Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 1907971, 12 pages
http://dx.doi.org/10.1155/2016/1907971

Research Article

A Semi-Markov Decision Model for Recognizing the Destination of a Maneuvering Agent in Real Time Strategy Games

Quanjun Yin, Shiguang Yue, Yabing Zha, and Peng Jiao

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

Correspondence should be addressed to Quanjun Yin; yin_quanjun@163.com

Received 19 August 2015; Accepted 16 December 2015

Academic Editor: M. I. Herreros

Copyright © 2016 Quanjun Yin 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.

Recognizing destinations of a maneuvering agent is important in real time strategy games. Because finding path in an uncertain environment is essentially a sequential decision problem, we can model the maneuvering process by the Markov decision process (MDP). However, the MDP does not define an action duration. In this paper, we propose a novel semi-Markov decision model (SMDM). In the SMDM, the destination is regarded as a hidden state, which affects selection of an action; the action is affiliated with a duration variable, which indicates whether the action is completed. We also exploit a Rao-Blackwellised particle filter (RBPF) for inference under the dynamic Bayesian network structure of the SMDM. In experiments, we simulate agents' maneuvering in a combat field and employ agents' traces to evaluate the performance of our method. The results show that the SMDM outperforms another extension of the MDP in terms of precision, recall, and F-measure. Destinations are recognized efficiently by our method no matter whether they are changed or not. Additionally, the RBPF infer destinations with smaller variance and less time than the SPF. The average failure rates of the RBPF are lower when the number of particles is not enough.

1. Introduction

In the recent decades, many commercial real time strategy (RTS) games such as Star Craft and War Craft become more and more popular. A key problem in developing these games is to create AI players who can recognize the intentions of their opponents. Then, the game will be more challenging and interesting [1].

A typical and significant intention in RTS games is the destination of a maneuvering player. In many attacking missions, players need to plan the path with a given destination and the current situation, move along the planned path, and then destroy the building of enemies. Thus, if the AI players can recognize the destination with observed traces of opponents, they can prepare for the defense. Because of these benefits, some recognizing methods have been applied in some digital games. Like the intention recognition, recognizing the destination of a maneuvering agent usually consists of three steps: formalization, parameter estimation, and destination inference [2]. In this paper, we only focus on formalization and inference. The parameters of the opponents' real decision model are directly used. We need to note that these parameters can also be estimated by some machine learning algorithms or simply counting [3].

Hidden Markov models (HMMs) are widely used to model the maneuvering process of an agent. The idea behind HMMs for destination recognition is as follows: regarding the position of the agent as a hidden state, probabilities of transiting between waypoints are modelled by a transition matrix. In other words, a Markov chain represents all possible paths. From the view of planning, HMMs focus on representing the system states but neglect the actions. However, the action determined by the situation affects the state transition matrix very much. Particularly, when the agent is in a dynamic environment, the concept of action is quite important to model the behavior precisely.

In the uncertain planning domain, planning is defined as a sequential decision problem: the agent selects actions sequentially based on the state [4]. Because the action will affect the future state, the agent needs to compute the accumulative reward to get the optimal solution. Solving sequential decision problems is under the framework of Markov decision process (MDP). Comparing to the HMM, a MDP can describe actions of the agent and the intersection

between the agent and the environment. Thus, many models based on the MDP framework are proposed for intention recognition.

In many cases, a complex mission will be decomposed into sublevel tasks repeatedly until the mission only consists of primitive actions. To present decision process hierarchically, Bui et al. [5] proposed an abstract hidden Markov model (AHMM) based on the notion of abstract Markov policies (AMPs), which can be described simply in terms of a state space and a Markov policy that selects among a set of other AMPs. When the AHMM is used in intention recognition, it only concerns the probabilities of selection of a policy or abstract policy and does need to build the reward functions as MDPs. A problem of the AHMM is that it does not allow the top-level policy to be interrupted when the subplan is not completed. To solve this problem, we refined the structure of AHMM and proposed a new model named AHMM with Changeable Top-level Policy (AHMM-CTP) [6]. In AHMM-CTP, the top policies are allowed to be changed and are executed from top to bottom, which means the subplans can be terminated forcedly when the top policies are interrupted. However, the execution of primitive actions of AHMM-CTP is still a MDP, which means a primitive action will be terminated at each step. In the RTS game, a primitive action of path planning is moving for a distance in a direction. Since the simulation step is quite short, the primitive action will keep for several steps. This process is consistent with the semi-Markov decision process (SMDP) in the domain of planning [7].

Based on the idea of the SMDP, we propose a semi-Markov decision model (SMDM) to formalize the maneuvering behaviors in RTS games. The SMDM has a similar structure as a three-layer AHMM-CTP. It models the intention (destination), the action, and the situation hierarchically: actions are selected depending on the intention and situations, and actions result in updated situations. One difference is that, in the SMDM, the primitive action is associated with a duration variable, which indicates whether the primitive action is completed. Obviously, the SMDM can also present multilayer policies as AHMM-CTP, but this complex model is not discussed in this paper.

Inferring destinations online is actually a filtering problem [8]. As an approximate inference method, the particle filter (PF) is suitable to infer the destinations modelled by SMDP, because it does not limit the sort of noise and can tackle partially missing data. Although other existing classic methods are also possible to solve this problem, they will be infeasible when the maximum duration of actions is unknown. Another advantage of the PF is that the costing time only depends on the number of particles. Thus, the time constraint can be satisfied by reducing the number of particles, if the computation resource is not enough. However, when the state space is multidimensional, a small number of particles will result in a very high estimating variance. One solution is to use the Rao-Blackwellised particle filter (RBPF) which combines accurate inference and the Monte Carlo sampling [9]. In the RBPF, some variables of a particle are not sampled but are presented as distributions. And their posterior distribution is computed by accurate inferring after other variables are instantiated. In this way, the state space is declined. The RBPF has been successfully applied in [5, 6]. However, the accurate inference is highly correlated to the dynamic Bayesian network (DBN) structure of the model. To compute the posterior distributions of uninstantiated variables accurately at each step, we exploit the link reversal method based on the DBN structure of the SMDM.

We design a combat scenario to validate our SMDM and the RBPF: on a grid map, a soldier moves to a predefined destination and tries to get far from a patrolling vehicle at the same time. Our goal is to recognize the destination of the soldier with observed traces. Based on this scenario, we design a decision model for the soldier and generate a dataset consisting of 100 traces. With this dataset, statistical metrics including precision, recall, and F-measure are computed by both the SMDM and AHMM-CTP. The results show that the SMDM outperforms the AHMM-CTP in all three metrics. The recognition results of two specific traces are also analyzed, which shows that the SMDM can perform well no matter the soldier's destination is changed on the half or not. We also compare the estimation variance and the costing time of standard particle filter (SPF) and the RBPF. The results show that the RBPF can get results with smaller weighted variance and cost less time at the same time, when the number of particles is large. Additionally, when the number of particles is not enough, the RBPF has a lower average failure rate than the SPF.

The rest of the paper is organized as follows: the next section introduces some related work of formalization and inference. Section 3 analyzes the maneuvering process in the RTS game and gives the formal definition of the SMDM as well as its DBN structure. Section 4 introduces how to use RBPF to infer destinations approximately. Section 5 presents the background, settings, and results of our experiment. Subsequently, we have conclusions and discuss future works in the last section.

2. Related Work

Since the problem of intention recognition or plan recognition was proposed as an intersection of psychology and artificial intelligence, people have used different ways to formalize the planning or the decision process. In early days, the formalization is usually correlated with the conception of plan library. The event hierarchy proposed by Kautz and Allen may be the earliest representation for plan recognition [10]. In Kautz's theory, plans and actions are both defined as events, and an event hierarchy describes abstraction, instantiation, components, and functions by first order logic. Kautz's theory is a milestone in plan recognition, but it may fail when there are two or more hypotheses explaining the observations. To speed up the reason process, Avrahami-Zilberbrand and Kaminka presented a Feature Decision Tree (FDT), which efficiently mapped observations to a plan. In a FDT, nodes correspond to features, and branches correspond to conditions on their values. Since the FDT is actually a special sort of decision tree, it can be built automatically [11]. Another famous model in the automated planning domain

is hierarchical task networks (HTN). A HTN recursively decomposes tasks into lower level components, until a plan constituted by a series of low-level actions or primitive tasks is got [12]. Although the HTN is proposed for auto planning, it can also be used to describe complex tasks in the recognition problem [13]. The common problem of event hierarchy, FDT and HTN, is that they do not use probabilistic theory to model the uncertainty and dynamics in real world or simulation systems and cannot provide us with probabilities of intentions. Actually, other formalization methods which aim to present plans in classic planning theory suffer the same problem.

The well-known Probabilistic Graphical Models (PGMs) use the graph-based representation to compactly encode complex distributions: the nodes correspond to the variables, and the edges correspond to direct probabilistic interactions between them [14]. Since PGMs are very expressive and can provide many effective learning and inferring algorithms, many researchers applied different sorts of PGMs to model the planning or strategy, such as conditional random fields (CRFs) [15], Markov logic networks (MLNs) [16], dynamic Bayesian networks (DBNs) [17], hidden Markov models (HMMs) [18], Markov decision processes (MDPs) [19], and other extensions [20]. Additionally, representations in some plan recognition theories such as parsing trees can also be viewed as special cases of directed PGMs [21].

PGMs have been successfully applied to recognize intentions in many domains. For example, the hidden semi-Markov model (HSMM) is applied to estimate the position of an opponent in the game Counter Strike [22]. The parameters of the model were learnt from a dataset, which consists of 190 game logs collected in a champion-level competition, and both prediction accuracy error and human similarity error are computed to evaluate the estimation performance. Southey et al. also used the HSMM for recognizing the destinations and start point in the RTS game War 3. Besides solving the recognition problem, they further abstract the map and make inference feasible even when number of grids is large and observations are partially missing [8]. Zouaoui-Elloumi et al. used the HMM to model the behaviors of ships in the harbor. Their goal was not to predict the destination or position of the object, but to recognize the behavior pattern of a ship when it enters into the harbor [23]. Duong et al. proposed a Coxian hidden semi-Markov model (CxHSMM) for recognizing human activities of daily living (ADL) [24]. The CxHSMM modifies HMM in two aspects: on one hand, it is a special DBN representation of two-layer HMM, and it also has termination variables; on the other hand, it used Coxian distribution to model the duration of primitive actions explicitly.

van Kasteren et al. further compared the performance of CRF, HMM, SMCRF, and HSMM in the ADL domain, and real data collected in the lab was used [25]. In their experiments the HSMM consistently outperformed the HMM, showing that accurate duration modeling can result in a significant increase in recognition performance. SMCRF only slightly outperformed CRF, showing that CRF was more robust in dealing with violations of the modeling assumptions. Auslander et al. evaluated the performances of the HMM, MLNs, and CRFs for detecting (small boat) maritime attacks. The data was obtained from the 2010 Trident Warrior exercise (Summer 2010) and the results showed that PGMs outperformed the deployed rule-based approach on these tasks [26].

Ullman et al. proposed a model for inferring social goals from peoples' actions, based on inverse planning in multiagent MDPs [27]. In their model, under assuming that agents are rational, the most likely goal which drives observed behaviors is estimated. Tastan et al. presented a framework to predict the positions of the opponent [28]. Unlike the work of Hladky, they used the learned MDP as the motion model of the PF. Another contribution was that they defined tactical features as the states instead of the directly observed data.

In the inference problem, the destination is regarded as a hidden state. Accurate inference algorithms such as HMM filter can be used to compute the posterior probabilities of destinations, but they usually cost much time and need perfect dataset. Another way is approximate inference, and one of the most widely used algorithms is the PF. Besides the works in [8, 22, 28], Weber et al. estimated the location of enemy units that have been encountered in Star Craft based on particle filter. In their work, each single particle, which consists of class, weight, and trajectory, corresponds to one previously encountered enemy unit [29]. Pfeffer et al. used DBNs to represent the tasks that units collaborated to attack targets in an urban environment. They used a factor PF to reason the team composition and goal [30].

3. Modeling Maneuvering by the SMDM

In this section, we discuss how to model the maneuvering process in RTS games by our SMDM. A simple example is used to explain how the agent plans path and moves between grids on a grid-based map. Then, definitions of the SMDM and their corresponding meanings in the maneuvering process are given. At last, we depict the DBN structure of our SMDM to analyze relations among variables.

3.1. Maneuvering in RTS Games. In RTS games, the maneuvering process of an agent consists of two levels: path planning based on the grids and moving between adjacent grids.

(a) Path Planning. No matter the agent is controlled by human or computer, path planning returns a sequence of nodes from the start point to the destination. But in a dynamic environment, path planning is essentially a sequential decision process: moving to an adjacent grid is a primitive action; the agent selects an action based on the current destination and situation. Obviously, the decision results may differ even in the same situation, especially when the agent is controlled by the human. Thus, a probabilistic model is needed to describe the path planning.

(b) Moving between Adjacent Grids. After identifying the oriented adjacent grid, the agent needs to move from the current position to it. This process is totally controlled by an engineering mechanism. Although the moving mechanisms

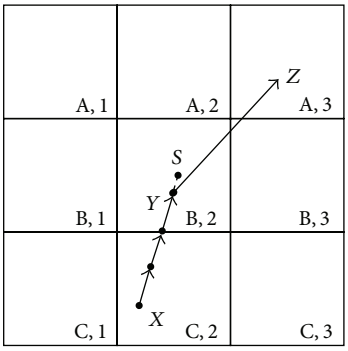


Figure 1: An example of maneuvering process on a grid map.

may differ in different systems, they are certain and can be known by the recognizer. Additionally, since the simulation step is short, the agent usually cannot reach the oriented grid in one step. In other words, the decision process is with a semi-Markov property.

A classic example of maneuvering process on a grid map in the RTS game is presented in Figure 1. Assume that an agent is now on point X in grid C2 and wants to go to point Z in A3. In the path planning level, the agent needs to choose a grid among the five adjacent grids (B1, B2, B3, C1, and C3). In this example, the agent decides to go to grid B2. In the moving level, the agent moves along the line from point X to point S which is the center of grid B2. Because the simulation step is a short time, the agent has to compute how long it will take to reach point S according to the current speed. Because the position of the agent is a continuous variable, it is very unlikely that the agent just gets the grid center when a simulation step ends. Thus, the duration of the moving is usually computed by

$$\text{duration} = \left\lfloor \frac{\|\text{position}_X - \text{position}_S\|}{\text{speed} \times T_{\text{step}}} \right\rfloor, \tag{1}$$

where speed is a constant in the moving process and T_{step} is the real time of a simulation step. $\|\text{position}_X - \text{position}_S\|$ is the distance between point S and point X; duration is computed by a floor operator. In this case, duration = 3. After moving for 3 steps from position X, the agent will get position Y and choose the next grid. This moving process will not be intercepted except that the intention is changed.

3.2. Definition. A MDP models a discrete system, and it assumes that the time between the decision epochs is fixed. However, for the maneuvering process mentioned in Section 3.1, the times of executing the same action are not certain. Thus, we use the SMDP as the basis of our formalization. However, there is no concept of the intention in the SMDP. The SMDP only defines the states which consist of all information needed for making a decision. When the model is used for intention recognition, the states should be further decomposed into inner states and external states, which correspond to the intentions and situations of the environment, respectively. The inner and external states determine selection of actions together. In the rest of the paper, the state is only denoted as the situation. Our SMDM at time t consists of variables as follows:

(a) The intention variable $\pi_t^2 \in \mathbf{\Pi}^2$, where $\mathbf{\Pi}^2 = \{I_1, I_2, \ldots, I_N\}$, is the set of possible intentions and N is the number of possible intentions. In the maneuvering process, each intention corresponds to a destination.

(b) The action variable $\pi_t^1 \in \mathbf{\Pi}^1$, $\mathbf{\Pi}^1 = \{a_1, a_2, \ldots, a_K\}$, is the set of possible actions and K is the number of possible actions. In path planning, we usually have 8 actions corresponding to going to 8 adjacent grids.

(c) The state (situation of the environment) is presented as the variable $s_t \in \mathbf{S}$, and the state can be discrete or continuous.

(d) The intention termination variable $e_t^2 \in \{0, 1\}$ indicates whether the current intention will be terminated.

(e) The action termination variable $e_t^1 \in \{0, 1\}$ indicates whether the current action will be terminated.

(f) The observation variable $o_t \in \mathbf{O}$ is a function of s_t; we can directly get its value by observing. $o_t = \text{Null}$ means that the observation is missing.

(g) The duration variable $d_t \in \{0, 1, 2, \ldots\}$ is a nature number which indicates the number of steps needed to complete the current action.

Besides the above mentioned variables, the SMDM uses functions and parameters to describe the behaviors of the observed object and the system evolution.

(a) Observation function $\text{obs} : \mathbf{S} \times \mathbf{O} \rightarrow [0, 1]$ is $p(o_t \mid s_t)$, which defines the relations between the observation and the real state.

(b) State transition function $T : \mathbf{S} \times \mathbf{\Pi}^1 \times \mathbf{S} \rightarrow [0, 1]$ is $p(s_t \mid s_{t-1}, \pi_t^1)$, which indicates the probability that the system state will transit to s_t from s_{t-1} after executing π_t^1.

(c) Policy of the observed agent is $p(\pi_t^1 \mid s_{t-1}, \pi_t^2)$, which indicates the probability of selecting the action π_t^1 when the previous state is s_{t-1} and the current intention is π_t^2.

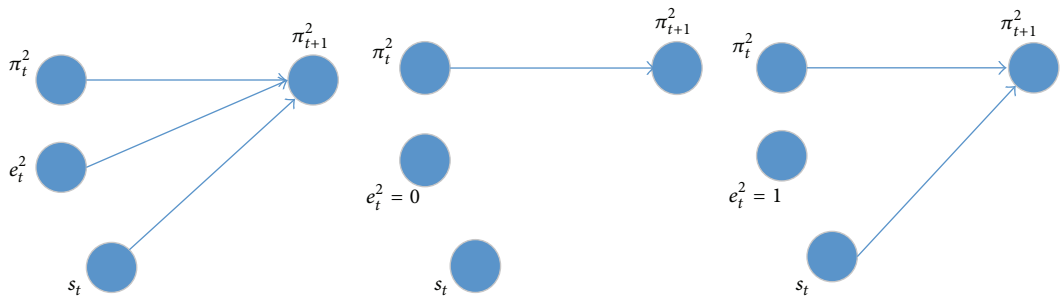
(d) Distribution of the duration $p(d_t \mid s_{t-1}, \pi_t^1)$ gives the probability that it will take d_t steps to complete the action π_t^1 when the previous state is s_{t-1}.

(e) Intention termination probability $p(e_t^2 \mid s_t, \pi_t^2)$ defines the probability of terminating the current intention π_t^2 when the current state is s_t.

(f) Intention transition probability $p(\pi_t^2 \mid \pi_{t-1}^2, s_{t-1})$ indicates the probability that the intention will transit to π_t^2 from π_{t-1}^2, when the previous state is s_{t-1}.

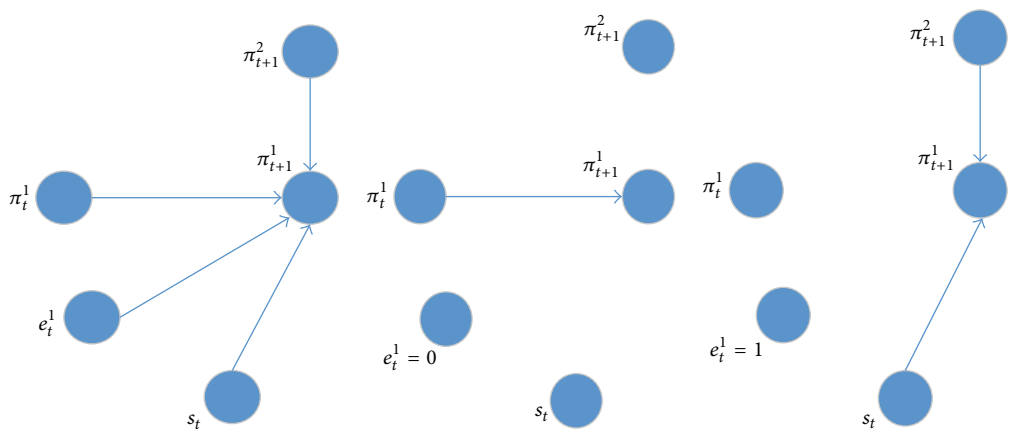
(g) The initial distribution of intentions is presented by symbol p_{intent}.

(h) The initial distribution of states is presented by symbol p_{state}.



(a) Full dependency (b) Dependency when $e_t^2 = 0$ (c) Dependency when $e_t^2 = 1$

Figure 2: Subnetwork for an intention.



(a) Full dependency (b) Dependency when $e_t^1 = 0$ (c) Dependency when $e_t^1 = 1$

Figure 3: Subnetwork for the action.

In this paper, we directly get these parameters and functions from the decision model in systems. However, they can be learned by EM algorithm or other parameter estimation algorithms. When all parameters are known, we can infer the destination online by computing $p(\pi_t^2 \mid o_{1:t})$, where $o_{1:t} = \{o_1, o_2, \ldots, o_t\}$.

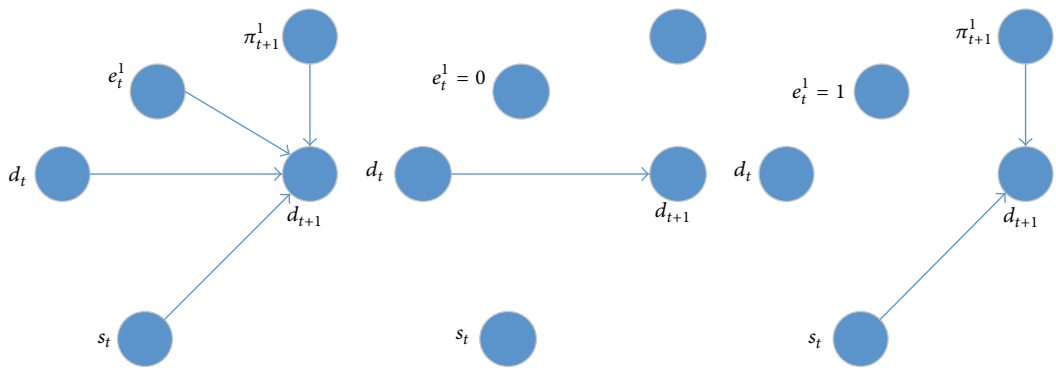
3.3. The DBN Structure. The SMDM is essentially a dynamic Bayesian network, and the DBN structure can depict all casualties in the model. In this section, we first use some subnetworks to explain how elements affect variables of the intention, action, and the duration. Then, the full structure is given.

Figure 2 depicts the subnetwork for the intention. In Figure 2(a), the intention at time $t+1$ is determined by the intention, the intention termination variable, and the state at time t. Figures 2(b) and 2(c) show that if intention π_t^2 is not terminated at time t ($e_t^2 = 0$), we have $\pi_{t+1}^2 = \pi_t^2$; if $e_t^2 = 1$, the agent selects π_{t+1}^2 based on $p(\pi_{t+1}^2 \mid \pi_t^2, s_t)$. In the plan planning, it means that when the agent changes its destination, it selects the next destination according to the previous destination and the situation.

Figure 3 depicts the subnetwork for the action. In Figure 3(a), the action at time $t+1$ is determined by the intention at time $t+1$, the intention termination variable, the action, and the state at time t. Figures 3(b) and 3(c) show that if action π_t^1 is not terminated at time t ($e_t^1 = 0$), we have $\pi_{t+1}^1 = \pi_t^1$; if $e_t^1 = 1$, the selection of π_{t+1}^1 depends on $p(\pi_{t+1}^1 \mid s_t, \pi_{t+1}^2)$. In the plan planning, it means that when the agent finishes its action, it selects the next gird according to the previous situation and the current destination.

Figure 4 depicts the subnetwork for the duration. In Figure 4(a), the duration at time $t+1$ is determined by the action at time $t+1$, the duration, the intention termination variable, and the state at time t. Figures 4(b) and 4(c) show that if action π_t^1 is not terminated at time t ($e_t^1 = 0$), d_{t+1} is determined by d_t according to $p(d_{t+1} \mid d_t)$; if $e_t^1 = 1$, the duration of π_{t+1}^1 will be initialized by $p(d_t \mid s_{t-1}, \pi_t^1)$. In this paper, $p(d_{t+1} \mid d_t)$ can be simply represented by $d_{t+1} = d_t - 1$, because the agent always moves with a certain result in the simulation system. Other dependencies are as follows.

The intention termination variable e_t^2 depends on π_t^2 and s_t with $p(e_t^2 \mid s_t, \pi_t^2)$, which means that the agent changes or keeps his destination based on the current situation. The action termination variable e_t^1 depends on e_t^2 and d_t. We have $e_t^1 = 1$ if $e_t^2 = 1$ or $d_t = 0$, which means the action is terminated by two cases: it is completed or the destination is changed. The state s_t depends on π_t^1 and s_{t-1}, which is the same as the state transition in the MDP.



(a) Full dependency (b) Dependency when $e^1_t = 0$ (c) Dependency when $e^1_t = 1$

Figure 4: Subnetwork for the duration.

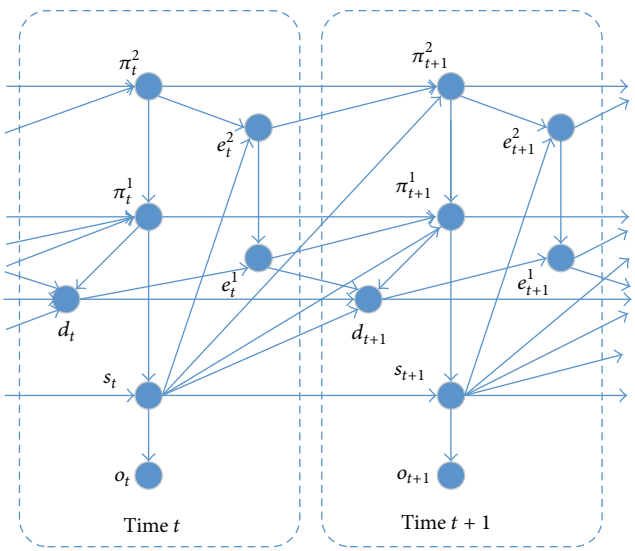


Figure 5: The DBN structure of the SMDM.

Figure 5 depicts the DBN structure of the SMDM in two adjacent time slices. As is shown in the structure, comparing to the SMDP, the SMDM ignores the reward function and focuses on the policies. Additionally, it models the intention separating from states. Comparing to AHMM-CTP, the SMDM models the duration of the primitive action explicitly, which is important to describe the maneuvering process of the player in RTS games. Actually, since the SMDM reflects the general characters of human behaviors, it can also be used in other domains.

4. Inference

Inferring the destination of a maneuvering agent in RTS games is to compute the distribution $p(\pi^2_t \mid o_{1:t})$, which can be got by either accurate inference or approximate inference. For the complex DBN structure, computing $p(\pi^2_t \mid o_{1:t})$ accurately is time consuming and needs perfect observations. However, the system can only provide very limited computation resources for destination recognizing, and the observation is always partially missing because of the war fog. Thus, we use the PF to infer the destinations approximately.

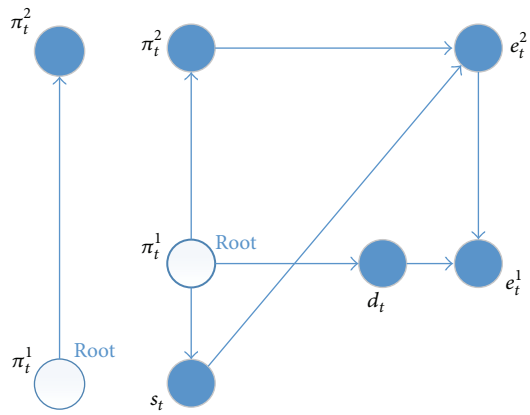
As a widely used state estimation method, the PF does not restrict the type of the system noise and can be applied in nonlinear and non-Gaussian dynamic systems. The PF consists of two steps: sequential importance sampling (SIS) and resampling. The idea of the SIS is to use weighted particles to approximate the posterior distribution of the hidden state. The computation complexity of this process is $O(Np)$ (Np is the number of particles), which means that the cost of time only depends on the number of particles. Since the computation resources are strictly limited in RTS games, the number of particles is small. However, when the state space is high-dimensional as that in the SMDM, the estimation variance will be very high. To solve this problem, we apply the RBPF to infer the destinations.

Comparing to the standard PF (SPF), the RBPF combines the exact filtering and Monte Carlo sampling in the SIS process (the SIS process of RBPF is also called RB-SIS). In the RB-SIS, each particle consists of the Rao-Blackwellizing (RB) variables and the posterior distributions of the uninstantiated variables. The RB-SIS first samples the RB variables. Then, the posterior distributions of the rest of the variables are computed exactly based on the DBN structure. This exact inference can be done by the HMM filter, junction tree algorithm, or any other finite dimensional optimal filters. In this paper, we exploit the link reversal method [5]. Since the DBN structures of the AHMM and SMDM are deferent, the exact inferring process of the SMDM also differs from that of the AHMM.

In this paper, $r_t = \{s_t, d_t, l_t\}$ are defined as the RB variables, where

$$l_t = \begin{cases} 2, & \text{if } e^2_t = 1, \\ 1, & \text{if } e^2_t = 0,\ e^1_t = 1, \\ 0, & \text{if } e^2_t = 0,\ e^1_t = 0. \end{cases} \tag{2}$$

Then, a particle in RB-SIS is defined as $\mathbf{x}_t = \{\Pr(\pi^1_t \mid d_t, s_t, l_t), \Pr(\pi^2_t \mid d_t, s_t, l_t), s_t, d_t, l_t\}$. Before sampling the RB variables, we need to decompose the network in a single time



(a) The DBN structure of C_t

(b) The DBN structure of B_t

Figure 6: The DBN structure of the SMDM in one time slice.

slice. We define $B_t = \Pr(\pi_t^1, \pi_t^2, s_t, d_t, l_t)$, $B_{t+} = \Pr(\pi_t^1, \pi_t^2 \mid s_t, d_t, l_t)$, and $C_t = \Pr(\pi_t^1, \pi_t^2)$. Figure 6 depicts the DBN structure of B_t and C_t.

The root node depicted in Figure 6 is a variable in C_t, which does not depend on any variables in B_t. The initial root is π_t^1 if $l_{t-1} = 0$; otherwise, the initial root is π_t^2. Because C_t is not influenced by e_t^k $(k = 1, 2)$, B_t can be further factorized by

$$\begin{aligned}\Pr\left(\pi_t^2, \pi_t^1, s_t, d_t, e_t^2, e_t^1\right) &= \Pr\left(d_t, e_t^2, e_t^1, s_t \mid \pi_t^2, \pi_t^1\right) \Pr\left(\pi_t^2, \pi_t^1\right) \\ &= \Pr\left(d_t, e_t^2, e_t^1, s_t \mid \pi_t^2, \pi_t^1\right) C_t.\end{aligned} \tag{3}$$

At each step, we can get C_t from the network in the previous time slice. And the exact inference process at time t can be decomposed into 2 steps: (a) sampling RB variables from C_t and $\Pr(d_t, e_t^2, e_t^1, s_t \mid \pi_t^2, \pi_t^1)$, getting B_{t+} from B_t; (b) transiting B_{t+} to C_{t+1} and B_{t+1}. In this paper, we use the simplest sampling method in the RB-SIS. Thus, we do not need to consider o_t in the exact inference. The detailed process of getting B_{t+} is as follows.

Find the root; if the root is π_t^2, reverse the links in C_t and make them from down to top and compute $\Pr(\pi_t^1)$ and $\Pr(s_t)$ by formula (4) to formula (6); otherwise, we only need to compute formula (6) and formula (7):

$$\Pr\left(\pi_t^1\right) = \sum_{\pi_t^2} \Pr\left(\pi_t^1 \mid \pi_t^2\right) \Pr\left(\pi_t^2\right), \tag{4}$$

$$\Pr\left(\pi_t^2 \mid \pi_t^1\right) \propto \Pr\left(\pi_t^1 \mid \pi_t^2\right) \Pr\left(\pi_t^2\right), \tag{5}$$

$$\Pr\left(s_t\right) = \sum_{\pi_t^1} \Pr\left(s_t \mid \pi_t^1\right) \Pr\left(\pi_t^1\right), \tag{6}$$

$$\Pr\left(\pi_t^1 \mid s_t\right) \propto \Pr\left(s_t \mid \pi_t^1\right) \Pr\left(\pi_t^1\right). \tag{7}$$

Sample s_t, and compute $\Pr(\pi_t^2 \mid s_t)$ by

$$\Pr\left(\pi_t^2 \mid s_t\right) = \sum_{\pi_t^1} \Pr\left(\pi_t^2 \mid \pi_t^1\right) \Pr\left(\pi_t^1 \mid s_t\right). \tag{8}$$

Reverse the link between π_t^2 and π_t^1 and make it from top to bottom by

$$\Pr\left(\pi_t^1 \mid \pi_t^2, s_t\right) \propto \Pr\left(\pi_t^2 \mid \pi_t^1\right) \Pr\left(\pi_t^1 \mid s_t\right). \tag{9}$$

Compute the posterior probability $\Pr(e_t^2 \mid s_t)$ by

$$\Pr\left(e_t^2 \mid s_t\right) = \sum_{\pi_t^2} \Pr\left(e_t^2 \mid \pi_t^2\right) \Pr\left(\pi_t^2 \mid s_t\right). \tag{10}$$

Sample e_t^2, and reverse the links between π_t^2 and e_t^2 by

$$\Pr\left(\pi_t^2 \mid e_t^2, s_t\right) \propto \Pr\left(e_t^2 \mid \pi_t^2, s_t\right) \Pr\left(\pi_t^2 \mid s_t\right). \tag{11}$$

Compute the posterior probability $\Pr(\pi_t^1 \mid e_t^2, s_t)$ by

$$\Pr\left(\pi_t^1 \mid e_t^2, s_t\right) = \sum_{\pi_t^2} \Pr\left(\pi_t^1 \mid \pi_t^2, s_t\right) \Pr\left(\pi_t^2 \mid s_t, e_t^2\right). \tag{12}$$

Reverse the link between π_t^2 and π_t^1 and make it from bottom to top by

$$\Pr\left(\pi_t^2 \mid \pi_t^1, s_t, e_t^2\right) \propto \Pr\left(\pi_t^1 \mid \pi_t^2, s_t\right) \Pr\left(\pi_t^2 \mid s_t, e_t^2\right). \tag{13}$$

Reverse the link between d_t and π_t^1 by

$$\begin{aligned}&\Pr\left(\pi_t^1 \mid d_t, s_t, e_t^2\right) \\ &\quad \propto \Pr\left(d_t \mid \pi_t^1, s_t, e_t^2\right) \Pr\left(\pi_t^1 \mid s_t, e_t^2\right).\end{aligned} \tag{14}$$

Compute $\Pr(d_t \mid s_t, e_t^2)$ by

$$\Pr\left(d_t \mid s_t, e_t^2\right) = \sum_{\pi_t^1} \Pr\left(d_t \mid \pi_t^1\right) \Pr\left(\pi_t^1 \mid s_t, e_t^2\right). \tag{15}$$

Sample d_t from $\Pr(d_t \mid s_t, e_t^2)$, and compute $\Pr(\pi_t^2 \mid s_t, e_t^2, d_t)$ by

$$\begin{aligned}&\Pr\left(\pi_t^2 \mid s_t, e_t^2, d_t\right) \\ &\quad = \sum_{\pi_t^1} \Pr\left(\pi_t^2 \mid \pi_t^1, s_t, e_t^2, d_t\right) \Pr\left(\pi_t^1 \mid s_t, e_t^2, d_t\right).\end{aligned} \tag{16}$$

Since e_t^2 and d_t have been both instantiated, $\Pr(\pi_t^2 \mid s_t, e_t^2, d_t) = \Pr(\pi_t^2 \mid s_t, l_t, d_t)$ and $\Pr(\pi_t^1 \mid d_t, s_t, e_t^2) = \Pr(\pi_t^1 \mid d_t, s_t, l_t)$. Then, sample e_t^1 from $\Pr(e_t^1 \mid d_t, s_t, e_t^2)$, and get l_t. After getting B_{t+}, value of l_t is known.

The next step is to get C_{t+1} and B_{t+1}. For C_{t+1}, if $l_t = 0$, $C_{t+1} = \Pr(\pi_{t+1}^2 \mid \pi_{t+1}^1) \cdot \Pr(\pi_{t+1}^1)$, $\Pr(\pi_t^2 \mid \pi_t^1, s_t, e_t^2)$ will be regarded as $\Pr(\pi_{t+1}^2 \mid \pi_{t+1}^1)$, and $\Pr(\pi_t^1 \mid d_t, s_t, e_t^2)$ will also be inherited as $\Pr(\pi_{t+1}^1)$; if $l_t > 0$ and $C_{t+1} = \Pr(\pi_{t+1}^1 \mid \pi_{t+1}^2) \cdot \Pr(\pi_{t+1}^2)$, $\Pr(\pi_{t+1}^1 \mid \pi_{t+1}^2)$ equals $p(\pi_{t+1}^1 \mid s_t, \pi_{t+1}^2)$, and

$\Pr(\pi_{t+1}^2) = \Pr(\pi_t^2 \mid s_t, e_t^2, d_t)$ when $e_t^2 = 0$; $\Pr(\pi_{t+1}^2) = p(\pi_{t+1}^2 \mid \pi_t^2, s_t)$ when $e_t^2 = 1$.

For B_{t+1}, $\Pr(s_{t+1} \mid \pi_{t+1}^1)$ is equal to $p(s_{t+1} \mid s_t, \pi_{t+1}^1)$; other factors are also known as the model parameters. In this way, the ith particle $\mathbf{x}_{t+1}^i$ can be computed from $\mathbf{x}_t^i$.

Since we use the simplest sampling and the observation o_t only depends on s_t, the weight of $\mathbf{x}_t^i$ can be updated by

$$w_t^i = w_{t-1}^i \cdot \Pr(o_t \mid s_t). \tag{17}$$

After exploiting the RB-SIS, a resampling will be also used as the standard PF. The distribution of the possible destinations at time t is computed by

$$\Pr(\pi_t^2 \mid o_{1:t}) = \sum_{i=1}^{N} w_t^i \Pr(\pi_t^2 \mid s_t, l_t, d_t). \tag{18}$$

5. Experiments

5.1. Scenario. To evaluate the performances of the SMDM and RBPF for recognizing the destination of the maneuvering agent in RTS Games, we design a typical game scenario. In this scenario, a vehicle is patrolling randomly on a grid-based map. A soldier departs from an initial position and tries to reach a destination. This scenario is quite common and reflects the uncertainty and dynamics of games. Our goal is to recognize the destination of the soldier with the observed trace. The map is depicted in Figure 7.

The map contains 22×22 grids: the red diamond point is the initial position of the soldier; the black grid indicates buildings which the agent cannot get through; the four green grids are possible destinations (from Destination 1 to Destination 4); the white girds make up passable ways for motion; the blue star is the initial position of the patrolling vehicle; the blue dashed lines are possible patrolling ways for the vehicle. In each step, the soldier moves for a distance in the way explained in Section 3.1. We need to note that when the soldier selects a grid, it tries to approach its destination and gets far away from the patrolling vehicle at the same time. Additionally, the soldier may change its destination on the half way. For the patrolling vehicle, it reaches the center of one adjacent grid which is on the possible way in one step. The vehicle cannot turn around but can choose a direction with equal probabilities when it reaches a cross. For simplification, the soldier has no influence on the maneuvering of the vehicle. This assumption is reasonable because the patrolling vehicle only needs to report to the base in many cases.

5.2. Settings. To generate a test dataset automatically, we design a decision model for the soldier. The policy of selecting grids is as follows: $\mathbf{U} = \{u_i\}^{i=1:8}$ is a utility vector, where u_i is the utility of reaching the ith adjacent grid. If the soldier cannot reach the ith grid in one step, $u_i = 0$; otherwise, $u_i = 1/(r_1+0.1) - 1/(r_2+0.1)$, where r_1 is the shortest distance between the center of the ith grid and the destination and r_2 is the shortest distance between the center of the ith grid and the patrolling vehicle. These two distances are computed by

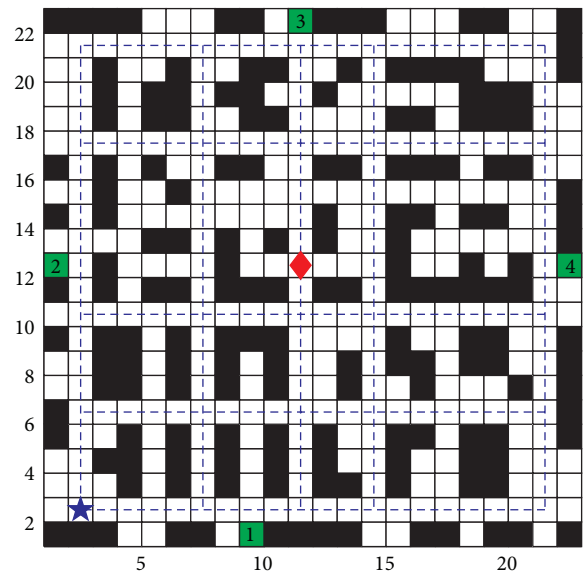


Figure 7: The grid-based map of the game scenario.

an A* algorithm. Obviously, the soldier prefers to select a grid with the large r_2 and the small r_1, but the decision result is uncertain. We define that the ith adjacent grid will be selected with a probability p_i, which is computed by

$$p_i = \frac{\exp(u_i/T)}{\sum_{i=1}^{8} \exp(u_i/T)}, \tag{19}$$

where $T = 0.1$ is the Boltzman temperature.

The soldier may change its destination on the half way. $p(e_t^2 \mid s_t, \pi_t^2)$ is defined as follows: if $r_2 \geq 8$, the current destination will be changed with a probability 0.05 for any π_t^2; otherwise, the destination will not be changed.

The accurate position of the soldier is not available. We can only know the grid where the soldier is, and the observations are got sequentially. Additionally, 10% of observations are missing in each trace.

The prior distribution of the four destinations is $[0.225, 0.225, 0.25, 0.3]$. The length of a side of the grid is 1. The moving speed is 0.3 per step. We simplify $p(\pi_t^2 \mid \pi_{t-1}^2, s_{t-1})$ as $p(\pi_t^2 \mid \pi_{t-1}^2)$ in this scenario; the transition matrix is as in Table 1.

5.3. Results and Discussion. With the parameters provided in Section 5.2, we generate a dataset consisting of 100 traces to test the SMDM and RBPF automatically. To prove the effectiveness of the SMDM, precision, recall, and F-measure of the recognition results are three statistical metrics computed by both the SMDM and AHMM-CTP. The AHMM-CTP can be simply regarded as the SMDM without duration modeling. In the AHMM-CTP, the action termination variable is always 1 and the agent needs to make a decision based on the intention and situation in each step. Because the AHMM-CTP cannot represent the maneuvering process in the RTS game accurately, the weights of particles may all become zero

Table 1: The transition matrix of the destinations.

π^2_{t-1}	π^2_t Destination 1	Destination 2	Destination 3	Destination 4
Destination 1	0	0.3	0.3	0.4
Destination 2	0.3	0	0.3	0.4
Destination 3	0.3	0.3	0	0.4
Destination 4	0.3	0.3	0.4	0

in some cases. To continue the inferring process of AHMM-CTP, we will set all weights $1/Np$ forcedly when the weights are all zero, where Np is the number of particles.

We use precision, recall, and F-measure to evaluate the recognition performance. Their meanings and computation details can be found in [25]. For the three metrics, the value of them 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 have different time lengths, we define a positive integer $k \in [1, 10]$. The metric with k means that the corresponding recognizing object set is $\{\text{object}^{j=1:100}_{t=\lceil k*\text{length}^j/10\rceil}\}$, where length^j is the length of the jth trace. Thus, metrics with different k show the performances of algorithms in different phases of simulation. Additionally, we regard the destination with largest probability as the final recognition result. Figure 8 shows the precision, recall, and F-measure of the recognition results computed by the SMDM and AHMM-CTP.

In Figure 8, it is obvious that the performance of the SMDM is better when we have more observations. Specifically, all the three metrics of the SMDM exceed 0.85 when we have observed the first half of traces. However, AHMM-CTP does not perform well in all three metrics. The main reason is that it does not model the duration of an action. Since the soldier does not select grid in each step in our scenario, the filtering process of the AHMM-CTP will fail in many traces.

To show detailed recognition results computed by the SMDM and RBPF, select two traces in the dataset. In Trace A, the real intention is Destination 3, and the soldier does not change the destination on the half way. In Trace B, the real intention is Destination 3 before $t = 9$; then, the intention is changed to Destination 4. The particle number is 1000, and the probabilities of destinations of two traces in each step are depicted in Figure 9.

In Trace A, the probability of the real destination increases very fast; it nearly reaches 0.9 at $t = 11$ and keeps very large until the end. In Trace B, the probability of the real destination is the largest before $t = 9$. When the intention is changed, the model responds very fast. We can also find that the probability of Destination 4 is not the largest before $t = 18$; the reason is that the soldier may move in the same direction for all the three destinations in this period. Anyway, the probability of the real destination keeps the largest after $t = 18$.

We can get the results in Figure 9 by both the SPF and RBPF. However, their variances of the results are not the same. In this paper, we use the weighted variance to compare the performances of the SPF and RBPF. The weighted variance at time t is computed by

$$\text{Var}_t = \sum_{i=1}^{n} w_t^i \left(d_t^i - \widehat{d}_t\right)\left(d_t^i - \widehat{d}_t\right)^T, \tag{20}$$

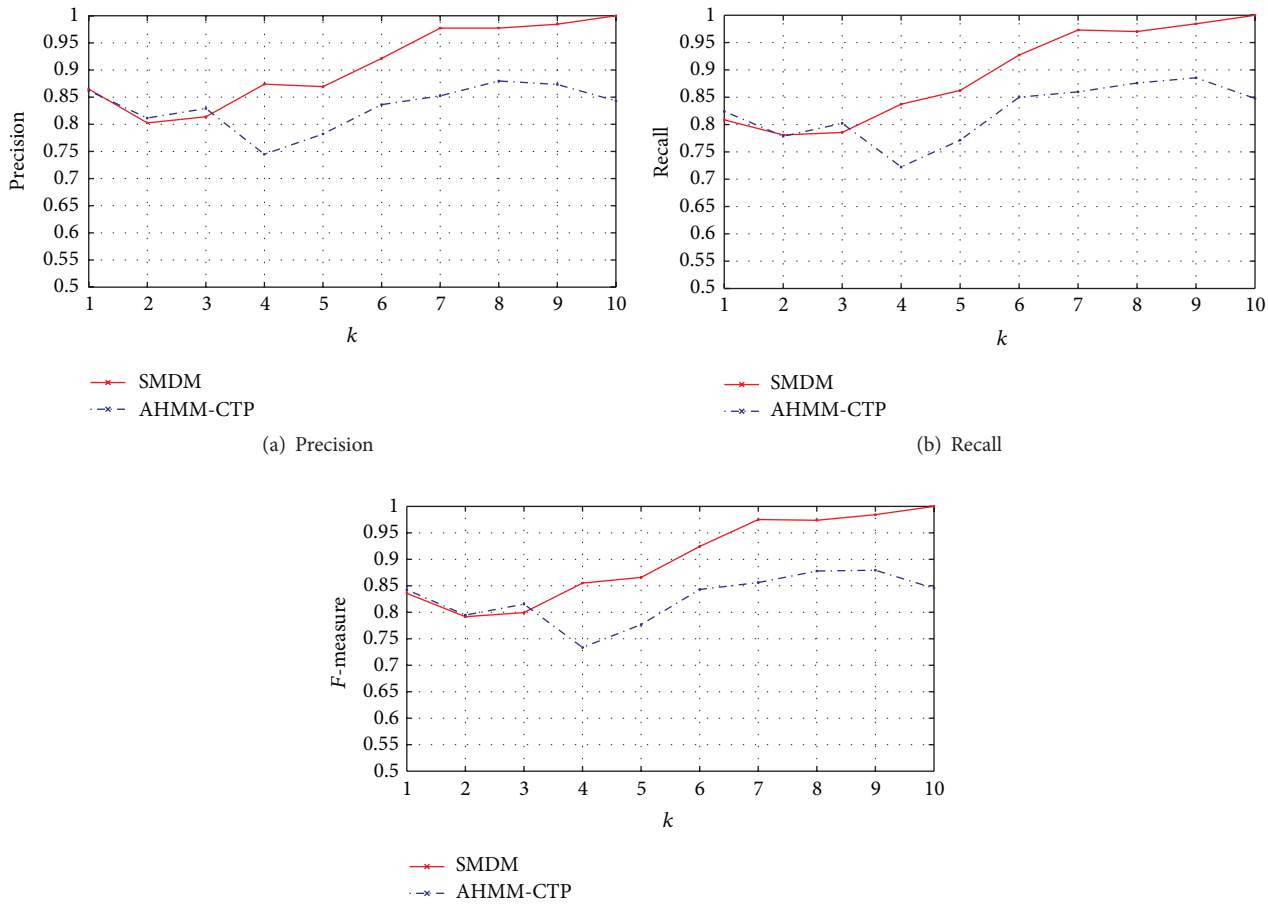
where w_t^i is the weight of the particle $\mathbf{x}_t^i$, $\widehat{d}_t$ is the estimated distribution of destinations, and d_t^i is the distribution of destinations in $\mathbf{x}_t^i$. In the RBPF, d_t^i is continuous four-dimension vector. But in the SPF, since destinations in SPF are instantiated, only one element in d_t^i is one and others are zeros. Figure 10 depicts the weighed variances of the results in Figures 9(a) and 9(b), respectively.

The difference between red solid line and the blue dashed line is that the former exploits 1000 particles and the latter exploits 500 particles. Figures 10(a) and 10(b) both show that the variance of the SPF is very high and it converges much slower than that of the RBPF. Additionally, the number of particles is not sensitive to the variances of the RBPF. It can get good performance with a few particles. To further compare the computation efficiency of the RBPF and SPF, their cost of time with different numbers of particles is also shown in Figure 11.

The program is written in Matlab script and is run in our computer which has a Pentium E5500 CPU (2.8 GHz) and 2 GB memory. The red solid line and black dashed line represent the cost of time of the RBPF and SPF, respectively. We can find that their cost of time will increase linearly with the number of particles. The RBPF costs time less than twice as much as that of the SPF when the numbers of particles are the same. However, we need to note that time of RBPF with 500 particles is less than that of SPF with 1000 particles. Combining with variance in Figure 10, we can conclude that the RBPF can get smaller variances than the SPF and consumes less time than the SPF at the same time.

When $Np < 500$, since the particles are not enough, both the SPF and RBPF may fail with a large probability. We run our dataset by the RBPF and SPF for 20 times, and average failure rates with different numbers of particles are shown in Table 2.

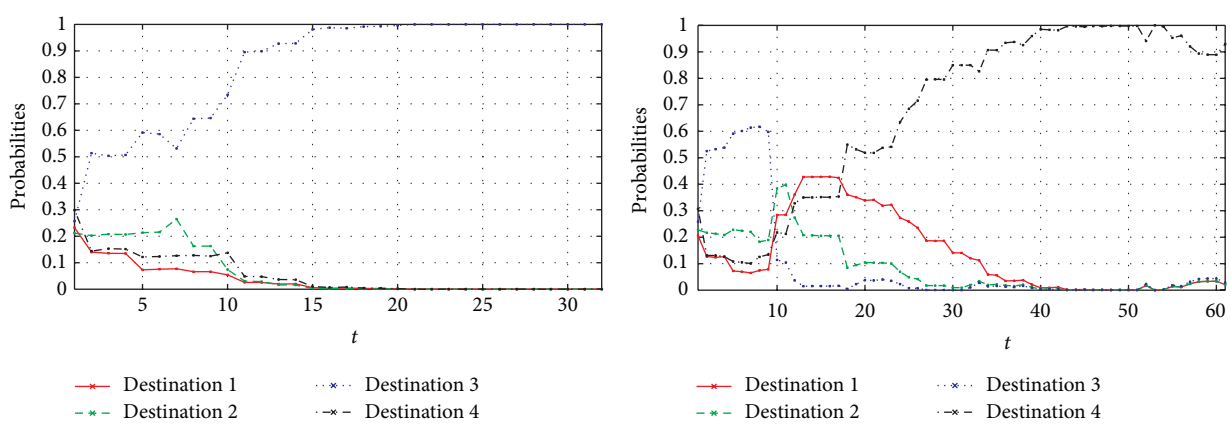
Obviously, the average failure rates of the RBPF are lower than those of the SPF. The main reason is that the destination distribution is continuous in the RBPF, and the probabilities may be very small but usually are not zeros. Thus, fewer of particles in the RBPF have zero weights.



(a) Precision (b) Recall

(c) F-measure

Figure 8: Three metrics of the recognition results computed by the SMDM and AHMM-CTP, the red solid and blue dashed lines are computed by the SMDM and AHMM-CTP, respectively.

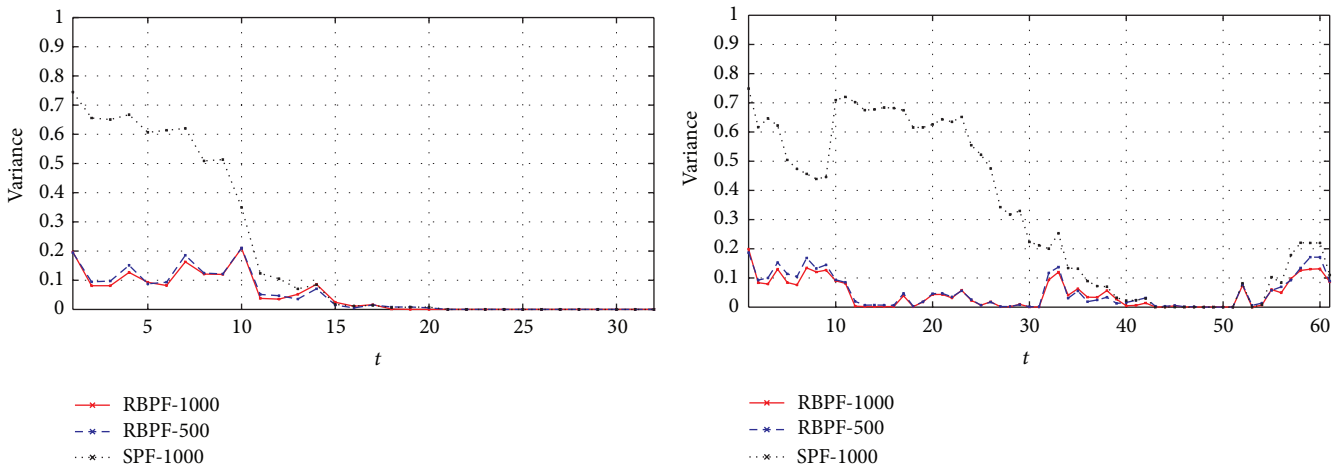


(a) Trace A (b) Trace B

Figure 9: The probabilities of destinations of two traces at each step, the red solid, green dashed, blue dotted, and black dash-dotted lines are probabilities of destinations from 1 to 4, respectively.

Table 2: The average failure rates with different numbers of particles.

Number of particles	50	100	200	300	400
Average failure rates of the RBPF	23.70%	13.50%	7.20%	4.45%	3.15%
Average failure rates of the SPF	31.15%	16.75%	8.25%	5.35%	4.30%



(a) Trace A (b) Trace B

Figure 10: The variances of recognition results of two traces at each step, the red solid line and the blue dashed line are both computed by the RBPF. The black dotted line is computed by the SPF with 1000 particles.

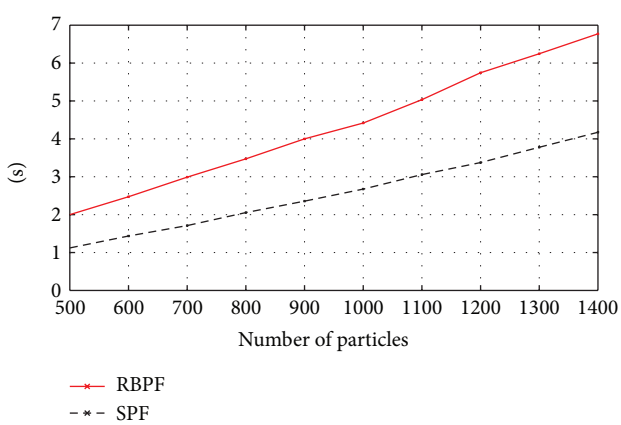


Figure 11: The cost of time by the RBPF and SPF with different numbers of particles.

6. Conclusion and Future Works

In this paper, we propose a semi-Markov decision model (SMDM) to recognize the destination of a maneuvering agent in RTS games. To decline the estimation variance, the RBPF is used to compute the distribution of the destination online. The results show that the SMDM can recognize the destination efficiently no matter the intention changes or not, and it performs better than the AHMM-CTP in terms of precision, recall, and F-measure. When the number of particles is large, the RBPF can get results with smaller variance and cost less time than the SPF; when the number of particles is not enough, the average failure rates of the RBPF are lower than those of the SPF.

In the future work, we plan to use the optimal sampling and compare the computation efficiency of link reversal and other exact inferring methods in the RB-SIS. Additionally, we are also interested to apply another recent family of hidden semi-Markov models with finite set of durations for recognizing destinations, which may allow fast exact filtering in our scenario [31].

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is sponsored by the National Natural Science Foundation of China under Grants no. 61473300 and no. 61573369.

References

[1] S. C. J. Bakkes, P. H. M. Spronck, and G. van Lankveld, "Player behavioural modelling for video games," *Entertainment Computing*, vol. 3, no. 3, pp. 71–79, 2012.

[2] S. Raghavana, P. Singlab, and R. J. Mooney, "Plan recognition using statistical-relational models," in *Plan, Activity, and Intent Recognition: Theory and Practice*, G. Sukthankar, R. P. Goldman,

C. Geib et al., Eds., pp. 57–85, Elsevier, Waltham, Mass, USA, 2014.

[3] G. Robertson and I. Watson, "A review of real-time strategy game AI," *AI Magazine*, vol. 35, no. 4, pp. 75–104, 2014.

[4] S. Ragi, C. Tan, and E. K. Chong, "Guidance of autonomous amphibious vehicles for flood rescue support," *Mathematical Problems in Engineering*, vol. 2013, Article ID 528162, 9 pages, 2013.

[5] 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.

[6] S. G. Yue, Y. B. Zha, Q. J. Yin, and Q. Zhang, "Path plan recognition by CGF based on abstract hidden Markov model," *Journal of National University of Defense Technology*, vol. 36, no. 1, pp. 148–153, 2014.

[7] M. Wiering and M. van Otterlo, *Reinforcement Learning: State-of-the-Art*, Springer, Berlin, Germany, 2011.

[8] 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.

[9] K. Murphy and S. Russell, "Rao-blackwellised particle filtering for dynamic Bayesian networks," in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. J. Gordon, Eds., Springer, 2001.

[10] H. A. Kautz and J. F. Allen, "Generalized plan recognition," in *Proceedings of the 5th National Conference on Artificial Intelligence*, pp. 32–37, Philadelphia, Pa, USA, 1986.

[11] D. Avrahami-Zilberbrand and G. A. Kaminka, "Fast and complete symbolic plan recognition," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI '05)*, pp. 79–84, San Francisco, Calif, USA, August 2005.

[12] H. H. Zhuo, H. Muñoz-Avila, and Q. Yang, "Learning hierarchical task network domains from partially observed plan traces," *Artificial Intelligence*, vol. 212, pp. 134–157, 2014.

[13] B. Auslander, K. M. Gupta, and D. W. Aha, "Maritime threat detection using plan recognition," in *Proceedings of the 23rd Annual Conference on Homeland Security Technology*, pp. 249–254, Waltham, Mass, USA, November 2012.

[14] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, MIT Press, Cambridge, Mass, USA, 2009.

[15] D. Masato, T. J. Norman, W. W. Vasconcelos, and K. Sycara, "Agent-oriented incremental team and activity recognition," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI '11)*, pp. 1402–1407, July 2011.

[16] A. Sadilek and H. Kautz, "Location-based reasoning about complex multi-agent behavior," *Journal of Artificial Intelligence Research*, vol. 43, pp. 87–133, 2012.

[17] D. I. Kosmopoulos, N. D. Doulamis, and A. S. Voulodimos, "Bayesian filter based behavior recognition in workflows allowing for user feedback," *Computer Vision and Image Understanding*, vol. 116, no. 3, pp. 422–434, 2012.

[18] L. He, C.-F. Zong, and C. Wang, "Driving intention recognition and behaviour prediction based on a double-layer hidden Markov model," *Journal of Zhejiang University: Science C*, vol. 13, no. 3, pp. 208–217, 2012.

[19] 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.

[20] W. J. Mao, J. Gratch, and X. C. Li, "Probabilistic plan inference for group behavior prediction," *IEEE Intelligent Systems*, vol. 27, no. 4, pp. 27–36, 2012.

[21] C. W. Geib and R. P. Goldman, "A probabilistic plan recognition algorithm based on plan tree grammars," *Artificial Intelligence*, vol. 173, no. 11, pp. 1101–1132, 2009.

[22] S. Hladky and V. Bulitko, "An evaluation of models for predicting opponent positions in first-person shooter video games," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG '08)*, pp. 39–46, IEEE, Perth, Australia, December 2008.

[23] S. Zouaoui-Elloumi, V. Roy, J. P. Marmorat et al., "Securing harbors by modeling and classifying ship behavior," in *Proceedings of the 20th Annual Conference on Behavior Representation in Modeling and Simulation*, pp. 114–121, Sundance, Utah, USA, March 2011.

[24] T. Duong, D. Phung, H. Bui, and S. Venkatesh, "Efficient duration and hierarchical modeling for human activity recognition," *Artificial Intelligence*, vol. 173, no. 7-8, pp. 830–856, 2009.

[25] T. L. M. van Kasteren, G. Englebienne, and B. J. A. Kröse, "Activity recognition using semi-Markov models on real world smart home datasets," *Journal of Ambient Intelligence and Smart Environments*, vol. 2, no. 3, pp. 311–325, 2010.

[26] B. Auslander, K. M. Gupta, and D. W. Aha, "Maritime threat detection using probabilistic graphical models," in *Proceedings of the 25th International FLAIRS Conference*, AAAI Press, Marco Island, Fla, USA, 2012.

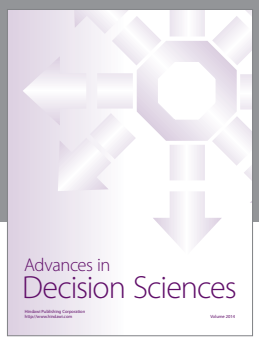
[27] T. D. Ullman, C. L. Baker, O. Macindoe, O. Evans, N. D. Goodman, and J. B. Tenenbaum, "Help or hinder: bayesian models of social goal inference," in *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems (NIPS '09)*, pp. 1874–1882, December 2009.

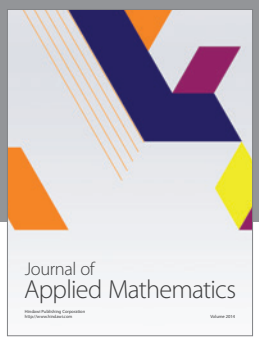
[28] B. Tastan, Y. Chang, and G. Sukthankar, "Learning to intercept opponents in first person shooter games," in *Proceedings of the IEEE International Conference on Computational Intelligence and Games (CIG '12)*, pp. 100–107, IEEE, Granada, Spain, September 2012.

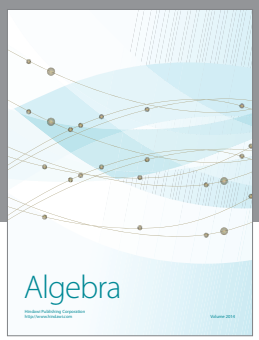
[29] B. G. Weber, M. Mateas, and A. Jhala, "A particle model for state estimation in real-time strategy games," in *Proceedings of the 7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE '11)*, pp. 103–108, AAAI, Stanford, Calif, USA, October 2011.

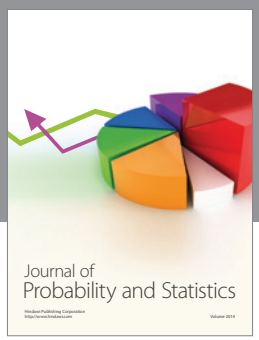
[30] A. Pfeffer, S. Das, D. Lawless, and B. Ng, "Factored reasoning for monitoring dynamic team and goal formation," *Information Fusion*, vol. 10, no. 1, pp. 99–106, 2009.

[31] J. Lapuyade-Lahorgue and W. Pieczynski, "Unsupervised segmentation of hidden semi-Markov non-stationary chains," *Signal Processing*, vol. 92, no. 1, pp. 29–42, 2012.









Hindawi

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