Comparison of MOMDP and Heuristic Methods to Play Hide-and-Seek

Alex GOLDHOORN [a] Alberto SANFELIU [a] René ALQUÉZAR [a]
[a] *Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain.*

Abstract. The hide-and-seek game is considered an excellent domain for studying the interactions between mobile robots and humans. Prior to the implementation and test in our mobile robots TIBI and DABO, we have been devising different models and strategies to play this game and comparing them extensively in simulations. We propose the use of MOMDP (Mixed Observability Markov Decision Processes) models to learn a good policy to be applied by the seeker. For two players the amount of states is quadratic in the number of discrete map cells. The number of cells were reduced by using a two-level MOMDP, where the policy is computed on-line at the top level with a reduced number of states independent of the grid size. In this paper, we also introduce a new fast heuristic method for the seeker and compare its performance to both off-line and on-line MOMDP approaches. We show simulation results in maps of different sizes against two types of automated hiders.

Keywords. Robotics, Human Robot Interaction, Hide-and-Seek, POMDP

Introduction

Hide-and-seek is an interactive game that has been suggested as an ideal domain for studying cognitive functions in robots and human-robot interaction [1], because the game requires the robot to search, navigate, anticipate on and predict the behaviour of the opponent. In the simple two-players setting, a seeker tries to find and catch a hider, while the latter tries to reach a base without being caught. Players of the game can follow several strategies to win depending on their role. Because of uncertainties in the location of the opponent due to obstacles Partially Observable Markov Decision Processes (POMDPs) [2,3] can be used to model the environment and the location of the agents. POMDPs have been applied in [4,5,6] to solve variants of the hide-and-seek game.

In our hide-and-seek version the seeker and the hider move – at maximum one discrete step – at the same time. The seeker computes a belief (a probabilistic state estimate) and chooses an action to maximize its expected future reward, meanwhile the hider chooses also an action following its own strategy. POMDPs have been successfully applied to various robotic tasks [7,8], but unfortunately, computing an optimal policy exactly is generally intractable (PSPACE-hard, [9]) because the size of the belief space grows exponentially with the number of states.

Instead of using POMDPs, we have recently proposed to use MOMDPs (Mixed Observability Markov Decision Processes) for this game [10] because there are fully observable and not fully observable components of the state (the own and the opponent po-

sition, respectively). The MOMDP uses a factored model to represent separately the fully and partially observable components of a state and derive a compact lower-dimensional representation of its belief space [4,5]. Although MOMDP solvers clearly improve the efficiency of the corresponding ones for POMDPs [4], they are still limited by the number of partially observable states in the problem. As an alternative to off-line MOMDP policy computation with the complete grid fine resolution, we have devised a two-level MOMDP, where the policy is computed on-line at the top level with a reduced number of states independent of the grid size so in principle it may be applied to larger maps.

In this work, we introduce a new fast heuristic method for the seeker and analyze its computational cost. Its effectiveness is compared experimentally to both off-line and on-line MOMDP approaches. We also investigate new variants of the on-line MOMDP model. We show simulation results in maps of different sizes against two types of automated hiders: a random one and a heuristically-driven one.

1. Definition of the Hide-and-Seek Game

In our version of the hide-and-seek there are only two players, a seeker and a hider, who play on a grid of $rows \times cols$ cells. The grid exists of *obstacle* (cells), *free cells* and a special free cell called the *base*. The seeker starts on the base and the hider can start on any free cell. At each time step, both players can take one of nine actions: stay in the same cell or move to a free neighbor cell in an 8-connectivity neighborhood. The seeker wins if it approaches the hider sufficiently and "catches" it (in the simulations we defined it as being in the same cell). The hider wins if it reaches the base before being caught by the seeker. And the result is a *tie* when no player has won within the maximum predefined time H. Both players are supposed to have 360° visibility at each time step, only limited by the obstacles. Hence, the visibility for each player is calculated with a ray-tracing algorithm.

The focus of the work has been from the point of view of the seeker, meaning that we want to apply or learn the best strategy to win the hide-and-seek game. It is also assumed that the seeker's position is fully observable for itself (i.e. no local uncertainty), whereas the hider's position is only known – again without uncertainty – if it is observable. Even though the focus is on the seeker, the same type of heuristic and models could be applied to play the game as a hider.

To quickly and thoroughly test and compare different strategies, models and parameters, two automated hiders have been used: a randomly moving hider and a "smart" (heuristically driven) hider; these will be explained in section 5.

2. Triangle Heuristic for Hide-and-Seek

The previously defined hide-and-seek game has three important distances: between the seeker and hider (d_{sh}), between the seeker and the base (d_{sb}), and between the hider and the base (d_{hb}). With this we have created a heuristic *reward* function:

$$R(s,h,b) = \begin{cases} D - d_{sh}, & \text{if } d_{hb} > d_{sb} \\ -d_{sh}, & \text{otherwise} \end{cases} \tag{1}$$

where s is the seeker, h the hider and b the base. D is a maximum value, in our simulations we defined it as $D = rows \times cols$. The score increases as the distance to the hider decreases (d_{sh}); to protect the base an extra score of D is given when the seeker is closer to the base than the hider. In order to compute these three distances one may use the simple Euclidean distance or the the shortest path length that depends on the map [10]. We have used the last one for our experiments.

With this heuristic an automated *SmartSeeker* has been made. This seeker calculates a score for each action it can take and then chooses the action with the maximum score. At maximum 9 actions are possible (one step or staying at the same position) for both the seeker and hider. One action gets the seeker to a position, which can be used to calculate R; but at the same time the hider can make a move, which we take into account by averaging the score over these moves: $w(s',h,b) = \sum_{h' \in \text{moves}(h)} R(s',h',b)/|\text{moves}(h)|$. When the hider is not visible to the seeker the only thing we know is that the hider is at a not visible position; therefore the score w is calculated for every possible hider's position and then averaged.

The complexity of the calculation of Eq. (1) depends on the complexity of the distance calculation. For the standard Dijkstra algorithm the complexity [11] is $O(|V|\log|V| + |E|)$ where V and E are the vertices and the edges of the tree. For Eq. (1) three distances have to be calculated. If the hider position is known then Eq. (1) has to be calculated at maximum 81 times (9 possible actions of both players). If the hider is not visible then the previous calculation has to be done for each hidden cell; thus the complexity grows linearly with the number of hidden cells. The scores can be calculated a priori and the memory usage will be at maximum N^2 where N is the number of cells.

3. Off-line MOMDP Models for Hide-and-Seek

The hide-and-seek game can be modelled using an MOMDP [4,5], where the state is composed by the grid cell positions of both players. Therefore the number of states is the square of the number of grid cells of the 2D map where the game is going to be played. The number of grid cells depends on the resolution that we want to consider in the game (e.g., a grid cell of 1×1 m^2 in a 2D map of 10×10 m^2 implies 10000 MOMPD states). Formally, the hide-and-seek game is modelled as:

$$\langle \mathscr{X}, \mathscr{Y}, \mathscr{A}, \mathscr{O}_{\mathscr{X}}, \mathscr{O}_{\mathscr{Y}}, T_{\mathscr{X}}, T_{\mathscr{Y}}, Z_{\mathscr{X}}, Z_{\mathscr{Y}}, R, \gamma \rangle \tag{2}$$

where:

- $\mathscr{X}$: the space of all values for fully-observable seeker's state $x = (x_{\text{seeker}}, y_{\text{seeker}})$, the seeker's position;
- $\mathscr{Y}$: the space of all values for partially-observable hider's state $y = (x_{\text{hider}}, y_{\text{hider}})$, the hider's position;
- $\mathscr{A}$: the 9 actions of the seeker: *north, northwest, west, ..., halt*. Each of the actions represents a movement of only one grid cell per time step, except for the action *halt* which represents staying at the same state;
- $\mathscr{O}_{\mathscr{X}}$: $\mathscr{O}_{\mathscr{X}} = \mathscr{X}$, since $o_x = x$ for all states of the seeker;

- $\mathcal{O}_{\mathcal{Y}}$: $\mathcal{O}_{\mathcal{Y}} = \mathcal{Y} \cup \{unknown\}$, which is the union of the set of hider's states and a special observation value *unknown*, which represents the cases when the hider is not visible to the seeker;
- $T_{\mathcal{X}}$: the transition probabilities of the seeker's state given an action, $T_{\mathcal{X}}(x,y,a,x') = p(x'|x,y,a)$. The next position of the seeker is directly given by its current position x, its action a and the map, but independently of the current position of the hider y. Therefore these probabilities will always be 1 or 0, taking into account that the result of an infeasible action is defined as staying on the same cell (e.g going north if a wall is there results in not moving). Also reaching the final state will result in staying in the same state;
- $T_{\mathcal{Y}}$: the transition probabilities of the hider's state given a seeker's action and locations of the seeker and hider, $T_{\mathcal{Y}}(x,y,a,x',y') = p(y'|x,y,a,x')$. These probabilities are not as evident as the previous ones since the own action of the hider is not known. There are two suggested solutions: the first is to spread the probabilities of the movement of the hider uniformly, the second option is to use historical data of human players. Both options are discussed in detail in [10]; and we have used uniform probabilities since using historical data did not give significantly better results, and gathering this information requires a great amount of games played by humans. Also here the probability will be 1 if a final state has been reached;
- $Z_{\mathcal{X}}$: the observation probabilities $Z_{\mathcal{X}}(x',y',a,o_x) = p(o_x|x',y',a)$ will be 1 if $o_x = x'$ and 0 otherwise;
- $Z_{\mathcal{Y}}$: the observation probabilities $Z_{\mathcal{Y}}(x',y',a,o_x,o_y) = p(o_y|x',y',a,o_x)$ depend on the locations of the seeker and hider and the map. The probability will be 1 if $o_y = y'$ and y' is visible from x' or o_y =*unknown* and y' is not visible from x', otherwise it will be 0;
- R: the instantaneous reward for a state, two reward functions were tried: The *simple* reward which has non-zero values only for final states (positive for $x = y$ and negative for y =*base*, $x \neq y$), and the *triangle* reward: using Eq. . (1).
- Finally γ: the discount factor.

While the triangle reward is much more informative than the simple reward, its computational cost is also higher. Note that the simple reward can be computed extremely fast at each step without the need of memorizing its values for each state. On the other hand, for the triangle reward, either its values are precalculated for each state (higher memory cost) or the computation time is increased considerably if calculated at each step.

The initial belief $b_{Y,0}$ is based on the position of the hider. If the hider is visible then the belief of that state is 1.0, otherwise the belief is uniformly distributed over the not visible states. As can be seen the definition of the transition and observation probabilities are simple, but they depend on the amount of cells N. Their complexity is $O(N^3)$ since three variables depend on the map size. Although $T_{\mathcal{Y}}$ has a four state variables the x' is not used by us, neither is o_x in $Z_{\mathcal{Y}}$. The calculation of the reward, if the triangle rule is used, depends on the distance calculation, as explained in the previous section.

The *off-line* usage of the MOMDP model means that the policy is calculated a priori. When playing the game the actions are based directly on the already available policy. The time of calculating a policy off-line took 2 hours for maps of 12×12 up to more than 40 hours, using the CPU described in section 5, for maps of 20×20. Furthermore the time and memory complexity grows with the number of states due to the curse of

history and dimensionality [12]. In order to still handle big maps we suggest an on-line method in which the states are segmented, as explained in the next section.

4. On-line MOMDP Model for the Hide-and-Seek Game

A hierarchical model is described next in which the bottom-level is an MOMDP as described in the previous section, and the top level is an MOMDP with less states. Only for the top level MOMDP a policy is calculated. Top level states are generated by grouping adjacent lower level states, at the same time the probability matrices have to be changed.

4.1. Bottom-level MOMDP

The bottom level MOMDP is defined as in the previous section, Eq. (2). However this model is only used to update the beliefs at full resolution and to calculate the top MOMDP transition and observation probabilities. The belief is initialized as in the off-line version. Before generating the top-level MOMDP, the bottom-level belief is updated with the new observations o_x and o_y, action a, belief state (b_Y, x) and new state x':

$$b'_Y(y') = \frac{p(o_x|x',y',a)p(o_y|x',y',a,o_x) \times \sum_{y \in \mathscr{Y}} p(x'|x,y,a)p(y',x,y,a,x')b_Y(y)}{p(o|b,a)} \quad (3)$$

4.2. Top-level MOMDP

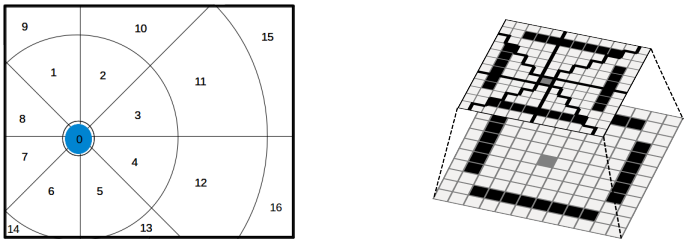
The partially observable states $\mathscr{Y}$ are reduced to $\mathscr{Y}_T$ by grouping spatially adjacent states. Formally, a function $\psi(y_T)$ is defined that gives the set of bottom-level adjacent states which are covered by each value of the top level states $y_T \in \mathscr{Y}_T$. The problem of finding a proper function ψ can be posed as a segmentation based on the map itself, the location of the players, the reward obtained in each state and/or the belief of each state.

We propose a method that centers on the robot location and divides the space in the eight directions and distance, as seen from the robot. Figure 1 shows the *robot centered* segmentation in which the robot is at location 0, and the segmentation is done from that point in the eight directions and based on a fixed distance to the center. Since the hider and base positions are of vital importance for the game, they are added as a separate superstate if known; these superstates will represent only one cell in the bottom level. Since the goal is to catch the hider and to protect the base, the direction to go is the most important; which is represented by this segmentation, taking into account that at each step a new top model is generated. The top-level MOMDP can be defined as follows:

$$\langle \mathscr{X}_T, \mathscr{Y}_T, A, \mathscr{O}_{\mathscr{X},T}, \mathscr{O}_{\mathscr{Y},T}, T_{\mathscr{X},T}, T_{\mathscr{Y},T}, Z_{\mathscr{X},T}, Z_{\mathscr{Y},T}, R_T, \gamma \rangle \quad (4)$$

where A is the same as the bottom MOMDP. The state reduction has been tried on only the partially visible states $\mathscr{Y}$ and on both state variables $\mathscr{X}$ and $\mathscr{Y}$. The advantage of the first case is that it reduces the belief space, but keeps the precision of the fully observable states $\mathscr{X}$. In the latter case it reduces the number of states even more.

The transition and observation probabilities and rewards are averaged from the bottom level. When only the $\mathscr{Y}$ states are segmented, then the new probabilities are:



(a) Robot centered segmentation (b) Layered segmentation

Figure 1. The *robot centered* segmentation (a) centers on the robot's location (0 in the figure) and from there on creates segments based on the direction and distance, based on the bottom layer (b).

$$p(x'|x,y_T,a) = \frac{1}{|\psi(y_T)|} \sum_{y \in \psi(y_T)} p(x'|x,y,a) \tag{5}$$

$$p(y'_T \mid x,y_T,a,x') = \frac{1}{|\psi(y_T)|} \sum_{y' \in \psi(y'_T)} \sum_{y \in \psi(y_T)} p(y'|x,y,a,x') \tag{6}$$

$$p(o_y|x',y'_T,a) = \frac{1}{|\psi(y'_T)|} \sum_{y' \in \psi(y'_T)} p(o_y|x',y',a) \tag{7}$$

When both state variables $\mathscr{X}$ and $\mathscr{Y}$ are segmented with functions ψ_X, ψ_Y and ψ_O respectively then the transition and observation probabilities change slightly:

$$p(x'_T|x_T,y_T,a) = \frac{1}{|\psi_X(x_T)||\psi_Y(y_T)|} \sum_{x' \in \psi_X(x'_T)} \sum_{x \in \psi_X(x_T)} \sum_{y \in \psi_Y(y_T)} p(x'|x,y,a) \tag{8}$$

$$p(y'_T|x_T,y_T,a,x'_T) = \frac{\sum_{y' \in \psi_Y(y'_T)} \sum_{x' \in \psi_X(x'_T)} \sum_{x \in \psi_X(x_T)} \sum_{y \in \psi_Y(y_T)} p(y'|x,y,a,x')}{|\psi_X(x_T)||\psi_Y(y_T)|} \tag{9}$$

$$p(o_{T,Y}|x'_T,y'_T,a) = \frac{\sum_{x' \in \psi_X(x'_T)} \sum_{y' \in \psi_Y(y'_T)} \sum_{o_y \in \psi_O(o_{T,Y})} p(o_y|x',y',a)}{|\psi_X(x'_T)||\psi_Y(y'_T)|} \tag{10}$$

Note that ψ_X and ψ_Y could be different, but we used the same functions if we segmented both state variables. ψ_O is the same as ψ_Y but has the *unknown* value added.

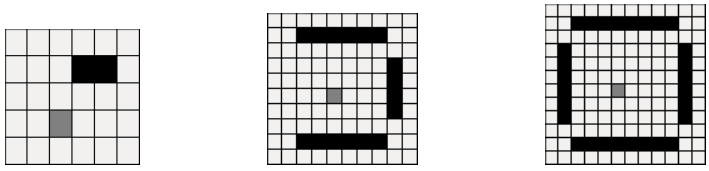
The top reward function $R_T(x_T,y_T,a)$ can be defined as an average of the rewards of the bottom states; however also an explicit reward was tested. The explicit reward is 1 when the seeker is in the hider's superstate ($x \in \psi(y_T)$ or $x \in \psi_X(x_T) \wedge x \in \psi_Y(y_T)$), and -1 if the hider is at the base. To speed up the process of finding a good policy the final state can be defined as staying in the same super state independent of the action a: $p(x_{T,f}|x_{T,f},y_{T,f},a) = 1.0$ and $p(y_{T,f}|x_{T,f},y_{T,f},a,x_{T,f}) = 1.0$ where $(x_{T,f},y_{T,f})$ is a final state. The final state is defined as either $y_{T,f}$ being on the base, or if $\exists x \in \mathscr{X} : x \in \psi_Y(y_{T,f}) \wedge x \in \psi_X(x_{T,f})$, i.e. the seeker is in the same superstate as the hider.

4.3. Summary

For the on-line layered method first an MOMDP model of the lower level has to be defined using Eq. (2), and an initial belief b_0, as described in section 3. The location of the robot is then used to segment the states using the robot centered segmentation (Figure 1), with which the top level states $\mathscr{X}_T$ and/or $\mathscr{Y}_T$ are generated. To generate the rest of the Top MOMDP, Eqs. (5)-(7) are used when only $\mathscr{Y}$ is segmented or Eqs. (8)-(10) when both $\mathscr{X}$ and $\mathscr{Y}$ are segmented. For the top model the initial belief is calculated using the bottom belief: $b_{\mathscr{Y},0,T}(y_T) = \sum_{y \in \psi(y_T)} b_{\mathscr{Y}}(y)$. Next a policy is generated and then the action for the current state is executed. Thereafter an observation is done, which is used to update the belief using Eq. (3). This process is continued until the game finishes.

5. Simulations

Simulations have been done on four manually made maps on three different sizes; Figure 2 shows some of those. Two versions of an automatic hider have been created: a



(a) Map 1: 6×5 (b) Map 2: 10×10 (c) Map 3: 12×12

Figure 2. The maps used in the simulated and real experiments. Black cells are obstacles, the gray cell is the base.

random hider, and a *smart hider*. The first moves completely randomly and the second uses the triangle rule: $R_h(s,h,b) = D - d_{hb} + 0.4d_{sh} + noise$ where $D = \text{rows} \times \text{cols}$, d_{hb} the distance between the hider and the base, and d_{sh} the distance between the seeker and the hider. The noise is 2 at maximum and reduces when the distance is less than 3 cells, because when a hider is either close to the seeker or to the base, it should respectively always flee or go to the base directly. Each action is scored and the action with the maximum score is chosen. Since the seeker can move at the same time, all the possible moves of the seeker are taken into account and R_h is calculated and averaged. When the seeker is not visible, a score is calculated for all the not visible grid cells and averaged.

To generate policies for the MOMDP models we used the *Approximate POMDP Planning Software* (APPL) [1] [4]. The simulations were done on a stand alone PC with 8 GB of RAM and an Intel Core™i5 CPU 760 @ 2.80 GHz with 4 cores and Ubuntu 12.04 as OS. APPL uses SARSOP [6,4], which is a state-of-the-art off-line solver for POMDPs, but can be used on-line by simply alternating a planning and an execution phase [13]. The policy of the off-line MOMDP method is run beforehand with a maximum time of one hour. The smart seeker is implemented such that it buffers the distance results. To prevent the game running endlessly a maximum number of time steps is set relative to the map size: $2(rows + cols)$, since in bigger maps it probably requires more steps to win. Reaching the maximum time without a winner is counted as a tie.

[1] `http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/`

5.1. Results

An overview of the win percentages of the different seekers can be seen in Table 1; in these simulations only maps until a size of 12×12 have been used because of the model's current limitations, and for the on-line MOMDP methods a maximum learning time of 300 s per step was allowed. Comparing the different seeker models the off-line MOMDP model wins the most games, next the heuristic *smart seeker* and finally the on-line MOMDP models ($p < 0.001$; Fisher's exact test, two-sided, this has been used to check all the win percentages). The off-line MOMDP with the triangle reward and the smart seeker won more often against the smart hider, while the off-line method with the simple reward and the on-line methods won more against the random hider ($p < 0.01$). In all the results the smart hider works better, like expected, with an 12% win against a 1% of the random hider ($p < 0.001$).

For the on-line methods several parameters have been tested. First a reduction of the number of states by also segmenting the fully observable space ($\mathcal{X}$), this gave no significant difference in the number of wins. Neither did the optimization of setting the final state and a reward of either 1 in winning states or -1 in losing states in the top level MOMDP, later called *on-line (top rew.)*. However limiting the time to learn the policy to 10 s reduces the win percentage to 69% of the latter, which is significantly less ($p < 0.001$) than the 84% of the standard on-line method. In the robot centered segmentation we - by default - used only the direction (and the location of the seeker, base, and hider if known), which gives at maximum 11 segments. We tested to add a layer more based on the distance (see Figure 1) which increased the number of segments to 19 at maximum. The latter gave marginally significant better results.

Table 1. The simulation statistics for the different models against the different hiders.

Model	Hider	Win	Lose	Tie	Total
off-line (simple)	random	1224 (98.6%)	3 (0.2%)	15 (1.2%)	1242
	smart	1051 (86.6%)	162 (13.4%)	0 (0.0%)	1213
off-line (triangle)	random	1317 (96.8%)	0 (0.0%)	43 (3.2%)	1360
	smart	601 (99.0%)	5 (0.8%)	1 (0.2%)	607
on-line	random	293 (93.9%)	15 (4.8%)	4 (1.3%)	312
	smart	109 (63.4%)	58 (33.7%)	5 (2.9%)	172
on-line (top rew.)	random	294 (96.7%)	3 (1.0%)	7 (2.3%)	304
	smart	124 (56.9%)	88 (40.4%)	6 (2.8%)	218
smart seeker	random	758 (89.5%)	0 (0.0%)	89 (10.5%)	847
	smart	455 (95.0%)	0 (0.0%)	24 (5.0%)	479
TOTAL		6226 (92.2%)	334 (4.9%)	194 (2.9%)	6754

The map size does influence the results, the bigger the map the lower the win percentage as can be seen in Table 2. Furthermore for bigger maps the on-line MOMDP methods need more time. Since the on-line MOMDP methods learn a policy on every time step, they take more time the bigger the map. The on-line MOMDP method with top rewards takes significantly longer on the 6×5 and 10×10 maps, but on the 12×12 maps the standard on-line method takes more time ($p < 0.05$, Wilcoxon ranksum test, 2-sided; see Table 2). Only against the *smart seeker* games on 40×40 maps were tested,

because for the other methods the number of states was too big to be able to calculate a policy.

Segmenting also the $\mathscr{X}$ states seems to take more time, but this is not significant. Using the on-line MOMDP with rewards redefined at the top takes less time than the normal MOMDP with rewards averaged from the bottom level, this is marginally significant. Using the extra layer of the robot centered segmentation method takes significantly more time than only segmenting based on the direction.

The off-line MOMDP and the smart seeker take less time than the on-line MOMDP models ($p < 0.001$,Wilcoxon ranksum). Nonetheless we should take into account that the off-line MOMDP method requires to learn the policy beforehand; this took from 1.4 s on average for the 6×5 maps to 1 hour (the set maximum) for bigger maps. The off-line MOMDP method wins the games in statistically less steps than any of the other methods ($p < 0.001$,Wilcoxon ranksum; see Table 2). And using the simple reward results in winning in less steps than using the triangle reward ($p < 0.001$).

Table 2. The win statistics per map size and seeker type played against both hiders. The last columns show the average $\pm$ standard deviation of the number of actions and the duration per action for won games.

Map Size	Model	Win	Lose	Tie	Total	Numb. actions	Duration/act.(s)
6×5	off-line	97.0%	1.2%	1.8%	2249	5.49 ± 4.21	0.15 ± 0.13
	on-line	93.3%	6.7%	0.0%	268	5.06 ± 3.49	0.69 ± 0.62
	on-line (t.r.)	93.0%	6.6%	0.4%	242	5.71 ± 4.22	0.88 ± 1.36
	smart seeker	91.7%	0.0%	8.3%	360	7.59 ± 4.99	0.13 ± 0.09
10×10	off-line	96.6%	2.1%	1.4%	1407	11.48 ± 7.11	0.11 ± 0.08
	on-line	78.9%	17.6%	3.5%	142	14.67 ± 7.57	14.73 ± 17.69
	on-line (t.r.)	81.4%	15.3%	3.4%	118	13.49 ± 7.84	64.63 ± 69.38
	smart seeker	91.6%	0.0%	8.4%	856	13.97 ± 9.22	0.12 ± 0.1
12×12	off-line	85.2%	14.8%	0.0%	766	9.87 ± 6.0	0.11 ± 0.1
	on-line	54.1%	40.5%	5.4%	74	14.43 ± 12.41	92.83 ± 63.01
	on-line (t.r.)	59.9%	35.2%	4.9%	162	15.06 ± 11.26	70.53 ± 61.63
	smart seeker	92.0%	0.0%	8.0%	100	15.68 ± 12.11	0.1 ± 0.07
40×40	smart seeker	55.4%	0.2%	44.4%	448	44.59 ± 24.94	0.1 ± 0.15

6. Conclusions

This work has presented several methods to play the hide-and-seek game as an automated seeker. A heuristic based player, and an MOMDP model to learn a policy. We have used an off-line version with two reward types, one based on the heuristic, and another based on a reward only in the final states. Because the MOMDP requires exponential time complexity on the square of the number of states, we have proposed a two level hierarchical structure for which the MOMDP policy is calculated at the top level, where the number of states has been reduced drastically by using a robot centered segmentation, a segmentation process that fixes the maximum number of states that we consider at that level.

The methods have been tested in simulation against two automated hiders: a random and an heuristically driven seeker. It has shown that the MOMDP models all work very well against the random hider winning 94% or more, whereas the smart seeker and the off-line MOMDP model with the triangle reward work best against the smart hider, winning 95% or more. Overall the off-line MOMDP and the smart seeker have the best performance in running time and winning, with the off-line MOMDP model winning in less steps. The MOMDP model and the smart seeker are comparable, however the smart seeker is greedy and does not have memory. A problem of the off-line method is that the complexity of finding a policy is exponential with the states. Our suggested on-line methods solve this by reducing the model to a fixed low number of states, and these work very well against the random hider, but worse against the smart hider.

The goal in the future is to learn strategies that do not depend on the size of the map. Furthermore, experiments with real world robots will be done to test the human robot interaction aspects.

Acknowledgements

Work supported by the Spanish Ministry of Science and Innovation under project RobTaskCoop (DPI2010-17112).

References

[1] E. Johansson and C. Balkenius, "It's a child's game: Investigating cognitive development with playing robots," in *International Conference on Development and Learning*, 2005, p. 0:164.

[2] D. Braziunas, "Pomdp solution methods," University of Toronto, Tech. Rep., 2003.

[3] M. Hauskrecht, "Value-function approximations for partially observable markov decision processes," *Journal of Artificial Intelligence Research*, vol. 13, pp. 33–94, 2000.

[4] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee, "Planning under Uncertainty for Robotic Tasks with Mixed Observability," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1053–1068, May 2010.

[5] M. Araya-López, V. Thomas, O. Buffet, and F. Charpillet, "A closer look at MOMDPs," in *22nd International Conference on Tools with Artificial Intelligence - ICTAI*, 2010.

[6] H. Kurniawati, D. Hsu, and W. Lee, "Sarsop: efficient point-based pomdp planning by approximating optimally reachable belief spaces," in *Robotics: Science and Systems, 2008*, 2008.

[7] A. Cassandra, L. Kaelbling, and J. Kurien, "Acting under uncertainty: discrete bayesian models for mobile-robot navigation," in *Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, vol. 2, nov 1996, pp. 963–972 vol.2.

[8] M. Spaan and N. Vlassis, "A point-based pomdp algorithm for robot planning," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 3, 2004, pp. 2399 – 2404 Vol.3.

[9] C. Papadimitriou and J. Tsisiklis, "The complexity of markov decision processes," *Mathematics of Operations Research*, vol. 12, no. 3, pp. 441–450, 1987.

[10] C. Georgaraki, "A POMDP approach to the hide and seek game," Master's thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 2012.

[11] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[12] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for pomdps," in *International Joint Conference on Artificial Intelligence, 2003*, 2003, pp. 477–484.

[13] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online Planning Algorithms for POMDPs." *The journal of artificial intelligence research*, vol. 32, no. 2, pp. 663–704, July 2008.