

Research Article

A Decentralized Partially Observable Decision Model for Recognizing the Multiagent Goal in Simulation Systems

Shiguang Yue,^{1,2} Kristina Yordanova,² Frank Krüger,² Thomas Kirste,² and Yabing Zha¹

¹College of Information System and Management, National University of Defense Technology, Changsha 410073, China

²Institute of Computer Science, University of Rostock, 18051 Rostock, Germany

Correspondence should be addressed to Shiguang Yue; yshg052@hotmail.com

Received 29 February 2016; Revised 17 July 2016; Accepted 24 July 2016

Academic Editor: Lu Zhen

Copyright © 2016 Shiguang Yue et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Multiagent goal recognition is important in many simulation systems. Many of the existing modeling methods need detailed domain knowledge of agents' cooperative behaviors and a training dataset to estimate policies. To solve these problems, we propose a novel decentralized partially observable decision model (Dec-POMDM), which models cooperative behaviors by joint policies. In this compact way, we only focus on the distribution of joint policies. Additionally, a model-free algorithm, cooperative colearning based on Sarsa, is exploited to estimate agents' policies under the assumption of rationality, which makes the training dataset unnecessary. In the inference, considering that the Dec-POMDM is discrete and its state space is large, we implement a marginal filter (MF) under the framework of the Dec-POMDM, where the initial world states and results of actions are uncertain. In the experiments, a new scenario is designed based on the standard predator-prey problem: we increase the number of preys, and our aim is to recognize the real target of predators. Experiment results show that (a) our method recognizes goals well even when they change dynamically; (b) the Dec-POMDM outperforms supervised trained HMMs in terms of precision, recall, and F-measure; and (c) the MF infers goals more efficiently than the particle filter under the framework of the Dec-POMDM.

1. Introduction

With the fast development of computational software and artificial intelligence techniques, agent-based simulation systems become more and more popular for staff training, policy analysis and evaluation, and even entertainments. In developing these systems, people always need to create human-like agents who can make decisions and have interactions with other agents or humans autonomously. For example, in the famous real-time strategy game Star-Craft, the AI players have to construct buildings, collect resources, produce units, and defeat their enemies [1]. Unfortunately, even though many decision and planning algorithms have been applied to improve the intelligence of these agents, they are still easily defeated, especially when human players play with them in the same scenario for several times. One important reason for that is that these agents are unable to recognize the goal of their opponents or friends. On the other hand, that is what the human players usually do in the game [2]. Obviously, if agents

know the goal of others, they can make counter decisions more efficiently.

Because goal recognition is significant for creating human-like agents and decision support, many related models and algorithms have been proposed and applied in different fields, such as hidden Markov models (HMMs) [3], partially observable Markov decision processes (POMDPs) [4], Markov logical networks (MLNs) [5], and particle filtering (PF) [6, 7]. However, most of the existing research focuses on single agent scenarios. However, in some scenarios missions are so complex that a number of agents have to constitute a group and achieve their joint goal through cooperation. And our aim is to identify the joint goal of the group but not one member. In most cases, it does not work to directly apply methods for recognizing the single agent in multiagent goal recognition, because we have to consider the relations or interactions between agents and the state space is usually very large.

There are three fundamental components in the framework of multiagent goal recognition: (a) modeling the agents' behaviors, the environment, and the observations for the recognizer; (b) estimating the parameters of the model obtained through learning or other methods, and (c) inferring the goals from the observations. In the past, people have done some works on all these aspects. However, we still have some difficulties in recognizing multiagent goals in simulation systems:

- (a) For modeling behaviors, we usually have little knowledge about the details of agents' cooperation, such as the decomposition of the complex task, the allocation of the subtasks, the communication, and other details. Even though this information is available, it is hard to present all of them formally in a model in practice.
- (b) For learning parameters, sometimes a training dataset for supervised or unsupervised learning cannot be provided. Even if we have a training set, the unsupervised learning is still infeasible, because the state space in multiagent scenarios is always very large. Additionally, the supervised learning may suffer from the overfitting problem, which will be shown in our experiments.
- (c) For inferring goals, traditional exact filters such as an HMM filter are infeasible because the state space is large. The widely applied PF is available for computing the posterior distribution of goals, but it may fail when there are not sufficient particles, and increasing the number of particles will consume much more computing time.

To solve the problems above, we present a solution for recognizing multiagent goals in simulation systems. The core of our method is a novel decentralized partially observable Markov decision model (Dec-POMDM). After modeling the agents' behaviors, the environment, and the observations for the recognizer by the Dec-POMDM, we use an existing multiagent reinforcement learning (MARL) algorithm to estimate the behavior parameters and a marginal filter (MF) to infer the joint goal of agents. Our method has several advantages considering the above problems:

- (a) For the modeling problem, the Dec-POMDM presents the agents' behaviors in a compact way. The Dec-POMDM can be regarded as an extension of the well-known decentralized partially observable Markov decision process (Dec-POMDP) [8]. As in the Dec-POMDP, all details of cooperation are hidden in joint policies in the Dec-POMDM. In this implicit way of behavior modeling, we only need to concern ourselves with the selection of primitive actions with given goals and situations. Further knowledge on interactions between agents is unnecessary. Another advantage of the Dec-POMDM is that it can make use of the large amount of existing algorithms for the Dec-POMDP, which will be explained later.
- (b) For the problem of estimating the agents' joint policies, the MARL algorithm does not need a training

dataset. We borrow the definition of goals from the domain of planning under uncertainty and associate each goal with a reward function. Then, we assume that agents achieve their joint goal by executing optimal policies, which can bring them the maximum cumulative reward. The optimal policies define cooperative behaviors well and can be computed accurately or approximated by any algorithm for the Dec-POMDP. In this way, the training dataset is unnecessary. In this paper, the cooperative colearning based on the Sarsa algorithm is exploited, because it does not need information of the model, which may be difficult to get in complex scenarios [9]. Actually, heuristic algorithms such as memory-bounded dynamic programming (MBDP) and joint equilibrium-based search for policies (JESP) may also work, if we have enough information of the environment model [10, 11].

- (c) For the inference, the MF outperforms the PF when the state space is discrete and large, which has been proved in [12, 13]. Additionally, we will also show that the MF solves the inference failure problem of the PF in our research. Another contribution is that we implement the MF under the framework of Dec-POMDM, in which filtering process is different from the work in [12, 13].

To validate the Dec-POMDM together with the MARL and MF in multiagent goal recognition, we modify the classic predator-prey problem and design a new scenario [9]. In this scenario, there are more than one prey, and predators have to choose one prey and capture it by moving on a grid map. Additionally, predators may change their goal on the half way. Our method is applied to recognize the real goal of predators based on the observed noisy traces. We also use the simulation model to generate different training datasets, which consist of different numbers of labeled traces. With these datasets, we compare performances of our method and the widely applied hidden Markov model (HMM), whose transition matrix is obtained through supervised learning (the MF is applied doing inference for both models). In the inference part, results computed by the MF are compared to those of PF with different numbers of particles. All performances are evaluated in terms of precision, recall, and F -measure.

The rest of the paper is organized as follows: Section 2 introduces related work. Section 3 gives the formal definition of the Dec-POMDM, the model's DBN representation, the cooperative colearning algorithm, and the baseline used for comparison. Section 4 introduces how to use the MF to infer the joint goal. Section 5 presents the scenarios, settings, and results of our experiments. Subsequently, we draw conclusions and discuss future works in Section 6.

2. Related Work

In many simulation systems such as training systems and commercial digital games, effects of actions are uncertain, which is usually caused by two reasons: (a) agents execute

erroneous actions with a given probability; (b) environment states are impacted by some events, which are not under control of the agents. Because of this uncertainty in simulation systems, a goal cannot be achieved by a certain sequence of primitive actions, as what we do in the classical planning. Instead, we have to find a set of policies, which define the distribution of selecting actions within given situations. Since policies essentially reveal the goal of planning, goals can be inferred as discrete parameters or hidden states, after knowing their corresponding policies, and this process is usually implemented under the Markov decision framework.

2.1. Recognizing Goals of a Single Agent. People have proposed many methods for recognizing goals of a single agent; some of them are foundations of methods for multiagent goal recognition. Baker et al. proposed a computational framework based on Bayesian inverse planning for recognizing mental states such as goals [14]. They assumed that the agent is rational: actions are selected based on an optimal or approximate optimal value function, given the beliefs about the world, and the posterior distribution of goals is computed by Bayesian inference. The core of Baker's method is that policies are computed by the planner based on the standard Markov decision process (MDP), which does not model the observing process of the agent. Thus, Ramirez and Geffner extended Baker's work by applying the goal-POMDP in formalizing the problem [4]. Compared to the MDP, the POMDP models the relation between real world state and observation of the agent explicitly; compared to the POMDP, the goal-POMDP defines the set of goal states. Besides, Ramirez and Geffner also solved the inference problem even when observations are incomplete. Works in [4, 14] are very promising but both of them suffer two limitations: (a) the input for goal recognition is an action sequence; however, sometimes we only have observations of environment states from real or virtual sensors, and translating observations of states to actions is not easy; (b) the goal is estimated as a static parameter; however, it may be interrupted and changed in one episode.

Recently, the computational state space model (CSSM) became more and more popular for human behavior modeling [15, 16]. In the CSSM, transition models of the underlying dynamic system can be described by any computable function using compact algorithmic representations. Krüger et al. also discussed the performances of applying the CSSM on intention recognition in different real scenarios [16, 17]. In this research, (a) intentions as well as actions and environment states are modeled as hidden states, which can be inferred by online filtering algorithm; (b) observations reflect not only primitive actions, but also environment states. The limitation of the research in [17] is that goal inference is not implemented in scenarios where results of actions are uncertain. Another related work on human behavior modeling under the MDP framework was done by Tastan et al. [18]. By making use of the inverse reinforcement learning (IRL) and the PF, they learned the opponent's motion model and tracked it in the game Unreal 2004. This work made

the following contributions: (a) the features for decision in the pursuit problem were abstracted; (b) IRL was used to learn the reward function of the opponent; (c) the solved decision model was regarded as the motion function in the PF. However, IRL relies on a large dataset, and Tastan's method is proposed for tracking but not for goal recognition.

2.2. Multiagent Goal Recognition Based on Inverse Planning. The inverse planning theory can also be used in the multiagent domain. Baker et al. inferred relational goals between agents (such as chasing and fleeing), by using multiagent MDP framework to model interactions between agents and the environment [19]. In this model, each agent selected actions based on the world state, its goal, and its beliefs about other agents' goals. Mental state inference is done by inverse planning, under the assumption that all agents are approximately rational. Ullman et al. also successfully applied this theory in more complex social goals, such as helping and hindering, where an agent's goals depend on the goals of other agents [20]. In the military domain, Riordan et al. borrowed Baker's idea and applied Bayesian inverse planning to inferred intents in multi-Unmanned Aerial Systems (UASs) [21]. Additionally, IRL was also used to learn reward function. Even though Baker's theory is quite promising, it can only work when every agent has accurate knowledge of the world state, because the multiagent MDP does not model the observing process. Besides, Bayesian inverse planning does not allow the goal to change. Another related work under the Markov decision framework in multiagent settings was done by Doshi et al. [22]. Although their main aim is to learn the agents' behavior models, without recognizing goals, the process of estimating mental states is very similar to Bayesian approaches for probabilistic plan recognition. In Doshi's work, the interactive partially observable Markov decision process (I-POMDP) was used to model interactions between agents. I-POMDP is an extension of POMDP for multiagent settings. Comparing to POMDP, I-POMDP defines an interactive state space, which combines the traditional physical state space with explicit models of other agents sharing the environment in order to predict their behavior. Thus, I-POMDP is applicable in situations where agents may have identical or conflicting objectives. However, I-POMDP has to deal with the problem "what do you think that I think that you think," which makes finding optimal or approximately optimal policies very hard [23]. Actually, in many multiagent scenarios such as the football game or the first person shooting game, the agents being recognized share a common goal. This makes the Dec-POMDP framework sufficient for modeling cooperation behaviors. Additionally, the increasing interests/number of works of planning theory based on Dec-POMDP can provide us with a large number of planners [24, 25].

2.3. Multiagent Goal Recognition Based on DBN Filtering. If all actions and world states (the agent and the environment) are defined as variables with time labels, the MDP can be regarded as a special case of directed probabilistic graphical models (PGMs). With this idea, some people ignore the

reward function but only concern themselves with the policies and infer goals under the dynamic Bayesian framework. For example, Saria and Mahadevan presented a theoretical framework for online probabilistic plan recognition in cooperative multiagent systems. This model extends the Abstract Hidden Markov Model and consists of a Hierarchical Multiagent Markov Process that allows reasoning about the interaction among multiple cooperating agents. The Rao-Blackwellized particle filtering (RBPF) is also used for the inference [26, 27]. Pfeffer et al. [28] studied the problem of monitoring goals, team structure, and state in a dynamic environment: an urban warfare field, where uncoordinated or loosely coordinated units attempt to attack a target. They used the standard DBN to model cooperation behaviors (communication, group constitution) and the world states. An extension of the PF named factored particle filtering is also exploited in the inference. We also proposed a Logical Hierarchical Hidden Semi-Markov Model (LHHSMM) to recognize goals as well as cooperation modes of a team in a complex environment, where the observation was noisy and partially missing and the goal was changeable. The LHHSMM is a branch of the Statistical Relational Learning (SRL) method, which combines the PGM and the first order logic; it also presents the team behaviors in a hierarchical way. The inference for the LHHSMM was done by a logical particle filter. These works based on directed PGM theory have the advantage that they can use filtering algorithm. However, they suffer some problems: (a) constructing the graph model needs a lot of domain knowledge, and we have to vary the structure in different applications; (b) the graph structure will be very complex when the number of agents is large, which will make parameter learning and goal inference time consuming, sometimes even infeasible; (c) they need a training dataset. Other models based on data-driven training, such as the Markov logic networks and deep learning, have the same problems listed above [29, 30].

3. The Model

We propose the Dec-POMDM for formalizing the world states, behaviors, and goals in the problem. In this section, we first introduce the formal definition of the Dec-POMDM and explain relations among variables in this model by a DBN representation. Then, the planning algorithm for finding out the policies is given.

3.1. Formalization. One of foundations of our POMDM is the widely applied Dec-POMDP. However, the Dec-POMDP is proposed for solving decision problem, and there is no definition of the goal and the observation model for the recognizer in the Dec-POMDP. Additionally, the multiagent joint goal may be terminated because of achievement or interruption. Thus, we design the Dec-POMDM as a combination of three parts: (a) the standard Dec-POMDP; (b) the joint goal and goal termination variable; (c) the observation model for the recognizer. The Dec-POMDP is the foundation of the Dec-POMDM.

A Dec-POMDM is a tuple $\langle D, S, A, T, R, \Omega, O, \lambda, h, I, G, E, Y, Z, C, B \rangle$, where

- (i) $D = \{1, 2, \dots, n\}$ is the set of n agents;
- (ii) S is a finite set of world states s , which contains all necessary information for making a decision;
- (iii) A is the finite set of joint actions;
- (iv) T is the state transition function;
- (v) R is the reward function;
- (vi) Ω is the finite set of joint observations for agents making a decision;
- (vii) O is the observation probability function for agents making a decision;
- (viii) λ is the discount factor;
- (ix) h is the horizon of the problem;
- (x) I is the initial state distribution at stage $t = 0$;
- (xi) G is the set of possible joint goals;
- (xii) E is the set of goal termination variables;
- (xiii) Y is the observation function for the recognizer;
- (xiv) Y is the finite set of joint observations for the recognizer;
- (xv) Z is the goal selection function;
- (xvi) C is the goal termination function;
- (xvii) B is the initial goal distribution at stage $t = 0$.

Symbols including $D, S, T, \Omega, O, \lambda, h, I$ in the Dec-POMDM have the same meanings as those in the Dec-POMDP. More definition details and explanations can be found in [9, 27]. The reward function is defined as $R : G \times S \times A \times S \rightarrow \mathbb{R}$, which shows that the reward depends on the joint goal; the goal set G consists of all possible joint goals; the goal termination variable set $E = \{0, 1\}$ indicates whether the current goal will be continued in the next step (if $e \in E$ is 0, the goal will be continued; otherwise, a new goal will be selected again in the next step); the observation function for the recognizer is defined as $Y : S \times Y \rightarrow [0, 1]$: $Y(s, y) = p(y | s)$ is the probability that the recognizer observes $y \in Y$ while the real worlds state is $s \in S$; the goal selection function is defined as $Z : S \times G \rightarrow [0, 1]$: $Z(s, goal) = p(goal | s)$ is the conditional probability that agents select $goal \in G$ as the new goal while the world state is s ; the goal termination function is defined as $C : S \times G \rightarrow [0, 1]$: $C(s, goal) = p(e = 1 | s, goal)$ is the conditional probability that agents terminate their $goal \in G$ while the world state is s .

In the Dec-POMDP, the policy of the i th agent is defined as $\Pi^i : \Omega_i^* \times A_i \rightarrow [0, 1]$, where $A_i \in A$ is the set of possible actions of agent i ; Ω_i^* is the set of observation sequences. Thus, given an observation sequence $\{o_1^i, o_2^i, \dots, o_t^i\}$, the agent i selects an action with a probability defined by Π^i . Since the selection of actions depends on the history of the observations, the Dec-POMDP does not satisfy the Markov assumption. This attribute makes inferring goals online very hard: (a) if we precompute policies and store them offline, it will require a very large memory because of the combination

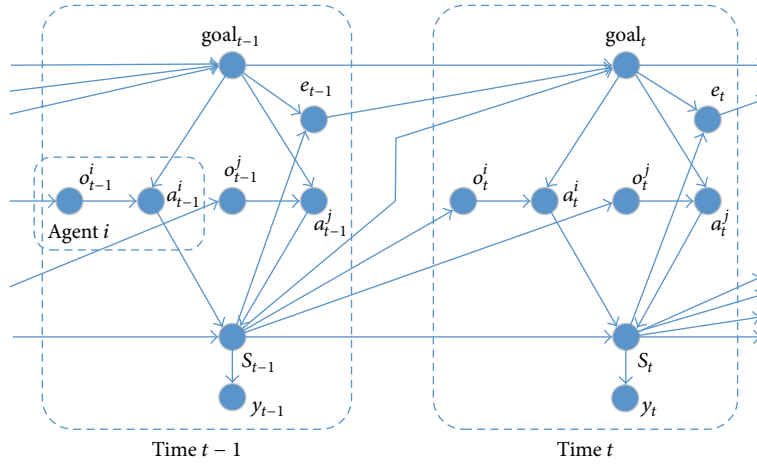


FIGURE 1: The DBN representation of the Dec-POMDM in two adjacent time slices.

of possible observations; (b) if we compute policies online, the filtering algorithm is infeasible because weights of possible states cannot be updated with only the last observation. One possible solution is to define an interior belief variable for each agent to filter the world state, but it will make the inferring process much more complex. In this paper, we simply assume that all agents are memoryless as the work in [9]. Then, the policy of the i th agent is defined as $\Pi^i : G \times \Omega_i \times A_i \rightarrow [0, 1]$, where Ω_i is the set of possible observations of agent i . The definition of policies in the Dec-POMDM shows that (a) an agent does not need the history of observations for making decisions; (b) selection of actions depends on the goal at that stage. In the next section, we will further explain the relations of variables in the Dec-POMDM by its DBN representation.

3.2. The DBN Representation. After estimating the agents' policies by a multiagent reinforcement learning algorithm, we do not need the reward function in the inference process. Thus, in the DBN representation of the Dec-POMDM, there are six sorts of variables in total: the goal, the goal termination, the action, observations for agents making decision, state, and observations for the recognizer. In this section, we first analyze how these variables are affected by other factors. Then, we give the DBN representation in two adjacent time slices.

(A) Goal ($goal$). The goal $goal_t$ at time t depends on the goal $goal_{t-1}$, the goal termination variable e_{t-1} , and the state s_{t-1} . If $e_{t-1} = 0$, agents keep their goal at time t ; otherwise, $goal_t$ is selected depending on s_{t-1} .

(B) Goal Termination Variable (e). The goal termination variable e_t at time t depends on the goal $goal_t$ and state s_t at the same time. The $goal_t$ is terminated with the probability $p(e_t = 1 \mid s_t, goal_t)$.

(C) Action (a). The action a_t^i selected by agent i at time t depends on the goal $goal_t$ and its observation at the same time, and the distribution of actions is defined by Π^i .

(D) Observations for Agents Making Decision (o). The observation o_t^i of agent i at time t reflects the real world state s_{t-1} at time $t-1$, and the agent i observes o_t^i with the probability $p(o_t^i \mid s_{t-1})$.

(E) State (s). The world state s_t at time t depends on the state s_{t-1} at time $t-1$ and the actions of all agents at time t . The distribution of the updated states can be computed by the state transition function T .

(F) Observations for the Recognizer (y). The observation y_t for the recognizer at time t reflects the real world state s_t at the same time, and the recognizer observes y_t with the probability $p(y_t \mid s_t)$.

The DBN representation of the Dec-POMDM in two adjacent time slices presents all dependencies among variables discussed above, as is shown in Figure 1.

In Figure 1, only actions and observations of agent i and agent j are presented for simplicity, and each agent has no knowledge about others and can only make decision based on its own observations. Although the Dec-POMDM has a hierarchical structure, it models the task decomposition and allocation in an inexplicit way: all information about cooperation is hidden in the joint policies. From filtering theory point of view, the joint policies actually play the role of the motion function. Thus, estimating policies is a key problem for goal inference.

3.3. Learning the Policy. Because the Dec-POMDP is essentially a DBN, we can simply use some data-driven methods to learn parameters from a training dataset. However, the training dataset is not always available in some cases. Besides, when the number of agents is large, the DBN structure will become large and complex, which makes supervised or unsupervised learning time consuming. To solve these problems, we assume that the agents to be recognized are rational, which is reasonable when there is no history of agents. Then, we can use an existing planner based on the Dec-POMDP framework to find out the optimal or approximately optimal policies for each possible goal.

Various algorithms have been proposed for solving Dec-POMDP problems. Roughly, these algorithms can be divided into two categories: (a) model-based algorithms, under the general name of dynamic programming; (b) model-free algorithms, under the general name of reinforcement learning [8]. In this paper, we select a multiagent reinforcement learning algorithm, cooperative colearning based on Sarsa [9], because it does not need a state transition function of the world.

The main idea of cooperative colearning is that at each step one chooses a subgroup of agents and updates their policies to optimize the task, given the rest of the agents have fixed plans; then, after a number of iterations, the joint policies can converge to a Nash equilibrium. In this paper, we only consider settings where agents are homogeneous. All agents share the same observation model and policies. Thus, we only need to define one POMDP M for all agents. All parameters of M can be obtained directly from the given Dec-POMDP, except for the transition function T' . Later, we will show how to compute T' from T , which is the transition function of the Dec-POMDP. The Dec-POMDP problem can be solved by the following steps [9].

Step 1. We set $t = 0$ and start from an arbitrary policy Π^0 .

Step 2. We select an arbitrary agent and compute the state transition function T' of M considering that policies of all agents are Π^t , except the selected agent that is refining the plan.

Step 3. We compute Π^* which is the optimal policy of M and set $\Pi^{t+1} = \Pi^*$.

Step 4. We update the policy of each agent to Π^{t+1} , set $t = t+1$, and return to Step 2.

In Step 2, the state transition function of the POMDP for any agent can be computed by formula (1) (we assume that we refine the plan for the first agent):

$$T'(s, a_1, s') = \sum_{(o_2, \dots, o_n)} T(s, a_1, \Pi_t(o^2), \dots, \Pi_t(o^n), s') \cdot \prod_{i=2}^n O'(s, o^n), \quad (1)$$

where O' is the observation function defined in M , a_i is the action of the i th agent, o^i is the observation of the i th agent, $T(s, a_1, \Pi_t(o^2), \dots, \Pi_t(o^n), s')$ is the probability that the state transits from s to s' while the action of the first agent is a_1 , and other agents choose actions based on their observations and policy Π^t .

Unfortunately, computing formula (1) is always very difficult in complex scenarios. Thus, we apply the Sarsa algorithm for finding out the optimal policy of M (in Steps 2 and 3) [8]. In this process, the POMDP problem is mapped to the MDP problem by regarding the observation as the world state. Then, agents get feedback from the environment and we do not need to compute the updated reward function and state transition function.

4. Inference

In the Dec-POMDM, the goal is defined as a hidden state indirectly reflected by observations. In this case, many filtering algorithms can be used to infer the goals. However, in the multiagent setting, the world state space is large because of combinations of agents' states and goals. Besides, the Dec-POMDM models a discrete system. To solve this problem, we use a MF algorithm to infer multiagent joint goals under the framework of the Dec-POMDM. It has been proved efficient when the state space is large and discrete.

Nyolt et al. discussed the theory of the MF and how to apply it in a casual model in [12, 13]. Their main idea can still work in our paper, but there are also differences between the Dec-POMDM and the causal model: (a) the initial distribution of states in our model is uncertain; (b) the results of actions in our model are uncertain; (c) we do not model duration of actions for simplicity. Thus, we have to modify the MF for casual models and make it available for the Dec-POMDM.

When the MF is applied to infer goals under the framework of the Dec-POMDM, the set of particles is defined as $\{x_t^i\}^{i=1:N_t}$, where $x_t^i = \langle goal_t^i, e_t^i, s_t^i \rangle$. N_t is the number of particles at time t , and the weight of i th particle is w_t^i . The detailed procedures of goal inference are given in Algorithm 1.

m_t is the set of $\{x_t^i, w_t^i\}^{i=1:N_t}$, which contains all particles and their weights at each stage. When we initialize $\{x_0^i, w_0^i\}^{i=1:N_0}$, the weights are computed by

$$w_0^i = p(s_0^i | y_0) p(goal_0^i) p(e_0^i | goal_0^i, s_0^i), \quad (2)$$

$p(goal_0^i)$ and $p(e_0^i | goal_0^i, s_0^i)$ are provided by the model, and $p(s_0^i | y_0)$ is computed by

$$p(s_0^i | y_0) \propto p(y_0 | s_0^i) p(s_0^i). \quad (3)$$

map is a temp memory of particles and their weights which are transited from one specific particle. Because a world state s_t may be after the execution of different actions from given s_{t-1} , we have to use a LOOKUP operation to query the record in map , which covers the new particle x_t . The operation LOOKUP(x_t, map) searches x_t in map ; if there is a record $\langle x_t^j, w_t^j \rangle$ in map which covers x_t , the operation returns this record; otherwise, it returns empty. This process is time consuming if we scan the map for every query. One alternative method is to build a matrix for each map , which records the indices of all reached particles. Then, if the index of x_t is null, we add a new record in map and update the index matrix; otherwise, we can read the index of x_t from the matrix and merge the weight directly. After we finish building map , its index matrix can be deleted to release memory. We also need to note that this solution saves computing time but needs extra memory.

The operation PUT($map, \langle x_t, w_t \rangle$) adds a new record $\langle x_t, w_t \rangle$ in map and indexes this new record. The generated map contains a group of particles and corresponding weights. Some of these particles may have been covered in m_t . Thus, we have to use a MERGE operation to get a new m_t by

```

Set  $t = 0, m_0 \leftarrow \{x_0^i, w_0^i\}^{i=1:N_0}$  % Initialization
For  $t = 1 : \max t, m_t \leftarrow \text{Null}$  % We totally have max  $t + 1$  observations
  Foreach  $\{x_{t-1}^i, w_{t-1}^i\} \in m_{t-1}$ 
     $map \leftarrow \text{Null}$ 
    If  $e_{t-1}^i = 1$ 
      Foreach  $goal_t$  which satisfies that  $p(goal_t | s_{t-1}^i) > 0$ 
        Foreach  $\vec{a} = a_1, a_2, \dots, a_n$  which satisfies that  $\prod_{k=1}^n p(a_k | goal_t, s_{t-1}^i) > 0$ 
          Foreach  $s_t$  which satisfies that  $T(s_{t-1}^i, \vec{a}, s_t) > 0$ 
            Foreach  $e_t$  which satisfies that  $p(e_t | s_t, goal_t) > 0$ 
               $w_t = w_{t-1}^i \cdot p(e_t | s_t, goal_t) \cdot T(s_{t-1}^i, \vec{a}, s_t) \cdot \prod_{k=1}^n p(a_k | goal_t, s_{t-1}^i) \cdot p(goal_t | s_{t-1}^i)$ 
               $x_t \leftarrow \{goal_t, s_t, e_t\}$  % A temp memory of a particle
               $\langle x_t^j, w_t^j \rangle \leftarrow \text{LOOKUP}(x_t, map)$  % Find out the record which is equal to  $x_t$  in  $map$ 
              If  $\langle x_t^j, w_t^j \rangle$  is not empty
                 $\langle x_t^j, w_t^j \rangle \leftarrow \langle x_t^j, w_t^j + w_t \rangle$  % Merge the weight
              Else
                 $\text{PUT}(map, \langle x_t, w_t \rangle)$  % Add a new record in  $map$ 
            Else
               $goal_t \leftarrow goal_{t-1}^i$ 
              Foreach  $\vec{a} = a_1, a_2, \dots, a_n$  which satisfies that  $\prod_{k=1}^n p(a_k | goal_t, s_{t-1}^i) > 0$ 
                Foreach  $s_t$  which satisfies that  $T(s_{t-1}^i, \vec{a}, s_t) > 0$ 
                  Foreach  $e_t$  which satisfies that  $p(e_t | s_t, goal_t) > 0$ 
                     $w_t = w_{t-1}^i \cdot p(e_t | s_t, goal_t) \cdot T(s_{t-1}^i, \vec{a}, s_t) \cdot \prod_{k=1}^n p(a_k | goal_t, s_{t-1}^i)$ 
                     $x_t \leftarrow \{goal_t, s_t, e_t\}$  % A temp memory of a particle
                     $\langle x_t^j, w_t^j \rangle \leftarrow \text{LOOKUP}(x_t, map)$  % Find out the record which is equal to  $x_t$  in  $map$ 
                    If  $\langle x_t^j, w_t^j \rangle$  is not empty
                       $\langle x_t^j, w_t^j \rangle \leftarrow \langle x_t^j, w_t^j + w_t \rangle$  % Merge the weight
                    Else
                       $\text{PUT}(map, \langle x_t, w_t \rangle)$  % Add a new record in  $map$ 
                  MERGE( $m_t, map$ )
                  UPDATE( $m_t$ )
                  PRUNE( $m_t$ )
                  NORMALIZE( $m_t$ )
                  DELETE( $m_{t-1}$ )
    
```

ALGORITHM 1: Goal inference based on the marginal filter under the framework of Dec-POMDM.

merging map and the existing m_t . In this process, if a particle x_t in map has not appeared in m_t , we directly put x_t and its corresponding weight w_t into m_t ; otherwise, we need to add w_t into the weight of the record in m_t which covers x_t . Similarly, an index matrix can also be used to save computing time in the MERGE operation.

Under the framework of Dec-POMDM, we update w_t^i by

$$w_t^i = w_{t-1}^i \cdot p(y_t^i | s_t^i), \quad (4)$$

where w_t^i and $x_t^i = \langle goal_t^i, e_t^i, s_t^i \rangle$ are the weight and particle, respectively, i th of record in m_t , and $p(y_t^i | s_t^i)$ is the observation function for the recognizer in the model. The details of PRUNE operation can be found in [13], and we can use the existing pruning technique directly in our paper.

5. Experiments

5.1. The Predator-Prey Problem. The predator-prey problem is a classic problem to evaluate multiagent decision algorithms

[9]. However, it cannot be used directly in this paper because predators have only one possible goal. Thus, we modify the standard problem by setting more than one prey on the map, and our aim is to recognize the real target of the predators at each time. Figure 2 shows the grid-based map in our experiments.

In our scenario, the map consists of 5×5 grids, two predators (red diamonds: Predator PX and Predator PY), and two preys (blue stars: Prey PA and Prey PB). The two predators select one of the preys as their common goal and move around to capture it. As is shown in Figure 2, the observation of the predators is not perfect: a predator only knows the exact position of itself and others which are in the nearest 8 grids. If another agent is out of its observing range, the predator only knows its direction (8 possible directions). For the situation in Figure 2, Predator PX observes that none is near to itself, Prey PB is in the north, Prey PA is in the southeast, and Predator PY is in the south; Predator PY observes that none is near itself, Prey PB and Predator PY are in the north, and Prey PA is in the east. In each step,

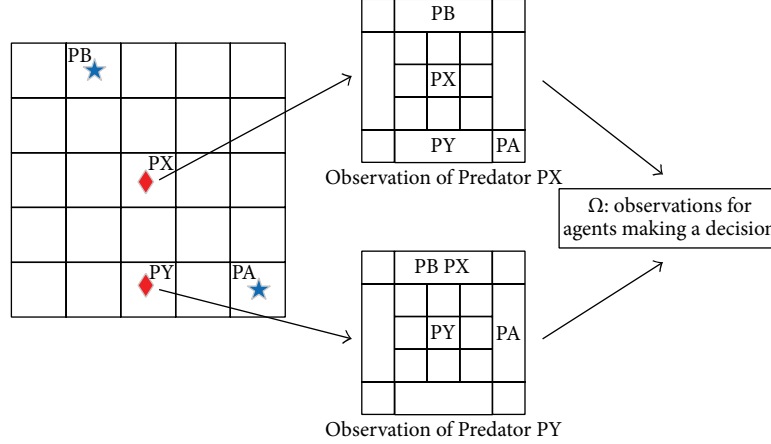


FIGURE 2: The grid-based map in the predator-prey problem.

all predators and preys can get into one of the four adjacent grids (north, south, east, and west) or stay at the current grid. When two or more agents try to get into the same grid or try to get out of the map, they have to stay in the current grid. The predators can achieve their goal if and only if both of them are adjacent to their target. Additionally, the predators may also change their goal before they capture a prey. The recognizer can get the exact positions of the two preys, but its observations of the predators are noisy. We need to compute the posterior distribution of predators' goals with the observation trace.

The modified predator-prey problem can be modeled by the Dec-POMDM. Some important elements are as follows:

- (a) D : the two predators;
- (b) S : the positions of predators and preys;
- (c) A : five actions for each predator, moving into four directions and staying;
- (d) G : Prey PA or Prey PB;
- (e) Ω : the directions of agents far away and the real positions of agents nearby;
- (f) Y : the real positions of preys and the noisy positions of predators;
- (g) R : predators getting a reward +1 once they achieve their goal; otherwise, the immediate reward is 0;
- (h) h : infinite horizons.

With the definition above, the effects of predator's actions are uncertain, and the state transition function depends on the distribution of preys' actions. Thus, actions of preys actually play the role of events in discrete dynamic systems.

5.2. Settings. In this section, we provide additional details for the scenario and the parameters used in the policy learning and inference algorithm.

(A) *The Scenario.* The preys are senseless: they select each action with equal probability. Initial positions of agents are

uniform. The initial goal distribution is that $p(goal_0 = \text{Prey A}) = 0.6$ and $p(goal_0 = \text{Prey B}) = 0.4$.

We simplify the goal termination function in the following way: if predators capture their target, the goal is achieved; otherwise, the predators change their goal with a probability of 0.05 for every state. The observation for the recognizer $y_t \in Y$ reveals the real position of each predator with a probability of 0.5, and the observation may be one of 8 neighbors of the current grid with a probability of 0.5/8. When the predator is on the edge of the map, the observation may be out of the map. The observed results of the agents do not affect each other.

(B) *Multiagent Reinforcement Learning.* The discount factor is $\lambda = 0.8$. The predator selects an action a_i given the observation o_t^i with a probability

$$p(a_i | o_t^i) = \frac{\exp(Q(o_t^i, a_i) / \beta)}{\sum_{i=1}^5 \exp(Q(o_t^i, a_i) / \beta)}, \quad (5)$$

where $\beta = 0.1$ is the Boltzmann temperature. We set $\beta > 0$ as a constant, which means that predators always select approximately optimal actions. In our scenario, the Q -value converges after 750 iterations. In the learning process, if predators cannot achieve their goal in 10000 steps, we will reinitialize their positions and begin next episode.

(C) *The Marginal Filter.* In the MF inference, a particle consists of the joint goal, the goal termination variable, and the positions of predators and preys. Although there are 25 possible positions for each predator or prey, after getting new observation, we can identify the positions of preys and there are only 9 possible positions for each predator. Thus, the number of possible values of particles at one step never exceeds $9 \times 9 \times 2 \times 2 = 324$ after the UPDATE operation. In our experiments, we simply set the max number of particles as 324; then we do not need to prune any possible particles, which means that we exploit an exact inference. We also make use of real settings in the Dec-POMDM and the real policies of predators in the MF inference.

TABLE 1: The details of two traces.

Trace number	Durations	Targets	Interrupted
1	$t \in [1, 7]$	Prey PA	Yes
	$t \in [8, 13]$	Prey PB	No
4	$t \in [1, 10]$	Prey PB	No

5.3. Results and Discussion. With the learned policy, we run the simulation model repeatedly and generated a test dataset consisting of 100 traces. There are 11.83 steps averagely in one trace, and the number of steps in one trace varies from 6 to 34 with a standard deviation of 5.36. We also found that the predators changed their goals for at least once in 35% of the test traces. To validate our method and compare it with baselines, we did experiments on three aspects: (a) to discuss the details of the recognition results obtained with our method, we computed the recognition results of two specific traces by our method; (b) to show the advantages of the Dec-POMDM in modeling, we compared the recognition performances when the problem was modeled as a Dec-POMDM and as an HMM; (c) to show efficiency of the MF under the framework of the Dec-POMDM, we compared the recognizing performances when the goal was inferred by the MF and the PF.

In the second and the third parts, performances were evaluated by statistic metrics: precision, recall, and F -measure. Their meanings and computation details can be found in [31]. The value of the three metrics is between 0 and 1; a higher value means a better performance. Since these metrics can only evaluate the recognition results at a single step, and traces in the dataset had different time lengths, we defined a positive integer k ($k = 1, 2, \dots, 5$). The metric with k means that the corresponding observation sequences are $\{y_{t \in 1: \lceil k * \text{length}^j / 5 \rceil}^{j \in 1: 100}\}$. Here, $y_{t \in 1: \lceil k * \text{length}^j / 5 \rceil}^j$ is the observation sequence from time 1 to time $\lceil k * \text{length}^j / 5 \rceil$ of the j th trace, and length^j is the length of the j th trace. And we need to recognize $goal_t$ for each observation sequence. Thus, metrics with different k show the performances of algorithms in different phases of the simulation. Additionally, we regarded the destination with largest probability as the final recognition result.

(A) The Recognition Results of the Specific Traces. To show the details of the recognition results obtained with our method, we selected two specific traces from the dataset (number 1 and number 4). These two traces were selected because Trace number 1 was the first trace where the goal was changed before it was achieved, and number 4 was the first trace where the goal was kept until it was achieved. The detailed information about the two traces is shown in Table 1.

In Trace number 1, predators selected Prey PA as their first goal from $t = 1$ to $t = 7$. Then, the goal was changed to Prey PB, which was achieved at $t = 13$. In Trace number 4 predators selected Prey PB as their initial goal. This goal was kept until it was achieved at $t = 14$. Given the real policies and other parameters of the Dec-POMDM including T , O , Y , I , Z , C , and B , we used the MF to compute the posterior

distribution of goals at each time. The recognition results are shown in Figure 3.

In Figure 3(a), the probability of the real goal (Prey PA) increases fast during the initial period. At $t = 3$, the probability of Prey PA exceeds 0.9 and keeps a high value until $t = 7$. When the goal is changed at $t = 8$, our method has a very fast response, because predators select highly certain joint actions at this time. In Figure 3(b), the probability of the false goal increases at $t = 2$, and the probability of the real goal (Prey PB) is low at first. The failure happens because the observations support that predators selected joint actions with small probability if the goal is Prey PB. Anyway, the probability of the real goal increases continuously and exceeds that of Prey PA after $t = 5$. With the recognition results of the two specific traces, we conclude that the Dec-POMDM and MF can perform well regardless of the goal change.

(B) Comparison of the Performances of the Dec-POMDM and HMMs. To show the advantages of the Dec-POMDM, we modeled the predator-prey problem as the well-known HMM as a baseline. In the HMM, we set the hidden state x as the tuple $\langle goal, p_X, p_Y, p_A, p_B \rangle$, where p_X , p_Y , p_A , and p_B are positions of Predator PX, Predator PY, Prey PA, and Prey PB, respectively. The observation model for the recognizer in the HMM was the same as that in the Dec-POMDM. Thus, there were $25 \times 24 \times 23 \times 22 \times 2 = 607,200$ possible states in this HMM, and the dimension of the transition matrix was 607200×607200 . Since the state space and transition matrix were too large, an unsupervised learning method such as Balm-Welch algorithm was infeasible in this problem. Instead, we used a simple supervised learning: counting the state transitions based on labeled training datasets. With the real policies of predators, we run the simulation model repeatedly and generated five training datasets. The detailed information of these datasets is shown in Table 2.

The datasets HMM-50, HMM-100a, and HMM-200a were generated in a random and incremental way (HMM-100a contains HMM-50, and HMM-200a contains HMM-100a). Since HMM-100a and HMM-200a both covered 79% of traces of the test dataset, which might cause an overfitting problem, we removed the traces covered by the test dataset in HMM-100a and HMM-200a and compensated them by extra traces. In this way, we got new datasets HMM-100b and HMM-200b which did not cover any trace in the test dataset. With these five labeled datasets, we estimate the transition matrix by counting state transitions. Then, the MF was used to infer the goal. However, in the inference process, we may reach some states that have not been explored in the training dataset (HMM-200a only explores 492,642 states, but there are totally 607,200 possible states). In this case, we assumed that the hidden state would transit to a possible state with a uniform distribution. The rest of the parameters in the HMM inference were the same as those in the Dec-POMDM inference. We compare performances of the Dec-POMDM and the HMMs. The recognition metrics are shown in Figure 4.

Figure 4 shows that comparing the results of the Dec-POMDM and the HMMs trained by different datasets is

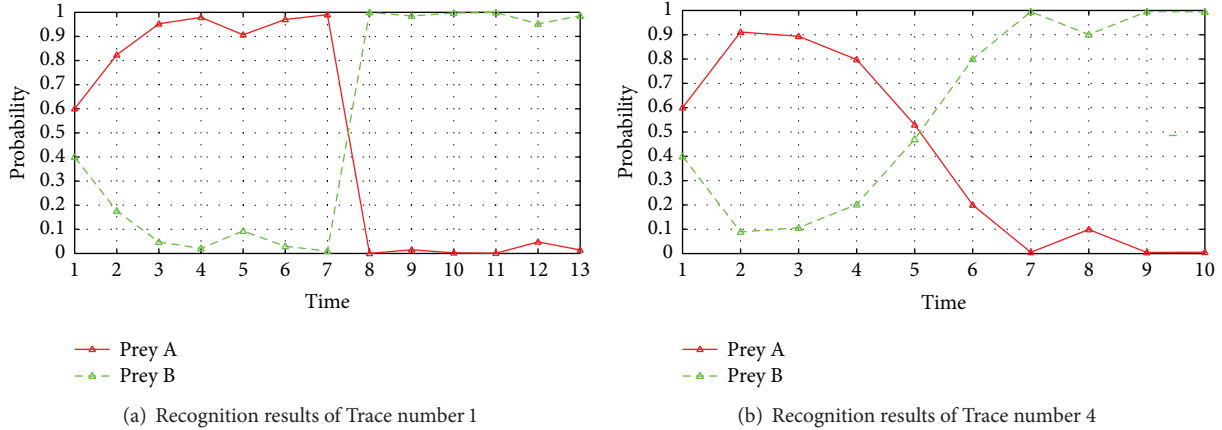


FIGURE 3: Recognition results of two specific traces computed by the Dec-POMDM and MF.

TABLE 2: Information of training datasets for the HMM.

Name of the dataset	Number of traces	Number of explored states	Percentage of traces in the test dataset covered by the training dataset
HMM-50	50,000	279,477	0
HMM-100a	100,000	384,623	79
HMM-100b	100,000	384,621	0
HMM-200a	200,000	492,642	79
HMM-200b	200,000	492,645	0

similar in terms of precision, recall, and F -measure. More precisely, HMM-200a had the highest performance; HMM-100a performed comparable to our Dec-POMDM, but Dec-POMDM performed better after $k = 4$; HMM-50 had the worst performance. Generally, there was no big difference between performances of the Dec-POMDM, HMM-100a, and HMM-200a, even though the number of traces in HMM-200a was twice as large as that in HMM-100a. The main reason is that the training datasets were overfitted. Actually, there was a very serious performance decline after we removed the traces covered in the test dataset from HMM-200a and HMM-100a. In this case, HMM-200b performed better than HMM-100b, but worse than our Dec-POMDM. The results in Figure 4 showed that (a) our Dec-POMDM performed well on three metrics, almost as well as the overfitted trained model; (b) the learned HMM suffered an overfitting problem, and there will be a serious performance decline if the training dataset does not cover most traces in the test dataset.

The curves of HMM-50, HMM-100b, and HMM-200b also showed that when we model the problem through the HMM, it may be possible to improve the recognition performances by increasing the size of the training dataset. However, this solution is infeasible in practice. Actually, the dataset HMM-200a which consisted of 200,000 traces only covered 81.13% of all possible states, and only 71.46% of the states in HMM-200a had been reached more than once. Thus, we can conclude that HMM-200a will have a poor performance if agents select actions with higher uncertainty. Besides, there is no doubt that the size of the training dataset

will be extremely large if most states are reached a large number of times. In real applications, it is very hard to obtain a training dataset with so large size, especially when all traces are labeled.

We also performed a Wald test over the Dec-POMDM and HMMs with a different training dataset to prove that their recognition results came from different distributions. Given our test dataset, there were 100 goals to be recognized for each value of k . Let TX be the set of samples obtained from the Dec-POMDM; then we set $TX_i = 1$ if the recognition result of the Dec-POMDM is correct on the test case i ($i = 1, 2, \dots, 100$) and $TX_i = 0$, otherwise; let TY be the set of samples obtained from the baseline model (one of the HMMs); then we set $TY_i = 1$ if the recognition result of the Dec-POMDM is correct on the test case i and $TY_i = 0$, otherwise. We define $TD_i = TX_i - TY_i$, and let $\delta = \mathbf{E}(TD_i) = \mathbf{E}(TX_i) - \mathbf{E}(TY_i) = P(TX_i = 1) - P(TY_i = 1)$; the null hypothesis is $\delta = 0$, which means the recognition results from different models follow the same distribution. A more detailed test process can be found in [32]. The matrix of p values is shown in Table 3.

The p values in Table 3 show that recognition results of the Dec-POMDM follow different distributions from those of HMM-50, HMM-100b, and HMM-200b, respectively, with a high confidence. We cannot reject the null hypothesis when the baseline is an overfitted trained model. The Wald test results are consistent with the metrics in Figure 4. We also performed the Wilcoxon test, and the test results showed the same trend.

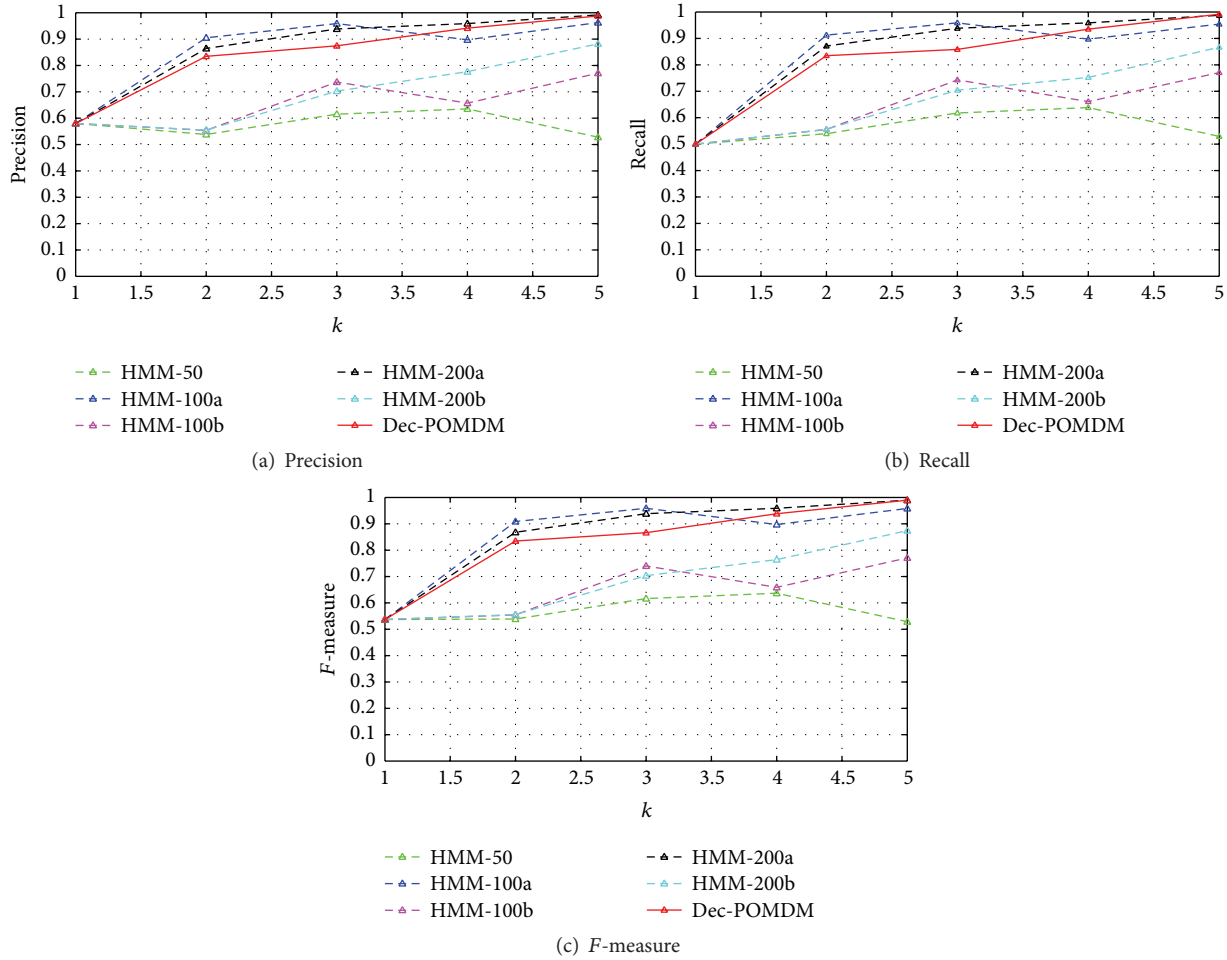


FIGURE 4: Recognition metrics of the Dec-POMDM and HMMs.

TABLE 3: The p values of the Wald test over the Dec-POMDM and HMMs.

	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
Dec-POMDM versus HMM-50	1	<0.01	<0.01	<0.01	<0.01
Dec-POMDM versus HMM-100	1	0.0412	0.0478	0.2450	<0.01
Dec-POMDM versus HMM-100b	1	<0.01	<0.01	<0.01	<0.01
Dec-POMDM versus HMM-200	1	0.3149	0.1929	0.4127	1
Dec-POMDM versus HMM-200b	1	<0.01	<0.01	<0.01	<0.01

(C) Comparison of the Performances of the MF and the PF. Here, we exploited the PF as the baseline. The model information used in the PF is the same as that in the MF. We evaluated the PF with different number of particles and compared their performances to the MF. All inference was done under the framework of Dec-POMDM. We have to note that when the PF used weighted particles to approximate the posterior distribution of the goal, it is possible that all weights decrease to 0 if the number of particles is not large enough. In this case, we simply reset all weights $1/N_p$ to continue the inference, where N_p is the number of particles in the PF. The recognition metrics of the MF and PF are shown in Figure 5.

In Figure 5, the red solid curve indicates the metrics of the MF. The green, blue, purple, black, and cyan dashed curves

indicate the metrics of PF with 1000, 2000, 4000, 6000, and 16000 particles, respectively. All filters had similar precision. However, considering the recall and the F -measure, the MF had the best performance, and the PF with the largest number of particles performed better than the rest of PF. We got these results because an exact MF (without pruning step) is used in this section, and the PF can approximate the real posterior distribution better with more particles.

Similar to the testing method we used in Part (B), here we also performed the Wald test on the MF and PF with different number of particles. The matrix of the p values is shown in Table 4.

The null hypothesis is that the recognition results of the baseline and the MF follow the same distribution. Generally,

TABLE 4: The p values of the Wald test over the MF and PFs.

	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
MF versus PF-1000	1	0.0176	<0.01	<0.01	<0.01
MF versus PF-2000	1	0.3637	0.2450	0.0300	<0.01
MF versus PF-4000	1	1	0.0910	0.0412	0.0115
MF versus PF-6000	1	0.4127	0.1758	0.1758	0.0218
MF versus PF-16000	1	0.5631	0.0412	1	0.1531

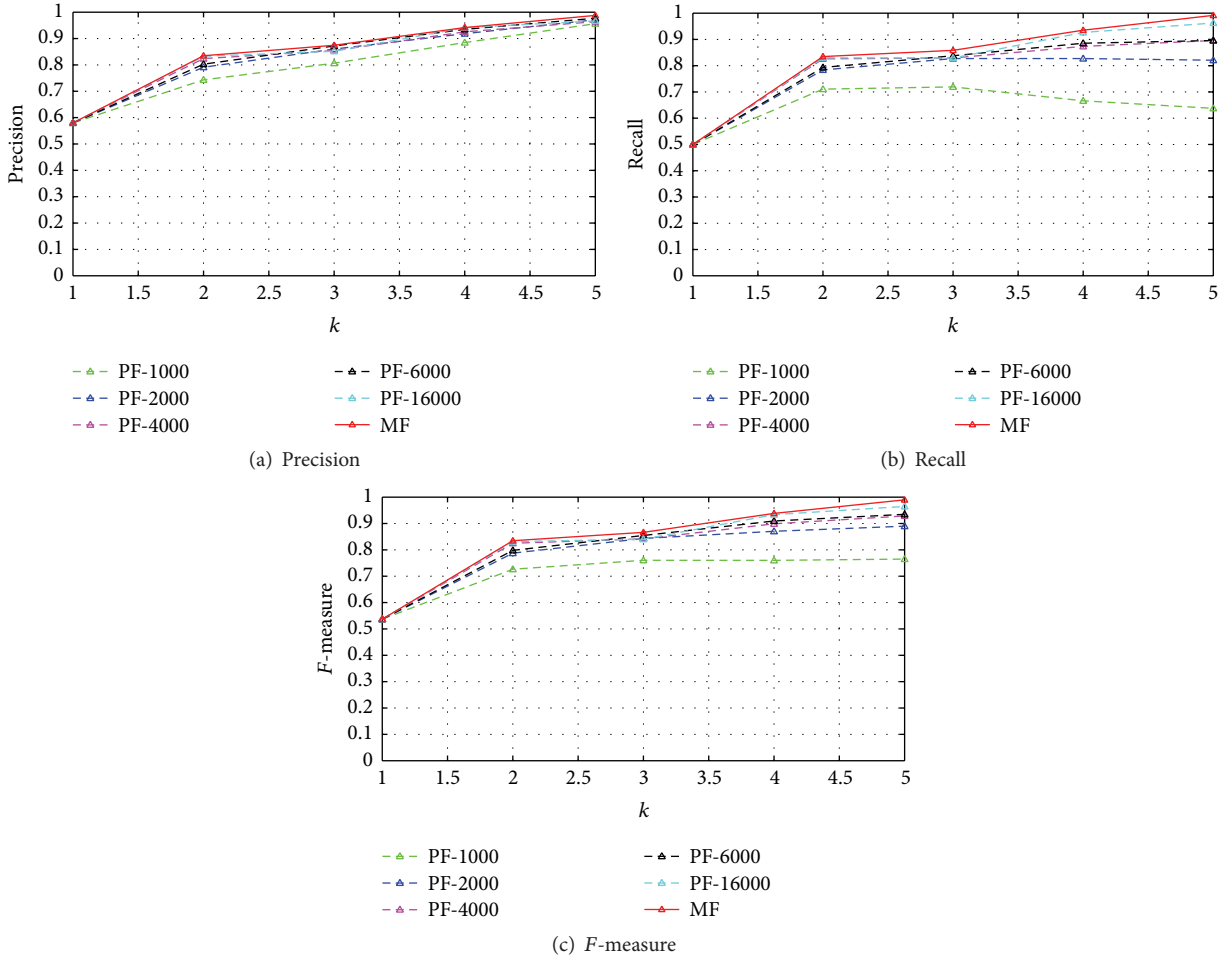


FIGURE 5: Recognition metrics of the MF and PF.

with larger k and smaller number of particles, we can reject the null hypothesis with higher confidence, which is consistent with the results in Figure 5.

Since there was a variance in the results inferred by the PF (this is due to the fact that the PF performs approximate inference), we run the PF with 6000 particles for 10 times. The mean value with 0.90 belief interval of the values of metrics at different k is shown in Figure 6.

The blue solid line indicates the mean value of the PF metrics, and the cyan area shows the 90% belief interval. We need to note that since we do not know the underlying distribution of the metrics, an empirical distribution was used to compute the belief interval. At the same time, because we

run PF for 10 times, the bound of the 90% belief interval also indicates the extremum of PF. We can see that the metrics of the MF are better than the mean of the PF when $k > 1$, and even better than the maximum value of PF except for $k = 4$. Actually, the MF also outperforms 90% of the runs of the PF at $k = 4$. Additionally, the MF only needs average of 75.78% of the time which is needed by the PF for inference at each step. Thus, the MF consumes less time and has a better performance than the PF with 6000 particles.

Generally, the computational complexities of the PF and the MF are both linear functions of number of particles. When there are multiple possible results of an action, the MF consumes more time than the PF when their numbers of

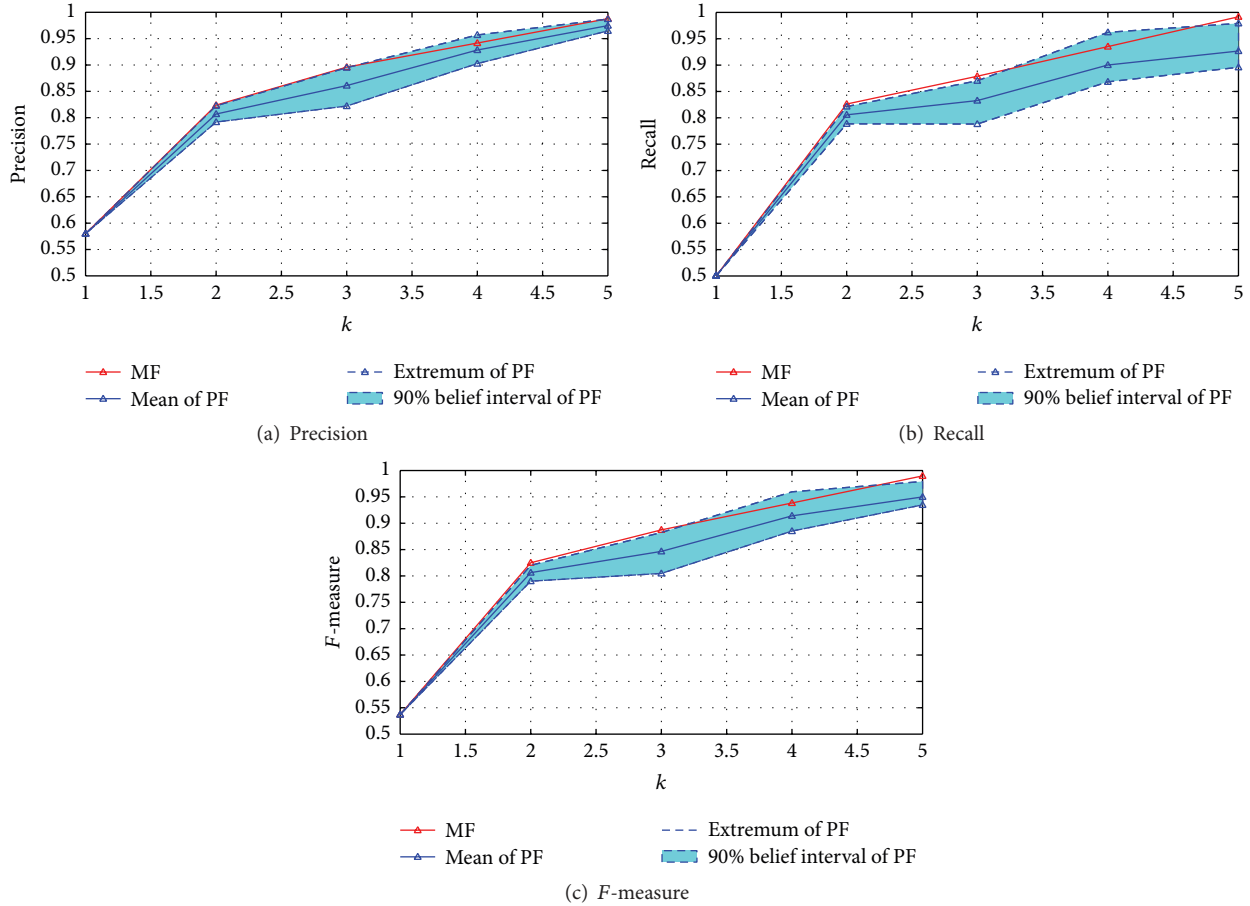


FIGURE 6: Metrics of the PF with 6000 particles and the MF.

particles are equal. However, since the MF does not duplicate particles, it needs much less particles than the PF when the state space is large and discrete. Actually, the number of possible states in the MF after UPDATE operation is never more than 156 in the test dataset. At the same time, the PF has to duplicate particles in the resampling step to approximate the exact distribution, which makes it inefficient under the framework of Dec-POMDM.

6. Conclusion and Future Work

In this paper, we propose a novel model for solving multiagent goal recognition problems of the Dec-POMDM and present its corresponding learning and inference algorithms, which solve a multiagent goal recognition problem.

First, we use the Dec-POMDM to model the general multiagent goal recognition problem. The Dec-POMDM presents the agents' cooperation in a compact way; details of cooperation are unnecessary in the modeling process. It can also make use of existing algorithms for solving the Dec-POMDP problem.

Second, we show that the existing MARL algorithm can be used to estimate the agents' policies in the Dec-POMDM assuming that agents are approximately rational. This method

does not need a training dataset and the state transition function of the environment.

Third, we use the MF to infer goals under the framework of the Dec-POMDM, and we show that the MF is more efficient than the PF when the state space is large and discrete.

Last, we also design a modified predator-prey problem to test our method. In this modified problem, there are multiple possible joint goals and agents may change their goals before they are achieved. With this scenario, we compare our method to other baselines including the HMM and the PF. The experiment results show that the Dec-POMDM together with MARL and MF algorithms can recognize the multiagent goal well whether the goal is changed or not; the Dec-POMDM outperforms the HMM in terms of precision, recall, and F -measure; and the MF can infer goals more efficiently than the PF.

In the future, we plan to apply the Dec-POMDM in more complex scenarios. Further research on pruning technique of the MF is also planned.

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

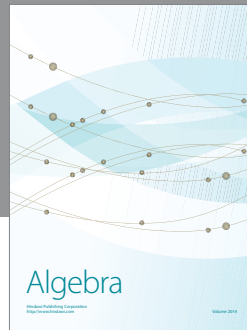

Acknowledgments

This work is sponsored by the National Natural Science Foundation of China under Grants no. 61473300 and no. 61573369 and DFG research training group 1424 “Multimodal Smart Appliance Ensembles for Mobile Applications.”

References

- [1] G. Synnaeve and P. Bessière, “A Bayesian model for plan recognition in RTS games applied to starcraft,” in *Proceedings of the 7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE '11)*, pp. 79–84, Stanford, Calif, USA, October 2011.
- [2] F. Kabanza, P. Bellefeuille, F. Bisson, A. R. Benaskeur, and H. Irandoust, “Opponent behaviour recognition for real-time strategy games,” in *Proceedings of the 24th AAAI Conference on Plan, Activity, and Intent Recognition*, pp. 29–36, July 2010.
- [3] F. Southey, W. Loh, and D. Wilkinson, “Inferring complex agent motions from partial trajectory observations,” in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI '07)*, pp. 2631–2637, January 2007.
- [4] M. Ramirez and H. Geffner, “Goal recognition over POMDPs: inferring the intention of a POMDP agent,” in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI '11)*, pp. 2009–2014, Barcelona, Spain, July 2011.
- [5] E. Y. Ha, J. P. Rowe, B. W. Mott, and J. C. Lester, “Goal recognition with markov logic networks for player-adaptive games,” in *Proceedings of the 7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE '11)*, pp. 32–39, Stanford, Calif, USA, October 2011.
- [6] K. Yordanova, F. Krüger, and T. Kirste, “Context aware approach for activity recognition based on precondition-effect rules,” in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops '12)*, pp. 602–607, IEEE, Lugano, Switzerland, March 2012.
- [7] Q. Yin, S. Yue, Y. Zha, and P. Jiao, “A semi-Markov decision model for recognizing the destination of a maneuvering agent in real time strategy games,” *Mathematical Problems in Engineering*, vol. 2016, Article ID 1907971, 12 pages, 2016.
- [8] M. Wiering and M. van Otterlo, Eds., *Reinforcement Learning: State-of-the-Art*, Springer Science and Business Media, 2012.
- [9] B. Scherrer and C. François, “Cooperative co-learning: a model-based approach for solving multi-agent reinforcement problems,” in *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '02)*, pp. 463–468, Washington, DC, USA, 2002.
- [10] S. Seuken and Z. Shlomo, “Memory-bounded dynamic programming for DEC-POMDPs,” in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI '07)*, pp. 2009–2015, Hyderabad, India, January 2007.
- [11] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella, “Taming decentralized POMDPs: towards efficient policy computation for multiagent settings,” in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI '03)*, pp. 705–711, Acapulco, Mexico, August 2003.
- [12] M. Nyolt, F. Krüger, K. Yordanova, A. Hein, and T. Kirste, “Marginal filtering in large state spaces,” *International Journal of Approximate Reasoning*, vol. 61, pp. 16–32, 2015.
- [13] M. Nyolt and T. Kirste, “On resampling for Bayesian filters in discrete state spaces,” in *Proceedings of the IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI '15)*, pp. 526–533, Vietri sul Mare, Italy, November 2015.
- [14] C. L. Baker, R. Saxe, and J. B. Tenenbaum, “Action understanding as inverse planning,” *Cognition*, vol. 113, no. 3, pp. 329–349, 2009.
- [15] L. M. Hiatt, A. M. Harrison, and J. G. Trafton, “Accommodating human variability in human-robot teams through theory of mind,” in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI '11)*, pp. 2066–2071, July 2011.
- [16] F. Krüger, K. Yordanova, A. Hein, and T. Kirste, “Plan synthesis for probabilistic activity recognition,” in *Proceedings of the 5th International Conference on Agents and Artificial Intelligence (ICAART '13)*, pp. 283–288, Barcelona, Spain, February 2013.
- [17] F. Krüger, M. Nyolt, K. Yordanova, A. Hein, and T. Kirste, “Computational state space models for activity and intention recognition. A feasibility study,” *PLoS ONE*, vol. 9, no. 11, Article ID e109381, 2014.
- [18] B. Tastan, Y. Chang, and G. Sukthankar, “Learning to intercept opponents in first person shooter games,” in *Proceedings of the IEEE International Conference on Computational Intelligence and Games (CIG '12)*, pp. 100–107, IEEE, Granada, Spain, September 2012.
- [19] C. L. Baker, N. D. Goodman, and J. B. Tenenbaum, “Theory-based social goal inference,” in *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, pp. 1447–1452, Austin, Tex, USA, July 2008.
- [20] T. D. Ullman, C. L. Baker, O. Macindoe, O. Evans, N. D. Goodman, and J. B. Tenenbaum, “Help or hinder: bayesian models of social goal inference,” in *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems (NIPS '09)*, pp. 1874–1882, December 2009.
- [21] B. Riordan, S. Brimi, N. Schurr et al., “Inferring user intent with Bayesian inverse planning: making sense of multi-UAS mission management,” in *Proceedings of the 20th Annual Conference on Behavior Representation in Modeling and Simulation (BRiMS '11)*, pp. 49–56, Sundance, Utah, USA, March 2011.
- [22] P. Doshi, X. Qu, and A. Goodie, “Decision-theoretic planning in multi-agent settings with application to behavioral modeling,” in *Plan, Activity, and Intent Recognition: Theory and Practice*, G. Sukthankar, R. P. Goldman, C. Geib et al., Eds., pp. 205–224, Elsevier, Waltham, Mass, USA, 2014.
- [23] F. Wu, *Research on multi-agent system planning problem based on decision theory [Ph.D. thesis]*, University of Science and Technology of China, Hefei, China, 2011.
- [24] F. A. Oliehoek, M. T. J. Spaan, P. Robbel, and J. V. Messias, *The MADP Toolbox 0.3.1*, 2015, <http://www.fransoliehoek.net/docs/MADPToolbox-0.3.1.pdf>.
- [25] M. Liu, C. Amato, X. Liao, L. Carin, and J. P. How, “Stick-breaking policy learning in Dec-POMDPs,” in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI '15)*, pp. 2011–2018, July 2015.
- [26] H. H. Bui, S. Venkatesh, and G. West, “Policy recognition in the abstract hidden Markov model,” *Journal of Artificial Intelligence Research*, vol. 17, pp. 451–499, 2002.
- [27] S. Saria and S. Mahadevan, “Probabilistic plan recognition in multi-agent systems,” in *Proceedings of the 24th International Conference on Automated Planning and Scheduling*, pp. 287–296, June 2004.

- [28] A. Pfeffer, S. Das, D. Lawless, and B. Ng, “Factored reasoning for monitoring dynamic team and goal formation,” *Information Fusion*, vol. 10, no. 1, pp. 99–106, 2009.
- [29] A. Sadilek and H. Kautz, “Location-based reasoning about complex multi-agent behavior,” *Journal of Artificial Intelligence Research*, vol. 43, pp. 87–133, 2012.
- [30] W. Min, E. Y. Ha, J. Rowe, B. Mott, and J. Lester, “Deep learning-based goal recognition in open-ended digital games,” in *Proceedings of the 10th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE '14)*, pp. 37–43, Raleigh, NC, USA, October 2014.
- [31] S.-G. Yue, P. Jiao, Y.-B. Zha, and Q.-J. Yin, “A logical hierarchical hidden semi-Markov model for team intention recognition,” *Discrete Dynamics in Nature and Society*, vol. 2015, Article ID 975951, 19 pages, 2015.
- [32] L. Wasserman, Ed., *All of Statistics: A Concise Course in Statistical Inference*, Springer Science and Business Media, 2013.

Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

