Incremental Sampling-Based Algorithms for a Class of Pursuit-Evasion Games

Incremental Sampling-based Algorithms for a class of Pursuit-Evasion Games

Sertac Karaman and Emilio Frazzoli

Abstract Pursuit-evasion games have been used for modeling various forms of conflict arising between two agents modeled as dynamical systems. Although analytical solutions of some simple pursuit-evasion games are known, most interesting instances can only be solved using numerical methods requiring significant offline computation. In this paper, a novel incremental sampling-based algorithm is presented to compute optimal open-loop solutions for the evader, assuming worst-case behavior for the pursuer. It is shown that the algorithm has probabilistic completeness and soundness guarantees. As opposed to many other numerical methods tailored to solve pursuit-evasion games, incremental sampling-based algorithms offer anytime properties, which allow their real-time implementations in online settings.

1 Introduction

Pursuit-evasion games have been used for modeling various problems of conflict arising between two dynamic agents with opposing interests [1, 2]. Some examples include multiagent collision avoidance [3], air combat [4], and path planning in an adversarial environment [5]. The class of pursuit-evasion games that will be considered in this paper is summarized as follows. Consider two agents, an evader and a pursuer; the former is trying to "escape" into a goal set in minimum time, and the latter is trying to prevent the evader from doing so by "capturing" her, while both

Sertac Karaman
Laboratory for Information and Decision Systems,
Department of Electrical Engineering and Computer Science,
Massachusetts Institute of Technology. e-mail: sertac@mit.edu

Emilio Frazzoli
Laboratory for Information and Decision Systems,
Department of Aeronautics and Astronautics,
Massachusetts Institute of Technology. e-mail: frazzoli@mit.edu

agents are required to avoid collision with obstacles. The evader is only aware of the initial state of the pursuer, while the pursuer has access to full information about the evader's trajectory. This class of pursuit-evasion games is of interest when the evader can be easily detected by stealthy pursuers, who operate from known locations. Problems in this class include the generation of trajectories for an airplane to avoid threats from known Surface-to-Air Missile (SAM) sites, or for a ship to avoid attacks by pirates based at known locations. The information structure of this class of pursuit-evasion games is such that the evader discloses her (open-loop) strategy first, and the pursuer decides his strategy accordingly. In this setting, the evader's strategy should be chosen carefully, considering the worst-case (from the evader's point of view) response of the pursuer. Rational players in this game will choose a Stackelberg strategy with the evader as a leader [6].

Analytical solutions to certain classes of pursuit-evasion games, e.g., the "homicidal chauffeur" and the "lady in the lake" games, exist [1,2]. However, for problems involving agents with more complex dynamics, or for problems involving complex environments (e.g., including obstacles), existing analytical techniques are difficult to apply. For example, the pursuit-evasion game addressed in this article can be solved in principle by determining the set of all states that can be reached by the evader before the pursuer, and then choosing the optimal trajectory for the evader, if one exists, within this set [1]. In the simple case of kinematic agents moving with bounded speed within an empty environment, such a set coincides with the evader's region in a Voronoi tesselation generated by the evader's and pursuer's initial conditions. However, analytical methods for computation of this set are not available in the general case in which non-trivial dynamics and obstacles are considered.

Standard numerical approaches for solving pursuit-evasion games are based on either direct or indirect methods [7]. The former reduce the problem to a sequence of finite dimensional optimization problems through discretization [8], whereas the latter solves the Isaacs partial differential equation with boundary conditions using, e.g., multiple shooting [9, 10], collocation [11, 12], or level-set methods [3, 13].

A number of algorithms for motion planning in the presence of dynamic, possibly adversarial obstacles, have been proposed in the context of mobile robotics. A common approach relies on planning in a 'space-time' state space, avoiding spatio-temporal regions representing possible motions of the dynamic obstacles [14, 15]. However, such regions, representing reachable sets by the dynamic obstacles, are typically hard to compute exactly in the general case, and conservative approximations are used, e.g., to estimate regions of inevitable collision [16]. Other recent contributions in this area include [17–24].

Several types of pursuit-evasion games have been studied from an algorithmic perspective. In particular, pursuit games on graphs [25–27] as well as on polygonal environments [28–30] have received significant attention during the last decade. More recently, pursuit-evasion games on timed roadmaps have also been considered [31]. All these approaches typically impose severe limitations on the allowable agents' dynamics, e.g., by considering only finite state spaces and discrete time.

Based on recent advances in incremental sampling-based motion planning algorithms, we propose a new method for solving the class of pursuit-evasion games

under consideration. In fact, the game we consider is a generalization of the kinodynamic motion planning problem [15]. During the last decade, a successful algorithmic approach to this problem has been the class of sampling-based methods including, e.g., Probabilistic RoadMaps (PRMs) [32], which construct a roadmap by connecting randomly-sampled states with feasible trajectories so as to form a strong hypothesis of the connectivity of the state space, and, in particular, the initial state and the goal region.

Incremental versions of sampling-based motion planning methods were proposed to address on-line planning problems [17, 33]. In particular, the Rapidly-exploring Random Tree (RRT) algorithm proposed in [33] has been shown to be very effective in practice, and was demonstrated on various platforms in major robotics events (see, e.g., [34]). Very recently, optimality properties of incremental sampling-based planning algorithms were analyzed and it was shown that, under mild technical assumptions, the RRT algorithm converges to a non-optimal solution with probability one [35]. In [35], the authors have proposed a new algorithm, called RRT*, which converges to an optimal solution almost surely, while incurring essentially the same computational cost when compared to the RRT. The RRT* algorithm can be viewed as an anytime algorithm for the optimal motion planning problem. Loosely speaking, an anytime algorithm produces an approximate solution and gradually improves the quality of the approximation given more computation time [36, 37]. The quality measure is defined, e.g., with respect to a cost function.

In this paper, inspired by incremental sampling-based motion planning methods, in particular the RRT* algorithm, we propose an incremental sampling-based algorithm that solves the pursuit-evasion game with probabilistic guarantees. More precisely, if evader trajectories that escape to the goal set while avoiding capture exist, then the output of the algorithm will converge to the minimum-cost one with probability approaching one as the number of samples increases.To the best of authors' knowledge, this algorithm constitutes the first algorithmic approach to numerically solve, with both asymptotic and anytime guarantees, the class of pursuit-evasion games under consideration.

The paper is organized as follows. Section 2 formulates the problem. The proposed solution algorithms are introduced in Section 3. The algorithm is shown to be probabilistically sound and probabilistically complete in Section 4. Simulation examples are provided in Section 5. Conclusions and remarks on future work can be found in Section 6.

2 Problem Definition

We consider a two-player zero-sum differential game in which one of the players, called the evader, tries to escape in minimum time to a goal set, while the second player, called the pursuer, tries the capture the evader before it reaches the goal set.

More formally, consider a time-invariant dynamical system described by the differential equation $\dot{x}(t) = f(x(t), u_{\mathrm{e}}(t), u_{\mathrm{p}}(t))$, where $x : t \mapsto x(t) \in X \subset \mathbb{R}^d$ is the

state trajectory, $u_\mathrm{e} : t \mapsto u_\mathrm{e}(t) \in U_\mathrm{e} \subset \mathbb{R}^{m_\mathrm{e}}$ is the evader's control input, $u_\mathrm{p} : t \mapsto u_\mathrm{p}(t) \in U_\mathrm{p} \subset \mathbb{R}^{m_\mathrm{p}}$ is the pursuer's control input. The sets X, U_e, and U_p are assumed to be compact, the control signals u_e and u_p are essentially bounded measurable, and $f(z, w_\mathrm{e}, w_\mathrm{p})$ is locally Lipschitz in z and piecewise continuous in both w_e and w_p. Consider also an obstacle set X_obs, a goal set X_goal, and a capture set X_capt; these sets are assumed to be open subsets of X, and X_goal and X_capt are disjoint.

Given an initial condition $x(0) \in X \setminus X_\mathrm{obs}$ and the control inputs of the evader and the pursuer, a unique state trajectory can be computed. The final time of the game is given by $T = \inf\{t \in \mathbb{R}_{\geq 0} : x(t) \in \mathrm{cl}\left(X_\mathrm{goal} \cup X_\mathrm{capt}\right)\}$. Since this is a zero-sum game, only one objective function will be considered, defined as follows: $L(u_\mathrm{e}, u_\mathrm{p}) = T$, if $x(T) \in \mathrm{cl}(X_\mathrm{goal})$; and $L(u_\mathrm{e}, u_\mathrm{p}) = +\infty$, otherwise. The evader tries to minimize this objective function by escaping to the goal region in minimum time, while the pursuer tries to maximize it by capturing the evader before she reaches the goal.

Let $\mathrm{BR} : U_\mathrm{e} \to U_\mathrm{p}$ denote a transformation that maps each evader trajectory to the best response of the pursuer, i.e., $\mathrm{BR}(u_\mathrm{e}) := \arg\max_{u_\mathrm{p}} L(u_\mathrm{e}, u_\mathrm{p})$. In the game described above, the evader picks her strategy so that $L^* = L(u_\mathrm{e}^*, \mathrm{BR}(u_\mathrm{e}^*)) \leq L(u_\mathrm{e}, e_\mathrm{p})$ for all u_e and all u_p. Let $u_\mathrm{p}^* := \mathrm{BR}(u_\mathrm{e}^*)$. Then, ($u_\mathrm{e}^*$ and u_p^*) are called the (open-loop) *Stackelberg strategies* of this differential game [2].

Note that open-loop Stackelberg strategies computed for the evader in this way would be conservative when compared to the saddle-point equilibrium of a pursuit-evasion game with feedback information pattern (see [2]). Open-loop Stackelberg strategies correspond to trajectories that would allow escape without any additional information on the pursuer other than the initial condition. Should other information become available, or should the pursuer not play optimally, the time needed to reach the goal set may be further reduced. In addition, even in the case in which escape is unfeasible (i.e., $L^* = +\infty$) under the open-loop information structure for the evader, there may exist feedback strategies that would allow the evader to escape while avoiding capture.

As common in pursuit-evasion games, the problem considered in this paper further possesses a separable structure, in the following sense. It is assumed that the state can be partitioned as $x = (x_\mathrm{e}, x_\mathrm{p}) \in X_\mathrm{e} \times X_\mathrm{p} = X$, the obstacle set can be similarly partitioned as $X_\mathrm{obs} = (X_\mathrm{obs,e} \times X_\mathrm{p}) \cup (X_\mathrm{e} \times X_\mathrm{obs,p})$, where $X_\mathrm{obs,e} \subset X_\mathrm{e}$ and $X_\mathrm{obs,p} \subset X_\mathrm{p}$, the goal set is such that $X_\mathrm{goal} = (X_{e,\mathrm{goal}} \times X_\mathrm{p}) \setminus X_\mathrm{capt}$, where $X_{e,\mathrm{goal}} \subset X_\mathrm{e}$, and the dynamics are decoupled as follows:

$$\frac{d}{dt}x(t) = \frac{d}{dt}\begin{bmatrix} x_\mathrm{e}(t) \\ x_\mathrm{p}(t) \end{bmatrix} = f(x(t), u(t)) = \begin{bmatrix} f_\mathrm{e}\left(x_\mathrm{e}(t), u_\mathrm{e}(t)\right) \\ f_\mathrm{p}\left(x_\mathrm{p}(t), u_\mathrm{p}(t)\right) \end{bmatrix}, \quad \text{for all } t \in \mathbb{R}_{\geq 0},$$

It is also assumed that the initial condition is an equilibrium state for the pursuer, i.e., there exists $u_\mathrm{p}' \in U_\mathrm{p}$ such that $f_\mathrm{p}(x_\mathrm{init,p}, u_\mathrm{p}') = 0$.

Assume that there exist a Stackelberg strategy enabling the evader to escape (i.e., $L^* < +\infty$). An algorithm for the solution of the pursuit-evasion game defined in this section is said to be *sound* if it returns a control input u_e' such that $\max_{u_\mathrm{p}} L(u_\mathrm{e}', u_\mathrm{p})$, is finite. An algorithm is said to be *complete* if it terminates in finite time returning a solution u_e' as above if one exists, and returns failure otherwise.

The pursuer dynamics can be used to naturally model one or more pursuing agents, as well as moving obstacles whose trajectories are a priori unknown. It is known that even when the number of degrees of freedom of the robot is fixed, the motion planning problem with moving obstacles is NP-hard, whenever the robot has bounds on its velocities. In fact, a simple version of this problem, called the *2-d asteroid avoidance problem*, is NP-hard [38].

The discussion above also suggests that complete algorithms aimed to solve the proposed pursuit-evasion game will be computationally intensive. To overcome this difficulty, we propose a sampling-based algorithm, which is both *probabilistically sound*, i.e., such that the probability that the returned trajectory avoids capture converges to one, and *probabilistically complete*, i.e., such that the probability that it returns a solution, if one exists, converges to one, as the number of samples approaches infinity. Finally, the proposed algorithm is *asymptotically optimal* in the sense that the cost of the returned trajectory converges to the value of the game L^*, almost surely, if $L^* < +\infty$.

3 Algorithm

In this section, an algorithm that solves the proposed pursuit-evasion game with probabilistic soundness and completeness guarantees is introduced. This algorithm is closely related to the RRT* algorithm recently introduced in [35], which will be discussed first. RRT* is an incremental sampling-based motion planning algorithm with the asymptotic optimality property, i.e., almost-sure convergence to optimal trajectories, which the RRT algorithm lacks [35]. In fact, it is precisely this property of the RRT* that allows us to cope with the game introduced in the previous section.

Before formalizing the algorithm, some primitive procedures are presented below. Let $\alpha \in \{\mathrm{e}, \mathrm{p}\}$ denote either the evader or the pursuer.

Sampling: The sampling procedure $\texttt{Sample}_\alpha : \mathbb{N} \to X_\alpha$ returns independent and identically distributed samples from X_α. The sampling distribution is assumed to be absolutely continuous with density bounded away from zero on X_α.

Distance Function: Given two states z_1 and z_2, let $\mathrm{dist}_\alpha(z_1, z_2)$ be a function that returns the minimum time to reach z_2 starting from z_1, assuming no obstacles. Clearly, the distance function evaluates to the Euclidean distance between z_1 and z_2 when $f_\alpha(x_\alpha, u_\alpha) = u_\alpha$ and $\|u_\alpha\| \leq 1$.

Nearest Neighbor: Given a tree $G = (V, E)$, where $V \subset X_\alpha$, and a state $z \in X_\alpha$, $\texttt{Nearest}_\alpha(G, z)$ returns the vertex $v \in V$ that is closest to z. This procedure is defined according to the distance function as $\texttt{Nearest}_\alpha(G, z) = \arg\min_{v \in V} \mathrm{dist}(v, z)$.

Near-by Vertices: Given a tree $G = (V, E)$, where $V \subset X_\alpha$, a state $z \in X_\alpha$, and a number $n \in \mathbb{N}$, $\texttt{Near}_\alpha(G, z, n)$ procedure returns all the vertices in V that are sufficiently close to z, where closeness is parameterized by n. More precisely, for any $z \in X_\alpha$, let $\texttt{Reach}_\alpha(z, l) := \{z' \in X \,|\, \mathrm{dist}(z, z') \leq l \vee \mathrm{dist}(z', z) \leq l\}$. Given, z and n, the distance threshold is chosen such that the set $\texttt{Reach}_\alpha(z, l(n))$ contains a ball of volume $\gamma_\alpha \frac{\log n}{n}$, where γ_α is an appropriate constant. (This particular scaling rate is cho-

sen since it ensures both computational efficiency and asymptotic optimality of the RRT* algorithm [35,39].) Finally, we define $\texttt{Near}_\alpha(G,z,n) := V \cap \texttt{Reach}_\alpha(z, l(n))$.

Collision Check: Given a state trajectory $x : [0,t] \to X_\alpha$, the $\texttt{ObstacleFree}_\alpha(x)$ procedure returns true if and only if x lies entirely in the obstacle-free space, i.e., if and only if $x(t') \notin X_{\text{obs},\alpha}$ for all $t' \in [0,t]$.

Local Steering: Given two states $z_1, z_2 \in X_\alpha$, the $\texttt{Steer}_\alpha(z_1,z_2)$ function returns a trajectory that starts from z_1 and ends at z_2, ignoring obstacles. We assume that the `Steer` procedure returns a time optimal trajectory that connects z_1 and z_2 exactly if z_1 and z_2 are sufficiently close to each other. More precisely, there exists an $\bar{\varepsilon} > 0$ such that for all $\|z_1 - z_2\| \leq \bar{\varepsilon}$, the $\texttt{Steer}(z_1,z_2)$ procedure returns (x,u,t) such that $x(0) = z_1$, $x(T) = z_2$, and $\dot{x}(t') = f_\alpha(x(t'),u(t'))$ for all $t' \in [0,t]$, and t is minimized.

Given a vertex v, let x_v be the unique trajectory in the tree that starts from the root vertex and reaches v. Let us denote the time that x_v reaches v by $T(v)$; given a state trajectory $x : [0,t] \to X$, let us denote the ending time t with $\texttt{EndTime}(x)$.

If the pursuer is inactive (e.g., it is not moving), the pursuit-evasion problem in Section 2 reduces to a standard time-optimal kinodynamic motion planning problem. The RRT* algorithm that solves this problem is presented in Algorithm 1.

The RRT* algorithm proceeds similarly to other incremental sampling-based motion planning methods (e.g., the RRT [33]) by first sampling a state a from the obstacle-free space (Line 4) and then extending the tree towards this sample (Line 5). The extension procedure of the RRT*, presented in Algorithm 2, first extends the vertex closest to the sample (Lines 2-3); if the extension is collision-free (Line 4), then the end point of the extension, say z_{new}, is added to the tree as a new vertex (Line 5), as in RRT. However, the RRT* $\texttt{Extend}_\alpha$ procedure differs from others in that it connects the new vertex z_{new} to the vertex that lies within a ball of volume $\Theta(\log(n)/n)$ centered at z_{new}, where $n = |V|$ is the number of vertices in the tree, and incurs the smallest cost to reach z_{new} with a collision-free trajectory (Lines 8-12). Moreover, the RRT* $\texttt{Extend}_\alpha$ procedure extends z_{new} back to the vertices in the tree that are within the ball of same size centered at z_{new}; if the extension to such a vertex, say z_{near}, results in a collision-free trajectory that reaches z_{near} with smaller cost, then tree is "rewired" by connecting z_{near} to z_{new}, instead of its current parent (Lines 13 - 18).

The algorithm that is proposed for the solution of the problem in Section 2 builds on RRT*, and relies on the following additional primitive procedures.

Near-Capture Vertices: The $\texttt{NearCapure}_\alpha$ procedure works in a way that is very similar to the $\texttt{Near}_\alpha$ procedure. Given a tree $G = (V,E)$, a state $z \in X_\alpha$, and a number n, the $\texttt{NearCapture}_\alpha(G,z,n)$ procedure returns all vertices z' that are "close" to being captured from z. In other words, and assuming $\alpha = \text{p}$ for simplicity, let $\texttt{CaptureSet}_\text{p}(z) := \{z' \in X_\text{e} : (z',z) \in X_{\text{capt}}\}$. Then, $\texttt{NearCapture}_\text{p}(G,z,n) = \{v \in V \,|\, \text{there exist } y \in \texttt{CaptureSet}_\text{p}(z) \text{ such that } v \in \texttt{Reach}_\text{e}(y, l(n))\}$.

Remove: Given a graph $G = (V,E)$ on X_α, and a vertex $z \in V$, the procedure $\texttt{Remove}(G,z)$ removes z, all its descendants, and their incoming edges from G.

The algorithm proposed to solve the pursuit-evasion game under consideration is given in Algorithm 3. The algorithm maintains two tree structures encoding candidate paths: the evader tree G_e and the pursuer tree G_p. At each iteration, the algo-

Algorithm 1: The RRT* Algorithm

1 $V_e \leftarrow \{z_{init}\}$; $E_e \leftarrow \emptyset$; $i \leftarrow 0$;
2 **while** $i < N$ **do**
3 $\quad G_e \leftarrow (V_e, E_e)$;
4 $\quad z_{e,rand} \leftarrow \texttt{Sample}_e(i)$;
5 $\quad (V_e, E_e, z_{e,new}) \leftarrow \texttt{Extend}_e(G_e, z_{e,rand})$;
6 $\quad i \leftarrow i+1$;

Algorithm 2: $\texttt{Extend}_\alpha(G, z)$

1 $V' \leftarrow V$; $E' \leftarrow E$;
2 $z_{nearest} \leftarrow \texttt{Nearest}_\alpha(G, z)$;
3 $(x_{new}, u_{new}, t_{new}) \leftarrow \texttt{Steer}_\alpha(z_{nearest}, z)$; $z_{new} \leftarrow x_{new}(t_{new})$;
4 **if** $\texttt{ObstacleFree}_\alpha(x_{new})$ **then**
5 $\quad V' \leftarrow V' \cup \{z_{new}\}$;
6 $\quad z_{min} \leftarrow z_{nearest}$; $c_{min} \leftarrow T(z_{new})$;
7 $\quad Z_{nearby} \leftarrow \texttt{Near}_\alpha(G, z_{new}, |V|)$;
8 $\quad$ **for** *all* $z_{near} \in Z_{nearby}$ **do**
9 $\quad\quad (x_{near}, u_{near}, t_{near}) \leftarrow \texttt{Steer}_\alpha(z_{near}, z_{new})$;
10 $\quad\quad$ **if** $\texttt{ObstacleFree}_\alpha(x_{near})$ *and* $x_{near}(t_{near}) = z_{new}$ *and* $T(z_{near}) + \texttt{EndTime}(x_{near}) < T(z_{new})$ **then**
11 $\quad\quad\quad c_{min} \leftarrow T(z_{near}) + \texttt{EndTime}(x_{near})$;
12 $\quad\quad\quad z_{min} \leftarrow z_{near}$;
13 $\quad E' \leftarrow E' \cup \{(z_{min}, z_{new})\}$;
14 $\quad$ **for** *all* $z_{near} \in Z_{nearby} \setminus \{z_{min}\}$ **do**
15 $\quad\quad (x_{near}, u_{near}, t_{near}) \leftarrow \texttt{Steer}_\alpha(z_{new}, z_{near})$;
16 $\quad\quad$ **if** $\texttt{ObstacleFree}_\alpha(x_{near})$ *and* $x_{near}(t_{near}) = z_{near}$ *and* $T(z_{near}) > T(z_{new}) + \texttt{EndTime}(x_{near})$ **then**
17 $\quad\quad\quad z_{parent} \leftarrow \texttt{Parent}(z_{near})$;
18 $\quad\quad\quad E' \leftarrow E' \setminus \{(z_{parent}, z_{near})\}$; $E' \leftarrow E' \cup \{(z_{new}, z_{near})\}$;
19 **else**
20 $\quad z_{new} = \text{NULL}$;
21 **return** $G' = (V', E', z_{new})$

rithm first samples a state, $z_{e,rand}$, in the evader's state-space (Line 4) and extends the evader tree towards $z_{e,rand}$ (Line 5). If the extension produces a new vertex $z_{e,new}$ (Line 6), then the algorithm checks whether the time that the evader reaches $z_{e,new}$ is less than that at which the pursuer reaches any pursuer vertex within certain distance to $z_{e,new}$ (Lines 7-10). This distance scales as $\Theta(\log(n)/n)$, where n is the number of vertices in the pursuer tree, G_p. If this condition does not hold, then $z_{e,new}$ is removed from evader's tree (Line 10).

Second, the algorithm samples a new state, $z_{p,rand}$, in the pursuer state space (Line 11) and extends the pursuer's tree towards $z_{p,rand}$ (Line 12). If this extension successfully produces a new vertex, $z_{p,new}$ (Line 13), then the algorithm checks whether the evader can reach any of the evader vertices that lie within a certain distance to $z_{p,new}$ in less time than the pursuer can reach $z_{p,new}$ (Lines 14-17). Any

evader vertex that is within a certain distance to $z_{\mathrm{p,new}}$ and that does not satisfy this requirement is removed from the tree with its descendants (Line 17). The distance scales as $\Theta(\log(n)/n)$, where n is the number of vertices in the evader's tree, G_{e}.

The algorithm returns two trees, namely G_{e} and G_{p}. From the evader's tree G_{e}, the control strategy that makes the evader reach X_{goal} in minimum time (if one exists) is the solution candidate after N iterations.

Algorithm 3: Pursuit-Evasion Algorithm

```
1  V_e ← {x_e,init}; E_e ← ∅; V_p ← {x_p,init}; E_p ← ∅; i ← 0;
2  while i < N do
3      G_e ← (V_e, E_e); G_p ← (V_p, E_p);
4      z_e,rand ← Sample_e(i);
5      (V_e, E_e, z_e,new) ← Extend_e(G_e, z_e,rand);
6      if z_e,new ≠ NULL then
7          Z_p,near ← NearCapture_e(G_p, z_e,new, |V_p|);
8          for all z_p,near ∈ Z_P,near do
9              if Time(z_p,near) ≤ Time(z_e,new) then
10                 Remove(G_e, z_e,new);
11     z_p ← Sample_p(i);
12     (V_p, E_p, z_p,new) ← Extend_p(G_p, z_p,rand) ;
13     if z_p,new ≠ NULL then
14         Z_e,near ← NearCapture_p(G_e, z_p,new, |V_e|);
15         for all z_e,near ∈ Z_e,near do
16             if Time(z_p,new) ≤ Time(z_e,near) then
17                 Remove(G_e, z_e,near);
18     i ← i + 1;
19 return G_e, G_p
```

4 Analysis

In this section, theoretical guarantees of the algorithm are briefly outlined. Due to lack of space, detailed proofs of the results are left to a full version of this paper.

Let us note the following technical assumptions, which we will assume throughout this section without reference. Firstly, it is assumed that the dynamical systems modeling the evader and the pursuer independently satisfy local controllability properties. Secondly, we will assume that there exists a Stackelberg strategy for the pursuit-evasion game with finite value of the game L^*, and such that sufficiently

small perturbations to the strategy also yield a finite value. A formal statement of these assumptions can be found (for the optimal motion planning case) in [39].

First, note the following lemma stating the optimality property of the RRT* algorithm (Algorithm 1) when the algorithm is used to solve a time-optimal kinodynamic motion planning problem. Let $G[i] = (V[i], E[i])$ denote the tree maintained by the RRT* algorithm at the end of iteration i. Given a state $z \in X$, let $T^*(z)$ denote the time an optimal collision-free trajectory reaches z, i.e., $T^*(z) := \inf_u \{T \,|\, x(T) = z \text{ and } \dot{x}(t) = f(x(t), u(t)), x(t) \notin X_{\text{obs}} \text{ for all } t \in [0,T]\}$. Let $z \in V[j]$ be a vertex that is in the tree at iteration j. The time that the unique trajectory that is in $G[i]$ for some $i \in \mathbb{N}$ and that starts from the root vertex and reaches z is denoted by $T(z)[i]$.

The following theorem follows directly from the asymptotic optimality of the RRT* algorithm shown in [39]. Let $\mu(\cdot)$ denote the Lebesgue measure.

Theorem 1 (Asymptotic Optimality of RRT* [39]). *If $\gamma > 2^d (1 + 1/d)\, \mu(X \setminus X_{\text{obs}})$, the event that for any vertex z that is in the tree in some finite iteration j the RRT* algorithm converges to a trajectory that reaches z optimally, i.e., in time $T^*(z)$, occurs with probability one. Formally,*

$$\mathbb{P}\big(\big\{ \lim_{i\to\infty} T(z)[i+j] = T^*(z), \quad \forall z \in V[j] \big\}\big) = 1, \qquad \forall j \in \mathbb{N}.$$

Let $T_\alpha(z_\alpha)[i]$ denote the time at which the vertex z_α in $V_\alpha[i]$ is reached, for $\alpha \in \{\mathrm{e}, \mathrm{p}\}$, and let $T^*_\alpha(z_\alpha)$ be the time the time-optimal collision-free trajectory reaches z_α (disregarding the other agent). Theorem 1 translates to the evader tree in a weaker form:

Lemma 1. *Under the assumptions of Theorem 1, applied to the evader tree,*

$$\mathbb{P}\big(\big\{ \lim_{i\to\infty} T_{\mathrm{e}}(z_{\mathrm{e}})[i+j] \geq T^*(z_{\mathrm{e}}), \quad \forall z_{\mathrm{e}} \in V[j] \big\}\big) = 1, \quad \forall j \in \mathbb{N}.$$

Lemma 1 follows directly from Theorem 1 noting that the evader's tree can only include fewer edges (due to removal of evader's vertices near capture), when compared to the standard RRT* algorithm.

A similar property can be shown in terms of capture time estimates. Given $z_{\mathrm{e}} \in X_{\mathrm{e}}$, define $\texttt{CaptureSet}_{\mathrm{e}}(z_{\mathrm{e}})$ as the set of all states in X_{p} reaching which the pursuer can capture the evader, and let $C^*(z_{\mathrm{e}})$ denote the minimum time at which this capture can occur, i.e., $C^*(z_{\mathrm{e}}) := \inf_{u_{\mathrm{p}}} \big\{ T \;\big|\; x_{\mathrm{p}}(T) \in \texttt{CaptureSet}_{\mathrm{p}}(z_{\mathrm{e}}) \big\}$.

Lemma 2. *Let $C_{\mathrm{p}}(z_{\mathrm{e}})[i] := \min \big\{ T_{\mathrm{p}}(z_{\mathrm{p}})[i] \;\big|\; z_{\mathrm{p}} \in \texttt{NearCapture}_{\mathrm{e}}(G_{\mathrm{p}}[i], z_{\mathrm{e}}, i) \big\}$. Then, under the assumptions of Theorem 1, applied to the pursuer tree,*

$$\mathbb{P}\big(\big\{ \lim_{i\to\infty} C_{\mathrm{p}}(z_{\mathrm{e}})[i] = C^*(z_{\mathrm{e}}) \big\}\big) = 1, \quad \forall z_{\mathrm{e}} \in X_{\mathrm{e}}.$$

Proof (Sketch). Let the set $\texttt{DomainNearCapture}_{\mathrm{e}}(z, n)$ be defined as $\{z_{\mathrm{p}} \in X_{\mathrm{p}} \,|\, \exists y \in \texttt{CaptureSet}_{\mathrm{e}}(z), z_{\mathrm{p}} \in \texttt{Reach}_{\mathrm{p}}(y, l(n))\}$, where $l(n)$ was introduced in the definition of the `NearCapture` procedure. Note that (i) $\texttt{DomainNearCapture}_{\mathrm{e}}(G_{\mathrm{p}}[i], z_{\mathrm{e}}, i) \supseteq \texttt{CaptureSet}_{\mathrm{e}}(z_{\mathrm{e}})$ for all $i \in \mathbb{N}$, and (ii) $\bigcap_{i\in\mathbb{N}} \texttt{DomainNearCapture}_{\mathrm{e}}(G_{\mathrm{p}}[i], z_{\mathrm{e}}, i) =$

$\texttt{CaptureSet}_\text{e}(z_\text{e})$. Thus, the set $\texttt{DomainNearCapture}_\text{e}(G_\text{p}[i], z_\text{e}, i)$ converges to $\texttt{CaptureSet}_\text{e}(z_\text{e})$ from above as $i \to \infty$. Let $X^*_\text{capt}(z_\text{e})$ be the subset of $\texttt{CaptureSet}_\text{e}(z_\text{e})$ that the pursuer can reach within time $C^*(z_\text{e})$. The key to proving this lemma is to show that the set $\texttt{DomainNearCapture}_\text{e}(G_\text{p}[i], z_\text{e}, i)$ is sampled infinitely often so as to allow the existence of a sequence of vertices that converges to a state in X^*_capt. Then, for each vertex in the sequence, by Theorem 1, the RRT* algorithm will construct trajectories that converge to their respective optimal trajectory almost surely, which implies the claim.

To show that the sets $\texttt{DomainNearCapture}_\text{e}(G_\text{p}[i], z_\text{e}, i)$ are sampled infinitely often as $i \to \infty$, note that the probability that there is no sample inside the set $\texttt{DomainNearCapture}(G_\text{p}[i], z_\text{e}, i)$ is $(1 - \frac{\gamma_\text{capt}}{\mu(X_\text{p})} \frac{\log i}{i})^i$. In addition, $\sum_{i \in \mathbb{N}} \left(1 - \frac{\gamma_\text{capt}}{\mu(X_\text{p})} \frac{\log i}{i}\right)^i \leq \sum_{i \in \mathbb{N}} (1/i)^{\frac{\gamma_\text{capt}}{\mu(X_\text{p})}}$ is finite for $\gamma_\text{capt} > \mu(X_\text{p})$. Thus, by the Borel-Cantelli lemma [40], the event that there are no samples inside $\texttt{NearCapture}(G_\text{p}[i], z_\text{e}, i)$ occurs only finitely often with probability one; hence, the same sequence of sets is sampled infinitely often with probability one. □

The next lemma states that all vertices satisfy the soundness property.

Lemma 3. *Let B_j denote the following event: for any vertex z_e that is in evader's tree by the end of iteration j, if the pursuer can reach z_e before the evader, then $C_\text{p}(z_\text{e})[i]$ converges to a value that is smaller than the value that $T_\text{e}(z_e)[i]$ converges to as i approaches infinity, i.e., $B_j := \{((C^*(z_\text{e}) \leq T^*(z_\text{e})) \Rightarrow (\lim_{i\to\infty} C_\text{p}(x_\text{e}) \leq \lim_{i\to\infty} T_\text{e}(z_\text{e})), \forall z_e \in V_e[j]\}$. Then, $\mathbb{P}(B_j) = 1$ for all $j \in \mathbb{N}$.*

Proof. Fix some $j \in \mathbb{N}$. Consider the events $\{\lim_{i\to\infty} T_e(z_e)[i+j] \geq T^*(z_e), \forall z_e \in V_e[j]\}$ and $\{\lim_{i\to\infty} C_p(z_e)[i+j] = C^*(z_e)\}$, both of which occur with probability one by Lemmas 1 and 2, respectively. Hence, their intersection occurs with probability one, i.e.,

$$\mathbb{P}\left(\left\{\lim_{i\to\infty} T_e(z_e)[i+j] \geq T^*(z_e) \wedge \lim_{i\to\infty} C_p(z_e)[i+j] = C^*(z_e),\ \forall z_e \in V_e[j]\right\}\right) = 1.$$

Finally, $\lim_{i\to\infty} T_e(z_e)[i+j] \geq T^*(z_e) \wedge \lim_{i\to\infty} C_p(z_e)[i+j] = C^*(z_e)$ logically implies $((C^*(z_\text{e}) \leq T^*(z_\text{e})) \Rightarrow (\lim_{i\to\infty} C_\text{p}(x_\text{e}) \leq \lim_{i\to\infty} T_\text{e}(z_\text{e}))$. Substituting the latter in place of the former in the equation above yields the result. □

Let $x_\text{e}[i]$ denote the trajectory that is in evader's tree, $G_e[i]$, by the end of iteration i and that reaches the goal region in minimum time. Recall that T^* is the ending time of the minimum-time collision-free trajectory that reaches the goal region and avoids capture.

The next theorem states the probabilistic soundness of Algorithm 3. That is, the probability that any evader strategy returned by the algorithm is sound (i.e., avoids capture by the pursuer) approaches one as the number of samples increases. More precisely, for all $\varepsilon > 0$ and all $t \in [0, T^*]$, the probability that the state $x_\text{e}[i](t)$ avoids capture, if the pursuer is delayed for ε units of time in the beginning of the game, approaches one as $i \to \infty$.

Theorem 2 (Probabilistic Soundness). *Let $A_{\varepsilon,t}[i]$ denote the event $\{t < C^*(x[i](t)) + \varepsilon\}$. Then, $\lim_{i\to\infty} \mathbb{P}(A_{\varepsilon,t}[i]) = 1$, for all $\varepsilon > 0$ and all $t \in [0,T]$.*

Proof (Sketch). Let $\mathscr{Z}[j] = \{z_1, x_2, \ldots, z_K\} \subseteq V_e[j]$ denote the set of all vertices in the evaders tree that are along the path $x_e[j]$. Let $\mathscr{T}[j] = \{t_1, t_2, \ldots, t_K\}$ denote the corresponding time instances, i.e., $z_k = x_e[t_j](t_k)$ for all $k \in \{1,2,\ldots,K\}$. By Lemma 3, the theorem holds for the time instances corresponding to the states in $\mathscr{Z}[j]$. However, it must also be shown that the same holds for all trajectories that connect consecutive states in $\mathscr{Z}[j]$. Such trajectories are referred to as intermediate trajectories from here on.

Let $t_{\max}[i] := \max_{t_k, t_{k+1} \in \mathscr{T}[i]} (t_{k+1} - t_k)$. The algorithm provided in this paper does not check the soundness of intermediate trajectories, but checks only that of the vertices. However, it can be shown that for any $\varepsilon > 0$, $\lim_{i\to\infty} \mathbb{P}(\{t_{\max}[i] < \varepsilon\}) = 1$. Roughly speaking, with probability one, the time-optimal path is never achieved, but the algorithm converges towards that optimal as the number of samples approaches infinity. Since each intermediate path that is along $x_e[j]$ is sub-optimal with probability one, in the process of convergence it is replaced with a lower cost path that includes two or more vertices of the tree in some later iteration $i > j$.

Since $t_{\max}[i] < \varepsilon$ logically implies that $t < C^*(x[i](t)) + \varepsilon$ for all $t \in [0,T]$, $\{t_{\max}[i] < \varepsilon\} \subseteq \{t < C^*(x[i](t)) + \varepsilon,\ \forall t \in [0,T]\}$, which implies $\mathbb{P}(\{t_{\max}[i] < \varepsilon\}) \leq \mathbb{P}(\{t < C^*(x[i](t)) + \varepsilon\})$. Taking the limit of both sides yields the result. □

Let us also note the following theorems regarding the probabilistic completeness and asymptotic optimality of the algorithm. The proofs of these theorems are rather straightforward and are omitted due to lack of space.

Theorem 3 (Probabilistic Completeness). *Under the assumptions of Theorem 1, Algorithm 3 finds a trajectory that reaches the goal region while avoiding collision with obstacles and capture by pursuers, if such a trajectory exists, with probability approaching one as the number of samples approaches infinity.*

Theorem 4 (Asymptotic Optimality). *Let $L[i]$ be the cost of the minimum-time trajectory in the evader's tree at the end of iteration i that reaches the goal region, if any is available, and $+\infty$ otherwise. Then, under the assumptions of Theorem 1, $L[i]$ converges to the value of the pursuit-evasion game, L^*, almost surely.*

5 Simulation Examples

In this section, two simulation examples are presented. In the first example, an evader modeled as a single integrator with velocity bounds is trying to reach a goal set, while avoiding capture by three pursuers, each of which is modeled as a single integrator with different velocity bounds. More precisely, the differential equation describing the dynamics of the evader can be written as follows:

$$\frac{d}{dt}x_{\mathrm{e}}(t) = \frac{d}{dt}\begin{bmatrix} x_{\mathrm{e},1}(t) \\ x_{\mathrm{e},2}(t) \end{bmatrix} = u_{\mathrm{e}}(t) = \begin{bmatrix} u_{\mathrm{e},1}(t) \\ u_{\mathrm{e},2}(t) \end{bmatrix},$$

where $\|u_{\mathrm{e}}(t)\|_2 \leq 1$. The dynamics of the pursuer is written as follows:

$$\frac{d}{dt}x_{\mathrm{p}}(T) = \frac{d}{dt}\begin{bmatrix} x_{\mathrm{p}_1}(t) \\ x_{\mathrm{p}_2}(t) \\ x_{\mathrm{p}_3}(t) \end{bmatrix} = \frac{d}{dt}\begin{bmatrix} x_{\mathrm{p}_1,1}(t) \\ x_{\mathrm{p}_1,2}(t) \\ x_{\mathrm{p}_2,1}(t) \\ x_{\mathrm{p}_2,2}(t) \\ x_{\mathrm{p}_3,1}(t) \\ x_{\mathrm{p}_3,2}(t) \end{bmatrix} = u_{\mathrm{p}}(t) = \begin{bmatrix} u_{\mathrm{p}_1}(t) \\ u_{\mathrm{p}_2}(t) \\ u_{\mathrm{p}_3}(t) \end{bmatrix} = \begin{bmatrix} u_{\mathrm{p}_1,1}(t) \\ u_{\mathrm{p}_1,2}(t) \\ u_{\mathrm{p}_2,1}(t) \\ u_{\mathrm{p}_2,2}(t) \\ u_{\mathrm{p}_3,1}(t) \\ u_{\mathrm{p}_3,2}(t) \end{bmatrix},$$

where $\|u_{\mathrm{p}_1}(t)\|_2 \leq 1$ and $\|u_{\mathrm{p}_k}(t)\|_2 \leq 0.5$ for $k \in \{2,3\}$.

First, a scenario that involves an environment with no obstacles is considered. The evader's trajectory is shown in Figures 1(a)-1(d) in several stages of the algorithm. The algorithm quickly identifies an approximate solution that reaches the goal and stays away from the pursuers. The final trajectory shown in Figure 1(d) goes towards the goal region but makes a small deviation to avoid capture. The same scenario is considered in an environment involving obstacles and the evader's tree is shown in different stages in Figure 2(a)-2(d). Notice that the evader may choose to "hide behind the obstacles" to avoid the pursuers, as certain parts of the state space that are not reachable by the evader are reachable in presence of obstacles.

In the second example, the motion of the pursuer and of the evader is described by a simplified model of aircraft kinematics. Namely, the projection of the vehicle's position on the horizontal plane is assumed to follow the dynamics of a Dubins vehicle (constant speed and bounded curvature), while the altitude dynamics is modeled as a double integrator. The differential equation describing dynamics of the evader is given as follows. Let $x_{\mathrm{e}}(t) = [x_{\mathrm{e},1}(t), x_{\mathrm{e},2}(t), x_{\mathrm{e},3}(t), x_{\mathrm{e},4}(t), x_{\mathrm{e},5}(t)]^T$ and $f(x_{\mathrm{e}}(t), u_{\mathrm{e}}(t)) = [v_{\mathrm{e}} \cos(x_{\mathrm{e},3}(t)), v_{\mathrm{e}} \sin(x_{\mathrm{e},3}(t)), u_{\mathrm{e},1}(t), x_{\mathrm{e},5}(t), u_{\mathrm{e},2}(t)]^T$, and $\dot{x}_{\mathrm{e}}(t) = f(x_{\mathrm{e}}(t), u_{\mathrm{e}}(t))$, where $v_{\mathrm{e}} = 1$, $|u_{\mathrm{e},1}(t)| \leq 1$, $|u_{\mathrm{e},2}(t)| \leq 1$, $|x_{\mathrm{e},5}| \leq 1$. In this case, v_{e} denotes the longitudinal velocity of the airplane, $u_{\mathrm{e},1}$ denotes the steering input, and $u_{\mathrm{e},2}$ denotes the vertical acceleration input. The pursuer dynamics is the same, except the pursuer moves with twice the speed but has three times the minimum turning radius when compared to the evader, i.e., $v_{\mathrm{p}} = 2$, $|u_{\mathrm{p},1}| \leq 1/3$.

A scenario in which the evader starts behind pursuer and tries to get to a goal set right next to the pursuer is shown in Figure 3. First, an environment with no obstacles is considered and the tree maintained by the evader is shown in Figure 3(a), at end of 3000 iterations. Notice that the evader tree does not include a trajectory that can escape to the goal set (shown as a green box). Second, the same scenario is run in an environment involving obstacles. The trees maintained by the evader is shown in Figure 3(b). Note that the presence of the big plate-shaped obstacle prevents the pursuer from turning left directly, which allows the evader to reach a larger set of states to the left without being captured. In particular, the evader tree includes trajectories reaching the goal.

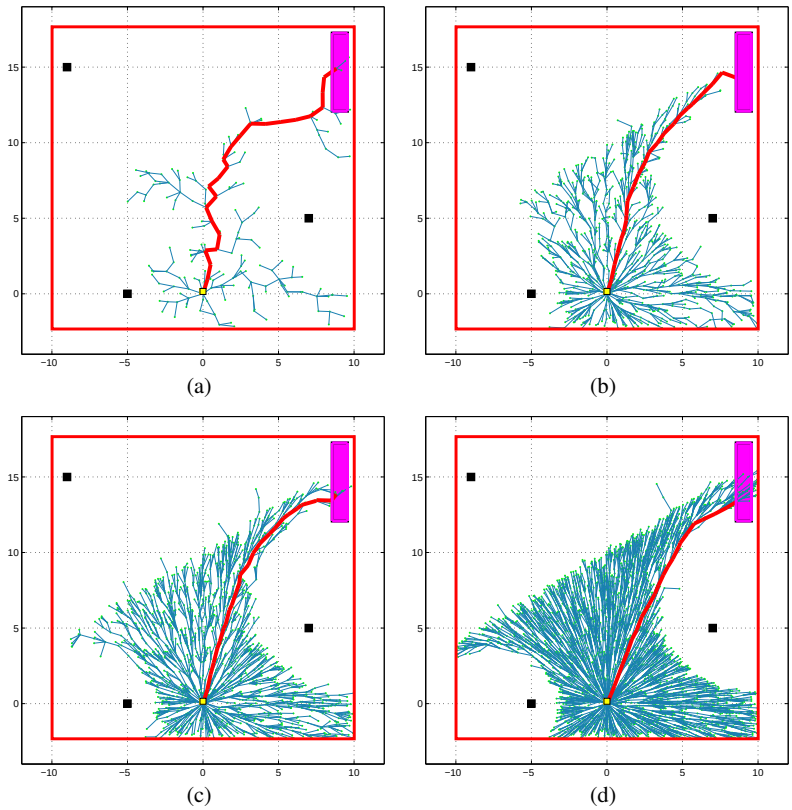


Fig. 1 The evader trajectory is shown in an environment with no obstacles at the end of 500, 3000, 5000, and 10000 iterations in Figures (a), (b), (c), and (d), respectively. The goal region is shown in magenta. Evader's initial condition is shown in yellow and the pursuers' initial conditions are shown in black. The first pursuer, P_1, which can achieve the same speed that the evader can achieve, is located in top left of the figure. Other two pursuers can achieve only half the evader's speed.

Simulation examples were solved on a laptop computer equipped with a 2.33 GHz processor running the Linux operating system. The algorithm was implemented in the C programming language. The first example took around 3 seconds to compute, whereas the second scenario took around 20 seconds.

6 Conclusions

In this paper, a class of pursuit-evasion games, which generalizes a broad class of motion planning problems with dynamic obstacles, is considered. A computationally efficient incremental sampling-based algorithm that solves this problem with

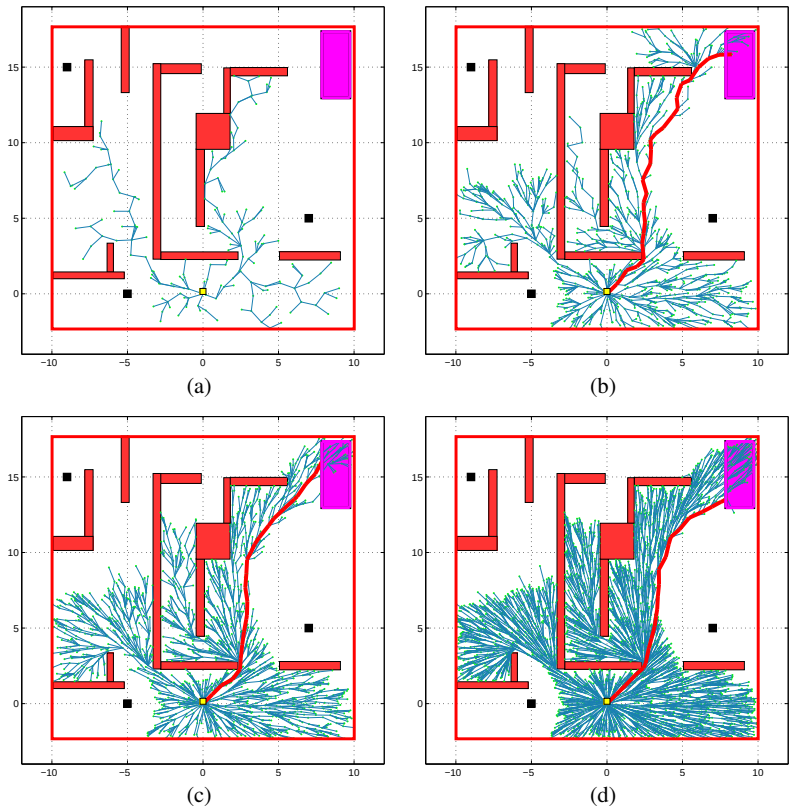
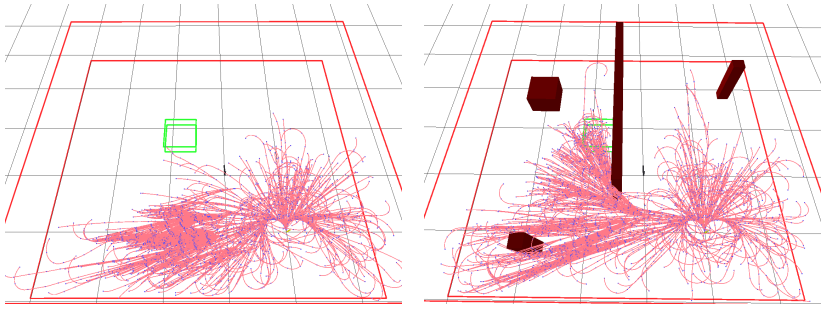


Fig. 2 The scenario in Figure 1 is run in an environment with obstacles.

probabilistic guarantees is provided. The algorithm is also evaluated with simulation examples. To the best of authors' knowledge this algorithm constitutes the first incremental sampling-based algorithm as well as the first anytime algorithm for solving pursuit-evasion games. Anytime flavor of the algorithm provides advantage in real-time implementations when compared to other numerical methods.

Although incremental sampling-based motion planning methods have been widely used for almost a decade for solving motion planning problems efficiently, almost no progress was made in using similar methods to solve differential games. Arguably, this gap has been mainly due to the inability of these algorithms to generate optimal solutions. The RRT* algorithm, being able to almost-surely converge to optimal solutions, comes as a new tool to efficiently solve complex optimization problems such as differential games. In this paper, we have investigated a most basic version of such a problem. Future work will include developing algorithms that converge to, e.g., feedback saddle-point equilibria of pursuit-evasion games, as well as relaxing the separability assumption on the dynamics to address a larger class of problems.



(a) (b)

Fig. 3 Figures (a) and (b) show the trees maintained by the evader at end of the 3000th iteration in an environment without and with obstacles, respectively. The initial state of the evader and the pursuer are marked with a yellow pole (at bottom right of the figure) and a black pole (at the center of the figure), respectively. Each trajectory (shown in purple) represents the set of states that the evader can reach safely (with certain probability approaching to one).

References

1. R. Isaacs. *Differential Games*. Wiley, 1965.
2. T. Basar and G.J. Olsder. *Dynamic Noncooperative Game Theory*. Academic Press, 2nd edition, 1995.
3. 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.
4. R. Lachner, M. H. Breitner, and H. J. Pesch. Three dimensional air combat: Numerical solution of complex differential games. In G. J. Olsder, editor, *New Trends in Dynamic Games and Applications*, pages 165–190. Birkhäuser, 1995.
5. O. Vanek, B. Bosansky, M. Jakob, and M. Pechoucek. Transiting areas patrolled by a mobile adversary. In *Proc. IEEE Conf. on Computational Intelligence and Games*, 2010.
6. M. Simaan and J. B. Cruz. On the Stackelberg strategy in nonzero-sum games. *Journal of Optimization Theory and Applications*, 11(5):533–555, 1973.
7. M. Bardi, M. Falcone, and P. Soravia. Numerical methods for pursuit-evasion games via viscosity solutions. In M. Barti, T.E.S. Raghavan, and T. Parthasarathy, editors, *Stochastic and Differential Games: Theory and Numerical Methods*. Birkhäuser, 1999.
8. H. Ehtamo and T. Raivio. On applied nonlinear and bilevel programming for pursuit-evasion games. *Journal of Optimization Theory and Applications*, 108:65–96, 2001.
9. M.H. Breitner, H. J. Pesch, and W. Grimm. Complex differential games of pursuit-evasion type with state constraints, part 1: Necessary conditions for optimal open-loop strategies. *Journal of Optimization Theory and Applications*, 78:419–441, 1993.
10. M.H. Breitner, H. J. Pesch, and W. Grim. Complex differential games of pursuit-evasion type with state constraints, part 2: Numerical computation of optimal open-loop strategies. *Journal of Optimization Theory and Applications*, 78:444–463, 1993.
11. R.D. Russell and L.F. Shampine. A collocation method for boundary value problems. *Numerische Mathematik*, 19(1):1–28, 1972.
12. O. van Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37:357–373, 1992.
13. J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Surfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1996.

14. J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.
15. S. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
16. S. Petti and Th. Fraichard. Safe motion planning in dynamic environments. In *Proc. Int. Conf. on Intelligent Robots and Systems*, 2005.
17. D. Hsu, R. Kindel, J. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21(3):233–255, 2002.
18. J. Reif and M. Sharir. Motion planning in presence of moving obstacles. In *Proceedings of the 25th IEEE Symposium FOCS*, 1985.
19. M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica*, 2:477–521, 1987.
20. K. Sutner and W. Maass. Motion planning among time dependent obstacles. *Acta Informatica*, 26:93–122, 1988.
21. K. Fujimura and H. Samet. Planning a time-minimal motion among moving obstacles. *Algorithmica*, 10:41–63, 1993.
22. M. Farber, M. Grant, and S. Yuzvinski. Topological complexity of collision free motion planning algorithms in the presence of multiple moving obstacles. In M. Farber, G. Ghrist, M. Burger, and D. Koditschek, editors, *Topology and Robotics*, pages 75–83. American Mathematical Society, 2007.
23. M. Zucker, J. Kuffner, and M. Branicky. Multipartite RRTs for rapid replanning in dynamic environments. In *IEEE Conference on Robotics and Automation (ICRA)*, 2007.
24. J. van der Berg and M. Overmars. Planning the shortest safe path amidst unpredictably moving obstacles. In *Workshop on Algorithmic Foundations of Robotics VII*, 2008.
25. T. Parsons. Pursuit-evasion in a graph. In *Theory and Applications of Graphs*, pages 426–441. 1978.
26. Volkan Isler, Sampath Kannan, and Sanjeev Khanna. Randomized pursuit-evasion with limited visibility. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1060–1069, New Orleans, Louisiana, 2004. Society for Industrial and Applied Mathematics.
27. M. Adler, H. Räcke, N. Sivadasan, C. Sohler, and B. Vöcking. Randomized pursuit-evasion in graphs. *Combinatorics, Probability and Computing*, 12(03):225–244, 2003.
28. I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on computing*, 21:863, 1992.
29. B. Gerkey, S. Thrun, and G. Gordon. Clear the building: Pursuit-evasion with teams of robots. In *Proceedings AAAI National Conference on Artificial Intelligence*, 2004.
30. V. Isler, S. Kannan, and S. Khanna. Locating and capturing an evader in a polygonal environment. *Algorithmic Foundations of Robotics VI*, pages 251–266, 2005.
31. V. Isler, D. Sun, and S. Sastry. Roadmap based pursuit-evasion and collision avoidance. In *Robotics: Science and Systems Conference*, 2005.
32. L.E. Kavraki, P. Svestka, J Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
33. S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.
34. Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J.P. How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems*, 17(5):1105–1118, 2009.
35. S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *Robotics: Science and Systems (RSS)*, 2010.
36. T. Dean and M. Boddy. Analysis of time-dependent planning. *Proceedings of AAAI*, 1989.
37. S. Zilberstein and S. Russell. *Imprecise and Approximate Computation*, volume 318, chapter Approximate Reasoning Using Anytime Algorithms, pages 43–62. Springer, 1995.
38. J. Canny. *The Complexity of Robot Motion Planning*. MIT Press, 1988.
39. S. Karaman and E. Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. In *IEEE Conf. on Decision and Control*, Atlanta, GA, 2010.
40. G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, Third edition, 2001.