Probabilistic Multi-Robot Search for an Adversarial Target

Ryan J. Marcotte[1] Acshi Haggenmiller[1] Gonzalo Ferrer[2]
Edwin Olson[1] *†

Abstract

The problem of planning the actions of several robots (*pursuers*) who are searching for another agent (an *evader*) has been frequently studied, with most methods focusing on finding strategies that guarantee capture of even a worst-case evader. However, in many real-world situations, the environment may be too complex or the pursuers too few in number to ensure capture. In such cases, the best that the pursuers can do is select actions that maximize the probability of capture for a given type of evader.

In this paper, we propose Probabilistic Adversarial Target Search (PATS), which computes joint search actions that approximately maximize capture probability against an evader with perfect knowledge but finite speed. PATS uses Monte Carlo tree search (MCTS) to compute pursuit plans given the pursuers' probabilistic belief about the evader's location. PATS then evolves this belief forward in time based on the expected actions of the evader, which are obtained from the search tree's empirical statistics. We show that PATS outperforms an existing probabilistic search method in a simulated search setting for which guaranteed search is impossible.

1 Introduction

We consider the problem of searching for an adversarial target (an *evader*) with multiple robots (*pursuers*). Whereas others (e.g. [1]–[4]) have presented approaches that can guarantee capture given a sufficient number of pursuers, we address the case in which the pursuit team is too small or the environment too complex for such a guarantee. In such a case, the pursuers must plan their collective actions based on their belief about the evader's location, and then update the belief after executing those actions. We propose a single algorithm, which we call probabilistic adversarial target search (PATS), that accomplishes both of these tasks, planning joint pursuit actions and then using the results of those computations to update the belief.

*[1]University of Michigan, Ann Arbor, MI, USA {ryanjmar,acshikh,ebolson}@umich.edu
†[2]Skolkovo Institute of Science and Technology, Moscow, Russia g.ferrer@skoltech.ru

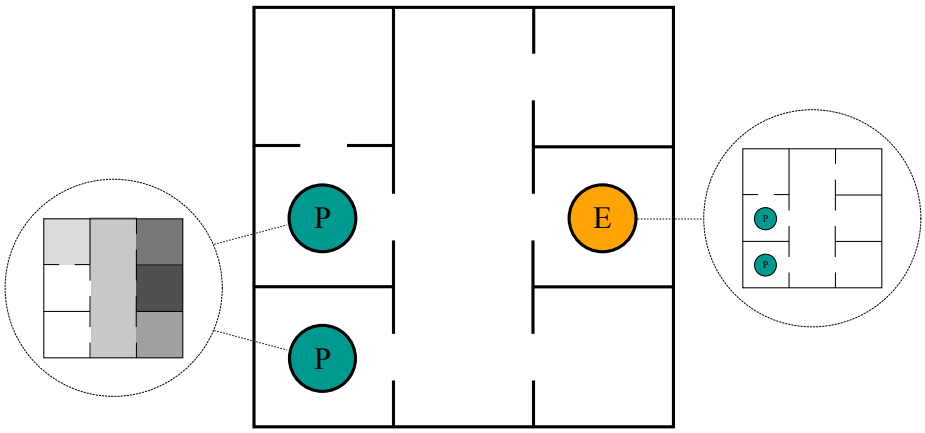

Figure 1: Illustration of search scenario in which a team of robots searches for an adversarial target that has perfect knowledge. The pursuers maintain a probabilistic belief over possible locations of the evader and generate pursuit strategies that maximize the probability of capture. Our proposed method, PATS, uses a collection of MCTS game trees to select pursuit actions and updates the belief based on the empirical statistics contained in those trees.

When a guaranteed approach is impossible, planning pursuit actions requires a belief about the location of the evader and model for how the evader will respond to pursuit actions. Existing non-adversarial multi-robot search methods (e.g. [5]–[7]) maintain a probabilistic belief and compute search actions that optimize the probability of target capture, assuming the target follows a static motion model (e.g. a random walk). We extend this idea to the adversarial setting in which the target actively tries to avoid capture. The pursuers maintain a probabilistic belief about the evader's location and use Monte Carlo tree search (MCTS) to generate actions. Because the evader's location is uncertain, the pursuers search in multiple game trees, each one representing a possible current configuration of the agents. Computation is distributed to these trees according to the believed probability of the associated configuration.

The search problem is sequential, so the pursuers must update their belief about the location of the target after each time step. In non-adversarial settings, the pursuers may evolve their belief according to a random walk or similar motion model. We instead use the results of the pursuit planning algorithm to inform this belief update. The game tree contains rich information about the actions an adversarial evader is likely to take in response to the selected pursuit joint action. We use these statistics generated during the planning process to efficiently update the pursuers' belief for the next time step.

Our contributions in this work include:

- A formulation of the search problem as an imperfect information variant of game-tree search,

- An MCTS-based method that computes pursuit plans given the pursuers' probabilistic belief of the evader's location, and
- A method for evolving the pursuers' target belief based the empirical statistics contained in the search tree, thus enabling re-execution of pursuit planning at the next time step.

2 Related Work

Our method builds upon work from both the multi-robot search and pursuit-evasion literature. For comprehensive overviews of these topics, please refer to the surveys of Chung, Hollinger, and Isler [8] and Robin and Lacroix [9] or the thesis of Noori [10].

When pursuing an adversarial evader, many methods assume worst-case characteristics for that adversary, such as complete knowledge, infinite speed, or unlimited computation. With such assumptions, the only way for pursuers to capture the evader is to plan strategies that force capture regardless of the evader's actions. Many researchers have studied the mathematical theory behind the number of pursuers required to guarantee capture in a particular environment or the properties of environments for which a pursuit team of a given size allows for such a strategy [11]. Others have considered the theoretical effects of different properties of the pursuit-evasion game, such as the effect of the evader's information access [12] and adversarialness [13].

Another area of research has focused on actually generating guaranteed search strategies, which are also known as clearing schedules. The problem of generating a clearing schedule with the minimum number of robots is NP-hard on general graphs, but can be solved in linear time on trees [14]. Recognizing this, Hollinger, Kehagias, and Singh [1] proposed Guaranteed Search with Spanning Trees (GSST), which transforms the structure of an environment into a spanning tree by positioning stationary guards at key locations. Different spanning trees may require different numbers of guards, so GSST considers many possible candidates and returns the clearing schedule that requires the fewest number of pursuers. Hollinger, Singh, and Kehagias [2] later show how GSST can be made more efficient (in terms of expected capture time) by including more pursuers who together search for the evader using a probabilistic, non-adversarial planning method [5]. The Iterative Greedy Node Search (IGNS) method of Kehagias, Hollinger, and Singh [3] also generates an offline clearing schedule. Although IGNS can generate clearing schedules that succeed regardless of the evader's capabilities, it can also generate strategies guaranteed to capture an evader with finite speed, which may reduce the number of pursuers required. Kolling, Kleiner, Lewis, *et al.* [15] demonstrated a large-scale real-world implementation of a guaranteed search algorithm by equipping eight humans with iPads that provided instructions on where to search. Kolling, Kleiner, and Carpin [16] recently proposed a method in which a team of robots traverse an environment in formations known as sweep lines. They show that

in many practical cases, they can compute sweep schedules (clearing schedules using sweep lines) in polynomial time.

The common thread in these guaranteed methods is that they all require some minimum number of robots in order to be executed. This minimum number depends on the complexity of the environment and on the particular guaranteed method used. If the pursuit team is not of sufficient size to execute such a strategy, however, these methods are inapplicable.

Other approaches have considered searching for a non-adversarial target using multiple robots. Some assume the target is stationary and provide search plans that minimize the expected time to target detection [17]. Others consider the target to be moving according to some non-adversarial motion model [18]. Hollinger, Singh, Djugash, *et al.* [5] formulate the problem of searching for a non-adversarial moving target as a partially observable Markov decision process (POMDP). They propose a method, finite horizon path enumeration (FHPE), that generates optimal pursuit trajectories within a finite time horizon, assuming a particular target motion model such as a uniform random walk. Hollinger, Singh, and Kehagias [2] later incorporate FHPE into the GSST guaranteed search method, using any extra searchers to reduce expected capture time. Renzaglia, Noori, and Isler [6] consider the effect of the target model (stationary or stochastic) on the resulting search strategies and find that a probabilistic planning approach performs better against a moving target than coverage path planning.

In this paper, we propose a method, PATS, that bridges the gap between existing guaranteed pursuit algorithms and non-adversarial search approaches. We address the case in which the pursuit team is too small to guarantee capture, a case which the guaranteed methods provide no solution for. We build from the ideas of the non-adversarial probabilistic search methods and demonstrate through empirical evaluation that PATS outperforms these methods when searching for an adversarial target.

3 Problem Formulation

The search game is sequential and unfolds over discrete time steps $t = 1, \ldots, T$. The agents occupy an environment discretized into an undirected graph $G(V, E)$ of $n = |V|$ vertices. See the survey of Bormann, Jordan, Li, *et al.* [19] for techniques that achieve such a segmentation.

The location of pursuer k at time t is given by $p_k(t) \in V$ for each of the K searchers. Similarly, the location of the evader at time t is given by $e(t) \in V$. The pursuers do not know the evader's location, but rather maintain a belief $\boldsymbol{b}(t) = [b_1, \ldots, b_n]$, which is a probability distribution over V. Each element b_i corresponds to the probability of the evader occupying vertex i.

The pursuers and evader alternate in taking actions. The game ends if the pursuers capture the evader, which occurs when a pursuer and evader occupy the same location (i.e. $e(t) = p_k(t)$ for some k). The reward earned by the pursuers depends on the number of time steps it took to capture the evader: for

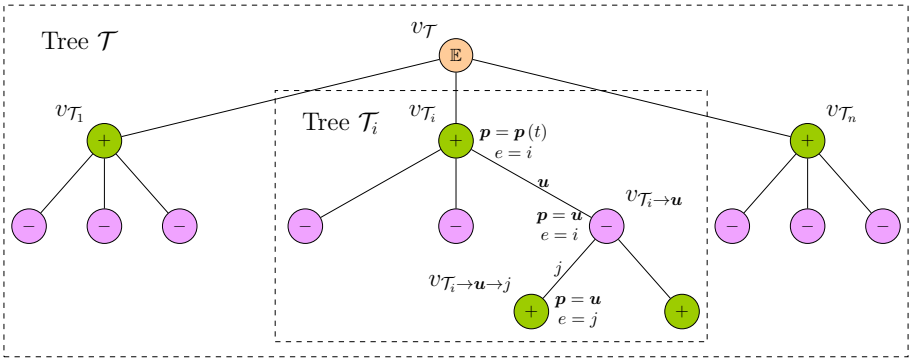

Figure 2: Illustration of the search tree $\mathcal{T}$. At the root is the expectation node $v_{\mathcal{T}}$. Each subtree $\mathcal{T}_i$ of the root is a game tree in which the evader begins in state i. Within the game tree $\mathcal{T}_i$, the pursuit team maximizes reward while the evader minimizes it. Computation is distributed to each game tree $\mathcal{T}_i$ based on the pursuers' belief that the evader is at location i, $b_i(t)$. The pursuers select the action $\boldsymbol{u}^*$ that maximizes expected reward. After taking action $\boldsymbol{u}^*$, the pursuers update their belief based on the expected actions of the evader, which comes from the empirical node statistics at each node $v_{\mathcal{T}_i \to \boldsymbol{u}^*}$

capture at time t_c the pursuit reward is $R_p = \gamma^{t_c}$, where $\gamma \in (0, 1]$ is a discount factor. If the pursuers fail to capture the evader before time $t_{\max}$, they receive zero reward. The search game is constant-sum, with evader reward $R_e = 1 - R_p$.

4 Approach

In this section, we describe a method to plan pursuit actions to search for an evader whose location is unknown to the pursuers. To develop the ideas we will need for this method, we first describe an approach to the perfect-information variant of this game in which the pursuers know the location of the evader at the time of planning. Once we have a technique in place to plan pursuit actions, we use the results of that method to update the pursuers' probabilistic belief of the evader's location.

4.1 Pursuit Planning with Known Evader Location

The challenge in the perfect-information variant of the game is to select a pursuit action leading toward capture of the evader, given that the evader will take future actions to avoid that fate. Game tree search provides a solution to such a task.

Consider a game tree rooted at a node representing a particular configuration of pursuers and evader. From this root node, the pursuers search forward in time, looking for actions that maximize pursuit reward R_p while knowing that

the evader will take opposing actions to minimize this reward.

More specifically, we use Monte Carlo tree search (MCTS), a type of game-tree search that randomly samples from the decision space and builds a search tree according to the results [20]. Unlike a traditional minimax game tree search, which evaluates non-terminal nodes according to a heuristic function, MCTS executes randomized forward simulations to sample from the distribution of rewards associated with a node and thereby determine its value. MCTS maintains empirical statistics for each node based on the results of these forward simulations. For MCTS node v, the visit count $N(v)$ is the total number of forward simulations that have passed through that node, and the total reward earned during those forward simulations is $Q(v)$. The ratio of these two statistics, $\frac{Q(v)}{N(v)}$, is an estimate of the game-theoretic value of the node. Note that by using Upper Confidence Bound for Trees (UCT) as a node expansion policy, the empirical estimate of an MCTS node's value will converge to the true game-theoretic (minimax) value given enough samples [21].

4.2 Pursuit Planning with Probabilistic Belief

As we have formulated it to this point, game tree search assumes that the pursuers know where the evader is at the time of planning. However, we are interested in the case that the pursuers only have a probabilistic belief about the evader's location. Because the evader can only begin at one location in a single game tree, we build n game trees, with each tree $\mathcal{T}_i$ corresponding to the evader being at location i. Each tree $\mathcal{T}_i$ has probability $b_i(t)$ of being the true game tree, according to the pursuers' belief.

An alternate way of viewing the collection of game trees is as a single tree with an expectation node $v_{\mathcal{T}}$ at its root, as illustrated in Fig. 2. The root $v_{\mathcal{T}_i}$ of each game tree $\mathcal{T}_i$ is a child of this expectation node. The approximate game-theoretic value of each such child node $v_{\mathcal{T}_i}$ is $\frac{Q(v_{\mathcal{T}_i})}{N(v_{\mathcal{T}_i})}$, as determined by the MCTS rollouts performed through that node. This value is an estimate of the reward the pursuers will earn by executing the best available action, given that the evader is in state i (i.e. that tree $\mathcal{T}_i$ represents the true configuration of agents). However, the pursuers do not know the evader's true location, but instead maintain a probabilistic estimate $b_i(t)$ that the evader is at location i at time t. From the perspective of the pursuers, the expected value of the root node of the entire tree, $v_{\mathcal{T}}$, then is given by

$$\begin{aligned} \mathbb{E}[v_{\mathcal{T}}] &= \sum_{i \in V} b_i(t)\, \mathbb{E}(v_{\mathcal{T}_i}) \\ &= \sum_{i \in V} b_i(t)\, \frac{Q(v_{\mathcal{T}_i})}{N(v_{\mathcal{T}_i})}. \end{aligned} \tag{1}$$

The pursuers have the same set of actions $U(t)$ available to them at the root node $v_{\mathcal{T}_i}$ of each tree $\mathcal{T}_i$. The value of each action $\boldsymbol{u} \in U(t)$ within game tree $\mathcal{T}_i$

is $\frac{Q\left(v_{\mathcal{T}_i \to \boldsymbol{u}}\right)}{N\left(v_{\mathcal{T}_i \to \boldsymbol{u}}\right)}$, where $v_{\mathcal{T}_i \to \boldsymbol{u}}$ is the child node of $v_{\mathcal{T}_i}$ corresponding to the pursuit team taking action $\boldsymbol{u}$. Therefore, the expected value of each pursuit action is given by

$$\mathbb{E}\left[\boldsymbol{u}\right] = \sum_{i \in V} b_i\left(t\right) \frac{Q\left(v_{\mathcal{T}_i \to \boldsymbol{u}}\right)}{N\left(v_{\mathcal{T}_i \to \boldsymbol{u}}\right)}. \tag{2}$$

A best action for the pursuers to take is an action $\boldsymbol{u}^*$ with highest expected reward:

$$\boldsymbol{u}^* = \arg\max_{\boldsymbol{u} \in U(t)} \mathbb{E}\left[\boldsymbol{u}\right]. \tag{3}$$

4.3 Evolving the Target Belief

Searching for an adversarial target is a sequential process. If the pursuers fail to capture the evader at one time step, they must repeat the planning process again at the next. While this occurs, the evader is also planning and executing actions. Therefore, after failing to capture the evader at one time step, the pursuers must evolve their belief forward to use in planning at the next time step. This evolved belief will then be input to the pursuit planning process of Section 4.2 at the next time step.

Non-adversarial search methods evolve their probabilistic belief by applying a fixed Markovian target model. For example, Hollinger, Singh, Djugash, *et al.* [5] update the belief as if the target performed a uniform random walk. However, such a belief update does not model the behavior of an adversarial target actively working to avoid capture.

To evolve the pursuers' belief, we draw inspiration from the Monte Carlo belief state update method of Silver and Veness [22]. They consider the case of evolving a belief state in a large POMDP. Because an exact Bayesian update is intractable for large belief state spaces, Silver and Veness sample particles from the belief space and evolve them using a black-box simulator. They approximate the new belief with all particles whose evolved state is consistent with the actual observation made by the agent.

In a similar way, the pursuers have access to the results of a series of forward simulations after executing the planning method of the previous section. Each game tree $\mathcal{T}_i$ has a child node $v_{\mathcal{T}_i \to \boldsymbol{u}^*}$ corresponding to the action $\boldsymbol{u}^*$ selected by the pursuers. Consider each MCTS forward simulation that passed through this node to be a sampled particle. Each of these simulations passes through a successor state corresponding to a particular action of the evader. The relative frequency of each evader action in these forward simulations provides an estimate of the probability of an adversarial evader taking those actions in reality.

These action probabilities provide the means to evolve the belief according to an adversarial random walk. Rather than uniformly distributing the probability mass from a location to neighboring locations, we distribute it according to the probability of an evader moving to each neighboring location.

Concretely, according to the MCTS forward simulations, the probability of an evader at location j moving to neighboring location i in response to pursuit

Algorithm 1 Probabilistic Adversarial Target Search (PATS)

for B iterations **do**
  Sample $i \sim \boldsymbol{b}(t)$
  GrowTree($\mathcal{T}_i$)
$\boldsymbol{u}^* \leftarrow \arg\max_{\boldsymbol{u} \in U(t)} \sum_i b_i(t) \frac{Q\left(v_{\mathcal{T}_i \rightarrow \boldsymbol{u}}\right)}{N\left(v_{\mathcal{T}_i \rightarrow \boldsymbol{u}}\right)}$
ExecuteJointAction($\boldsymbol{u}^*$)
if evader not captured **then**
  for pursuer location i **do**
    $b_i(t+1) \leftarrow 0$
  for location i not occupied by a pursuer **do**
    $b_i(t+1) \leftarrow \sum_{j\,:\,ji \in E} b_j(t) \frac{N\left(v_{\mathcal{T}_j \rightarrow \boldsymbol{u}^* \rightarrow i}\right)}{N\left(v_{\mathcal{T}_j \rightarrow \boldsymbol{u}^*}\right)}$
  Normalize($\boldsymbol{b}(t+1)$)

action $\boldsymbol{u}^*$ is given by $\frac{N\left(v_{\mathcal{T}_j \rightarrow \boldsymbol{u}^* \rightarrow i}\right)}{N\left(v_{\mathcal{T}_j \rightarrow \boldsymbol{u}^*}\right)}$. The probability of the evader moving to location i is then the sum of these probabilities for each neighbor j of i, weighted by the probability of the evader starting in j. That is,

$$b_i(t+1) = \sum_{j\,:\,ji \in E} b_j(t) \frac{N\left(v_{\mathcal{T}_j \rightarrow \boldsymbol{u}^* \rightarrow i}\right)}{N\left(v_{\mathcal{T}_j \rightarrow \boldsymbol{u}^*}\right)} \tag{4}$$

4.4 Complete Search Algorithm

Algorithm 1 details the complete method for planning pursuit actions, which we call probabilistic adversarial target search (PATS). Note that growing all of the search trees equally would waste computation on unlikely configurations. We therefore distribute a fixed computational budget B according to the believed probability of each configuration. Each loop iteration samples a tree $\mathcal{T}_i$ according to belief $\boldsymbol{b}(t)$ and performs a single forward simulation on that tree using the method GrowTree. GrowTree encompasses the steps of a standard MCTS iteration, which include selecting a node of the tree for expansion, executing a forward simulation from that node, and propagating the results back up the tree. Each call to GrowTree adds a single leaf node to the tree.

After the computational budget is exhausted, the function computes the action with highest expected value. It then evolves the pursuers' belief based on the node statistics contained in the search tree. The belief is set to zero for all locations occupied by the pursuers. For all other locations, the belief is updated according to the expected probability of the evader moving to that location i from an adjacent location j, weighted by the current belief $b_i(t)$. This probability comes from the empirical node statistics of the MCTS search trees.

Figure 3: Floorplan of the simulated test environment, an office building first introduced by Hollinger, Singh, Djugash, *et al.* [5].

5 Evaluation

5.1 Experimental Setup

We conducted a series of simulated experiments consisting of two pursuers searching for a single evader. We compare the performance of PATS to that of the FHPE method of Hollinger, Singh, Djugash, *et al.* [5]. FHPE assumes that the search target moves according to a fixed Markovian policy, such as a uniform random walk. It then selects the set of paths that maximizes the probability of target capture out to a finite horizon. Between time steps, FHPE evolves the pursuers' belief according to the fixed motion policy.

We tested against an evader implementation that plans using MCTS. Specifically, we implemented the plain UCT variant proposed by Kocsis and Szepesvári [21]. This implemented evader knew the locations of each pursuer throughout the experiments. Within the confines of its allocated computational budget, the evader planned actions that maximized its own expected reward. We varied the evader's computational budget (in terms of MCTS samples) to test against a range of adversaries: a budget of 1 sample corresponds to a non-adversarial random walk agent, whereas increasing the budget results in an increasingly capable and adversarial evader.

Our simulated experiments took place in the test environment shown in Fig. 3. This environment, which was first introduced by Hollinger, Singh, Djugash, *et al.* [5], has 60 nodes and 124 edges. Hollinger, Kehagias, and Singh [1] demonstrated a guaranteed search schedule for the environment using three evaders. To the best of our knowledge, no such schedule has been achieved for the two-evader team we use in this evaluation. This evaluation therefore addresses our scenario of interest, namely situations in which the environment is too complex or the pursuers too few in number to guarantee capture.

We conducted 400 trials for each tested parameter setting. In each trial, the pursuers and evader began at distinct, randomly generated locations. The pursuers initially knew the location of the evader but received no further information about the evader's location until capture.

The reward assigned to the pursuers for a particular trial followed the definition given in Section 3; that is, $R_p = \gamma^{t_c}$ when the pursuers captured the evader at time t_c, and $R_p = 0$ if the pursuers failed to capture the evader before time $t_{\max}$. We used discount factor $\gamma = 0.98$ and time step limit $t_{\max} = 120$, though the specific values do not affect our conclusions.

Error bars in all figures correspond to one standard error of the mean.

5.2 Effect of FHPE Search Depth

An important factor in the effectiveness of any pursuit planning method is the computational budget allocated to it, which affects the depth its search tree reaches. The computational budget of PATS is expressed in terms of the number of search nodes expanded (or equivalently, the number of forward simulations performed). Recall that the MCTS search trees grow asymmetrically, so the tree's leaves do not reach a uniform depth. In contrast, FHPE enumerates all sets of paths of a selected length, which corresponds to the depth of the search tree.

Because the two tested methods have different ways of varying computational budget, it is important to control for this parameter. We therefore begin by evaluating the performance of FHPE as its search depth is varied.

Fig. 4 shows the results of this experiment. Each curve corresponds to the reward earned by pursuers planning with FHPE against an evader with a particular computational budget. For any search depth, the pursuers' reward decreases as the evader's budget increases. Against a fixed-budget evader, pursuit reward saturates quickly with FHPE search depth. This indicates that a non-adversarial method cannot perform better in an adversarial setting simply by planning farther into the future.

The saturation effect occurs more slowly against the more capable adversaries, but a search depth of 12 approximately maximizes FHPE's performance against a range of adversaries. We will use search depth to compare the performance of FHPE to PATS in Section 5.4.

5.3 Effect of PATS Budget

Having tested the effect of FHPE's search depth on pursuit reward, we now evaluate the performance of PATS as a function of its computational budget.

Fig. 5 shows the results of this experiment. Each curve represents the reward earned by pursuers with PATS against an evader with a particular computational budget. The effect of the PATS computational budget is most pronounced against the stronger evaders. For example, the reward earned by the PATS pursuers against the strongest evader increases from approximately 0.5 with a

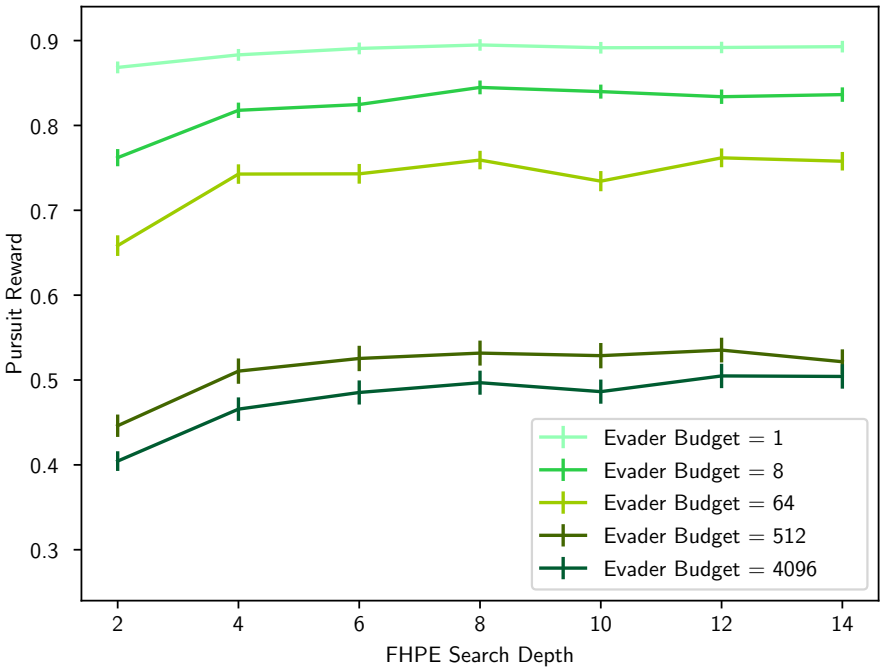

Figure 4: Reward earned by pursuers planning with the FHPE algorithm [5] with variable search depth. Pursuit reward saturates quickly: increasing FHPE search depth leads to diminishing returns.

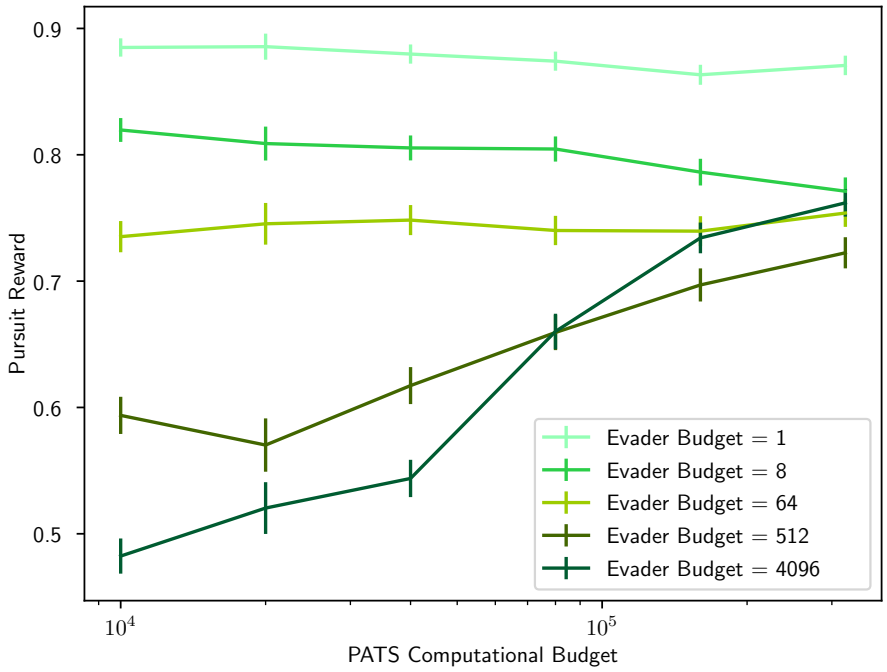

Figure 5: Reward earned by pursuit team as a function of pursuit planning budget. A larger planning budget helps the pursuers account for the adversarial behavior of a strong evader.

computational budget of 10,000 to approximately 0.75 with a computational budget of 320,000.

Interestingly, the pursuers with the largest computational budget perform better against the strongest evader (budget 4096) than they do against the slightly weaker evader (budget 512). Recall that PATS uses game tree search, so the evader modeled in the search tree tries to minimize pursuit reward while the pursuers try to maximize it. Increasing the PATS computational budget also increases the budget of the evader modeled in the game tree. Therefore, increasing the PATS computational budget can lead to better performance against stronger adversaries.

5.4 Effect of Evader Budget

We have now evaluated the effect of computational budget on the non-adversarial method (FHPE) of Hollinger, Singh, Djugash, *et al.* [5] and our own method (PATS). To complete the comparison, we fix the computational budget of both methods and evaluate performance against evaders of varying capabilities. Specifically, we fixed the search depth of FHPE at 12 and the budget of PATS at 320,000. We varied the budget of the evader between 1 (non-adversarial) and 4096 (adversarial).

Fig. 6 shows the results of this experiment. At the left of the plot, the evader has a low computational budget, so its behavior is non-adversarial. FHPE performs well in this region since its non-adversarial assumption matches the actual behavior of the evader. As the evader's budget increases, however, PATS significantly outperforms FHPE. The reward earned by PATS decreases by only 12 percent between the least and most adversarial evaders; the FHPE reward decreases by 43 percent across that same range. Against the strongest adversary (budget 4096), the pursuit reward of PATS corresponds to capturing the evader in an average of 14 steps, compared to an average capture by FHPE in 31 steps. Overall, this demonstrates the effectiveness of probabilistic search for an adversarial target.

6 Conclusion

In this paper, we proposed Probabilistic Adversarial Target Search (PATS), a method by which a team of robots can search for an adversarial target. PATS is particularly suited for scenarios in which the environment is too complex or the pursuers too few in number to guarantee capture of an evader. PATS computes pursuit plans that approximately maximize the probability of capturing an evader with finite speed but perfect knowledge. PATS uses Monte Carlo tree search (MCTS) to compute pursuit joint actions based on a probabilistic belief about the evader's location. Because the search process is sequential, PATS provides a means to update this probabilistic belief using the results of the MCTS planning procedure. We demonstrated that PATS outperforms an existing probabilistic search method in a simulated scenario for which guaranteed

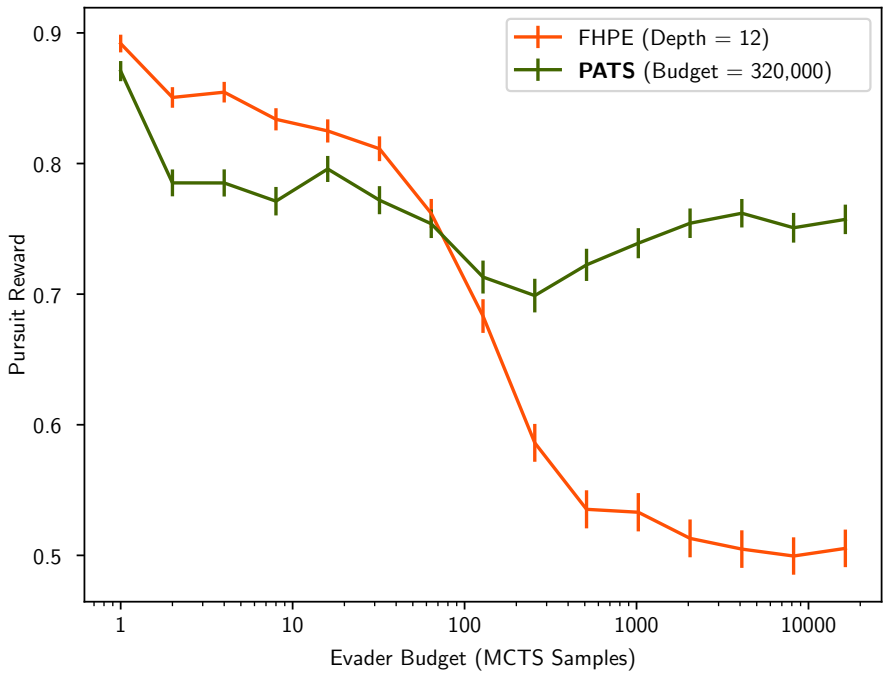

Figure 6: Reward earned by pursuit team against a variable-budget evader. Though the non-adversarial method (FHPE [5]) performs well against a weak adversary, our proposed adversarial method (PATS) continues to receive a high level of reward and outperforms FHPE as the evader becomes more capable.

capture is impossible.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1256260 and the National Science Foundation Grant No. NRI 1830615. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] G. Hollinger, A. Kehagias, and S. Singh, "GSST: Anytime Guaranteed Search," *Autonomous Robots*, vol. 29, no. 1, pp. 99–118, 2010.

[2] G. Hollinger, S. Singh, and A. Kehagias, "Improving the Efficiency of Clearing With Multi-Agent Teams," *International Journal of Robotics Research*, vol. 29, no. 8, pp. 1088–1105, 2010.

[3] A. Kehagias, G. Hollinger, and S. Singh, "A Graph Search Algorithm for Indoor Pursuit/Evasion," *Mathematical and Computer Modelling*, vol. 50, no. 9-10, pp. 1305–1317, 2009.

[4] A Kleiner and A. Kolling, "Guaranteed search with large teams of unmanned aerial vehicles," in *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 2977–2983.

[5] G. Hollinger, S. Singh, J. Djugash, and A. Kehagias, "Efficient multi-robot search for a moving target," *International Journal of Robotics Research*, vol. 28, no. 2, pp. 201–219, 2009.

[6] A. Renzaglia, N. Noori, and V. Isler, "The Role of Target Modeling in Designing Search Strategies," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4260–4265.

[7] L. D. Stone, J. O. Royset, and A. R. Washburn, "Path-constrained search in discrete space and time," in *Optimal Search for Moving Targets*, Springer International Publishing, 2016, ch. 4, pp. 189–207.

[8] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous Robots*, vol. 31, no. 4, p. 299, 2011.

[9] C. Robin and S. Lacroix, "Multi-Robot Target Detection and Tracking: Taxonomy and Survey," *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, 2015.

[10] N. Noori, "Adversarial and Stochastic Search for Mobile Targets in Complex Environments," PhD thesis, University of Minnesota, 2016.

[11] A. Bonato, *The game of cops and robbers on graphs*. The American Mathematical Society, 2011.

[12] A. Kehagias, D. Mitsche, and P. Pralat, "The role of visibility in pursuit/evasion games," *Robotics*, vol. 3, no. 4, pp. 371–399, 2014.

[13] ——, "Cops and invisible robbers: The cost of drunkenness," *Theoretical Computer Science*, vol. 481, pp. 100–120, 2013.

[14] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou, "The complexity of searching a graph," *Journal of the ACM*, vol. 35, no. 1, pp. 18–44, 1988.

[15] A. Kolling, A Kleiner, M Lewis, and K Sycara, "Computing and executing strategies for moving target search," in *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 4246–4253.

[16] A. Kolling, A. Kleiner, and S. Carpin, "Coordinated search with multiple robots arranged in line formations," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 459–473, 2018.

[17] L. D. Stone, J. O. Royset, and A. R. Washburn, "Search for a stationary target," in *Optimal Search for Moving Targets*, Springer International Publishing, 2016, ch. 2, pp. 189–207.

[18] ——, "Search for a moving target in discrete space and time," in *Optimal Search for Moving Targets*, Springer International Publishing, 2016, ch. 3, pp. 189–207.

[19] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele, "Room Segmentation: Survey, Implementation, and Analysis," in *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE, 2016, pp. 1019–1026.

[20] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[21] L. Kocsis and C. Szepesvári, "Bandit-based Monte-Carlo planning," in *Proceedings of the European Conference on Machine Learning*, Springer, 2006, pp. 282–293.

[22] D. Silver and J. Veness, "Monte-Carlo Planning in Large POMDPs," in *Proceedings of the Advances in Neural Information Processing Systems Conference*, 2010, pp. 2164–2172.