Pursuit Planning with Spatial Action Abstraction

Anonymous

Abstract

The Pursuit-Evasion (PE) problem is a well-known problem in which an agent – the Pursuer – attempts to catch another agent – the Evader – before it reaches some target destination. We explore a variant of the PE problem called PE with Fixed Evader plans (PE-FEP), in which an external opponent modeling tool provides a distribution over the plans the Evader agent may select. This PE variant, which is motivated by real-life problems, is particularly challenging for existing PE algorithms when the agents operate in a large-scale 3D grid. In this paper, we explore how techniques from the automated planning literature can be adapted and extended to effectively solve PE-FEP. We propose three ways to model PE-FEP as a Markov Decision Process (MDP): two modeling that approximate the problem and one that fully describes the problem. To solve this large MDP, we employ the well-known Real-Time Dynamic Programming (RTDP) algorithm and develop heuristics to initialize it for each of our MDP modeling scheme. Then, we propose novel techniques for automatically generating abstract, higher-level, actions and show how to use it inside the RTDP framework. Experimental results on large grids with up to $3 \cdot 10^6$ cells, and with sets of up to 30 possible Evader plans, show that our algorithms can scale gracefully with the size of the grid and number of Evader plans.

Introduction

In the Pursuit Evasion (PE) problem, one agent, a Pursuer, attempts to catch a second agent, an Evader. PE problems were explored in many areas of research, ranging from theoretical computer science to applied robotics (Chung, Hollinger, and Isler 2011; Cheng 2003; Hespanha, Kim, and Sastry 1999). In this paper, we explore a specific PE variant called PE with Fixed Evader plans (PE-FEP) problem, which has important real-world applications. In PE-FEP, the agents operate in a large empty 3D grid, with movements bounded by physical constraints. The Pursuer and the Evader have different abilities. The Evader is faster than the Pursuer but has no sensors, and cannot observe the Pursuer. As such, the Evader uses a non-reactive policy, selecting a plan from a set of predefined plans and following it until either reaching its target, or getting caught by the Pursuer. In contrast, the Pursuer has perfect sensors, enabling it to constantly track the position of the Evader and adjust its velocity accordingly. The task is to find a policy for the Pursuer so that it catches the Evader. The main contribution of this paper is a PE-FEP algorithm that can scale up, producing policies for large grids with many possible Evader plans. Our algorithms work by modeling PE-FEP as a Markov Decision Process (MDP), and solving this MDP using the Real-Time Dynamic Programming (RTDP) algorithm (Barto, Bradtke, and Singh 1995) with novel enhancements.

The first challenge we address when developing our PE-FEP algorithm is modeling. We explore MDP models and analyze them, showing a tradeoff between the size of the state space and the fidelity of the model. Notably, we propose a belief-based MDP that compactly represents a sufficient statistic of the agent's belief about the Evader's plans.

The second challenge we address when developing our PE-FEP algorithm is scalability. While RTDP is a highly-effective MDP solver that can converge to an optimal policy without iterating over the entire state space, it cannot scale to the size of problems we aim to solve. To this end, we propose an spatial action abstraction mechanism based on the Options framework (Sutton, Precup, and Singh 1999). In our spatial abstraction, each abstract action is composed of a sequence of basic actions designed to achieve a given velocity and follow it for a time. Key to the effectiveness of this abstraction is that the length of our abstract actions is adaptive, allowing longer sequences when the Pursuer is farther from the Evader and shorter sequences when it is closer. In addition, we introduce several heuristics that speed up RTDP by initializing its Q function intelligently. Using heuristics to initialize the RTDP value function is known to improve performance, but for RTDP to converge, such a heuristic must never underestimate the value returned by the optimal value function. We propose such a heuristic that is based on analyzing the potential future steps of the Evader and provide a sufficient condition for their admissibility.

We evaluated different configurations of the algorithm experimentally on grid sizes of up to $550 \times 200 \times 50$ positions, with up to 60 different Evader plans. These problems have about 1215×66^9 possible states, yet we solve them in less than two minutes. We analyze the contribution of each of our suggested modifications, showing that a naive RTDP implementation cannot scale beyond a $30 \times 30 \times 5$ grid, while we scale to the grid sizes described above.

Background

Markov Decision Process (MDP) is a popular formalism for sequential decision making under uncertainty (Howard 1960; Puterman 2014). Formally, an MDP is specified by a tuple $\langle S, A, T, R, \gamma \rangle$, where S is a finite state space, A is a finite set of actions, T is a state transition function, R is a reward function, and $\gamma \in (0, 1)$ is a discount factor. At each decision step, the system is at a state $s \in S$ and the agent is required to choose an action $a \in A$ that is applicable at this state. When the agent performs an action a in a state s, it receives an immediate reward $R(s, a)$ and the system transitions to a new state $s' \in S$ with probability $T(s, a, s')$. The discount factor $\gamma \in (0, 1)$ models the preference for immediate over future rewards. In many cases, one of the states in an MDP is marked as the initial state $s_0 \in S$, and one or more states are marked as *terminal states*. This indicates that the system is initially at s_0 and it halts when reaching one of the terminal states. A solution to an MDP is a *policy* denoted by π, which is a mapping between states and actions. The optimal policy, denoted π^*, maximizes the expected sum of discounted rewards.

Real-Time Dynamic Programming RTDP (Barto, Bradtke, and Singh 1995) is a well-known algorithm for solving large MDPs that works by performing *simulations* in the search space. A simulation here is to perform actions and sampling their outcomes according to the MDP transition function, starting at the initial state and ending at a terminal state. The sequence of actions and outcomes resulting from performing a simulation is called a *trajectory*. Every "step" in a trajectory can be represented by a quadruple $\langle s_{i-1}, a_i, s_i, r_i \rangle$ where s_i is the system state after i actions, a_i is the i^{th} action performed, and r_i is the reward obtained after executing action a_i at state s_{i-1}.

RTDP maintains a Q table that stores for every state-action pair (s, a) a value that estimates the expected sum of discounted rewards for executing action a in state s, and following the best policy afterwards. To update the entries of this Q table, after every step $\langle s, a, s', r \rangle$ in a trajectory, the Q table is updated using a Bellman update (Barto, Bradtke, and Singh 1995):

$$Q(s, a) := r + \gamma \cdot \sum T(s, a, s') \cdot \max_{a' \in A}\{Q(s', a')\}. \quad (1)$$

RTDP is an anytime algorithm that keeps performing simulations and performing Bellman updates until it halts. The policy RTDP returns is induced by its Q table as follows:

$$\pi(s) := \arg\max_{a \in A}\{Q(s, a)\}. \quad (2)$$

Partially Observable MDP A Partially Observable Markov Decision Process (POMDP) (Kaelbling, Littman, and Cassandra 1998) is an MDP in which the agent cannot directly observe the current state. Instead, the agent receives observations after performing an action. The POMDP model includes an observation function that maps the probability the agent is at a state, given the action it performed and the observation it received. The standard approach to solve POMDP problems is to maintain a *belief state*, which is a probability distribution function over the possible states. The agent maintains the belief state during execution, and a solution to a POMDP is a policy mapping belief states to actions. A known technique for generating such a policy is to compile the POMDP to a *belief-state MDP*, whose states are the POMDP belief states, and apply an MDP solver. However, the resulting MDP is often very large and may even be infinite. In general, solving POMDP problems is undecidable and notoriously difficult in practice. Thus, POMDP-based solutions often fail to scale gracefully to problems with large state spaces. For example, existing POMDP-based approaches that have been applied to other PE variants were limited to grids that are significantly smaller than those we deal with (Yi, Nam, and Sycara 2019).

Problem Definition

Next, we describe the specific PE problem that we address in this work, which is inspired by a real-world problem. There is a single Evader agent, denoted e, and a single Pursuer agent, denoted p. Both e and p operate in a finite 3D grid, where each agent occupies a single grid cell at each time. The goal of e is to reach one of m possible *target locations*. The goal of p is to prevent e from achieving its goal by *colliding* with e, that is, enter a grid cell that is occupied by e, before e reaches one of its target locations. Both agents use a movement model based on the race-car domain (Barto, Bradtke, and Singh 1995). Each agent has a discrete velocity $\Delta\vec{v} = (\Delta x, \Delta y, \Delta z)$ in each of the 3 axes. The velocity in each axis can be between $-max_p$ and $+max_p$ for the Pursuer, and $-max_e$ and $+max_e$ for the Evader. The initial velocity of the agents is $(0, 0, 0)$. After an agent starts to move, it can never return to a velocity of $(0, 0, 0)$, that is, the agents cannot remain in the same position after they move from their initial location. We assume that time is discretized into time steps, and in each time step each agent can change its velocity by ± 1 in each direction. If an agent is currently at position (x, y, z) and has velocity $\vec{v} = (\Delta x, \Delta y, \Delta z)$, then it reaches $(x+\Delta x, y+\Delta y, z+\Delta z)$ in the next time step. The agents are not allowed to move beyond the boundaries of the grid. We currently assume deterministic actions, but the approach we proposed in this paper is applicable to stochastic actions as well. The agents e and p are not homogeneous. Specifically, p is slower than e, i.e., $max_p < max_e$, but e cannot sense the position of p.

Consequently, e must decide on its plan before starting to move, i.e., it uses a non-reactive policy. By contrast, p is able to sense the location of e while in motion, and adjust its plan accordingly. In this paper, we focus on the problem of generating such a policy for the p agent.

In general, this problem setting is naturally formalized as a zero-sum game, and can be solved with standard game-theoretic algorithms such as Minimax, Alpha-Beta, Monte Carlo Tree Search Solver (Winands, Björnsson, and Saito 2008), and more modern techniques based on Reinforcement Learning (RL) (Silver et al. 2016; Vinyals et al. 2019). These methods, however, are not expected to scale well to large 3D grid sizes without computing resources that are beyond the reach of most people. In addition, since in our case the e agent cannot react to the p agent during a game, its policy can be summarized as a possible stochastic selection between a set of plans. We assume that an *opponent modeling* tool is used to adequately characterize this stochastic selection, e.g., by applying learning techniques to analyze previously observed behavior of e (Shen and How 2021; He et al. 2016; Billings et al. 1998)

Specifically, we assume that the opponent modeling tool returns a set of plans B and a probability distribution $pr : B \rightarrow [0,1]$ over these plans. Each plan in B leads agent e from its initial location to a target location. Agent e is assumed to randomly choose a plan $\rho \in B$ with probability $pr(\rho)$ before moving, and follow it until either reaching a target location or getting caught by p. We refer to our variant of the PE problem as the Pursuit Evasion with Fixed Evasion plans problem (PE-FEP), defined formally as follows.

Definition 1 (PE-FEP)**.** *The PE-FEP problem is defined by a tuple $\langle G, s_p, s_e, max_p, max_e, B, pr, R_{catch}, R_{miss}\rangle$ where G is a 3D grid, s_p and s_e are the initial locations of p and e in the grid, max_p and max_e are the maximal velocities of p and e, B is the set of plans for agent e, pr is a probability density function over B, R_{catch} is the positive reward given when p catches e, and R_{miss} is the negative reward given when e reaches a target location or when p exits the grid. A solution to an PE-FEP problem is a policy for the p agent.*

To motivate agent p to catch e as fast as possible, we consider a small discount factor γ of 0.987. This means the reward of catching e decreases by a factor of 0.987 after every time step in which it has not catched e. All our contributions are agnostic to the exact value of γ.

PE-FEP as a Markov Decision Process

PE-FEP can be formalized as a POMDP, where the state comprises the current locations and velocities of the agents, and the plan chosen by e. The locations and velocities of the agents are observable, but the plan $\rho \in B$ chosen by e is not observable (by p). The size of the state space of this POMDP is

$$\left(|G| \cdot (2 \cdot max_e + 1)^3\right) \cdot \left(|G| \cdot (2 \cdot max_p + 1)^3\right) \cdot |B| \quad (3)$$

which is way beyond modern POMDP solvers capabilities for large grid sizes. In this section, we propose two alternative modeling approaches that enable better scaling.

Abstract MDP Models

In this modeling approach, we define two MDP models that abstract some aspects of full PE-FEP problem to gain efficiency. We expect that policies for such MDP models may be easier to solve, but the quality of the effectiveness of resulting policy in PE-FEP may be lower.

Position-based MDP (M_{pos}) In the first MDP model we consider, denoted M_{pos}, a state contains the location of both agents and the velocity of only the p agent. The rationale for this abstraction is the fact that the location of the e agent depends only on its current location and the plan it has chosen. This is a lossy abstraction, in the sense that remembering where e has been in previous time steps may allow agent p to update its belief over the plans e has chosen.

In more details, a state in M_{pos} is a tuple $s = \langle \vec{l_p}, \vec{v_p}, \vec{l_e}\rangle$, where $\vec{l_p}$ and $\vec{l_e}$ are the locations of p and e, and $\vec{v_p}$ is the velocity of p. An action is a vector $\vec{a} = (\delta_x, \delta_y, \delta_z)$, where $\delta_x, \delta_y, \delta_z \in \{-1, 0, 1\}$, indicates by how much to change the velocity of p. The reward function is defined based on R_{catch} and R_{miss}, assigning rewards only to terminal states in which either e has been caught or e has reached its target. The transition function $T(s, \vec{a}, s')$ for a state $s = \langle \vec{l_p}, \vec{v_p}, \vec{l_e}\rangle$, an action $\vec{a} = (\delta_x, \delta_y, \delta_z)$, and a state $s' = \langle \vec{l_p} + \vec{v_p} + \vec{a}, \vec{v_p} + \vec{a}, \vec{l_e}'\rangle$, is computed by considering all plans in B in which e is at $\vec{l_e}$ and the probability of choosing these plans is

$$T(s, \vec{a}, s') = \sum_{\rho \in B : (\vec{l_e}, \vec{l_e}') \in \rho} pr(\rho). \quad (4)$$

The size of the M_{pos} state space is

$$\left(|G| \cdot (2 \cdot max_p + 1)^3\right) \cdot |B| \cdot L_{max} \quad (5)$$

where L_{max} is the length of the longest plan in B. This is considerably smaller than the size of the POMDP (Equation 3).

Time-and-position-based MDP ($M_{pos(t)}$) The position-based MDP does not preserve the Markov propriety, since e may occupy the same location at different times and move to different places. To partially remedy this, our second MDP model, denoted $M_{pos(t)}$, includes also the current time in the state. Tracking time steps of states reduces the effect of *perceptual aliasing* (Chrisman 1992), where the same observation is obtained in distinct states where different actions should be performed. That being said, the new model is still an abstraction of PE-FEP problem.

In more details, in the time-and-position based MDP a state is a tuple $s = \langle \vec{l_p}, \vec{v_p}, \vec{l_e}, t\rangle$, where $\vec{l_p}$ and $\vec{l_e}$ are the locations of p and e, $\vec{v_p}$ is the velocity of p and t is the current time step. The actions and reward function remain the same as in the position-based model. For a plan $\rho \in B$, let $\rho[t]$ be the location of e at time t according to ρ. The transition function $T(s, \vec{a}, s')$ for a state $s = \langle \vec{l_p}, \vec{v_p}, \vec{l_e}, t\rangle$, an action $\vec{a} = (\delta_x, \delta_y, \delta_z)$, and a state $s' = \langle \vec{l_p} + \vec{v_p} + \vec{a}, \vec{v_p} + \vec{a}, \vec{l_e}', t+1\rangle$, is computed by considering all plans in B in which e is at $\vec{l_e}$ at time t and at $\vec{l'_e}$ at time $t+1$:

$$T(s, \vec{a}, s') = \sum_{\rho \in B : \vec{l_e} = \rho[t] \wedge \vec{l_e}' = \rho[t+1]} pr(\rho). \quad (6)$$

The size of the $M_{pos(t)}$ state space is

$$\left(|G| \cdot (2 \cdot max_p + 1)^3\right) \cdot |B| \cdot L_{max}^2. \quad (7)$$

Belief State MDP (M_{bel})

The MDP models described so far abstract some aspects of the PE-FEP problem, and their state information is not sufficient to preserve the Markov propriety across all states in the problem. As a result, an optimal policy for these models may not be an optimal policy for the given PE-FEP problem.

To this end, we proposed an MDP model, denoted M_{bel}, that accurately captures all aspects of the PE-FEP problem. M_{bel} can be viewed as a compact version of the belief-state MDP corresponding to our POMDP model of PE-FEP. Thus, we refer to this MDP as the *belief MDP* and show that its size is finite and tractable.

The actions and reward function of our belief MDP is the same as in the previous MDPs. A state is a tuple $s = \langle \vec{l_p}, \vec{v_p}, t, F\rangle$, where $\vec{l_p}$, $\vec{v_p}$, and t are defined as above, and F is a subset of the Evader plans B, representing the set of plans agent e may be following given where it visited so far. In the initial state $F = B$. Unlike the states in the position-based and the position-and-time-based MDPs, a state in our belief MDP is a *sufficient statistic* for the PE-FEP problem. That is, given a state $s = \langle \vec{l_p}, \vec{v_p}, t, F\rangle$, action $\vec{a} = (\delta_x, \delta_y, \delta_z)$, and state $s' = \langle \vec{l_p} + \vec{v_p} + \vec{a}, \vec{v_p} + \vec{a}, t+1, F'\rangle$, we can compute the exact probability of reaching s'. The updated belief F' comprises every plan in F except plans that are inconsistent with the new location of e. That is, $F' = \{\rho | \rho \in F \text{ where } \rho[t+1] = \vec{l_e}\}$. Since agent e is not reactive, pruning inconsistent plans does not change the relative probabilities of the remaining plans. Hence, to compute the probability that a plan ρ is the one agent e is following, we normalize the prior probability of choosing ρ over all the plans in F. Formally, the probability that e chose ρ given F is $\frac{pr(\rho)}{\sum_{\rho' \in F} pr(\rho')}$. We denote this conditional probability by $Pr(\rho|F)$. Finally, the transition function for the state s, action $\vec{a}$, and state s' specified above is given by:

$$T(s, \vec{a}, s') = \sum_{\rho \in F: \vec{l_e} = \rho[t] \wedge \vec{l_e}' = \rho[t+1]} Pr(\rho, F). \qquad (8)$$

To compute the number of states in our belief MDP, we analyze the possible values for F. Observe that changes to F are agnostic to the actions of agent p. In addition, as time progresses plans are only removed from F but never added. Thus, we can create *belief tree* that represents the possible values of F over time. A node in this tree is a pair $\langle t', F'\rangle$ representing all states in which $F = F'$ and $t = t'$. The root node is $\langle 1, B\rangle$. The children of the root correspond to the possible beliefs at the first time step t_1 in which exists two plans $\rho_1, \rho_2 \in B$ in which e occupies different cells. More generally, the children of a node $\langle t', F'\rangle$ are the possible beliefs at the first time step $t'' > t'$ in which there exists two plans $\rho_1, \rho_2 \in F'$ in which e occupies different cells. The leaves of the belief tree are nodes in which F' consists of a single plan. Since all plans have a fixed length, a plan can only appear in a single leaf. Thus, the size of the belief tree is at most $2|B|$. Practically, we compute the belief tree only once, and every state in the belief MDP stores a pointer to the respective node in the belief tree instead of storing all the plans in the corresponding belief tree node. Consequently, the size of the M_{bel} state space is

$$\left(|G| \cdot (2 \cdot max_p + 1)^3\right) \cdot 2 \cdot |B|^2 \cdot L_{max}^2 \qquad (9)$$

where $|G| \cdot (2 \cdot max_p + 1)^3$ represents all possible locations and velocities of p, $|B| \cdot L_{max}$ represents the possible locations of e, L_{max} represents the possible time steps, and $2|B|$ represents the possible nodes in the belief tree.

Model	State	State Space Size
M_{pos}	$s = \langle \vec{l_p}, \vec{v_p}, \vec{l_e}\rangle$	$\left(\lvert G\rvert \cdot (2 \cdot max_p + 1)^3\right) \cdot \lvert B\rvert \cdot L_{max}$
$M_{pos(t)}$	$s = \langle \vec{l_p}, \vec{v_p}, \vec{l_e}, t\rangle$	$\left(\lvert G\rvert \cdot (2 \cdot max_p + 1)^3\right) \cdot \lvert B\rvert \cdot L_{max}^2$
M_{bel}	$s = \langle \vec{l_p}, \vec{v_p}, t, F\rangle$	$\left(\lvert G\rvert \cdot (2 \cdot max_p + 1)^3\right) \cdot 2 \cdot \lvert B\rvert^2 \cdot L_{max}^2$

Table 1: A summary of our different MDP models.

Table 1 summarizes the main properties of our models. As expected, the higher fidelity models have a larger state space, suggesting a possible tradeoff between solution time and solution quality. We investigate this tradeoff experimentally later.

Solving the PE-FEP MDPs

In either MDP model, the size of the corresponding search space is too large to be solved by a standard implementation of MDP algorithms, such as RTDP (Barto, Bradtke, and Singh 1995), for the grid sizes we are interested in. For example, consider a PE-FEP probelm with a grid size of $800 \times 500 \times 5$, speed limit of 1 ($max_e = 1$), 50 possible plans for agent e ($|B| = 50$), and a maximal Evader plan length of 400 ($L_{max} = 400$). The state space size of even the position-based MDP is approximately 108×10^{10}. Of course, many of these states are not reachable when following a reasonable policy, but this is still far more than what a naive implementation of RTDP can handle. In the rest of this section, we propose to augment RTDP with two novel, orthogonal, techniques that allow solving large PE-FEP problems. The first technique is a special form of spatial action abstraction. The second technique is a heuristic for initializing the RTDP Q table. We describe these techniques in details below.

Spatial Action Abstraction

We formulate our spatial action-abstraction technique using the Options framework (Sutton, Precup, and Singh 1999). An Option can be viewed as a macro-operator (Korf 1985) for an MDP. Planning over a set of options instead of the atomic MDP actions of a domain can speedup planning significantly (Mann and Mannor 2014; Mankowitz, Mann, and Mannor 2014). Formally, an *option* is a tuple $O = \{I, \pi, \beta\}$ where I is the set of states where the option can be initiated, $\pi : S \rightarrow A$ is the policy followed while executing the option, and $\beta : S \rightarrow \{0, 1\}$ is the option termination condition, specifying when to stop executing π.

Defining an effective set of options for a given problem is a challenge. Prior work assumed the set of available options is either provided as input to the solver (Sutton, Precup, and Singh 1999) or learned (Stolle and Precup 2002; Mankowitz et al. 2018). Existing algorithms for learning options are not expected to be effective in our context, as positive rewards in our MDPs are only encountered deep in the search space, and the size of the search space is very large for PE-FEP problems that we are interested in.

We present here an dedicated type of options that are especially designed for large spatial-based MDPs such as ours. Using these spatial-based options is intended to vary the granularity of the agent's decision-making based on the distance between the agents. When the distance is large, the available options correspond to maintaining a chosen velocity for a large number of time steps. As the distance between the agents diminishes, the available options maintain a chosen velocity for a smaller number of time steps, allowing p more controllability to direct itself towards e. Eventually, when the agents are near, we make a decision at each time step and the available options reduce to the atomic actions of changing the agent's velocity for a single time step.

Formally, we create an option $O_{s,\vec{D}}$ for every state $s = \langle \vec{l_p}, \vec{v}_p, \vec{l_e} \rangle$ and direction $\vec{d} = (d_x, d_y, d_z)$ that the agent can move to, where $\vec{d} \in \{-1, 0, 1\}^3$. Thus, in every state s there are exactly 27 applicable options. The policy for an option $O_{s,\vec{D}}$ is to move in the direction $\vec{d}$ a fixed number of cells. Importantly, that fixed number is a function of the distance between the agents in s. Specifically, we propose the following exponential decaying function of the $\mathcal{L}_\infty$ distance between the agents in s:

$$D(s) = 2^{\max\{\lfloor log_2(||\vec{l}_p - \vec{l}_e||_\infty)\rfloor - 3, 0\}} \tag{10}$$

For example, consider a state s in which the distance between the agents is 1024. In this case, the policy for the options in this state would move in the chosen direction for 128 grid cells. In a state where the agents are closer and the distance between them is 100, the corresponding options perform the same basic action 8 times. Finally, when the distance between the agents is smaller than 16, the available options represent applying a basic action once.

We use the options generated in the above manner in RTDP by using options instead of basic actions. This means the RTDP Q table stores a Q value for every (state, option) pair instead of (state, action) pair, and the policy it learns maps states to options. In addition, we update the RTDP using the Bellman update operator to reflect that an "action" in a state s corresponds to performing a sequence of $D(s)$ atomic actions, as follows:

$$Q(s, O_{s,\vec{d}}) = R(s, O_{s,\vec{d}}) + \gamma^{D(s)} \sum_{s'} T(s, O_{s,\vec{d}}, s') V(s') \tag{11}$$

where $R(s, O_{s,\vec{d}})$ is the sum of discounted rewards collected when executing the option, and

$$V(s') = \max_{O_{s',\vec{d}'}} Q(s', O_{s',\vec{d}'}). \tag{12}$$

Heuristics

A known technique to speedup the performance of RTDP is to use a heuristic function to intelligently initialize the values in the Q table (Bonet and Geffner 2003; McMahan, Likhachev, and Gordon 2005). These heuristic functions receive a state and an action, and output the value with which to initialize the corresponding entry in the Q table. Let $Q^*(s, a)$ be the expected discounted reward obtained by performing action a at state s and subsequently following an optimal policy. A heuristic function H is called *admissible* if it is an upper bound on $Q^*(s, a)$ for every state-action pair (s, a), i.e., if $H(s, a) \geq Q^*(s, a)$. Initializing RTDP with an admissible heuristic guarantees that its Q table will eventually converge to Q^*, and subsequently the policy it returns is optimal. Next, we describe several admissible heuristics for our MDP models.

Discounted Reward Heuristics In all our MDP models, rewards are only given at a terminal state. Thus, the maximal expected discounted reward in a state is R_{catch}, the reward for catching e. However, that reward is only obtained when p reaches e. If this occurs t time steps after the current state, then the discounted reward at that state is only $R_{catch} \cdot \gamma^t$. We say that heuristic H is a *discounted reward heuristic* if it is of the form $H(s, a) = R_{catch} \cdot \gamma^t_{\min}(s)$ for some function $t_{\min}(s)$.

Observation 1. *Any discounted reward heuristic in which $t_{\min}(s)$ is a lower bound on the time it will take e to catch p when at state s, is admissible.*

A trivial admissible discounted reward heuristic sets $t_{\min}(s)$ to zero. The resulting heuristic, referred to as the *zero heuristic* and denoted H_0, returns R_{catch} for every (s, a) pair. A more informed approach to obtain an admissible heuristic is to consider the distance between the agents in s when computing $t_{\min}(s)$. If the distance between the agents is d, then the fastest way they can reach each other is by moving towards each other at their highest speed. Correspondingly, we can obtain an admissible discounted reward heuristic by setting $t_{\min}(s)$ to be $\frac{||\vec{l}_e(s) - \vec{l}_p(s)||_\infty}{\max_p + \max_e}$, where $\vec{l}_e(s)$ and $\vec{l}_p(s)$ are the locations of e and p in state s. We refer to the resulting heuristic as the *air distance* heuristic and denoted it by H_{Air}.

Plan-Aware Heuristics The zero and air distance heuristics ignore the fact that the Evader is limited to move along a plan from B. The following discounted reward heuristics, referred to as the *plan-aware heuristics*, exploit that knowledge to provide tighter computation of $t(s)$. To define these heuristics, we introduce the following helper functions:

$$T_p(\vec{l}_1, \vec{l}_2) = \frac{\mathcal{L}_\infty(\vec{l}_1 - \vec{l}_2)}{max_p} \tag{13}$$

$$h(\vec{l}_p, t, \rho) = \underset{t' \geq t}{\arg\min} \begin{cases} T_p(\rho[t'], \vec{l}_p) & t' \geq T_p(\rho[t'], \vec{l}_p) \\ \infty & otherwise \end{cases} \tag{14}$$

$T_p(\vec{l}_1, \vec{l}_2)$ computes a lower bound on the time it takes p to move from $\vec{l}_1$ to $\vec{l}_2$. $h(\vec{l}_p, t, \rho)$ computes a lower bound on the time it takes p to reach e given that it is following plan $\rho \in B$ and the current time step is t. We use this h function to define a plan-aware heuristic function for each of our MDP modeling.

For M_{pos}, we set $t_{\min}(s)$ to be the minimum over all Evader plans and time steps that are consistent with the location of e in the current state. Specifically, we propose a discounted reward heuristic that computes $t_{\min}(s)$ as

$$\min_{(\rho,t) \in \{\rho,t | \rho \in B \wedge \rho[t] = \vec{l}_e(s)\}} h(\vec{l}_p(s), t, \rho) \tag{15}$$

We denote the resulting heuristic by H_{pos}.

For $M_{pos(t)}$, the states contain additional information that allows further refinement of the set of Evader plans consistent with s. Specifically, for $M_{pos(t)}$, we propose a discounted reward heuristic that computes $t_{\min}(s)$ as

$$\min_{\rho,\in\{\rho|\rho\in B\wedge\rho[t(s)]=\vec{l}_e(s)\}} h(\vec{l}_p(s), t(s), \rho) \quad (16)$$

where $t(s)$ denotes the time step of s. We denote the resulting heuristic by $H_{pos(t)}$.

For M_{bel}, a state s explicitly contains the set of consistent Evader plans F as well as probability distribution over them. Here, we slightly break the format of a discounted reward heuristic and propose the following heuristic:

$$H_{bel}(s,a) = \sum_{\rho\in F(s)} Pr(\rho|F(s)) \cdot \gamma^{h(\vec{l}_p(s),t(s),\rho)} \cdot R_{catch} \quad (17)$$

where $F(s)$ is the consistent Evader plans at state s.

Observation 2 (Admissibility). *All our proposed heuristics are admissible.*

Proof. All our heuristics, except H_{bel}, are admissible due to Observation 1. For H_{bel}, observe that $\gamma^{h(\vec{l}_p(s),t(s),\rho)} \cdot R_{catch}$ is a lower bound the expected discounted reward when at state s and e is following plan ρ, then $H_{bel}(s,a)$ is lower bound the expected discounted reward when at state s. Thus, H_{bel} is admissible. □

Experimental Results

We implement all the RTDP-based algorithms we proposed and evaluated them experimentally on a set of PE-FEP problems. In each experiment, we selected (1) an MDP model (M_{pos}, $M_{pos(t)}$, or M_{bel}); (2) whether or not the spatial action abstractions were used; and (3) an RTDP heuristic function (H_0, H_{Air}, H_{pos}, $H_{pos(t)}$, or H_{bel}). Then, we run the corresponding RTDP one a given PE-FEP problem until either the RTDP policy converged or a predefined *budget* of simulations has exceeded. Unless stated otherwise, the budget was set to $5 \cdot 10^6$ simulations. All experiments were conducted on a single machine with 4 CPU cores 2.60GHz and 16 GB of RAM. Our solver is implemented in C++.

This goal of this experimental evaluation is to answer the following research questions: (**RQ1**) Is the proposed spatial action effective? (**RQ2**) Are the proposed heuristics effective, and which heuristic works best? (**RQ3**) Can our RTDP-based algorithm solve large-scale PE-FEP problems? (**RQ4**) When each MDP model — M_{pos}, $M_{pos(t)}$, or M_{bel} — should be used?

The main performance metric we consider is the *collision rate*. After an algorithm returns a policy, we execute 1,000 simulations. The collision rate is the number of simulations ended up with p catching e, divided by 1,000. Note that the collision rate is computed w.r.t the actual PE-FEP problem (Definition 1), not w.r.t. the chosen MDP model, which may be simpler, e.g., in the case of M_{pos}. The second metric we measure is the *simulations rate*, which is the number of simulations performed, as part of the learning phase of the algorithm, divided by the given budget. Lastly, the third metric we consider is called *convergence rate*, which is the number of episodes it took to obtain the best policy (the policy that is ultimately returned), in terms of collision rate, divided by the budget. For example, let us assume that the algorithm halted after performing 500 simulations (out of a budget of 10,000 simulations) and that the best policy was achieved after performing 400 simulations. In this case, the simulations rate is $500/10,000 = 0.05$ and the convergence rate is $400/10,000 = 0.04$. To measure the collision rate, let us assume that we executed 1,000 simulations of the resulted policy and that the agents collided in 900 of the simulations. Then, the collision rate is $900/1,000 = 0.9$.

Creating PE-FEP Problems

We created PE-FEP problems for our experiments as follows. First, we set the length, width, and height of the grid (G), denoted $\max_x$, $\max_y$, and $\max_z$, respectively. All target locations and the initial locations of p and e (i.e., s_p and s_e) are positions on the ground. The initial locations of p and e are in opposing sides of the grid, and the targets are located between them but closer to agent p.

To create non-trivial Evader plans B, we enforced each evader plan $\rho \in B$ to pass through a set of waypoints. When moving between waypoints, e follows the shortest path between them. Figure 1 illustrates one of the waypoints distribution used in our experiments (waypoints are marked by "x"). The waypoints are arranged in columns over the y-axis. Every evader plan starts at S_e, moves from right to left by choosing a single waypoint from each column, until eventually reaching one of the target locations (marked by a green start). The exact benchmark of PE-FEP problems will be made available upon request.

The Wait-For-It Baseline

Since PE-FEP is a new variant of PE, there are no established baseline algorithm to compare with. Nevertheless, we consider two algorithms as our baselines. The first baseline is vanilla RTDP, without spatial action abstraction and only the zero heuristic. The second baseline implements the rule-based policy in which p waits in its initial location until it either infers the Evader's plan and is able to catch it, or it infers that it cannot infer the Evader's exact plan in time at which case it selects one Evader plan and targets it. In more details, this baseline solver, which we refer to as the Wait-For-It (WFI) algorithm, computes for each Evader plan $\rho \in B$ the last time step in which p can still catch e assuming p remains in its initial location. Then, its policy for p is remains in its initial location as long as e follows plans for which p can still catch e. When e is about to cross the last time step described above, p randomly chooses a plan from the remaining consistent plans and tries to catch e accordingly. Note that, as time passes, the set of relevant plans F decreases with fewer plans remain in the Pursuer beliefs

Spatial Action Abstraction: Results

The first set of experiment evaluates the benefit of our spatial action abstraction. We report here on a subset of a larger set of experiments we conducted to address this research question (RQ1), In this subset, we used the M_{pos}model and run RTDP with H_{pos} on grids of different sizes and 6 Evader plans. For each grid size, we generated 20 different PE-FEP problem instances. We repeated this experiment with and without our spatial action abstractions. Table 2 shows the average CPU time, collision rate, and simulation rate for both options. We mark *out of memory* (OOM) when the running process exceeded the maximum RAM memory.As can be seen, the benefit of using our spatial action abstraction is significant. In some cases, it yielded a speedup of more than two orders of magnitude. The results also show that, beyond $600 \times 300 \times 5$, RTDP with only atomic actions exceeds the RAM capacity. While the time required to reach a collision rate of 1.0 increases rapidly with the grid size for RTDP with atomic actions, using our spatial action abstractions allowed RTDP to achieve a perfect collision rate even for the very large $1000 \times 600 \times 5$ grids. Recall that the corresponding MDP has more than 16,875,000,000 states. We also observed the same trend in the other models ($M_{pos(t)}$ and M_{bel}). Since the benefit of using our options is clear, we only use this method in all experiments below. Thus, using spatial abstractions are shown to be highly beneficial and allow scaling to very large grids, suggesting that the answers to RQ1 and RQ3 are affirmative.

	Time (sec.)		Collison Rate		Simulation Rate	
Grid Size	Options	Atomic	Options	Atomic	Options	Atomic
$20 \times 10 \times 5$	0.01	0.78	1.00	1.00	**0.00**	0.03
$40 \times 20 \times 5$	0.01	9.41	1.00	1.00	**0.00**	0.17
$80 \times 40 \times 5$	0.07	26.50	1.00	0.99	**0.00**	0.20
$160 \times 80 \times 5$	0.19	249.20	1.00	0.97	**0.00**	1.00
$320 \times 160 \times 5$	0.27	553.40	1.00	0.95	**0.00**	1.00
$600 \times 300 \times 5$	0.90	919.70	1.00	0.90	**0.01**	1.00
$1000 \times 600 \times 5$	1.20	OOM	1.00	OOM	**0.05**	OOM

Table 2: Spatial action abstraction and atomic actions for different grid sizes with 6 Evader plans, averaged over 20 random settings.

Heuristics : Results

Next, we evaluated the impact of using our discounted reward heuristics to initialize RTDP. To this end, we designed a set of experiments over various grid sizes with 30 distinct Evader plans. The grid setting was similar to the one used in the previous experiment, and we used the M_{pos}model. We repeated this experiment with each of the relevant heuristics, namely, H_0, H_{Air}, and H_{pos}. Table 3 shows the collision rate and convergence rate achieved by RTDP with the different heuristics. The rows show the different grid sizes and the columns represent the collision and convergence rates for each of the heuristic functions. In all problem settings, the H_{pos} heuristic achieved the highest collision rate, excluding the smallest grid where all methods achieved a collision rate of 1.0. Another significant advantage for H_{pos} can be seen in the convergence rate. The results show that H_{pos} outperformed the other two heuristics. We also observed the same trend when using $M_{pos(t)}$and M_{bel}. Thus, we use the plan-based heuristics in all subsequent experiments.

Modeling : Results

Next, we investigate the performance of RTDP on the different MDP models we proposed, namely M_{pos}, $M_{pos(t)}$, and M_{bel}. We report here results for a grid size of $550 \times 200 \times 5$, on the PE-FEP problems depicted in Figure 1. Figure 1 shows $2D$ maps with different S_p points for the Pursuer (blue). We gradually increased the x coordinate of the Pursuer initial location to see the effect on each of the modeling methods. The Evader initial location (red) is fixed across all problems. The X marks are the waypoints and the green stars are target locations. We present some possible Evader plans to demonstrate how an Evader plan looks in a $2D$ viewpoint.

Grid Size	Collision Rate			Convergence Rate		
	H_0	H_{Air}	H_{pos}	H_0	H_{Air}	H_{pos}
$20 \times 10 \times 5$	**1.00**	**1.00**	**1.00**	0.59	0.27	**0.05**
$40 \times 20 \times 5$	0.78	0.89	**1.00**	0.92	0.88	**0.17**
$80 \times 40 \times 5$	0.54	0.88	**1.00**	0.76	0.99	**0.04**
$160 \times 80 \times 5$	0.70	0.74	**1.00**	0.91	0.51	**0.03**
$320 \times 160 \times 5$	0.77	0.85	**0.96**	0.85	0.96	**0.09**
$600 \times 300 \times 5$	0.58	0.74	**1.00**	0.86	0.95	**0.10**
$1000 \times 600 \times 5$	0.61	0.72	**0.92**	0.71	0.51	**0.10**

Table 3: Different heuristics for the position model for different grid sizes with 30 distinct Evader plans.

	Metric	Model	1	2	3	4	5	6
(a)	Coll. Rate	WFI	0.61	0.61	0.61	0.62	0.34	0.19
		M_{pos}	**1.00**	**1.00**	0.60	0.62	0.34	**0.33**
		$M_{pos(t)}$	**1.00**	**1.00**	0.65	0.65	0.38	**0.33**
		M_{bel}	**1.00**	**1.00**	**0.90**	**0.99**	**0.54**	**0.33**
	Conv. Rate	M_{pos}	**0.00**	**0.00**	**0.53**	**0.30**	0.43	**0.47**
		$M_{pos(t)}$	**0.00**	**0.00**	0.57	0.49	0.54	**0.47**
		M_{bel}	0.01	0.04	0.60	0.48	**0.36**	**0.47**
(b)	Coll. Rate	WFI	0.51	0.51	0.50	0.50	0.50	0.36
		M_{pos}	**1.00**	**1.00**	0.94	0.83	0.62	0.44
		$M_{pos(t)}$	**1.00**	**1.00**	0.95	0.86	0.74	0.56
		M_{bel}	**1.00**	**1.00**	**0.97**	**0.92**	**0.78**	**0.60**
	Conv. Rate	M_{pos}	**0.00**	**0.00**	0.21	0.37	0.49	0.67
		$M_{pos(t)}$	**0.00**	**0.00**	0.05	**0.07**	0.14	**0.20**
		M_{bel}	0.01	0.01	**0.03**	0.16	**0.12**	0.30
(c)	Coll. Rate	WFI	0.47	0.47	0.47	0.43	0.25	0.11
		M_{pos}	**1.00**	**1.00**	**1.00**	0.79	**0.43**	0.17
		$M_{pos(t)}$	**1.00**	**1.00**	**1.00**	0.79	**0.43**	0.17
		M_{bel}	**1.00**	**1.00**	**1.00**	**0.80**	**0.43**	**0.16**
	Conv. Rate	M_{pos}	**0.00**	**0.00**	**0.00**	**0.17**	**0.63**	**0.42**
		$M_{pos(t)}$	0.01	0.01	0.01	0.34	0.71	0.89
		M_{bel}	0.01	0.01	0.01	0.38	**0.63**	0.89

Table 4: Comparing model methods on the grids from Fig. 1.

Table 4 shows the maximum collision rate and the corresponding convergence rate (2nd column). Both metrics were measured for each model (3rd column) on each of the three problems illustrated in Figure 1 (1st column). Each column (columns 4-9) represents the different initial location of agent p where 1 is the closest location to the goals and 6 is the farthest.

The results show that, in all three problems, M_{bel} achieved the highest collision rate, which was close to 1.00 in the three closest locations of p. However, in these cases, M_{pos} and $M_{pos(t)}$ were converged faster than $Belief$. In problem (c), while all three modeling methods achieved almost the same collision rate, M_{pos} converged faster than the other methods. This phenomenon is related to the fact that when the approximation MDPs are good enough to model the problem, they update fewer states than M_{bel}, since model-

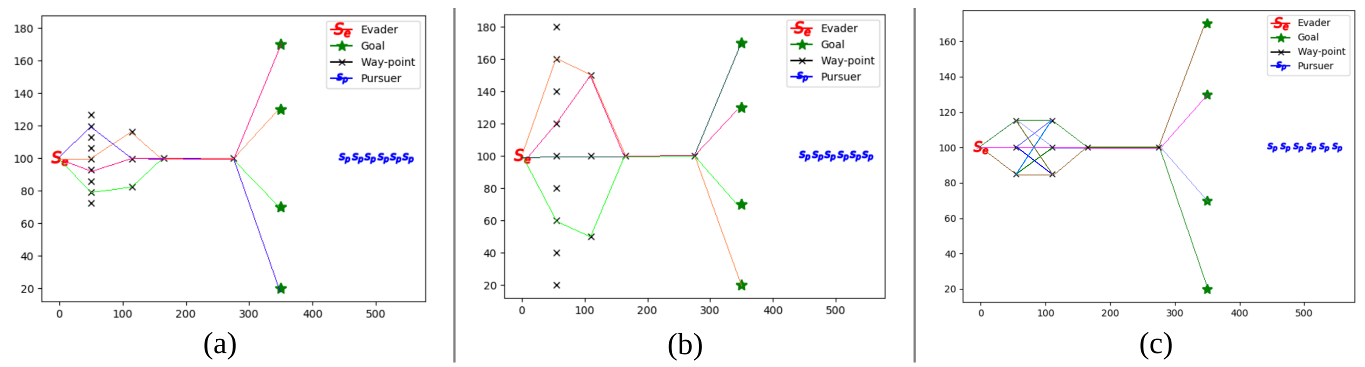

Figure 1: 2D overview on the grid with 4 goals and (a) 24 Evader plans with small waypoint gap, (b) 24 Evader plans with large waypoint gap, and (c) 36 Evader plans.

#Evader Plans	4	20	40	60
60	0.81	0.66	0.55	0.49
100	0.87	0.83	0.70	0.63
200	0.87	0.87	0.77	0.73
400	0.94	0.86	0.86	0.85

Table 5: Collision Rate as a function of the number of Evader plans (rows) and waypoints (columns) when using M_{bel}

ing with the $Belief$ results in more states to update.

To demonstrate the impact of the waypoints and the number of Evader plans on M_{bel}, we designed an experiment in which we increased the number of evader plans ($|B|$) but fixed the grid size to $550 \times 200 \times 50$ with 4 different goals. Table 5 shows the results in terms of collision rate. The rows represent the number of possible unique plans that can be created on the given grid and the columns are the number of unique Evader plans that were generated. For example, when the number of available different plans was 60 and the actually generated plans was 60, M_{bel} achieved only 0.49 collision rate. The first trend observed is that as more plans exist the collision rates decreases. This trend implies that the problem is harder. As we add more Evader plans, we directly increase the state space. We also observed that the collision rate increases as the number of possible plans increases. This means that the Evader plans are more sparse across the grid and fewer parts of the Evader plans overlap.

Related Work

Approaches to solve different variants of PE using Markov models have been proposed in the past. This includes POMDP models for PE variants with partial observability where the agent's (Pineau, Gordon, and Thrun 2003) and PE variants for multiple pursuer agents (Mottaghi and Vaughan 2007). These prior work cannot scale to the large grids we consider. For example, while the point-based later-tag domain is over a 2D grid with only 22 cells, we experiment in 3D grids with size of up to $800 \times 400 \times 5$ cells. Yi et al. (2019) focused on indoor PE problems using multiple pursuers. They proposed a hybrid hierarchical POMDP structure that utilizes the convex hulls of the environment to create abstract states. The approach reduces the state space for improved scalability. Yet, they could not scale to the large graphs we consider. Hollinger (2009) uses a POMDP formulation for a multi-robot PE variant, where a team searches in a known environment for a moving non-adversarial target. They show that the resulting planning problem is NP-hard, and that the optimal solution scales exponentially in the number of searchers. Thus, their approach is intractable for large environments or numerous searchers. Baltes sand Park (2001) investigated machine learning techniques in the context of the PE problem, directly considering the kinematics of the robots. The movement model is based on the physical robots on 2D plane, while PE-FEP has discrete actions in 3D space.

Conclusion

We presented a scalable algorithm for solving the *Pursuit Evasion with Fixed Evader Plans (PE-FEP)* problem, a real-world variant of the PE problem. We propose three MDP-based approaches to model PE-FEP and analyze them. Then, we introduce two novel techniques that enable using RTDP over the proposed MDP models, to efficiently solve PE-FEP problems. The first technique is a spatial abstraction of the action space based on the options framework (Sutton, Precup, and Singh 1999). In this abstraction, we force the Pursuer to choose a sequence of actions whose length is in correlation with the distance between the Pursuer and the Evader. The second technique involves admissible heuristic functions for each of our MDPs to initializing the RTDP Q table. We evaluated the different models and RTDP enhancements experimentally on a large 3D grid of up to $550 \times 200 \times 50$ grid cells and up to 60 different Evader plans. Our results show that RTDP with all our enhancements can solve such large problems, yielding over two orders of magnitude improvement over baseline techniques.

References

Baltes, J.; and Park, Y. 2001. Comparison of several machine learning techniques in pursuit-evasion games. In *Robot Soccer World Cup*, 269–274. Springer.

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1995. Learning to act using real-time dynamic programming. *Artificial intelligence*, 72(1-2): 81–138.

Billings, D.; Papp, D.; Schaeffer, J.; and Szafron, D. 1998. Opponent modeling in poker. *Aaai/iaai*, 493(499): 105.

Bonet, B.; and Geffner, H. 2003. Faster heuristic search algorithms for planning with uncertainty and full feedback. In *IJCAI*, 1233–1238.

Cheng, P. 2003. A short survey on pursuit-evasion games. *Department of Computer Science, University of Illinois at Urbana-Champaign*.

Chrisman, L. 1992. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *AAAI*, volume 1992, 183–188. Citeseer.

Chung, T. H.; Hollinger, G. A.; and Isler, V. 2011. Search and pursuit-evasion in mobile robotics. *Autonomous robots*, 31(4): 299.

He, H.; Boyd-Graber, J.; Kwok, K.; and Daumé III, H. 2016. Opponent modeling in deep reinforcement learning. In *International conference on machine learning (ICML)*, 1804–1813.

Hespanha, J. P.; Kim, H. J.; and Sastry, S. 1999. Multiple-agent probabilistic pursuit-evasion games. In *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*, volume 3, 2432–2437. IEEE.

Hollinger, G.; Singh, S.; Djugash, J.; and Kehagias, A. 2009. Efficient multi-robot search for a moving target. *The International Journal of Robotics Research*, 28(2): 201–219.

Howard, R. A. 1960. *Dynamic programming and markov processes*. John Wiley.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2): 99–134.

Korf, R. E. 1985. Macro-operators: A weak method for learning. *Artificial intelligence*, 26(1): 35–77.

Mankowitz, D. J.; Mann, T. A.; Bacon, P.-L.; Precup, D.; and Mannor, S. 2018. Learning robust options. In *AAAI Conference on Artificial Intelligence*.

Mankowitz, D. J.; Mann, T. A.; and Mannor, S. 2014. Time regularized interrupting options. In *Internation Conference on Machine Learning*.

Mann, T.; and Mannor, S. 2014. Scaling up approximate value iteration with options: Better policies with fewer iterations. In *International conference on machine learning*, 127–135.

McMahan, H. B.; Likhachev, M.; and Gordon, G. J. 2005. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *international conference on Machine learning*, 569–576.

Mottaghi, R.; and Vaughan, R. 2007. An integrated particle filter and potential field method applied to cooperative multi-robot target tracking. *Autonomous Robots*, 23(1): 19–35.

Pineau, J.; Gordon, G. J.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In Gottlob, G.; and Walsh, T., eds., *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, 1025–1032. Morgan Kaufmann.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Shen, M.; and How, J. P. 2021. Robust Opponent Modeling via Adversarial Ensemble Reinforcement Learning. In *International Conference on Automated Planning and Scheduling (ICAPS)*, volume 31, 578–587.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489.

Stolle, M.; and Precup, D. 2002. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, 212–223. Springer.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Winands, M. H.; Björnsson, Y.; and Saito, J.-T. 2008. Monte-Carlo tree search solver. In *International Conference on Computers and Games*, 25–36.

Yi, S.; Nam, C.; and Sycara, K. 2019. Indoor Pursuit-Evasion with Hybrid Hierarchical Partially Observable Markov Decision Processes for Multi-robot Systems. In *Distributed Autonomous Robotic Systems*, 251–264. Springer.