

Game Theoretic Control for Robot Teams*

Rosemary Emery-Montemerlo, Geoff Gordon and Jeff Schneider

*School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15312
{remery,gordon,schneide}@cs.cmu.edu*

Sebastian Thrun

*Stanford AI Lab
Stanford University
Stanford, CA 94305
thrun@stanford.edu*

Abstract—In the real world, noisy sensors and limited communication make it difficult for robot teams to coordinate in tightly coupled tasks. Team members cannot simply apply single-robot solution techniques for partially observable problems in parallel because they do not take into account the recursive effect that reasoning about the beliefs of others has on policy generation. Instead, we must turn to a game theoretic approach to model the problem correctly. Partially observable stochastic games (POSGs) provide a solution model for decentralized robot teams, however, this model quickly becomes intractable. In previous work we presented an algorithm for lookahead search in POSGs. Here we present an extension which reduces computation during lookahead by clustering similar observation histories together. We show that by clustering histories which have similar profiles of predicted reward, we can greatly reduce the computation time required to solve a POSG while maintaining a good approximation to the optimal policy. We demonstrate the power of the clustering algorithm in a real-time robot controller as well as for a simple benchmark problem.

Index Terms—Planning, Multi-Robot Coordination, Decentralized control, Partially Observable Domains.

I. INTRODUCTION

Robotic teams and distributed multi-robot systems provide numerous advantages over single-robot systems. For example, distributed mapping and exploration allows an area to be covered more efficiently than by a single robot and it is tolerant to individual robot failures. When designing a multi-robot team, however, a key question is how to assign tasks to individual robots and best coordinate their behaviours. In loosely coupled domains, much success in task allocation has been achieved with both auction based approaches ([1], [2]) and behaviour based robotics ([3], [4]). For more tightly coupled domains in which robots must coordinate on specific actions in order to achieve a goal, probabilistic frameworks provide a model for optimal control.

The frameworks of Markov decision processes (MDPs) and partially observable Markov decision processes (POMDPs) are very powerful models of probabilistic domains and have been applied successfully to many single-robot planning problems ([5], [6]). These algorithms can also be extended to the multi-robot case to be used for coordinated planning. The most straightforward of these extensions simply treats the entire robot team as a single

robot with multiple actuators and plans in joint state and action space. This is not necessarily an efficient use of resources: the planner must run either on a central machine or simultaneously on each robot, with all sensor information from each robot being sent in real time to every other copy of the planner. Without full communication the robots cannot coordinate their beliefs about the current joint state, and may select conflicting actions to implement.

In the real world, communication bandwidth is limited, and in some domains not available at all. For example, robots that function underground or on the surface of another planet are limited to point-to-point communication. Even in cases where communication is available, such as indoor mobile robots, limited bandwidth can lead to latency in communication between the robots, and critical information may not be received in time to make decisions.

Instead, robots in teams should be able to independently reason about appropriate actions to take in a decentralized fashion that takes into account both their individual experiences in the world and beliefs about the possible experiences of their teammates. In the case where no communication is available to the team, this type of reasoning is necessary for the entire duration of the problem. If communication is available but with latency, this reasoning allows robots to generate optimal policies between synchronization episodes [7].

The partially observable stochastic game (POSG) is a game theoretic approach to multi-robot decision making that implicitly models a distribution over other robots' observations about the world. It is, however, intractable to solve large POSGs. In [8] we proposed that a viable alternative is for robots to interleave planning and execution by building and solving a smaller approximate game at every time step to generate actions. In this paper we improve the efficiency of our algorithm through the clustering of observation histories. This clustering decreases the computational requirements of the algorithm, which allows us to apply it to larger and more realistic robot applications. We only consider the harder case of coordinating robots that have no access to communication for the duration of the task. The addition of communication can only improve the overall performance of the team and reduce the computational complexity of the problem.

A. Related Work

There are several frameworks that have been proposed to generalize POMDPs to distributed multi-robot

*R. Emery-Montemerlo is supported in part by a Natural Sciences and Engineering Research Council of Canada postgraduate scholarship, and this research has been sponsored by DARPA's MICA program.

systems. DEC-POMDP [9], a model of decentralized partially observable Markov decision processes, MTDP [10], a Markov team decision problem, and I-POMDP [11] are all examples of these frameworks. They formalize the requirements of an optimal policy; however, as shown by Bernstein et al., solving decentralized POMDPs is NEXP-complete [9]. Hansen et al. [12] have developed an exact dynamic programming algorithm for POSGs that is able to handle larger finite-horizon problems than other exact methods but is still limited to relatively small problems.

These results suggest that locally optimal policies that are computationally efficient to generate are essential to the success of applying the POSG framework to the real world. In addition to our own algorithm for finding approximate solutions to POSGs [8], other algorithms that attempt to find locally optimal solutions include POIPSG [13], which finds locally optimal policies from a limited set of policies, and Xuan et al.'s algorithm for solving decentralized POMDPs [14]. In [14] each robot receives only local information about its position and robots only ever have complementary observations. The issue in this system then becomes the determination of when global information is necessary to make progress toward the goal, rather than how to resolve conflicts in beliefs or how to augment one's own belief about the global state. Nair et al. have looked at both a dynamic programming style algorithm for finding locally optimal policies to the full POSG [15] and the computational savings gained by enforcing periodic synchronizations of observations in between episodes of game theoretic reasoning about the beliefs of others [7].

The work in this paper goes a step beyond these algorithms in that it shows a method for limiting the number of observations histories for which policies must be created while still maintaining a good approximation of the true distribution over histories. We also demonstrate how this clustering allows for our algorithm to be used as a real-time robot controller in relatively large problems.

II. BASIC ALGORITHM FOR FINDING APPROXIMATE SOLUTIONS TO POSGS

Stochastic games, a generalization of both repeated games and MDPs, provide a framework for decentralized action selection [16]. POSGs are the extension of this framework to handle uncertainty in world state. A POSG is defined as a tuple (I, S, A, Z, T, R, O) . $I = \{1, \dots, n\}$ is the set of robots, S is the set of states and A and Z are respectively the cross-product of the action and observation space for each robot, i.e. $A = A_1 \times \dots \times A_n$. T is the transition function, $T : S \times A \rightarrow S$, R is the reward function, $R : S \times A \rightarrow \mathfrak{R}$ and O defines the observation emission probabilities, $O : S \times A \times Z \rightarrow [0, 1]$. At each time step of a POSG the robots simultaneously choose actions and receive a reward and observation. In this paper, we limit ourselves to finite POSGs with common payoffs (each robot has an identical reward function R).

Think of the POSG as a large tree. The robots are assigned starting states and, as they progress through the tree, different observations occur and different actions can be taken. Solving the problem requires finding the best

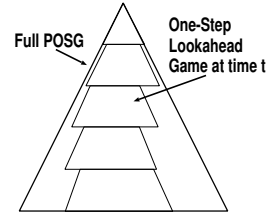


Fig. 1. A high level representation of our algorithm for approximating POSGs. Please refer to [8] for full details.

policy through this tree: an optimal set of actions for each possible history of observations and states. Unlike a game of chess or backgammon, however, the robots do not have full observability of the world and each other's actions. Instead, different parts of the tree will appear similar to different robots. A robot's policy must be the same for all of the nodes of the tree between which it cannot distinguish.

In [8] we proposed an algorithm for finding an approximate solution to a POSG with common payoffs. Our algorithm transforms the original problem into a sequence of smaller Bayesian games that are computationally tractable. Bayesian games model single-state problems in which each robot has private information about something relevant to the decision making process [16]. In our algorithm, this private information is the specific history of observations and actions taken by each robot.

Our algorithm uses information common to the team (e.g. problem dynamics) to approximate the policy of a POSG as a concatenation of a series of policies for smaller, related Bayesian games (Fig. 1). In turn, each of these Bayesian games uses heuristic functions to evaluate the utility of future states in order to keep policy computation tractable. Theoretically, this approach allows us to handle finite horizon problems of indefinite length by interleaving planning and execution.

This transformation is similar to the classic one-step lookahead strategy for fully-observable games [17]: the robots perform full, game-theoretic reasoning about their current knowledge and first action choice but use a heuristic function to evaluate the quality of the resulting outcomes. The resulting policies are always coordinated; and, if the heuristic function is fairly accurate, perform well in practice. A similar approach is used to find near-optimal solutions for a scaled down version of Texas Hold'Em [18].

The key part of our algorithm is the transformation from a time slice of the full POSG to a Bayesian game. If the private information in the Bayesian game is the true history of a robot, then the solution to the Bayesian game is a set of policies that defines an action for each robot r to take for each of its possible histories h_i^r .¹ The action taken for history h_i^r should be the action that maximizes expected reward, given the robot's knowledge (from h_i^r) of the previous observations and actions of its teammates, and given the robot's inferences about the actions which its teammates will take at the current step. In order to properly coordinate on actions, i.e., to compute the same

¹In game theory a solution to a game is a set of policies for each robot that are all best responses to each other. Therefore, if one robot solves a game it also knows the policies of all of the other robots. For more information on Nash equilibria and best response strategies see [16].

set of policies for all robots, each robot must maintain the same distribution over joint histories. Our algorithm enforces this in two ways. First, it synchronizes the random number generators of the robots in the team in order to ensure that any randomization in policy generation occurs in the same way for each robot, and second, it uses only common knowledge (including the generated policies from the current time step) to propagate forward the set of joint histories and its corresponding probability distribution from one time step to the next.

III. IMPROVING ALGORITHM EFFICIENCY

The benefit of using our Bayesian game approximation over the full POSG for robot controllers is that it calculates a good approximation of the optimal policy in tractable time. The overall desirability of our approximation is therefore closely tied to how fast it can construct these policies. The required computation time, however, is directly related to the number of individual and joint histories that are maintained. Limiting the algorithm to use only a representative fraction of the total number of histories is key to applying this approach to real robot problems.

In this section we will compare different ways of limiting the number of histories maintained by the algorithm. For this purpose, we will use a problem (the *Lady and the Tiger*) which is small enough that we can maintain all possible joint histories, in order to compare each algorithm's results to the best possible performance for the domain. In contrast, for the *Robot Tag* problem presented later in this paper, it is not possible to compute all or even a large fraction of the possible joint histories due to both memory and computation limitations.

The *Lady and The Tiger* problem is a multi-robot version of the classic tiger problem [19] created by Nair et al. [15]. In this two-state, finite-horizon problem, two robots are faced with the problem of opening one of two doors, one of which has a tiger behind it and the other a treasure. Whenever a door is opened, the state is reset and the game continues until the fixed time has passed. The crux of the *Lady and the Tiger* is that neither robot can see the actions or observations made by their teammate, nor can they observe the reward signal and thereby deduce them. It is, however, necessary for the robots to reason about this information in order to coordinate with each other.

A. Low Probability Pruning

In the original implementation of our Bayesian game approximation, the number of histories that we maintained per robot was kept manageable by pruning out low probability joint histories. If the probability of a joint history was below some threshold, it was cut and the probability of the remaining joint histories was renormalized. The allowable individual histories for each robot were then taken to be those that existed in the set of remaining joint histories.

At run time, a robot's true individual history may be one of the ones which we have pruned. In this case, we find the best match to the true history among the histories which we retained, and act as if we had seen the best match instead. There are many possible ways to define the best match, but a simple one which worked well in our experiments was to

look at *reward profiles*: the reward profile for an individual history h is the vector r^h whose elements are

$$r_a^h = E(R(s, a) | h) \quad (1)$$

That is, r_a^h is the robot's belief about how much reward it will receive for performing action a given its belief (based on h) about the state of the world.

Given the reward profiles for the true history h and some candidate match h' , we can compare them using their *worst-case reward difference*

$$\max_{a \in A} |r_a^h - r_a^{h'}|$$

and select the match which has the smallest worst-case reward difference from the true history.

B. Clustering

Low-probability pruning can be effective in small domains like the *Lady and the Tiger* where relatively few joint histories have to be pruned. However, it is a crude approximation to pretend that a history's importance for planning is proportional to its probability: when there are a large number of similar histories that each have a low probability of occurring but together represent a substantial fraction of the total probability mass, low-probability pruning can remove all of them and cause a significant change in the predicted outcome of the game.

Instead, we need to make sure that each history we prune can be adequately represented by some other similar history which we retain. That is, we need to cluster histories into groups which have similar predicted outcomes, and retain a representative history from each group. If done properly, the probability distribution over these clusters should be an accurate representation of the true distribution over the original histories. As our experimental results will show, making sure to retain a representative history from each cluster can yield a substantial improvement in the reward achieved by our team. In fact, we can see an improvement even in runs where we are lucky enough not to have pruned the robots' true histories: our performance depends not just on what we have seen but how well we reason about what we might have seen.

At each time step t , we start from the representative joint histories at time $t - 1$ and construct all one-step extensions by appending each possible joint action and observation. From these joint histories we then identify all possible individual histories for each robot and cluster them, using the algorithms described below to find the representative individual histories for step t . The representative joint histories are then the cross product of all of the representative individual histories. (Unlike low probability pruning, we prune individual histories rather than joint histories; a joint history is pruned if any of its individual histories are pruned. This way each robot can determine, without communication, to which cluster its current history belongs.) Finally we construct a reduced Bayesian game using only the representative histories, and solve it to find a policy for each robot which assigns an action to each of its representative individual histories. As before, at runtime we use worst-case reward difference to match the true history to its closest retained history and select an action.

There are many different types of clustering algorithms [20], but in this paper we use a type of agglomerative clustering. In our algorithms, each history starts off in its own cluster and then similar clusters are merged together until a stopping criterion is met. Similarity between clusters is determined by comparing the reward profiles either of representative elements from each cluster or of the clusters as a whole. The reward profile for a cluster c is defined by a straightforward generalization of equation 1: it is

$$r_a^c = E(r_a^h | h \in c) = E(R(s, a) | c)$$

We could use worst-case reward difference as defined above to compare reward profiles, but since we have extra information not available at runtime (namely the prior probability of both clusters) we will use *worst-case expected loss* instead. Worst-case expected loss between two histories h and h' is defined to be

$$\max_{a \in A} [P(h)|r_a^h - r_a^c| + P(h')|r_a^{h'} - r_a^c|] / P(c)$$

where c is the cluster that would result if h and h' were merged. Worst-case expected loss between two clusters is defined analogously in terms of $r_a^{c \cup c'}$. Worst-case expected loss is a good measure of the similarity of the reward profiles for two clusters. It is important to determine if two clusters have similar reward profiles because robots will end up performing the same action for each element of the cluster.

We looked at two different algorithms which differ in how they select pairs of clusters to merge and how they decide when to stop. These algorithms are called low probability clustering and minimum distance clustering. Low-probability clustering was designed to be similar to the low-probability pruning algorithm described above, while minimum-distance clustering is designed to produce a better clustering at a higher computational cost.

1) *Low Probability Clustering*: In this approach, we initially order the single-history clusters at random. Then, we make a single pass through the list of clusters. For each cluster, we test whether its probability is below a threshold; if so, we remove it from the list and merge it with its nearest remaining neighbor as determined by the worst-case expected loss between their representative histories. The neighbor's representative history stays the same, and its probability is incremented by the probability of the cluster being merged in.

This clustering method relies on randomization and so the synchronization of the random number generators of the robots is crucial to ensure that each robot performs the same series of clustering operations.

Low-probability clustering has some similarities to low-probability pruning: we cluster histories that are less likely to occur and therefore a robot's true history is much more likely to match one of the representative histories at execution time. Like low-probability pruning, it has the drawback that all low-probability histories must be matched to another history even if they are far away from all available candidates. Unlike pruning, however, it maintains a distribution over joint histories that more closely matches the original distribution. By randomly ordering the

candidate clusters it is also possible that individual histories which on their own have too low probability to keep will have other low probability histories matched with them and so survive to the final set of histories.

2) *Minimum Distance Clustering*: In minimum-distance clustering we repeatedly find the most similar pair of clusters (according to worst-case expected loss between the clusters' reward profiles) and merge them. We stop when the closest pair of clusters is too far apart, or when we reach a minimum number of clusters.

The representative element for a cluster is simply the highest-probability individual history in that cluster. This representative element is only used for generating the joint histories at the next time step; cluster merging and matching of the observed history to a cluster are both governed by the cluster's reward profile.

C. Benchmark Results

The *Lady and The Tiger* problem is simple yet still large enough to present a challenge to exact and approximate methods for solving POSGs. Table I shows the performance of the basic Bayesian game approximation of the full POSG for a 10-step horizon and compares it to using pruning or clustering to keep the number of histories low.²

In the case of low probability pruning, the performance results in Table I for thresholds of 0.000001 and 0.000005 show that even for cases in which the actual observations of the robots always match exactly to one of the retained histories, removing too many joint histories from consideration negatively impacts performance. If low probability clustering is applied instead, we can maintain many fewer histories without compromising performance. If the probability threshold is set too high, however, performance starts to degrade. The distribution over joint histories is no longer being approximated well because histories that are too different are being aliased together. Minimum distance clustering, however, is able to avoid this problem at higher levels of clustering and lets the system cluster more histories together while still representing the joint history space. An additional advantage of minimum distance clustering is that its parameter is maximum allowable expected loss, which is easier to interpret than the probability threshold used in low probability clustering.

A drawback of the minimum clustering approach over low probability clustering is that it is a more expensive algorithm because it iterates over the set of possible clusters until the termination criteria is met. In contrast, low probability clustering performs only a single pass through the set of possible clusters. However, the minimum distance clustering approach is able to maintain a fewer number of overall histories with no loss in performance.

IV. ROBOT TAG

In this section we present a more realistic version of the *Robotic Tag 2* problem discussed in [8]. In this problem two robots are trying to coordinate on finding and tagging an opponent that is moving with Brownian motion within

²Unlike the results in [8], a simpler utility function was used as our focus is the relationship between performance and retained histories.

TABLE I

RESULTS FOR 10-STEP *Lady and The Tiger* AVERAGED OVER 10000 TRIALS WITH 95% CONFIDENCE INTERVALS. TOTAL NUMBER OF JOINT HISTORIES IS THE SUM OF ALL THE JOINT HISTORIES MAINTAINED OVER THE 10 STEPS. POLICY COMPUTATION TIME IS THE PERCENTAGE OF TIME TAKEN RELATIVE TO THE FIRST CONDITION IN WHICH ALL HISTORIES ARE MAINTAINED (A RUNNING TIME OF 199018ms). THE PERCENT TRUE HISTORY RETAINED IS THE PERCENTAGE OF STEPS IN WHICH THE TRUE ROBOT HISTORY IS PRESENT IN THE SET OF RETAINED HISTORIES. PERFORMANCE IS NOT DEPENDENT ON THIS VALUE BUT RATHER ON HOW WELL THE ENTIRE SET OF POSSIBLE HISTORIES IS APPROXIMATED.

Condition	Average Reward	Total # Joint Histories	Computation Time	% True History Retained
All possible joint histories maintained	10.68 \pm 0.10	231205	100%	100%
Low probability pruning with cutoff of 0.000001	10.66 \pm 0.10	74893	56.28%	100%
Low probability pruning with cutoff of 0.000005	9.89 \pm 0.13	34600	51.98%	100%
Low probability pruning with cutoff of 0.001	5.32 \pm 0.26	1093	1.54%	93.8%
Low probability clustering with threshold 0.01	10.69 \pm 0.09	7715	4.81%	91.3%
Low probability clustering with threshold 0.05	10.68 \pm 0.10	563	0.25%	71.2%
Low probability clustering with threshold 0.1	6.38 \pm 0.21	201	< 0.10%	63.3%
Minimum distance clustering with max. loss 0.01	10.58 \pm 0.10	335	< 0.10%	56.4%
Minimum distance clustering with max. loss 0.1	10.72 \pm 0.09	177	< 0.10%	56.1%
Minimum distance clustering with max. loss 0.5	10.69 \pm 0.10	109	< 0.10%	55.1%

a portion of Stanford University’s Gates Hall. It requires only one robot to tag the opponent, but the robots must coordinate to search the environment efficiently. We first solve the problem in a grid-world version of Gates Hall by converting the environment into a series of cells which are roughly $1.0m \times 3.5m$. Then we execute the resulting policy on real robots by using lower-level behaviors to move between cells while avoiding obstacles.

In the original *Robotic Tag 2* problem, the state of the robots and their observations are highly abstracted. Each robot’s state consists of its current cell and it can only observe the opponent if they are in the same cell. Actions and observations are noise free and each robot knows its teammate’s position exactly. This problem has a branching factor of at most 3: that is, the number of possible joint types at time $t + 1$ is at most 3 times that at time t .

In this paper, we introduce the more-realistic *Robot Tag A* and *Robot Tag B* problems. In these two variants, we model the fact that the robots can see some distance forward but not at all backward; this change results in a larger state space and a much larger observation space. The robot’s state now includes both its cell location and a discrete heading. The heading can face in either direction along a corridor or in all cardinal directions at intersections; this additional state variable increases the size of the problem from the 18252 states of *Robotic Tag 2* to 93600 states. Each robot can sense its current cell location and the cell immediately in front of it. This corresponds to robots that have a sensing range of about $5m$. Motion actions can now fail to execute with a known probability.

In the first variation of this problem, *Robot Tag A*, observations are noisy. The robots still know the state of their teammate, but they may not sense the opponent even if it is within their visible range. Positive detections of the opponent, however, are always correct. In *Robot Tag A*, the branching factor for joint histories can be up to 7, more than twice what it was in *Robotic Tag 2*.

Robot Tag B looks at the effects of not knowing the location of one’s teammate. The robots’ sensors are not noisy, but the robots do not know the location of their teammate unless they sense it. This results in a branching factor of up to 20, making *Robot Tag B* substantially large.

A. Grid-World Results

A fully observable policy was calculated for a robot team that could always observe the opponent and each other using dynamic programming for an infinite horizon version of the problem with a discount factor of 0.95. The Q-values generated for this policy were used as a Q_{MDP} heuristic for calculating utility in the Bayesian game approximation.

For both variants, performance statistics were found for two common heuristics for partially observable problems as well as for our Bayesian game approximation. In these two heuristics each robot maintains an individual belief state based only upon its own observations (as there is no communication) and uses that belief state to make action selections. In the Most Likely State (MLS) heuristic, the robot selects the most likely opponent state and then applies its half of the best joint action for that state as given by the fully observable policy. In the Q_{MDP} heuristic, the robot implements its half of the joint action that maximizes its expected reward given its current belief state. In the case where the teammate is not visible, each heuristic assumes that its teammate did execute the other half of the selected joint action and updates its believed position of its teammate using this action. Positive observations of the teammate are also used to update this believed position.

1) *Robot Tag A*: For this variation of the problem we applied all three approaches to maintaining a reasonable number of individual histories as described in Section III. Table II shows a comparison of these approaches with respect to performance, the total number of histories maintained for each robot and the average number of true observation histories that were not maintained in the set of tracked histories. These results were gathered using random starting positions for both robots and the opponent. Our Bayesian game approximation allowed the robots to find the opponent about 2 to 3 time steps faster on average. Minimum distance clustering resulted in the most computationally efficient robot controllers.

2) *Robot Tag B*: This variation of the problem is well tuned to the use of minimum distance clustering. The large number of possible joint histories resulted in memory problems when a very low threshold was used for low probability pruning or clustering, while raising the threshold caused poor performance due to over aliasing of histories.

TABLE II

Robot Tag RESULTS WITH 95% CONFIDENCE INTERVALS. RESULTS ARE AVERAGED OVER 10000 TRIALS EXCEPT FOR MINIMUM DISTANCE CLUSTERING IN *Robot Tag B* WHICH IS OVER 100 TRIALS. AVERAGE ITERATIONS IS THE NUMBER OF STEPS IT TAKES TO CAPTURE THE OPPONENT. TOTAL NUMBER OF HISTORIES PER ROBOT IS THE AVERAGE NUMBER OF HISTORIES MAINTAINED PER ROBOT, AND NUMBER OF TRUE HISTORIES NOT RETAINED IS THE AVERAGE NUMBER OF TIMES A ROBOT'S TRUE HISTORY WAS NOT IN THE MAINTAINED SET.

Algorithm	Average Reward	Average Iterations	Total # Histories Per Robot	# True Histories Not Retained
Robot Tag A: Know Teammate Position, Noisy Observations, Random Starting Positions				
MLS Heuristic	-7.99 ± 0.21	15.91 ± 0.25	n.a.	n.a.
Q_{MDP} Heuristic	-7.02 ± 0.21	15.44 ± 0.29	n.a.	n.a.
Low Prob. Pruning with cutoff 0.00025	-6.02 ± 0.18	13.10 ± 0.21	166.73 ± 4.17	0.31 ± 0.05
Low Prob. Clustering with threshold 0.0025	-6.06 ± 0.19	13.24 ± 0.22	118.68 ± 2.71	0.32 ± 0.03
Min.Dist.Clustering with max. loss 0.05	-5.99 ± 0.18	13.10 ± 0.21	50.45 ± 0.87	1.54 ± 0.08
Robot Tag B: Do Not Know Teammate Position, Noise-free Observations, Fixed Starting Positions A				
MLS Heuristic	-22.12 ± 0.18	30.63 ± 0.17	n.a.	n.a.
Q_{MDP} Heuristic	-9.91 ± 0.15	16.38 ± 0.25	n.a.	n.a.
Min.Dist.Clustering with max. loss 0.01	-7.95 ± 1.45	13.41 ± 1.94	434.22 ± 204.62	6.88 ± 3.53
Robot Tag B: Do Not Know Teammate Position, Noise-free Observations, Fixed Starting Positions B				
MLS Heuristic	-12.57 ± 0.20	15.89 ± 0.17	n.a.	n.a.
Q_{MDP} Heuristic	-14.71 ± 0.14	21.58 ± 0.18	n.a.	n.a.
Min.Dist.Clustering with max. loss 0.01	-10.07 ± 0.90	14.7 ± 0.97	318.75 ± 26.34	12.75 ± 3.54

Minimum distance clustering, however, was successful in solving the problem as can be seen in Table II. It generated policies that were able to be implemented in real-time. These results were generated using two different fixed starting locations, A and B, for the robots and the opponent. In the first scenario (A) the two robots start at the base of the middle corridor (see Fig. 4), while the opponent starts in the far right side of the loop. In the second scenario (B) the two robots start at the far left of the top corridor and the opponent starts at the far left of the bottom corridor. The first starting condition proved problematic for MLS and the second for Q_{MDP} yet our Bayesian game approximation was able to perform well in both conditions. The average number of total individual histories retained for each robot is much greater than that for *Robot Tag A*, yet the number of times the robot's true history is not in the retained set of histories is larger which is a reflection of the greater branching factor of this problem.

B. Robot Results

The grid-world versions of *Robot Tag A* and *Robot Tag B* were mapped to the real Gates Hall and Pioneer-class robots (Fig. 2) using the CARMEN software package for low-level control [21]. In these mappings, observations and actions remain discrete and there is an additional layer that converts between the continuous world and the grid-world. Localized positions of the robots are used to calculate their specific grid-cell positions and headings. Robots navigate by converting the actions selected by the Bayesian game approximation into goal locations. CARMEN allows us to use the same interface run our algorithm in both a high fidelity simulator as well as on the real robots.

Simulation and real robot runs were conducted both with and without an opponent. If no opponent is present then the robots continue to search the environment until stopped by a human operator. This gives the observer a better idea of how well the robots coordinate on their coverage of the environment. We implemented the MLS and Q_{MDP} heuristics and the minimum distance clustering version of our Bayesian game approximation for both variations of *Robot Tag*. This version of our controller was chosen because it is the most computationally efficient



Fig. 2. One of the robots used for experiments.

and therefore most appropriate for real-time robot control. Similar paths were generated in the simulator and by the physical robots for the same initial conditions.

For the *Robot Tag A* trials the two robots were started in a similar location. With this starting configuration, the MLS heuristic resulted in the two robots following each other around the environment and therefore not taking advantage of the distributed nature of the problem. This is because with MLS, if the two robots have similar observations, then they will believe the opponent to be in the same location and so will both move there. In the Q_{MDP} heuristic, however, similar sensor readings can still lead to different actions being selected because Q_{MDP} selects actions that maximize reward with respect to the current belief of where the opponent is, not just the most likely state. If the belief of the opponent's position is bimodal, a joint action will be selected that sends one robot to each of the possible locations. However, if robots have different beliefs about the opponent location, they can start to duplicate their efforts by selecting actions that are no longer complementary. The robots' beliefs start to drift apart and coordination fails as the robots move away from each other in the environment. Any observed cooperation is therefore an artifact of having similar sensor readings.

As the Bayesian game approximation uses the Q_{MDP} heuristic for the utility of future actions, it is not surprising that the policies generated by it appear similar to those of Q_{MDP} at the beginning of a trial (e.g. both exhibit guarding behaviours where one robot waits at an intersection while the other travels around a loop or down a hallway). Unlike Q_{MDP} , however, the robots are coordinating their action selection and so are able to continue to do a good

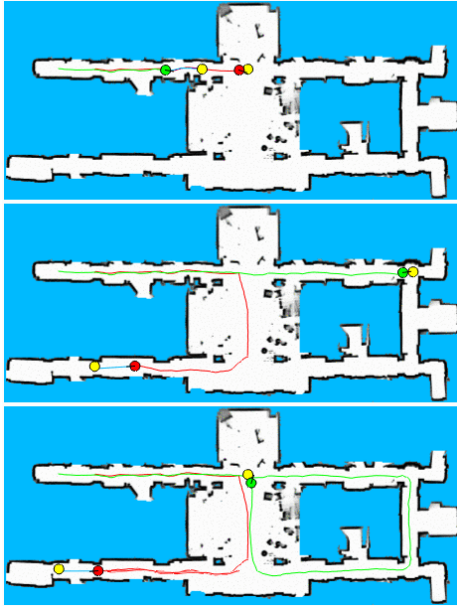


Fig. 3. *Robot Tag A*: An example of the path taken by the robots using our controller. When controlled with the MLS heuristic the robots follow each other around and when controlled with the Q_{MDP} heuristic the robots start out with similar paths but do not travel down the bottom left corridor. Yellow circles represent the robots' current goal locations.

job of covering the environment even as their positions and sensor readings move away from each other (Fig. 3).

In *Robot Tag B*, the robots no longer know the position of their teammate which has subtle effects on how they coordinate. In MLS the robots continue to follow each other because they receive similar sensor readings. Robots running Q_{MDP} start to follow a similar path to the one they would follow in *Robot Tag A*, but as they lose track of their teammate's position they stop covering the environment as well and can get trapped in guarding locations for a teammate that is also guarding a different location. The Bayesian game approximation, however, still covers the environment in a similar fashion as to when teammate positions are known (Fig. 4). This is because through the retained histories they are able to maintain a probability distribution over the true location of their teammate.

V. DISCUSSION AND FUTURE WORK

Clustering is an effective way to reduce the number of maintained history is a principled way. This results in faster computation making our algorithm more appropriate as a controller for real-world problems. While low probability clustering is computationally faster, minimum distance clustering is able to find a more natural set of histories that best represent the overall history space of each robot.

Another effective way to reduce the number of histories is to permit communication between robots on the team. If done in a sensible manner, robots would only need to communicate infrequently and therefore not exceed bandwidth limitations. We are currently investigating communication within the framework of our algorithm.

REFERENCES

[1] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multi-robot coordination," *IEEE Trans. on Robotics and Automation, Special Issue on Multi-robot Systems*, vol. 18, no. 5, 2002.

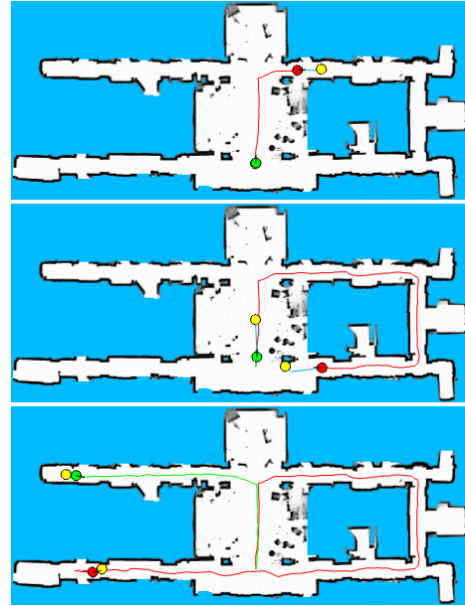


Fig. 4. *Robot Tag B*: An example of the path taken by the robots using our controller. The robots start at the bottom of the middle corridor and the green robot guards the intersection while its teammate moves around the loop. Unlike the MLS and Q_{MDP} heuristics, this path is similar to the one taken by the robots when they know their teammates' positions.

[2] M. B. Dias and A. Stentz, "Free market architecture for distributed control of a multirobot system," in *IAS*, 2000.

[3] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multi-robot cooperation," *IEEE Trans. on Robotics and Automation*, vol. 14, no. 2, 1998.

[4] R. Arkin, "Cooperation without communication: Multi-agent schema based robot navigation," *J. Robotic Systems*, vol. 9, no. 3, 1992.

[5] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun, "Towards robotic assistants in nursing homes: Challenges and results," *Robotics and Autonomous Sys.*, vol. 42, pp. 271–281, March 2003.

[6] M. T. J. Spaan and N. Vlassis, "A point-based POMDP algorithm for robot planning," in *ICRA*, 2004.

[7] R. Nair, M. Roth, M. Yokoo, and M. Tambe, "Communication for improving policy computation in distributed POMDPs," in *AAMAS*, 2004.

[8] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun, "Approximate solutions for partially observable stochastic games with common payoffs," in *AAMAS*, 2004.

[9] D. Bernstein, S. Zilberstein, and N. Immerman, "The complexity of decentralized control of Markov decision processes," in *UAI*, 2000.

[10] D. Pynadath and M. Tambe, "The communicative multiagent team decision problem: Analyzing teamwork theories and models," *J. Artificial Intelligence Research*, 2002.

[11] P. Gmytrasiewicz and P. Doshi, "A framework for sequential planning in multi-agent settings," in *Int. Symp. on AI and Math.*, 2004.

[12] E. Hanson, D. Bernstein, and S. Zilberstein, "Dynamic programming for partially observable stochastic games," in *AAAI*, 2004.

[13] L. Peshkin, K.-E. Kim, N. Meuleau, and L. P. Kaelbling, "Learning to cooperate via policy search," in *UAI*, 2000.

[14] P. Xuan, V. Lesser, and S. Zilberstein, "Communication decisions in multi-agent cooperation: Model and experiments," in *Agents*, 2001.

[15] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella, "Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings," in *IJCAI*, 2003.

[16] D. Fudenberg and J. Tirole, *Game Theory*. MIT Press, 1991.

[17] S. Russell and P. Norvig, "Section 6.5: Games that include an element of chance," in *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2nd ed., 2002.

[18] J. Shi and M. L. Littman, "Abstraction methods for game theoretic poker," *Computers and Games*, pp. 333–345, 2000.

[19] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman, "Acting optimally in partially observable stochastic domains," in *AAAI*, 1994.

[20] B. S. Everitt, S. Landau, and M. Leese, *Cluster Analysis*, ch. 4. Arnold Publishers, 4th edition ed., 2001.

[21] M. Montemerlo, N. Roy, and S. Thrun, "Carnegie Mellon robot navigation toolkit," 2002. www.cs.cmu.edu/~carmen.