An Open-Loop Framework for Reach-Avoid Games

Zhengyuan Zhou, Haomiao Huang, Jerry Ding, Ryo Takei, and Claire J. Tomlin

Abstract

We consider a game in which one or more attacking players attempt to reach a target while avoiding both obstacles and capture by opposing, defending players. Unlike pursuit-evasion games, in this reach-avoid game the players must consider both opposing players and the target. This complexity makes finding solutions to such games difficult and computationally challenging, especially as the number of players grows. We propose an approach to solving such games in an open-loop sense, where the players commit to their control actions prior to the beginning of the game. This reduces the dimensionality of the required computations, and allows us to quickly compute feasible solutions in real time for domains with arbitrary obstacle topologies. We describe two such formulations, each of which is conservative towards one side, and derive algorithms based upon modified fast-marching methods (FMM) for efficiently computing their solutions. The formulations and algorithms are discussed in detail, along with simulation results demonstrating their use in scenarios with complex obstacle geometries and player speed profiles.

I. Introduction

In a reach-avoid game, one set of players attempt to arrive at a target set in the state-space, while avoiding a set of unsafe states, as well as interceptions by an opposing set of players. Throughout this paper, we refer to the two opposing sides as attackers and defenders, respectively. Such games encompass a large number of robotics and control applications. For example, many safe motion-planning problems may be formulated as reach-avoid games, in which the objective is to control one or more agents into a desired target region, while avoiding a set of obstacles or

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).

Z. Zhou, H. Huang, J. Ding, and C. J. Tomlin are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA {zhengyuan, haomiao, jding, tomlin}@eecs.berkeley.edu

possibly adversarial agents. Finding solution strategies for such games can be computationally expensive, even for games involving a single attacker and a single defender, and the computational burden can grow exponentially with the number of players.

This paper presents a method to mitigate the computational requirements for finding a solution to reach-avoid games, by addressing them in an open-loop sense, such that the two sides select their plans of action prior to the beginning of the game and do not change them over the course of play. In particular, we consider an *upper-value* game, in which the attacking side first selects its control inputs, which are then made known to the defenders, and a *lower-value* game, in which the defending side chooses first. The upper-value game is conservative from the point of the attackers, and gives an upper bound on the time to accomplish their objectives. The lower-value game, which is conservative from the point of the defenders, provides a lower bound on this time. The scenario we consider involves kinematically controlled players moving with spatially varying speed limits in a bounded domain containing obstacles.

The primary contribution of this work is an open-loop framework for efficiently computing solutions to reach-avoid games in the two-player case and also certain multi-agent cases. More specifically, we discuss formulations of the reach-avoid games in terms of the open-loop upper value and the open-loop lower value, and develop a set of modified fast-marching methods (FMM) that allow us to quickly compute solutions to these open-loop games, in the form of a set of player trajectories with provable properties. We emphasize here that the two open-loop values, in addition to being interesting for study in their own right, also provide bounds on the closed-loop value of the reach-avoid game. This closed-loop value is in general the solution to a Hamilton-Jacobi-Isaacs (HJI) equation, which can be very difficult to compute, in particular for games with a large number of players. Thus, our open-loop framework can be interpreted as a computationally efficient approximation framework for reach-avoid games, through the trade-off of a certain degree of optimality for a reduction in computational complexity.

Some of the theoretical results were presented in two previous publications [1], [2]. This work unifies the presentation of the open-loop games and their relationship to each other, while providing more detailed proofs of the technical results and extending the solutions to classes of multi-agent games. Additional simulation results are also included for the multi-agent cases.

We begin by reviewing related work in this domain in Section II. We then proceed to describe formulations of the problem for a single attacker and a single defender in Section III. The

classical closed-loop, Hamilton-Jacobi-Isaacs (HJI) formulation is first introduced in this section. The computational complexity of this approach then provides the motivation for open-loop formulations of the game. The theory of the upper and lower value games are presented in Section IV. For the upper-value game, we present an algorithm to compute a time-optimal path for the attacker to reach the target while avoiding obstacles and the defender in the low-dimensional state-space of a single player rather than in the joint state-space of both players. For the lower-value game, we present an algorithm which computes a lower bound to the time-to-reach function and derive control inputs for a defending player that achieves this bound. The properties of the two open-loop values as bounds on the closed-loop value is also discussed. The algorithms for computing these solutions are discussed in detail in Section V, based upon modifications of the FMM. Section VI extends the discussion of open-loop games to scenarios with multiple players on each side. Section VII presents simulation results demonstrating the application of the computational algorithms. We conclude with a discussion of the open-loop framework and future work in Section VIII.

II. Related Work

Multi-agent differential games, to which the reach-avoid game considered here belongs, have been a subject of significant past research. Given the complexity of these games, there has been no "one-size fits all" approach. Instead, the approaches that have been proposed often trade off between optimality of the solution (with respect to the time to achieve a player's objective), and the complexity of the computation. This section presents an overview of some relevant past work on reach-avoid games.

For certain games and game configurations, it is possible to construct strategies for the agents geometrically. A class of methods has been proposed for safe motion-planning in the presence of moving obstacles by computing the future set of states an obstacle may occupy, given the dynamics of the controlled agent and the obstacles. This set of states are then treated as obstacles in the joint state-time space, and paths are planned which avoid these states [3], [4], [5]. They are well suited for scenarios in which the obstacle motion can be unpredictable or even adversarial, so that in the presence of hard constraints on safety, one needs to account for the worst-case possibility that the disturbances may actively attempt to collide with the agent. However, these approaches tend to be limited to simple game configurations without complex static obstacle

configurations or inhomogeneous speed constraints from varying terrain.

For general cases, the classical approach is to formulate the game as a minimax problem, in which a value function is defined representing the time-to-reach of the attackers at the target, subject to the constraint that it is not captured by the defender. This value is then computed via a related Hamilton-Jacobi-Isaacs (HJI) partial differential equation (PDE), with appropriate boundary conditions [6], [7], [8], [9]. Solutions are typically found either using the method of characteristics [6], [8], where single trajectories are found by integrating backward from a known terminal condition, or via numerical approximation of the HJI equation on grids [9], [10]. Problem size limits the numerical computations, as the grid size required for approximating the value function can grow exponentially in the number of continuous states, while the method of characteristics is limited by the need to integrate backward from terminal states. Nonetheless, successful applications of such methods have been seen in a number of applications [11], [12].

In many cases, model-predictive control (MPC) is used when higher computational speed is required. In an MPC formulation, an optimization problem is solved over the control actions of one side while using a model to predict opponent actions. This solution is implemented for a short time period, and the optimization is then re-solved using the new agent states. Feedback via this re-computation is used to correct for errors in the prediction model. In the reach-avoid context, a defense strategy for intercepting multiple attackers with multiple defenders was developed in [13] using mixed-integer linear programming, with the assumption that attackers proceed toward the target in straight lines. Another method to solve for defender control actions was proposed in [14], in which linear programming was used under an assumed attacker strategy in the form of a linear feedback law. MPC approaches work best when the prediction model used is a good approximation of the actual strategy of the opponent. However, optimality and guarantees of capture for the solutions can be difficult to derive.

The approach we take in this work sacrifices some optimality in exchange for an improvement in computational speed, allowing solutions to be computed in real time. These solutions are conservative towards one set of players, in the sense that it is possible for those players to achieve better performance (in terms of how long the game lasts) if they were allowed to select control inputs in closed-loop rather than open-loop. On the other hand, this conservatism maintains certain guarantees of performance. In particular, the upper value solution ensures that for initial conditions where the upper value is finite, there exists a feasible, quickly computable

control such that the attackers will arrive at the target within some finite time. Similarly, the lower value solution ensures the existence of controls preventing the attacker from entering the target within certain finite time. Thus, these solutions can be interpreted as conservative strategies which ensure the objectives of one side to be achieved, regardless of the choice of strategy by the opposing side. In certain cases, they also coincide with the the closed-loop optimal solutions. Moreover, for many real-world applications, fast, feasible solutions with guarantees are often preferable to optimal solutions that require too much time to compute.

III. The Reach-Avoid Problem

We proceed by first discussing the reach-avoid game and our approach for a single attacker and defender. This serves to illustrate the theoretical and algorithmic approaches, and the methods will be generalized in a subsequent section to games with multiple players on each side.

A. Payoff Function

Suppose there are two players P_A and P_D, whose states are confined in a bounded, open domain $\Omega \subset \mathbb{R}^n$. The domain Ω can be further partitioned as follows: $\Omega = \Omega_{free} \cup \Omega_{obs}$, where Ω_{free} is a compact set representing the free space in which the two players can move, while $\Omega_{obs} = \Omega \setminus \Omega_{free}$ corresponds to obstacles in the domain. Let $x_A, x_D \in \mathbb{R}^n$ denote the state of players P_A and P_D, respectively. Then given initial conditions $x_A^0, x_D^0 \in \Omega$, we assume the dynamics of the players to be defined by the following decoupled system for $t \geq 0$:

$$\begin{aligned} \dot{x}_A(t) &= f_A(x_A(t))a(t), \quad x_A(0) = x_A^0, \\ \dot{x}_D(t) &= f_D(x_D(t))d(t), \quad x_D(0) = x_D^0. \end{aligned} \tag{1}$$

where f_A, f_D represent the spatially-varying maximum speeds for P_A and P_D (e.g. due to terrain variations), and a, d represent the controls of P_A and P_D. It is assumed that $f_A, f_D \colon \Omega \to [0, \infty)$ are bounded and Lipschitz continuous, and that a, d 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 ball in $\mathbb{R}^n$. Per the rules of the reach-avoid game, we constrain the controls of both players to be those which allow the player states to remain within the confines of the free space. In particular, for initial conditions $x_A^0, x_D^0 \in \Omega_{free}$, we define the admissible control set $\Sigma_A(x_A^0, x_D^0)$ for P_A to be those controls in Σ such that $x_A(t) \in \Omega_{free}$, $\forall t \geq 0$. The admissible control set $\Sigma_D(x_A^0, x_D^0)$ for P_D is defined in a similar

fashion. For convenience, we will drop the dependence of these sets on the initial condition when the context is clear.

We now define the payoff for the *reach-avoid game*. We first define two sets: $\mathcal{T} \subset \Omega_{free}$, the *target set*, and $\mathcal{A} \subset \Omega^2$, the *avoid set*, both assumed to be compact. The avoid set describes the capture condition in the reach-avoid game, namely the set of joint player states (x_A, x_D) at which P_A is captured by P_D. A typical example of an avoid set is given by $\mathcal{A} = \{(x_A, x_D) \in \Omega^2 \mid \|x_A - x_D\| \leq r\}$, for some $r \geq 0$, corresponding to a scenario in which P_A is captured if it comes within a capture radius r of P_D.

In a reach-avoid game, the goal for P_A is to reach the target set $\mathcal{T}$ as quickly as possible, while steering clear of the avoid set $\mathcal{A}$. On the other hand, the goal for P_D is to either capture P_A before P_A reaches the target set, or to delay P_A from entering $\mathcal{T}$ for as long as possible. Denoting the joint state by $\mathbf{x} = (x_A, x_D)$ and the joint initial condition by $\mathbf{x^0} = (x_A^0, x_D^0)$, we define the payoff of the game as the first time instant that P_A reaches the target set without being captured:

$$\mathcal{J}(\mathbf{x^0}, a, d) = \inf\{t \geq 0 \mid x_A(t) \in \mathcal{T}, \mathbf{x}(s) \notin \mathcal{A}, \forall s \in [0, t]\}, \tag{2}$$

where the infimum of the empty set is by convention $+\infty$. The adversarial interaction is represented by the fact that P_A and P_D attempt to minimize and maximize this payoff, respectively.

Next we briefly discuss the closed-loop HJI formulation of the game, whose complexity then provides the motivation for our open-loop formulations.

B. Closed-loop Game Formulation

Given the payoff function (2), the value function of the reach-avoid game depends on the information pattern employed by each player. In this section, we will introduce the value functions for a closed-loop game, defined using a *non-anticipative* information pattern, as commonly adopted in the differential game literature (see for example [15], [16]). Under this information pattern, a strategy for P_A defines the responses of P_A to possible control selections by P_D, and vice versa for player P_D. In order to preclude strategies which allow foreknowledge of the control selections by the opposing player, we only consider strategies which select controls in a causal fashion. In particular, the set of admissible strategies for P_A is given by

$$\mathcal{F}_A := \{\gamma : \Sigma_D \to \Sigma_A \mid \sigma_1(\tau) = \sigma_2(\tau), \text{ a.e. } \tau \in [0, t] \Rightarrow \gamma[\sigma_1](\tau) = \gamma[\sigma_2](\tau), \text{ a.e. } \tau \in [0, t]\}.$$

The set $\mathcal{F}_D$ of admissible strategies for P_D is defined similarly.

For $\mathbf{x^0} \in \Omega^2_{free}$, the upper value of the closed-loop reach-avoid game is given by

$$\overline{V}(\mathbf{x^0}) = \overline{V}(x_A^0, x_D^0) = \sup_{\gamma_D \in \mathcal{F}_D} \inf_{a \in \Sigma_A} \mathcal{J}(\mathbf{x^0}, a, \gamma_D[a]). \tag{3}$$

and the lower value of the closed-loop reach-avoid game is given by

$$\underline{V}(\mathbf{x^0}) = \underline{V}(x_A^0, x_D^0) = \inf_{\gamma_A \in \mathcal{F}_A} \sup_{d \in \Sigma_D} \mathcal{J}(\mathbf{x^0}, \gamma_A[d], d). \tag{4}$$

The closed-loop game is said to have value if the upper and lower values are equal:

$$\overline{V}(\mathbf{x^0}) = \underline{V}(\mathbf{x^0}), \ \forall \mathbf{x^0} \in \Omega^2. \tag{5}$$

Due to the decoupled system dynamics (1), the Isaacs' condition [8] holds, implying that the upper and lower values are equal, and hence the closed-loop game has a value V. Moreover, by standard techniques in differential games, it can be shown that this value is the constrained viscosity solution to a Hamilton-Jacobi-Isaacs (HJI) equation [17]. In particular, wherever it is finite and nonzero, V satisfies the following HJI equation in a viscosity solution sense:

$$-H(\nabla_{x_A} V, \nabla_{x_D} V, x_A, x_D) = 1, \tag{6}$$

where $H(p, q, x, y) = \inf_{a \in \overline{B}_n} \{p \cdot a f_A(x)\} + \sup_{b \in \overline{B}_n} \{q \cdot b f_D(y)\}$.

Ideally, we would like to solve the HJI equation (6) to obtain the value function $V(\mathbf{x^0})$. This would then allow us to derive, at least in principle, equilibrium closed-loop strategies for both players [8]. Although numerical methods for solving (6) on a Cartesian grid exist, they generally suffer from the curse of dimensionality. Specifically, with a uniform grid, the number of grid nodes grows exponentially with the dimension of the joint state space, while the dimension scales linearly with the number of players. This results in an exponential growth in complexity with the number of players, thus making the problem computationally intractable even for a modest number of players. Moreover, for cases in which computational solutions are possible (e.g. with two players and two-dimensional domains), the closed-loop value function often cannot be computed in real time [12].

C. Open-loop Game Formulation

Motivated by the difficulty of computing the closed-loop value function, we now introduce an open-loop formulation of the reach-avoid game. This will form the basis of the solution

approach presented in this paper. Under an open-loop information pattern, each player selects *controls* from the control spaces Σ_A or Σ_D, rather than *strategies* from the non-anticipative strategy space $\mathcal{F}$. This then results in the following modified definitions of the upper and lower values of the reach-avoid game.

Definition 1: The *open-loop upper value* and the *open-loop lower value* of the reach-avoid game are defined respectively as follows:

$$\begin{aligned} \overline{v}(\mathbf{x^0}) &= \inf_{a\in\Sigma_A} \sup_{d\in\Sigma_D} \mathcal{J}(\mathbf{x^0}, a, d), \\ \underline{v}(\mathbf{x^0}) &= \sup_{d\in\Sigma_D} \inf_{a\in\Sigma_A} \mathcal{J}(\mathbf{x^0}, a, d). \end{aligned} \tag{7}$$

These two open-loop values can be viewed as conservative calculations of the payoff, under the assumption that the opposing player has complete knowledge of one's choice of control, whether in an anticipative or non-anticipative fashion. Specifically, for the upper value, P_A first chooses its control over the whole time horizon $[0, \infty)$ and makes this control known to P_D, which then chooses its control in response. Thus the upper value is conservative toward P_A, who must consider all possible responses by P_D when choosing its control a. Similarly, the lower value is conservative towards P_D, who chooses first in this case. Due to this conservative bias towards each player, it can be shown that $\underline{v}(\mathbf{x^0}) \leq \overline{v}(\mathbf{x^0})$ for all $\mathbf{x^0}$. Moreover, using the definitions given in (3) and (4), one can verify in a straightforward manner that these two open-loop values in fact provide bounds on the two closed-loop values:

$$\underline{v}(\mathbf{x^0}) \leq \underline{V}(\mathbf{x^0}) \leq \overline{V}(\mathbf{x^0}) \leq \overline{v}(\mathbf{x^0}), \ \forall \mathbf{x^0} \in \Omega^2_{free}. \tag{8}$$

In particular, if the closed-loop reach-avoid game has a value, then the open-loop values provide upper and lower bounds on the value of the closed-loop game.

The open-loop formulation of the reach-avoid game represents a simplification of the information pattern, in that all decisions are made at the very beginning of the game. The conservatism adopted by each side allows approximations and bounds for the open-loop value functions to be quickly computed via the algorithms developed in this paper. This provides us with an approach to address the intractability of solving the HJI equations at some sacrifice of optimality. Furthermore, for initial states at which the open-loop upper and lower values coincide, the closed-loop value is simply given by $V(\mathbf{x^0}) = \underline{v}(\mathbf{x^0}) = \overline{v}(\mathbf{x^0})$. In such cases, the value of the closed-loop game can be obtained for the initial state $\mathbf{x^0}$ without explicitly solving the HJI equation (6).

For the synthesis of player controls in an open-loop reach-avoid game, we interpret the optimal controls with respect to either the open-loop upper value or the open-loop lower value as the Stackelberg equilibria of a zero-sum game [8]. In particular, we say that a pair of controls $\bar{a} \in \Sigma_A$ and $\bar{d} \in \Sigma_D$ is optimal with respect to the open-loop upper value if it satisfies:

$$\begin{aligned} \bar{a} &\in \arg\min_{a \in \Sigma_A} \sup_{d \in \Sigma_D} \mathcal{J}(\mathbf{x}^{\mathbf{0}}, a, d), \\ \bar{d} &\in \arg\max_{d \in \Sigma_D} \mathcal{J}(\mathbf{x}^{\mathbf{0}}, \bar{a}, d). \end{aligned} \tag{9}$$

Similarly, we say that a pair of controls $\underline{a} \in \Sigma_A$ and $\underline{d} \in \Sigma_D$ is optimal with respect to the open-loop lower value if it satisfies:

$$\begin{aligned} \underline{d} &\in \arg\max_{d \in \Sigma_D} \inf_{a \in \Sigma_A} \mathcal{J}(\mathbf{x}^{\mathbf{0}}, a, d), \\ \underline{a} &\in \arg\min_{a \in \Sigma_A} \mathcal{J}(\mathbf{x}^{\mathbf{0}}, a, \underline{d}). \end{aligned} \tag{10}$$

In the next section, we present the theoretical aspects related to finding the two open-loop values. This will motivate numerical methods for computing solutions to the open-loop games.

IV. The Open-Loop Upper Value and Lower Value

This section provides the theoretical foundations for solving the open-loop reach-avoid games. The two solutions have a number of similarities which we highlight. However, they are not exactly symmetric. In particular, while the upper value can be found exactly, the lower value is not amenable to direct computation. We present methods to find the exact value of the open-loop upper value game, and an approximation that finds a lower bound for the lower value. We also briefly discuss the relationship between the open-loop solutions and the HJI closed-loop solution.

A. Upper Value

We first examine the upper value game, which is conservative toward P_A. The value function for this game is defined in (7). Under this information pattern, the game is played in the following manner. First, P_A decides upon a control, attempting to reach the target set $\mathcal{T}$ while considering the worst-case possible actions of P_D. P_D then chooses its control, with full knowledge of P_A's control. If P_A is able to find a control that allows it to reach the target set even with this advantage given to P_D, then P_A is guaranteed to reach the target set no matter what P_D does.

The open-loop value $\overline{v}$ can be found by characterizing the set of points in the state space that P_A can reach no matter what P_D does and selecting a path within this set. Given a joint initial condition $\mathbf{x}^0$, we say that a point $y \in \Omega_{free}$ is *safe-reachable* if there exists a player P_A control such that for every player P_D control, P_A can reach y in finite time, while avoiding capture by player P_D. More precisely, we define a *safe-reachable set* for P_A as follows:

$$\mathcal{S} := \{y \in \Omega_{free} \mid \exists a \in \Sigma_A, \forall d \in \Sigma_D, \exists t \geq 0, x_A(t) = y, \mathbf{x}(s) \notin \mathcal{A}, \forall s \in [0,t]\}, \tag{11}$$

where $\mathbf{x}(\cdot) = (x_A(\cdot), x_D(\cdot))$ is the solution to (1). Note that the set $\mathcal{S}$ implicitly depends on the initial condition $\mathbf{x}^0$ of the reach-avoid game.

The safe reachable set provides us with a necessary and sufficient condition for $\overline{v}$ to be finite. Clearly, $\overline{v} < \infty$ if and only if $\mathcal{S} \cap \mathcal{T} \neq \emptyset$. Figure 1 illustrates $\mathcal{S}$ for an example where P_A is twice as fast as P_D. In this case, $\mathcal{A} = \{(x_A, x_D) \in \Omega^2 \mid x_A = x_D\}$. As shown here, the safe-reachable set is in general not the set of equal time-to-reach points for the two agents, as there may be points that are reachable by P_A, but the choice of controls to achieve this condition could result in capture by P_D. For comparison, the Apollonius circle showing the equal time-to-reach points of P_A and P_D is overlaid. In the following, we will provide a characterization of the set $\mathcal{S}$ in terms of two minimum-time functions, defined respectively from the perspectives of P_A and P_D.

Definition 2: Given $x_D^0 \in \Omega_{free}$ and $y \in \Omega_{free}$, the minimum time-to-capture for P_D is

$$\overline{t}(y; x_D^0) := \inf_{d \in \Sigma_D} \inf\{t \geq 0 \mid (y, x_D(t)) \in \mathcal{A}, x_D(0) = x_D^0\}. \tag{12}$$

That is, for a stationary P_A at y, $\overline{t}(y; x_D^0)$ is the shortest time for P_D, starting at x_D^0, to capture P_A. For compactness of notation, we shall also write $\overline{t}(y)$ when the context is clear.

Definition 3: Given an initial state $x_A^0 \in \Omega_{free}$, a set $R \subseteq \Omega_{free}$, and a location $y \in \Omega_{free}$, the minimum time-to-reach for P_A with constraint R is

$$\overline{w_R}(y; x_A^0) := \inf_{a \in \Sigma_A} \inf\{t \geq 0 \mid x_A(t) = y, x_A(0) = x_A^0, x_A(s) \in R, \forall s \in [0,t]\}. \tag{13}$$

Intuitively, $\overline{w_R}(y; x_A^0)$ quantifies how quickly P_A can reach a point y from an initial condition x_A^0, while remaining within the set R. We can now relate the two minimum-time functions to the safe-reachable set $\mathcal{S}$.

Proposition 1: Given a joint initial state $\mathbf{x^0} = (x_A^0, x_D^0) \in \Omega_{free}^2 \setminus \mathcal{A}$, let $\mathcal{M} \subseteq \Omega_{free}$ be the maximal set such that $\overline{w_{\mathcal{M}}}(y; x_A^0) < \overline{t}(y; x_D^0),\ \forall y \in \mathcal{M}$. Then $\mathcal{M} = \mathcal{S}$.

Proof: Let $\mathcal{E}$ be the collection of all sets $E \subseteq \Omega_{free}$ satisfing the inequality in the proposition, namely $\overline{w_E}(y; x_A^0) < \bar{t}(y; x_D^0),\ \forall y \in E$. Note that $\mathcal{E}$ is non-empty, as $E = \{x_A^0\}$ is an element of $\mathcal{E}$. Now consider the set $E^* := \bigcup_{E \in \mathcal{E}} E$. We claim that $E^* \in \mathcal{E}$, and hence E^* is the maximal element of $\mathcal{E}$ (i.e. $\mathcal{M} = E^*$). Indeed, for any sets $R, R' \subseteq \Omega_{free}$ such that $R \subseteq R'$, we have by the definition for the minimum time-to-reach function in (13) that $\overline{w_{R'}}(y; x_A^0) \leq \overline{w_R}(y; x_A^0)$, for all $y \in \Omega_{free}$. In particular, for any set $E \in \mathcal{E}$, we have $\overline{w_{E^*}}(y; x_A^0) \leq \overline{w_E}(y; x_A^0)$, for all $y \in \Omega_{free}$. Let $y \in E^*$, then given the definition of the set E^*, $y \in E$ for some $E \in \mathcal{E}$. Thus, $\overline{w_{E^*}}(y; x_A^0) \leq \overline{w_E}(y; x_A^0) < \bar{t}(y; x_D^0)$. This proves the claim.

To see that $E^* = \mathcal{S}$, first we note that both E^* and $\mathcal{S}$ are non-empty. In particular, given an initial condition $\mathbf{x^0} = (x_A^0, x_D^0) \in \Omega_{free}^2 \setminus \mathcal{A}$, x_A^0 is an element of both E^* and $\mathcal{S}$. Now take any point $y \in \mathcal{S}$. By definition, there exists $a^* \in \Sigma_A$, such that for every $d \in \Sigma_D$, there exists $t \geq 0$ so that $x_A(t) = y$ and $\mathbf{x}(s) \notin \mathcal{A}$, $\forall s \in [0, t]$. Let $t^* := \min\{t \geq 0 \mid x_A(t) = y\}$, where $x_A(\cdot)$ is the trajectory of P_A under (1), with the choice of control a^*. Then it can be seen that $x_A(\cdot)$ satisfies $x_A(s) \in \mathcal{S}$, $\forall s \in [0, t^*]$. This implies that $\overline{w_{\mathcal{S}}}(y; x_A^0) \leq t^*$. Furthermore, by the properties of a^*, we have that for any $d \in \Sigma_D$, the joint path satisfies $\mathbf{x}(t^*) \notin \mathcal{A}$, or equivalently $(y, x_D(t^*)) \notin \mathcal{A}$. By the compactness of $\mathcal{A}$ and the definition of the minimum time-to-capture function in (12), this in turn implies that $t^* < \bar{t}(y; x_D^0)$. Thus, we have $\overline{w_{\mathcal{S}}}(y; x_A^0) < \bar{t}(y; x_D^0),\ \forall y \in \mathcal{S}$. It then follows that $\mathcal{S} \in \mathcal{E}$, and so $\mathcal{S} \subseteq E^*$.

We will now proceed to show that $\mathcal{S}$ is a superset of every element in $\mathcal{E}$. Let $E \in \mathcal{E}$ and $z \in E$. For simplicity of argument, we assume that there exists a minimum-time control with respect to $\overline{w_E}(z; x_A^0)$. Namely, there exists $a^* \in \Sigma_A$ such that the corresponding trajectory x_A^* satisfies $t^* := \min\{t \geq 0 \mid x_A^*(t) = z\} = \overline{w_E}(z; x_A^0)$ and $x_A^*(s) \in E, \forall s \in [0, t^*]$. Since $E \in \mathcal{E}$, we have $\overline{w_E}(x_A^*(s); x_A^0) < \bar{t}(x_A^*(s); x_D^0),\ \forall s \in [0, t^*]$. Moreover, by the fact that x_A^* is a time-optimal path, we have $\overline{w_E}(x_A^*(s); x_A^0) = s,\ \forall s \in [0, t^*]$. By Definition 2, one can then infer that for every $d \in \Sigma_D$, the joint path under a^* and d satisfies $(x_A^*(s), x_D(s)) \notin \mathcal{A},\ \forall s \in [0, t^*]$. This implies that $z \in \mathcal{S}$. In the case that the infimum with respect to $\overline{w_E}(z; x_A^0)$ is not achieved, one can apply a similar line of reasoning to ϵ-optimal controls. From this result, it then follows that $E^* = \bigcup_{E \in \mathcal{E}} E \subseteq \mathcal{S}$. Combining the two set inclusions, we have $E^* = \mathcal{S}$. ■

Intuitively, the above result states that $\mathcal{S}$ is the largest set in the free space Ω_{free} such that player P_A can reach all points in $\mathcal{S}$ safely by traveling along a path contained in $\mathcal{S}$, regardless of the controls of player P_D. Such a path will be referred to as a *safe-reachable path*. The upper

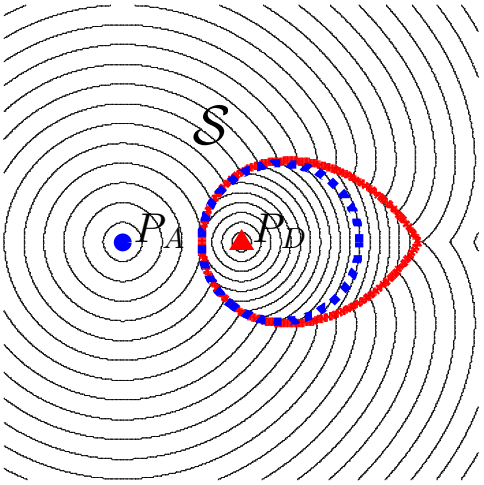


Fig. 1. **$\mathcal{S}$ is the subset partitioned by the solid red curve containing x_A^0. Level sets of $\overline{w_{\mathcal{S}}}$ on $\mathcal{S}$ and $\bar{t}_c$ on $\Omega\backslash\mathcal{S}$ are plotted. The dotted blue circle is the set of equal time-to-reach points of P_A and P_D when capture is not considered.**

value $\overline{v}$ can now be found using the minimum time-to-reach function $\overline{w_{\mathcal{S}}}$.

Theorem 1: $\overline{v}(\mathbf{x^0}) = \inf\{\overline{w_{\mathcal{S}}}(y; x_A^0) \mid y \in \mathcal{T}\}$, $\forall \mathbf{x^0} \in \Omega^2_{free}$.

Proof: First, consider the case in which $\overline{v}(\mathbf{x}^0) = \infty$. By definition of the upper value, this implies that for every P_A control $a \in \Sigma_A$, we have $\sup_{d\in\Sigma_D} \mathcal{J}(\mathbf{x^0}, a, d) = \infty$. Using (2), one can then infer that for every $a \in \Sigma_A$, there exists $d \in \Sigma_D$ such that for every $t \geq 0$, the joint path satisfies either $x_A(t) \notin \mathcal{T}$ or $\mathbf{x}(s) \in \mathcal{A}$ for some $s \in [0, t]$. Thus, for every $y \in \mathcal{T}$, we have $y \notin \mathcal{S}$, and hence $\overline{w_{\mathcal{S}}}(y; x_A^0) = \infty$, $\forall y \in \mathcal{T}$, as desired.

Next we consider the case in which $\overline{v}(\mathbf{x^0}) < \infty$. This implies that there exists $a \in \Sigma_A$ such that $\sup_{d\in\Sigma_D} \mathcal{J}(\mathbf{x^0}, a, d) < \infty$. Let Σ'_A be the set of all such controls, and $\mathcal{J}'(\mathbf{x^0}, a) := \sup_{d\in\Sigma} \mathcal{J}(\mathbf{x^0}, a, d)$. Fix any $a^* \in \Sigma'_A$, then by definition of the payoff function $\mathcal{J}$, there exists $t \geq 0$ such that $x_A(t) \in \mathcal{T}$ and $x(s) \notin \mathcal{A}$, $\forall s \in [0, t]$, $\forall d \in \Sigma_D$. For this choice of P_A control, let $t^* := \inf\{t \geq 0 \mid x_A(t) \in \mathcal{T}\}$, then it follows that for every $d \in \Sigma_D$, $\mathcal{J}(\mathbf{x^0}, a^*, d) = t^*$. Thus, $\mathcal{J}'(\mathbf{x^0}, a^*) = t^*$. Moreover, one can verify that $x_A(s) \in \mathcal{S}$, $\forall s \in [0, t^*]$. Let $z^* = x_A(t^*)$, then we have $\overline{w_{\mathcal{S}}}(z^*; x_A^0) \leq t^* = \mathcal{J}'(\mathbf{x^0}, a^*)$. Thus, $\inf_{y\in\mathcal{T}} \overline{w_{\mathcal{S}}}(y; x_A^0) \leq \mathcal{J}'(\mathbf{x^0}, a^*)$, $\forall a^* \in \Sigma'_A$, and so $\inf_{y\in\mathcal{T}} \overline{w_{\mathcal{S}}}(y; x_A^0) \leq \overline{v}(\mathbf{x^0})$.

Furthermore, $\overline{v}(\mathbf{x^0}) < \infty$ also implies $\mathcal{S} \cap \mathcal{T} \neq \emptyset$ and $\mathbf{x^0} \notin \mathcal{A}$. Let $y \in \mathcal{S} \cap \mathcal{T}$. For simplicity of argument, we assume that there exists a minimum-time control with respect to $\overline{w_{\mathcal{S}}}(y; x_A^0)$. Let $x_A^*(\cdot)$ be the corresponding time-optimal path. Define $t^* := \overline{w_{\mathcal{S}}}(y; x_A^0)$, then it follows by

Proposition 1 that $\overline{w_{\mathcal{S}}}(x_A^*(s); x_A^0) < \bar{t}(x_A^*(s); x_D^0),\ \forall s \in [0, t^*]$. Hence, for every $d \in \Sigma_D$, the joint path satisfies $(x_A^*(s), x_D(s)) \notin \mathcal{A},\ \forall s \in [0, t^*]$. From this, we have $\mathcal{J}(\mathbf{x^0}, a^*, d) \leq t^*$, $\forall d \in \Sigma_D$, which in turn implies that $\overline{v}(\mathbf{x^0}) \leq t^* = \overline{w_{\mathcal{S}}}(y; x_A^0)$. In the case that the infimum with respect to $\overline{w_{\mathcal{S}}}(y; x_A^0)$ is not achieved, one can apply a similar line of reasoning to ϵ-optimal controls. Since $y \in \mathcal{S} \cap \mathcal{T}$ is arbitrary, $\overline{v}(\mathbf{x^0}) \leq \inf_{y \in \mathcal{T}} \overline{w_{\mathcal{S}}}(y; x_A^0)$. The theorem hence follows. ■

Given this result, the problem of evaluating the upper value $\overline{v}(\mathbf{x}^0)$ becomes a problem of computing the minimum time-to-reach function $\overline{w_{\mathcal{S}}}$. However, the function $\overline{w_{\mathcal{S}}}$ also depends on the safe reachable set $\mathcal{S}$, thus $\overline{w_{\mathcal{S}}}$ and $\mathcal{S}$ must be computed simultaneously. We will demonstrate how to compute $\overline{w_{\mathcal{S}}}$ and $\mathcal{S}$ simultaneously along with the minimum time-to-capture function $\bar{t}$ in Section V. For now, it is important to observe that, as a consequence of Theorem 1, the problem of computing $\overline{v}(\mathbf{x}^0)$ in the high dimensional *joint state space* of the two agents reduces to computing $\overline{w_{\mathcal{S}}}$ in the lower dimensional state space of each individual player. This result not only speeds up the computation for the game with two agents, but also has important consequences for finding solutions when additional defending agents are introduced into the game.

B. Lower Value

This section studies the second value function defined in Equations (7). The information pattern embodied in this definition is that P_D is considering the worst possible situation that can occur to it and selects the control at the beginning. After P_D has decided on the control, P_A then chooses its optimized control with P_D's control revealed to P_A. Both players will then stick to the controls chosen. In general, computing $\underline{v}$ is not a trivial task. This is due to an asymmetry inherent in the games: unlike P_A, P_D does not have a target set which solely depends on the state of P_D. Instead, the goal of P_D is to prevent P_A from entering $\mathcal{T}$ through the possibility of capture via the set $\mathcal{A}$, which depends on the joint states of P_A and P_D. This interdependence makes the exact computation of the open-loop lower value difficult, since P_D must consider P_A's actions with respect to P_D's entire trajectory.

Therefore, we seek to find a computable lower bound to $\underline{v}$ using a particular choice of strategy. To simplify the task for P_D, we consider a strategy in which P_D moves to a particular location and then remains stationary at that location for the remainder of the game. Thus, instead of evaluating options over entire trajectories, P_D essentially considers locations where it may place

itself as an obstacle. The resulting value of this strategy will provide a lower bound on the value of the game. Furthermore, we show that, in some cases, the lower bound is equal to $\underline{v}$.

For this strategy, we must evaluate possible positions for P_D to place itself and the resulting payoff of the game for each position. For any candidate position, we must compute the arrival time of P_A to the target if P_D were to place itself at that location, serving as a static obstacle. Computing this for all points in the state space is computationally expensive, but the set of candidate points can be substantially reduced via the following observations. First, a candidate location is only viable if P_D can reach that position before P_A has a chance to end the game by arriving at the target. Second, a location for P_D can only affect P_A if P_D can arrive there before P_A, otherwise P_A can effectively treat that location as being empty. Once this set of candidate locations is found, the defender can choose as its target location the position which results in the longest arrival time for the attacker.

We will proceed by first examining the criterion that the defender must reach its target point before the attacker can finish the game.

Definition 4: Given an initial state $x_A^0 \in \Omega_{free}$ for P_A and a location $y \in \Omega_{free}$ for P_D, the minimum time-to-reach for P_A with respect to a stationary P_D is

$$\underline{t}(y; x_A^0) := \inf_{a \in \Sigma_A} \inf\{t \geq 0 \mid x_A(t) \in \mathcal{T}, x_A(0) = x_A^0, (x_A(s), y) \notin \mathcal{A}, \forall s \in [0, t]\}. \tag{14}$$

That is, suppose P_D remains at the point x throughout, then $\underline{t}(x; x_A^0)$ is the minimum time for P_A, starting at x_A^0, to reach the target without being captured.

Remark 1: From (14), it follows easily that $\underline{v}(\mathbf{x^0}) \geq \underline{t}(x_D^0; x_A^0)$. Namely, by remaining at its initial position x_D^0 for the entire duration of the game, P_D makes the set of points $\{x \in \Omega \mid (x, x_D^0) \in \mathcal{A}\}$ an obstacle and ensures that the game will last for at least $\underline{t}(x_D^0; x_A^0)$. Furthermore, if we restrict our attention to the reach-avoid game with point capture (i.e. P_A is captured if and only if the instantaneous positions of P_A and P_D coincide), then equality is in fact achieved: $\underline{v}(\mathbf{x^0}) = \underline{t}(x_D^0; x_A^0)$. To see this, let $x_A^*(\cdot)$ be a time-optimal path from x_A^0 to $\mathcal{T}$, without accounting for P_D. Then given that P_A knows the future actions of P_D, and capture can only be achieved if both P_A and P_D arrive at a point at the exact same time, P_A can simply make small perturbations as necessary to its optimal path $x_A^*(\cdot)$ in order to prevent its location from coinciding with P_D. In particular, by defining this perturbation in terms of an appropriate variation of x_A^* with some parameter $\epsilon > 0$, and letting ϵ tend to zero, the infimum of the time-to-reach is simply given by

the time-to-reach of the path x_A^*, regardless of the controls of P_D.

The function $\underline{t}(y; x_A^0)$ as defined above provides us with a naïve bound on the open-loop lower value. However, we can do better by expanding the set of possible points to consider for placement of P_D. Note that x_D^0 obviously fills the requirements of a candidate point that P_D can reach first, before P_A could end the game by entering into the target set $\mathcal{T}$. For any other candidate point y, we must ensure that the game does not terminate for any point along the path between x_D^0 and y. That is, we must ensure that there is no way for the attacker to end the game while the defender is enroute to its destination. We will call such paths *non-terminating*, since the game cannot be terminated while the defender is on such a path. In a similar manner to the safe-reachable set of the upper value game, we will find such paths by putting appropriate state constraints on the motion of P_D.

Definition 5: Given an initial state $x_D^0 \in \Omega_{free}$, a set $R \subset \Omega_{free}$, and a location $y \in \Omega_{free}$, the minimum time-to-reach for P_D with constraint R is

$$\underline{w_R}(y; x_D^0) := \inf_{d \in \Sigma_D} \inf\{t \geq 0 \mid x_D(t) = y, x_D(0) = x_D^0, x_D(t) \in R, \forall s \in [0, t]\}. \tag{15}$$

Intuitively, $\underline{w_R}(y; x_D^0)$ is the minimum time for P_D to reach y from x_D^0 by traveling along a path that is contained in R. Given that our objective is to find controls for P_D such that P_D can reach its desired destination without P_A first terminating the game, we will proceed to define, using the minimum time functions $\underline{t}$ and $\underline{w_R}$, what can be viewed as the largest set of states in Ω_{free} for which this is possible.

As a first step, for any initial state $\mathbf{x^0} = (x_A^0, x_D^0) \in \Omega_{free}^2$ such that $x_A^0 \notin \mathcal{T}$, let $\mathcal{E}$ be the collection of all sets $E \subseteq \Omega_{free}$ which satisfy $\underline{w_E}(y; x_D^0) < \underline{t}(y; x_A^0)$, $\forall y \in E$. Note that $\mathcal{E}$ is non-empty, as $\{x_D^0\}$ is an element of $\mathcal{E}$. We define a set $\mathcal{Z} \subseteq \Omega_{free}$ as the union of all sets in $\mathcal{E}$, namely $\mathcal{Z} := \bigcup_{E \in \mathcal{E}} E$. Then by a similar argument as in the proof of Proposition 1, one can show that $\mathcal{Z}$ belongs to $\mathcal{E}$ and hence is the maximal set in Ω_{free} for which the following inequality holds: $\underline{w_{\mathcal{Z}}}(y; x_D^0) < \underline{t}(y; x_A^0),\ \forall y \in \mathcal{Z}$.

Now to find our desired candidate set, we must find the points in $\mathcal{Z}$ that the defender can reach first, since the avoid set $\mathcal{A}_y := \{x \in \Omega \mid (x, y) \in \mathcal{A}\}$ for a fixed location y of P_D can only serve as a meaningful obstruction to P_A if P_D can arrive at y before the attacker. In order to do this, we will need to introduce the notion of a t-reachable set $\mathcal{R}_1(t)$ for P_A, defined as follows.

$$\mathcal{R}_1(t) := \{x \in \Omega_{free} \mid \exists a \in \Sigma_A, \exists s \in [0, t], x_A(s) = x, x_A(0) = x_A^0\} \tag{16}$$

In other words, this is the set of all states in Ω_{free} that P_A can reach within t time units. We can now use this definition, along with that of $\mathcal{Z}$, to find our set of candidate points as well as the target point for the defender, giving us the desired lower bound in the process.

Definition 6: Given a joint initial state $\mathbf{x^0} \in \Omega^2_{free}$, we define $\underline{\underline{v}}(\mathbf{x^0})$ as follows:

$$\underline{\underline{v}}(\mathbf{x^0}) := \begin{cases} \sup_{y \in \mathcal{Z}^*} \underline{t}(y; x_A^0), & x_A^0 \notin \mathcal{T} \wedge \mathbf{x^0} \notin \mathcal{A} \\ 0, & x_A^0 \in \mathcal{T} \wedge \mathbf{x^0} \notin \mathcal{A} \\ \infty, & \mathbf{x^0} \in \mathcal{A} \end{cases} \tag{17}$$

where

$$\mathcal{Z}^* := \{y \in \mathcal{Z} \mid \mathcal{A}_y \cap \mathcal{R}_1(T(y)) = \emptyset\},\ T(y) := \underline{w_{\mathcal{Z}}}(y; x_D^0). \tag{18}$$

Intuitively, $\underline{\underline{v}}(\mathbf{x^0})$ encodes the time it takes for the game to end if the defender picks the best point in $\mathcal{Z}^*$, reaches that point via a time-optimal path and thereafter stays at that point. From the above definition, it can be seen that $\mathcal{Z}^*$ contains all the useful points y in $\mathcal{Z}$, at which P_D can arrive before P_A, and along a path that ensures P_A cannot end the game before P_D reaches y. This leads to the following result.

Theorem 2: $\underline{\underline{v}}(\mathbf{x^0}) \leq \underline{v}(\mathbf{x^0})$, $\forall \mathbf{x^0} \in \Omega^2_{free}$.

Proof: For any $\mathbf{x^0} = (x_A^0, x_D^0) \in \Omega^2_{free}$ such that either $x_A^0 \in \mathcal{T}$ or $\mathbf{x^0} \in \mathcal{A}$, we have by the definition of $\mathcal{J}$ in (2) and $\underline{\underline{v}}$ in (17) that the result holds with equality.

Now consider an initial state $\mathbf{x^0} \in \Omega^2_{free}$ such that $x_A^0 \notin \mathcal{T}$ and $\mathbf{x^0} \notin \mathcal{A}$. Note that in such a case, $\mathcal{Z}^*$ is non-empty. In particular, x_D^0 is an element of $\mathcal{Z}^*$. Let $y \in \mathcal{Z}^* \subseteq \mathcal{Z}$, then by the preceding definition, $\underline{w_{\mathcal{Z}}}(y; x_D^0) < \underline{t}(y; x_A^0)$. Again for simplicity of argument, we assume that there exists $d^* \in \Sigma_D$ such that the path of P_D reaches the point y in time $T(y) = \underline{w_{\mathcal{Z}}}(y; x_D^0)$. Let $\tilde{d} \in \Sigma_D$ be a choice of control for P_D such that that $\tilde{d}(t) = d^*(t)$, $\forall t \in [0, T(y)]$, and $\tilde{d}(t) = 0$, $\forall t > T(y)$. Under this control, P_D reaches y at time $T(y)$ and remains stationary thereafter. Now suppose that there exists a control $a \in \Sigma_A$ for P_A such that $\mathcal{J}(\mathbf{x^0}, a, \tilde{d}) < \infty$ (otherwise the statement of the theorem holds trivially). Then there exists $t \geq 0$ such that $x_A(t) \in \mathcal{T}$ and $\mathbf{x}(s) \notin \mathcal{A}$, $\forall s \in [0, t]$. Let $t^* := \inf\{t \geq 0 \mid x_A(t) \in \mathcal{T}\}$. We claim that $x_A(s) \notin \mathcal{A}_y$, $\forall s \in [0, t^*]$. Indeed, if not, then there exists $s \in [0, t^*]$ such that $x_A(s) \in \mathcal{A}_y$. Now consider two cases. If $s \geq T(y)$, then $x_D(s) = y$. This implies that $\mathbf{x}(s) \in \mathcal{A}$. Thus, by definition of t^*, for every $t \geq 0$ such that $x_A(t) \in \mathcal{T}$, we have $\mathbf{x}(s) \in \mathcal{A}$ for some $s \in [0, t]$, and hence $\mathcal{J}(\mathbf{x^0}, a, \tilde{d}) = \infty$, which

is a contradiction. On the other hand, if $s < T(y)$, then $x_A(s) \in \mathcal{A}_y \cap \mathcal{R}_1(T(y))$, which implies that $y \notin \mathcal{Z}^*$. This is again a contradiction. Note that one can also arrive at the same conclusion using ϵ-optimal controls for P_D. The claim then follows, with the result that $a \in \Sigma_A$ is a choice of control such that $x_A(t^*) \in \mathcal{T}$ and $x_A(s) \notin \mathcal{A}_y$, $\forall s \in [0, t^*]$. By the definition of $\underline{t}$ in (14), this implies $\underline{t}(y; x_A^0) \leq t^* = \mathcal{J}(\mathbf{x^0}, a, \tilde{d})$. Since this inequality holds for every $a \in \Sigma_A$ such that the payoff is finite, we have $\underline{t}(y; x_A^0) \leq \inf_{a \in \Sigma_A} \mathcal{J}(\mathbf{x^0}, a, \tilde{d})$, and hence $\underline{t}(y; x_A^0) \leq \underline{v}(\mathbf{x^0})$. Given that $y \in \mathcal{Z}^*$ is arbitrary, the result of the theorem then follows. ■

With this understanding of $\mathcal{Z}^*$, we may select the control for P_D as moving to a point $x_D^* \in \arg\max_{x \in \mathcal{Z}^*} \underline{t}(x; x_A^0)$, whenever the supremum in (17) is achieved, and remaining stationary at x_D^* thereafter. Since P_D is guaranteed to arrive at x_D^* before P_A, $\mathcal{A}_{x_D^*}$ appears to P_A as though a permanent obstacle, forcing P_A to take at least $\underline{t}(x_D^*, x_A^0)$ time units to reach the target.

C. Relationship to Closed-loop Game

In this section, we briefly discuss the connection between the solutions to the open-loop games and the solution to the closed-loop reach-avoid game as described in Section III-B. First, recall that by (8), the open-loop upper and lower values provide bounds for the closed-loop upper and lower values. Thus, at initial conditions $\mathbf{x^0}$ such that the open-loop value functions coincide, namely $\underline{v}(\mathbf{x^0}) = \overline{v}(\mathbf{x^0})$, the closed-loop game locally has a value at $\mathbf{x^0}$. By the conclusions of Theorem 1 and Theorem 2, this is the case if

$$\sup_{y \in \mathcal{Z}^*} \underline{t}(y; x_A^0) = \inf_{y \in \mathcal{T}} \overline{w_{\mathcal{S}}}(y; x_A^0), \;\; x_A^0 \notin \mathcal{T}, \;\; \mathbf{x^0} \in \Omega^2_{free} \setminus \mathcal{A}. \tag{19}$$

Thus, at initial conditions such that (19) holds, one can compute the local value $V(\mathbf{x^0})$ of the game using the minimum time-to-reach functions $\underline{t}$ and $\overline{w_{\mathcal{S}}}$, without explicitly solving the HJI equation (6). Given that $\underline{t}$ and $\overline{w_{\mathcal{S}}}$ are defined over the individual player domain Ω_{free}, and V is defined over the joint domain Ω^2_{free}, this can result in significant computational savings.

Moreover, the closed-loop saddle-point solution can be potentially synthesized in terms of the open-loop saddle-point solution. In particular, suppose there exists $y_1^* \in \arg\min_{y \in \mathcal{T}} \overline{w_{\mathcal{S}}}(y; x_A^0)$ and $y_2^* \in \arg\max_{y \in \mathcal{Z}^*} \underline{t}(y; x_A^0)$, and optimal controls $\bar{a}$ and $\underline{d}$ with respect to $\overline{w_{\mathcal{S}}}(y_1^*; x_A^0)$ and $\underline{w_{\mathcal{Z}}}(y_2^*; x_D^0)$. Then it can be shown that the pair $(\bar{a}, \underline{d})$ forms a saddle-point solution to the open-loop game. They can also be interpreted as a saddle-point solution to the closed-loop game at $\mathbf{x^0}$ via non-anticipative strategies (γ_A^*, γ_D^*) such that $\gamma_A^*[d] := \bar{a}, \;\forall d \in \Sigma_D$ and $\gamma_D^*[a] := \underline{d}, \;\forall a \in \Sigma_A$.

We note that the conditions under which the open-loop values coincide with the HJI value are not particularly restrictive, and we are not limited to initial conditions where player actions are entirely independent of each other. It is certainly true that if the players begin the game in a way such that neither can affect the other's paths (e.g. if the players are initially far from each other with the attacker close to the target), then the open-loop and close-loop values will be equal. However, this is not a necessary condition, and as the computational example in Section VII shows, there are configurations where the open-loop values produce optimal results that require both P_A and P_D to modify their actions to account for the presence of the other player.

In addition, if the open-loop upper value does not coincide with the open-loop lower value, but the difference between the two values is within some small tolerance, then by (8), the difference between the closed-loop value and either of the two open-loop values will also be within this tolerance. For example, this would be the case if $\sup_{y\in\mathcal{Z}^*} \underline{t}(y;x_A^0) < \inf_{y\in\mathcal{T}} \overline{w_{\mathcal{S}}}(y;x_A^0) + \epsilon$, for some small $\epsilon > 0$. In such scenarios, we are still provided with a framework in which one can trade-off optimality with efficiency under some tolerance on the loss of optimality.

V. The Modified Fast Marching Method for Computing open-loop Values

Computing solutions to the open-loop upper and lower value games as described in the previous section requires finding $\overline{w_{\mathcal{S}}}$ and $\mathcal{S}$ for the upper value game and $\underline{w_{\mathcal{Z}}}$ and $\mathcal{Z}$ for the lower value game. In each case, the desired value is a minimum time-to-reach function constrained within some set. If the set is known *a priori*, the computation of the function values can be straightforwardly performed by obtaining the solution to a constrained optimal control problem. However, in both cases the set is also unknown and actually depends upon the time-to-reach function, so both must be computed together. To do this in an efficient manner, we utilize modified versions of the fast marching method (FMM) to compute both the desired function values and the corresponding sets simultaneously on a grid. This section briefly summarizes the FMM algorithm, and then presents the computations of $\overline{v}$ and $\underline{\underline{v}}$ via modified FMM.

A. The Fast Marching Method Algorithm

FMM [18], [19], [20], is a single-pass method used to numerically approximate the *Eikonal equation*

$$v(y)\|\nabla\phi(y)\| = 1, \quad y \in \Omega \setminus \mathcal{O}, \tag{20}$$

with Dirichlet boundary conditions $\phi = \infty$ on $\partial\Omega$ and $\phi(y) = b(y), \forall y \in \mathcal{O}$, for some closed set $\mathcal{O} \subset \Omega$, some boundary condition $b(y)$ and some known function $v(\cdot)$. For the system dynamics here, the Eikonal equation can be used to compute the minimum time-to-reach function from an initial condition or to a target set. That is, if $v(\cdot)$ is a positive scalar function representing the maximum speed, then the solution ϕ to equation (20) gives the minimum time-to-reach function starting from point y and ending in the region $\mathcal{O}$, or starting from $\mathcal{O}$ and ending at y.

The value ϕ in equation (20) is approximated by a grid function $\phi_{i,j}$ on a uniform 2-D Cartesian grid $\mathcal{G}$, where $\phi_{i,j} \approx \phi(y_{i,j})$, $y_{i,j} = (ih, jh) \in \mathcal{G}$ and h is the grid spacing. To simplify the treatment of the boundary conditions, assume that $\partial\Omega$ is well-discretized by the grid points $\partial\mathcal{G} \subset \mathcal{G}$. The solution to equation (20) is found via the finite difference approximation,

$$\frac{v(y_{i,j})}{h} \left[\begin{array}{l} (\phi_{i,j} - \min\{\phi_{i\pm1,j}, \phi_{i,j}\})^2 + \\ \quad (\phi_{i,j} - \min\{\phi_{i,j\pm1}, \phi_{i,j}\})^2 \end{array} \right]^{1/2} = 1 \tag{21}$$

The solution to equation (21) for $\phi_{i,j}$ can be found through the quadratic formula, and is referred to as the Eikonal update

$$\phi_{i,j} = \phi^*_{i,j}(\phi_{i\pm1,j}, \phi_{i,j\pm1}, v(y_{i,j})). \tag{22}$$

The scheme in equation (21) is consistent and stable, and converges to the viscosity solution of equation (20) as $h \to 0$ [21]. The FMM algorithm computes the solution to equation (21) for the entire grid by sequentially computing the value for each grid node in a particular order. The value for each node is computed only a small number of times, resulting in efficient computation of the value function even on large grids.

We will use the basic structure of FMM with some modifications to compute the appropriate values: $\overline{w_{\mathcal{S}}}$ for the upper value and $\underline{w_{\mathcal{Z}}}$ for the lower value. We now briefly outline the FMM algorithm. We keep track of two sets of nodes: Accepted and NarrowBand. Accepted is the set of nodes where the approximation values of the minimum time-to-reach function((i.e. $\overline{w_{\mathcal{S}}}$ or $\underline{w_{\mathcal{Z}}}$) are known. NarrowBand is the set of candidate nodes to be added to Accepted. In each iteration, one or more nodes from NarrowBand will have their values computed using the known node values in Accepted and then be removed from NarrowBand and added to Accepted. At the same time, additional nodes on the boundary of NarrowBand will be added to NarrowBand. The intuition for fast marching is a thin boundary that expands outward from a known initial point or set. The algorithm begins with the known boundary condition as the Accepted set, then expands

the set outward, computing the value for points near the boundary (points in the NarrowBand) until the values for all points on $\mathcal{G}$ have been found.

The key principle exploited by the FMM is that of "causality" [18], which states that the solution at a node only depends on adjacent nodes that have smaller values. This defines an ordering of the nodes, in increasing values of the minimum time-to-reach function (i.e. $\overline{w_{\mathcal{S}}}$ or $\underline{w_{\mathcal{Z}}}$). This ordering is realized by adding the smallest valued candidate from NarrowBand into Accepted at each step, until either all nodes are in Accepted or the Accepted set is enclosed by nodes from NarrowBand, each of which contains ∞ as the minimum time-to-reach value. This ordering allows the value for each node to be computed only a small number of times, while the node is within NarrowBand, thus speeding up computation.

B. Upper Value

We can now present the modified FMM in detail to compute the safe-reachable set $\mathcal{S}$ and the corresponding minimum time-to-reach function $\overline{w_{\mathcal{S}}}$. Our algorithm computes the solution to the following Hamilton-Jacobi-Bellman(HJB) equation in $\mathcal{S}$:

$$-\inf_{a\in\overline{B}_n}\{\nabla\overline{w_{\mathcal{S}}}(y;x_A^0)\cdot f_A(x_A)a\}=1 \tag{23}$$

along with the set $\mathcal{S}$ itself, using the boundary conditions

$$\overline{w_{\mathcal{S}}}(x_A^0;x_A^0)=0;\ \overline{w_{\mathcal{S}}}(y;x_A^0)=\infty,\ y\in\Omega\setminus\mathcal{S}. \tag{24}$$

We note that if the speed function f_A is identified with $v(\cdot)$ in the Eikonal equation (20), then the previous HJB equation is equivalent to the Eikonal equation, provided f_A is isotropic, which is true for the assumptions in this paper.

To compute $\bar{t}_{(i,j)}$, we define $\mathcal{A}^x := \{y\in\Omega \mid (x,y)\in\mathcal{A}\}$ as a slice of the avoid set at a fixed P_A location $x\in\Omega$. Intuitively, this corresponds to the set of points which allows P_D to capture P_A at x. Then $\bar{t}_{(i,j)}$ for any node $y_{i,j}$ can be found as $\inf_{y\in\mathcal{A}^x}\phi_D(y)$, where $\phi_D(y)$ is the unconstrained minimum time-to-reach function representing the shortest time to reach y starting from x_D^0.

In the following, we provide a schematic description of the modified FMM, with the numerical approximation of $\overline{w_{\mathcal{S}}}(y;x_A^0)$ at a grid node (i,j) denoted as $\overline{w_{\mathcal{S}}}^{(i,j)}$.

1) Initialize $\overline{w_{\mathcal{S}}}^{(i,j)}=\infty$, $\forall$ node $(i,j)\in\mathcal{G}$.

2) Compute $\bar{t}_{(i,j)}$ for every $(i,j) \in \mathcal{G}$ from $\inf_{y_{i,j} \in \mathcal{A}^x} \phi_D(y_{i,j})$ by using standard FMM to compute the defender time-to-reach ϕ_D for every node (i,j).
3) Set $\overline{w_{\mathcal{S}}}^{(i,j)} = 0$, if node (i,j) is the initial position of P_A, set it to be in Accepted.
4) For all nodes (i,j) adjacent to a node in Accepted, set $\overline{w_{\mathcal{S}}}^{(i,j)} = \overline{w_{\mathcal{S}}}^{*(i,j)}$ via the Eikonal update as per Equation (22) and label them to be in NarrowBand.
5) Choose a node (i,j) in Narrow-Band with the smallest $\overline{w_{\mathcal{S}}}^{(i,j)}$ value. If there is no node remaining or $\overline{w_{\mathcal{S}}}^{(i,j)} = \infty$, continue to Step 6. Otherwise, if $\overline{w_{\mathcal{S}}}^{(i,j)} \geq \bar{t}_{(i,j)}$, then set $\overline{w_{\mathcal{S}}}^{(i,j)} = \infty$. Place node (i,j) in Accepted, return to Step 4.
6) Return the two arrays containing the values of $\overline{w_{\mathcal{S}}}^{(i,j)}$ and $\bar{t}_{(i,j)}$. Compute $\mathcal{S}$ by taking all nodes (i,j) with finite $\overline{w_{\mathcal{S}}}^{(i,j)}$ values. Compute $\overline{v}$ by first intersecting $\mathcal{S}$ with $\mathcal{T}$ and pick the smallest $\overline{w_{\mathcal{S}}}^{(i,j)}$ in the intersection and designate the corresponding node (i,j) to be the final point in $\mathcal{T}$.

The algorithm terminates in a finite number of iterations, since the total number of nodes is finite. On a grid with M nodes, the complexity is $O(M \log M)$ and the algorithm naturally extends to three or higher dimensions [19].

The algorithm presented above differs from the standard FMM in the addition of Step 5, which rejects points that are reachable by P_D in less time than P_A. To see why this modification is sufficient, suppose that, at the start of Step 4 of the current iteration, all Accepted nodes have the correct $\overline{w_{\mathcal{S}}}^{(i,j)}$ values. Suppose also that node (i,j) is the smallest element in NarrowBand ordered by $\overline{w_{\mathcal{S}}}$ value and $\overline{w_{\mathcal{S}}}^{(i,j)} \geq \bar{t}_{(i,j)}$. Since $\overline{w_{\mathcal{S}}}^{(i,j)}$ is computed using neighboring Accepted nodes (which are assumed to have the correct values), this implies that an optimal path in $\mathcal{S}$ would take longer than $\bar{t}_{(i,j)}$ to reach (i,j). This in turn implies that P_D will capture P_A should P_A attempt to reach (i,j). Therefore, $\overline{w_{\mathcal{S}}}^{(i,j)} = \infty$, since (i,j) cannot be safe-reachable. On the other hand, if $\overline{w_{\mathcal{S}}}^{(i,j)} < \bar{t}_{(i,j)}$, there is a safe-reachable path in $\mathcal{S}$ that takes less than $\bar{t}_{(i,j)}$ time to reach (i,j). Thus, right before Step 6, $\overline{w_{\mathcal{S}}}^{(i,j)} < \infty$ if and only if $(i,j) \in \mathcal{S}$. Since all Accepted nodes are initially correct (Step 3), the above argument holds inductively until all Accepted nodes are computed correctly. The result of the computation above is a grid $\mathcal{G}$ with nodes where $\overline{w_{\mathcal{S}}}^{(i,j)}$ approximates the value of $\overline{w_{\mathcal{S}}}$ for each node (i,j). The safe-reachable set $\mathcal{S}$ can then be approximated as $\mathcal{S} \approx \{(i,j) \mid \overline{w_{\mathcal{S}}}^{(i,j)} < \infty\}$.

Remark 2: Note that with more general dynamics that are not isotropic, the FMM is not directly applicable. However a similar causality-ordering procedure is possible via the Ordered

Upwind Method (OUM) [22], allowing the method to be eventually extended to anisotropic [22] and non-holonomic [23] dynamics.

C. Lower Value

We use a different modified FMM algorithm to compute $\underline{v}$, the bound on the open-loop lower value. Computing $\underline{v}$ has some conceptual similarity to the computation of $\overline{v}$, but requires some extra steps. Here, instead of computing $\overline{w_{\mathcal{S}}}, \bar{t}$, and $\mathcal{S}$, we are interested in computing $\underline{t}(x; x_A^0)$, $\underline{w_{\mathcal{Z}}}(x; x_D^0)$, $\mathcal{Z}$ and $\mathcal{Z}^*$. The computation of $\underline{t}$ requires more computation than that of $\overline{w_{\mathcal{S}}}$ and $\bar{t}$, as for each point in $\mathcal{Z}^*$ a separate FMM computation must be performed to find P_A's arrival time at the target. However, by using a modified FMM method, we are able to compute the $\underline{t}$, $\underline{w_{\mathcal{Z}}}$, and $\mathcal{Z}$ simultaneously and avoid computing $\underline{t}$ unnecessarily, reducing computation time.

In the following, we denote the numerical approximation of the functions $\underline{t}(y; x_A^0)$ and $\underline{w_{\mathcal{Z}}}(y; x_D^0)$ as $\underline{t}_{(i,j)}$ and $\underline{w_{\mathcal{Z}}}^{(i,j)}$, respectively. The notation $\mathcal{A}_{(i,j)}$ is used to denote the slice $\mathcal{A}_{y_{i,j}} = \{x \in \Omega \mid (x, y_{i,j}) \in \mathcal{A}\}$ of the avoid set at a fixed P_D location $y_{i,j}$. The sets Accepted and NarrowBand have the same meaning as before: Accepted represents the set of nodes whose corresponding $\underline{w_{\mathcal{Z}}}^{(i,j)}$ values have been computed. NarrowBand represents the set of nodes that are about to be added to Accepted. $W_{(i,j)}$ and $T_{(i,j)}$ are two arrays which are used to store values for $\underline{w_{\mathcal{Z}}}^{(i,j)}$ and $\underline{t}_{(i,j)}$, respectively. The algorithm then proceeds as follows:

1) Initialize $W_{(i,j)} = \infty$, $\forall$ node $(i, j) \in \mathcal{G}$.
2) $W_{(i,j)} = 0$, if (i, j) is the initial position of P_D, and set (i, j) to be in Accepted.
3) For each node (i, j) adjacent to a node in Accepted, run the Eikonal update as described in (22) to obtain the $\underline{w_{\mathcal{Z}}}^{(i,j)}$ value for (i, j). Set $W_{(i,j)} = \underline{w_{\mathcal{Z}}}^{(i,j)}$ for all (i, j) adjacent to a node in Accepted and place these nodes in NarrowBand.
4) Choose a node (i, j) in NarrowBand with the smallest $W_{(i,j)}$ value. If there is no node remaining or if it is equal to ∞ return W and T and continue to Step 5. Otherwise compute $\underline{t}_{(i,j)}$ by doing the following: treat $\mathcal{A}_{(i,j)}$ as an obstacle, compute $\underline{t}_{(i,j)}$ using standard FMM. Set $T_{(i,j)}$ to be $\underline{t}_{(i,j)}$ and put (i, j) into Accepted. If $W_{(i,j)} < \underline{t}_{(i,j)}$, set $W_{(i,j)}$ to $\underline{t}_{(i,j)}$; otherwise set $W_{(i,j)}$ to be ∞. Now return to Step 3.
5) Find a node (i, j) with the largest $T_{(i,j)}$ value, record this value in a variable M and set $T_{(i,j)}$ to be $-\infty$. Compute the reachable set $\mathcal{R}_1(M)$ using FMM and test if it has nonempty intersection with $\mathcal{A}_{(i,j)}$. If so, return to Step 5. Otherwise, set $\underline{v} = M$ and return $\underline{v}$.

Note that instead of computing $\underline{t}$ at all points, we compute it "on the fly" in the sense that we stop immediately when the smallest $W_{(i,j)}$ in Narrowband is equal to ∞. This results in a significant amount of computational savings, and is justified by the fact that if $W_{(i,j)} \geq \underline{t}_{(i,j)}$, then (i,j) is not in $\mathcal{Z}$, which means $W_{(i,j)}$ should be ∞ by definition of the function $\underline{w_{\mathcal{Z}}}$. Later, if we want to extract $\mathcal{Z}$, we need only look at the nodes that have finite values.

D. Extracting Control Inputs

Given the value function approximations described above, we can also extract the optimal controls and paths corresponding to these value functions.

For the upper value game, if $\overline{v} < \infty$, the next step is to identify P_A's optimal control $\bar{a} \in \arg\min_{a\in\Sigma_A} \sup_{d\in\Sigma_D} \mathcal{J}(\mathbf{x^0}, a, d)$. Note that $\bar{a}$ is not necessarily unique, but all such inputs yield the same value $\overline{v}$. Given the result of Theorem 1, one can interpret $\bar{a}$ as a control input for P_A realizing a time-optimal safe-reachable path from x_A^0 to a final point $x_A^f \in \arg\inf_{y\in\mathcal{T}\cap\mathcal{S}} \overline{w_{\mathcal{S}}}(y; x_A^0)$. In fact, this final point is returned in Step 5 of our algorithm. Thus, $\overline{w_{\mathcal{S}}}$ can be used to extract the optimal control where it is smooth. In particular, given the dynamics in (1) and the assumption on the control set Σ_A, one can verify that the optimal control for P_A at a location $y \in \Omega_{free}$ where the value function $\overline{w_{\mathcal{S}}}$ is differentiable is given by $\mu(y) = -\frac{\nabla \overline{w_{\mathcal{S}}}(y;x_A^0)}{\|\nabla \overline{w_{\mathcal{S}}}(y;x_A^0)\|}$. In general, the value function $\overline{w_{\mathcal{S}}}$ may not be differentiable at every point $y \in \Omega_{free}$. Non-differentiable points typically correspond to locations at which the optimal input is not unique.

The optimal path $x_A^*(\cdot)$ of P_A can be then computed using $\mu(y)$ by solving the ordinary differential equation

$$\dot{x}_A^*(t) = f_A(x_A^*(t))\mu(x_A^*(t)) \tag{25}$$

from $t = \overline{w_{\mathcal{S}}}(x_A^f; x_A^0)$ to $t = 0$ backward in time, with the terminal condition $x_A^*(\overline{w_{\mathcal{S}}}(x_A^f; x_A^0)) = x_A^f$. Due to the construction of $\overline{w_{\mathcal{S}}}$, this results in $x_A^*(0) = x_A^0$. The realization of the optimal control is then given by $\bar{a}(t) = \mu(x_A^*(t))$. We note that the solution to (25) can be numerically approximated by standard ODE solvers.

For the open-loop lower value case, we can find the final point x_D^f for P_D by selecting the point within $\mathcal{Z}^*$ with the lowest $\underline{t}$ value. Then the optimal input map $\mu(\cdot)$ for P_D can be found using $\underline{w_{\mathcal{Z}}}(y; x_D^0)$ in a similar fashion $\overline{w_{\mathcal{S}}}(y; x_A^0)$, resulting in an optimal control $\underline{d}$ and an optimal path $x_D^*(\cdot)$.

VI. Generalization to Open-Loop Multi-Player Games

In this section, we generalize the two-player open-loop game to multi-player games. As mentioned before, solving HJI equations in high dimensions is in general an intractable problem, and the state space typically scales exponentially with the number of players. However, for certain game formulations with multiple players, the open-loop framework and the modified FMM methods provide a fast way to compute the open-loop values and controls that scales linearly with the number of players. Due to the inherent difference between the open-loop upper value computation and the open-loop lower value computation, we will provide two open-loop multi-player game formulations corresponding to the upper and lower value, each leveraging techniques developed previously.

A. The Open-Loop Upper Value for Multi-Player Games

Suppose that there are N attackers P_A^1,...,P_A^N with initial conditions x_{A1}^0,...,x_{AN}^0, and M defenders P_D^1,...,P_D^M, with initial conditions x_{D1}^0,...,x_{DM}^0, each having its own decoupled dynamics. We assume that each defender has its own capture set, and use $\mathcal{A} \subset \mathbb{R}^{N+M}$ to encode capture conditions in which at least one defender captures at least one attacker. For convenience, we also define the partial avoid set $\mathcal{A}^{ij}(x)(1 \leq i \leq N, 1 \leq j \leq M)$ to be the set of all states that P_A^i needs to avoid if P_D^j stays at state x. We use x_{Ai} and a^i to denote the state and the control of P_A^i, respectively. x_{Dj} and d^j are similarly defined for defenders. $\mathbf{x^0} = (x_{A1}^0, ..., x_{AN}^0; x_{D1}^0, ..., x_{DM}^0)$ is the joint initial state and $\mathbf{x}$ is the joint state of all agents.

In the problem that we consider, the goal for the attackers is to minimize the time needed for every (and hence the last) attacker to reach the target set $\mathcal{T}$, while the defenders are trying to maximize this time. We note that as soon as an attacker reaches the target, it is removed from the game and hence safe thereafter. Observe that under this formulation, if the defenders can prevent at least one attacker from reaching the target, then the total arrival time for the attackers is infinity. In this subsection, we adopt the information pattern that is conservative towards attackers. Namely, the controls of every attacker will be revealed to every defender. Then all defenders will jointly optimize their controls with this knowledge.

Under this formulation, the payoff of the game is given by

$$\mathcal{J}_{multi}(\mathbf{x^0}; a^1, ..., a^N; d^1, ..., d^M) = \inf \{t \geq 0 \mid \forall 1 \leq i \leq N, \exists t_i \leq t, x_{Ai}(t_i) \in \mathcal{T}, \\ \mathbf{x}(s) \notin \mathcal{A}, \forall s \in [0, t]\}. \tag{26}$$

The multi-player open-loop upper value is then

$$\overline{v}_{multi}(\mathbf{x^0}) = \inf_{a^i \in \Sigma_A} \sup_{d^j \in \Sigma_D} \mathcal{J}_{multi}(\mathbf{x^0}; \{a^i\}_{1 \leq i \leq N}; \{d^j\}_{1 \leq j \leq M}). \tag{27}$$

In general, what makes a multi-agent problem difficult is that simply adopting the optimal control for each agent does not necessarily yield a global optimal solution for the whole system. However, we shall see here that in this case the global optimal solution in fact decentralizes into each attacker's local optimal control, thus eliminating the need for centralized coordination.

First, observe that we can define for each attacker a function $\overline{w_{R_i}}(x; x^0_{Ai})$ similarly as in Definition 3 and for each defender a function $\overline{t}_j(y; x^0_{Dj})$ similarly as in Definition 2. Second, given that all defenders cooperate to delay the attackers from reaching the target, we can assign a function $\overline{t}(y; \{x^0_{Dj}\}_{1 \leq j \leq M}) := \min_{1 \leq j \leq M} \overline{t}_j(y; x^0_{Dj})$ to the whole defending side. For each attacker P^i_A, we define the corresponding safe reachable set $\mathcal{S}_i$ to be the maximal set such that the following is satisfied: $\overline{w_{\mathcal{S}_i}}(y; x^0_{Ai}) < \overline{t}(y; \{x^0_{Dj}\}_{1 \leq j \leq M}),\ \forall y \in \mathcal{S}_i$.

The following result generalizes Theorem 1 to the multi-player case.

Theorem 3:

1) $\overline{v}_{multi}(\mathbf{x^0}) < \infty$ if and only if $\mathcal{S}_i \cap \mathcal{T} \neq \emptyset$, $\forall i = 1, 2, ..., N$;
2) $\overline{v}_{multi}(\mathbf{x^0}) = \max_{1 \leq i \leq N} \inf\{\overline{w_{\mathcal{S}_i}}(y; x^0_{Ai}) \mid y \in \mathcal{T}\}$.

Proof: The first part can be inferred in a straightforward manner from the formulation of the game. For the second part, first observe that if $\overline{v}_{multi}(\mathbf{x^0}) = \infty$, then by part 1) of the theorem, there exists $i \in \{1, 2, ..., N\}$ such that $\inf\{\overline{w_{\mathcal{S}_i}}(y; x^0_{Ai}) \mid y \in \mathcal{T}\} = \infty$. Thus, $\overline{v}_{multi}(\mathbf{x^0}) = \max_{1 \leq i \leq N} \inf\{\overline{w_{\mathcal{S}}i}(y; x^0_{Ai}) \mid y \in \mathcal{T}\} = \infty$.

Now consider the case in which $\overline{v}_{multi}(\mathbf{x^0})$ is finite. Also by part 1), we know that $\mathcal{S}_i \cap \mathcal{T} \neq \emptyset$, $\forall i = 1, 2, ..., N$. This implies that for every P^i_A, $t^*_i := \inf\{\overline{w_{\mathcal{S}}i}(y; x^0_{Ai}) \mid y \in \mathcal{T}\} < \infty$. Let $t^* := \max_i t^*_i$. Then clearly, $\overline{v}_{multi}(\mathbf{x^0}) \leq t^*$. Moreover, there exists a collection of controls $\{a^i\}_{1 \leq i \leq N}$ such that $\sup_{d^j \in \Sigma_D} \mathcal{J}_{multi}(\mathbf{x^0}; \{a^i\}_{1 \leq i \leq N}; \{d^j\}_{1 \leq j \leq M}) < \infty$. For any such controls, there exists $t \geq 0$ such that for every $i = 1, 2, ..., N$, there exists $\tau_i \leq t$ such that $x_{Ai}(\tau_i) \in \mathcal{T}$ and $\mathbf{x}(s) \notin \mathcal{A}, \forall s \in [0, t]$. Let $\tau^*_i := \inf\{t \geq 0 \mid x_{Ai}(t) \in \mathcal{T}\}$, then it can be checked that

$t_i^* \leq \tau_i^*$, $\forall i = 1, 2, .., N$ and $\mathcal{J}_{multi}(\mathbf{x^0}; \{a^i\}_{1\leq i\leq N}; \{d^j\}_{1\leq j\leq M}) = \max_i \tau_i^*$, $\forall d^j \in \Sigma_D$. Given that this holds for every $\{a^i\}_{1\leq i\leq N}$ such that the payoff is finite, we have $t^* \leq \overline{v}_{multi}(\mathbf{x^0})$. The second part of the theorem then follows. ■

Remark 3: We note that under this formulation, it is not necessary for every attacker to use an optimal control to ensure that the upper value is achieved. In particular, suppose that it takes attacker P_1^A the longest to reach the target set, namely $t_1^* \in \arg\max_i t_i$ in the proof above. Then attackers other than P_1^A can use suboptimal controls while still achieving the same overall payoff, as long as the suboptimal controls do not yield a time-to-reach longer than t_1^*. We note that under this formulation, attackers other than P_1^A can still take suboptimal controls while still achieving the same overall payoff as long as the suboptimal controls do not yield time longer than $\inf\{\overline{w_{\mathcal{S}i}}(y; x_{Ai}^0) \mid y \in \mathcal{T}\}$.

B. The Open-Loop Lower Value for Multi-Player Games

Under the open-loop lower value information pattern, an unfortunate difference from the open-loop upper value is that the optimal controls for the defending agents do not decentralize, and the defenders must coordinate their control inputs jointly. This distinction is again created because unlike the attacking agents, whose common goal is to enter a stationary target set, the defenders' goals depend upon where the attackers are at each time instant as well as where other defending agents are. This restricts the type of generalization which can be realized.

Adopting the notations introduced previously, we will consider a scenario with N attacking agents and a single defending agent. The attacking agents and the defending agent both have the same objectives as stated in the previous subsection. The generalized open-loop lower value is then given by $\underline{v}_{multi}(\mathbf{x^0}) = \sup_{d^1\in\Sigma_D} \inf_{a^i\in\Sigma_A, 1\leq i\leq N} \mathcal{J}_{multi}(\mathbf{x^0}; \{a^i\}_{1\leq i\leq N}; d^1)$.

Each attacking agent P_i^A has its own $\underline{t}^i(x; x_{Ai}^0)$ function, as introduced in Definition 14 and the single defender has the function $\underline{w_R}^i(x; x_{D1}^0)$, as introduced in Definition 15. Therefore, for each attacking agent i and the defending agent, we can associate a corresponding set $\mathcal{Z}_i^*$ similarly defined as in equation (18), and a function $\underline{\underline{v}}^i(x_{Ai}^0, x_{D1}^0)$ similarly defined as in equation (17), irrespective of all the other attacking agents. The following generalization can be then shown using a similar line of reasoning as presented in the proof to Theroem 2.

Theorem 4: $\underline{v}_{multi}(\mathbf{x^0}) \geq \max_{1\leq i\leq N}\{\underline{\underline{v}}^i(x_{Ai}^0, x_{D1}^0)\}$

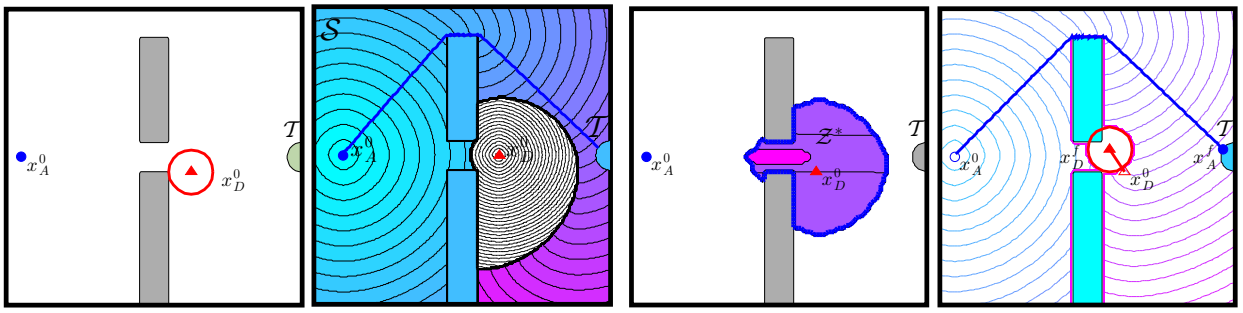


Fig. 2. **Example scenario showing (a) the initial conditions of the players, the target set $\mathcal{T}$, and game domain, (b) the set $\mathcal{Z}$ with contours plotted for $\underline{t}$ within, and (c) the trajectories taken for each player, with equal time-to-reach contours for player 1 plotted.**

VII. Simulation Results

In this section we present simulation results for games played under the two open-loop information patterns. We first provide a simple scenario involving two players such that the open-loop game solutions coincide with the closed-loop game solutions. We then demonstrate the utility of the method in more complicated game scenarios with multiple players.

A. 1 vs. 1 Example

We first present an example of a game with a single attacker and defender to illustrate solutions to both the upper and lower value games, and to show a situation in which both game solutions coincide. The game configuration is illustrated in Figure 2(a) and consists of a square region with the target on the right and two central obstacles with a small space between them. In this scenario, P_A begins the game on the far left, with P_D to the right of the obstacles. Here P_D has a maximum speed of 0.25 and P_A has a maximum speed of 1. Intuitively, the slower defender will not be able to successfully catch the attacker, and thus the best option for the defender is to block the central opening, forcing the attacker to move over the top of the obstacles.

The game results reflect this intuition. Figure 2(b) shows the optimal trajectory for P_A computed in the upper value game. $\mathcal{S}$ denotes the set of points that P_A can safely reach before P_D, given that P_D has knowledge of P_A's inputs. In this case, $\mathcal{S}$ clearly does not contain a clear path through the central opening, thus for any path passing through this opening there are points at which P_D can reach before P_A. Therefore P_A is forced to move up over the obstacles.

The computations were performed on a 400x400 grid using compiled C++ code in Matlab, on a Macbook Pro laptop with a 2.4 GHz Intel Core i7 processor and 8 GB RAM. Total computation time required was 0.5s.

Figure 2(c) and (d) show the computation of the solutions for the lower value game. Figure 2(c) shows $\mathcal{Z}^*$ and contour plots for $\underline{t}$ within $\mathcal{Z}^*$. Recall that $\underline{t}(x)$ represents the time taken to reach the target if P_D is stationary at x. Obviously, $\underline{t}$ increases in the opening between the obstacles, as those positions force P_A to move up around the obstacles in order to reach the target. The final paths of the two players in the lower value game are shown in Figure 2(d): as expected P_D moves to block the opening, as that results in the maximum $\underline{t}$. The computations were performed on a 100x100 grid using compiled C++ code in Matlab, on a Macbook Pro laptop with a 2.4 GHz Intel Core i7 processor and 8 GB RAM. Total elapsed time for computing $\mathcal{Z}^*$ and associated $\underline{t}$ was 1.25 seconds. The computation for the lower value bound is much more time-intensive than the upper value, as $\underline{t}$ has to be computed independently for every point in $\mathcal{Z}^*$.

This game illustrates an example of a situation where the upper and lower value games coincide, and hence the open-loop value is equal to the closed-loop value. This highlights one of the advantages of the open-loop game formulation: by quickly computing the upper and lower value solutions, it is possible to quickly verify whether the solution found is in fact a local closed-loop saddle-point pair, in which case it becomes unnecessary to carry out HJI computations on the joint state space. For comparison, an HJI solution (not pictured) for an identical scenario required 40s for a four dimensional grid, 40 nodes wide in each dimension. Thus, for these initial conditions at least, the open-loop solutions exhibited substantial computational savings over the HJI computation for the same result.

B. Multi-Player Examples

This section demonstrates the open-loop game formulation applied to more complex games and games with multiple players on each side. The game scenarios are computed on a 2-D map of size 400^2 pixels, representing the UC Berkeley campus (see Figure 3). To represent the varying terrain in the area, the map data $f_{i,j}$, $i, j \in 1, \ldots, 400$ represent the fraction of the players' maximum speeds that are allowed at each point on the map, with 1 being maximum speed and 0 representing impenetrable obstacles. They have values 0 in the buildings and 1 in the walkways; other regions have intermediate values in $(0, 1)$ selected in proportion to the

estimated density of vegetation.

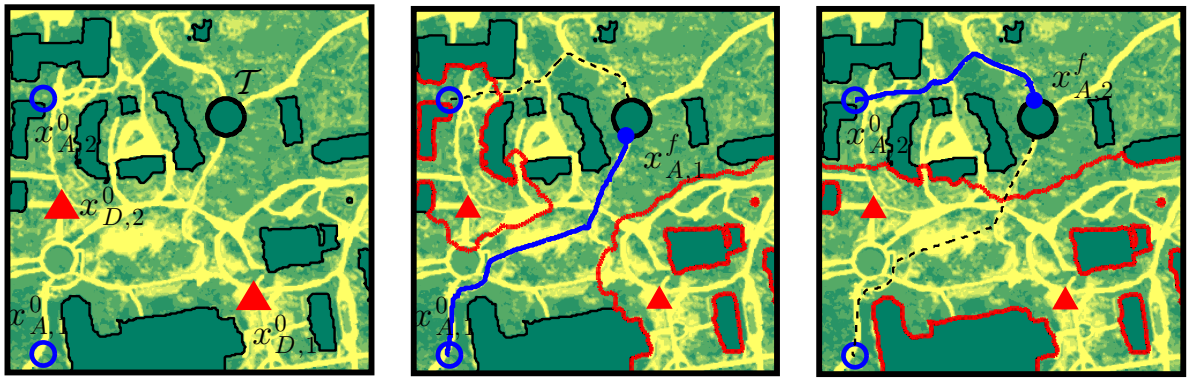
Two sets of simulations are presented here. The first set of simulations show an upper value game with two attackers and two defenders, as illustrated in Figure 4. In this scenario, P_D^1 and P_D^2 have the same maximum speed of $f_{i,j}$ at each map node, while P_A^1 has maximum speed of $3f_{i,j}$ and P_A^2 has maximum speed of $2f_{i,j}$. The computed optimal paths for each player are shown in Figures 4(a) and (b). Figure 4(a) shows the optimal trajectory for P_A^1, highlighted in solid black, with solid red lines delineating the boundary between points where P_A^1 can reach before any defender. In this scenario, the defenders are forbidden from moving beyond the magenta boundary line to the left. P_A^2's trajectory is denoted here by a lighter, dashed line. Similarly, Figure 4(b) shows the game from P_A^2's perspective. In each case, the upper value game computation is able to quickly compute an optimal path for each player, considering the possible actions of both defenders. The second set of simulations are shown in Figure 5 for the lower value game, with two attackers and one defender. The maximum speeds for $x_{A,1}^0$, $x_{A,2}^0$ and x_D^0 are $0.9f_{i,j}$, $0.8f_{i,j}$ and $0.25f_{i,j}$ respectively. Due to the nature of the modified FMM algorithm, the obstacles and complexity of the varying speed profiles is naturally accounted for.

All computations were performed on a desktop computer with 3.33 GHz Intel Core-II duo processors. The code was implemented in C using the Matlab MEX compiler to allow function calls within Matlab. Both tests were completed (including the computation of optimal paths) in less than 0.5 seconds each. Note that the implementation and the computational efficiency are independent of variations in the speed function, and the addition of an extra defender did not increase the computation time substantially.

VIII. Conclusion and Future Work

The open-loop games and solutions presented in this paper can quickly generate solutions for adversarial games with multiple players. Formulating the problem as an open-loop reach-avoid game allows the value functions to be characterized via an HJB equation directly in the state space of individual players, as opposed to the joint state space of all players. Compared with the HJI approach, this results in considerable computational savings, at the expense of conservatism with respect to the optimality of the solutions. Nevertheless, the speed with which solutions can be found can be a major advantage over the HJI solutions, in particular allowing solutions to be found in real-time and with multiple players. As the numerical results demonstrated, the open-

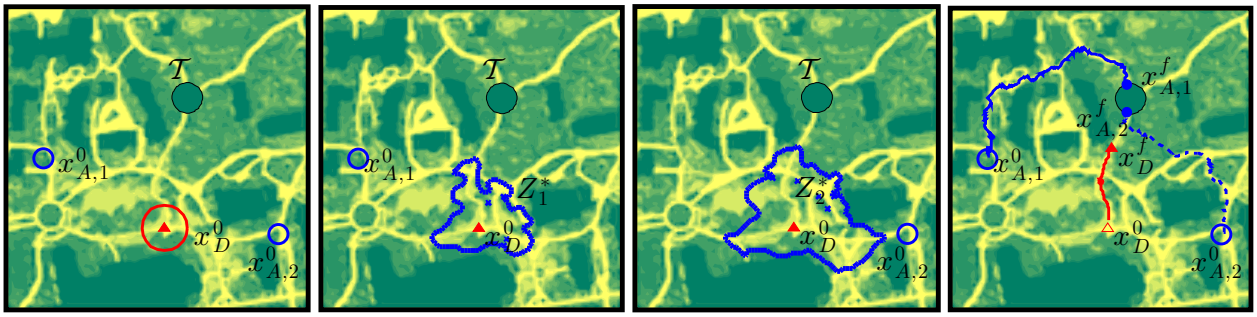
Fig. 3. **A segment of the UC Berkeley campus used for the simulations (map image courtesy of maps.google.com).**



(a) (b) (c)

Fig. 4. **Simulation of a game between two attackers and two defenders for the upper value game, showing (a) the initial positions of the players and the target region, (b) the path taken by attacker 1(solid black line) and the boundary of its safe-reachable set with respect to the two defenders (solid red lines), and (c) the path taken by attacker 2 (solid black line) and the boundary of its safe-reachable set with respect to the defenders (solid red lines).**

loop solutions are efficient, accurate, and readily adaptable to complicated domain geometry and inhomogeneous agent speeds. In particular, for the upper value game, where the open-loop solution does exist, the attacker path found is guaranteed to be safe from capture by the defenders. The FMM algorithms can be used to quickly check if an open-loop solution exists. If it does, then this solution can be used directly, with a guarantee of safety, without resorting to other



(a) (b) (c) (d)

Fig. 5. **Simulation of a game between two attackers and one defender for the lower value game, showing (a) the initial positions of the players and the target region, (b) $\mathcal{Z}^*$ with respect to the first attacker (solid blue line), (c) $\mathcal{Z}^*$ with respect to the second attacker (solid blue line), and (d) the paths taken by all of the players.**

sub-optimal solutions that may not have any provable properties.

Future work will focus on mitigating the conservatism inherent in the open-loop formulations. In the upper value game, this conservatism means that for certain initial conditions, the open-loop solution value will be greater the HJI value, requiring more time to arrive at a target. Similarly the lower value game may predict a smaller arrival time. More research will need to be conducted on the best that can be done in such situations, as well as in situations where a player cannot win if the opponents play optimally, but may exploit some sub-optimal opponent actions to eventually achieve victory. One direction is to use the ability to compute solutions in real-time to enable an iterative scheme to recompute the solutions in an receding horizon manner, using feedback to reduce the conservatism as the game progresses. Another possible direction, also enabled by the fast computation, would be to incorporate sensing to update the domain as the game is played. These directions have the potential to expand the kind of games that are currently addressable.

REFERENCES

[1] R. Takei, H. Huang, J. Ding, and C. Tomlin, "Time-optimal multi-stage motion planning with guaranteed collision avoidance via an open-loop game formulation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, 2012.

[2] Z. Zhou, R. Takei, H. Huang, and C. Tomlin, "A general, open-loop formulation for reach-avoid games," in *Proceedings of the IEEE International Conference on Decision and Control*, Maui, Hawaii, 2012.

[3] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *International Journal of Robotics Research*, vol. 17, no. 7, p. 760, 1998.

[4] T. Fraichard and H. Asama, "Inevitable collision states - a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[5] J. Van Den Berg and M. Overmars, "Planning time-minimal safe paths amidst unpredictably moving obstacles," *International Journal of Robotics Research*, vol. 27, no. 11-12, p. 1274, 2008.

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

[7] L. C. Evans and P. E. Souganidis, "Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations," *Indiana University Mathematics Journal*, vol. 33, no. 5, pp. 773–797, 1984.

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

[9] 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, 2005.

[10] M. Falcone and R. Ferretti, "Semi-Lagrangian schemes for Hamilton-Jacobi equations, discrete representation formulae and Godunov methods," *Journal of Computational Physics*, vol. 175, no. 2, pp. 559–575, 2002.

[11] J. Ding, J. Sprinkle, S. S. Sastry, and C. J. Tomlin, "Reachability calculations for automated aerial refueling," in *Proceedings of the IEEE International Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 3706–3712.

[12] 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 *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011.

[13] 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.

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

[15] P. Varaiya, "On the existence of solutions to a differential game," *SIAM Journal on Control*, vol. 5, no. 1, pp. 153–162, 1967.

[16] R. J. Elliott and N. J. Kalton, "The existence of value in differential games," in *Memoirs of the American Mathematical Society*, 1972, no. 126.

[17] M. Bardi and I. Capuzzo-Dolcetta, *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*, ser. Systems & Control: Foundations & Applications. Boston, MA: Birkhäuser, 1997, with appendices by Maurizio Falcone and Pierpaolo Soravia.

[18] J. A. Sethian, "Fast marching methods," *SIAM Review*, vol. 41, no. 2, pp. 199–235, 1999.

[19] ——, *Level set methods and fast marching methods*, 2nd ed., ser. Cambridge Monographs on Applied and Computational Mathematics. Cambridge, UK: Cambridge University Press, 1999, vol. 3.

[20] M. Falcone, "Fast marching methods for front propagation," November 2007, lecture notes at "Introduction to Numerical Methods for Moving Boundaries".

[21] E. Rouy and A. Tourin, "A viscosity solutions approach to shape-from-shading," *SIAM Journal of Numerical Analysis*, vol. 29, no. 3, pp. 867–884, 1992.

[22] J. A. Sethian and A. Vladimirsky, "Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms," *SIAM Journal of Numerical Analysis*, vol. 41, no. 1, pp. 325–363, 2003.

[23] R. Takei, R. Tsai, H. Shen, and Y. Landa, "A practical path-planning algorithm for a vehicle with a constrained turning radius: a Hamilton-Jacobi approach," in *Proceedings of the IEEE American Control Conference*, July 2010.