D_i} and P_{A_j} if P_{D_i} wins against P_{A_j}.
4) 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].

After finding a maximum matching, we can determine an upper bound for the number of attackers that can 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, and thus at most $N - m$ attackers can reach the target. In particular, suppose that the attackers win when M of them reach the target. In this case, the defenders are guaranteed to win if they can prevent at least $N - M + 1$ attackers from reaching the target. To achieve this, the size of the maximum matching would need to be of size $m = N - M + 1$; this would give $M - 1$ as an upper bound for the number of attackers that are able to reach the target.

VII. Computational Results

We illustrate our path-defense and maximum matching approach in the example below. The 2D slices of the 4D reach-avoid sets are calculated using our path defense approach. The example is shown in Figures 5, 6, and 7. In this example, there are four attackers and four defenders playing on a square domain with obstacles. All players have equal speeds ($v_D = v_A$).

Figure 5 illustrates the defense of a single path $\bar{\mathcal{P}}(e_a, e_b)$. The regions $\mathcal{R}_a, \mathcal{R}_b$ induced by p^* are shown in red. p^* is calculated based on the position of the attacker, shown as a red cross. Any defender within the region $\mathcal{D}(e_a, e_b)$, shown in green, will be able to defend $\bar{\mathcal{P}}(e_a, e_b)$, and therefore defend the target set. Taking the union of all the defender winning regions for many paths, we obtain an approximation of the 2D slices shown in Figure 6.

Each subplot in Figure 6 shows the boundary of 2D slices for each fixed attacker position. Defenders, shown as blue stars, that are inside the boundary win against the particular attacker in each subplot. For example, in the top left subplot of Figure 6, the defender at $(0.3, 0.5)$ wins against the attacker at $(-0.2, 0)$, while the other three defenders lose against the attacker at $(-0.2, 0)$.

Figure 7 shows the resulting bipartite graphs (edges shown as thin solid blue lines) and the maximum matching (edges shown as thick dashed blue lines) after applying the algorithm described in Section VI. The maximum matching is of size 3, which means that at most 1 attacker is able to reach the target. Therefore, if $M > 1$ attackers getting to the target is required for the attackers to win, then the defenders are guaranteed to win.

As the players play out the game and reach a new joint-configuration, the bipartite graph and the maximum matching can be recomputed to obtain new optimal pairings for the defending team.

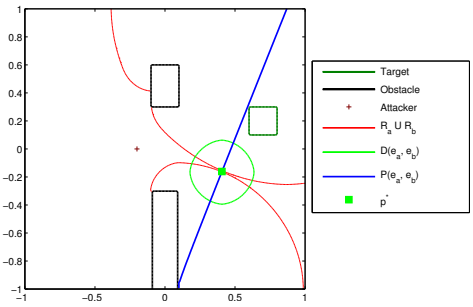

Fig. 5: Defense of a single path that encloses the target set.

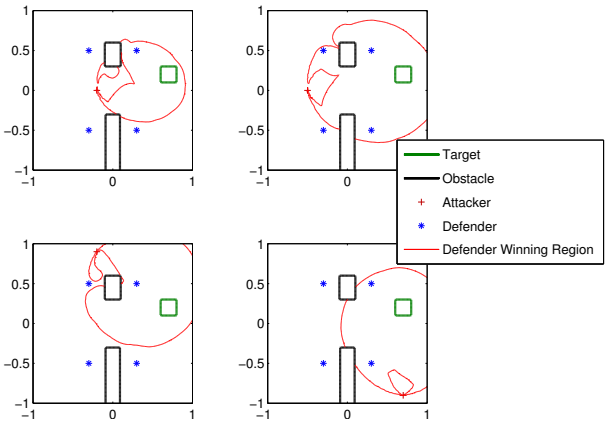

Fig. 6: 2D slices of reach-avoid sets at the positions of the four attackers. Defenders inside the 2D slice boundary are guaranteed to win against the attacker in each subplot.

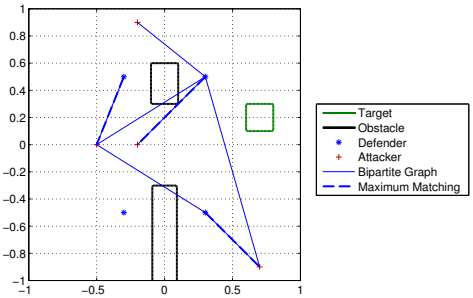

Fig. 7: Bipartite graph and maximum matching. Each edge, (thin solid blue line), connects a defender to an attacker against whom the defender is guaranteed to win, creating a bipartite graph. The dashed thick blue lines show a maximum matching of the bipartite graph. Here, a maximum matching of size 3 indicates that at most one attacker can reach the target.

Computations were done on a 200×200 grid, a grid resolution that was not possible when solving 4D HJI PDEs [18]. 937 paths were used to compute the results in Figure 6. Computation time varies with the number of paths we chose in steps 1 and 2 in the algorithm in Section V. A summary of the performance of our algorithm is shown in Figure 8. Computations were done on a Lenovo T420s laptop with a Core i7-2640M processor.

Preprint submitted to 53rd IEEE Conference on Decision and Control.
Received March 20, 2014.

Figure 8(top) shows the area of the 2D slices as a function of the number of paths used for the computation. The areas are expressed as an average (over the four attacker positions) fraction of the 2D slices computed using 937 paths. From this plot, we can see that very few paths are needed to approximate a 2D slice: Even with as few as 59 paths, the computed 2D slice covers more than 95% of the area of the 2D slice computed using 937 paths.

Figures 8(middle) and 8(bottom) show the paths computation and 2D slice computation times, denoted C_1 and C_2 respectively in Section V. If one uses 59 paths to compute the 2D slices, we would have $C_1 = 3.2$ seconds, and $C_2 = 2.1$ seconds, with a resulting overall complexity of $3.2 + 2.1N$ *seconds*. This is vastly superior to solving 4D HJI PDEs, whose complexity is $C_3 N^2$ with $C_3 \approx 30$ *minutes* [18].

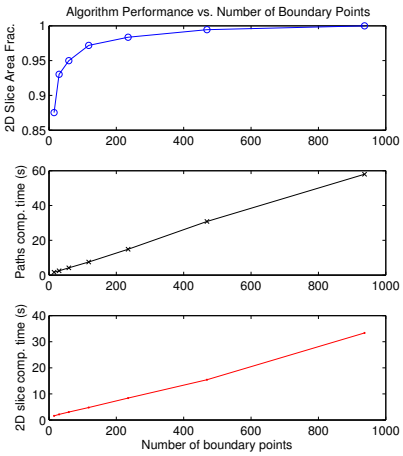

Fig. 8: Performance of the Path Defense Solution

VIII. Conclusions and Future Work

We have presented an efficient method, based on the idea of path defense, for approximating 2D slices of 4D reach-avoid sets in the two player reach-avoid game. By leveraging the properties of shortest paths between points on the domain boundary, we were able to compute these slices using a series of 2D distance calculations, which can be done very quickly using FMM. Our method is conservative towards the defender, but two to three orders of magnitude faster than the previous HJI reachability method [18].

By using maximum matching tie together two-player game solutions, we were also able to approximate the solution to the multiplayer reach-avoid game without significant additional computation overhead. A maximum matching algorithm determines the defender-attacker pairing that prevents the maximum number of attackers from reaching the target. This way, we were able to guarantee an upper bound on the number of attackers that are able to get to the target without directly solving the corresponding high dimensional, numerically intractable HJI PDE.

An immediate extension of this work is to investigate defender strategies that optimally promote an increase in maximum matching size. Many other extensions of this work naturally arise, including analyzing reach-avoid games with unfair teams with different numbers of players on each team, more complex player dynamics, or partial observability. u

References

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

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

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

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

[5] 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.

[6] R. D'Andrea and M. Babish, "The roboflag testbed," *Proceedings of the American Control Conference*, vol. 1, pp. 656–660, 2003.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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

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

[18] M. Chen, Z. Zhou, and C. Tomlin, "Multiplayer reach-avoid games via low dimensional solutions and maximum matching," in *Proceedings of the American Control Conference*, 2014.

[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] 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.

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

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