Realization of a Bridge between High-Level Information Need and Sensor Management Using a Common DBN

Robert Suzić
Swedish Defense Research Agency (FOI)
Department of Systems Modeling
SE - 172 90 Sundbyberg
robert.suzic@foi.se; rsu@nada.kth.se
SWEDEN
Tel: +46 8 55 50 32 17

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

Abstract

In a decision support system for military decision makers a plan recognition process provides estimates of enemy plans. To respond to a changing and uncertain environment the plan recognition process requires timely and relevant information.

We address the rarely discussed, yet crucial, issue of connecting the information needs of plan recognition to management of sensors. We have previously presented a framework for this purpose and here we give details of an implementation and provide some results. In our implementation both plan recognition, sensor management and the functions that connect them utilize the a priori knowledge stored in a Dynamic Bayesian Network.

1 Introduction

We consider a data fusion process [1] to be a component of an enclosing system, e.g. a decision support system. Its purpose is to exploit information provided by disparate sources.

In a series of previous articles (incl. [2]) we addressed the data fusion issue of plan recognition (PlR) [3, 4] using a Dynamic Bayesian Network (DBN). Inference of plans may facilitate predictive situation awareness for decision-makers in a decision support system. However, the performance of the PlR process is heavily dependent on the observations it receives.

Recently, our focus has expanded to include also the aspect of information acquisition [5] to improve the PlR. One motive for this expanded focus is that effective measurements of environments are rarely accessible for passive sensors (save for controlled industrial environments). A second motive is that purposeful control of sensors promises to yield more relevant information that will increase the performance of the PlR process.

In [6], we introduced a framework for connecting the high-level information need of PlR to *information acquisition* in an automatic manner. The framework addresses emerging needs for multiple objectives and multiple sensors. Those issues have not been studied previously. Here, we extend the efforts in [6] by further discussing the details of our "bridge" between PlR and sensor management and present some results from a prototype implementation.

For a selection of related efforts in the literature see references in [6].

2 High-level information need and sensor management

2.1 High-level information and information acquisition

PlR produces estimates of plans of agents acting in the environment. We label this "high-level" information since it is interpretative and tries to provide an explanation. In contrast, "low-level" information typically originates directly from sensors and simply estimates observable properties of the environment (such as position, feature, etc.).

The purpose of the information acquisition part of the data fusion system (sometimes called adaption or sensor management) is to improve the quality (e.g., the accuracy or relevance) of the information generated by data fusion. As support for PlR, the information acquisition process should strive to make observations that will generate plan estimates with less uncertainty. However, not only the needs of data fusion processes should be satisfied by information acquisition, but also

the objectives of the decision maker should be considered (e.g., the decision maker might value correct estimates of some plans more than others).

2.2 The connecting framework and information reuse

Our framework, depicted in Fig. 1, prescribes that information need arising in some system (we call the source of this need 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.

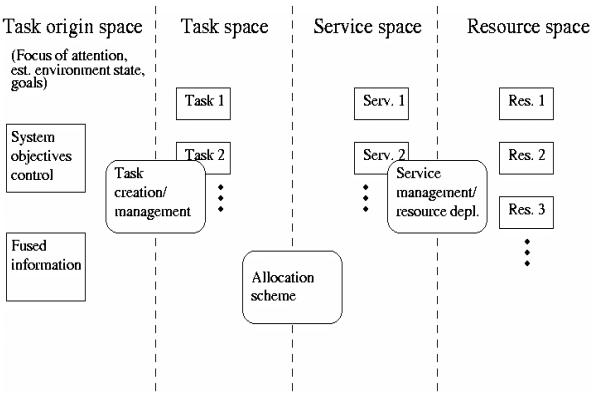

Figure 1. Framework

The materialization of tasks from information need could be the responsibility of a *task creation and management* function. The *service space* contains services that the sensors in the *resource space* (independently or jointly) can perform. The benefit of utilizing these services to satisfy tasks is, subsequently, the (more) relevant data that eventually is returned to the data fusion process by employed sensors. The *allocation scheme* describes how tasks are connected to feasible services. These functions are thoroughly discussed in [6].

Usage and reuse of domain knowledge is crucial to sustain the connection between PlR and information acquisition. First, PlR uses plenty of expert and domain knowledge (such as enemy doctrine, trafficability of various vehicles in different terrains, sensor characteristics for various weather conditions, etc) to swiftly infer estimates of enemy plans from observations. Second, the same expertise can be reused to prioritize information tasks and employ appropriate sensors. Even though this approach biases the PlR process towards a limited set of plans and sensor management towards a limited set of alternatives; it is necessary in many real-time applications to focus on the most likely plans given the *a priori* knowledge. We discuss this further in Section 4.

3 The concept of information agents in threat analysis

3.1 Role of information agents in threat analysis

Threat analysis is a function of information fusion. The purpose of this process is to estimate and predict the impact of observed objects or phenomena. Input to this process is supposed to be the aggregated situation picture (common understanding of situation). Before entering threat analysis the information about entities is processed. Such information is usually information about an entity's position and its identity.

We use the concept of knowledge representation agents for threat analysis. Generally, an agent [7] is anything that can act using its effectors and perceive using its perceptors (i.e. sensors). An agent-based software concept that is aimed to represent our knowledge or qualified guess/es about observed, existing object(s) we call *information agent* (IA). The entity or object that we observe we just call (real world) agent.

If we observe an entity (real world agent) of a certain type we use a corresponding software model, IA, of that entity's type to represent it. In the next step, data received from sensors such as position and uncertainty about position are entered. Also more sophisticated data about the entity may be entered such as pattern data. Data are also combined with *a priori* knowledge to obtain new, *a posteriori*, information. Such information may contain estimated intentions of the observed agent or its (predicted) impact.

3.2 Responsibility of IAs

IAs have three main properties:

1) To package current information about real-world agents (e.g. individual vehicles) or their aggregates (e.g. platoons);
2) Infer (fuse) a priori knowledge and gain new knowledge on different abstraction levels;
3) Choose suitable object model depending on context, user interests and previous observations (e.g. for predicting future states).

IA can be hierarchical, meaning that an IA may consist of other IAs. IAs can also be interpreted as IAs on different abstraction levels. This concept could be useful when modelling flexible decision support systems such as [8].

4 Knowledge reuse for adapting plan recognition

4.1 Soft computing model

It is generally difficult to derive conclusions about the agent's (enemy's) intentions from a chaotic, uncertain and complex environment. To achieve agility, military commanders need to have good (predictive) situation awareness. Recognition of threats gives users, military commanders in this case, hints about what the agent is going to do next, provided relevant sensor information and a priori knowledge about the enemy.

The goal of PlR is to derive a threat estimate given information about an agent's (hostile force's) plan estimates that gives clues about threats that own (friendly) forces may be exposed to. In the next step we use this information for prioritization of automatic sensor management.

PlR requires modeling of a priori knowledge, using (dynamical) sensor data and finally inferring a qualified guess of enemy intentions. To infer knowledge with dynamical information we use a DBN, see [2].

We fuzzify sensor data and use fuzzy membership degree as the subjective probability measure in DBN nodes. The output is a probability distribution of the plan alternatives on different abstraction levels. PlR is important to see in light of enemy capabilities, our force capabilities and the strategic value for the attacking enemy. To date our DBN models have only one node (*force balance*) that models this aspect. We hope that in future work we will have more sophisticated models that handle this capability/cost/utility/strategic value issue. A challenging approach for reasoning about other agents' reasoning, i.e. Game Theory [9], could be used to improve plan recognition.

4.1.1 DBN component

The DBN model's abstraction levels are corresponding to a hierarchical organization structure of an enemy company that is considered as a higher level agent in this case. It consists of three tank platoons, each platoon containing three tanks that are agents of the lowest (abstraction) level in this model. For each abstraction level there is a certain set of plausible *plans* for that level. Those plans are influenced by *plans* on higher abstraction levels. The simplest plans, the *atoms*, consist only of a set of *actions*. In this example the simplest plans are platoon actions. More complex plans consist of other plans, also referred to as *sub-plans*, or a mixture of sub-plans and actions. Higher level plans invoke lower level plans down to their actions.

The IAs that we described in Section 3 are used for representation of fused information. The relations between such agents should follow the abstraction levels modelled in the DBN. In our DBN model an information agent is a part of a DBN that contains information or a qualified estimate of the corresponding (hostile) agents plans.

4.1.2 Fuzzy component

An important part of PlR is the use of fuzzified sensor data. This process takes sensor data as input. Calculated fuzzy set membership degrees are translated into a corresponding node in the DBN.

By using fuzzy functions conclusions can be drawn about e.g. elusive behavior patterns. Given uncertain data, we need qualified guesses about some patterns. Some behaviour patterns give support for certain plan hypotheses. Actually, the fuzzification process performs classification of patterns into different fuzzy sets of pattern types.

There are also some other values of interest such as a fuzzified notion of how long time it takes for the enemy to reach us. Therefore we define the fuzzy sets *short time to impact* and *long time to impact*. Those fuzzy sets define a family fuzzy set corresponding to the DBN variable time to impact.

4.2 Sensor management

Information acquisition through management of sensors is foremost necessary to provide PlR with information to sustain a minimum level of performance and beyond that to optimize performance. Here we discuss two parts of our implementation that utilize expert knowledge.

4.2.1 Estimation of task priorities

For the task management function of the framework, we implement creation and prioritization of tasks. We create one task for each known enemy unit in the environment. Each task demands up-to-date information about the corresponding enemy unit. The tasks represent the information need of the PlR process. The origin of these tasks is the IAs that we say belong to the task origin space.

Moreover, in order to achieve purposeful information acquisition, task management prioritizes (i.e., orders) tasks. We introduce the notion of *threat* and the idea that the higher threat posed by an enemy unit the higher the priority of its corresponding task should be. The threat calculation integrates knowledge of estimated plans (*ep*). We introduce threat weights (w_j) whose magnitude is dependent on the danger (threat) corresponding to each plan alternative x_j. The probability for a plan alternative x_j is $p(x_j \mid obs)$ given observations. E.g. the weight corresponding to a plan alternative `attack` has greater magnitude than a weight for alternative `march`.

Hence,

$$ep = \sum_j w_j \cdot p(x_j \mid obs)$$

is a summarized threat value of the plan distribution. The threat estimate (T) becomes:

$$T = ep$$

We cannot just be satisfied with this calculation of threat estimation for task prioritization. It does not fully respect the sensitivity caused by the properties such as position of the enemy units. Ideally, we want to find the expected threat and threat variance of each enemy unit given estimated properties and uncertainties. In general, the expected threat and threat variance cannot be calculated analytically. We would therefore like to approximate these properties using Monte Carlo simulation. However, our current position uncertainty model is simply an uncertainty radius (*ur*) that grows with time when no observations are made. The model is unfortunately both ignorant of terrain characteristics and sampling from it is computationally costly. Instead in the experiments in the following section, we simply add the *ur* of each position estimate to the threat estimate. A possibly feasible approach would be to let a terrain-aware particle filter [10] represent the position uncertainty. Sampling from the particles would yield more accurate estimates of expected threat and variance.

For prioritization of tasks additional factors could be considered. The expected impact (*ei*) of the enemy unit attacking one of our units or essential resources could be explicitly represented. The priority could also depend on the time duration (*td*) before a particular enemy unit can engage in an operation against our resources; longer duration gives less threat. We also motivate the use of *td* due to sensors limited velocity that may result in a considerable difference in time delay for different observations.

Hence in principle, the task priority becomes:

$$Priority = T + ur + ei + td$$

4.2.2 Expected utilities of services

The allocation scheme part of the framework (Fig. 1) provides another example of reuse of knowledge in the military decision support system discussed here. The allocation scheme function connects tasks to services, where services correspond to sensing resources belonging to the system.

Allocation scheme evaluates candidate allocations of tasks to services based on the cost of using the service and the utility of the task when using a specific service. If the cost for available services for specific tasks is too high the allocation scheme may decrease the priority of the task for the time being.

Ideally, we would like to achieve cost and utility estimates by simulating the performance of each task for each service. High utility should be awarded to allocations that are expected to greatly affect the threat value (which is based on the same knowledge as used by PlR) of an enemy unit. This approach is, however, generally infeasible in real-time applications and we therefore use simplified utility models in our experiments in Section 5. These models do consider sensor and estimated enemy unit properties. Such as, *service quality*, *expected time to task completion* and *cost of resource use*. Service quality is the expected quality of the data/information returned by an allocated sensor that a certain service employs. By expected time to task completion we mean the estimated time that is required for a sensor to make an observation. There are static costs assigned to the initiation of a service (e.g. allocating personnel for deploying a UAV) and dynamic costs (e.g. fuel consumption for task completion).

5 Results

In this section we present a tactical scenario containing a hostile battalion and two strategic sites (own forces). We show two experiments using the implemented framework. Our aim is to demonstrate that our implementation gives reasonable results.

In the first experiment, we illustrate the need of sensor management for plan recognition. We run the implemented framework on different sensor configurations for evaluation.

In the second experiment, we study the effects of task prioritization on sensor management by using the framework.

5.1 Scenario

Enemy forces have performed air landing and one company, Company South (CS), is observed in region south (Rs), see Fig. 2. CS is observed when advancing in northerly direction. At approximately the same time in Region north (Rn) two heavily armed tank companies, Company north1 and north2 (CN1 and CN2), are observed. It has been observed that they are not moving, according to some reports due to fuel problems. CS is advancing towards the town in Region center (Rc) where one of our tank platoons (OF1) is located. The other own force (OF2) unit is located in Rc west of the town. The

scenario evolves through time and CS is close to the town when the OF1 destroys the bridge that separates them. CS withdraws and now heads in the direction of OF2, presumably to the take remaining bridge between Rc and Rs.

Our sensor resources are limited to one UAV and a few ground troop soldiers. After some time (90 time steps) CN1 and CN2 suddenly start moving towards OF1 in Rc.

The problem of efficient sensor allocation given a complex threat situation arises.

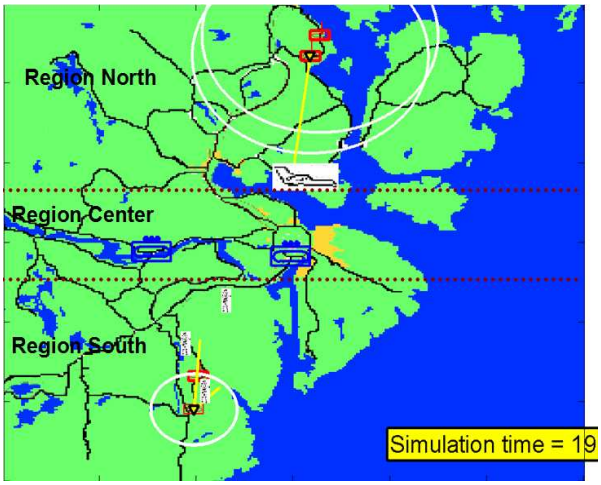

Figure 2. Scenario Map
*yellow line represents sensor's task to observe enemy force

5.2 Plan estimate loss caused by limited number of sensors

In this simulation we vary the number of sensors in region north. In our case, these sensors are of type "Markus" (ground troop soldier) and are assumed to observe objects (agents). In this section we focus on one of the enemy companies in the north and estimate the probability that this company will attack our force that is located in the town. We perform four simulations of 160 time steps. Each of them returns the probability for attack given a varying number of sensors. In the first simulation we assume that we are able to observe the enemy at all times steps. This is equivalent to using an infinite or sufficient amount number of sensors in the simulation. The attacking probability estimate for an infinite number of sensors is used as a reference when comparing to other attacking probability estimates with a limited number of sensors (observations).

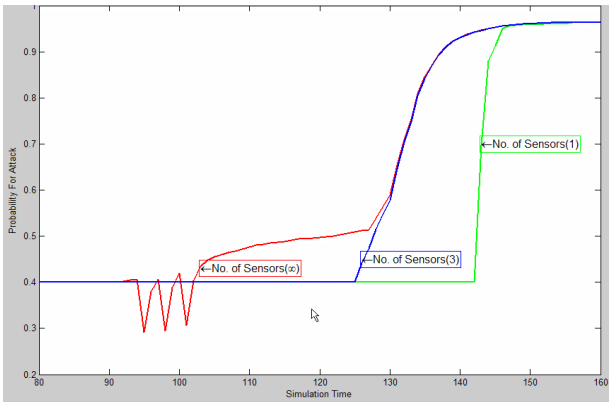

Figure 3. Attacking probability estimate over time given sensor configurations with one, three and infinitely many sensors

In Fig. 3, all attacking probabilities are equal while the enemy company is not moving. In other words, estimated position is the real position. The CN2 starts moving and we observe first diverge of the plan estimate for unlimited number of sensors (red line in Fig. 3). In the case of three sensors we get a result that underestimates attacking probability in a time interval (blue line). The time difference for the case of one sensor is even larger (green line). This result leads us to a definition of *plan estimate loss* caused by limited number of sensors i and tracking is performed by some sensor resource method $M_r(i)$. In order to quantify how critical each plan alternative is we define the following penalty loss function. It assigns penalty measure $Pen(\mathbf{x}_j)$ over plan space (χ) for each plan alternative ($\mathbf{x}_j$). Then we calculate $I(M_r(i), \mathbf{x}_j)$, see Eq. 1, which is the area between plan estimate for the case of unlimited number of sensors $p_\infty(x_j)$ and for the case of limited number of sensors $p_i(x_j)$ in relevant surrounding. Finally, we define plan estimate loss as in Eq. 2:

$$I(M_r(i), x_j) = \int_{t\_start}^{t\_end} | \, p_\infty(x_j) - p_i(x_j) \, | \, dt \qquad (1)$$

$$Loss(i) = \sum_{xj \in \chi} Pen(x_j) \cdot I(M_r(i), x_j) \qquad (2)$$

5.3 Threat variation and focus change

In our second experimental simulation, we illustrate how the system changes its focus of attention by changing tasks for sensors. Task priorities are calculated by the task management function of the implemented

framework. Focus of attention is maintained by the allocation scheme which has to consider both task priorities and the availability of services. We use the same scenario as presented in Section 5.1 with three enemy companies, three ground observers (located in Rs), and one UAV (located in Rc).

In Fig. 4 we show how the calculated task priority of the three companies varies during 35 simulation time steps. Initially, CS (the solid line in Fig. 4) is the greatest and it increases as long as the company has not been observed. The UAV sensor is accordingly allocated to CS.

CS is moving north along the road in the beginning of the scenario (while the companies in Rn remain stationary). After a short while (about time step 7), before the UAV has a chance to observe CS, CS is spotted by one of the ground observers in region. Since the uncertainty of the whereabouts of CS has been lowered, the priority decreases. At this point priorities of CN1 and CN2 (the dashed lines) exceed that of CS and the UAV starts to look for CN2 instead.

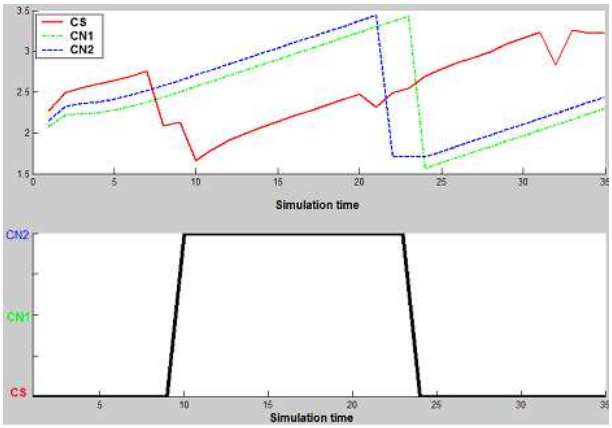

Figure 4. Task priorities and focus of attention

Around time step 20, the UAV observes CN2, but also CN1 which is in the vicinity. The threat levels of both CN1 and CN2 drop rapidly and CS has once again the highest threat level. The UAV changes its selected target back to CS as expected. This result suggests that the automatic management of sensors presented in this article agrees with an intuitive sensor control.

6 Conclusions and future work

We have demonstrated how high-level information, in our case enemy intentions, can be used when prioritizing tasks for information acquisition. Reuse of PlR knowledge models when performing task prioritization has turned out to be an effective strategy.

We show that our implementation can be used to evaluate the performance of PlR for various sensor configurations.

The simple simulation experiments presented in Section 5 provide a glance at the qualities of the system. However, future more complex scenarios where the need for automatic control is apparent will exhibit more interesting results.

7 References

[1] A. N. Steinberg and C. L. Bowman. *Handbook of multisensor data fusion.* Eds. David L. Hall, James Llinas, Ch. 2. CRC Press, 2001.

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

[3] H. H. Bui, S. Venkatesh, and G. West, "Policy recognition in the Abstract Hidden Markov Model", *Journal of A rtificial Intelligence Research*, 17: 451-499, 2002.

[4] A. R. Pearce and C. A. Heinze and G. Goss, "Enabling perception for plan recognition in multi-agent air-mission simulations". *Proc. Fourth International Conference on Multi-Agent Systems (ICMASS2000),* pp 427-428, 2000.

[5] L. R. M. Johansson, *Information Acquisition in Data Fusion Systems*, TRITA-NA-0328, ISSN 0348-2952, ISRN KTH/NA/R--03/28--SE, ISBN 91-7283-655-5, CVAP283, November 2003.

[6] L. R. M. Johansson and R. Suzić, "Bridging the Gap between Information Need and Information Acquisition", *Proceedings of the 7th International Conference on Information Fusion*, ISIF, Fusion, 2004.

[7] S. J. Russell and P. Norvig, *Artificial Intelligence*, ISBN 0-13-103805-2, Prentice-Hall, New Jersey 1995.

[8] K. Wallenius, *Generic Support for Decision Making in Management and Command and Control*, Licentiate Thesis, Nada, Royal Institute of Technology, Stockholm, Sweden, 2004.

[9] J. Brynielsson and S. Arnborg, "Bayesian Games for Threat Prediction and Situation Analysis", *Proceedings of the 7th International Conference on Information Fusion*, ISIF, Fusion, 2004.

[10] M. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking", *IEEE Transactions on signal processing*, Vol. 50, No. 2, February 2002.