# Efficient Algorithms for Open-Loop Reach-Avoid Games☆

Zhengyuan Zhou[a], Jerry Ding[b], Haomiao Huang[c], Ryo Takei[d], Claire Tomlin[e]

[a]*Department of Electrical Engineering, Stanford University, Stanford, CA, 94305, USA*
[b]*Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, 94720, USA*
[c]*Kuna Systems, CA, USA*
[d]*Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, 94720, USA*
[e]*Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, 94720, USA*

**Abstract**

We consider a multi-player differential game, referred to as a reach-avoid game, in which one set of attacking players attempts to reach a target while avoiding both obstacles and capture by a set of defending players. Unlike pursuit-evasion games, in this reach-avoid game one set of players must not only consider the other set of players, but also the target. This complexity makes finding solutions to such games 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, thus enabling efficient computation of 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 numerical algorithms based upon modified fast-marching methods (FMM) for 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.

*Keywords:* differential games, Fash Marching Methods, open-loop

## 1. Introduction

In a reach-avoid game, one set of players (attackers) attempts 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 (defenders). Such games encompass a large number of robotics and control applications. For example, many safe motion-planning problems (see [16] and the references therein) 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 possibly adversarial agents.

Reach-avoid games considered here belong to the general class of games, known as multi-player differential games [13], which encompass a wide variety of interesting problems (e.g. pursuit-evasion games [9, 21], network consensus problems under adversarial attacks [18, 19], reach-avoid games [12, 15]) and have been a subject of significant past research. For reach-avoid related games, the approaches that have been proposed often feature trade-offs between optimality of the solution (with respect to the time to achieve a player's objective), and the complexity of the computation.

Our approach in this paper to reach-avoid games is to formulate them as an open-loop game, and more specifically, as a framework of open-loop games that includes the open-loop upper value game and open-loop lower values game. The open-loop games for the reach-avoid problem formulated here are an instantiation of a general Stackelberg game [1, 11], an important class of games for modeling strategic behavior in dynamic games. In an open-loop game, the granularity of a strategy at which a player chooses is a control function that maps the entire time horizon to a trajectory. Depending on which player(s) choose(s) first, and which one(s) choose(s) in response, they are leader(s) and follower(s) respectively.

The open-loop games are conservative towards one side of players: the side of players that chooses first. Consequently, this level of conservatism offers performance guarantees of the solution. Namely, if a solution exists, the player is guaranteed to achieve the desired objective, irrespective of the actions of the opponent, without needing to incorporate any future state information. This is particularly well-suited for certain safety-critical applications, where state update is hard or costly to obtain, for example in GPS-denied environments. Another safety-critical application is robotic surgery, where the robotic system needs to navigate inside human body to eliminate certain tissues while ensuring that no invasive damage is induced. Those safety-critical applications tend to demand the solution to satisfy strong performance guarantees (e.g. achieving the goal regardless of what the disturbance does, adversarial or non-adversarial alike), a property guaranteed by the open-loop framework.

In a static Stackelberg game (only one-round of interaction between players), an exhaust tree search can be used to find the optimal action [1, 11]. However, at least in the context of reach-

☆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).

*Email addresses:* zyzhou@stanford.edu (Zhengyuan Zhou), jerryding@berkeley.edu (Jerry Ding), haomiao@stanford.edu (Haomiao Huang), rrtakei@eecs.berkeley.edu (Ryo Takei), tomlin@eecs.berkeley.edu (Claire Tomlin)

avoid open-loop dynamic games, no efficient algorithms have been devised to produce the optimal controls. Consequently, this paper is particularly focused upon the study of efficient and scalable computational algorithms for solving open-loop reach-avoid games. We emphasize that the proposed approach does not involve solving *any* Hamilton-Jacobi-Isaacs (HJI) equations [1] in the high dimensional space of all player states. Instead, as discussed in detail later, we reduce the problem to solving Hamilton-Jacobi-Bellman (HJB) equations in the low dimensional space of individual player states.

### *1.1. Related Work*

We now provide a brief overview of related work. For certain games, it is possible to construct strategies (sometimes even optimal) for the players analytically or geometrically. Defending a line segment on a plane without obstacles have been considered in [24] [17], where the motion has been restricted to fixed maximum speed and moving along line segments, respectively. In both cases, optimal strategies have been found for defending the target set (a line segment). In [20], a geometric approach based on Voronoi partition has been employed to establish optimal strategies for the two-player reach-avoid game in a setting where maximum speeds for both players are again fixed and no obstacles are present. 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 [8, 10, 32]. 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, issues that our formulation (Section 2.1) captures and that our computational framework (Section 4) addresses.

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 [1, 5, 13, 22]. Solutions are typically found either using the method of characteristics [1, 13], where optimal trajectories are computed by integrating backward from a known terminal condition, or via numerical approximation of the HJI equation on grids [7, 22]. For the HJI method, the number of grid points needed to approximate the value function typically grow exponentially in the number of continuous states. As such, finding solution strategies for such games can be computationally expensive, even for games involving a single attacker and a single defender. For differential games with multiple players, the computational burden also scale exponentially in the number of players. On a related note, in particular differential game scenarios, approximate dynamic programming (ADP) techniques have been developed to efficiently compute game solutions. For example, clever policy iterations have been designed to solve certain two-player zero-sum games [14, 31]. However, the differential game scenario considered in this paper has a non-smooth value function. The application of ADP methods to such scenarios have been found to be challenging from a numerical convergence standpoint [23]. In closing, we mention that another related research thread concerns reach-avoid games under imperfect or incomplete information [3, 25], where the players do not necessarily know the locations of the other players at all times.

As a final note, closely related to reach-avoid problems is the class of pursuit-evasion problems [9, 21, 36]. A reach-avoid game shares some similar aspects of a pursuit-evasion game: the attacker (similar to an evader) needs to avoid the capture of the defender (similar to a pursuer) because capture would result in the attackers losing the game instantly. However, there is a key distinction between the two: the utility of an agent in a reach-avoid game is fundamentally different from that of a pursuit-evasion game. In the latter, an evader's utility is solely based on whether it will ever be captured, whereas in the former, the attacker's utility depends on how long it takes to reach the target. Consequently, a reach-avoid problem is much more challenging because the attacker not only needs to avoid capture, but also needs to reach a per-determined target set. The recent work [36] studies a multi-pursuer-single-evader pursuit-evasion problem, where an analytical cooperative pursuit strategy for the pursuers is derived using Voronoi partitions. This geometric approach (and the analytical results therein) are feasible because all agents in the pursuit-evasion problem are assumed to have constant speed. Conversely, in our current reach-avoid setting, in addition to the complexity just mentioned, we also allow for arbitrary speed profiles for all agents. As a result, these two levels of generality dictate that a completely different and non-analytical approach be taken.

### *1.2. Our Contributions*

In this paper, we provide a computationally tractable framework for solving open-loop reach-avoid games with two or more players. Our major contribution is the algorithmic framework for computing open-loop values. Specifically, we develop (Section 4) efficient and novel numerical algorithms (a set of modified fast-marching methods) that allow us to quickly compute solutions to these open-loop games, in the form of a set of open-loop player trajectories with provable properties. To the best of our knowledge, this is the first set of efficient algorithms for computing open-loop values.

We note 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 (in general the solution to an HJI equation, often intractable to compute for games with more than two players) of the reach-avoid game. In this sense, our open-loop

[1] This corresponds to a closed-loop formulation. See Section 1.1 for a discussion and related work.

framework can be interpreted as a computationally efficient approximation of the closed-loop value for reach-avoid games, through the trade-off of a certain degree of optimality for a reduction in computational complexity. In particular, there exist non-trivial cases where solving for open-loop values also yield closed-loop values.

Some of the theoretical results were presented in two previous publications [30, 35]. This work unifies the presentation of the open-loop games and their relationship to each other. Specifically, we establish a theoretical framework (Section 3) that puts on rigorous footing the procedure for relating various optimal time-to-reach functions. We also expand the discussion on the novel aspects of the developed algorithmic framework, all of which were previously omitted in the conference papers due to space limitation.

## 2. The Reach-Avoid Problem

### 2.1. 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 $\Omega$ can be further partitioned as follows: $\Omega = \Omega_{free} \cup \Omega_{obs}$, where $\Omega_{free}$ 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 directions in which $P_A$ and $P_D$ are moving. 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 $\Sigma = \{\sigma \colon [0, \infty) \to \overline{B}_n \mid \sigma \text{ is continuous}\}$, 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 $\Sigma$ 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 function $\mathcal{J} : \mathbb{R}^n \times \Sigma \times \Sigma \to \mathbb{R}$ as that of a minimum time differential game problem with state constraints (see for example Appendix A of [2]):

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

where the infimum of the empty set is by convention $+\infty$. Given that this is a minimum time problem, the payoff function (2) is in general discontinuous. In particular, it is finite on the set of initial conditions $\mathbf{x^0} \in \mathbb{R}^n$ for which $P_A$ can reach the target set $\mathcal{T}$ without being captured by $P_D$. It is equal to $+\infty$ everywhere else.

### 2.2. 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 [4, 33]). 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 $\mathcal{F}_A$ of admissible strategies for $P_A$ is given by

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

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

For $\mathbf{x^0} \in \Omega_{free}^2$, the upper value and lower value of the closed-loop reach-avoid game are respectively given by:

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

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

### 2.3. Open-loop Game Formulation

In principle, one can compute $\overline{V}(\mathbf{x^0})$ (or $\underline{V}(\mathbf{x^0})$) by solving a corresponding Hamilton-Jacobi-Isaacs (HJI) partial differential equation [13]: they are shown to be viscosity solutions to HJI equations [5, 22]. See [35] for more details. Although numerical methods for solving an HJI equation on a Cartesian grid exist, they 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].

Motivated by the difficulty of computing the closed-loop value function, we now consider the open-loop formulations of the reach-avoid game, where each player selects *controls* from the control spaces $\Sigma_A$ or $\Sigma_D$, 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:*

$$\begin{aligned} \overline{v}(\mathbf{x^0}) &= \inf_{a\in\Sigma_A} \sup_{d\in\Sigma_D} \mathcal{J}(\mathbf{x^0}, a, d), \\ \underline{v}(\mathbf{x^0}) &= \sup_{d\in\Sigma_D} \inf_{a\in\Sigma_A} \mathcal{J}(\mathbf{x^0}, a, d). \end{aligned} \tag{5}$$

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. It is known that:

$$\underline{v}(\mathbf{x^0}) \le \underline{V}(\mathbf{x^0}) \le \overline{V}(\mathbf{x^0}) \le \overline{v}(\mathbf{x^0}),\ \forall \mathbf{x^0} \in \Omega^2_{free}. \tag{6}$$

Consequently, 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. As a very important note, the open-loop values $\overline{v}(\mathbf{x^0})$ and $\underline{v}(\mathbf{x^0})$ are not solutions to HJI equations. Instead, they can be computed by solving a certain constrained Hamilton-Jacobi-Bellman equations discussed in Section 4, as opposed to HJI equations.

The optimal controls $\bar{a} \in \Sigma_A$, $\bar{d} \in \Sigma_D$, $\underline{a} \in \Sigma_A$ and $\underline{d} \in \Sigma_D$ for the open-loop upper and lower value games, are respectively:

$$\begin{aligned} \bar{a} &\in \arg\min_{a\in\Sigma_A} \sup_{d\in\Sigma_D} \mathcal{J}(\mathbf{x^0}, a, d), \quad \bar{d} \in \arg\max_{d\in\Sigma_D} \mathcal{J}(\mathbf{x^0}, \bar{a}, d), \\ \underline{d} &\in \arg\max_{d\in\Sigma_D} \inf_{a\in\Sigma_A} \mathcal{J}(\mathbf{x^0}, a, d), \quad \underline{a} \in \arg\min_{a\in\Sigma_A} \mathcal{J}(\mathbf{x^0}, a, \underline{d}). \end{aligned} \tag{7}$$

Note that $(\bar{a}, \bar{d})$ can be interpreted as the Stackelberg equilibrium of a zero-sum game [1]; similarly for $(\underline{a}, \underline{d})$.

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.

## 3. 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.

### 3.1. Upper Value

We first examine the upper value game, which is conservative toward $P_A$. The value function for this game is defined in (5). 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:

$$\begin{aligned} \mathcal{S} :=& \{y \in \Omega_{free} \mid \exists a \in \Sigma_A, \forall d \in \Sigma_D, \exists t \ge 0, x_A(t) = y, \\ & \mathbf{x}(s) \notin \mathcal{A}, \forall s \in [0, t]\}, \end{aligned} \tag{8}$$

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 $x^0_D \in \Omega_{free}$ and $y \in \Omega_{free}$, the minimum time-to-capture for $P_D$ is*

$$\bar{t}(y; x^0_D) := \inf_{d\in\Sigma_D} \inf\{t \ge 0 \mid (y, x_D(t)) \in \mathcal{A}, x_D(0) = x^0_D\}. \tag{9}$$

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

**Definition 3.** *Given an initial state $x^0_A \in \Omega_{free}$, 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*

$$\begin{aligned} \overline{w_R}(y; x^0_A) := \inf_{a\in\Sigma_A} \inf\{&t \ge 0 \mid x_A(t) = y, x_A(0) = x^0_A, \\ & x_A(s) \in R, \forall s \in [0, t]\}. \end{aligned} \tag{10}$$

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 $\mathcal{S}$.

**Lemma 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 $\overline{w_{\mathcal{M}}}(y; x_A^0) < \bar{t}(y; x_D^0),\ \forall y \in \mathcal{M}$. Then $\mathcal{M} = \mathcal{S}$.*

*Proof.* Let $\mathcal{E}$ be the collection of all sets $E \subseteq \Omega_{free}$ satisfying 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 $\mathcal{E}$. 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 (10) 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, which in turn implies $\mathcal{M} = E^*$.

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 (9), 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 $\epsilon$-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 $\mathcal{S} = E^* = \mathcal{M}$. □

Intuitively, the above result states that $\mathcal{S}$ is the largest set in the free space $\Omega_{free}$ 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 value $\overline{v}$ can now be found using the minimum time-to-reach function $\overline{w_{\mathcal{S}}}$.

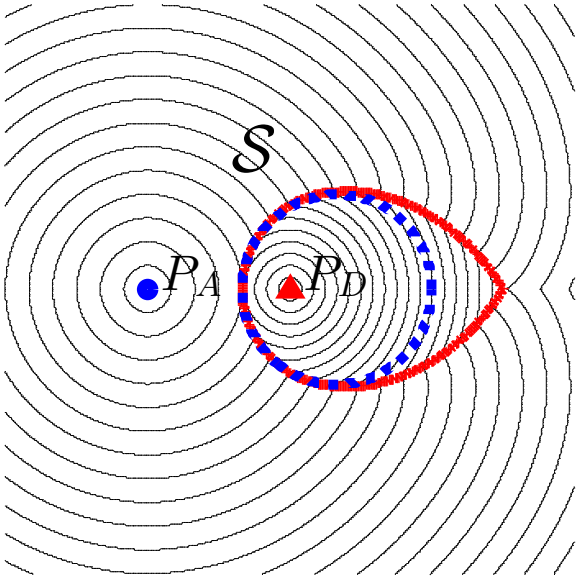



Figure 1: **$\mathcal{S}$ is the subset partitioned by the solid red curve containing $x_A^0$. Level sets of $\overline{w_{\mathcal{S}}}$ on $\mathcal{S}$ and $\bar{t}_c$ on $\Omega \setminus \mathcal{S}$ are plotted. The dotted blue circle is the set of equal time-to-reach points of $P_A$ and $P_D$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

**Theorem 1.** *Given compact sets $\mathcal{T}, \mathcal{A} \subset \Omega_{free}$, and initial condition $\mathbf{x^0} = (x_A^0, x_D^0) \in \Omega_{free}^2$, let $\mathcal{S}$ be as defined in (8), and $\overline{w_{\mathcal{S}}}(y; x_A^0)$ be as defined in (10). Let the upper value $\overline{v}$ be as defined in (5). Then*

$$\overline{v}(\mathbf{x^0}) = \inf\{\overline{w_{\mathcal{S}}}(y; x_A^0) \mid y \in \mathcal{T}\}.$$

*Proof.* Let $\mathbf{x^0} \in \Omega^2$. 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 $a \in \Sigma$, we have $\sup_{d \in \Sigma} \mathcal{J}(\mathbf{x^0}, a, d) = \infty$. Using the definition of the payoff in (2), one can then infer that for every $a \in \Sigma$, there exists $d \in \Sigma$ 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 where $\overline{v}(\mathbf{x^0}) < \infty$. This implies that there exists $a \in \Sigma$ such that $\sup_{d \in \Sigma} \mathcal{J}(\mathbf{x^0}, a, d) < \infty$. Let $\Sigma_1$ be the set of all such controls, and $\mathcal{J}_1(\mathbf{x^0}, a) := \sup_{d \in \Sigma} \mathcal{J}(\mathbf{x^0}, a, d)$. Fix any $a^* \in \Sigma_1$, then it can be verified that for every $d \in \Sigma$, there exists $t \geq 0$ such that $x_A(t) \in \mathcal{T}$ and $x(s) \notin \mathcal{A}$, $\forall s \in [0, t]$. For a given $a^*$, define $t^* := \inf\{t \geq 0 \mid x_A(t) \in \mathcal{T}, x(s) \notin \mathcal{A}, \forall s \in [0, t]\}$, then it follows that for every $d \in \Sigma$, $\mathcal{J}(\mathbf{x^0}, a^*, d) = t^*$. Thus, $\mathcal{J}_1(\mathbf{x^0}, a^*) = t^*$. Moreover, by a similar argument as in the proof of Proposition 1, one can show that $x_A(s) \in \mathcal{S},\ \forall s \in [0, t^*]$. Let $z^* = x_A(t^*)$, then we have $\overline{w_{\mathcal{S}}}(z^*; x_A^0) \leq t^* = \mathcal{J}_1(\mathbf{x^0}, a^*)$. Thus, $\inf_{y \in \mathcal{T}} \overline{w_{\mathcal{S}}}(y; x_A^0) \leq \mathcal{J}_1(\mathbf{x^0}, a^*)$, $\forall a^* \in \Sigma_1$, 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 $y \in \mathcal{S} \cap \mathcal{T}$ and $x_A^*(\cdot)$ be the optimal path from $x_A^0$ to $y$ in $\mathcal{S}$. By the assumptions on the system dynamics (1) and the control set $\Sigma$, there exists a choice of control $a^* \in \Sigma$ which realizes such a path. Define $t^* := \overline{w_{\mathcal{S}}}(y; x_A^0)$, then by Proposition 1, it follows 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

$d \in \Sigma$, the joint path satisfies $(x_A^*(s), x_D(s)) \notin \mathcal{A},\ \forall s \in [0, t^*]$. From this, we have $\mathcal{J}(\mathbf{x^0}, a^*, d) \leq t^*,\ \forall d \in \Sigma$, which in turn implies that $\overline{v}(\mathbf{x^0}) \leq t^* = \overline{w_{\mathcal{S}}}(y; x_A^0)$. 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 conclusion of the theorem then 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 4. 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.

**Remark 1.** *Several remarks are in order here. First, from the perspective of playing a reach-avoid game in practice, this open-loop assumption is merely a manifestation of the attacking side's own mindset. That is, the attacker, in computing its attacking strategy, proceeds under the worst-case but imaginary constraint that its final chosen strategy will be made available to the defending side before the defending side selects its strategy. The implication of this mindset is that the attacking strategy (if one exists) thus obtained, will enjoy the worst-case guarantees as described in the paper. Of course, the attacking side, in playing the game, will not reveal his strategy information to the defending side, thereby potentially achieving better results than the worst-case guarantees.*

*Second, the resulting optimal open-loop trajectory for the attacker can be quite conservative, as it disallows the attacker from incorporating any new information when the game proceeds. One way for relaxing this conservatism is via an iterative open-loop update, where the attacher, instead of committing entirely to the open-loop trajectory, follows the trajectory for $\delta t$, collects the current state of the defender and recomputes the optimal open-loop trajectory, with the joint initial condition now being the current state of both players. This process is then repeated until the attacker reaches the target set.*

*Such an iterative update scheme can improve the solution quality (i.e. the time it takes the attacker to reach the target), particularly when the defender is taking a suboptimal trajectory, which creates new opportunities for the attacker; in the original open-loop formulation, such opportunities cannot be taken advantage of. We note that such improvements for the attacker always hold, independent of what the defender does (e.g. whether or not the defender itself is also executing a similar iterative open-loop scheme). This idea, while similar in spirit to Model Predicitve Control (MPC), has a crucial difference: it does not impose any specific model on the defender's actions; it merely equips itself with a worse-case mind-set at every iteration.*

### *3.2. Lower Value*

This section studies the second value function defined in Equations (5). 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.

**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*

$$\begin{aligned}\underline{t}(y; x_A^0) := \inf_{a \in \Sigma_A} \inf\{t \geq 0 \mid x_A(t) \in \mathcal{T}, x_A(0) = x_A^0, \\ (x_A(s), y) \notin \mathcal{A}, \forall s \in [0, t]\}.\end{aligned} \tag{11}$$

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.

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 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*

$$\underline{w_R}(y; x_D^0) := \inf_{d \in \Sigma_D} \inf\{t \geq 0 \mid x_D(t) = y, x_D(0) = x_D^0, \\ x_D(t) \in R, \forall s \in [0, t]\}. \tag{12}$$

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 $\Omega_{free}$ 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 $\Omega_{free}$ 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.

$$\mathcal{R}_1(t) := \{x \in \Omega_{free} \mid \exists a \in \Sigma_A, \exists s \in [0, t], x_A(s) = x, \\ x_A(0) = x_A^0\} \tag{13}$$

In other words, this is the set of all states in $\Omega_{free}$ 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 $\mathbf{x^0} \in \Omega_{free}^2$, we define $\underline{\underline{v}}(\mathbf{x^0})$ as follows:*

$$\underline{\underline{v}}(\mathbf{x^0}) := \begin{cases} \sup_{y \in \mathcal{Z}^*} \underline{t}(y; x_A^0), & x_A^0 \notin \mathcal{T} \wedge \mathbf{x^0} \notin \mathcal{A} \\ 0, & x_A^0 \in \mathcal{T} \wedge \mathbf{x^0} \notin \mathcal{A} \\ \infty, & \mathbf{x^0} \in \mathcal{A} \end{cases} \tag{14}$$

*where*

$$\mathcal{Z}^* := \{y \in \mathcal{Z} \mid \mathcal{A}_y \cap \mathcal{R}_1(T(y)) = \emptyset\}, \; T(y) := \underline{w_{\mathcal{Z}}}(y; x_D^0). \tag{15}$$

$\mathcal{Z}^*$ contains all points $y$ from $\mathcal{Z}$ at which $P_D$ can reach before $P_A$, and along a path that ensures $P_A$ cannot end the game before $P_D$ reaches $y$. Thus, if $P_D$ reaches a point in $\mathcal{Z}^*$ via a shortest path contained in $\mathcal{Z}^*$, then by the time $P_D$ reaches that point, it is guaranteed that $P_D$ still has not entered the target yet. $\underline{\underline{v}}(\mathbf{x^0})$ encodes the time it takes for the game to end if the defender picks the best point in $\mathcal{Z}^*$ (points in the pink region in this picture), reaches that point via a time-optimal path and thereafter stays at that point. This leads to the following result.

**Theorem 2.** *Given initial condition $\mathbf{x^0} = (x_A^0, x_D^0) \in \Omega_{free}^2$, let function $\underline{\underline{v}}$ be as defined in (14), and lower value $\underline{v}$ be as defined in (5). Then*

$$\underline{\underline{v}}(\mathbf{x^0}) \leq \underline{v}(\mathbf{x^0}).$$

*Proof.* For any $\mathbf{x^0} = (x_A^0, x_D^0) \in \Omega_{free}^2$ 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 (14) that the result holds with equality.

Now consider an initial state $\mathbf{x^0} \in \Omega_{free}^2$ 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 $\epsilon$-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 (11), 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 (14) 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.

## 4. 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 $\overline{w_S}$

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 $\underline{v}$ via modified FMM. Due to space limitation, we the readers to [6, 26, 27, 30] for a discussion on FMM.

### *4.1. Upper Value*

We 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 HJB equation in $\mathcal{S}$:

$$-\inf_{a\in\overline{B}_n}\{\nabla\overline{w_{\mathcal{S}}}(y;x_A^0)\cdot f_A(x_A)a\}=1 \tag{16}$$

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

$$\overline{w_{\mathcal{S}}}(x_A^0;x_A^0)=0;\ \overline{w_{\mathcal{S}}}(y;x_A^0)=\infty,\ y\in\Omega\setminus\mathcal{S}. \tag{17}$$

We note that if the speed function $f_A$ is identified with $v(\cdot)$ in the Eikonal equation, 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 $\phi_D$ 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 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 [27].

**Remark 2.** *Here we highlight the novel aspects of the modified FMM as given above. First, at a high level, we note that FMM is a numerical method for computing shortest paths (i.e. as solutions to certain minimum-to-reach type PDE as in HJB). However, note that here it is far from sufficient to just compute the shortest path from the attacker to the target: such a path must be computed subject to a crucial constraint that at each point along this path, the attacker must be safe (i.e. arriving there before the defender can intercept it).*

*Next, we give a detailed description of the modified aspects employed here. 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 $\mathcal{S}\approx\{(i,j)\mid\overline{w_{\mathcal{S}}}^{(i,j)}<\infty\}$.*

### *4.2. 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 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 $\infty$ 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 $\infty$. 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 $\underline{\underline{v}} = M$ and return $\underline{\underline{v}}$.

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 $\infty$. 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 $\infty$ 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.

**Remark 3.** *Here we outline the novel aspects of the modified FMM in the open-loop lower value computation. Again, it is insufficient for the defender to just compute the shortest path because it must arrive there before the attacker can in order to serve effectively as a obstacle. Furthermore, here, as opposed to both the traditional FMM and the modified FMM for the open-loop upper value, the defender is interested in selecting the largest $\underline{t}_{(i,j)}$, value that encodes how long it can delay the attacker from reaching the target. The addition of Steps 4) and 5) serve precisely those two purposes.*

*Finally we note that, for both the open-loop upper and lower computation, 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) [28], allowing the method to be eventually extended to anisotropic [28] and non-holonomic [29] dynamics.*

### 4.3. 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 $\Sigma_A$, 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

$$\dot{x}_A^*(t) = f_A(x_A^*(t))\mu(x_A^*(t)) \tag{18}$$

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 $\bar{a}(t) = \mu(x_A^*(t))$.

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 $x_D^*(\cdot)$. We also mention that extracting the inputs and synthesizing the resulting controls can also be useful and instrumental even when the precise trajectory is not needed: they can still provide intuitive verification tools on the algorithm implementation.

### 4.4. Multilayer Reach-Avoid Games and Simulations

The modified fast marching methods extend straightforwardly to a multi-player reach-avoid game setting, where the attacking and/or defending side can contain any number of players. Here we demonstrate 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 2). 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.

Figure 2: **A segment of the UC Berkeley campus used for the simulations (map image courtesy of maps.google.com).**

Two sets of simulations are presented here. The first set of simulations shows an upper value game with two attackers and two defenders, as illustrated in Figure 3. In Figure 3 players $P_A^i$ and $P_D^j$ are denoted by their positions $x_{A,i}$, $x_{D,j}$ respectively. 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 3(a) and (b). Figure 3(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 3(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 $P_A^1$, $P_A^2$ and $P_D$ 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.

## 5. Conclusion and Future Work

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.

## References


[1] T. Başar and G. Olsder. *Dynamic Noncooperative Game Theory*. SIAM, Philadelphia, PA, 2 edition, 1999.

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

[3] Laurent Doyen and Jean-François Raskin. Games with imperfect information: Theory and algorithms. *Lectures on Games Theory for the Computer Scientist*, pages 185–212, 2010.

[4] R. J. Elliott and N. J. Kalton. The existence of value in differential games. In *Memoirs of the American Mathematical Society*, number 126. 1972.

[5] L. C. Evans and P. E. Souganidis. Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations. *Indiana University Mathematics Journal*, 33(5):773–797, 1984.

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

[7] M Falcone and R Ferretti. Semi-Lagrangian schemes for Hamilton-Jacobi equations, discrete representation formulae and Godunov methods. *Journal of Computational Physics*, 175(2):559–575, 2002.

[8] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760, 1998.

[9] James Flynn. Lion and man: the general case. *SIAM Journal on Control*, 12(4):581–597, 1974.

[10] Thierry Fraichard and Hajime Asama. Inevitable collision states - a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

[11] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[12] H. Huang, J. Ding, W. Zhang, and C.J. 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] R. Isaacs. *Differential Games*. Wiley, New York, 1967.

[14] M. Johnson, S. Bhasin, and W.E. Dixon. Nonlinear two-player zero-sum game approximate solution using a policy iteration algorithm. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 142–147, Dec 2011.

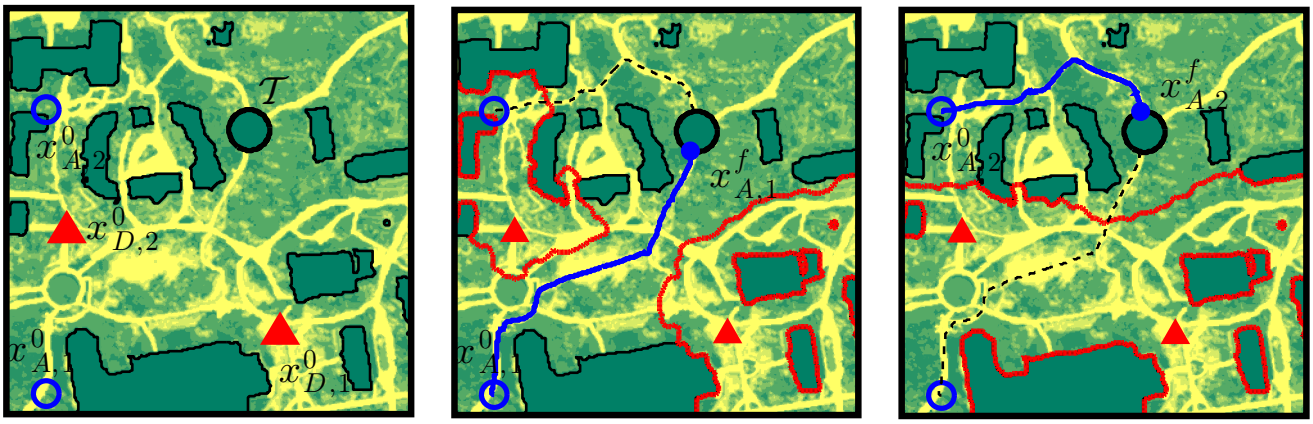
[15] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for a class of pursuit-evasion games. In David Hsu, Volkan Isler, Jean-Claude Latombe, and MingC. Lin, editors, *Algorithmic Foundations of Robotics IX*, volume 68 of *Springer Tracts in Advanced Robotics*, pages 71–87. Springer Berlin Heidelberg, 2011.

[16] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[17] P. Kawecki, B. Kraska, K. Majcherek, and M. Zola. Guarding a line segment. *Systems & Control Letters*, 58(7):540–545, 2009.

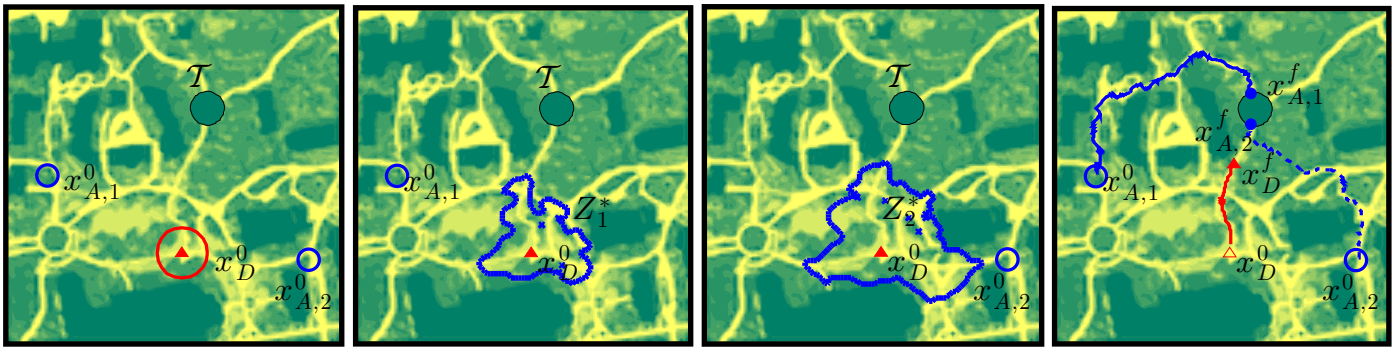
[18] A. Khanafer, B. Touri, and T. Basar. Consensus in the presence of an adversary. *3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems (NecSys)*, 2012.

[19] A. Khanafer, B. Touri, and T. Basar. Robust distributed averaging on networks with adversarial intervention. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 7131–7136, Dec 2013.




(a) (b) (c)

Figure 3: **Simulations of a game between two attackers (blue circles) and two defenders (red triangles) 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 blue line) and the boundary of its safe-reachable set with respect to the two defenders (solid red regions), indicating that this attacker can safely arrive at the target by successfully avoid both defenders. (c) The path taken by attacker 2 (solid blue line) and the boundary of its safe-reachable set with respect to the defenders (solid red regions), similarly indicating that this attacker can safely arrive at the target by successfully avoid both defenders..**




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

Figure 4: **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. The defender, by taking this route, can force the attackers to take the two routes respectively (of course they can take other suboptimal routes that may take them longer to reach the target), thereby guaranteeing that the attackers will not enter the target unless a certain amount of time has passed.**


[20] Ngoc-Minh Le. On determining optimal strategies in pursuit games in the plane. In Zoltn Flp and Ferenc Gcseg, editors, *Automata, Languages and Programming*, volume 944 of *Lecture Notes in Computer Science*, pages 499–510. Springer Berlin Heidelberg, 1995.

[21] J Lewin. The lion and man problem revisited. *Journal of optimization theory and applications*, 49(3):411–430, 1986.

[22] I.M. Mitchell, A.M. Bayen, and C.J. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7):947–957, 2005.

[23] Remi Munos, Leemon C Baird, and Andrew W Moore. Gradient descent approaches to neural-net-based solutions of the Hamilton-Jacobi-Bellman equation. In *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, volume 3, pages 2152–2157. IEEE, 1999.

[24] W. Rzymowski. A problem of guarding line segment. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 6444–6447, Dec 2009.

[25] Y.B. Sebbane. *Planning and Decision Making for Aerial Robots*. Intelligent Systems, Control and Automation: Science and Engineering. Springer International Publishing, 2014.

[26] J. A. Sethian. Fast marching methods. *SIAM Review*, 41(2):199–235, 1999.

[27] J. A. Sethian. *Level set methods and fast marching methods*, volume 3 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, UK, second edition, 1999.

[28] J. A. Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms. *SIAM Journal of Numerical Analysis*, 41(1):325–363, 2003.

[29] 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.

[30] Ryo Takei, Haomiao Huang, Jerry Ding, and C.J. 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.

[31] Kyriakos G. Vamvoudakis and F.L. Lewis. Online solution of nonlinear two-player zero-sum games using synchronous policy iteration. *International Journal of Robust and Nonlinear Control*, 22(13):1460–1483, 2012.

[32] J. Van Den Berg and M. Overmars. Planning time-minimal safe paths amidst unpredictably moving obstacles. *International Journal of Robotics Research*, 27(11-12):1274, 2008.

[33] P. Varaiya. On the existence of solutions to a differential game. *SIAM*

*Journal on Control*, 5(1):153–162, 1967.

[34] Zhengyuan Zhou, Jonathan Shewchuk, Haomiao Huang, and C.J. Tomlin. On 3-pursuer guaranteed capture in general planar domains. In *Proceedings of the IEEE International Conference on Decision and Control, submitted*, Maui, Hawaii, 2012.

[35] Zhengyuan Zhou, Ryo Takei, Haomiao Huang, and C.J. Tomlin. A general, open-loop formulation for reach-avoid games. In *Proceedings of the IEEE International Conference on Decision and Control*, Maui, Hawaii, 2012.

[36] Zhengyuan Zhou, Wei Zhang, Jerry Ding, Haomiao Huang, Dušan M Stipanović, and Claire J Tomlin. Cooperative pursuit with voronoi partitions. *Automatica*, 72:64–72, 2016.