Multiagent Collaborative Task Learning through Imitation

Sonia Chernova and **Manuela Veloso** [1]

Abstract. Learning through imitation is a powerful approach for acquiring new behaviors. Imitation-based methods have been successfully applied to a wide range of single agent problems, consistently demonstrating faster learning rates compared to exploration-based approaches such as reinforcement learning. The potential for rapid behavior acquisition from human demonstration makes imitation a promising approach for learning in multiagent systems. In this work, we present results from our single agent demonstration-based learning algorithm, aimed at reducing demonstration demand of a single agent on the teacher over time. We then demonstrate how this approach can be applied to effectively train a complex multiagent task requiring explicit coordination between agents. We believe that this is the first application of demonstration-based learning to simultaneously training distinct policies to multiple agents. We validate our approach with experiments in two complex simulated domains.

1 Introduction

Programming robots is a challenging problem due to sensor complexity, noise, and the non-deterministic effects of robot actions. To address this challenge, autonomous learning approaches have been developed that allow robots to learn task execution through interaction with the environment [14]. Most of these approaches, however, rely on a long trial-and-error experimental process that is impractical due to time constraints and physical wear on the robot. Learning in systems with multiple robots is further complicated by the complex interactions that can occur between distributed agents, such as communication via message passing, physical interaction and resource contention. To address these problems, natural and intuitive approaches must be developed that allow new skills to be taught to multiple of robots in a timely manner.

Learning from demonstration, a collaborative learning approach based on human-robot interaction, offers an alternative to exploration-based methods. The goal of this approach is to learn to imitate the behavior of a teacher by watching a demonstration of the task. Demonstration-based learning has been successfully applied to a variety of single agent learning problems [5, 8, 18, 28]; its fast learning rate compared to exploration-based learning methods, such as reinforcement learning, makes learning from demonstration a promising approach for multiagent systems.

In this work, we first present results of our single agent demonstration-based learning algorithm, the *confident execution* framework [10]. We then apply this framework to a collaborative multiagent domain, demonstrating its effectiveness in simultaneously training multiple robots to perform a joint task. Our learning framework aims to reduce each agent's demonstration demands on the teacher by allowing the agent to perform its task autonomously when it is confident about its actions, and request expert assistance at times of uncertainty. As a result, each agent operates with gradually increasing autonomy as the task is learned, relieving the teacher from repeated demonstrations of acquired behavior and allowing simultaneous supervision of multiple agents.

In the next section we discuss related work in the areas of demonstration and imitation learning, followed by a complete description of the confident execution learning framework in Section 3. In Section 4 we present experimental results demonstrating our approach in single and multi agent domains.

2 Related Work

Learning from demonstration is an interactive learning method in which the agent aims to imitate the behavior of an expert teacher. Demonstration-based methods have been successfully applied to a wide range of single agent learning problems.

Nicolescu and Mataric [17, 18] present a learning framework based on demonstration, generalization and teacher feedback, in which training is performed by having the robot follow a human and observe its actions. A high-level task representation is then constructed by analyzing the experience with respect to the robot's underlying capabilities. The authors also describe a generalization of the framework that allows the robot to interactively request help from a human in order to resolve problems and unexpected situations. This interaction is implicit as the agent has no direct method of communication; instead, it attempts to convey its intentions by communicating though its actions.

Lockerd and Breazeal [8, 15] demonstrate a robotic system where high-level tasks are taught through social interaction. In this framework, the teacher interacts with the agent through speech and visual inputs, and the learning agent expresses its internal state through emotive cues such as facial and body expressions to help guide the teaching process. The outcome of the learning is a goal-oriented hierarchical task model.

Bentivegna et al. [5, 6, 7] and Saunders et al. [25] present demonstration learning approaches using memory-based techniques. Both groups use the k-nearest neighbor (KNN) [16] algorithm to classify instances based on similarity to training examples, resulting in a policy mapping from sensory observations to actions. Our algorithm takes a similar approach by utilizing Gaussian mixture models for classification, but includes an interactive learning component similar to Nicolescu and Mataric. Inamura et al. [13] present a similar method based on Bayesian Networks [20] limited to a discretely-

[1] Computer Science Department, Carnegie Mellon University, email: soniac@cs.cmu.edu, veloso@cs.cmu.edu

valued feature set.

A handful of studies have also examined imitation in the context of multiagent systems. In the Ask For Help framework [11], reinforcement learning agents request advice from other similar agents in the environment. Help is requested when an agent is confused about what action to take, an event characterized by relatively equal quality estimates for all possible actions in a given state.

A similar approach is presented by Oliveira and Nunes [19], in which agents are able to select, exchange and incorporate advice from other agents, combining it with reinforcement learning to improve learning performance. The authors examine when and how agents should exchange advice, and which of an agent's teammates should be communicated with. Their results show that exchange of information can improve the average performance of learning agents, although it may reduce the exploration of the state space, preventing the optimal policy from being found in some cases.

Alissandrakis et al. [2, 3] present a general framework that enables a robotic agent to imitate another, possibly differently embodied, agent through observation. Using this framework, the authors demonstrate the transmission of skills between individuals in a heterogeneous community of software agents. Their results indicate that transmission of a behavior pattern through a chain of agents can be achieved despite differences in the embodiment of some agents in the chain. Additionally, the authors show that groups of mutually imitating agents are able to converge to a common shared behavior.

Price and Boutilier [21] present a multiagent system in which novice agents learn by passively observing other agents in the environment. Each learning agent is limited to observing the actions of others and no explicit teaching occurs. By observing a mentor, the reinforcement learning agent can extract information about its own capabilities in, and the relative value of, unvisited parts of the state space. However, the task of an observed agent may be so different that the observations provide little useful information for the learner, in which case direct imitation of this expert must be avoided by the algorithm.

The above methods study imitation from the perspective of a community of agents, where a single agent seeks advice from other members of its group. A different approach is taken in the study of coaching [23], where an external coach agent provides advice to a team of agents in order to improve their performance at a task. The coach has an external, often broader, view of the world and is able to provide advice to the agents, but not control them. The agents must decide how to incorporate the coach's advice into their execution. Riley [23] presents an approach for training the coach using imitation based on example executions.

Our approach differs from the presented techniques in that it enables a single human to simultaneously train multiple agents. The agents may be differently embodied, and may learn different policies and perform different tasks. In our proposed system, the human teacher is the only source of advice, providing demonstrations in the form of action commands.

3 The Confident Execution Framework

In this section, we present a summary of our confident execution learning framework which allows a single agent to learn a task policy from demonstration (for a more detailed description, please see [10]). We then describe how this framework can be applied to simultaneously training multiple robots to perform a joint task.

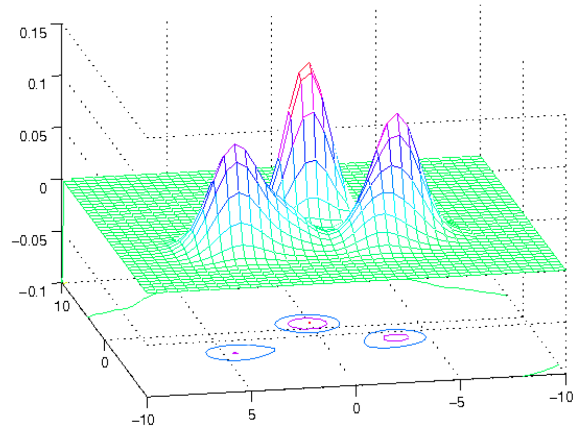

Figure 1. An example of a 2-dimensional Gaussian mixture model with three components. Contour lines below the GMM mark the one- and two-standard deviation ellipses.

3.1 Task Representation

Our approach utilizes the *learning by experienced demonstration* technique [18], in which the agent is fully under the expert's control while continuing to experience the task through its own sensors. During each training timestep, the agent records sensory observations about its environment and executes the action selected by the human expert. We assume the expert attempts to perform the task optimally, without necessarily succeeding.

Observations o are represented using an n-dimensional feature vector that can be composed of continuous or discrete values representing the state of the robot. The agent's actions a are bound to a finite set $\mathcal{A}$ of action primitives [4], which are the basic actions that can be combined together to perform the overall task. The goal of the system is to learn a policy $\pi : o \rightarrow \mathcal{A}$, mapping observations to action primitives. Each labeled training point (o, a) consists of an observation labeled by its corresponding expert-selected action.

During training, the algorithm separates all datapoints into classes based on their action label. A Gaussian mixture model (GMM), Figure 1, is then trained for each action class using the expectation-maximization (EM) algorithm [12]. We selected Gaussian mixture models for our approach due to previous successes of classification methods in demonstration learning [5, 25], and because GMMs provide a built-in measure of classification confidence. Their robustness to noise and ability to generalize and capture correlations between continuous features make them a powerful tool for robotic data analysis.

Since a single action is often associated with a number of distinct states (the action *turn left* may be taken from several different locations), we use a separate Gaussian mixture to represent each action class. Components within the mixture represent the different state regions and the number of components is determined using cross-validation. New datapoints are classified by selecting the Gaussian mixture with the maximum likelihood. The output of the classification is the action represented by the selected GMM. Additionally, the model returns a confidence value representing the certainty of the classification based on the likelihood.

Algorithm 3.1: THE LEARNING FRAMEWORK()

procedure INITIALTRAINING()
$observation \leftarrow$ GETSENSORDATA()
$expertAct \leftarrow$ GETEXPERTACTION()
$(gmmAct, conf) \leftarrow$ CLASSIFY($observation$)
$predAccuracy \leftarrow$ TRACKPRED($gmmAct, expertAct$)

if $numNewDatapoints >$ maxNew :
then UPDATEMODEL($observation, expertAct$)

EXECUTEACTION($expertAct$)
return ($predAccuracy$)

procedure CONFIDENTEXECUTION()
$observation \leftarrow$ GETSENSORDATA()
$(gmmAct, conf) \leftarrow$ CLASSIFY($observation$)

if $conf > confThresh$:
then {EXECUTEACTION($gmmAct$)
else {$expertAct \leftarrow$ GETEXPERTACTION(); UPDATEMODEL($observation, expertAct$); EXECUTEACTION($expertAct$)

Table 1. Pseudocode overview of the learning framework.

3.2 The Learning Process

Table 1 shows a pseudocode summary of the learning process. Learning begins with a non-interactive demonstration training phase during which each action of the robot is controlled by the expert through teleoperation. The algorithm uses training examples acquired from the demonstrations to generate a task model. Every time $maxNew$ additional training points are acquired, the algorithm updates the GMM model based on the new data. Additionally, the performance of the current learned policy is evaluated by comparing how closely it matches the behavior of the expert. Prior to updating the model with a new training point (o, a), the algorithm classifies observation o using the current model. It then compares the model-selected action to the demonstrated action a. Performing this comparison over a window of consecutive training points results in an estimate of the prediction accuracy of the model that relates how closely the policy matches the behavior of the expert.

The teacher performs non-interactive training until the model prediction accuracy is sufficiently high, as determined by the expert. At this point, learning transitions to the *confident execution* stage, during which the agent selects between autonomously executing its learned policy action and requesting help from the expert based on the classification confidence of the above model. The algorithm adjust the agent's autonomy by comparing the classification confidence to an autonomy threshold. Classification confidences greater than the threshold result in autonomous execution of the model-selected action, while confidences below the threshold cause the agent to pause its execution of the task and signal the teacher that a demonstration is needed.

Since the threshold value is continuous, our approach allows smooth adjustment of the autonomy level. This type of mechanism is referred to as adjustable, or sliding, autonomy and has been proven effective in a wide range of applications, from personal assistants [26] to space exploration [27]. Our algorithm combines learning with adjustable autonomy, resulting in an interactive teaching method that targets low confidence regions of the state space and reduces dependence on the human expert as the agent gains proficiency at its task. In the presented experiments, the human teacher manually sets the confidence threshold value that determines the level of autonomy. We are currently developing a technique for calculating this value automatically.

As the agent's model improves over time, the agent will encounter fewer observations with low classification confidence, resulting in fewer demonstration requests. Learning terminates when the agent is able to execute the task completely autonomously, or when the expert is satisfied with the performance of the model. The agent then deterministically executes the action selected by the model, regardless of the classification confidence. This mode of operation is typical of traditional learning approaches where the learned policy is always trusted once learning is complete.

3.3 Multiagent Approach

The confident execution learning framework is a promising approach for multiagent learning due to its fast learning rate compared to exploration-based methods such as reinforcement learning [10], and reduced demand on the expert over time. In this work, we examine how it can be directly applied to training multiple agents simultaneously.

In a multiagent setting, the expert's workload and teaching style differ depending on the degree of collaboration required between the agents. Domains with little collaboration allow each agent to operate with little regard for the actions of others, and training can be done independently for each agent. In such cases, it may be possible to introduce new agents at different times, resulting in a mixture of novice and expert agents to avoid overloading the expert at the beginning of the training stage. Domains that require greater collaboration between agents benefit from demonstration-based approaches because exploration over the joint action space of multiple robots is quite costly [9]. In these domains, it is beneficial to demonstrate the task to multiple collaborating agents at the same time.

Using our approach described in the previous section, each agent is able to learn its own individual policy regardless of the level of collaboration required. Our approach scales to an arbitrary number of robots without any modifications to the learning framework.

4 Experimental Results

We validate our approach using two simulated domains with continuous and multidimensional feature spaces.

4.1 Single Agent Driving Domain

In this section we present results of a fast and dynamic simulated car driving domain (Figure 2). In this domain, the agent takes the shape of a car that must be driven by the expert on a busy road. The speed of the car is fixed at 60 mph while all other cars move in their lanes at predetermined speeds between 20-40 mph. The learning agent can not change its speed, and must navigate between other cars to avoid collision. The agent is limited to three actions: remaining in the current lane, and shifting one lane to the left or right of the current position. The road has three normal lanes and a shoulder lane on both sides; the car is allowed to drive on the shoulder but can not go further off the road.

The environment is represented using four features, a distance to the nearest car in each of the three lanes and the current lane of the agent. The agent's lane is represented using a discrete value symbolizing the lane number. The distance features are continuously valued

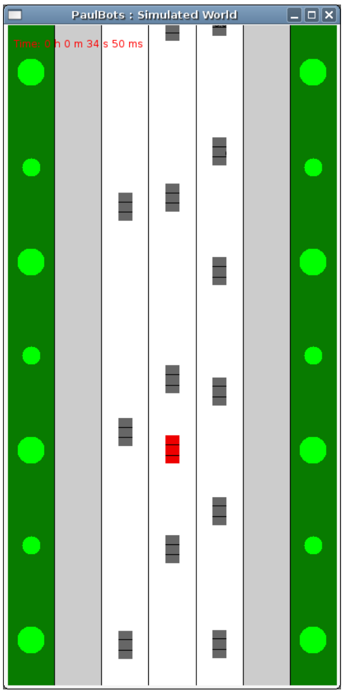

Figure 2. Screenshot of the driving simulator.

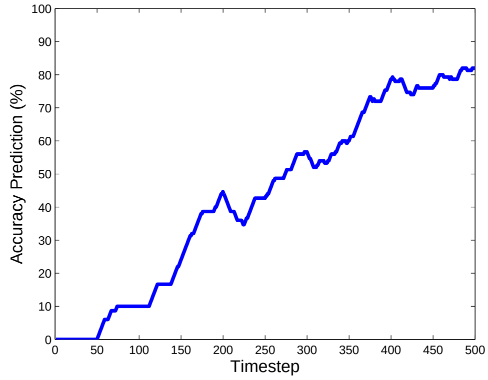

Figure 3. Prediction accuracy of the learned model over the non-interactive training phase using a window of 150 timesteps.

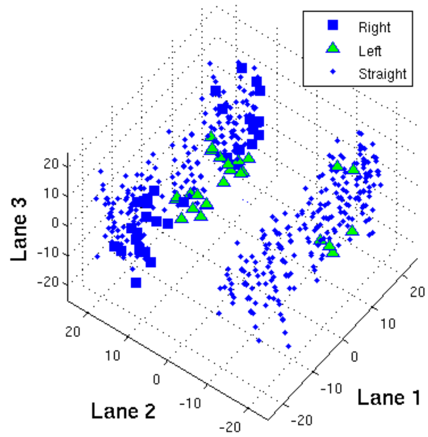

Figure 4. Driving training data representing the driving strategy used when the agent is in the middle lane. Graph axes represent distance to the nearest car in each of the three driving lanes.

in the [-25,25] range; note that the nearest car in a lane can be behind the agent.

Demonstration of the task was performed by a human using a keyboard interface. Figure 3 shows the prediction accuracy of the model during the initial non-interactive training phase. Training was performed until the model reached 80% prediction accuracy over a 150-timestep window, which resulted in a demonstration length of 500 timesteps, or approximately 2.1 minutes. After transitioning to the confident execution phase, the expert completed the training after 150 demonstration timesteps when the model exhibited good performance. During the confident execution phase all demonstrations were done as sequences of ten consecutive moves to simplify the task of the expert due to the fast-paced nature of this domain.

The feature space of this domain is complex as the different action classes frequently overlap. Figure 4 shows a small sample of the data representing how the agent should drive in the middle lane. The data is split into two regions based on the relative position (in front or behind) of the nearest car in the agent's current lane. No samples appear in the 10 to -10 distance range along the Lane2 axis as the expert avoids collisions that would occur from having another car in such close proximity.

The final model consisted of 34 Gaussian components across three GMMs (one for each action class). The final policy was able to imitate the expert's driving style and navigate well in the complex driving domain. Since the algorithm aims to imitate the behavior of the expert, no 'true' reward function exists to evaluate the performance of a given policy. However, we present two domain-specific evaluation metrics that capture the key characteristics of the driving task.

Since the demonstrated behavior attempts to navigate the domain without collisions, our first evaluation metric is the number of collisions experienced under each policy. Collisions are measured as the percentage of the total timesteps that the agent spends in contact with another car. Always driving straight and colliding with every car in the middle lane results in a 30% collision rate.

Our second evaluation metric is the proportion of the time the agent spends in each lane over the course of a trial. This metric captures the driving preferences of the expert and provides an estimate of the similarity in driving styles. Each evaluation trial was performed for 1000 timesteps over an identical road segment.

Figure 5 compares the performance of the algorithm at different stages in the learning process using these two metrics. Each line in the figure represents a composite bar graph showing the percentage of total time spent by the agent in each lane. Collision percentages for each policy are reported to the right of the bar graphs. The bottom line in the figure shows the performance of the expert over the evaluation road segment (not used for training). We see that the expert successfully avoids collisions, and prefers to use the left three lanes, only rarely using the right lane and right shoulder.

The top five lines summarize the behavior of the agent during the non-interactive training phase. Training was stopped after every 100 training examples for evaluation. Initially the agent always remains in the center lane, accumulating a 30.4% collision rate in the process. As learning progresses, the agent learns to change lanes effectively, beginning to use all five available lanes after 500 demonstration instances, with a collision rate of only 1.3%. However, the agent's lane preference differs significantly from the expert as the agent spends most of its time driving on the right shoulder.

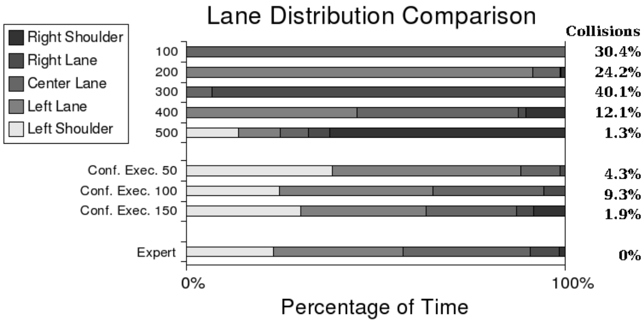

Figure 5. Policy performance comparison using lane distribution and collision evaluation metrics.

The three middle lines display performance during the confident execution phase at 50-timestep intervals. Similarity in lane preference improves over this final training phase, reaching final performance very similar to that of the expert. Additionally, our agent's performance is comparable to that learned using Inverse Reinforcement Learning by Abbeel et al. in [1]. For further evaluation of this domain, including empirical results demonstrating how adapting execution based on confidence focuses training on relevant areas of the domain and a comparison between confident execution and non-interactive demonstration, please see our previous work [10].

4.2 Multiagent Furniture Movers Domain

In this section we present a multiagent collaborative furniture movers domain, Figure 6. In this domain, two agents must move a long, heavy couch from one room to another through a narrow hallway and stairs. We assume that the agents hold opposite ends of the furniture piece throughout this task. Each agent uses six noisy local sensors to determine distances to nearby walls. Additionally, each agent is equipped with a stair sensor that reports a binary value representing the presence or absence of a staircase in the immediate vicinity. The complete feature vector for each agent consists of six continuous distance measurements, and two binary stair features, one for the agent's own location and one for its teammate's. Note that each agent only has a local view of the world, and its teammate's stair information is only updated via a special *communicate* action.

A total of six actions are available to the agents: *forward, back, left, right, communicate*, and *stair*. At each timestep, each agent executes an action based on its own individual policy, and their overall movement is determined by the joint action of both agents. Progress can only be made if the agents select complimentary actions; for example, pulling in opposite directions or attempting to rotate and push at the same time will have no effect on the overall position of the furniture piece. The *communicate* action has no special penalty associated with it, but it does not allow any other action to be activated during that cycle. Since the communicating agent remains stationary for that turn, it prevents any movement regardless of the action taken by the other agent (we assume the couch is too heavy for one agent to move on its own). All movements of the robots are discretized, and the domain can be completed optimally in 39 steps.

The staircase poses a special challenge in this domain, as it requires explicit coordination between the agents. Both agents must select the *stair* action to navigate over the stair segment successfully. However, the corridor is narrow, and the agents are forced to move one after the other instead of side-by-side. As a result, the rear agent is not able to sense when the front agent reaches the staircase. To successfully pass through this region, the front agent must communicate its stair data in order for the rear agent to recognize that the *stair* action is required. Similarly, once the front agent moves past the stairs, the rear agent must communicate its stair information to ensure that the front agent knows to continue executing the *stair* action.

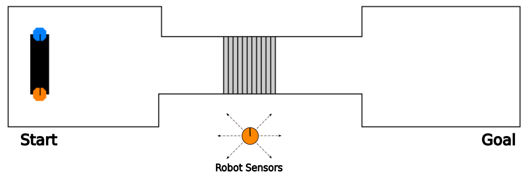

Figure 6. Screenshot of the furniture movers domain. Two agents must collaborate to move a couch from one room to another through a narrow hallway with stairs.

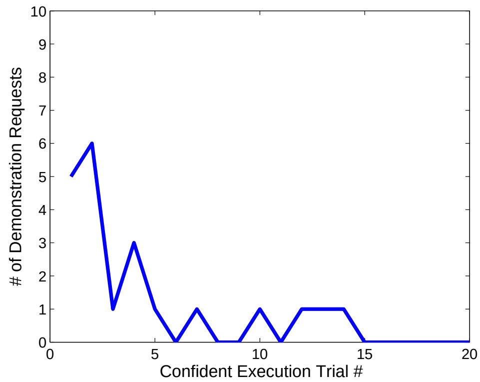

Figure 7. Total number of demonstration requests made by the agents during each trial of the confident execution training phase.

Since a single agent can not perform the task alone, both agents were trained to perform the task at the same time. Demonstrations were performed on an individual basis for each agent. During the confident execution stage, an agent requesting a demonstration waited for the teacher's response, while the other agent was free to continue its execution of the task. Note that in this task, the second agent is not able to make progress on its own due to the constraints of the domain, however, the algorithm places no restrictions upon this agent's actions.

We first evaluate the performance of our learning method using only the non-interactive demonstration technique, in which the agents have no autonomy and the expert performs exhaustive demonstrations of the task. We then present results using the complete confident execution framework. This comparison allows us to evaluate confident execution independently in the context of imitation learning.

Using only the non-interactive demonstration technique, the agents required four demonstrations of the complete domain, or a total of 156 examples per agent, to achieve 100% prediction accuracy and learn the optimal policy. This result confirms that learning from demonstration allows the agents to imitate the behavior of the expert from a small number of examples. Each agent learned its own, unique, policy; the final learned model for each agent consisted of six 8-dimensional Gaussian mixture models.

Confident execution was used to reduce the number of required demonstrations even further by eliminating demonstrations of already acquired behavior. Training was performed using non-interactive demonstration until both models reached 80% prediction accuracy over a window of 15 timesteps, resulting in a total of 65 demonstrations per agent. Under confident execution, the agents continued to perform the task, requesting assistance from the expert at times of uncertainty. Figure 7 shows the total number of demonstration requests made by both agents during each confident execution trial. The number of demonstration requests made decreases with training, until no further requests are made after the 14th learning trial. This resulted in an overall total of 86 demonstrations per agent, approximately half of the number of demonstrations required by the non-interactive method.

Finally, we compare the performance of our algorithm to reinforcement learning. Specifically, Q-learning with a non-deterministic update function was used the learn a policy for each agent. To simplify the task, all action combinations that did not have an effect (such as one agent moving forward, while the other moves back) were not taken into account. This approach was able to learn the optimal policy after 470 iterations, and a total of 58370 exploration steps. Table 2 summarizes the results of all three learning approaches. Note that reinforcement learning performs poorly in this domain because the state of the world is not fully observable as each agent does not know the action taken by its teammate. Partial observability makes this a very challenging problem [22], and a number of special approaches have been developed for dealing with this case [24]. We plan to evaluate and compare these approaches in future work.

Algorithm	# Steps to Learn
Non-Interactive Demonstration	156
Confident Execution	86
Reinforcement Learning	58370

Table 2. Comparison of the number of cycles required to learn the optimal policy in the furniture movers domain.

5 Conclusion

In this paper, we proposed imitation as an alternative to exploration-based methods for learning in multiagent systems. We demonstrated the effectiveness of this approach using our demonstration-based learning framework in a complex simulated multiagent domain. Using our technique, we were able to quickly and accurately train the agents to imitate a human demonstration of the task. Additionally, our results showed that the confident execution approach effectively reduces the workload of the expert, allowing training to scale to a greater number of agents.

REFERENCES

[1] Pieter Abbeel and Andrew Y. Ng, 'Apprenticeship learning via inverse reinforcement learning', in *International Conference on Machine learning*, New York, NY, USA, (2004). ACM Press.

[2] Aris Alissandrakis, Chrystopher L. Nehaniv, and Kerstin Dautenhahn, 'Synchrony and perception in robotic imitation across embodiments', in *IEEE International Symposium on Computatonal Intelligence in Robotics and Automation*, Kobe, Japan, (2003).

[3] Aris Alissandrakis, Chrystopher L. Nehaniv, and Kerstin Dautenhahn, 'Towards robot cultures?', *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, **5**(1), 3–44, (2004).

[4] R.C. Arkin, *Behavior-based robotics*, MIT Press, 1998.

[5] D. C. Bentivegna, C. G. Atkeson, and G. Cheng, 'Learning from observation and practice using primitives', *AAAI Fall Symposium Series, 'Symposium on Real-life Reinforcement Learning'*, (2004).

[6] D. C. Bentivegna, G. Cheng, and C. G. Atkeson, 'Learning from observation and from practice using behavioral primitives', *11th International Symposium of Robotics Research*, (2003).

[7] D. C. Bentivegna, A. Ude, C. G. Atkeson, and G. Cheng, 'Learning to act from observation and practice', *International Journal of Humanoid Robotics*, **1**(4), (2004).

[8] C. Breazeal, G. Hoffman, and A. Lockerd, 'Teaching and working with robots as a collaboration', in *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1030–1037, Washington, DC, USA, (2004). IEEE Computer Society.

[9] Georgios Chalkiadakis and Craig Boutilier, 'Coordination in multiagent reinforcement learning: a bayesian approach', in *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 709–716, New York, NY, USA, (2003). ACM Press.

[10] S. Chernova and M. Veloso, 'Confidence-based policy learning from demonstration using gaussian mixture models', in *Joint Conference on Autonomous Agents and Multi-Agent Systems*, (2007).

[11] Jeffery Allen Clouse, *On integrating apprentice learning and reinforcement learning*, Ph.D. dissertation, University of Massachisetts, Department of Computer Science, 1996. Director-Paul E. Utgoff.

[12] A.P. Dempster, N.M.Laird, and D.B. Rubin, 'Maximum likelihood from incomplete data via the em algorithm', *Journal of Royal Statistical Society*, **8**(1), (1977).

[13] T. Inamura, M. Inaba, and H. Inoue, 'Acquisition of probabilistic behavior decision model based on the interactive teaching method', in *Ninth International Conference on Advanced Robotics (ICAR)*, pp. 523–528, (1999).

[14] L.P. Kaelbling, M.L. Littman, and A.W. Moore, 'Reinforcement learning: A survey', *Journal of Artificial Intelligence Research*, **4**, 237–285, (1996).

[15] A. Lockerd and C. Breazeal, 'Tutelage and socially guided robot learning', in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (2004).

[16] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[17] M. N. Nicolescu and M. J. Mataric, 'Learning and interacting in human-robot domains', in *IEEE Transaction on Systems, Man and Cybernetics*, pp. 419–430, (2001).

[18] M. N. Nicolescu and M. J. Mataric, 'Natural methods for robot task learning: instructive demonstrations, generalization and practice', in *Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 241–248, New York, NY, USA, (2003). ACM Press.

[19] Eugnio Oliveira and Luis Nunes, *Learning by exchanging Advice*, Springer, 2004.

[20] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann.

[21] Bob Price and Craig Boutilier, 'Accelerating reinforcement learning through implicit imitation.', *J. Artif. Intell. Res. (JAIR)*, **19**, 569–629, (2003).

[22] D. Pynadath and M. Tambe, 'Multiagent teamwork: Analyzing the optimality and complexity of key theories and models', in *1st Conference on Autonomous Agents and Multiagent Systems*, (2002).

[23] Patrick Riley, *Coaching: Learning and Using Environment and Agent Models for Advice*, Ph.D. dissertation, Computer Science Dept., Carnegie Mellon University, 2005. CMU-CS-05-100.

[24] Maayan Roth, Reid Simmons, and Manuela Veloso, 'Reasoning about joint beliefs for execution-time communication decisions', in *The Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, (2005).

[25] J. Saunders, C. L. Nehaniv, and K. Dautenhahn, 'Teaching robots by moulding behavior and scaffolding the environment', in *HRI '06: Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pp. 118–125, New York, NY, USA, (2006). ACM Press.

[26] P. Scerri, D. Pynadath, and M. Tambe. Towards adjustable autonomy for the real world, 2003.

[27] M. Sierhuis, J. Bradshaw, A. Acquisti, R. Hoof, R. Jeffers, and A. Uszok. Human-agent teamwork and adjustable autonomy in practice, 2003.

[28] W. D. Smart and L. P. Kaelbling, 'Effective reinforcement learning for mobile robots', in *IEEE International Conference on Robotics and Automation*, (2002).