Exploiting Key Events for Learning Interception Policies

Yuan Chang and **Gita Sukthankar**
Department of EECS
University of Central Florida
Orlando, FL 32816, U.S.A
{*dvchangyuan@gmail.com, gitars@eecs.ucf.edu*}

Abstract

One scenario that commonly arises in computer games and military training simulations is predator-prey pursuit in which the goal of the non-player character agent is to successfully intercept a fleeing player. In this paper, we focus on a variant of the problem in which the agent does not have perfect information about the player's location but has prior experience in combating the player. Effectively addressing this problem requires a combination of learning the opponent's tactics while planning an interception strategy. Although for small maps, solving the problem with standard POMDP (Partially Observable Markov Decision Process) solvers is feasible, increasing the search area renders many standard techniques intractable due to the increase in the belief state size and required plan length. Here we introduce a new approach for solving the problem on large maps that exploits *key events*, high reward regions in the belief state discovered at the higher level of abstraction, to plan efficiently over the low-level map. We demonstrate that our hierarchical key-events planner can learn intercept policies from traces of previous pursuits significantly faster than a standard point-based POMDP solver, particularly as the maps scale in size.

Introduction

Designing non-player characters with good interception skills is an important aspect of creating intelligent automated non-player characters that serve as effective and interesting adversaries. This is particularly relevant for military training simulations where the trainees must apply skills learned from combats with virtual adversaries to real-world combat situations. Many different game genres, including first-person shooters, squad games, and sports games, include segments in which the game AI has to chase the player, although the emphasis is on creating adversaries that are unpredictable and fun to play. Interception can be formulated as a classic multi-agent pursuit problem in which a group of predators has to apprehend the fleeing prey. However, unlike completely simulated prey that are often modeled as moving randomly or merely maximizing distance from the pursuit, actual human users can employ diverse and sophisticated evasion strategies. Yet one weakness that humans have is repeatability, often preferring familiar map regions and repeating favorite tactics (Tastan, Chang, and Sukthankar 2012). In this case, a learning predator holds a decisive advantage over a non-learning system.

In this paper, we demonstrate a model for rapidly learning effective interception strategies based on previous game data. Our method uses data from previous pursuits to build a probability map of the prey's movement patterns. Using this map, we frame the problem as a Partially Observable Markov Decision Process (POMDP) in which the prey's position is unknown until the predator is within a certain radius and the predator is rewarded for successfully intercepting the prey before it achieves its goal and exits the map. To achieve good interception rates in complex scenarios, coupling learning with effective planning over a longer time horizon is important. Without both elements, the predator can succeed at learning and predicting the prey's future position yet ultimately fail at the interception task by selecting a non-optimal route. Interception is achieved by moving to where the prey will be in the future, rather than continuing to trail the prey, assuming that the predator and prey have comparable speeds.

This is very similar in spirit to the classic robotic tag problem used as a benchmark for POMDP solvers; however in this case the predator relies more on previously learned game experience and less on incoming observations to catch the prey. Although the POMDP is an expressive model and can also be simplified to represent a dynamic domain with mixed observable and unobservable state elements, it is greatly hampered by enlargements in state space which lead to exponential increases in the size of belief state, "the curse of dimensionality".

Hierarchical representations, such as the one proposed in this paper, can substantially reduce the state size, but there is always the risk that the abstract solution will not translate effectively to the original domain. To overcome this, we propose using the hierarchical abstraction to identify key events, points in the spatio-temporal trajectory with higher reward, and a low-level planner to identify the best path to intersect the key events. In this paper, we demonstrate that our method achieves comparable results to a point-based solver with significantly reduced computation time. Moreover, it represents a significant improvement over the greedy approach of having the low-level planner create macro actions to directly execute the POMDP policy generated at the higher level of abstraction. The next section describes related work in the

Copyright © 2013, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

area of learning interception strategies.

Related Work

Two general approaches to improving the computation time of POMDP planners are the use of hierarchical abstractions and macro actions. Pineau et al. introduced a hierarchical POMDP framework in which the original POMDP is manually decomposed into a collection of POMDPs that are individually smaller than the original POMDP, yet collectively define a complete policy (Pineau, Roy, and Thrun 2001). This greatly reduces the state space and computational complexity but requires solving multiple POMDPs. In our planner, we only use a POMDP solver at the highest level of abstraction and a breadth-first search planner for the lower level of abstraction. Charlin et al. demonstrate how a hierarchy can be found automatically by formulating the planning problem as a non-convex optimization problem with constraints from parameters of the policy (Charlin, Poupart, and Shioda 2007). Although in many cases, the POMDP mainly offers computational time reductions and not necessarily performance improvements, Toussiant et al. developed a maximum likelihood approach for solving a hierarchical POMDP that yields better performance than a standard POMDP solver when the hierarchical solution exists. (Toussaint, Charlin, and Poupart 2008)

An alternative method is to reduce the plan depth by having the planner combine longer sequences of actions, also known as macros. Theocharous et al. proposed an approximate planning method using this strategy (Theocharous and Kaelbling 2003). In their system, a macro action is a sequence of hand coded actions that can be taken at each state. A fast interpolation procedure is employed to compute the value of a belief point using neighboring states. He et al. proposed an online technique, the planning under uncertainty algorithm (PUMA), which first samples subgoals and then searches these subgoals using a MDP (He, Brunskill, and Roy 2010). This method automatically constructs macro actions without human labor; based on the reachability of each subgoal, a macro action is encoded. Forward search is used to decide which macro action to take. This type of approach was used in a target-tracking vehicle simulation with good results (He, Bachrach, and Roy 2010). In our approach, a local planner is used to automatically create action sequences for moving around the actual map. Rather than using the planner to create macro actions which are directly associated with the abstracted states, the local planner creates a trajectory that intersects the key events, high reward points in the spatio-temporal trajectory identified by the abstract POMDP. The next section covers the details of our proposed approach.

Method

The pursuit scenario transpires in a $N \times N$ grid world with entry gates and obstacles. The layout is designed to correspond to the types of maps commonly seen in first-person shooter games such as Unreal Tournament. The "prey" (meant to represent the human player) enters the map from one gate and leaves through another. The "predator" follows

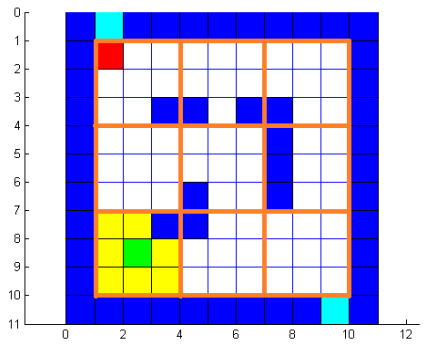


Figure 1: Example 11×11 map of predator-prey pursuit problem. Blue cells denote walls and obstacles; cyan denotes entry gates for prey; the green and red cells mark the predator and prey, respectively. The predator's limited visibility is shown in yellow. A coarser version of the map (orange) is used to create the abstract POMDP. The predator must intercept the prey before the latter leaves the map, using a policy computed offline by our planners.

the interception policy extracted from the POMDP solver and attempts to catch the prey before it exits the map; this is analogous to the task that game AI non player characters face when they are chasing human players while guarding an area. The final interception policies are rated on interception rate, time to interception, average reward earned, and computation time based on map size, N. Figure 1 shows an example map.

Our proposed approach can be outlined as follows:

1. previous pursuit sequences are used to learn a probability map based on the prey's action distribution;
2. the map is re-represented as a 3×3 region and a valid POMDP model is created for the abstract map;
3. the hierarchical POMDP is solved using finite-time value iteration;
4. the pursuit sequence data is then used to estimate the spatio-temporal points that result in peak rewards (key events);
5. a detailed path on the original map that intersects the key events is calculated using a local planner.

Partially Observable Markov Decision Process

Although the Partially Observable Markov Decision Process is a highly expressive model capable of representing many types of planning problems, increases in state space size cause exponential increases in the belief state size. Hence, although it is possible to represent the original problem directly as a POMDP, we create a hierarchical model to calculate an intercept policy at a coarser level and solve it using finite-time value iteration.

The original discrete-time POMDP model is a 6-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \Omega, \mathcal{R})$.

- $\mathcal{S}$ is the set of states, which is simply defined by the predator and prey's locations on the complete map;

- $\mathcal{A}$ is the set of actions available to the predator, including moving up, down, left and right;
- $\mathcal{O}$ is the set of observations made by the predator. The predator's own location is completely observable, but the location of the prey is observed only if it lies within the predator's visibility region;
- $\mathcal{P}$ is the set of transition probabilities between states, indexed by action. The predator's movement is completely deterministic, but the prey's transitions are governed by a learned transition model extracted from previous pursuit sequences;
- Ω is the set of conditional observations;
- $\mathcal{R}$, reward, depends on both the system state and predator's action selection, $\mathcal{A} \times \mathcal{S} \Rightarrow \mathcal{R}$. If the predator intercepts the prey, it collects a reward of 100;
- $\mathcal{B}$ is a set of beliefs, a probability distribution of the prey's current location;
- γ is a discount factor, used to force the predator to intercept the prey early;
- τ is the operator denoting the belief update process.

The system dynamics are as follows. When the system is in a state $s \in \mathcal{S}$ and the predator takes an action $a \in \mathcal{A}$, it will cause the system to transition to state s' with probability $P(s'|s,a)$. The predator will then gain a reward $r \in \mathrm{R}$.

At each time period t, the predator updates its belief after taking an action a and observing o. The new belief will solely depends on the previous belief at time $t-1$, the action taken at current time $t-1$ and the observation made at time t after taking the action. The belief update is denoted as:

$$b_t = \tau(o_t, b_{t-1}, a_{t-1}); \tag{1}$$

After taking action a_{t-1} resulting in state s_t, the predator observes $o_t \in \mathcal{O}$ with probability $\Omega(o_t|s_t, a_{t-1})$. Here in the predator-prey pursuit problem, observations are only related to system state. Therefore the probability is conditioned only on state $\Omega(o_t|s_t)$

$$b_t(s_t) = \eta\Omega(o_t|s_t) \sum_{s_{t-1}\in\mathcal{S}} P(s_t|s_{t-1}, a_{t-1}) b_{t-1}(s_{t-1}) \tag{2}$$

where $\eta = 1/Pr(o_t|b_{t-1}, a_{t-1})$ is a normalizing factor that ensures the probabilities sum to 1.

$$Pr(o_t|b_{t-1}, a_{t-1}) = \sum_{s_t\in\mathcal{S}} \Omega(o_t|s_t) \sum_{s_{t-1}\in\mathcal{S}} (s_t|s_{t-1}, a_{t-1}) b_{t-1}(s_{t-1}) \tag{3}$$

We define the expected reward for policy π from belief b_0:

$$V^{\pi}(b_0) = \sum_{t=0}^{T} (\gamma^t r(b_t, a_t)) \tag{4}$$

$$\pi^* = \arg\max_{\pi} \sum_{t=0}^{T} (\gamma^t r(b_t, a_t)) \tag{5}$$

where

$$r(b_t, a_t) = \sum_{s_t\in\mathcal{S}} b_t(s_t) r(s_t, a_t) \tag{6}$$

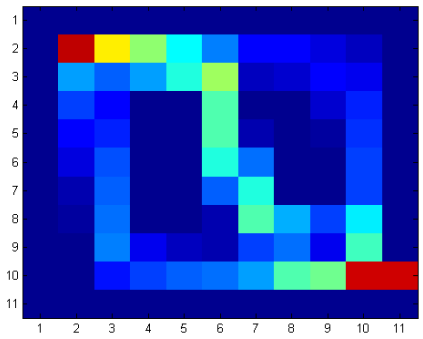


Figure 2: Probability map of prey evasion habits (red cells denote higher probability). This map was learned with a set of sequences in which the prey enters from the upper left and leaves the map at the lower right. Three paths are clearly visible, of which the middle is most likely.

The optimal policy π^* yields the highest reward value, V^*, and can be calculated through a series of Bellman updates:

$$V^*(b_{t-1}) = \max_{a_{t-1}\in\mathcal{A}} \left[r(b_{t-1}, a_{t-1}) + \gamma V^*_{a_{t-1}}(b_t) \right] \tag{7}$$

Learning the Prey Action Distribution Model

The main advantage a learning predator has over a non-learning one is the ability to anticipate that the prey is highly likely to repeat its favorite tactics. In this scenario, the goal of the prey is to exit the map quickly from a different entrance. The predator, who is tasked with guarding the area, has to intercept the prey before it departs the area. To learn the prey's habits, the predator uses the previously viewed pursuit sequences to construct a probability map showing which areas the prey frequents (Figure 2).

In the baseline condition, we use the pursuit data simply to learn the transition model used by a point-based POMDP solver on the original map. In our proposed method, we additionally use the data to evaluate the policy identified by the hierarchical POMDP and extract peaks in the expected reward, the key events.

Constructing the Hierarchical Representation

Separating planning into a two-level hierarchy with an abstract representation of the world allows us to tackle larger maps than are possible with the point-based solver. One state in the high level representation represents a collection of states in the original state space. The intuition is that solving the abstract POMDP guides the predator toward the promising region of the map, and the low-level planner handles the problem of obstacle avoidance.

Asynchrony problems can occur when we try to use a small state space to describe a large state space. To reduce asynchrony problems, the transition probability, reward and actions of the abstract model need to be recalculated to approximate the system dynamics of the original POMDP. In the pursuit problem, a reduced grid world can be used to approximate the original grid world. The width and the height

of the reduced grid world are factors of the original grid world.

Each state in the small grid world is a *macro state* representing several states in the original grid world. Each *macro action* represents an as yet undetermined sequence of actions leading from one macro state to another macro state. The transition probability between macro states is determined by 1) verifying the existence of direct path on the original map and 2) calculating the time required to transition between the two states by following the minimum length path. The optimal policy for the abstract POMDP is then solved using value iteration over a finite horizon dictated by the size of the map as described in Section .

Extracting Key Events

After generating the policy in the abstract representation, a low-level planner is required to convert the abstract policy into an actionable pursuit sequence for the predator. One simple way to do this is simply to use a breadth-first search path planner to calculate the minimum length path that avoids obstacles in the detailed map while achieving the action recommended by the abstract policy.

However, the interception problem has a relatively sparse reward function since the predator spends a long time traversing relatively unproductive sections of the map to reach the more promising areas. These promising areas can be identified using the probability map learned from the prior pursuit data. Using this probability map, we can calculate the expected undiscounted reward yielded by different macro states while following the policy calculated by the hierarchical POMDP. Our aim is to identify the macro states that frequently yield interceptions and to determine what time the interception is likely to occur. These states along with the time information and undiscounted reward are recorded as *key events*, $e_i(s_i, t_i, r_i), e_i \in E$.

In the predator-prey problem, reward is sparse (only occurring when the prey is captured), so the number of events with non-zero reward is relatively small. In more general cases, where an agent may be constantly receiving positive reward, a threshold can be introduced to limit the number of key events. Figure 3 shows the macro states, actions, and one possible sequence of key events.

Results

We benchmarked our algorithm against a general solver, SARSOP (Kurniawati, Hsu, and Lee 2008), a point-based POMDP planner that calculates an optimal sampling based on the reachability of belief space. SARSOP can be run at a variable precision level so we evaluated both a high and low precision setting. The high precision setting is more computationally intensive but has the potential to find a better policy. All tests were run on a quad core 3.2 GHz Intel Xeon machine with 18 GB memory; computational time is reported in wall clock seconds. The experiments consisted of evaluating the interception outcomes of 10,000 pursuit sequences from three different predator starting locations (A, B, and C shown in Figure 4)). Location B is farthest away from the prey's starting location and thus has the potential

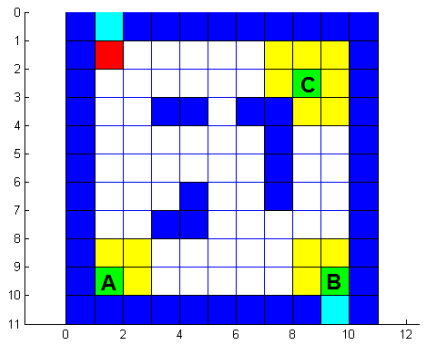


Figure 4: Starting locations for prey (red) and predator (A, B, C).

to require longer paths for interception. In the case where the predator fails to intercept the prey, the simulation stops after 100 time steps and the interception time is noted as the maximum possible (100).

Tables 1, 2, 3, 4 summarize our experimental results. We make the following observations:

- the proposed method consistently obtains interception time (Fig. 5) and rewards (Fig. 6) that compare favorably to high-precision SARSOP;
- the hierarchical method significantly reduces the running time required to calculate the policy in all cases;
- the hierarchical key events planner continues working on the larger maps (23×23 and 53×53) where SARSOP fails with out of memory errors and problems loading the policy file;
- the key event method significantly outperforms a greedy local search baseline (Fig. 7) in which the low-level planner simply calculates action sequences to directly execute the policy identified by the abstract POMDP.

Conclusion and Future Work

The ability to learn interception policies in an important aspect of creating an effective adaptive opponent in games and simulations. In this paper we present a method for learning interception policies that can scale to large maps that are problematic for general POMDP solvers. Our proposed method has a significantly lower computation time while yielding interception results comparable to a state of the art planner.

By gathering data from prior pursuit sequences, we can accurately estimate the expected reward of macro states and actions from the abstract policy, facilitating the identification of key events, spatio-temporal points where the predator is highly likely to apprehend the prey. Having the low level planner create a trajectory that simply attempts to intersect these key events rather than directly executing the abstract policy makes the predator significantly more effective at catching the prey. In future work, we plan to extend our approach to handle larger multi-agent pursuit problems and deeper state and action hierarchies.

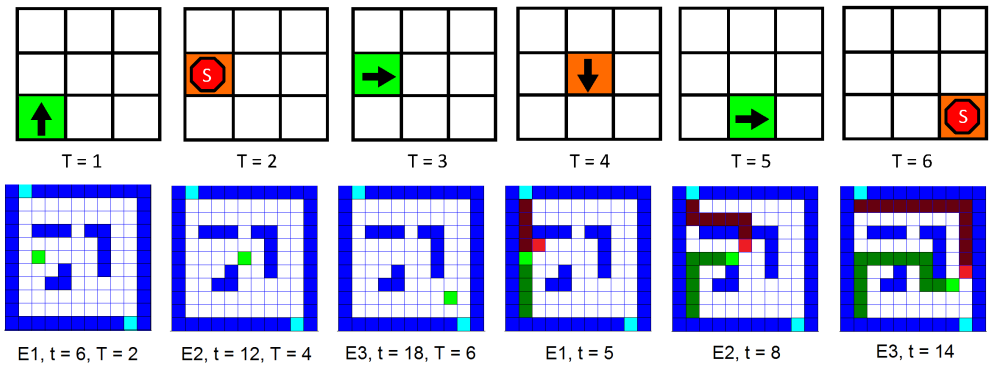


Figure 3: Optimal abstract policy found by proposed model. T and t denote time in the abstract representation and real world, respectively. $E1$, $E2$, $E3$ are the key events in the policy. The top row shows the policy generated for macro states, with arrows denoting the predator's movement actions. The STOP signs indicate that the predator should wait in the current location. The green grid cells mark macro states without immediate reward and orange mark those with a positive reward. Bottom row (left) shows locations of the key events in green at the noted times. Bottom row (right) shows execution samples of the chase with predator in green and prey in red. The bright cells show their current locations while darker ones are traces showing visited cells. The predator successfully intercepts at t=14.

Table 1: Comparison of rewards, intercept and run times on 11×11 map.

Policy	Steps to Intercept			Actual Reward			Running Time		
	Loc A	Loc B	Loc C	Loc A	Loc B	Loc C	Loc A	Loc B	Loc C
Hierarchical	6.29±1.94	13.43±0.34	5.34±1.18	90.19±3.34	77.83±0.54	91.75±2.07	0.22	0.21	0.21
SARSOP (low)	15.53±0.13	15.04±0.27	14.42±0.01	74.56±0.22	75.31±0.04	76.26±0.23	5.38	5.53	5.43
SARSOP (high)	8.98±1.66	14.33±0.30	5.35±1.19	85.45±2.75	76.36±0.47	91.74±2.10	8.15	15.36	8.78

Table 2: Comparison of rewards, intercept and run times on 17×17 map.

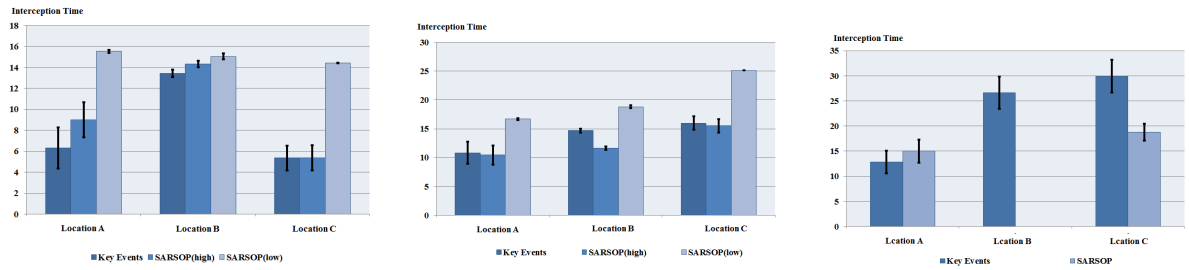
Policy	Steps to Intercept			Actual Reward			Running Time		
	Loc A	Loc B	Loc C	Loc A	Loc B	Loc C	Loc A	Loc B	Loc C
Hierarchical	10.85±2.21	14.693±3.12	15.99±2.16	82.46±3.18	76.20±4.86	74.2±2.92	0.54	0.53	0.53
SARSOP (low)	15.53±0.13	15.04±0.27	14.42±0.01	74.56±0.22	75.31±0.04	76.26±0.23	136.43	108.11	112.81
SARSOP (high)	10.46±1.78	11.65±1.92	15.51±1.50	82.88±2.70	76.36±0.47	91.74±2.10	283.67	303.49	449.15

Table 3: Comparison of rewards, intercept and run times on 23×23 map. The map is too large for SARSOP (high).

Policy	Steps to Intercept			Actual Reward			Running Time		
	Loc A	Loc B	Loc C	Loc A	Loc B	Loc C	Loc A	Loc B	Loc C
Hierarchical	12.82±2.24	26.66±3.22	29.916±3.24	79.42±3.13	56.73±4	56.61±4.08	0.92	0.91	0.89
SARSOP (low)	15.03±2.30	–	18.8±1.67	75.96±3.12	–	70.50±2.41	2616	11341	2625
SARSOP (high)	–	–	–	–	–	–	–	–	–

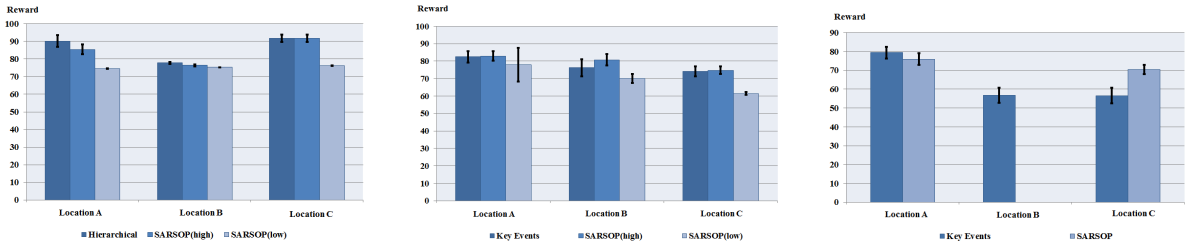
Table 4: Comparison of rewards, intercept and run times on 53×53 map. The maps are too large for SARSOP.

Policy	Steps to Intercept			Actual Reward			Running Time		
	Loc A	Loc B	Loc C	Loc A	Loc B	Loc C	Loc A	Loc B	Loc C
Hierarchical	36.08±1.01	60.75±0.69	48.65±1.10	51.37±0.89	30.73±0.40	40.59±0.76	27.00	25.10	28.22
SARSOP (low)	–	–	–	–	–	–	–	–	–
SARSOP (high)	–	–	–	–	–	–	–	–	–



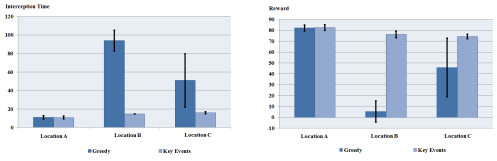
(a) 11×11 (b) 17×17 (c) 23×23

Figure 5: Interception times comparing proposed method (key events) to SARSOP low and SARSOP high precision. Our interception rates are comparable to those of high-precision SARSOP but we can continue to perform well even when maps are too large for SARSOP.



(a) 11×11 (b) 17×17 (c) 23×23

Figure 6: Comparison of reward metrics obtained by key events against SARSOP under both low- and high-precision. The proposed method continues to perform well even when maps are too large for SARSOP.



(a) intercept time (b) reward

Figure 7: Proposed method yields significant improvement over the greedy baseline both in intercept time and reward.

Acknowledgments

This research was supported in part by DARPA award N10AP20027 and NSF award IIS-0845159.

References

Charlin, L.; Poupart, P.; and Shioda, R. 2007. Automated hierarchy discovery for planning in partially observable environments. In *NIPS*.

He, R.; Bachrach, A.; and Roy, N. 2010. Efficient planning under uncertainty for a target-tracking micro-aerial vehicle. In *Proceedings of ICRA*.

He, R.; Brunskill, E.; and Roy, N. 2010. PUMA: Planning under uncertainty with macro-actions. In *Proceedings of AAAI Conference on Artificial Intelligence*.

Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of Robotics: Science and Systems*.

Pineau, J.; Roy, N.; and Thrun, S. 2001. A hierarchical approach to POMDP planning and execution. In *ICML Workshop on Hierarchy and Memory in Reinforcement Learning*.

Tastan, B.; Chang, Y.; and Sukthankar, G. 2012. Learning to intercept opponents in first person shooter games. In *IEEE Conference on Computational Intelligence in Games*.

Theocharous, G., and Kaelbling, L. P. 2003. Approximate planning in POMDPs with macro-actions. In *NIPS*.

Toussaint, M.; Charlin, L.; and Poupart, P. 2008. Hierarchical POMDP controller optimization by likelihood maximization. In *Proceedings of Conference on UAI*.