An Adaptive Situation Assessment Based Decision Making System

F. Mirmoeini and V. Krishnamurthy
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, BC, Canada
farnoush,vikramk@ece.ubc.ca

Abstract - This paper describes the development of a hierarchical situation assessment system using Bayesian networks and also a situation assessment based decision making system for battlespace environment. The situation assessment system consists of two levels of reconfigurable Bayesian networks that are adapted with changes that occur in the battlespace on two different timescales. The decision making system uses this adaptive situation assessment system to make decisions that in turn affect the battlespace dynamics. An algorithm is provided to model these interactions and dynamics of the battlespace. Furthermore, a Markovian model for the battlespace dynamics is provided.

Keywords: Situation assessment, Bayesian networks, battlespace modeling.

1 Introduction

Situation assessment (SA) is the ongoing process of inferring relevant information about the forces of concern in a battlespace environment. It is, in other words, the task of inferring “what is going on” or “what is happening” from data collected by sensors and processed by lower level data fusion and target tracking systems in a battlefield environment.

In this paper we briefly describe a novel hierarchical Bayesian network based situation assessment algorithm that can be dynamically adapted to cope with changes in a battlespace environment. For a full description see [1] and [2]. The Bayesian networks used in situation assessment are hierarchical and are dynamically reconfigured to cope with the changes that occur as the battle proceeds on two different timescales. This hierarchy of Bayesian networks is able to process the data in different levels of abstraction. The situation assessment system can be passed to the higher level commander to make decisions upon. After performing situation assessment, one needs to make decisions about what to do under the assessed situation and take action based on the hypotheses generated by the situation assessment system. This task is usually performed by a military commander. In this paper, we propose means to automate the process of decision making by using the information provided to us by the situation assessment system. Furthermore, we provide a Markovian model of the battlespace that enables us to estimate quantities such as the average duration of the battle and the probability of winning for each side. This paper is organized in six sections. In the next section we briefly explain the situation assessment system that uses a hierarchy of Bayesian networks. In section 3 we describe the algorithms used for reconfiguring the situation assessment system. Section 4 describes the process of decision making and also the Markovian model for the battlespace. We provide a numerical example in section 5 and some concluding remarks in section 6.

2 A Hierarchical Multi-Timescale Architecture for Situation Assessment System

Consider a battlefield situation that comprises of friendly and enemy forces and their platforms, assets and interactions. At each time instant, sensors record measurements from the battlefield. The data collected by sensors is plentiful but usually low in information content; furthermore this data cannot be used directly in a situation assessment system because it is noisy and also does not contain all quantities that represent the battlespace situation (since some quantities are not directly measurable and some are not observable all the time), therefore it needs further processing before it can be used in decision making.

The adaptive situation assessment system consists of five functionality blocks, as outlined in Figure 1. The function of this hierarchical adaptive situation assessment system is to assess the situation by evaluating the likelihood of different hypotheses. The lowermost block is the sensor system, which collects data using sensors. The second block is the integrated tracking system (integrated tracker), which processes the sensor readings. The integrated tracker is mainly responsible for processing the incoming noisy sensor data and feeding it to the Bayesian networks for inference. The third block is an averaging device whose purpose is explained later on in this section. The fourth and fifth blocks consist of Bayesian networks that perform adaptive situation assessment. The reason we have two dif-
Different levels of Bayesian networks is that these two levels operate on different timescales and therefore they cannot be integrated in one large Bayesian network. For this situation assessment system, we introduce a multi-timescale model where each block processes the data on a faster timescales than the ones above it. The function of the situation assessment system is to produce concrete and tangible hypotheses that can be used in command and decision making. These hypotheses are derived by summarizing lower level data from sensors and the integrated tracking system. The Bayesian networks are responsible for estimating the states of hidden variables as well as making inferences from data provided by sensors and the integrated tracking system. These Bayesian networks are reconfigured periodically in order to provide hypotheses that reflect the current state of the battlespace best.

The training data and evidence generated by the integrated tracker (see Figure 1) are fed into the lower level Bayesian networks. We use Bayesian networks in the adaptive situation assessment since they can represent dependence relationships between uncertain and incomplete information, which is otherwise very hard to capture. Bayesian Networks are able to perform inference and learning on incomplete data and hidden variables, which is needed in a situation assessment system because not all variables that represent the battlespace situation are observable. Although the integrated tracker processes and summarizes the sensor data, it is not able to make inferences and handle hidden variables, so we need to use Bayesian networks to perform these tasks.

We formulate a generic Bayesian network (in the higher or lower level) for situation assessment as follows. Consider the set of \( n \) random variables, \( U = \{X_1, ..., X_n\} \), where each \( X_i \) represents a variable output by the integrated tracker. A Bayesian network is a pair \( B = \langle G, \Theta \rangle \). \( G \) is a directed acyclic graph (DAG) that encodes the dependence relationships, in the sense that the nodes represent elements of \( U \) and the directed connections represent the conditional probability of inferring the state of a particular node given the states of its parent nodes. \( G \) also encodes the conditional independence assumption that each node \( X_i \) is conditionally independent of the rest of the network given its direct parents. \( \Theta \) represents the set of parameters that quantify the network. For each node \( X_i \) it contains the conditional probability table (CPT) of each node, which in turn contains the conditional probability of a node given its parents. Parameters are of the form \( \theta_{x_i|\Pi_{x_i}} = P(x_i|\Pi_{x_i}) \) for each possible value of \( X_i \) and \( \Pi_{X_i} \). Therefore, a Bayesian network \( B \) defines a joint probability distribution over \( U \) that can be factored as follows:

\[
P_B(X_1, ..., X_n) = \prod_{i=1}^{n} P_B(X_i|\Pi_{X_i})
\]

In our design we use a fixed structure for the lower level Bayesian networks. One of these networks is shown in Figure 2. The variables chosen to represent these Bayesian networks are quantities with known and fixed dependence relationships. The parameters \( \Theta \) of these networks, however, can change, and we need to adapt these parameters to changes that occur in the battlespace. We compute these parameters using the MLE learning algorithms introduced in the next section. The purpose of the lower level Bayesian networks is to infer the variables used in the higher level networks. Since it is not possible to know the state of these variables from the processing devices inside the integrated tracker, we use these lower level Bayesian
networks in order to determine their states. We then use these estimated samples together with the data fed by the integrated tracker via the averaging device to learn the parameters and structure of the higher level Bayesian network. This way we have all the data we need to reconfigure and learn both the higher and lower level Bayesian networks in the adaptive situation assessment system.

The higher level of the situation assessment system consists of a Bayesian network that represents the most abstract quantities in the battlefield. The dependence relationships (existence of arcs in the corresponding Bayesian Network) among these variables, as well as the degrees of these dependencies (parameters in the corresponding Bayesian Network) are not fixed and change with time. These changes, however, happen very slowly with frequency \( f_4 = \frac{1}{\tau_4} (\tau_3 \ll \tau_4) \). The higher level Bayesian network’s structure is a direct function of the state of the battlespace, and therefore it changes when the state of the battlespace changes.

One of the possible structures of the higher level Bayesian network is shown in Figure 3, where \( P \) and \( A \) indicate friendly forces (protagonist) and enemy forces (antagonist) respectively. All the nodes are discrete-valued and are all partially observable except for the node situation that is completely hidden. The variable situation contains information about how well the friendly forces are doing in the battlespace conditions. Our goal in this stage is to estimate the state of the battlespace situation. The state of situation together with the information available to us from inferring other variables as desired would give a thorough assessment of the battlespace situation and enable us to generate relevant hypotheses about the battlespace conditions.

The training data for some of these nodes is inferred from the lower level Bayesian networks. The data for others (except the node situation) is fed from the integrated tracker through the averaging device. The averaging device computes the average of each incoming batch of data fed to it. The purpose of this device is to make sure the data samples fed to the higher level Bayesian network arrive at the same frequency. Since the state of the battlespace is time-varying, we need to adapt both the structure and the parameters of the higher level Bayesian network. We explain how this adaptation is done in the next section.

3 Multi-Timescale Reconfiguration of the Bayesian Networks for Adaptive Situation Assessment

In this section we present the Maximum-Likelihood Estimation (MLE) algorithms used in learning and reconfiguration of the situation assessment system. We first present the parametrical re-estimation algorithm in the context of a Bayesian network at the lower level; and then explain the reconfiguration algorithm used at the higher level network. The algorithms for re-estimating the parameters is the same in the higher and lower level Bayesian networks, and the algorithm to determine the correct structure is only executed at the higher level.

3.1 Reconfiguration of the Lower Level Bayesian Networks

At each instance \( k \), the integrated tracker outputs a data sample \( x^k \) which is a vector of dimension \( n \). As data samples \( x^k \) from the integrated tracker accumulate, the likelihood of the current model drops from its maximum since we calculate the likelihood of the model given the current data, but the models parameters had been maximized with respect to previous set of data, therefore after a while we need to change \( \Theta \) to get a model with an acceptable fit. This is done by learning the parameters of the Bayesian network periodically using the Expectation-Maximization (EM) algorithm. The EM algorithm is an iterative optimization approach that can be used to estimate some unknown parameter \( \Theta \) given data \( [3] \). As every data batch of size \( N \) arrives from the integrated tracker, we run the MLE algorithm explained below to recalculate \( \Theta \). We then use the new network \( G = \Theta \) (where \( G \) is fixed) as the current model until the next \( N \) data samples accumulate. We use Algorithm 1 to re-estimate the parameters of the Bayesian network.
Next we explain how the parameter $\Theta$ can be updated using the EM algorithm. Given a training set $D = \{x^1, ..., x^N\}$ of $N$ data samples, we would like to find the network $B$ that best matches the data in $D$. The log-likelihood of $B$ given $D$ is

$$L(B|D) = \sum_{k=1}^{N} \log(P_B(x^k))$$

(2)

since $G$ is fixed.

Since we have missing values and hidden variables in our system, we need to compute the maximum likelihood approximation for the network parameters, $\Theta$ using some iterative approach. Similar to Friedman [4], Chickering and Heckerman [5] and [6], we employ the EM algorithm to find a local maximum. It is worthwhile to introduce the concept of sufficient statistics [5] before we proceed to explaining the re-estimation algorithm. Let $N_X(x)$ be the number of cases where $X = x$, we call $N$ the **sufficient statistics** for $X$. When $X_i$ and all its parents are observed, the computation is the trivial addition of zeros and ones. Otherwise we need to use a Bayesian network inference algorithm to compute the probabilities, $E_{p(x|D,\theta,B)}(N_{x_i},x_{si})$ indicates the expected sufficient statistics given the current state of parameters, the model and the data.

If all data points $x^k$ in the training set $D$ are complete (i.e. they describe the values of all variables in each sample), it can be shown [7] that the likelihood is maximized when

$$\theta_{x_i|x_{si}} = \frac{N_{x_i},x_{si}}{N_{x_{si}}}$$

(3)

In the case of incomplete data, however, we need to calculate the parameters $\Theta$ using the EM algorithm. The **expectation** step of the EM algorithm enables us to estimate the expected sufficient statistics of the hidden variables based on the states of the observable variables. This is done using the following formula.

$$E_{p(x|D,\theta,B)}(N_{x_i},x_{si}) = \sum_{k=1}^{N} p(x_i,x_{si}|x^k,\theta,B)$$

(4)

where $x^k$ is a data sample. We then calculate the parameters using the expected sufficient statistics used in place of real sufficient statistics to find the MLE of the parameters. This is the **maximization** step of the EM algorithm. The maximum likelihood estimation of the parameters therefore can be calculated via the following formula [7]:

$$\theta_{x_i|x_{si}} = \frac{E_{p(x|D,\theta,B)}(N_{x_i},x_{si})}{E_{p(x|D,\theta,B)}(N_{x_{si}})}$$

(5)

It can be shown that the sequence of these iterations of expectation and maximization is guaranteed to converge to a local maximum [3].

### 3.2 Reconfiguration of the Higher Level Bayesian Network

We now explain how we adapt the higher level Bayesian network to cope with the changes in the battlespace. As mentioned at the beginning of the section, the algorithm for re-estimation of the parameters of this network is the same as the one for the lower level. Therefore, we directly proceed to describing the structural reconfiguration algorithm.

To decide which structure of the higher level network best represents the battlespace, we evaluate the minimal description length (MDL) score of all three networks and pick the one that results in the highest score. The MDL score is calculated using the following formula [8].

$$\text{score}(B|D) = L(B|D) - \frac{\log N}{2} \#(B)$$

(6)

where $\#(B)$ is the number of parameters in the network and $N$ is the size of the training set. It should be noted that the size of training set, $N$ has to be large in order for the MDL score system to produce a meaningful result. We chose $N \geq 100$ in our numerical studies.

Overall, the following algorithm can be used to reconfigure the higher level Bayesian network, where $N$ is the size of batches we perform parametrical and structural reconfiguration on. We call this Algorithm 2.

**Algorithm 2 Structural Re-estimation Algorithm**

Set $\Theta$ and $G$ to be the initial parameters and structure respectively.

Read data case $x^k$.

If $k \mod N = 0$

Run the EM algorithm over the last $N$ data cases to get new $\Theta$

Evaluate the MDL scores of all possible structures to and set $G$ to be the one with maximum score

Output $\Theta$ and $G$
the Bayesian networks. The hypotheses that are generated by the situation assessment system (see Figure 1) are fed to the higher level decision making systems and a high-level decision is made. This decision together with the state of variables from the lower level Bayesian networks is fed to the lower level decision system which makes a decision. The lower level decision together with the actions taken by the smart targets affects the battlespace. The changes in the battlespace are detected by the sensors and fed to the integrated tracker to be summarized and prepared for the next stage of situation assessment.

We first describe how hypotheses are constructed and how decisions are made using the hypotheses generated by the situation assessment system. To generate hypotheses, some (or all) of these variables that describe the battlespace situation best are picked. Each combination of their states is then formed, and is called a hypothesis. These variables (that can belong to either higher or lower level Bayesian networks) are chosen in the way that describes the situation of the battlespace best.

We define a hypothesis, \( h_k \) (where \( k \) is the time index) to be a vector consisting of these variable. In this vector, St is Situation, P is Potential damage, T is threat, Sr in Strength and I is Information.

\[
h_k = (St, P, T, Sr, I)
\]

(7)

The total number of different hypotheses is therefore equal to the product of the number of states these variables can take. If situation takes \( m_1 \) states, potential damage takes \( m_2 \) states, threat takes \( m_3 \) states, strengths takes \( m_4 \) states and information takes \( m_5 \) states, the total number of hypotheses \( m \) would be

\[
m = \prod_{i=1}^{5} m_i
\]

(8)

When a hypothesis has been chosen, it is deterministically mapped to a set of actions. There are two sets of actions, where one set corresponds to the variables at the lower level decision system and the other to the higher level. The higher level decision is based on the states of the variables at the higher level Bayesian network. The decisions at the lower level are based on the states of the variables at the lower level Bayesian network together with the decision made at the higher level. We map these \( m \) hypotheses deterministically to the set of higher level actions. By a deterministic map we mean there is no randomization in the process of mapping, and therefore a certain hypothesis is always mapped to a certain action. The higher and lower level actions are summarized in Table 1.

The higher level action set, in turn, is mapped to the lower level action set. The lower level action set consists of actions that are less abstract and affect the battlespace and targets directly. Each action at the higher level action set can be translated to several actions in the lower level. For example the action attack at the higher level can be followed by any of the 6 actions at the lower level. Only one low level action is taken at each decision epoch. Low level decisions affect the targets, their responses and other physical entities in the environment. They also trigger a response from the enemy forces that are assumed to be smart targets. After the enemy forces respond, the sensors collect data from the battlespace and the situation assessment system is reconfigured.

We now explain how our decisions affect the battlespace and then proceed to explaining how smart target’s behavior changes the battlespace. A decision affects the battlespace attributes \( V_i \) with probability \( p_i \), where \( p_i \) depend on various quantities such as environmental conditions, the effectiveness of the response and so on. \( V_{ij} \) denoted the variable \( V_i \) in the Bayesian network \( j \) where \( 1 \leq j \leq r \) where \( r \) is the number of lower level Bayesian networks and in our case \( r = 3 \) and also \( 1 \leq i \leq n_j \) where \( n_j \) is the number of nodes in Bayesian network \( j \). \( p_{ijk} \) denotes the probability of going to state \( s_{ijk} \) for variable \( V_{ij} \). \( q_j \) denoted the probability of an enemy aircraft killing a friendly aircraft. This table would go on to list all the actions and the variables affected by them and the probabilities of their new states. To save space, we only gave an example of how the table is constructed instead of including the entire table. It should be noted that the numerical values in Table 2 are for illustration purposes only.

In the table, \( a_l \) indicates the lower level action taken, \( V_{ij} \) is the variable(s) affected by this action, \( s_{ijk} \) is the state of the variable \( V_{ij} \). \( p_{ijk} \) is the probability of \( V_{ij} \) being at state \( s_{ijk} \), and \( q_j \) is the probability of destroying an aircraft of the opposite forces.

We now model the response of enemy forces that we assume are smart targets. We expand the definition of smart targets in [9] (which defines them as targets that are capable of hiding from sensors) to include targets whose behavior changes with the action we take. The response of these smart targets is modeled as completely random and the policy that governs their response is not modeled. The reason is that these policies are not known to us, so we can view their decisions as

<table>
<thead>
<tr>
<th>Table 1: Higher and Lower Level Action Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action Set</strong></td>
</tr>
<tr>
<td>Attack</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Defend</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Gather Information</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Do Nothing</td>
</tr>
</tbody>
</table>
Table 2: An example of the lookup table used in determining the variables that change after an action taken by friendly forces

<table>
<thead>
<tr>
<th>Variables affected</th>
<th>$a_1$</th>
<th>$V_{ij}$</th>
<th>$p_{ijk}$</th>
<th>$s_{ijk}$</th>
<th>$q_e$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{11}$</td>
<td>0.3</td>
<td>0.2</td>
<td>0.5</td>
<td>0.2</td>
<td>0.8</td>
</tr>
<tr>
<td>$V_{23}$</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>$V_{15}$</td>
<td>0.3</td>
<td>0.2</td>
<td>0.5</td>
<td>0.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

if generated by a completely random source. A similar table (as Table 2) is constructed to reflect the effect of enemy’s action on the battlespace environment. The only difference is that we are not aware of the enemy’s decision and the course of reasoning behind it, so the first column of the table, $a_1$, is empty. The only thing we observe is the effect of the enemy forces’ decision on the battlespace variables. For the purpose of simulation, we generate these responses probabilistically, but do not infer about the enemy action nor do we simulate their decision-making system and how they perceive the battlespace.

Without loss of generality, we assume that the responses from the friendly and enemy forces happen at even intervals. So if in total $N$ data samples are collected by the integrated tracker, the first $\frac{N}{2}$ comes from the decision the friendly forces have made and its effect on the battlespace environment and the next $\frac{N}{2}$ is generated by the actions of the enemy forces. These $N$ data samples are fed into the lower level Bayesian network by the integrated tracker. We assume all these $N$ data samples are equally valuable and the ones that have been generated at an earlier time are given the same weight as the ones that are generated at a later time.

The friendly and enemy forces’ actions also affect the number of friendly and enemy aircrafts. We assume that $\alpha_k$ is the number of friendly aircrafts and $beta_k$ is the number of enemy aircrafts at time $k$. $\alpha_k$ and $beta_k$ are Markov chains and are explained more thoroughly in the next section. Every action taken by the friendly or enemy forces causes $\alpha_k$ and $beta_k$ to change. These probabilities are provided by field experts and stored in the lookup Table 2.

In should be noted that all numerical values in Table 2 and also affected variables and their future states can be changed to suit the circumstances in different battlespace conditions without hurting the structure and design of the algorithm itself.

4.1 A Stochastic Feedback Algorithm to Model the Battlespace Dynamics

We now present the algorithm we devised to model battlespace dynamics as a stochastic feedback system. We described a feedback system that models the battlespace dynamics in the previous section. In this algorithm, $\alpha_k$ and $\beta_k$ are the number of enemy and friendly aircrafts respectively. $N$ is the number of samples in each learning batch (or batch size) used by the lower level Bayesian networks. $M$ is the batch size used in learning the higher level Bayesian network.

Algorithm 3 Algorithm for Modeling the Battlespace

Get $N$ data samples from the integrated tracker
While ($\beta_k > 0$ and $\alpha_k > 0$)
   For i=1:M
      Execute MLE algorithm to find the parameters of the lower level networks
      Find the MLE for missing nodes and take the average over $N$ samples
   End
   Execute MLE algorithm to find the parameters and structure of higher level BN
   Find the state of any missing variable
   Take the average over the $M$ data points
   Read the states of the hypothesis variables from the output of the higher level Bayesian network.
   Determine the most probable hypothesis using the state of hypothesis variables.
   Choose the high level decision from high level action set $A_{H}$ (Table 1) based on the hypothesis.
   Choose the low level decision from the low level action set $A_{L}$ (Table 1) based on the high level action and the state of variables in the lower level Bayesian Networks.
   Use lookup tables (Tables 1 and 2) to get new states of variables that are changed and the probability of kill for enemy and friendly aircrafts
   Generate $\frac{N}{2}$ data samples.
   Reduce the number of enemy aircrafts with probability taken from the lookup table
   To model enemy response, look up the affected variables and their new state from the lookup table
   Generate another $\frac{N}{2}$ samples.
   Reduce the number of friendly aircrafts with probability taken from the lookup table.
   Feed the resulting $N$ samples to the lower level Bayesian network.
End

Algorithm 3 enables us to model the battlespace dynamics and changes that occur as time passes. It further enables us to model the actions taken by friendly forces and smart targets and examine their effects on the battlespace. This will help us explore different courses of action in a battlespace situation and also to plan the battle beforehand by exploring what would the most probable outcome of certain actions be.

4.2 Markovian Analysis of Battlespace Dynamics

In this section we present a mathematical model that helps us analyze the dynamics of the simulated battlespace. This model is based on the situation assessment based stochastic feedback system presented in the
previous section. We first discuss this model and then proceed to examining some battlespace quantities such as expected duration of the battle and probability of winning for each side.

We model the feedback system as a vector Markov chain. The variables that comprise the states of the Markov chain are the same variables used in hypothesis generation vector $h_k$ in (1) augmented with the two quantities $\alpha_k$ and $\beta_k$. We denote the resulting process $Z$. Again, the hypothesis can contain some or all of the variables in the higher or lower Bayesian networks. For each of these quantities, namely, $h_k, \alpha_k$ and $\beta_k$ we have:

$$P(h_{k+1}|h_k, \alpha_k, \beta_k, h_{k-1}, \alpha_{k-1}, \beta_{k-1}, ...) = \pi_{h_k}(\alpha_k, \beta_k)$$

$$P(h_{k+1}|h_k, \alpha_k, \beta_k)$$

$$P(\alpha_{k+1}|h_k, \alpha_k, \alpha_{k-1}, \beta_{k-1}, ...) = \pi_{\alpha_k}(\alpha_k, \beta_k)$$

$$P(\beta_{k+1}|h_k, \alpha_k, \beta_k, h_{k-1}, \alpha_{k-1}, \beta_{k-1}, ...) = \pi_{\beta_k}(\alpha_k, \beta_k)$$

$$P(\beta_{k+1}|h_k, \alpha_k, \beta_k)$$

The first of these equalities hold since at any given time $k$, all the current information about the battlespace is summarized in the hypothesis, $h_k$, and this hypothesis depends on the action taken by the friendly and enemy forces in the previous epoch, which in turn depends on the hypothesis (i.e. the state of battlespace) at time $k-1$ and the number of enemy and friendly aircraft. In other words, the past states of this process are summarized in the current state of the process, and therefore, it is a Markov chain. The same reasoning can be made for $\alpha_k$ and $\beta_k$, and we can conclude that those are Markov chains as well. Based on this, the augmented process, $Z$ is also a vector Markov chain.

The reason we can model the behavior of this stochastic feedback system as a Markov chain is as follows. We made decisions based on the inferences made by Bayesian networks. These Bayesian networks do not add any delay to the data samples provided by the integrated tracking system. The integrated tracking system in turn does not add any delay to the data provided by the sensors. Also, the actions in one decision epoch are in phase with the sensor readings, since the integrated tracking system and the Bayesian networks do not add any delay to the data and only process it. The effect of the decisions on the battlespace environment, however, is taken into account in the next decision epoch, so the state of these readings depend on the decisions made in the previous epoch and therefore on the state of sensor readings in the previous epoch in a probabilistic manner. Therefore, we can say that the dynamics of sensor readings and also the variables that comprise the Bayesian networks are all Markovian.

The result is a therefore a vector Markov chain with a transition probability matrix of size $m(a_0 + 1)(\beta_0 + 1) \times m(a_0 + 1)(\beta_0 + 1)$ since we need to take into account states with $\{0, ..., a_0\}$ and $\{0, ..., \beta_0\}$. The problem now is to estimate the expected time till absorption of this Markov chain. The absorbing states are states in which $\alpha_k$ or $\beta_k$ are zero. An absorbing state is a state that if the Markov chain enters it, it would remain there forever. The absorbing states in our problem are the states at which either $\alpha_k$ or $\beta_k$ are zero. The reason for this is that when a side completely dies out the battles stops. The expected time till absorption, therefore, would be the expected duration of the battle. The mean time till absorption of a Markov chain can be calculated as follows:

$$v_i = 1 + \sum_{j=0}^{s-1} P_{ij}v_j \text{ for } i = 0, 1, ..., s - 1$$

where $v_i$ is the expected duration of the battle if we start in state $i$ and $P$ is the Markov chain transition probability matrix. We assume the Markov chain has $N_Z$ states, where $0, ..., s - 1$ are transient and states $s_1, ..., N_Z$ are absorbing. We define $T$, the random absorption time to be

$$T = \min\{n \geq 0; X_n \geq s\}$$

We also define $v_i$ to be the expected time till absorption starting at state $i$. It would therefore take the following form:

$$v_i = E[T|X_0 = i]$$

For more information on this derivation see [10].

Another quantity of interest is the probability of winning the battle for each side. It is calculated using the invariant distribution of the Markov chain presented above. The invariant distribution of a Markov chain with transition matrix $P$ is calculated using the system of linear equations below:

$$\pi_i = \sum_{k=0}^{N} \pi_k P_{ki} \text{ for } i = 0, 1, ..., N$$

and

$$\pi_0 + \pi_1 + ... + \pi_N = 1$$

The probability of winning for friendly forces ($p_{win}$) can now be calculated as

$$p_{win} = \sum \pi_i \text{ i : } \beta_k \text{ in state i is zero}$$

The invariant distribution indicates the probability of finding the process in any of the states after the process had been in operation for a long time. Since the Markov chain in our problem has both transient and absorbing states, the probability of the process being at the transient state after a long time is by definition zero and the only the probabilities of the absorbing states are nonzero. Since all the states for which $\alpha_k$ or $\beta_k$ are positive are transient, and the states for which $\alpha_k$ or $\beta_k$ are zero are absorbing, the invariant distribution would be equivalent the distribution of probabilities of the battlespace being at each of the recurrent states. Based on this, summing over the states for which $\alpha_k$ is positive results in the probability of winning for friendly forces, since we know either $\alpha_k$ or $\beta_k$ has to be zero, and if $\alpha_k$ is not, $\beta_k$ has to be zero, and the sum of these probabilities would indicate the probability of $\alpha_k$ being positive, that is the probability of winning for friendly forces.
Table 3: Mean Duration of the Battle and its simulated Variance for various Starting States

<table>
<thead>
<tr>
<th>Initial State Z</th>
<th>μ</th>
<th>σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4,2,2,2,4)</td>
<td>9.23</td>
<td>1.28</td>
</tr>
<tr>
<td>(0,0,0,0,4)</td>
<td>6.72</td>
<td>0.97</td>
</tr>
<tr>
<td>(2,1,1,1,4)</td>
<td>13.12</td>
<td>2.11</td>
</tr>
</tbody>
</table>

5 Numerical Examples

We now present numerical examples for the stochastic feedback system and Algorithm 3 here. We assume initially both α₀ and β₀ are equal to 4, i.e. each side has four aircrafts. We also assign m₁ = 4 and m₂ = m₃ = m₄ = m₅ = 3 which gives us the total number of hypotheses of m = 324. The states are valued as {0, ..., m₁ – 1} for each i. It should be noted that a higher number for a state indicates a better condition, for example a “very good” situation is shown with 4 whereas a “bad” situation is indicated by 0. Similarly, a high state value for threat indicates low threat and a small state value means high threat, which is a bad condition. In this case the total number of states would be 5 × 5 × 324 = 8100. The problem now is to estimate the expected time till absorption of this Markov chain. This is done using (14). The absorbing states would be the same, i.e. states in which either αₖ or βₖ are zero. We calculate the mean duration of the battle for several initial states (see Table 3). The average duration of the battle together with its variance for different initial states is listed in the table (μ is the calculated mean duration of the battle and σ is the simulated variance of the duration.) The variance is calculated over 50 runs. As shown in the table, the mean duration of the battle is maximum when the friendly forces start at a moderately “good” state and smallest when they start in a “bad” state. We also calculate the probability of winning for each side, which is the invariant distribution of the Markov chains. It turns out that the probability of winning (17) for friendly forces p힘 is 0.784 and the probability of winning for enemy forces is 0.216 regardless of what state we start in.

6 Conclusion

In this paper we have presented a situation assessment system that is capable of adapting itself to the changes that occur in the battlespace environment. We have used Bayesian Networks in this situation assessment system and developed novel algorithms for their learning, reconfiguration and adaptation to changes in battlespace environment as time passes. We have also designed a stochastic feedback system which makes decision that are constrained by inferences made by the situation assessment system. These decisions are made on two levels of abstraction using the hypotheses provided by the Bayesian networks. Furthermore, we have introduced a Markov chain analysis of the battle and provided numerical examples to show the effectiveness of our approach in battlespace modeling and decision making.

References


