Particle filter-based information acquisition for robust plan recognition

L. Ronnie M. Johansson
The Royal Institute of Technology (KTH)
Nada, Centre for Autonomous Systems (CAS)
SE-100 44 Stockholm
SWEDEN
rjo@nada.kth.se

Robert Suzić
The Swedish Defence Research Agency (FOI)
Department of Systems Modeling
SE-172 90 Stockholm
SWEDEN
robert.suzic@foi.se

Abstract – *Plan recognition generates high-level information of opponents' plans, typically a probability distribution over a set of plausible plans. Estimations of plans, are in our work, made at different decision-levels, both company-level and the subsumed platoon-level. Naturally, successful plan recognition is heavily dependent on the data that is supplied, and, hence, sensor management is a necessity. A key feature of the sensor management discussed here is that it is driven by the information need of the plan recognition process.*

In our research, we have presented a general framework for connecting information need to sensor management. In our framework implementation, an essential part is the prioritization of sensing tasks, which is necessary to efficiently utilize limited sensing resources. In our first implementation, the priorities were calculated from, for instance, the estimated threats of opponents (as a function of plan estimates), the distance to the opponent, and the uncertainty in its position.

In this article, we add a particle filter method to better represent the uncertainty in the opponent state estimate to make prioritization more well-founded and, ultimately, to achieve robust plan recognition. By using the particle filter we can obtain more reliable state estimates (through the particle filter's ability to represent complex probability distributions) and also a statistically based threat variation (through Monte-Carlo simulation). The state transition model of the particle filter can also be used to predict future states to direct sensors with a time delay (a common property of large-scale sensing systems), such as sensors mounted on UAVs which have to travel some distance to make a measurement.

Keywords: plan recognition, sensor management, particle filter tracking, predicted state, delayed measurements, stochastic simulation, high-level information, large-scale system, task prioritization

1 Introduction

Raw (sensor) data obtained from sensor resources is combined in level one of the JDL model, i.e., multi-sensor data fusion [1]. In this function, fusion activities such as identification, association, and classification, are performed. The data are further refined into high-level information in levels two and three of the JDL model, i.e., *situation awareness* and *impact assessment*. Here, we focus on finding an appropriate methodology for translating *high-level information* to *information need* and connecting it to *information acquisition* [2].

By high-level information we mean results obtained from on-line stochastic multi-agent *plan recognition* (PlR) that produces an estimate of a distribution over possible plan alternatives of *agents* acting in an environment. One of the plan alternatives, in our case, is `attack`. The results of PlR are interpretative and try to provide an explanation.

Predictive situation awareness [3] projects a situation into the near future. Recognition of plans is one of the methodologies that are aimed to support predictive situation awareness. PlR gives users hints about what the agent is going to do next given relevant sensor information and a priori knowledge about the agent. Due to high complexity and uncertainty even experienced tacticians are only able to consider two or three possible courses of action for all but the simplest situations [4]. Moreover, we reuse this high-level information for *sensor management* that, in our case, takes into account both derived *threat* estimate and *uncertainty* of data.

PlR is heavily dependent on the acquired information. If sensor resources are limited and cannot provide relevant information in a timely fashion, the results of the PlR will be poor. By relevant we mean that more dangerous plan alternatives are desired to be better known than plan alternatives that are less dangerous. Hence, the information acquisition (IA) or sensor management aspect (JDL level four) is crucial for PlR.

In our previous publications we propose [5] and implement [6] a methodology where the information of PlR is reused in order to connect high-level information need with IA. Here, we further develop this methodology in a more statistically robust and reliable manner.

In this article, we replace our previous coarse (circular shaped) state uncertainty representation [6] with a more convenient one by using a particle filter (PF) [7, p. 11] (see [8] for another use of PF for uncertainty representation in terrain). To deal with the problem of *lacking* sensor data we use the PF and introduce (Bayesian) *robust* plan recognition (RPlR). By robust we mean in the sense of robust Bayesian analysis [13]. This contributes to a better predictive situation awareness and improved sensor task prioritization.

Sensor resources have dynamic constraints and therefore we need a prediction of future agent states to enable *proactive* sensor control. Here, we use the PF for predicting states in a number of following time steps. Finally, we combine

the properties above in our methodology for our proactive multi-object sensor control.

Section 2 explains the context of the problem we are considering, presents the framework we use, and the experiment scenario. Section 3, provides details about the PF we use to represent and maintain state uncertainty. Section 4 presents PlR in general and the robust representation of the current work. Section 5 provides details of various parts of the implemented framework that relates to sensor management. Section 6 provides the results of a simulation experiment that visualizes the results of the PlR. The article is summarized in Section 7, and future research opportunities are discussed.

2 The decision support context

The primary subject of our study is *plan recognition*, i.e., the estimation of the intention of some *agent* observed in a mission-relevant environment.[1] The purpose of PlR is here to support some *information consumers* acting (e.g., performing a mission) in an uncertain environment. The consumers are, furthermore, assumed to be inter-connected through a *network* that connects a set of *nodes*. Formally, we could represent the network with a graph $G = (\mathbf{N}, \mathbf{E})$, where $\mathbf{N}$ is a set of nodes and $\mathbf{E}$ is a set of edges (or communication links) that connects the nodes. The consumers themselves are considered to be members of $\mathbf{N}$. The network structure facilitates, e.g., *reliability* (through the redundancy of multiple communication paths between nodes in the network) and *flexibility* through information exchange between arbitrary nodes in the network. Each node of the network is assumed to have at least communication and computational skills. The individual success of a consumer is dependent on the result of the its (local) PlR for each observed agent, but the resources used to acquire the information that fuels the PlR process are shared.

We do not make any assumption about the network concerning, e.g., topology, communication protocols and information security. Many of those questions have to be settled by the designer of the network, based on available technology, resources, and possibly of the designing organization's policies. However, we do require that the network is capable of conveying information throughout itself and that it (somehow) can collect sensor measurements and perform tracking based on this information. Network nodes should also be prepared to share fused and inferred information with interested nodes, and to assign tasks to *sensing resources* (i.e., sensors) and appropriate sensor configuration. At this point we do not make an attempt to describe how this could be performed.

The instance of the general problem discussed above, which we simulate and present in this article, is the scenario depicted in Figure 1. It concerns an extensive geographic environment including two consumers located in the middle of the view (`a1` and `a2`). The two consumers have individual goals (for the one on the right it is to defend the city in which it is located), but belong to the same network. They both perform PlR based on information about agent states. In the scenario there are nine (hostile) agents, i.e., *platoons*. Groups of three platoons belong to a *company*. There are two companies near the perimeter to the north (labeled `cn1` and `cn2` respectively) and one in the far south of the view (`cs`). There are two types of resources modeled. One type is the UAV observer which can travel quickly but can only give state estimates from a distance (to ensure its own security). The other one is the ground soldier who is limited in speed but who can hide itself close to the road and make comparatively precise state estimates of a passing agent. The network, invisible in the figure, has duties such as collecting measurements to track hostile agents, and to configure and engage sensors in IA tasks (an activity widely known as *sensor management*).

The objective of a consumer is to know as much as possible about the varying *threats* derived from PlR estimates of the hostile agents. Threat estimation is defined in Section 5.1.

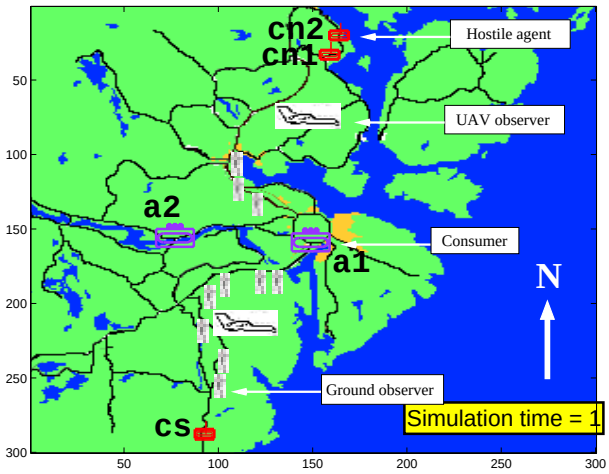

Figure 1: Scenario for use of plan recognition

To approach the problem of resource sharing to support PlR processes, we proposed a framework that emphasizes aspects which we believe are useful to flexibly connect *information need* to *information acquisition* [5]. Information need is interpreted as a lack of information about the state of the environment, that if it was relieved, is believed to improve the decision-making of the system. Aspects that the framework tries to capture include: multiple consumers or objectives, heterogeneous sensors, dependencies among tasks and services, and a separation between the actual sensing resources and the interface of services they provide. In a previous article [5], we compared the framework to others that have been proposed.

The general structure of the framework (depicted in Figure 2) involves two types of entities: *space* and *function*. The four space entities: *task origin*, *task*, *service* and *resource* are containers of structured information. The structure of information of each space entity should suit the intersecting function entities: *task creation and management*, *allocation scheme*, and *service management and resource allocation*.

[1] The part of the observable environment that can have any effect on the mission held by the consumer.

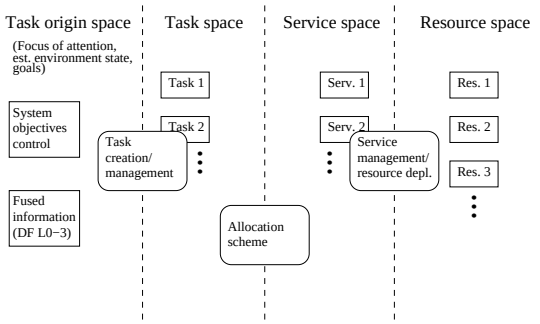

Figure 2: The framework

The framework prescribes that information need (contained within the task origin space) is formulated as information tasks with assigned properties (e.g., priority or time horizon depending on what properties the system is designed to handle). Such tasks belong to the task space in our framework.

The materialization of tasks to satisfy a certain information need could be the responsibility of the task creation and management function. The service space contains services that the sensors in the resource space (independently or jointly) can perform. The allocation scheme describes how tasks are connected to feasible services. These functions are further discussed in [5].

In the current work, the functions of the framework perform the following chores.

Task origin space (TO): It is populated by two origins, i.e., the two consumers $a_1, a_2 \in \mathbf{TO}$. As the state of the environment evolves over time, the two consumers are in constant need of new information (i.e., state estimates) of the hostile agents.

Task management and space: Tasks are created every time step in the simulation from the information need expressed by the consumers. The tasks in the task space $\mathbf{T} = \{\tau_i\}_i$ are simple and basically only expressed in terms of the agent to observe (α) and a priority value $prio$, i.e., $\tau_i = (\alpha, prio)$.

Resource space (R): Contains UAVs and ground soldiers. Both types can make observations, but they have different properties: UAVs are fast, but make uncertain observations. Ground soldiers on the other hand are slow, but their observations are more certain.

Service management and space: In our implementation, for each resource in $\mathbf{R}$ there is a corresponding service s in the service space $\mathbf{S}$. The service management makes sure that the resources are properly configured (e.g., have flight paths) when associated with tasks.

Allocation scheme: Connects created tasks to available services, based on their priorities. Connected pairs of tasks and services are referred to as allocations, e.g., $alloc = (t, s) \in \mathbf{Allocs}$. A service may refuse to perform a task if the cost, for using the resource for the task at hand is too high. Services may also be *preempted*, i.e., removed from one task and connected to another.

3 Particle filter-based state estimation

In a previous article [6], we used a circular agent position uncertainty representation that grew homogeneously whenever observations were not made. To get an assessment of the threat (basically as a function of an estimated plan distribution) posed by an agent with an associated uncertainty circle, we considered a Monte-Carlo approach, i.e., to draw samples uniformly from the uncertainty circle and estimate expected threat and variance. However, the rough uncertainty representation of the circle would have produced poor estimates, since samples within the circle could have been drawn from highly unlikely positions in the environment.

Here we replace the uncertainty circle with a PF for each agent. The PF has several advantages: (i) the position uncertainty representation is more expressive; (ii) the state estimate becomes less sensitive to spurious observations, and (iii) Monte-Carlo simulation can be performed by simply drawing particles from the particle set.

In the particle filter algorithm, the state probability $p(\mathbf{X}_t|\mathbf{z}_{0:t})$ at time t, where $\mathbf{X}_t$ is the agent state and $\mathbf{z}_{0:t}$ is a sequence of observations, is inferred and approximately represented by a set of N particles $\{\mathbf{x}_t^{(i)}\}_{i=1}^N$. The inference at each time step is generally performed in two steps: *importance sampling* and *selection*. In the importance sampling step, for each particle a sample $\tilde{\mathbf{x}}_{t+1}^{(i)}$ is drawn from the *state transition* distribution $p(\mathbf{X}_{t+1}|\mathbf{x}_t^{(i)})$. Each sample is, furthermore, associated with a weight related to its *likelihood* given a new observation $\mathbf{z}_{t+1}$, $\tilde{w}_{t+1}^{(i)} = p(\mathbf{z}_{t+1}|\tilde{\mathbf{x}}_{t+1}^{(i)})$. The weights are then normalized. In the subsequent selection step, N particles are drawn (with replacement) from the weighted set of samples. This set is the posterior at time $t+1$ and will be the prior in the next time step.

The particle filter we use is the one in [7, p. 11] slightly modified. The reason we chose to modify it is because observations may sometimes be infrequent. As a consequence, the particles will drift randomly (possibly away from the area of interest) and also, if a new observation is made it may have little effect since the closest particle may be far away from the observation (i.e., in that case even the most likely particle is unlikely). To counter these problems, we add two modifications to the original algorithm: (i) we let the particles (through the transition model used in the prediction step) be attracted by the consumers, and (ii) the PF is re-initialized whenever observations are too far away from the nearest particle (i.e., a fraction of the particles are relocated to the vicinity of the observation). The rationale behind these modifications is that we use some extra a priori information (i.e., a generic estimation of the opponents' intentions and that a correct observation should be reflected in the configuration of the particle set).

In Section 3.1 and 3.2 we describe the details of our transition and likelihood models.

3.1 State transition model

We describe the state of each particle $\mathbf{x}_t^{(i)}$ by four values. The first two values represent the position of the particle in

x and y-coordinates. The following two are direction, ϕ, and speed, $|v|$.

$$\mathbf{x}_t^{(i)} = [\ x \quad y \quad \phi \quad |v|\ ] \tag{1}$$

Each particle in this case represents a hypothesis of the corresponding (hostile) agent's state. Therefore we say that the particle set is a representation of a multi-modal statistical distribution. Additionally, by using our state transition model we can make a (short time) prediction depending on the sensors mobility and distance to particles.

The agent is influenced by physical and doctrinal properties. Our transition model takes into account the following factors: *previous state* $\mathbf{x}_t^{(i)}$, *terrain properties*, *strategic sites* (such as own forces).

1: **for all** particles $\{\mathbf{x}_t^{(i)}\}_{i=1}^N$ **do**
2: Randomly choose a strategically important place c from the consumers in the task origin space $\mathbf{TO}$
3: $path \leftarrow$ find the shortest road path from $\mathbf{x}_t^{(i)}$ to c
4: $\hat{\mathbf{e}}_g \leftarrow$ calculate the direction of $\mathbf{x}_t^{(i)}$ in $path$
5: $\hat{\mathbf{e}}_v \leftarrow (\cos(\phi), \sin(\phi))$
6: $\hat{\mathbf{e}}_r \leftarrow (\hat{\mathbf{e}}_g + \hat{\mathbf{e}}_v) \,/\, \|\hat{\mathbf{e}}_g + \hat{\mathbf{e}}_v\|_2$
7: $(x_{\text{new}}, y_{\text{new}}) \leftarrow \hat{\mathbf{e}}_r \cdot |v| + \left(\mathbf{x}_t^{(i)}.x, \mathbf{x}_t^{(i)}.y\right)$
8: $(x_{\text{new}}, y_{\text{new}}) \leftarrow most\_likely\_neighbor(x_{\text{new}}, y_{\text{new}})$
9: $\tilde{\mathbf{x}}_{t+1}^{(i)} \leftarrow [\ x_{\text{new}} \quad y_{\text{new}} \quad \phi + \epsilon_\phi \quad |v| + \epsilon_{|v|}\ ]$
10: **end for**

We run our state transition algorithm for each time step, i.e., we propagate particles for each time step with respect to strategic sites (attractors), particles' previous state $\mathbf{x}_{t-1}^{(i)}$ and local terrain properties.

Each particle propagation is *independent* of other particles, but is influenced by a global property that we call *gravity*. In our military scenario, we say that particles are attracted to *strategically important places* (lines *2-4*). To calculate the gravity vector, $\hat{\mathbf{e}}_g$, we calculate the shortest terrain path. Therefore $\hat{\mathbf{e}}_g$ is not in the direction of the straight line between the particle and the consumer. Instead, $\hat{\mathbf{e}}_g$ is calculated based on positions of the particle, terrain and strategic site. Our shortest path is the shortest distance between the particle and the randomly chosen strategic site, given terrain restrictions. The shortest path consists of a number of nodes and the traversability costs associated with edges between nodes. After calculation of the shortest path, the gravity vector is pointing in the direction of the shortest path's first node.

In the next step of the algorithm (lines *5* and *6*), a direction vector, $\hat{\mathbf{e}}_v$, is added to $\hat{\mathbf{e}}_g$. This calculation gives us a resulting vector, $\hat{\mathbf{e}}_r$. Given the resulting vector's direction and speed, the new position of the particle is calculated in line *7*.

The new position of the particle is adjusted to local terrain properties. Each particle position in our discrete terrain representation can be associated with a traversability cost. We specify the size of the local *surrounding* as a parameter. We place a particle at the position of minimum cost in the specified surrounding (line *8*). In line *9* we add white noise to the particle state.

Finally, particles that end up outside the area of interest are replaced by copies of other (propagated) particles, where the probability for each particle is uniform.

3.2 Likelihood model

Inspired by [9], we propose the following likelihood function for sensor observations given a particle

$$p(\mathbf{z}_t|\tilde{\mathbf{x}}_t^{(i)}) \propto \begin{cases} \epsilon + 1, & d < d_s \\ \epsilon + e^{-(d-d_s)^2/(2\sigma_s^2)}, & d \geq d_s \end{cases}$$

Here $\mathbf{z}_t$ is the observation at time t. d_s and σ_s are sensor specific and related to the accuracy of the sensor, and d is the Euclidean distance between the particle and the observation. Therefore, all particles within a circular distance with radius d_s of the observation will receive the same weight. For instance, in our experiments, we use two sensors with different levels of accuracy. The small $0 < \epsilon \ll 1$ improves the particle filter's ability to recover a track when observations are scarce.

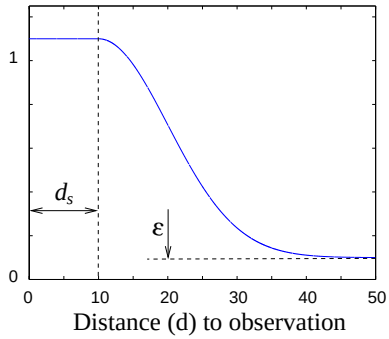

Figure 3: Likelihood model

When we have several concurrent observations, say k, of the same platoon, we let the joint likelihood be the product of all the individual likelihoods for each observation, i.e.,

$$p(\underline{\mathbf{z}}_t|\tilde{\mathbf{x}}_t^{(i)}) = \prod_{j=1}^{k} p(\mathbf{z}_t^j|\tilde{\mathbf{x}}_t^{(i)}).$$

4 Multi-agent stochastic plan recognition

Multi-agent stochastic PlR deals with stochastic outcomes of actions, uncertain observations [10], and incomplete knowledge. In military applications, the use of PlR could be valuable when utilizing sensor data and is of decisive importance in achieving information superiority and predictive situation awareness. One reason is that sensors themselves are not able to reveal the agents' true intentions. Military commanders have to act agilely and they do not have much time (especially on tactical level) to interpret data. In some cases, the difficulty to recognize different patterns is caused by space-time separation, and limited capability to correlate patterns. In some cases, behavior of individual agents can be classified as harmless, but put in some greater context such as agents' mutual interrelations, environment and their assumed doctrines the threat might be identified as much higher. The methodology of PlR helps mitigating these difficulties.

Our model for PlR [11] combines Bayesian statistics and fuzzy membership functions [12]. The latter is used for modeling incomplete knowledge and connecting observation data to a Dynamic Bayesian network (DBN) by entering *context* relevant evidence. The DBN is used for PlR reasoning (inference) about agents on different decision levels. Here, by decision levels we mean platoon and company levels.

4.1 Bayesian robust plan recognition

A problem in Bayesian analysis is its sensitivity to priors. In Robust Bayesian Analysis [13], a single prior distribution is replaced by a set of priors resulting in a set of posteriors. Such an approach is also called *global* Robust Bayesian Analysis, see [13, pages 1-32]. In this section, we introduce multi-agent stochastic Bayesian *robust* PlR (RPlR), which extends multi-agent stochastic PlR by *robustly* dealing with uncertainty in state estimates and lack of observations.

In our original multi-agent stochastic PlR [11], we assumed that a unique, unimodal, qualified guess on agents state, containing agent's positions and velocities, could be obtained at each time step. Later we relaxed the problem by dropping the assumption of continual observations [6]. The new approach was to infer plans only in cases when observations were received; between observations we assumed that the latest plan alternative was valid. The latter approach implies that when new observations arrive, the new plan estimate could greatly differ from the previous plan estimate.

Here, we introduce a PF that maintains a state estimate, even when observations are lacking, by using our state transition model, i.e., $p(\mathbf{X}_t|\mathbf{x}_{t-1})$. The PF produces a multi-modal state representation with each particle as a mode. This representation cannot be used directly with our PlR model. One way would be to take the average of particle sets and take that state estimate into our PlR model. However, bearing in mind that particles could spread in different directions, we do not consider such an approach to be sufficiently robust. The centers of particle sets could represent places with no particles, i.e., unlikely states; and the expressiveness of the particle set is largely ignored.

Here, we propose reconstruction of the multi-modal state representation into a *set of priors* ($\mathbf{\Theta}$) (required by the DBN) instead of one single prior (θ). In the following step, we use each prior separately and run PlR achieving a set of posteriors of plan estimates ($\mathbf{\Pi}_t$). We call this approach RPlR where plan estimates are $P(\mathbf{\Pi_t}|\mathbf{\Theta})$, i.e., instead of one posterior, π, we get a *family* of posteriors, $\mathbf{\Pi_t}$.

The reconstruction of priors could be performed in several manners. Each prior distribution, $\theta_{\mathbf{i,j}}$, consists of two parts that are the *state prior*, $i = 1, \ldots, |\mathbf{Sp}|$, and the *previous plan's prior*, $j = 1, \ldots, |\mathbf{\Pi_{t-1}}|$. The state prior $\mathbf{sp}_i \in \mathbf{Sp}$ is one of the hypotheses of the agent's state. A set of plan priors estimated from the previous time step, $\{\pi^{\mathbf{i,j}}_{\mathbf{t-1}}\}_{i=1\ldots,|Sp|,j=1,\ldots,|\mathbf{\Pi}_{t-1}|}$, is required since we use the DBN for PlR, i.e., estimates of plans at the previous time step has influence on plan estimates at the current time step. Our plan distribution estimate for one state prior with this notation could then be written as $P(\pi^{i,j}_t|\mathbf{s}_i \in \mathbf{Sp}_{t-1}, \pi_{j,t-1} \in \mathbf{\Pi}_{t-1})$.

When reconstructing state priors, a straightforward approach is to select all of the N particles. We construct state priors $\mathbf{S}$ where each $\mathbf{s}_i$ is assigned the value of the particle state $\mathbf{x}^{(i)}_t$, for $i = 1, \ldots, n$. We consider each of the selected particles $\mathbf{x}^{(i)}_t$ to be equal to $\mathbf{s_i}$ of the state priors $\mathbf{S}$. The $|\mathbf{\Pi}_{t-1}|$ previous plan alternatives have to be combined with the state prior. The result is a set of the posteriors where, at the next time step, the previous plan distributions are replaced by $N \cdot |\mathbf{\Pi}_{t-1}|$-plan distributions where each plan distribution corresponds to a combination of a certain state and a previous plan alternative. The number of particles and the number of the state priors is constant (N). However, the number of plan distributions, $|\mathbf{\Pi}_t|$, could grow exponentially $(N \cdot |\mathbf{\Pi}_0|) \cdot N^{\mathbf{t-1}}$. Such calculations soon become intractable. A more convenient way is to assume that our priors are the convex hull (of the previous posteriors) and take only perimeter values of the previous plan alternatives into account, i.e., $P(\pi^{i,j}_t|\mathbf{s}_i \in \mathbf{Sp}_{t-1}, \pi^j_{t-1} \in \text{convhull}(\mathbf{\Pi}_{t-1}))$.

For estimation of the prior plan distribution, π_{t-1}, we test two different approaches given a family of plan distributions from time step $t-1$. The first approach is the medium value of each plan alternative that we denote π^{cg}_{t-1} and the second approach is finding an estimate based on maximum entropy, $\pi^{\text{ent}}_{\mathbf{t-1}}$. In our case, we use only one previous plan distribution.

Due to the large number of particles used in our experiment we construct the state priors from a set of randomly drawn particles. The cardinality of the state priors is typically lower than the number of the particles, i.e., $|\mathbf{S}| \leq N$.

The first approach is based on finding the medium value (center of gravity) of probability for each plan alternative, h, given all plan distributions ($\mathbf{\Pi}_t$), with $L = |\mathbf{\Pi}_{t-1}|$,

$$P^{cg}(\pi^h_{t-1}) = \frac{1}{L} \sum_{l=1}^{L} P^l(\pi^h_{t-1}). \tag{2}$$

The second approach is choosing the estimate that has the maximum entropy compared to other distributions, see Eq. 3 and [14],

$$P^{\text{ent}}(\pi_{t-1}) = \max\left(\texttt{Entropy}(P(\mathbf{\Pi}_{t-1}))\right). \tag{3}$$

5 Sensor management/Information acquisition

Sensor management, i.e., the active control of sensor parameters (such as viewing angles, position, on/off, etc) to acquire information about the mission-relevant environment, is an important support for PlR.

Here, we address the problem of controlling a set of sensors to improve the PlR process. The context provides the following features: there are multiple task origins, heterogeneous sensors, long time-intervals between initiated sensing action and result of the sensing actions, and resource constraints.

To deal with the aforementioned scenario features, we make a number of design choices within the framework (described in Section 2) for our current implementation.

Our current design of task management is described in Section 5.1, allocation scheme in Section 5.2, and, finally, service management in Section 5.3.

5.1 Task management

Tasks are generated by members of the task origin space, e.g., the consumers `a1` and `a2` (from our discussion in Section 2) that require information about the observed agents for their current mission. In the current work, each consumer maintains an estimate of the plans of each known agent. The consumers formulate tasks themselves, one for each opponent platoon. Naturally, the consumers never have enough information about agents and would like to know more about each one of them. Still for the network (system) to perform efficiently with its limited sensing resources the consumers need to prioritize their tasks.

Here, we try to model the prioritization of the consumer that assigns priorities to its tasks. The consumer is not primarily interested in a precise estimation of plan distributions. Instead it should favor information that has the most relevance to its mission. A consumer has, for instance, little use of knowing the plans of some hostile agent precisely if that agent only has little impact on the consumer's mission. We decompose this mission-related prioritization into three parts: (level of) *hostility*, *time-separation* and *impact*. The first one concerns to what degree the agent's plan is hostile towards the consumer, the second to what degree the agent is separated in time from the consumer (all other properties equal, closer agents should have higher priority), and the third concerns to what degree the agent can cause harm to the consumer's mission.

One way to try to capture this is to use *fuzzy set theory*, where the membership of an element to a set is not a binary condition (in or not in). We tentatively propose a "high threat" fuzzy set, HT, expressing the membership degree of an agent state $\mathbf{x}_t^{(i)}$ to the fuzzy set. The fuzzy set HT is now a conjunction of the three parts, i.e.,

$$\texttt{HT} = \left(\texttt{Host}_\texttt{c} \cup \texttt{Host}_\texttt{p}\right) \cap \texttt{STimeS} \cap \texttt{GI}, \tag{4}$$

where $\texttt{Host}_\texttt{p}$ is the "hostile platoon" fuzzy set and $\texttt{Host}_\texttt{c}$ is the "hostile company" fuzzy set. The underpinning explanation of the disjunction of the two hostility degrees is that the hostility of a platoon should not be less than the hostility inferred on the superordinate company. $\texttt{Host}_\texttt{p}$ and $\texttt{Host}_\texttt{c}$ are calculated as normalized and weighted linear combinations of the associated plan distributions (here, denoted $\pi(\mathbf{x}_t^{(i)})$), making the membership degree one when the probability of the plan alternative with the highest weight is one, i.e.,

$$\texttt{Host}_\texttt{p}(\mathbf{x}_t^{(i)}) = \frac{w_p^T\ \pi(\mathbf{x}_t^{(i)})}{\max(w_p)},$$

where w_p is a column vector of weights for platoon plan alternatives. The calculation of $\texttt{Host}_\texttt{c}$ is analogous except for the change of weights.

STimeS expresses the degree to which the separation in time between the consumer and agent is small. This value is based on a function that calculates the agent's least expensive (in terms of traversability) route from its current position to the consumer (also discussed in Section 3.1).

Finally, the GI fuzzy set expresses to what degree the agent can have a great impact on the consumer's mission. In this work, we do not distinguish between the impact of the hostile agents and always use $\texttt{GI}(\mathbf{x}_t^{(i)}) = 1$.

Using the standard fuzzy set operators, Eq. 4 mathematically conforms to

$$\texttt{HT} = \min\left(\max\left(\texttt{Host}_\texttt{c}, \texttt{Host}_\texttt{p}\right), \texttt{STimeS}, \texttt{GI}\right).$$

A calculation of HT is based on a single sample $\mathbf{x}_t^{(i)}$ from the particle set of the corresponding agent. What we want to do is to also capture the statistical properties (i.e., expected value and standard deviation) of the HT membership degree given the state uncertainty expressed by the configuration of the particle set.

Calculating the HT membership degree for each of the particles of the state uncertainty representation of an agent is computationally costly (as the number of particles is typically high). To alleviate this problem, we perform a Monte-Carlo simulation estimation of the expected membership degree of HT, $\mu_{\texttt{HT}}$, and its standard deviation, $\sigma_{\texttt{HT}}$, by drawing M (typically much less than N) samples from the particle set, $\{\mathbf{x}_t^{(j)}\}_{j=1}^{M}$. The calculations are then the following basic estimates

$$\hat{\mu}_{\texttt{HT}} = \frac{\sum_{j=1}^{M} \texttt{HT}(\mathbf{x}_t^{(j)})}{M}, \quad \hat{\sigma}^2_{\texttt{HT}} = \frac{\sum_{j=1}^{M} \left(\texttt{HT}(\mathbf{x}_t^{(j)}) - \hat{\mu}_{\texttt{HT}}\right)^2}{M-1}. \tag{5}$$

The estimates $\hat{\mu}_{\texttt{HT}}$ and $\hat{\sigma}_{\texttt{HT}}$ are calculated by each consumer for each mission-relevant agent, and stored in a task structure, $\tau = (\alpha, prio)$, where $prio = (\hat{\mu}_{\texttt{HT}}, \hat{\sigma}_{\texttt{HT}})$. In our current implementation, we leave it up to the allocation scheme (Section 5.2) to order the tasks by comparison. For the comparison to be fair and make sense, we require that all consumers use the same threat calculation (i.e., Eq. 4) and statistical estimates (i.e., Eq. 5).

The task management performed for one consumer can be summarized in the following algorithm

1: **for all** agents α **do**
2: calculate $\hat{\mu}_{\texttt{HT}}$ and $\hat{\sigma}_{\texttt{HT}}$ according to Eq. 5
3: $prio \leftarrow (\hat{\mu}_{\texttt{HT}}, \hat{\sigma}_{\texttt{HT}})$
4: **if** there already exists a τ s.t. $\tau.\alpha == \alpha$ **then**
5: $\tau.prio \leftarrow prio$
6: **else**
7: create a new task $\tau' = (\alpha, prio)$
8: **end if**
9: **end for**

The only line in the algorithm that requires an explanation is line 4. It checks whether there is already a task τ concerning agent α. If so, line 5, updates its priority (i.e., the priority of a task is allowed to change over time).

5.2 Allocation scheme

The allocation scheme maintains a set of prioritized tasks $\mathbf{T}$ (possibly updated as described in Section 5.1), references

(and means to contact) to the services $\mathbf{S}$, and connections between tasks and services, i.e., allocations $\mathbf{Allocs}$.

Some of the tasks concern the same hostile agent, this is because several consumers may be interested in information about the same agent. In this implementation, the allocation scheme merely considers the maximum value over all tasks that concern the same agent.

```
1: T_t ← sort tasks according to preference relation PR
2: for all sorted tasks τ_t in T_t at time t do
3:    cur_alloc ← get_current_alloc(τ_t)
4:    best_alloc ← get_best_alloc(τ_t, S)
5:    if curr_alloc == best_alloc then
6:       continue with next task in T_t
7:    end if
8:    while best_alloc is not empty do
9:       s ← best_alloc.service
10:      cur_alloc' ← get_current_alloc(s)
11:      if (s not occupied) or (τ_t is preferred to
         cur_alloc'.task according to PR) then
12:         if s occupied then
13:            remove previous allocation for s
14:         end if
15:         if cur_alloc is not empty then
16:            remove cur_alloc
17:         end if
18:         add new allocation best_alloc
19:         break
20:      end if
21:      S ← S \ {s}
22:      best_alloc ← get_best_alloc(τ_t, S)
23:   end while
24: end for
```

Line 1, in the algorithm, orders the presented tasks according to the selected preference relation (which will be described below in this section). The task with the highest priority will be served first. Line 3 finds the current allocation of τ if there is one (thus, we allow for a task to change to a more beneficial service). Line 4 finds the best (most beneficial) allocation for τ_t, i.e., the allocation that gets the best payoff based on a calculation of utility and cost.[2] If the cost for the service is too high, the service might reject τ. Line 5 checks whether τ is already connected to its most preferred service. If so, it continues with the next task.

Line 10 finds the current allocation cur_alloc' of the service s in $best\_alloc$ if it exists. If s is occupied in another allocation and if τ_t has a lower priority than the task of s's current allocation cur_alloc', the program continues on line 21 where the next best allocation is found (if any). Otherwise $best\_alloc$ is employed. Line 13 realizes the preemption property of the algorithm, i.e., that an allocation may be removed if there is a task that is in more need of a service than the one currently allocated.

In Section 5.1, we explained that the priority stored for a task is actually the tuple $(\hat{\mu}_{\texttt{HT}}, \hat{\sigma}_{\texttt{HT}})$. Optionally, we could have combined these two into a summarized value by a weighted sum in the task management function. Here, instead, we want to allow the network to decide what is more important, expected threat or standard deviation. To do so, we introduce two preference relations PR_μ and PR_σ, that represent both desires, respectively.

PR_μ prefers a task τ_1 to another τ_2, if $\tau_1.prio.\mu - \tau_2.prio.\mu > \delta$. If $0 \leq \tau_1.prio.\mu - \tau_2.prio.\mu \leq \delta$, τ_1 is only preferred to τ_2 if $\tau_1.prio.\sigma > \tau_2.prio.\sigma$. PR_σ is defined symmetrically.

5.3 Service management and resource deployment

The resource deployment part of our implementation performs a simple path planning for the UAV sensors and sends them on their way. The path planning we choose reuses the particle set approximation of the state uncertainty for an agent by applying the state transition algorithm (Section 3.1) to predict the configuration of the particles when the UAV is likely to be able to make an observation. A path for the UAV is then constructed by drawing path nodes from the predicted particle set. An evaluation of the efficiency of this heuristic is beyond the scope of the current article.

It is the responsibility of the service management to check whether services have completed or reached the end of its path and if so make the service available (even to tasks with low priority).

Note that, although not implemented here, some of the responsibilities of service management and resource deployment could be deferred to the resources themselves.

6 Experiments

Evaluating the proposed framework and implementation of the scenario described in Section 2 is difficult considering its complexity (i.e., it involves a multitude of parameters concerning PlR, IA, and their connection). The proposed problem space is also uncommon and there is little to compare to in the literature.

Figure 4 shows a comparison between the best possible (`bp`) estimates of the `attack` probability (probably the most interesting plan alternative for a decision-maker) and two of our RPlR estimates. The `bp` estimate is achieved given continual and accurate observations of all agents. The first RPlR estimate represents our center of gravity (`cg`) attack estimate from the set of posteriors (Eq. 2). Here, we represent uncertainty intervals (dashed), $[\min \mathbf{\Pi}_t^{\texttt{attack}}; \max \mathbf{\Pi}_t^{\texttt{attack}}]$, of the posteriors as well. We also show the maximum entropy estimate (`me`) calculated using Eq. 3 (dash and dot in the figure). Both estimates are dependent on the implemented IA, described in Section 5, using the PR_μ preference relation.

In Figure 4, initially `cs` is observed by both UAVs and ground observers and the estimated attack probability is close to `bp` and the uncertainty interval is small. The increasing attack probability attracts the interest of the sensors and the uncertainty interval is kept relatively small. By the end of the scenario, near time step 80, the attack probability has decreased (because `cs` is moving away

[2] The utility is based on the quality of the expected observation and the time it is anticpated to take before the observation can be made. The cost is based on the cost of initiating and running the sensor (e.g., in terms of fuel to transport the UAV).

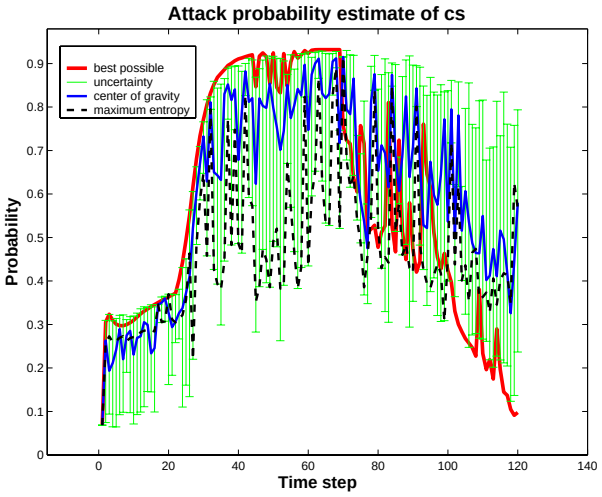

Figure 4: Attack probability estimate for agent `cs` for predictive situation awareness.

from the consumers) and the interest of the sensors has been lowered. This, together with the bias of the state transition model (which pulls particles towards the consumers), explains why the uncertainty interval fails to cover the `bp` estimate in the last time steps.

For this experiment, the `cg` rather appears to better approximate `bp` than `me`. The `cg` approximation considers the whole set of distributions unlike the `me` estimate.

7 Conclusions and discussion

The consequence of using a PF for PlR is that we obtain more reliable results for agent plan estimates, than our previous results in [6], by introducing RPlR. The reason is basically that the PF better represents the uncertainty in the state estimates.

The proposed PF also contributes to the IA part of this research. The state transition model of the PF can be exploited to predict (the most likely) future agent states to proactively control sensors. The PF in conjunction with the RPlR can also be used to estimate expected threat and variance for sensor task prioritization.

In terms of the proposed framework, we have practically explored parts of its domain. We have, however, yet to explore, e.g., dependencies among services and tasks, which appears to be challenging to master efficiently.

In the future, we would like to enrich the state transition model by conditioning particle behavior on plan distributions, and introduce other agent-like properties. Concerning RPlR, it needs to be both formalized and its potential investigated (e.g., to evaluate the trade-offs between completeness and tractability).

Acknowledgments

This work was financially supported by the Swedish Defence Research Agency (FOI), Division of Command and Control. The authors also wish to thank the following people for their contributions to this research: Hedvig Sidenbladh (FOI), Christian Mårtenson (FOI), Niklas Wallin (FOI), Stefan Arnborg (KTH) and Klas Andersson (Swedish National Defense College).

References

[1] A. N. Steinberg and C. L. Bowman. Revisions to the JDL data fusion model. In D. L. Hall and J. Llinas, editors, *Handbook of Multisensor Data Fusion*, chapter 2, pages 2–1 – 2–19. CRC Press, 2001.

[2] L. R. M. Johansson. Information acquisition in data fusion systems. Licentiate Thesis TRITA-NA-03-28, KTH, 100 44 Stockholm, SWEDEN, November 2003.

[3] R. A. Piccerillo and D. A. Brumbaugh. Predictive battlespace awareness: Linking intelligence, surveillance and reconnaissance operations to effects based operations. In *Proc. of the 9th Intl C2 Research and Technology Symposium*, 2004.

[4] P. W. Phister, T. Busch, and I. G. Plonisch. Joint synthetic battlespace: Cornerstone for predictive battlespace awareness. In *Proc. of the 8th Intl C2 Research and Technology Symposium*, 2003.

[5] R. Johansson and R. Suzić. Bridging the gap between information need and information acquisition. In *Proc. of the 7th Intl Conf. on Information Fusion*, volume 2, pages 1202–1209, 2004.

[6] R. Suzić and R. Johansson. Realization of a bridge between high-level information need and sensor management using a common DBN. In *The 2004 IEEE Intl Conf. on Information Reuse and Integration*. IEEE, 2004.

[7] A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2001.

[8] H. Sidenbladh. Multi-target particle filtering for the probability hypothesis density. In *Proc. of the 6th Intl Conf. on Information Fusion*, pages 800–806. International Society of Information Fusion, 2003.

[9] J. Lichtenauer, M. Reinders, and E. Hendriks. Influence of the observation likelihood function on particle filtering performance in tracking applications. In *Proc. of the 6th Intl Conference on Automatic Face and Gesture Recognition*, 2004.

[10] H. H. Bui, S. Venkatesh, and G. West. Policy recognition in the abstract hidden markov model. *Journal of AI Research*, 17:451–499, 2002.

[11] R. Suzić. Representation and recognition of uncertain enemy policies using statistical models. In *Proc. of the NATO RTO Symposium on Military Data and Information Fusion*, October 2003.

[12] L. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[13] D. Rios Insua and F. Ruggeri. *Robust Bayesian Analysis (Lecture Notes in Statistics)*, volume 152. Springer-Verlag, 2000.

[14] E. T. Jaynes. *Probability theory: The logic of science*. Cambridge University Press, 2003.