Multi-robot planning under uncertainty with communication: a case study

João V. Messias
Institute for Systems and Robotics
Instituto Superior Técnico
Lisbon, Portugal
jmessias@isr.ist.utl.pt

Matthijs T.J. Spaan
Institute for Systems and Robotics
Instituto Superior Técnico
Lisbon, Portugal
mtjspaan@isr.ist.utl.pt

Pedro U. Lima
Institute for Systems and Robotics
Instituto Superior Técnico
Lisbon, Portugal
pal@isr.ist.utl.pt

ABSTRACT

Although Dec-POMDP techniques can be useful to modeling a wide range of problems, their practical application is limited by the inherent computational complexity of the algorithms currently available to solve such models. The application of these techniques is typically restricted to theoretical examples. This work studies the application of a particular type of Dec-POMDP (a multiagent POMDP) model to solve a simple task in a realistically simulated robotic soccer environment. The multiagent POMDP exploits the availability of a communication channel, as is often the case in multi-robot systems. The necessary constraints on the problem are identified, and the steps taken to accomplish efficient cooperative behavior within real robot middleware are explained. Finally, results are presented for the proposed task that highlight further possible improvements.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Performance, Experimentation

Keywords

Multiagent Decision Processes. Planning Under Uncertainty. Cooperative Robotics.

1. INTRODUCTION

Planning and decision-making is one of the central topics of robotics and operations research. In most realistic environments, it is necessary to take into account uncertainty in the agent's actions and/or observations. A Markov Decision Process (MDP) is a widely known and well-studied mathematical framework to model problems where the outcome of an agent's actions is probabilistic, but knowledge of the agent's state is assumed [6]. For this type of problems, several algorithms exist that provide optimal and approximate solutions to MDPs in reasonable time.

When the agent's knowledge is insufficient to directly determine its state, for example a mobile robot with noisy sensors, then the uncertainty of its observations must also be considered. Such problems are where the Partially Observable Markov Decision Process (POMDP) framework finds its domain of application. Planning under uncertainty for a single agent using POMDPs has been the object of active research for the last decade [7, 8]. While solving a POMDP optimally in its general case is an difficult problem, various algorithms exist to compute approximate solutions to moderately-sized POMDPs [2, 8, 11] that may find some application in real-world scenarios.

In certain applications, however, a single agent is not enough to model the full scale of the problem. Such is the case, for example, in cooperative robotics, where multiple agents must work together to achieve a common goal. The Decentralized POMDP framework is a natural extension of the POMDP paradigm for multiple agents. In this type of model, not only do the agents need to consider the uncertainty of their own actions and observations, but also that of their partners, which may (or may not) be aided by the use of communication [12, 13]. Naturally, being more general, this type of models is also harder to solve than their POMDP and MDP counterparts. In fact, optimally solving a general Dec-POMDP is provably intractable [5, 1]. However, approximate solutions can be found [4], even if the computational complexity of the current existing algorithms restricts their application to small-scale problems. This same problem limits their usability in real world scenarios, and so far their application has been restricted to theorical experiments and small simulated examples [1]. Special classes of this model include Multiagent MDPs and POMDPs (MMDPs) and decentralized MDPs (Dec-MDPs) [1].

This work is means as an example of the application of POMDP-related techniques in a small-scale realistic scenario. By doing so, the necessary constraints and possible simplifications to the general model can be identified, and the underlying implementation problems become apparent. Specifically, the problem under study is identified as a multiagent POMDP, a particular form of Dec-POMDPs. The case-study for this work is robotic soccer, a widely known environment for cooperative robotics. Before applying these techniques directly to a real team of soccer robots, which would be impractical, a reasonable approach is to first test the behavior of such robots in realistic simulated environments. This work describes the steps taken from the design of a cooperative robotic task to its implementation in a real robotic middleware using techniques within the Dec-POMDP framework. We establish that, with some reason-

AAMAS 2010 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2010, Toronto, Canada.

able abstractions and simplifications, (Dec-)POMDP techniques can be applied in real problems, and that, within the context of this particular case-study, the obtained results are comparable, and in some aspects advantageous, to other established decision-making frameworks.

2. THE DEC-POMDP MODEL

A Dec-POMDP is a tuple $< n, S, A, \Omega, T, O, R, h >$ where:

- n is the number of agents;
- S is the discrete set of states for the system. The initial state, $s_0 \in S$, is assumed to be unknown to the agents;
- A is the discrete set of joint actions. $A = \times_i A_i$, with A_i the set of actions available to agent i. At each step a joint action $a =< a_1, ..a_n >\in A$ is selected, where $a_i \in A_i$ is the action selected by agent i;
- Ω is the set of joint observations. As with the joint actions, $\Omega = \times_i \Omega_i$, and at each step a joint observation $< o_1, ..., o_n >\in \Omega$ is taken;
- T is the transition function, i.e. for states $s, s' \in S$, and joint action $a \in A$, $T(s', s, a) = P(s'|s, a)$, the probability of making a transition from state s to s' by applying action a;
- O is the observation function. $O(o, a, s') = P(o|a, s')$ specifies the probability of observing o after applying action a and ending up in state s';
- R is the reward structure. $R : S \times A \rightarrow \Re$ specifies the immediate reward granted to the agents as a whole, for taking joint action a in state s;
- h is the horizon of the problem, which represents the number of steps (decisions) that the agents can take.

The framework also allows models for communication between the agents [9]. In this particular context, communication is assumed free (although not instantaneous, refer to section 5). For a more thorough description of Dec-POMDPs and their mathematical properties, see [5].

In the most general case, in order to choose a specific action at each time step, each agent needs to take into account all the possible actions taken by every other agent up until that time step, and all the observations that they might have received. However, in scenarios where the agents are able to communicate freely with each other, the problem becomes simpler, since it is then possible to compute a probability distribution over the set of joint states (a *joint belief*) given only the latest information gathered by the agents. This Markovian signal eliminates the need to consider the complete history of the agents' histories. In this sense, the problem then reduces to a Multiagent POMDP problem. In each step, a joint action for all agents is selected based on this joint belief, which is then updated given the joint observation gathered by the agents.

3. MODELING A ROBOTIC SOCCER TASK

The task proposed in this work is based on a simple situation where two robotic soccer players must cooperate in order to take the ball towards their opponent's goal. During the course of their task, the robots may encounter obstacles that they should be able to avoid, although the position of these obstacles is not known beforehand. One of these players should carry the ball forward, and the other should position itself so that it may receive a pass from its partner, if necessary. The robots may choose to pass the ball in order to avoid imminent obstacles, since it is difficult to avoid obstacles while carrying the ball. The robot that carries the ball at any given time will be referred to in this case study as the "Attacker", and its partner the "Supporter". Whenever a pass occurs, the roles of the robots should switch, meaning that an Attacker becomes a Supporter and vice-versa. The Attacker should then kick the ball to the goal as soon as it detects an opportunity to score. The initial position of the robots and of the ball in their field of play is unknown, and so is their role. They should then determine which robot should carry the ball. The robots possess sensors to detect their own location, the position of the ball, and any surrounding obstacles.

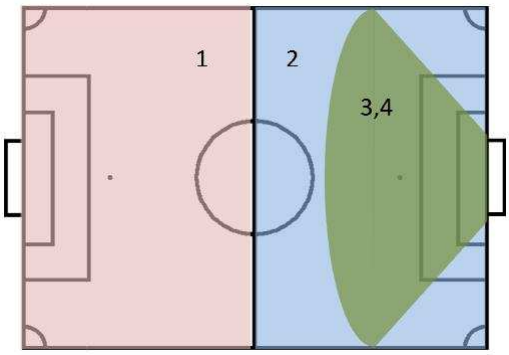

Figure 1: The different sections into which the field of play is divided according to an agent's localization information: 1-Its own half; 2-The opponent's half; 3-Near the goal; 4-In a shooting opportunity. Note that 3 and 4 are coincident with respect to position but vary in their orientation requirements.

3.1 Identifying states, observations, and actions

The first step in the modeling process of this task as a Multiagent POMDP is identifying the states of the overall system. It is assumed that the robot's field of play only contains the agents themselves, the ball, and an unknown number of opponents, and except for these obstacles, their navigation is free inside the field. The state of the robots can then be encoded through their localization information, the position of the ball, and the presence of obstacles. Regarding localization, the field of play is discretized into four different sections, as represented in Figure 1. The agent may be located in its own half-field, in its opponent's half, near the opponent's goal, or in a shooting opportunity, which requires not only for the robot to be near the goal while carrying the ball, but also turned towards it. The robot may also use localization information to sense if its ready to receive a pass from its partner. The information regarding obstacles can be encoded in a binary form, in the sense that the robot is either blocked by obstacles or free to move in its current direction. Finally, the robot is also able to detect whether or not it is in possession of the ball. Note that the robots share their localization information —they require this information in order to be able to follow each other and to be able

to sample their own observations (see Section 5). This is accomplished through explicit communication, as described in the following sections. None of the remaining state variables (information about the ball and obstacles) is shared, since they are not required by each robot's partner. This means that, given this information, the robot is able to estimate (with uncertainty) its own state, but not his partner's.

According to this description, each agent may then be in one of the 13 local states $s_i \in S_i$ described in the diagram in Figure 2. However, these local states are not independent, i.e., the problem specification mandates that one of the robots is an Attacker and the other a Supporter. Therefore, the total number of states is 60, which results from the admissible combinations of local states. It is important to note that this does not affect the partial observability of the agents. They may still receive conflicting observations (for example, that they are both Attackers or Supporters).

Since the complexity of most POMDP algorithms is heavily dependent on the number of observations, it follows that this set should be as reduced as possible. Since the robots must be able to switch roles, this set is necessarily the same for each agent. The set of possible observations is described, for each agent, as follows:

- Own Half —Having the ball in the half-field belonging to the agent's team;
- Opponent's Half —Having the ball in the opposing team's half-field;
- Near Goal —Having the ball near the goal;
- Ready —for an Attacker, this signals that the robot is in a shooting opportunity. For a Supporter, this implies that the robot is able to receive a pass;
- Not Ready —signal received by an Attacker without the ball, or a Supporter which isn't ready to receive a pass.
- Blocked —if the agent is blocked by obstacles.

The preceding signals encapsulate all necessary information about the environment, and result in 36 possible joint observations. The construction of the associated observation model, $O = \langle o^i, o^j \rangle, i = 1, ..., 6, j = 1, ..., 6$, may be further simplified by exploiting observation independence between agents (Section 3.2).

Note that the observation set for each agent does not depend on its specific role as either an Attacker or a Supporter. In some instances (the Ready and Not Ready signals), it is possible to use the same representations implicitly for both cases, since each agent will take its own observation into context through the observation function.

These observations are, in themselves, quite abstract, and each of them depends on possibly more than one source of information. To achieve this level of abstraction in real robots, it is necessary to implement high-level "virtual sensors" that classify the information collected by the robots' physical sensors (and, in the case of localization, information shared by the partner robot) into one of the observations defined above. The observation function can then be constructed in practice by collecting the output of these classifiers while setting the robot in a specific, *a priori* known state. The observation function is then estimated through the collected data.

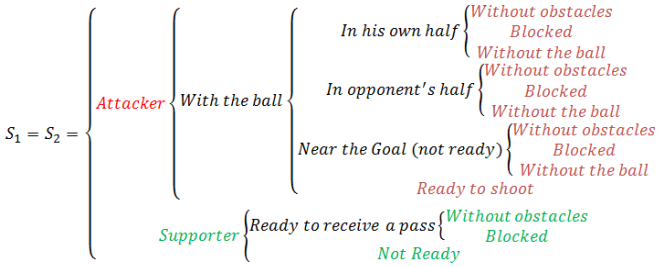

Figure 2: Diagram showing the different possible local states the agent may be in. Note that they are not independent, since for one agent to be the Attacker the other must be the Supporter and vice-versa.

The remaining component of this model that must be described is the set of joint actions, A. As with the observation set, this set is identical for both agents. For this particular task, the following actions are sufficient:

- Dribbling the ball towards the goal;
- Shooting (or kicking) the ball;
- Passing the ball towards the other robot;
- Recovering the ball if it becomes lost by the attacker;
- Following the attacker;
- Finding a position where a pass can be received (finding clearance for the pass).

Logically, the first four actions described in this manner should be performed by the Attacker robot, while the remaining actions should be taken by the Supporter. Therefore, $A = \langle a^i, a^j \rangle, i = 1, ..., 6, j = 1, ..., 6$.

Notice that these actions are defined as high-level behaviors that each robot can assume. Each of these high-level actions is then interpreted by the robot's middleware, and triggers a series of more basic behaviors, that may possess their own local decision-making loops. When the robot decides to dribble towards the goal, for example, these lower-level behaviors ensure that the robot is always turned towards the goal, and supplies the robot with the necessary controls so that it may drive the ball and try to avoid any imminent obstacles. The specific mechanisms through which this is accomplished lie outside of the scope of this work. However, it is important to note that the actions taken at this level impact the transition function of the Multiagent POMDP model. To define such functions rigorously, it is necessary to collect experimental data to such an extent that the transition probabilities estimated through this data approximate the correct values of the transition function (although this may be aided by the use of simulators).

3.2 Exploiting Local Independence

Given the set of joint observations and the set of joint actions, it is then necessary to describe the uncertainty in each of these elements, which is to say that the transition function T and observation function O must be defined. However, in constructing these models, it is advantageous to reduce as much as possible the required information about

the environment. Ideally, if the local state of an agent was not influenced by the actions of the other agent, the model would be completely conditional independent, i.e.:

$$P(s'|s, a_1, a_2) = P(s'|s, a_1)P(s'|s, a_2) \quad (1)$$

This would mean that there is an independent transition function for each agent, T_i, such that

$$T(s', s, a) = T_1(s', s, a_1)T_2(s', s, a_2) \quad (2)$$

for every $s, s' \in S, a = \langle a_1, a_2 \rangle \in A$. From the above description of the action set, it is evident that for some of the actions (namely, passing and shooting), this assumption is not valid, since the application of one of these actions by an agent may induce its partner to switch its role. However, it is valid for all of the remaining actions. The problem may still be further simplified by noting its symmetry. Since there is no characteristic feature to distinguish one agent from the other, their transition functions are identical, $T_1 = T_2$. This means that its only necessary to consider the effects of the 4 possible independent actions for each agent (which is a considerable reduction from the 36 possible joint actions). Also, since only half of the states correspond to a specific agent being Attacker or Supporter, this means that, in matrix form, the transition function for each of the independent actions is block-diagonal (i.e. it is impossible to transition from being an Attacker to a Supporter by applying these actions). For the joint actions that are not conditional independent, their distribution over the possible 60 states must be obtained.

For the observation model, a similar rationale can be taken, but in this case the problem is further simplified by noting the full observation independence in this particular Multiagent POMDP model. At first sight, the passing and kicking actions could be understood to also influence the observations of the respective partner robot, but this is indeed not the case, since the observations have been defined independently for each state, and the actions taken by the partner robot do not influence the ability of each agent to perceive its respective information. The joint actions in this task can then be said to be non-informative, in that they may influence the state of the system, but not the sensors of the agents. In practice, this means that:

$$P(o|a, s') = P(o_1|a, s')P(o_2|a, s') \quad (3)$$

$$= O_1(o_1, a, s')O_2(o_2, a, s'), o_{1,2} \in \Omega_{1,2} \quad (4)$$

It should also be noted that $O_1 = O_2$.

So far nothing has been said about the reward structure for this particular robotic task. The agents will choose a course of action (a policy) according to the expected reward obtained in future steps (see Section 4). Although the behavior of the agents can be indirectly influenced by manipulating the reward structure in such a way that the desired actions are promoted, this is undesirable since it would in fact encode the optimal policy in the model itself, and would remove any merit from its solution. The definition of the reward model is simply to assign a high reward for kicking the ball in a shooting opportunity, and to penalize (lightly) every other step taken.

3.3 Comments on the Functionality of the Multiagent POMDP model

Although the transition and observation models for this particular Multiagent POMDP have been found to be relatively simple to define in theory, in order to rigorously obtain these models in practice, it is necessary to estimate the respective probability distributions by collecting large amounts of experimental data. However, due to restrictions in time, this was overlooked in favor of using empirically estimated values. The effect of this decision on the optimality of the resulting joint policy will be noted. The manner in which the environment was discretized into states may also prove some difficulties with respect to the definition of these models. Since the states were defined in a loose topological manner, they are coarse relative to the dimension of the agents, and so the probability of transitioning to a neighbour state depends heavily on the particular configuration of the agent inside a given state. Although this effect can be modelled, to some extent, in a given action's transition distribution, the resulting distribution will be necessarily flat, and little information can be taken from it. A possible way to overcome this problem is to take advantage of the specific sensor information (for localization) of the agent, which contains much more information than what is used by the Multiagent POMDP.

4. OBTAINING AN APPROXIMATELY OPTIMAL POLICY

Although various algorithms allow for the approximate solution of Dec-POMDPs in its most general form, such as JESP [3], Bayesian Game based approaches [4], and Memory Bounded Dynamic Programming [14], these algorithms typically take into account the *history* of each of the agents, i.e. all the past actions taken, as well as the perceived observations. This greatly increases the computational complexity of the problem. For the simpler case of Multiagent POMDPs, more efficient algorithms exist. The Perseus algorithm [2] was chosen to solve the Multiagent POMDP in this work, due to its efficiency in handling moderately-sized POMDPs. Perseus belongs to the family of point-based POMDP solvers, but it is by no means the only one [11, 15]. While it is true that often the algorithm to solve a given POMDP model should be chosen according to the problem's structure, this does not create, in this case, a dependancy on any particular algorithm. In fact, if communications were assumed instantaneous, the model could simply be reduced to a single-agent POMDP and dealt with accordingly by any appropriate solver.

The main objective of any MDP-based model is to maximize the expected reward of the agent(s) obtained for a given number of steps, the problem's horizon. This is described by the optimal *value function* associated with a particular MDP. The optimal action to take at each step is then simply the action that maximizes the value function. In the POMDP case, this function has the useful property of being piecewise-linear convex, i.e. the value of a given belief is a linear combination of a set of vectors:

$$V^*(b) = \max_k \sum_i b_i \alpha_i^k, \quad (5)$$

where b represents the belief (in this case the joint belief), t is the number of remaining decisions to make, and α^k are the vectors that define the linear segments of the value function. Although the value function itself can be efficiently described in this manner, the number of vectors that define the value function k, increases, in the worst case, exponentially in

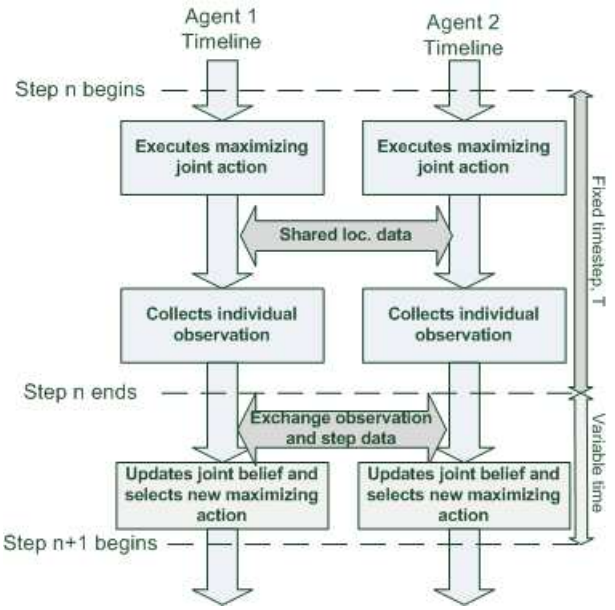

Figure 3: Synchronization timeline with instantaneous communication.

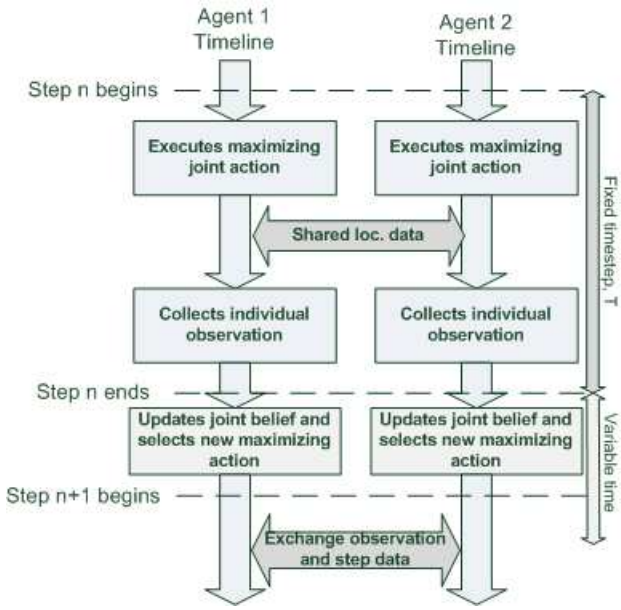

Figure 4: Synchronization timeline with delayed communication of observations.

the number of observations as the horizon of the problem increases, which is problematic when attempting to calculate infinite-horizon solutions. Instead of taking the whole belief space into account, the Perseus algorithm [2] considers only a set of reachable belief points, obtained through "simulated" exploration using the POMDP's transition and observation models. The algorithm then samples belief points from this reachable set at each step and calculates its corresponding α-vector (a reasonably inexpensive step) until the value for all points in the belief set has been improved. The algorithm then continues to iterate until a convergence criteria is met.

5. COMMUNICATION

In order for the problem to be modeled as a multiagent POMDP, it was assumed that the robots could communicate their observations freely to each other. However, the agents' optimal actions are dependent on the joint belief, which depends on each of their partners' observations (their actions are already known since the joint policy is common knowledge). For this information to be coherent, the robots must keep synchronized when carrying out their policies, i.e. they both must execute an action at (approximately) the same time. This way, the agents will calculate the same joint belief at each step.

The synchronization process and necessary explicit communication is performed according to the diagram in Figure 3. Each agent is assumed to perform its own component of the maximizing joint action for a fixed time step T. The agents' observations are only available after the outcome of this action is known. Therefore, after T has passed, the agents sample their own observations and exchange it with their partners'. If one of the agents is delayed, then its partner will wait for this information before proceeding. This step is where synchronization is enforced between both robots. This information is then used to locally calculate the joint belief. After the joint belief is obtained, each agent computes a new maximizing joint action and proceeds to the next step.

In a realistic scenario, it might not be feasible to assume instantaneous communication between the agents. This is often the case in robotic soccer, since the high number of agents quickly saturate the communication medium. In such a setting, it can be advantageous not to wait for synchronization between all involved agents. In this sense, the agent would select an optimal action based on its own local observation, and receive its partners' data throughout the decision step (Figure 4). It is then necessary to investigate the effect of receiving delayed information. Spaan et al. determined that the solution of a Dec-POMDP in such a setting, with up to one step of delay in the communication process, can be acheived through a series of Bayesian games [10]. Using a slightly modified version of the Perseus algorithm, results were obtained for this case.

6. RESULTS

Since the planning and execution phases must be carried out separately in the proposed task, here too they should be made distinct. With respect to planning, the Perseus algorithm performed favorably, and converged in as few as 100 iterations, as can be seen in Figure 5. Such a value function is a good approximation of a stationary solution, i.e., a solution that assumes an infinite horizon. Such a result demonstrates the tractability of the proposed Multiagent POMDP model.

The execution of the task itself was tested in the Webots simulation environment, which allows for realistic physics and control of the robotic agents. Furthermore, the real soccer robots upon which these agents were modeled may be controlled by directly using the same code as in the simulator, increasing its overall realism. The two agents were placed in arbitrary initial positions in the field of play, and the ball was initially placed in the center of the field (which is

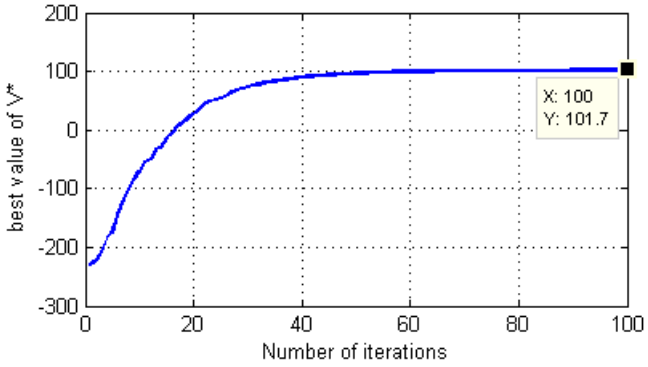

Figure 5: Convergence of the Perseus algorithm for the proposed Multiagent POMDP.

the origin of the world frame for the robots). The time step with which these agents should select an action was set to 3 seconds. The robots possessed an estimated 0.4 probability of performing a transition to a neighbour state when dribbling the ball (again, this was done empirically). The observation model for these robots considered both the possibility of having false positive and false negative detections of obstacles (0.1 probability of failing to detect an obstacle and 0.05 probability of detecting an inexistent obstacle). The localization uncertainty was deliberatily made small (around 0.05) since the information it provides is coarse and may heavily affect the agents' joint belief in the case of an erroneous observation.

Generally, the robots succeeded in their task in an efficient manner. In the instantaneous communication case, the expected reward was approximately ~ 360, with 150 the immediate reward for scoring a goal, -1 for all other actions and a discount factor of 0.95. The discrepancy between the expected reward through simulation and the final expected value as obtained by Perseus is caused by the automatic termination of each episode by the solver after scoring a goal, whereas in the simulator the ball was automaticaly reset to the center of the field and rewards continued to be accumulated.

When introducing a one-step delay in the communications, the robots performed less favorably with $\sim$ 30 expected reward. The comparative results shown in in Figure 6 were obtained by simulating both policies for 500 runs. The fact that this reward is positive, however, demonstrates that even in such a case, the robots are still able to cooperate in order to score goals, albeit notably less efficiently.

Although it is difficult to demonstrate the behaviors carried out by the robots in practice, two different situations are here presented that highlight the correct performance of the desired robotic task. These refer to the policy obtained assuming instantaneous communication.

In Figure 7, the positions of the robots and of the ball that were recorded from the simulator in a typical situation are shown. The supporter robot (here shown initially through a dashed line) maintains a fixed distance to the attacker, until at t_2 the attacker scores a goal. The ball is reset by the simulator back to the center of the field. Since the initial supporter is now the closest robot to the ball, it assumes the role of attacker, and at t_3 their roles have been exchanged

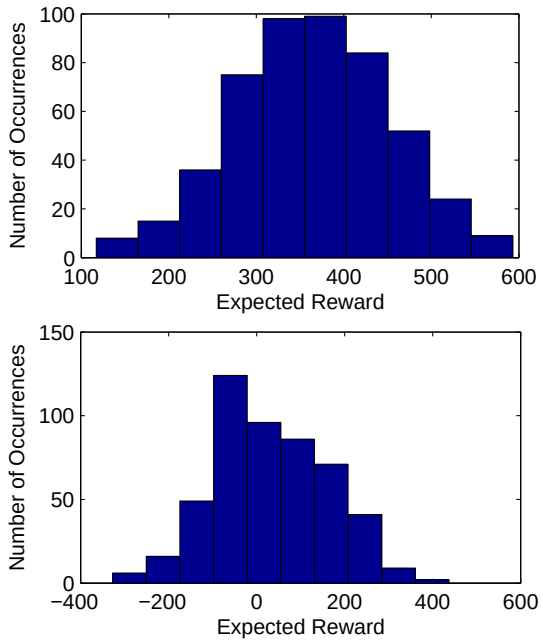

Figure 6: Expected reward collected from simulating the proposed task with instantaneous communication (top) and one-step delay (bottom).

from their initial configurations. The process then repeats itself. Note that in this case, there are no obstacles in the field other than the agents themselves.

A second situation occurs in the presence of obstacles, and is depicted in Figure 8. A barrier of obstacles is placed in front of the initial attacker agent (shown in a dashed line) and at t_2 it selects a pass action since its partner has better clearance to the goal. Their roles then switch, and the agent shown by a filled line then carries the ball until it scores a goal at t_3. Note that the initial attacker still decided to carry the ball for a short amount of time before passing. This is due to the fact that the agent is commited to performing the latest selected joint action until the predefined time-step expires. This presents a problem in dynamic environments, since the robot may not have enough time to select the optimal action when presented with a sudden change in its state. However, the time-step for these decisions cannot be reduced too much, since otherwise the robots do not have enough time to experience the effects of their own actions, i.e., they would most likely remain in the same state when using such coarse topological state definitions as the one proposed in this robotic task.

It is apparent that the resulting policy in both of these cases provides results that are comparable to those obtained with decision-making frameworks that are more common in this context (for example, manually defined policies through finite-state automata), despite the simplifications that were made while modeling the problem. These policies were obtained naturally as the solution of the associated Multiagent POMDP, and, once the observation and transition functions are obtained, only the reward model needs to be adjusted if a different task must be performed (provided that the possible

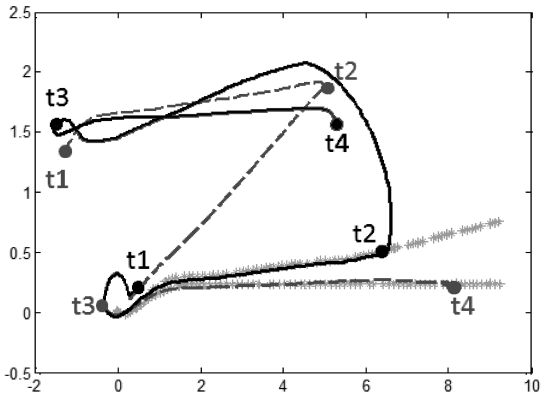

Figure 7: Behavior of the robots after scoring a goal. The positions of the initially attacking and supporting robots are shown by a filled and dashed line, respectively. The position of the ball is marked in gray. The goal is located at (9,0) and is 2m wide.

actions and observations remain the same). This may prove advantageous in environments where the possible tasks are repetitive and predictable.

7. CONCLUSIONS AND FUTURE WORK

This work described the implementation of a robotic task based on a Multiagent POMDP model in a realistic simulated scenario. The steps taken to model, solve and implement the task itself were presented, and results were taken from various simulations. These simulations show that the robots are able to efficiently complete the proposed task with the given model, even if their transition and observation functions are not rigorously defined. In this manner, it was established that it is possible to perform effective decision-making in realistic scenarios by using the Dec-POMDP framework.

The advantage of this approach over most current policy-definition methods in the robotic soccer environment is that it deals with uncertainty in the local information of the robots (regarding the position of obstacles or the ball, for example) in a natural way, without having to manually describe the desired policy for each of the agents. The model itself proved to be solvable in reasonable time. The effect of losing immediate synchronization between the agents was studied. It was shown that the control quality of the agents suffers, although the resulting policy still permits the agents to complete its task.

One problem with this type of approach is related to the time that must elapse between two successive decision steps. If this time value is too large, then the robot loses its ability to react to sudden changes in its state, as when it encounters obstacles along its course, for example. Even if the robot is able to receive an observation consistent with these changes, it may not be enough to alter the belief of the agent sufficiently for it to perform the desired action (the observation model has to be narrow in this sense). If the elapsed time between iterations is too small, then there is not enough time for the robots to transition to another state, and so the transition function would become flat and uninformative.

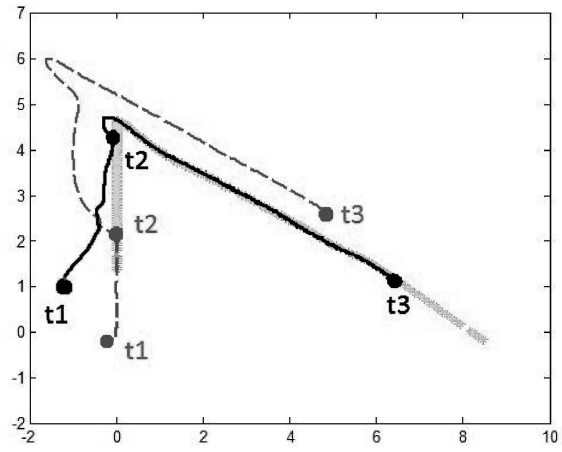

Figure 8: Behavior of the robots when passing to avoid obstacles. The initial attacker, in this case, is shown by the dashed line. At t_2 a pass occurs.

As future work, it would be advantageous to develop a mechanism to condition the observation model on each agent's local information, which is typically much more accurate than the information contained in the coarse topological definition of the system's state.

8. REFERENCES

[1] Claudia V. Goldman, Shlomo Zilberstein, *Decentralized Control of Cooperative Systems: Categorization and Complexity Analysis* , Journal of Articial Intelligence Research, 22: 143-174, 2004.

[2] Matthijs T. J. Spaan and Nikos Vlassis, *Perseus: Randomized Point-based Value Iteration for POMDPs.* Journal of Articial Intelligence Research, 24: 195-220, 2005.

[3] Ranjit Nair, David Pynadath, Makoto Yokoo, Milind Tambe, and Stacy Marsella, *Taming Decentralized POMDPs: Towards Efficient Policy Computation for Multiagent Settings.* In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI), pages 705-711, 2003.

[4] Frans A. Oliehoek, Matthijs T.J. Spaan, Nikos Vlassis, *Optimal and Approximate Q-value Functions for Decentralized POMDPs.* Journal of Artificial Intelligence Research, 32: 289-353, 2008.

[5] D. Bernstein, R. Givan, N. Immerman, S. Zilberstein, *The complexity of decentralized control of Markov decision processes.* Mathematics of Operations Research, 27(4): 819-840, 2002.

[6] R. Bellman, *Dynamic programming.* Princeton University Press, 1957.

[7] H. T. Cheng, *Algorithms for partially observable Markov decision processes.* Ph.D. thesis, University of British Columbia, 1988.

[8] M. L. Littman, A. R. Cassandra, L. P. Kaelbling, *Learning policies for partially observable environments: Scaling up.* In International Conference on Machine Learning, San Francisco, CA, 1995.

[9] P. Xuan, V. Lesser, S. Zilberstein, *Communication decisions in multi-agent cooperation: Model and experiments.* In Proc. of the International Conference on Autonomous Agents, 2001.

[10] Matthijs T. J. Spaan, Frans A. Oliehoek, and Nikos Vlassis, *Multiagent Planning under Uncertainty with Stochastic Communication Delays.* In Proc. of Int. Conf. on Automated Planning and Scheduling, pp. 338-345, 2008.

[11] H. Kurniawati, D. Hsu, and W. Lee, *SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces.* In Proc. Robotics: Science and Systems, 2008..

[12] M. Roth, R. Simmons and M. Veloso, *Reasoning about joint beliefs for execution-time communication decisions.* In Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp. 786-793, 2005.

[13] D. V. Pynadath and M. Tambe, *The communicative multiagent team decision problem: Analyzing teamwork theories and models.* Journal of Artificial Intelligence Research, 16, 389-423, 2002.

[14] S. Seuken and S. Zilberstein, *Memory-bounded dynamic programming for DECPOMDPs.* In Proc. of the International Joint Conference on Artificial Intelligence, pp. 2009-2015, 2007

[15] J. Pineau, G. Gordon and S. Thrun, *Point-based value iteration: An anytime algorithm for POMDPs* In Proc. Int. Joint Conf. on Artificial Intelligence, Acapulco, Mexico, 2003