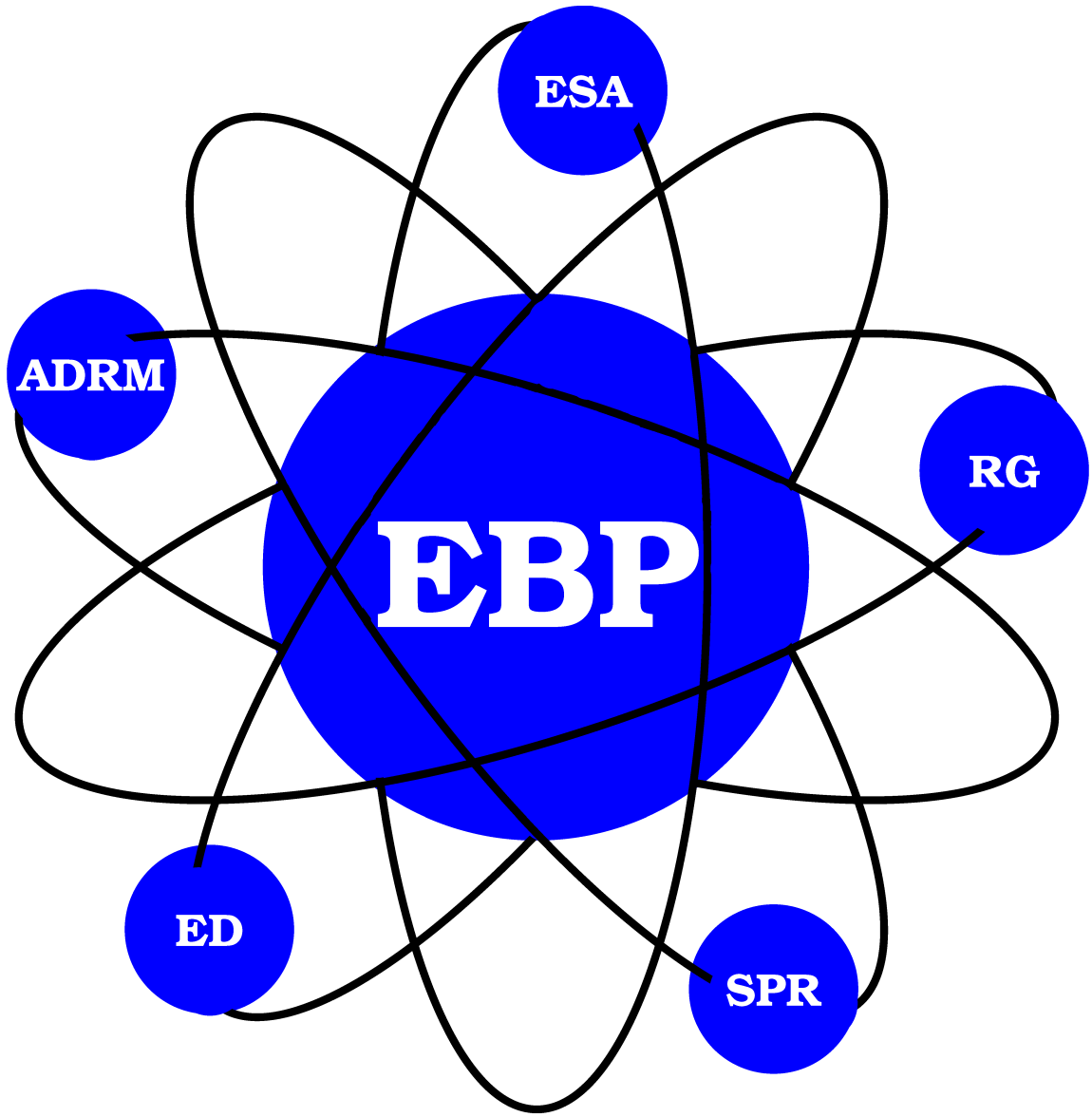
FOI

Simulation-based Decision Support Evaluating Operational Plans

Final Report

JOHAN SCHUBERT, FARSHAD MORADI, HIRAD ASADI,
LINUS LUOTSINEN, ERIC SJÖBERG, PONTUS HÖRLING,
ANNA LINDERHED, FRIDA HINSHAW, DANIEL OSKARSSON



FOI-R--3635--SE
ISSN 1650-1942 January 2012

Johan Schubert, Farshad Moradi, Hirad Asadi, Linus Luotsinen, Eric Sjöberg, Pontus Hörling, Anna Linderhed, Frida Hinshaw, Daniel Oskarsson

Simulation-based Decision Support Evaluating Operational Plans

Final Report

Titel	Simuleringsbaserat beslutsstöd för utvärdering av operativa planer -Slutrapport
Title	Simulation-based Decision Support Evaluating Operational Plans - Final Report
Rapportnr/Report no	FOI-R--3635--SE
Månad/Month	Januari/January
Utgivningsår/Year	2012
Antal sidor/Pages	55 p
ISSN	1650-1942
Kund/Customer	Försvarsmakten / Swedish Armed Forces
Forskningsområde	1. Beslutsstödssystem och informationsfusion
FoT-område	Modellering och simulering
Projektnr/Project no	E36710
Godkänd av/Approved by	Lars Höstbeck
Ansvarig avdelning	Informations- och aerosystem

Sammanfattning

I denna rapport beskriver vi simuleringsbaserade beslutstödstekniker för utvärdering av operativa planer inom effektbaserad planering. Med ett beslutsstödsverktyg kan utvecklare av operativa planer bedöma tusentals alternativa planer mot möjliga händelseutvecklingar och avgöra vilka av dessa planer som kan uppnå ett önskat sluttillstånd. Syftet är att förstå konsekvenserna av olika planer genom simulering och utvärdering. Operativa planer beskrivs enligt konceptet för en effektbaserad syn på operationer som en uppsättning aktioner och effekter. Vi kan ha flera olika alternativa sätt att utföra varje aktion. Tillsammans utgör de alla möjliga planer, som representeras som ett träd av handlingsalternativ som kan genomsökas för att finna den mest effektiva följden av alternativa för alla aktioner. Som ett testfall använder vi en expeditionär operation med en plan omfattande 43 aktioner och totalt 109 alternativ för dessa aktioner, samt ett scenario med 40 gruppaktörer som var och en beskrivs av 15 parametrar. Beslutsstöd till planerare ges av flera metoder för att analysera effekterna av en plan gentemot de 40 aktörerna, exempelvis genom att visualisera flera planers tidsserier över avståndet till sluttillståndet och visualisera tidsutvecklingen för alla aktörers tillstånd för den bästa planen syftande till att ge planerare en översikt över planens prestanda. Detaljerat beslutsstöd ges genom observation av de mest inflytelserika aktionerna med hjälp av känslighetsanalys och analys av regressionsträd. Slutligen, lära vi gränserna som en operation inte får överskrida utan risk för drastiskt misslyckande.

Nyckelord: Datorsimulering, dataanalys, beslutsstödssystem, beslutsträd, planering.

Summary

In this report we describe simulation-based decision support techniques for evaluation of operational plans within effects-based planning. With a decision support tool developers of operational plans are able to evaluate thousands of alternative plans against possible courses of events and decide which of these plans are capable of achieving a desired end state. The purpose is to understand the consequences of different plans through simulation and evaluation. Operational plans are described in the effects-based approach to operations concept as a set of actions and effects. For each action we may have several different alternative ways to perform the action. Together they make up all possible plans, which are represented as a tree of action alternatives that may be searched for the most effective sequence of alternative actions. As a test case we use an expeditionary operation with a plan of 43 actions and a total of 109 alternatives for these actions, and a scenario of 40 group actors each described by 15 parameters. Decision support for planners is provided by several methods analyzing the impact of a plan on the 40 actors, e.g., by visualizing multiple plan end state time series and visualizing actors time development for the best plan in order to give planners a performance overview. Detailed decision support is provided by observation of the most influential actions using sensitivity analysis and regression tree analysis. Finally, we learn the boundaries that an operation must not move beyond without risk of drastic failure.

Keywords: Computer simulation, data analysis, decision support systems, decision trees, planning.

Contents

1 Introduction

In this report we develop simulation-based decision support through an event-based simulation that model military operational plans according to effects-based planning (EBP) [1]. The methods developed can be used in an incremental manner by testing the plans as they are developed step-by-step and new activities are added. How we model a phenomenon depends on the purpose of the model and the questions we want to answer. Since our simulation system aims to support decision-making within an effects-based approach to operations (EBAO) the modeling has to be based on EBAO concepts as a set of effects and actions that together will lead to a desired military end state. Using a decision support tool, a decision maker is able to test a number of feasible plans against possible courses of events and decide which of those plans is capable of achieving a desired military end state. The purpose is to evaluate plans against a large set of actors and understand their consequences through simulating the events and producing outcomes which result from making alternative decisions regarding actions. Each plan consists of many actions, where several actions can be performed in a number of alternative ways. We model the plan and evaluate alternative plan instances on how well they are able to drive the entire state of the simulation model, simulating a large set of actors, towards a predetermined military end state. These plan instances are evaluated as to their performance and clustered into clusters where all plan instances have both common characteristics and outcomes. The idea is that these clusters, whenever they contain plan instances of good performance, are a robust set of alternative plans that can be used for minor dynamic re-planning whenever necessary.

Actors and actions are modeled using a scenario used by the Swedish Armed Forces in their Combined Joint Staff Exercises, and multinational "Viking" exercises. The actions of the plan are simulated together with all actors and their reactions and possible follow-on interactions. As the actions may have several different alternative ways they can be carried out, together these alternatives span-up an action tree. This tree is searched where each level in the tree corresponds to an action and each node in the tree is an alternative for that action. As the action tree is searched, each node is evaluated by the simulator and results are stored. By using search to guide the tasks of the simulator we let the simulator work in a manner that achieves maximum information value gain. In an experiment we simulate 10 000 plans out of 2.164×10^{23} possible plans. Simulated plans that are similar in both their structure and in their consequences are clustered together. These plans make up a robust set of similar plans that constitute ready alternatives should dynamic re-planning be necessary as the situation evolves.

Decision support is achieved through a series of statistical analysis, information fusion, machine learning, and information visualization techniques. For example, we develop methods for effects and end state times series visualization for easy overview over the time development of several alternative plans as action-by-action is being executed.

We develop information fusion explanation functions for simulation-based decision support for evaluation of military plans in expeditionary operations. Primarily, this methodology highlights the dangerous options in an operational plan, leaving the decision maker free to focus his attention on the set of remaining actions. By systematically varying one action at a time keeping all the other actions unchanged in a series of simulations, we are able to perform a sensitivity analysis for each action in the plan based on the change in evaluation score of the plans. This sensitivity analysis shows the relative level of importance of making the correct selection of alternative for each action. Using the explanation function, a decision maker becomes informed as to which actions of the plan are crucial to its success.

To differentiate between minor re-planning and whenever major re-planning becomes necessary in order to avoid drastic negative consequences of plans that begin to deviate substantially from the initial planning, we adopt indicators as warning bells. An indicator is the boundary between two clusters beyond which drastic changes can occur. We learn

boundaries from simulated data from alternative plan instances of an expeditionary operation, beyond which drastic changes can occur. We provide decision support during execution of a plan by calculating the distance from the plan to the closest boundary step-by-step as action-by-action is being executed. By visualizing the change in distance during execution a commander may observe if the operation is approaching a boundary beyond which outcomes may be uncertain.

This report describes a five year research effort performed at the Swedish Defence Research Agency 2008-2012 on developing simulation-based decision support for plan evaluation of operational plans constructed experts.

In Sec. 2 we describe the effects-based planning approach. In Sec. 3 we present an overview over the Bogaland scenario used for experimentation. In Sec. 4 we present a simulation control approach where a decision maker can focus the attention of the simulator. We then model actors and actions (Sec. 5) and develop a simulation methodology (Sec. 6). In Sec. 7 we develop a decision support methodology and in Sec. 8 we analyze simulation results. Finally, in Sec. 9 we provide a discussion of the approach and in Sec. 10 draw conclusions.

2 Effects-based planning

How we model a phenomenon depends on the purpose of the model and the questions we want to answer. Since our simulation system aims to support decision-making within an effects-based approach to operations (EBAO) [2][3] the modeling has to be based on EBAO and the concepts used within it, such as plan, action, effect, end state, etc. EBAO is a military approach to the management and implementation of efforts at the operational level. According to the United States Joint Forces Command (USJFCOM) EBAO are "operations that are planned, executed, assessed, and adapted based on a holistic understanding of the operational environment in order to influence or change system behavior or capabilities using the integrated application of selected instruments of power to achieve directed policy aims" [4].

Within the framework of EBAO, EBP is a method for developing objectives and effects to be achieved through a series of synchronized actions within a military operational plan, conceptually developed starting top-down from a desired end state. The methodology in EBP is iterative in nature where the development of the plan is made step-by-step and tested as it is gradually emerging. To provide decision support for this planning work, we develop methods that can be used iteratively when successively modeling different elements of the plan and testing them by simulation and evaluation against a scenario with operators' models that reacts to the execution of plan elements. It is possible to measure the change in state of all the actors in relation to the desired end state.

A control theory model of EBP [5] is shown in Fig. 1. As input we have the required situation R_s which is compared with the current situation C_s received from assessment. The first process is an end state analysis (ESA), followed by effects development (ED). Initially when there is no operation the military end state defines the goal of the operation. Later when a campaign assessment is carried out, the comparison between R_s and C_s may require further analysis in ESA. The output from ED is the required effects R_e which is compared with the current effects C_e, also received from assessment.

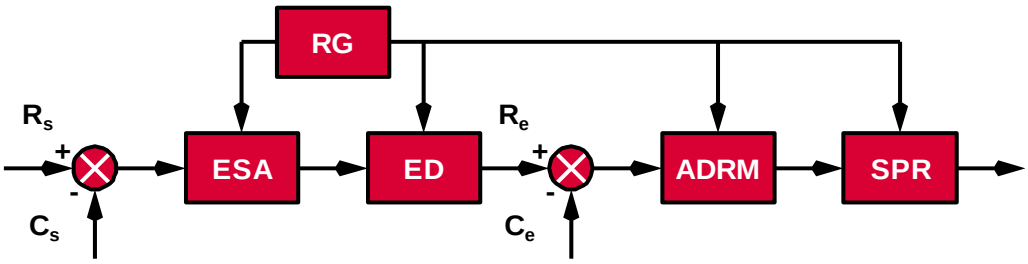


Fig. 1. Effects-based planning.

In terms of this model the focus of simulation-based decision support is primarily on effects development (ED).

In terms of the Allied Command Operations Comprehensive Operations Planning Directive (COPD) [6] we focus the simulation-based decision support on generation and testing of alternatives at Joint Force Command (JFC) Operational Concept Development, Fig. 2 (JFC Phase 4a). This does not exclude the use of these methods on an earlier strategic level.

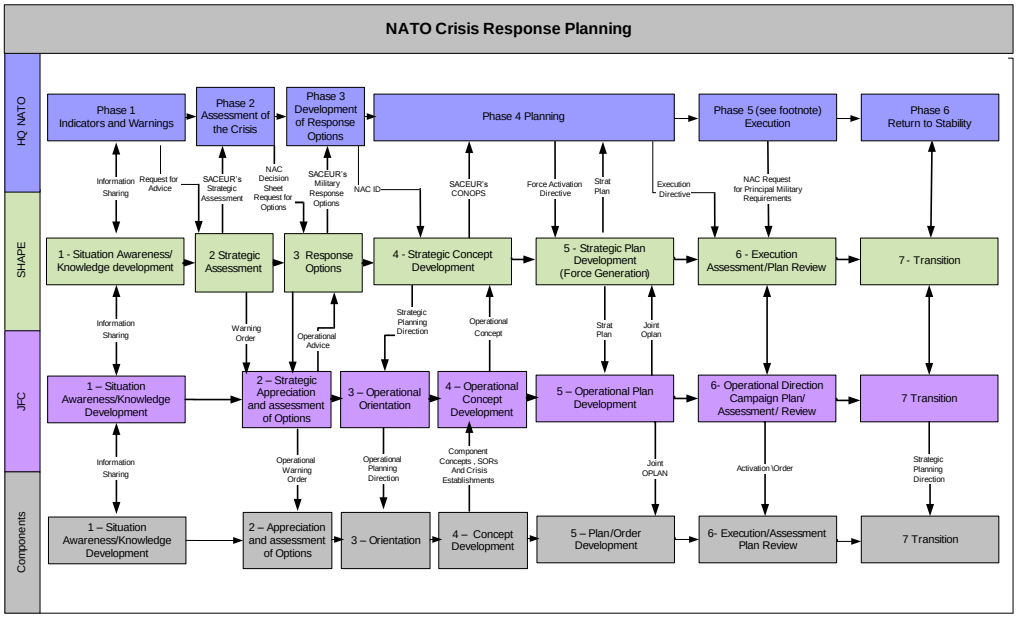


Fig. 2. Operational level crisis response planning (COPD).

3 The Bogaland scenario

We make use of the same scenario that has regularly been used by the Swedish Armed Forces in the Combined Joint Staff and "Viking" Exercises. The scenario comprises several fictitious countries, two of which, Xland and Bogaland, have been described in-depth. Background histories offer explanations to why and how sentiments, stances, identities, loyalties, economic dependencies and inequalities have evolved over time, occasionally resulting in shifts of power. Phenomena that are commonly found in conflict areas and post conflict areas have been embedded in scenario contexts that make the origins of the phenomena plausible, Fig. 3.

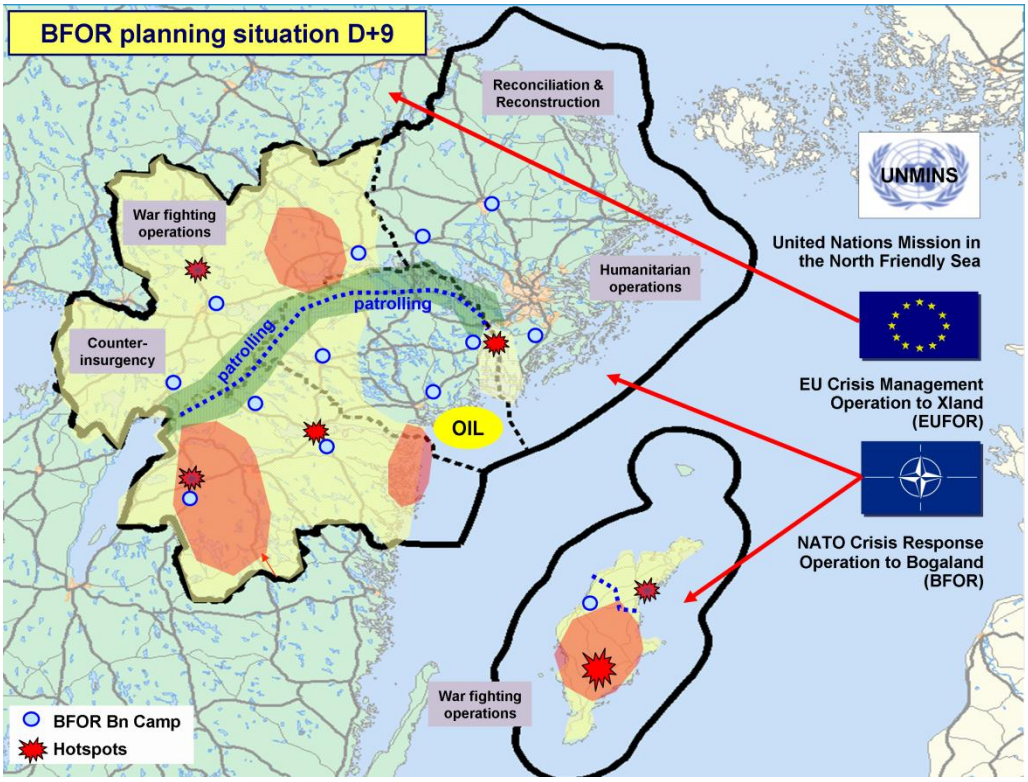


Fig. 3. The Bogaland test scenario.

In Xland demographic change constitutes a threat to the privileged majority group, and puts severe pressure on the government. The country has a constitution that does not give the fast growing minority group the same rights as the dwindling majority group. Irregular groups originating from the minority group have taken control of the rural parts of the country.

In Bogaland, a newly industrialized country, a civil war broke out ten years ago when discontent within the minority ethnic-religious group had reached very high levels. The root cause was increasing social stratification caused by what members of the minority group perceived as unjust distribution of revenues from a natural resource located in an area populated by the minority group. The civil war put an end to the exploitation of the resource, in this case oil, and revenues dropped to very low levels. The country was split into two parts, roughly along ethnic lines, with each part having its own government. A post-war economy evolved over the next decade, and several irregulars and insurgents are now challenging the incumbent presidents.

The incumbent presidents have signed a peace-agreement, and an international force, BFOR, is present to support the implementation of the agreement. Irregular groups in Bogaland seek to preserve or increase their influence by undermining the efforts of BFOR, the governments or competing irregulars. Two of the neighboring countries have much at

stake in the conflict, because of economic interests and shared identities with parties within Bogaland. Actors within these neighboring countries support irregulars in Bogaland.

4 Simulation control

The planning process we analyze corresponds to selecting a sequence of actions from sets of alternative actions. Most actions have between two and eight alternative ways of execution. A chosen sequence of alternative actions constitutes a plan for trying to reach a peaceful end-state in Bogaland. The number of possible plans can theoretically grow very large since each combination of alternative actions for the different actions will constitute a separate plan. In our test planning problem we have 2.164×10^{23} possible plans. Of course, in practice, many of these plans can be ruled out because they start out with a sequence of actions already evaluated that leads to early failure.

However, it is also possible to give the simulator instructions on how to select combinations of alternatives it should prefer during simulation. In this way the simulator can focus its attention on plans within the decision maker's interests. Three ways to do this is to focus the simulator's efforts towards plans with actions:

- executed within a specified geographical area,
- executed within a specified timeframe,
- that may lie outside the area and timeframe specified, if they strongly influence actions within the area and timeframe as described by a cross-impact matrix (CIM) [7][8].

The CIM is a matrix, set up for all actions where it is specified how much the actions counteract or support each other during execution due to resource conflict or one action laying the foundation for another one, etc. CIM's have been used for planning purposes in industry since the 1960's. The CIM is described more in detail in [7].

In the graphical user interface, we may accordingly make these selections as a preferred area of interest in a map, a timeframe in a Gantt chart, and an action group in a chart with actions grouped according to their inter-influencing in the CIM. Each of these three types of selections gives each action a weight between 0.0 and 1.0. The spatial and temporal weight, for each action, is calculated as the overlap between the selected focus area/timeframe, and the corresponding for each action. The final weight for an action, to be used for its importance in the simulation, is simply the product of these three weights. The simulator will focus its attention in relation to these weights. Fig. 4 shows the screen of this user interface.

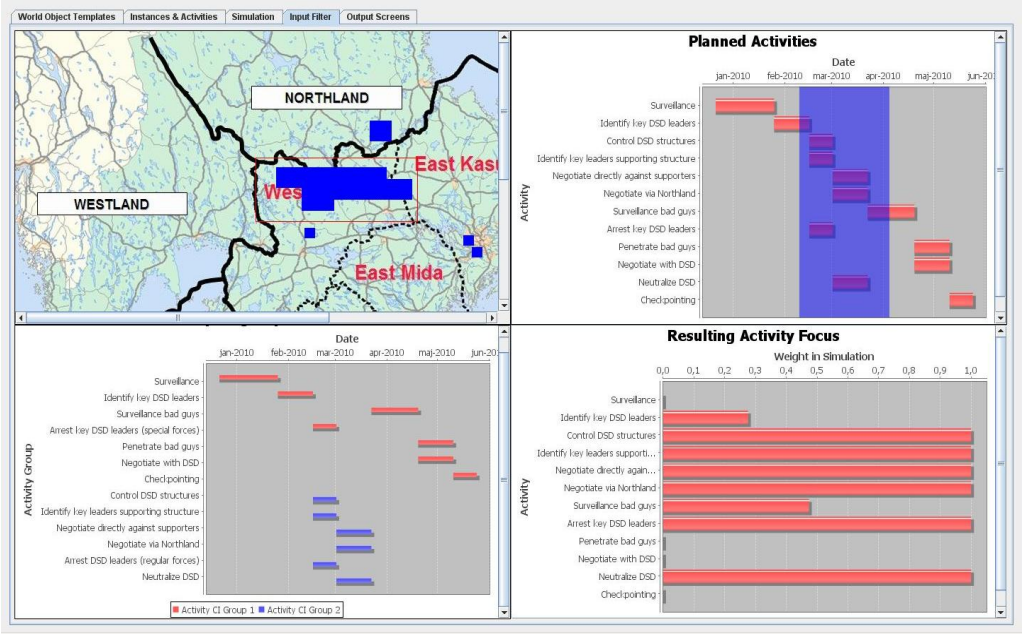


Fig. 4. The simulation control tab of the GUI. *Upper left:* Each action has an area where it is executed in Bogaland (blue filled rectangles). Selection of geographical focus area is done with a red rectangle. *Upper right:* Selection of focus timeframe (purple). *Lower left:* Selection of connected actions that are closely tied in the CIM. One group can be selected (here, purple or red) *Lower right:* Fused (product) weights for the actions which give their importance in the simulation.

5 Modeling actors and actions

In this work we employ an actor modeling approach targeted towards modeling aggregate entities of human groups and organizations such as civilians, armed forces, etc., in military conflict zones. The modeling approach uses a combination of Bayesian networks and rule-based methods that operate on state variables that have been selected to represent the characteristic properties of aggregate entities representing groups and organizations. Specifically,

- state variables are used to represent the actor's knowledge or beliefs about itself, other actors and the environment,
- Bayesian networks are used to model the behavior and action selection mechanism of the actor,
- rules are used to model actions and their effects on the actor's state variables.

5.1 State variables

State variables are used here to represent the actor's knowledge and beliefs about itself, other actors and the environment. We have separated the state variables into sets representing an actor's internal state and its relationships to others. Note that in our modeling approach all actors known to the actor, including it, are represented using separate sets of the abovementioned state variables. The purpose of the state variables is to provide a common knowledge representation that can be used when developing behaviors and actions as described below. The state variables presented in this section were identified using subject matter experts and chosen to represent a wide range of characteristics among groups and organizations in military conflict zones.

The internal state variables, which originate from previous work [1], are represented here by a vector I that contains 15 discrete state variables. The name, label ($A-O$) and a brief (non-exhaustive) qualitative interpretation for each variable value is presented in Table 1. The variables in the internal state vector are limited to four integer values [0, …, 3]. This design decision was made to keep the knowledge space of the actor relatively small which ensures that behavior and action modeling remains pragmatic and not too time-consuming. Also, such limitation significantly reduces the complexity in terms of search space when embedding actor models in real-time planning tools such as the one presented in this report.

Table 1. Internal state variables.

State variable		Interpretation of state variable values 0	1	2	3
Weapon power	A	Less than 0.1 brigade	0.25 brigade	1 brigade	4 brigades
Living conditions	B	Suffering, scarce resources, being killed by others	Suffering, scarce resources	Not suffering, limited resources	Not suffering, abundant resources
Stance	C	Submissive	Defensive	Defiant	Violent
Sympathizers	D	No supporters	Supported by marginal others	Supported by the local majority	Supported by the wider majority
Economy	E	< 1000 times GNP/capita	1000 – 10000 times GNP/capita	10000 – 100000 times GNP/capita	> 100000 times GNP/capita
Stability	F	Quick reduction in group size	Slow reduction in group size	Slow increase in group size	Quick increase in group size
Geographical dominance	G	At risk in the area	Can move and talk freely	Can impose restrictions on others	Can dominate others
Infrastructure	H	Man-to-man, word-of-mouth	Terrain vehicles, leaflets	Trucks, cellphones	Complete, Internet
Propaganda channels	I	Limited reach outside primary group	Reaches local communities	Reaches communities of similar identity	Reaches all types of communities
Social network	J	No ties	Ties to uncommitted	Ties to committed	Ties to highly committed
Reputation	K	Despised	Light-weight	Recognized	Highly regarded
Dissatisfaction	L	No grievance	Would like to see responsible for grievance fail	Prepared to use violence in act of revenge	Prepared to sacrifice life in act of revenge
Group feeling	M	Power struggle	Friction	Harmony	Cohesive
Ideological conviction	N	None	Little	Medium	High
Goal orientation	O	None	Preserving	Advance	Vision
Moral stance	P	Indiscriminate use of violence	Low barrier/concern for out-groups	Restricted but pragmatic use of violence	Violence only as a last resort in self-defence

Similarly to the internal state of an actor, its relationships to others are encoded in a relationship state vector *R* as illustrated in Table 2. Each row in the table represents the relationship of this actor towards another actor. Note that, unlike *I* which is fixed, the number of variables in *R* varies with the number of other actors *N* known to the actor. The relationship variables have four integer values [0, …, 3] which are interpreted as *enemy*, *suspicious*, *neutral* and *friendly*, respectively.

Table 2. Relationship state variables.

State variable		Interpretation of state variable values 0	1	2	3
Relationship_1	R_1	Enemy	Suspicious	Neutral	Friendly
⋮	⋮	⋮	⋮	⋮	⋮
Relationship_n	R_N	Enemy	Suspicious	Neutral	Friendly

Given the state variables described above we introduce the notation used in the remainder of this report. Another actor a_i known to the actor is represented by $\omega_i = \{I_{ij}, R_{in}\}$ where $j = \{A, B, \ldots, O\}$ and $n = \{1, 2, \ldots, N\}$. That is, I_{ij} represents actor a_i's internal state variable j, and R_{in} represents actor a_i's relationship to actor an. Given that the actor knows about N actors (including itself) its complete knowledge space is $\Omega = \{\omega_1, \ldots, \omega_N\}$.

Let's also introduce the concept of roles that is used here to generalize action and behavior modeling: the *initiator* role is assigned to the actor that initiates an action; the *target* role is assigned to the actor who's state variables are directly affected by the *initiator*'s action; and the *bystander* role is assigned to all other actors, other than the *initiator* and *target*, that may be affected by the action. Henceforth, when referring to the state variables of the *initiator*, *target* and *bystander* actors the subscripts *i*, *t* and *b* are used, respectively. For instance, I_{tA} refers to the internal state variable *A* of *target* actor a_t.

5.2 Behavior Modeling

Using the state variables we introduce a behavior modeling method. The behavior of an actor is in essence an action selection strategy implemented as a function that use as input the actor's state variables, Ω, and generates as output the alternative α to execute as shown in (1).

$$\alpha = f(\mathbf{\Omega}) \tag{1}$$

In this work a Bayesian network (BN) [9] approach has been adopted to model *f* using either subject matter expert knowledge in cases where too little or no data is available or using machine learning algorithms in cases where large data sets representing the historic behavior of an actor are available. We have primarily chosen to use BNs due to their:

- capability to graphically represent actor behavior using directed acyclic graphs (DAGs) which ultimately improves the general understanding of the model,
- capability to perform inference, or select actions, even in the presence of missing or uncertain information,
- modularity and re-usability.

The Bayes rule defined in (2) represents the core of any Bayesian modeling approach [9]. We have,

$$p(\alpha_n|\mathbf{\Omega}) = \frac{p(\alpha_n) \times p(\mathbf{\Omega}|\alpha_n)}{\sum_{m=1}^{M} p(\alpha_m) \times p(\mathbf{\Omega}|\alpha_m)} \tag{2}$$

Using the Bayes rule a probability value, the posterior, is calculated for each action available to the actor. Typically, the action with the maximum posterior is selected by the actor. This is however not always the case as will be discussed below.

From the Bayes rule it is clear that the *posterior* $p(\alpha_n|\ \Omega)$ of action α_n is calculated using the *prior* $p(\alpha_n)$ and *likelihood* $p(\Omega|\ \alpha_n)$ functions. The denominator, or the *evidence*, is a normalizing factor that spans all actions *M*. That is, using the Bayesian approach it is ultimately the prior and likelihood functions that the modeler manipulates or that the learning algorithm estimates to represent desired actor behaviors. The problem with Bayes rule is that one rarely can find enough data to model the likelihood function due to, in our case, the high dimensional state variable vector Ω. This is where BNs comes to rescue by introducing conditional independence between variables, hence, simplifying the likelihood estimation process.

At its simplest a BN is identical to the naïve Bayes classifier in which all variables in Ω are assumed to be conditionally independent. Using this assumption, Bayes rule can be reduced to (3),

$$p(\alpha_n|\mathbf{\Omega}) = \frac{p(\alpha_n) \times \prod_{k=1}^{K} p(\Omega_k|\alpha_n)}{\sum_{m=1}^{M} p(\alpha_m) \times \prod_{k=1}^{K} p(\Omega_k|\alpha_m)}, \quad (3)$$

where K is the dimensionality of Ω. The DAG of an example naïve Bayes classifier BN is presented in Fig. 5.

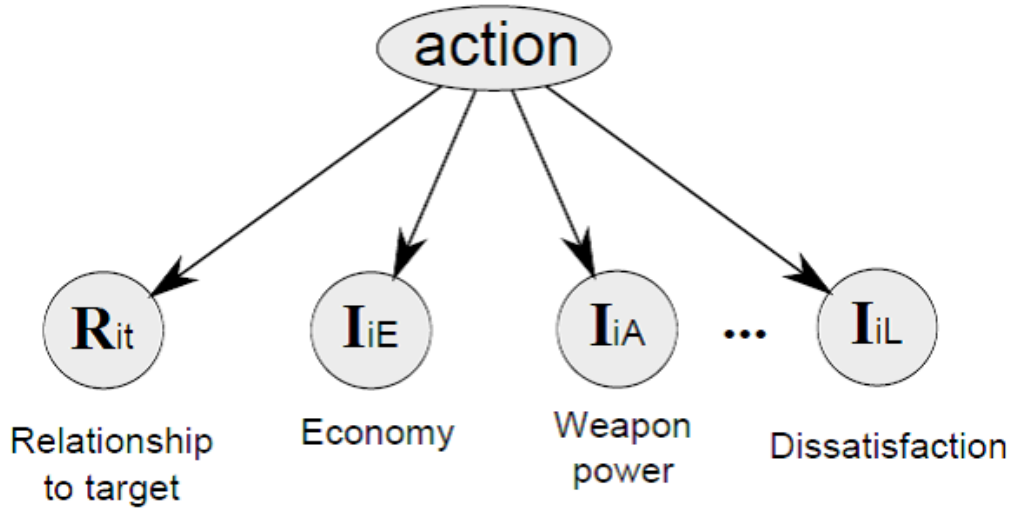


Fig. 5. Naïve Bayes classifier.

However, clearly not all variables in Ω are conditionally independent of each other. As an example, an actor a_i's dissatisfaction I_{iL} to another actor at is conditionally dependent on its perceived relationship R_{it} to a_t. A modified network incorporating this conditional dependency is presented in Fig. 6. Links between any two variables in the DAG indicates that there exists a conditional dependency between them. Many inference algorithms that are capable of calculating the probabilities at arbitrary nodes in arbitrary structured BNs have been discussed in the literature [10]. In this study we have chosen to use the algorithm presented in [11]. It is important to know that the time required to infer probabilities varies depending on the structure of the BN as well as the amount of evidence (or knowledge) that are known prior to inference.

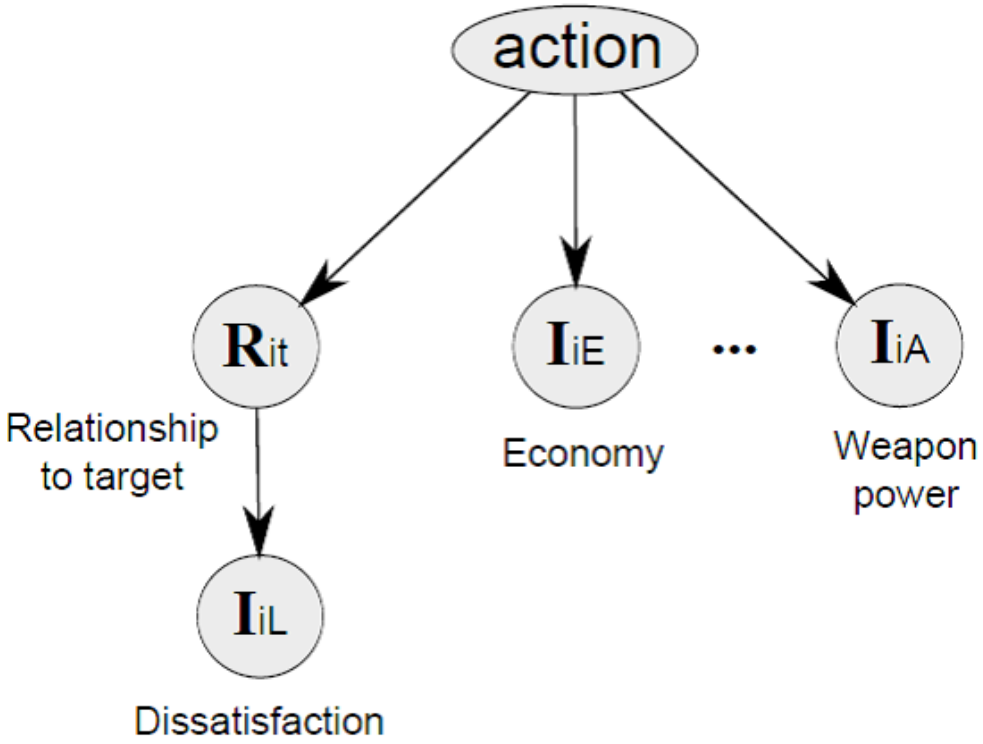


Fig. 6. BN classifier where I_{iL} is conditionally dependent on R_{it}. All other state variables are assumed to be independent of each other.

Using the probabilities inferred at the *action* node of the BN it is possible to select an action in several ways. Which action selection method to use is ultimately the modeler's choice. This actor model supports the following action selection methods:

- maximum a posterior (MAP),
- random draw.

The MAP approach simply selects the action with the maximum *posterior*. The random draw approach selects an action by randomly sampling the *posterior* values with respect to their proportions.

5.3 Action Modeling

Actions are the means by which an actor may alter its state, Ω. An action is represented here by a set of rules each consisting of a condition, the *if*-part, and a list of effects, the *then*-part, such that if the condition is *true* then the list of effects will execute, ultimately resulting in state variable changes. On the other hand, if the condition is *false* then none of the effects will execute.

In this work, subject matter experts have developed hundreds of rules modeling the following actions: *attack*, *neutralize*, *negotiate*, *support*, *protect*, and *nothing*. In addition to the rules governing the effects of actions, subject matter experts have developed global rules modeling phenomena such as the Stockholm syndrome and radicalization. Global rules are also used to introduce constraints that filter out invalid or unwanted state variable values.

The conditions (*if*-parts) of the rules are described using Boolean expressions. The effects (*then*-parts) of the rules are described using a function notation where *set*, *inc* and *dec* functions are used to set increment and decrement specific state variable values. For instance, *set*(I_{tA}, 1) assigns the value 1 to the target actor's internal state variable A.

Similarly, *inc*(I_{tA}, 1) and *dec*(I_{tA}, 1) increases and reduces the same value by 1 respectively. Table 3 illustrates the notation using an example rule that partially models the effects of the protect-action.

Table 3. Example rule partially modeling the protect action.

If	Then	Description
$\mathbf{I}_{tA} > 0 \wedge \mathbf{I}_{tC} > 2 \wedge \mathbf{I}_{tD} > 1 \wedge \mathbf{I}_{tG} > 1 \wedge \mathbf{I}_{tI} > 0 \wedge \mathbf{I}_{tJ} > 1 \wedge \mathbf{I}_{tK} > 1 \wedge \mathbf{I}_{tL} > 1 \wedge \mathbf{I}_{tN} > 0 \wedge \mathbf{I}_{tO} > 1$	set($\mathbf{I}_{tB}$,2) set($\mathbf{I}_{tG}$,1)	Update living conditions and geographical dominance of target.

6 Simulation methodology

The simulation contains models of participating actors and their initial states including a probability distribution for different actions they are capable of carrying out, environmental data, and the plan that is to be evaluated. A plan in this context is defined as a set of actions executed (sequentially or in parallel) by a military force intended to lead to a desired end state. Furthermore, the simulation scenario contains an event list which consists of actions derived from the other actors' agendas, and spontaneous/natural events. This list is dynamic and changes during the course of the simulation.

In order to describe the simulation process we define the system state S_n as the combination of all actors' state variables and all environment parameters. Now, consider action A_n. It transforms system state S_{n-1} according to $S_n = f(S_{n-1}, A_n)$, in the time interval $[t_{n-1}, t_n]$. The implementation of A_n is rarely instantaneous. Instead, it is an interaction between our own action, other actors' agendas and response operations, and other external events. Hence, our function $f(S_{n-1}, A_n)$ is designed as an event-driven simulation model in order to manage the complex interactions in a transparent manner. The events in this case are: launching of actions (our own or any other actors'), an actor's observations of initiated actions, and occurrence of an external event.

Furthermore the outcome of A_n can vary depending on the circumstances (e.g., the operation may even fail), which can be addressed by making the simulation stochastic, where the outcome of an action depends on a number of random variables drawn according to some given distributions. The disadvantage of this is that we can obtain a per se reasonable, but rather unlikely outcome, which would mean that we might needlessly throw out a mostly good plan. In order to avoid this outcome we use Monte Carlo simulations, thereby obtaining a frequency function of the entire outcome space.

A consequence of implementing the function $f(S_{n-1}, A_n)$ as an event-driven stochastic simulation model is that, although the state variables from the beginning are absolute values, after a completed action they will be represented by statistical distributions. Hence, we can choose to represent the initial states by statistical distributions as well. Similarly, the external events can be listed with typical probabilities for the actual operational theatre, season, etc.

We know that the goal of the simulation is to execute different plans and identify those plans that result in system states that are closest to our end state, i.e., has the shortest *distance* to it. Given the approach discussed above, the distance to the end state will be stochastic. Hence, by calculating the distance value in each Monte Carlo loop we create the distribution of this distance in the form of a histogram which approximates the frequency function. This means that the A^*- algorithm (described in the next section) needs to evaluate not only a single distance value, but also the importance of the spread in the given situation. A large spread around a small average value indicates that we are on track, but that this path is unstable and could easily lead to failure.

Our Monte Carlo simulation is therefore structured as follows, Fig. 7.

For each round of the Monte Carlo loop:
Initialize event list with our action A
Randomly draw the external events and add them to the event list
Randomly draw a starting state for each state parameter from resp. distribution
For each actor:
Randomly draw the next action from the current agenda and add to the event list.
For each event in the event list as long as time is less than tn:
Environmental parameters may change (which could generate new events).
For each actor (including "our own" operator "):
Note directly or indirectly through filtered or biased information
Analyse the information → internal state and resources are changing
Action repertoire is updated with new probabilities
Randomly generate the next action
Add a new action to the event list.
Save the results for each state parameter.
Create a summary of results for each state parameter in the form of a histogram, which serves as an approximation for resp. output distribution.

Fig. 7. Monte Carlo algorithm.

During the actual time interval $[t_{n-1}, t_n]$ our action A_n is initiated. Probable external events are in the same way chosen and placed in the event list according to their given distributions. The action A_n is observed via an information channel by the other actors immediately or eventually. Directly, or after a period of analysis which may be biased or colored by the information channel, the respective actor's state is changed, which can lead to a new set of probabilities in the action repertoire. An action from each actor's action repertoire is randomly chosen and placed in the event list. As the simulation proceeds and actions/events in the event list are executed new actions/events are added in the list (as the result of observations and reactions) until the end of the time interval is reached. Finally, a summary of the results for the state variables is created. These state variables are represented by histograms and serves as an approximation for the respective output distribution.

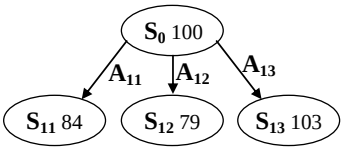
6.1 A*-search

The purpose of our simulation system is to search for a sequence of actions that best suits the decision maker's desired end state. However, we also want the simulation to be capable of suggesting an alternative solution at any moment in time. Hence, such a simulation system can neither be designed according to the principle of "breadth first search" nor "depth first search". In the former case it will take too much time before we reach a reasonably correct prediction. In the latter case we get stuck with just one plan, and will not have a general view when we are asked to forecast the best solution. Instead, a suitable approach in our case is to apply an A*-search algorithm. Below is the classic representation of the A*-search algorithm,

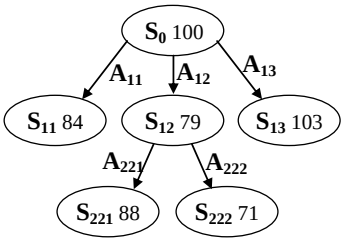
$$f'(n) = g(n) + h'(n) \tag{4}$$

where $g(n)$ is the total distance from the starting position to the current location, and $h'(n)$ is the estimated remaining distance from the current position to the goal (end) state. A heuristic function is used to create this estimate on how far it is to the goal state. The function $f'(n)$ is the sum of $g(n)$ and $h'(n)$. This is the current estimated shortest path $f(n)$ is the true shortest path which is not discovered until the A*-algorithm is finished.

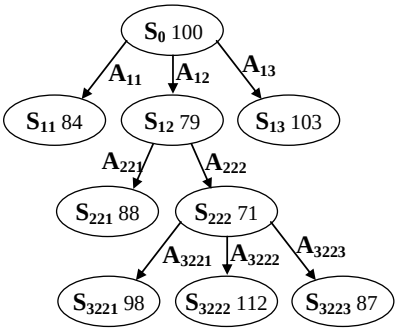
It means that, on the basis of a given system state, we simulate the effect of each alternative action in our plan, but only one step at a time. Doing so, for every alternative, we get a new system state whose distance to the desired end state is calculated. Given the alternative that is best, i.e., closest to our end state, we simulate the possible subsequent alternative actions that are provided, but only one step ahead in our action/event list. One of these alternatives leads to a condition that is closer than the others. However, it is possible that all the alternatives actually lead away from the target as seen by Fig. 8.



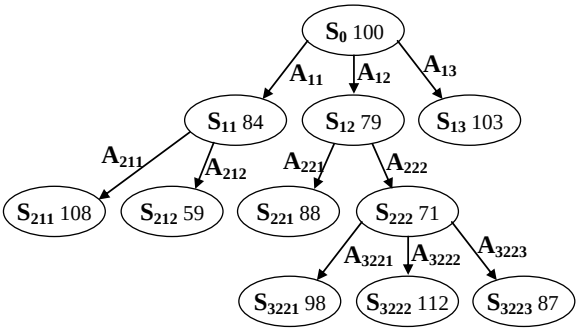
Step 1: From the initial state all available alternatives are simulated. S_{12} appears to be "closest" to the target.



Step 2: After execution of alternative activities that follow S_{12}, S_{222} is the "closest" to the target.



Step 4: From S_{222} all the alternative activities that are presented are executed. $S_{11,}$ which was calculated earlier appears to be "closest" now.



Step 4: Activities following S_{11} are now simulated and S_{212} is the "closest" and next to simulate.

Fig. 8. An example illustrating the four first steps in a simulation of a plan starting with initial system state S_0 with the *distance* of 100 to the desired end state. The available action alternatives A_x are executed successively in the currently most favorable plan option.

Therefore, we must also compare the new distance with the best of the distances that have been simulated and recorded in the previous simulation steps, but then had opted out in favor of a better sequence of alternative actions. The best sequence now becomes the basis for the next simulation step. At any time the user can then ask for the sequence, which at that time seems to be the best, i.e., the sequence of alternative actions that leads to a simulated state, which is closest to the desired end state. Action lists in the investigated plans are obviously not infinite, which means that they will gradually terminate. Consequently, the simulation program continues to execute the options that follow the second best system state. Given enough execution time all options will eventually be investigated. For the tool to function in this way the simulation system stores a list of all executed actions, the corresponding system state, and the distance value. Therefore, the simulation kernel provides a service to store all this information in a dynamic list and is also able to restart the simulation from a previously stored state.

6.2 Distance functions in A*-search

A problem in applying the A*-search algorithm is to find a proper distance function. In our model the states of the actors and the environment are described by a large amount of parameters, which complicates the task of the defining a credible distance function. The solution chosen is to define a function that calculates the distance based on the difference between parameter values of a given state and the parameter values of the end state. The parameters are not represented by real numbers, but rather as histograms.

A state S_{i,y_i} is a vector of length *n* with different sub-states $S_{i,y_i,j}$, where $S_{i,y_i,j}$ is a distribution over {0, 1, 2, 3}, e.g., $S_{i,y_i,j}$ = (0.2, 0.5, 0.2, 0.1) where the first 0.2 is the frequency of "0", and 0.5 the frequency of "1", etc. We have,

$$S_{i,y_i} = (S_{i,y_i,1}, S_{i,y_i,2}, \ldots, S_{i,y_i,n}), \tag{5}$$

where y_i is the current sequence of choices made for all activities A_1 to A_i. The initial state is called $S_{0,0}$, and the end state is called S_e.

The distance $\Delta(S_{i,y_i}, S_{i+1,y_{i+1}})$ between two successive states S_{i,y_i} and $S_{i+1,y_{i+1}}$ is calculated as

$$\Delta\left(S_{i,yi}, S_{i+1,y_{i+1}}\right) = \sum_{k=0}^{3} \left| S_{i,y_i,j}(k) - S_{i+1,y_{i+1},j}(k) \right|. \tag{6}$$

During simulation an assessment is made of how well each action is performed. This is done by the functions *g* and *h*. Function *g* measures the consequence of all actions performed as a distance from the initial state $S_{0,0}$ to the current simulated state S_{x,y_x} action-by-action [1]. We have,

$$g(y_x) = \sum_{i=0}^{x-1} \Delta\left(S_{i,y_i}, S_{i+1,y_{i+1}}\right). \tag{7}$$

Function *h* is a heuristic estimate of the remaining distance from S_{x,y_x} to the end state S_e. The estimated distance from the current state to the end state is given by

$$h(y_x) = \Delta\left(S_{x,y_x}, S_e\right). \tag{8}$$

With the total distance from the initial state to the end state via the current state is

$$f(y_x) = g(y_x) + 80h(y_x). \tag{9}$$

This is the distance function that is minimized by A*. The weight "80" was derived by experimentation to balance the performance of minimizing *g* and *h* and is domain dependent.

7 Decision support methodology

Decision support is given as a set of plans that are similar in structure and consequences. We cluster the patterns of plan instances that are similar in structure and consequences. Similar in structure means that they have more or less carried out similar alternative actions. Similar in consequences means that they travel on average the same distance action-by-action towards the end state.

We observe the difference in consequences between two plans. We compare the difference in the incremental changes of g and h called ΔG and ΔH, respectively, for each action A_k and both plans P_i and P_j as they progress down the sequence of additional actions A_k. We have, for each A_k,

$$\Delta G\left(P_i.A_k, P_j.A_k\right) = \left|\Delta g\left(P_i.A_k\right) - \Delta g\left(P_j.A_k\right)\right| \tag{10}$$

and

$$\Delta H\left(P_i.A_k, P_j.A_k\right) = \left|\Delta h\left(P_i.A_k\right) - \Delta h\left(P_j.A_k\right)\right| \tag{11}$$

where

$$\Delta g\left(P_i.A_k\right) = g\left(P_i.A_k\right) - g\left(P_i.A_{k-1}\right) \tag{12}$$

and

$$\Delta h\left(P_i.A_k\right) = h\left(P_i.A_k\right) - h\left(P_i.A_{k-1}\right), \tag{13}$$

and i is an index for different plan instances and k is the index for actions.

Thus, $P_i.A_k$ is a variable referring to the kth action of the ith plan. It takes an integer value that is the number of the alternative chosen for this action, e.g., $P_1.A_3 = 41$ imply that action number 3 of plan number 1 executes alternative number 41.

In addition, we need to measure the structural distance between two plans. This is done by the Hamming [12] distance *Ha* which measures the structural distance between P_i and P_j.

We have,

$$Ha\left(P_i.A_k, P_j.A_k\right) = \begin{cases} 0, & P_i.A_k = P_j.A_k \\ 1, & P_i.A_k \neq P_j.A_k \end{cases} \tag{14}$$

when both actions $P_i.A_k$ and $P_j.A_k$ exist within the simulated sequences P_i and P_j, otherwise 0 by definition.

Using this measure, we compare each action in two different plans to calculate the structural distance between the plans. For each action we observe the alternative chosen in both plans.

We put these three measures together into an interaction function that measures the overall distance between plan P_i and P_j.

We have,

$$J_{ij}^{-} = 1 - \left[1 - \frac{1}{|A_k|} \sum_k Ha(P_i.A_k, P_j.A_k) \right] \times \left[1 - \frac{1}{|A_k|} \frac{\sum_k |\Delta g(P_i.A_k) - \Delta g(P_j.A_k)|}{\max_k \{ |\Delta g(P_i.A_k) - \Delta g(P_j.A_k)| \}} \right] \times \left[1 - \frac{1}{|A_k|} \frac{\sum_k |\Delta h(P_i.A_k) - \Delta h(P_j.A_k)|}{\max_k \{ |\Delta h(P_i.A_k) - \Delta h(P_j.A_k)| \}} \right]. \quad (15)$$

Here the sums of the second and third lines are normalized by the maximum difference, and all sums in the three factors are normalized by the number of actions of the plan. Thus, $J_{ij}^{-} \in [0,1]$ and is "1" if one of the three measures is at maximum, and is "0" is all three measures are at minimum.

We partition the set of all simulated plans into clusters using the Potts spin model [13] in such a way as to minimize the overall sum of all interactions J_{ij}^{-} within each cluster.

The Potts spin problem consists of minimizing an energy function

$$E = \frac{1}{2} \sum_{i,j=1}^{N} \sum_{a=1}^{q} J_{ij}^{-} W_{ia} W_{ja} \quad (16)$$

by changing the states of the spins W_{ia}, where $W_{ia} \in \{0, 1\}$ and $W_{ia} = 1$ means that plan P_i is in cluster a. This model serves as a clustering method if J_{ij}^{-} is used as a penalty factor when plan P_i and P_j are in the same cluster.

For computational reasons we use a mean field model, where spins are deterministic with $V_{ia} = \langle W_{ia} \rangle$, $V_{ia} \in [0, 1]$, in order to find the minimum of the energy function. The Potts mean field equations are formulated [14] as

$$V_{ia} = \frac{e^{-H_{ia}[V]/T}}{\sum_{b=1}^{K} e^{-H_{ib}[V]/T}} \quad (17)$$

where

$$H_{ia}[V] = \sum_{j=1}^{N} J_{ij} V_{ja} - \gamma V_{ia} \quad (18)$$

and T is a parameter called the temperature that is used to control the influence of the interaction. This is a system parameter initialized to

$$\frac{1}{K} \cdot max(-\lambda_{min}, \lambda_{max}) \quad (19)$$

where K is the number of clusters, and λ_{min} and λ_{max} are the extreme eigenvalues of M, where

$$M_{ij} = J_{ij}^{-} - \gamma \delta_{ij}. \quad (20)$$

In order to minimize the energy function (17) and (18) are iterated until a stationary equilibrium state has been reached for each temperature. Then, the temperature is lowered step-by-step by a constant factor until $\forall i, a.\ V_{ia} = \{0, 1\}$ in the stationary equilibrium state, Fig. 9 [15][16].

INITIALIZE

K (number of clusters); N (number of plans);

Assign $J_{ij}^{-}\ \ \forall i, j$; $s = 0$; $t = 0$; $\varepsilon = 0.001$; $\tau = 0.9$; $\gamma = 0.5$;

$T_0 = T_c$ (a critical temperature) = $\frac{1}{K} \cdot \max\left(-\lambda_{\min}, \lambda_{\max}\right)$, where $\lambda_{\min}$ and $\lambda_{\max}$ are the extreme eigenvalues of M, where $M_{ij} = J_{ij}^{-} - \gamma\delta_{ij}$;

$$V_{ia}^{0} = \frac{1}{K} + \varepsilon \cdot rand[0,1]\ \ \forall i, a;$$

REPEAT

- REPEAT-2

  $\forall i$ Do:

  - $H_{ia}^{s} = \sum_{j=1}^{N} J_{ij}^{-} V_{ja}^{\begin{cases} s+1, j<i \\ s, j \geq i \end{cases}} - \gamma V_{ia}^{s}\ \ \forall a$;
  - $F_i^s = \sum_{a=1}^{K} e^{-H_{ia}^{s}/T^t}$;
  - $V_{ia}^{s+1} = \frac{e^{-H_{ia}^{s}/T^t}}{F_i^s} + rand[0,1]\ \ \forall a$;
  - $s = s + 1$;

  UNTIL-2

  $$\frac{1}{N} \sum_{i,a} \left| V_{ia}^{s} - V_{ia}^{s-1} \right| \leq 0.01;$$

- $T^{t+1} = \tau \cdot T^t$;
- $t = t + 1$;

UNTIL

$$\frac{1}{N} \sum_{i,a} \left(V_{ia}^{s}\right)^2 \geq 0.99;$$

RETURN

$$\left\{ \chi_a \middle| \forall S_i \in \chi_a.\ \forall b \neq a\ \ V_{ia}^{s} > V_{ib}^{s} \right\};$$

Fig. 9. Potts spin clustering of simulated plans partition the set of simulated plans into clusters of similar plans.

To find the optimal number of clusters K we plot the energy function (16) in a graph for different number of clusters K. We use a convex hull algorithm to calculate the lower envelope of E. At an arbitrary abscissa, the envelope function is bisected in a left and right part, each of which is fitted by least squares to a straight line. The acute angle between the two lines is maximized over all bisection abscissas and the maximizing abscissa is chosen as the number of clusters [17].

These clusters are sets of alternative plans available, should re-planning be necessary. If a plan is in the midst of execution the decision maker can observe evaluations of alternative continuations of the plan, and see which alternative activities to avoid and which are preferable as they are within a robust subset of plans.

In the next few sections 7.1−7.7 we present a number of different analysis and decision support methodologies.

7.1 Recording Decision Makers Selection of Action Alternatives

A modified version of the A*-algorithm is used in the simulation engine that not only searches for the best path, but also records all paths that have hit a leaf node in the search tree. These recorded paths can later be visualized with our tree visualization GUI. This means that a decision maker will be able to browse the complete tree and see which nodes were included in the best path in addition to other nodes that almost were included as well as those that were discarded.

In Fig. 10 we see an example of paths that were visualized with the tree visualization GUI. Here, we see that nodes 1, 2 and 41 on the top were included in the best path (green color). If the user clicks on node 61 which is on the next level in the tree, he will continue down the best path, however, if he clicks on node 42 (orange color) he will select a path of lower quality.

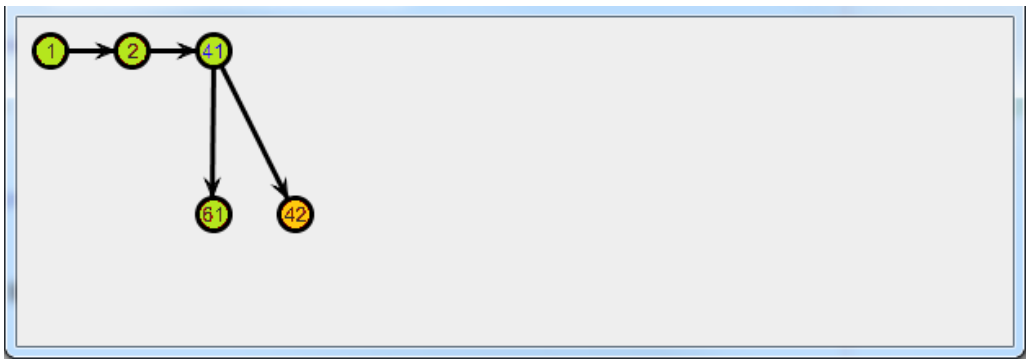


Fig. 10. Tree visualization of a plan (level 4).

If the user clicks on node 61, its children on the next level are shown, see Fig. 11. Here, we see that there are two choices, either we continue along the best path by selecting node 6, or we can browse the tree through a node (white node 108) that has not been included in any path.

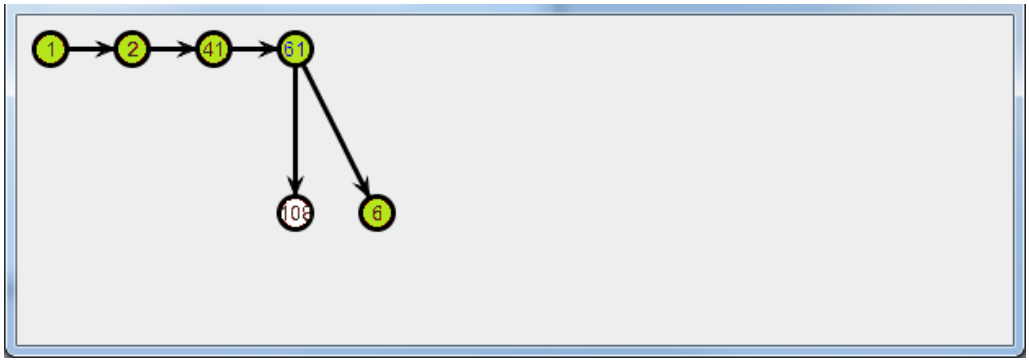


Fig. 11. Tree visualization of a plan (level 5).

Continuing further down the tree we see in Fig. 12 that node 6 has five children. Node 8 and 7 are colored red, since traveling to those nodes diverts too much from the optimum path. Nodes 10 and 5 are close to the best path, which goes through node 9.

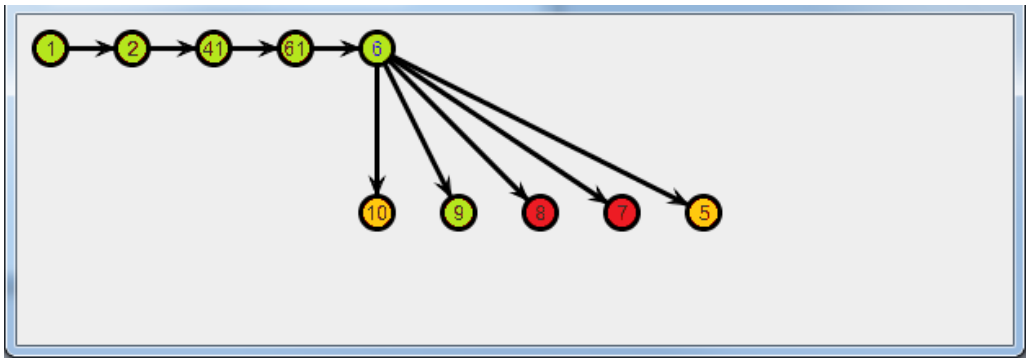


Fig. 12. Tree visualization of a plan (level 6).

Using this type of visualization that shows the best path in a context where also other alternatives are expressed, gives the decision maker the ability to recognize which actions affect the total outcome at a certain step in the plan. This mechanism is important in order to show action traceability in the system.

7.2 Visualizing Best Plan Effects Time Series

During plan execution it is valuable to analyze whether the entire operation is approaching the desired *end state*. One way to find if this is the case is to define a set of advantageous key states that have to be achieved. Achieving such a state (hence moving from a present worse state) is called obtaining an *effect*. They are often stated on a high semantic level. An effect in the Bogaland scenario could typically be "Establish order and stability in East Kasuria".

As effects-based planning will typically involve designating a number of effects, described in natural language, whose fulfillment are assumed to constitute the path towards achieving the end state, it becomes interesting to examine what role such effects play in the context of a simulation that is steered in the direction of a desired end state. Since the end state is a point in parameter space which the simulation tries to reach, then the effects should be seen as partitions of the parameter space that the simulation increasingly occupies, and the intersection or center of gravity of these partitions should be close to the end state. Visualizing the fulfillment of effects along the progression of simulation for the best found plan (i.e., action numbers on the *x*-axis), should shed light on whether the best course of action does indeed correlate with step-wise achieving the designated effects, or whether success is best achieved through other paths.

Conversely, monitoring the degree of fulfillment of effects allows one to spot whether a plan that eventually leads to a desirable end-state does so by passing through unacceptable sub-states (as given by dips in fulfillment of critical effects).

The total actor state is at each time defined by the matrix of 15 parameters with value 0, 1, 2, 3 for all 40 actors, where each parameter can change value as a result of each action. Formally, an effect is defined as a limited volume, or a union of volumes, in the 15-dimensional parameter space. This is equivalent with 15 sub-intervals of the allowed parameter values 0, 1, 2, 3 (or a union of such). The distance of the present collective actors' state to an effect is the sum of the Manhattan (L_1 metric) distances from all actors' present parameter values to the closest points (corners, sides or hyperplane) of the effect, measured from each actor.

Fig. 13 shows an example when monitoring four effects during execution of the best plan. In this example we observe no trends but notice that three out of four effects are mostly achieved. This is explained by that we are actively striving towards the end state (point) and not towards an effect (volume) in parameter space as effects are here only monitored but not actively aimed at. Hence, in this example, we are neither approaching, nor distancing ourselves from any of the effects in any major way.

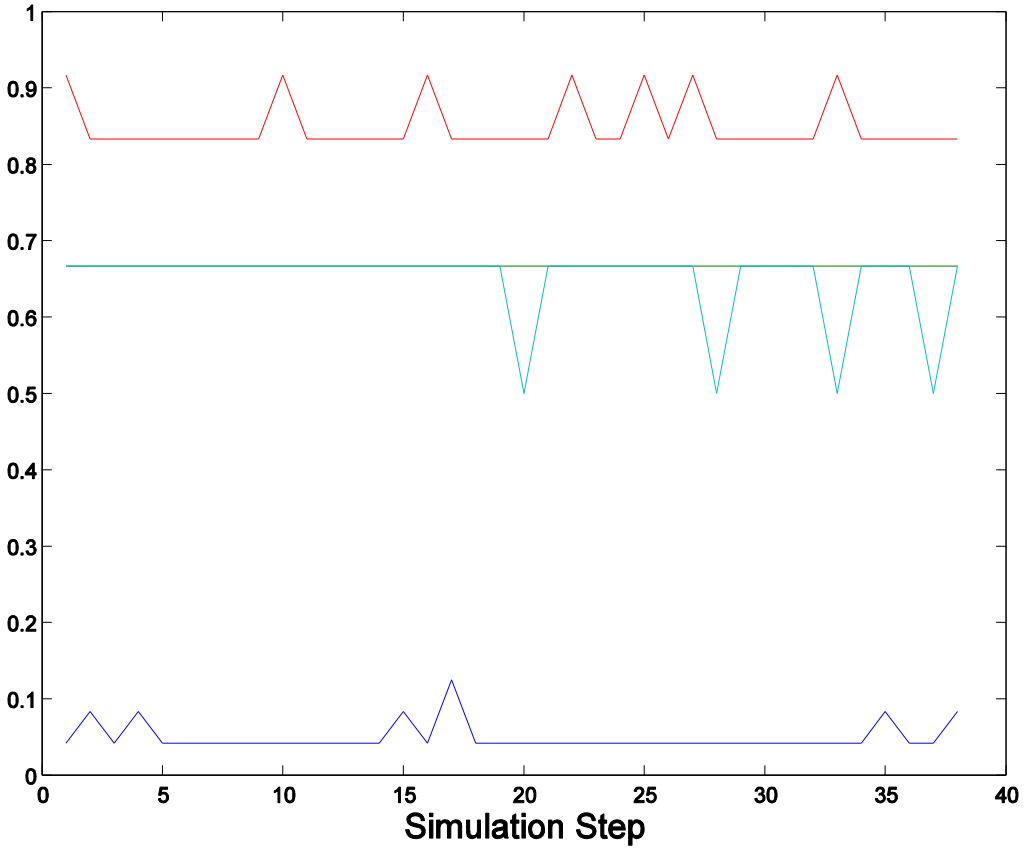


Fig. 13. Time series of supporting effects and decisive points.

Another way to assess the probability for achievements of the effects – and finally the end state – is to use the CIM. In the CIM expected impacts of actions on effects, and effects on the end-state are stated. Depending on the observed progress of actions, together with the commanders own observations of the situation on any effect-level, the probability of achieving these higher effects can be assessed [8].

7.3 Visualizing Multiple Plan End State Time Series

During A*-search the simulator tries to find a way to traverse the search tree that minimizes *f*; the sum of the distance travelled so far *g* plus the expected distance to the end state *h*. That is to minimize the expected total effort of the operation to move the actors to the end state. These measures are computed and stored for every time step, i.e., after each execution of an action. A monotonous decrease of *h* means we are continuously approaching the end state. Function *h* has an analytical definition according to section 6.2; the sum of the Manhattan distances from all actors' present parameter values to the end state (a point in parameter space). In practice we do not know exactly how to get to the end state, even if we can assess the distance. Function *g* is the length of the path travelled so far, i.e., the sum of the L_1 parameter value changes for each actor over all actions executed so far. Function *g* will always increase, but *h*, the distance to the end state, might decrease in proportion to how successful an action is or increase for an action shifting these parameters away from the end state.

We can plot the measures vs. action number for a few of the best plans to get a feeling of how well the simulation manages to approach the end state, see Fig. 14. In the figure, it seems as each plan step roughly moves us about the same distance (nearly linear development of *f* and *g*), but it is not reflected as well in the development of *h*, that is we are certainly not marching straight towards the end state (which should give a reduction of *h* for each step as large as the increase of *g*). Rather, as seen in the noisy behavior of *h*, some actions tend to take us farther from the end state, giving increases about every second step in *h*. For the example plan under investigation we start at simulation-step 0 with a distance $h = 744.0$ to the end state. The situation deteriorates for all 10 best simulations and turns favorable after the fifth action. The quick deterioration is due to a change of most of the parameters directly when BFOR enters Bogaland and initiate its first action. After the fifth action, the general trend seen here (as well as for all 10 000 simulations) produces a slowly decreasing *h*. The best result reached is $h = 792.3$, as seen in the figure. Hence, the plans currently under investigation does not take us as far towards the end state (formally at $h = 0$). From the analysis of these time series it is obvious to the decision maker whether any plans under investigations are successful or need more development work.

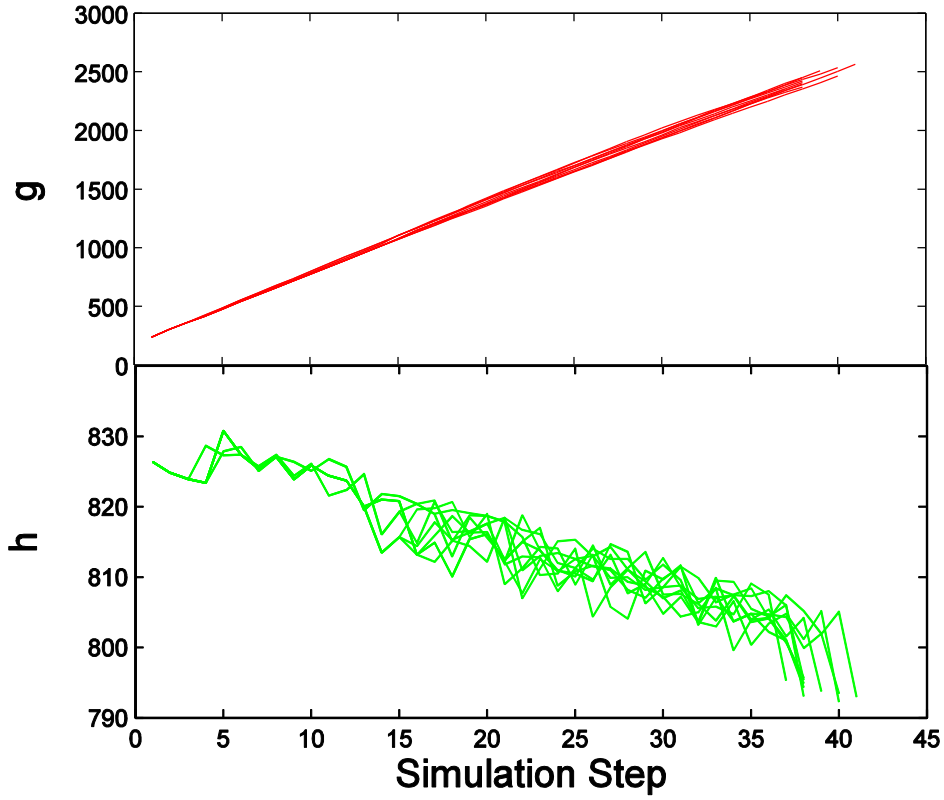


Fig. 14. Functions *f*, *g*, *h* plotted vs. simulation step for the 10 best plans.

7.4 Actors Time Development

The simulation engine gives us different paths where each one contains a solution to the problem, that is, a chain of actions that need to be executed in a specific order. Within those actions the actors' parameters are affected based on logic developed by a Subject Matter Expert (SME). For an analyst it is interesting to see how the actors progress during the entire execution of all actions.

There are different approaches how to visualize multiple variables at once. In our case we would like to visualize the following variables:

- actors' parameters,
- actions,
- actors' relationship to the blue forces,
- how important an actor is based on economy, stability and dominance in the region,
- temporal changes.

For this visualization we use a bubble chart combined with some of the animation effects demonstrated by Rosling during one of his lectures at TED [18], Fig. 15 and Fig. 16.

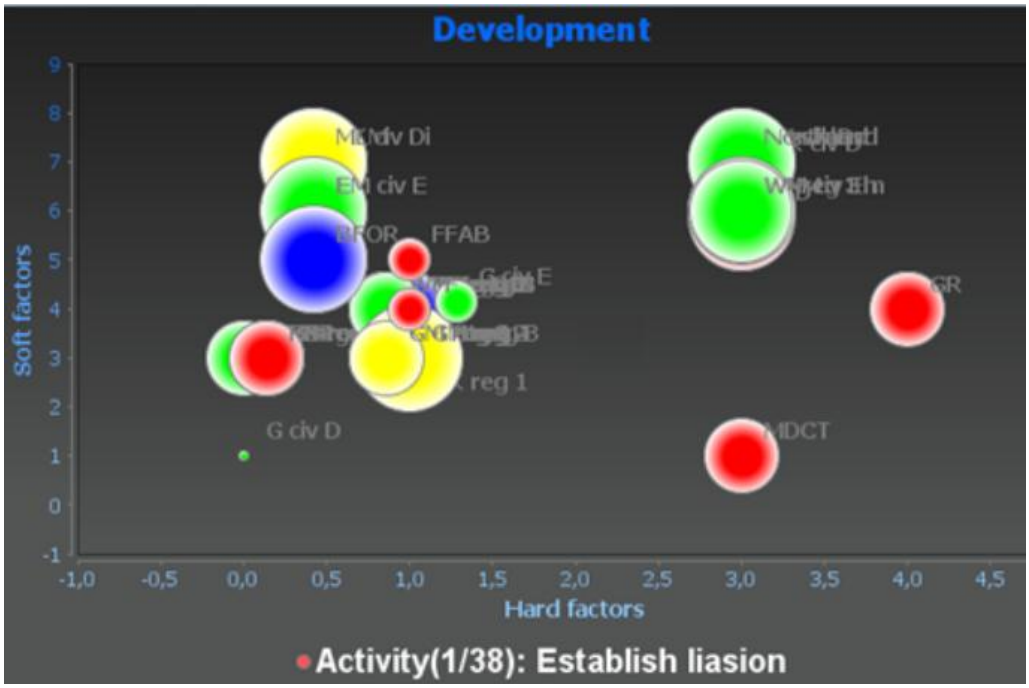


Fig. 15. Animated bubble chart visualization of the best plan (at start of operations).

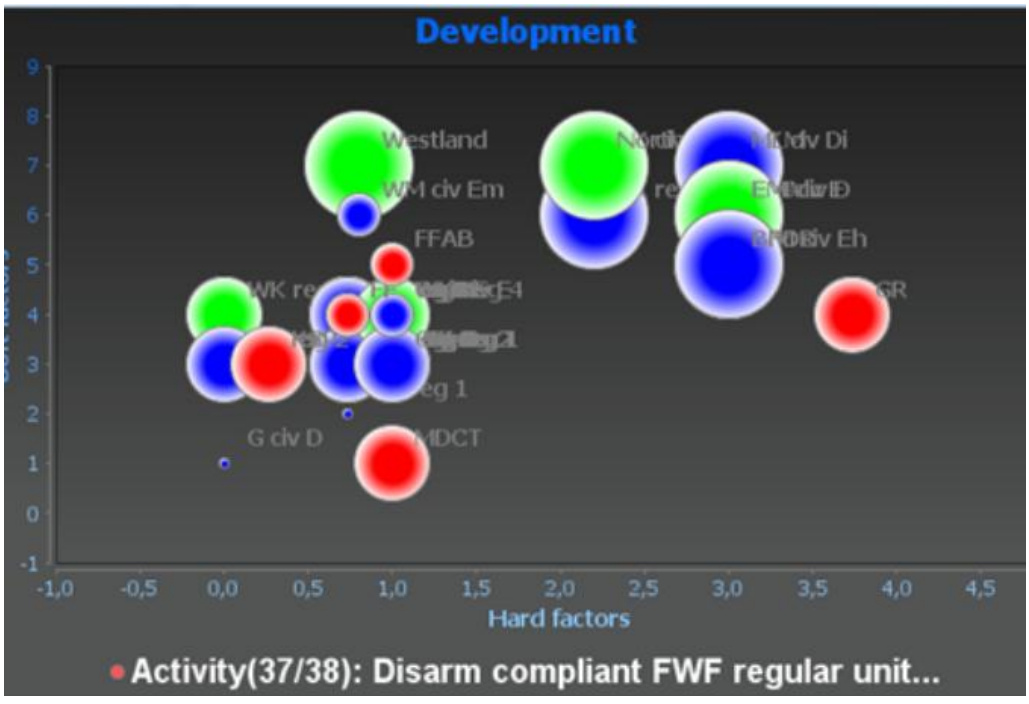


Fig. 16. Animated bubble chart visualization of the best plan (at end of operations).

We group the parameters into three different groups. One group is *soft factors* which consisted of parameters related to an actor's social status, such as his social network, feeling as a group, etc.; this is plotted on the *y*-axis. A higher value indicates a more socially connected actor. Another group is *hard factors* which consist of parameters such as weapon power, infrastructure, etc.; this is plotted on the *x*-axis. A higher value indicates a more military advanced actor. The third group is made of parameters such as economy, geographical dominance and stability. This group is represented as the size of the bubble. A bigger bubble represented a more important actor.

The bubbles also have colors which represents the relationship between the actor and the blue force BFOR. Green indicates a neutral actor; blue indicates an ally, red an enemy and yellow an unknown or suspicious actor.

For each action a complete new set of visualized data is rendered. Putting all of those renderings together the summarized effect is animated bubbles representing system changes occurring with respect to time (i.e., execution of action). Furthermore, the visualization software also has different tool features such as fast-forwarding and filtering based on visible actors in the GUI.

7.5 Explaining the impact of actions

An explanation function for explaining the impact of actions is based on sensitivity analysis of the impact of different actions upon the success of the plan where we systematically vary the alternatives of each action of the plan, one action at a time, keeping all the other actions unchanged in a series of simulations. This sensitivity analysis shows the relative level of importance of making the correct selection of alternative for each action. Using the explanation function, a decision maker can find the most important actions of a plan and focus his attention on actions where successful decision making is crucial to the success of the entire plan.

As we work with plans consisting of several actions A_k we like to find the impact of each action on the evaluation $\{f_{ikl}(P_i.A_k = l)\}_l$ of plan P_i, where i is the index of the plan, k is the index of the action, and l is the index of the alternative. This impact can be denoted $\partial f_{ikl}/\partial A_k$. Given that we have a discrete set of evaluations $\{f_{ikl}(P_i.A_k = l)\}_l$ we approximate the differentiation as a normalized difference between $\max_l f_{ikl}(P_i.A_k = l)$ and the average of all $\{f_{ikl}(P_i.A_k = l)\}_l$. We have,

$$\left.\frac{\partial f_{ikl}(P_i.A_k)}{\partial A_k}\right|_l = \frac{\frac{1}{n_{ik}}\sum_{j=1}^{n_{ik}} f_{ikj} - f_{ikl}}{\frac{1}{n_{ik}}\sum_{j=1}^{n_{ik}} f_{ikj}} \tag{21}$$

where $n_{ik} = \left|\{f_{ikl}(P_i.A_k = l)\}_l\right|$ is the number of alternatives for $P_i.A_k$ [19].

As the variance in this measure can be large between different plans P_i we may choose to study box plots for a small number of good plans for each action A_k. For example, we will study box plots for the five best plans over all alternatives for averages of all actions A_k,

$$\left\{\left.\frac{\partial f_{ikl}(P_i.A_k)}{\partial A_k}\right|_l\right\}_{i=1}^{5}. \tag{22}$$

In order to find the impact of the actions we need to perform additional simulations. The A*-search algorithm is intended to deliver the best plans it finds concerning the success in reaching the end state, as reflected in the distance f from start to end state; the lower, the better. Each of these plans consists of a sequence of actions where the actions have several alternative ways of execution, and a plan must choose one alternative from each of these actions. Some actions in the simulation turn out to be more important than others for plan success. In order to find out how much a plan relies on a selection of a certain action alternative for its success, one might compare a good plan P_i found by the A*-algorithm with plans that are structurally similar to it in some respect. This can be done by comparing P_i with neighboring plans that only differs from P_i in the selection of alternatives for one single multi-alternative action, see Fig. 17. Thus, we have

$$\sum_{k=1}^{n}\begin{cases}0, P_i.A_k = P_j.A_k \\ 1, P_i.A_k \neq P_j.A_k\end{cases} = 1. \tag{23}$$

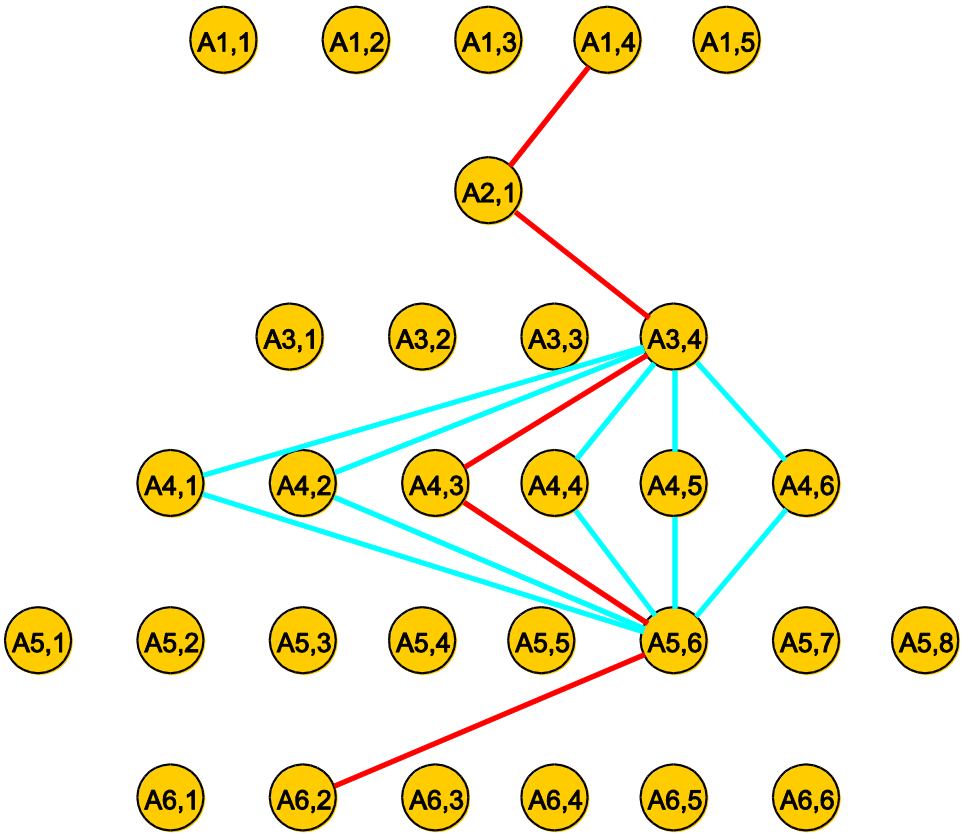


Fig. 17. Conceptually, a plan P_i is a choice of alternatives for a sequence of actions, one for each consecutive action to be executed, like the red colored path. Each cyan colored path in this six-action planning problem corresponds to one neighboring plan with $P_i.A_4$ = {1, 2, 4, 5, 6} for action 4.

We simulate all neighbors to each good plan P_i already found with a variation compared to P_i of exactly one action alternative a time. For each action A_k, we simulate P_i where the selected alternative for action A_k is replaced by another alternative to A_k in the additional simulations. This is the set P_{ik} consisting of $|P_i.A_k| - 1$ neighboring plans where

$$\forall P_i, P_i.A_k : P_{ik}(P_i, P_i.A_k) = \left\{ P_j \middle| \begin{matrix} P_i.A_m = P_j.A_m, \forall m \neq k \\ P_j.A_k \neq P_i.A_k \end{matrix} \right\}_{j=1}^{|P_i.A_k|-1}. \tag{24}$$

After having worked through all actions with alternatives, changing only one action at a time, we get as many neighboring plans to P_i as the total number of additional alternative actions, excluding the alternatives that are part of P_i itself. For a set of n actions there are $|P_i.A_k| - 1$ alternatives to an action A_k in addition to the one in P_i. We have a total of

$$\sum_{k=1}^{n}\left(|P_i.A_k| - 1\right) \tag{25}$$

neighboring plans to be compared with P_i. In our analysis, we will now use g and h instead of f as a refined quality measure of a plan and investigate how it is affected by systematic variations of each action of the plan.

We look at $\partial g/\partial A$ and $\partial h/\partial A$ when varying only a single action at a time, Fig. 18.

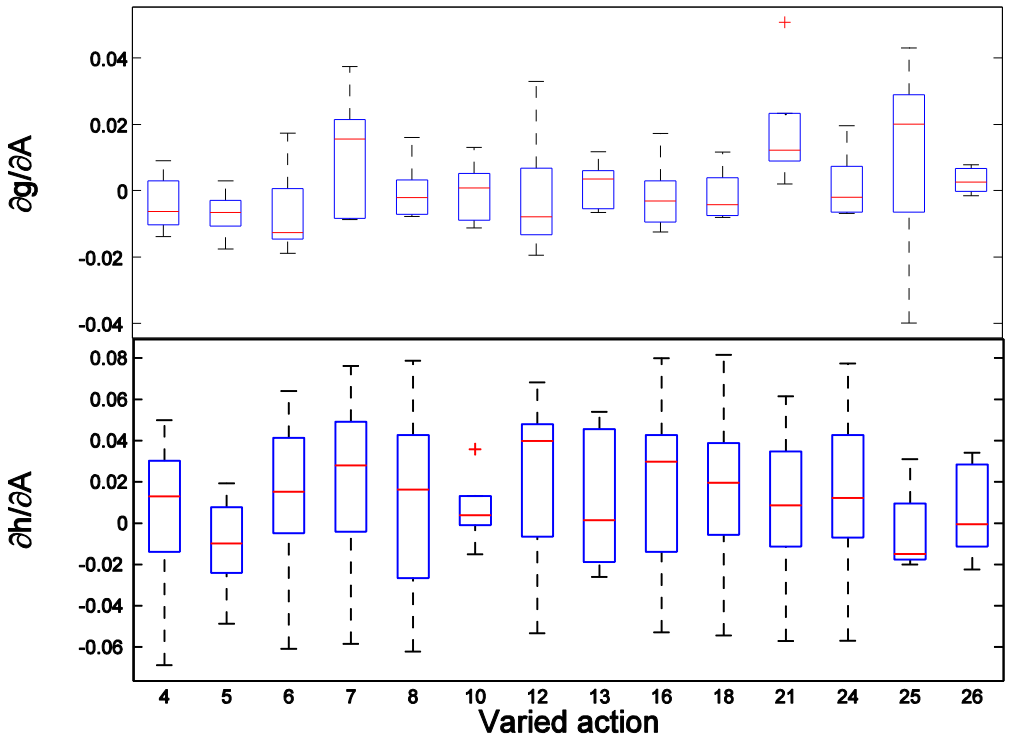


Fig. 18. The spread in sensitivity of the five best plans where each action sensitivity is computed for *g* and *h* similarly to (22) for *f*.

The sensitivity of *g* does not seem to be pronounced except for action 7 and 25 where a selection of another alternative than the one present in the main plan seems to give a slightly worse (higher) value of *g*. For *h*, a larger number of the actions show a pronounced sensitivity, e.g., actions 7, 12 and 16.

With this tool a decision maker can focus his attention on making the best selection of alternatives where it is most important.

7.6 Regression Tree Analysis

Regression Analysis Trees [20] can be used to hierarchically find the importance of a set of input variables on a dependent continuous output variable. After simulation and traversal of the A*-search tree, the 10 000 best plans have been obtained. As described above, each plan consists of a set of input actions where some have several discrete alternatives where each plan produced from A*-search is a certain combination of action alternatives, and the continuous *g* or *h* value may be chosen as the dependent output for each plan. The 10 000 plans make a good statistical basis for building a regression tree on these data to find the most important actions, see Fig. 19. It seems as the chosen alternative of action A_{25} has largest influence on *h*, followed by A_7 in both next branches, etc. Note that the split can depend on a certain action more than once at different levels.

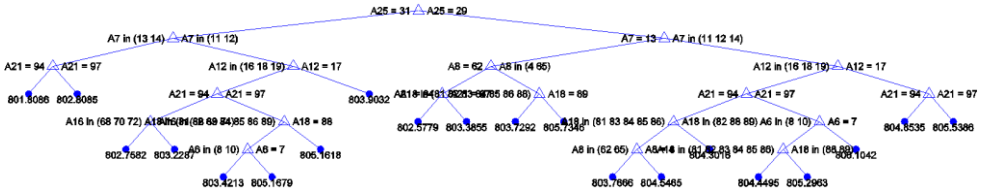


Fig. 19. A regression tree of h based on 10 000 simulations. Only the 18 most important "bifurcations" or branches on 5 levels are shown; the full tree with the default MATLAB statistical toolbox setting gets nearly 1500 branching points on 22 levels.

When a regression tree has been built, it can be used as a rough prediction tool for the dependent variable, given a new plan. A condition has to be decided for when the further splitting into branches should stop; eventually the tree would split into 10 000 leafs, one per simulated plan, but its predictive power will decrease the deeper down it is traversed, and be more based on random noise from the Monte Carlo process than the major statistical tendencies that are of interest. Given a set of alternatives a planner has chosen, one can follow the path this plan would take in the tree and find the *g* and *h* values of the plan proposed by the full tree based on the 10 000 simulations. This has been done for the 100 best plans resulting from a second set of 10000 simulations with a new seed: the paths of these plans are followed in the tree to find the values of *g* and *h* the tree then proposes. A comparison with the real values of *g* and *h* for those 100 plans can be seen in Fig. 20. The trackability is good for *g*, but worse for *h*. However, the variance is higher for the predicted *g* than for *h*. The predicted *h* is higher than the real value. For the 4000 plans with worst (highest) *f*, i.e., with a Plannumber > 4000 (not visible in the figure), the situation is actually reversed. Perhaps this is not difficult to understand; it is easier for the regression analysis to find the systematics in how far we have walked *g* than how far it is to the goal *h* since the previous lies inherently in the effect of the conducted actions, whereas the latter is not as easy to assess. Let be that it is possible to compute in parameter space as the distance from the final actor parameter state to the end state, but this is not as easy.

Two figures of merit to estimate this are the *resubstitution* error and the *cross-validation* error of a tree. With the resubstitution error we mean the root mean square of the *g* and *h* values predicted by the tree, compared to the true ones when we use the plans from which we built the tree. The cross-validation used “splits the training data into 10 parts at random. It trains 10 new trees, each one on nine parts of the data. It then examines the predictive accuracy of each new tree on the data not included in training that tree. This method gives a good estimate of the predictive accuracy of the resulting tree, since it tests the new trees on new data” [21]. In the case of *g* and *h* for the 100 best plans shown in Fig. 20, these errors are around 13.5 and 20.4 for *g*, and 1.77 and 2.6 for *h*, respectively.

Of course, in a real planning situation, it is not this simple since the planning process most often does not start at the top of the regression tree, the importance of the actions are not ordered concerning their typical order of execution, but the planning procedure is often done in a certain order.

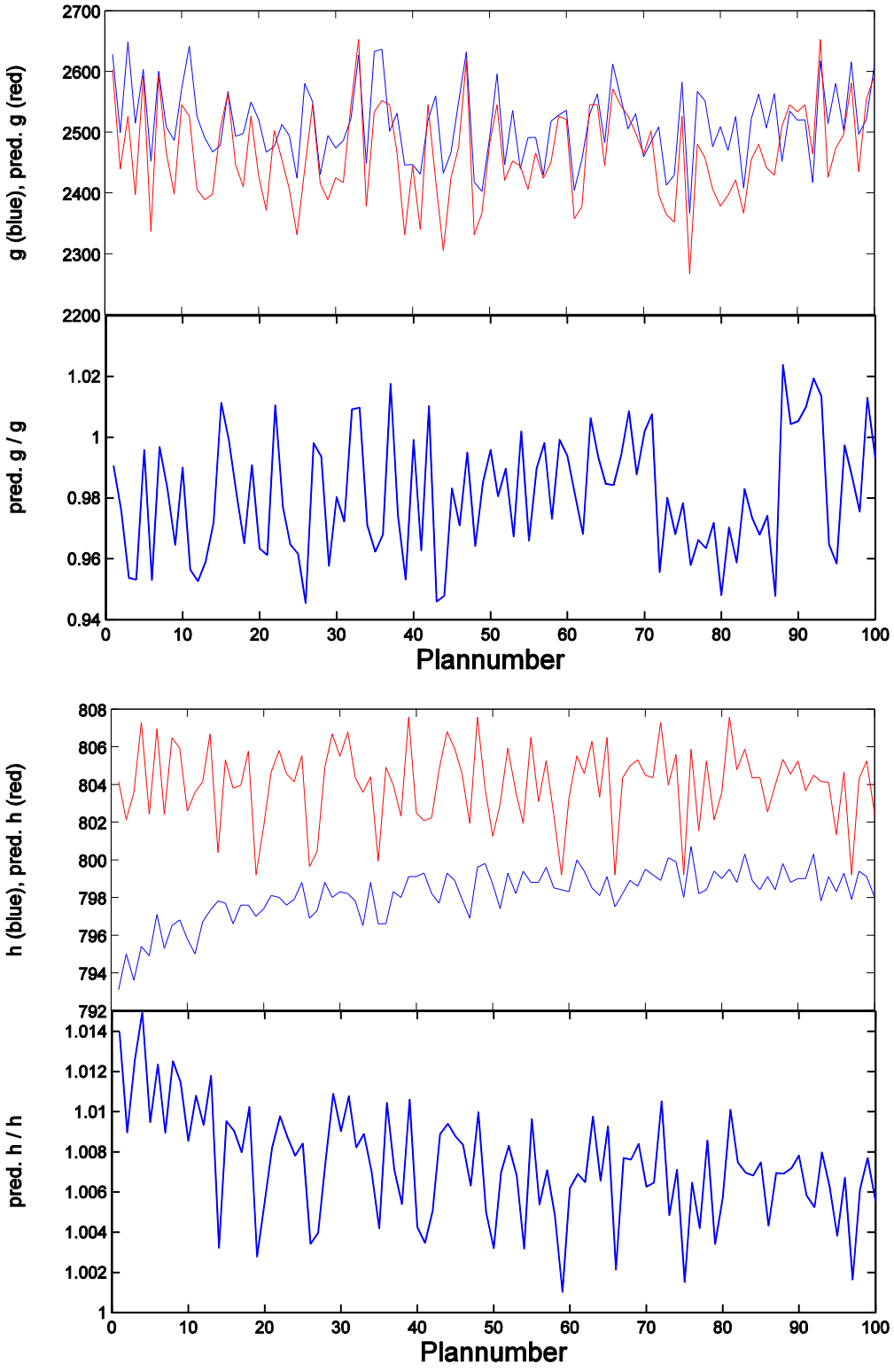


Fig. 20. Plots of real (blue) values of *g* and *h* for the best 100 plans from a simulation with a new seed as well as predicted (red) values of *g* and *h* from the regression tree in Fig. 19 trained with the 10 000 plans from the first simulations. The ratio of predicted and real values is shown in the respective lower plot.

7.7 Estimating the Boundary of Potential Failure of an Operational Plan

We summarize the information contained in a cluster of plans by using a hyperplane created by a Support Vector Machine (SVM). We are mainly interested in the distances from a chosen plan to its boundary with classes other than its own. Several stages are needed to achieve the result. First, we need to find the best way to represent the training data for use in the SVM, this includes normalization. Secondly, we must analyze the problem of finding optimal SVM-parameters and a kernel. Finally, we analyze the distances. An SVM analysis finds the hyperplane that is oriented so that the margin between the support vectors of different classes is maximized.

The concept of treating the objects to be classified as points in a high-dimensional space and finding a hyperplane that separates them is not unique to the SVM. The SVM, however, is different from other hyperplane-based classifiers in how the hyperplane is chosen. If we use linear kernel and define the distance from the separating hyperplane to the nearest data point as the margin of the hyperplane, then the SVM selects the maximum margin separating hyperplane. Selecting this hyperplane maximizes the SVM's capability to calculate the correct classification of up to that time unseen plan instances. When representing the classification boundary by the SVM optimal hyperplane, each dimension has a bound for the corresponding action in the plan. Using the SVM decision function, each action can be evaluated by its presence in the tested plans presented to the decision function. In this way, we can correct our bad plans to become good plans by simply changing the bad actions.

The first step is to adapt the plans to the SVM machinery. SVM requires that each data instance is represented as a vector of real numbers. Let a plan contain R actions which can take any value representing a valid alternative for this action. We generate N number of R-dimensional vectors for training. The plans are clustered into different classes to be used as training targets y_i. Training plans are represented by vectors $x_i = \{x_{i1}, \ldots, x_{il}\}$. The plan vectors x_i are all normalized. Scaling them before applying the SVM is very important. This is done to avoid that attributes in greater numeric ranges dominate those in smaller numeric ranges.

The basic idea of SVM is to find a linear decision boundary to separate instances of two classes within a space. In the case of linear functions f, a separating hyperplane, written in terms of a weight vector $\boldsymbol{w}$ and a threshold b takes the form $f(x) = (\boldsymbol{x}, \boldsymbol{w}) + b$ with $\boldsymbol{w} \in \boldsymbol{X}$, $b \in R$ where (,) denotes the dot product. We want to minimize the norm $\|\boldsymbol{w}\|^2 = (\boldsymbol{w},\boldsymbol{w})$ as shown in Fig. 21. This can be formulated as a convex optimization problem.

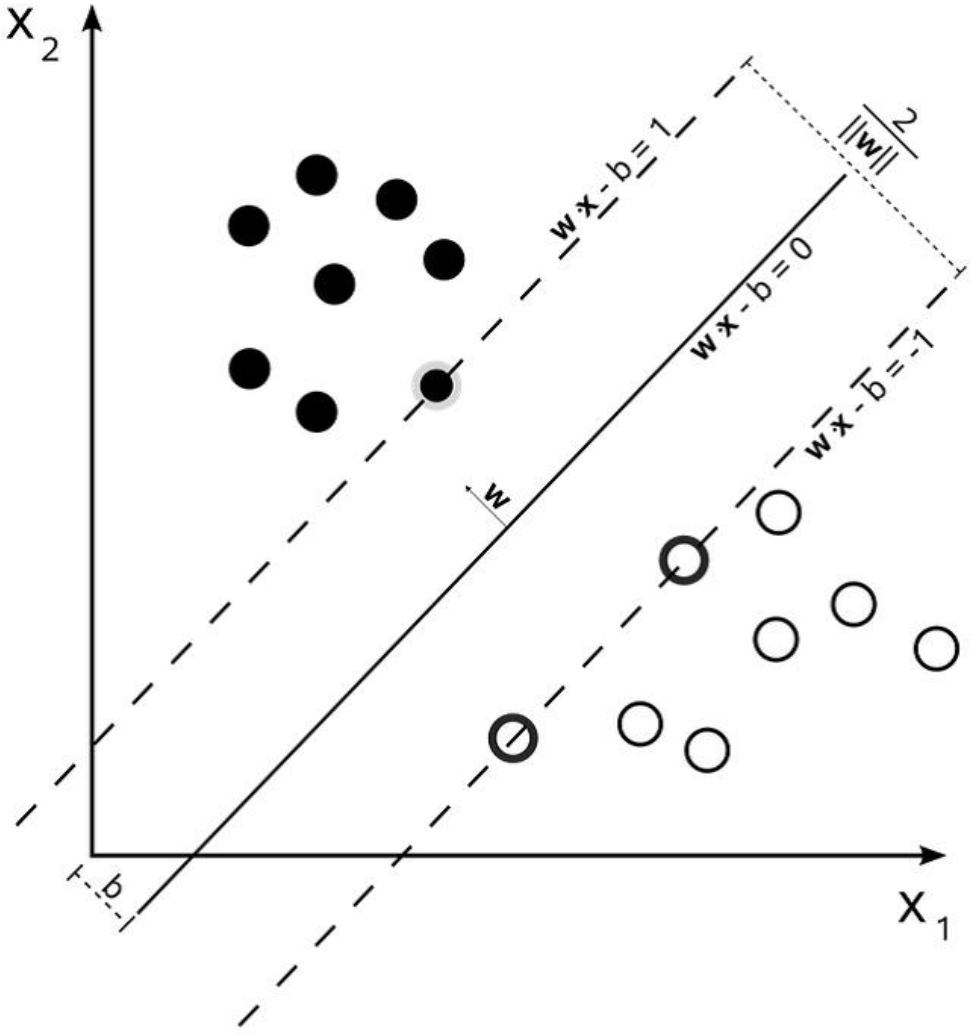


Fig. 21. Optimal linear divider of two separate classes.

Minimize

$$\frac{1}{2}\|w\|^2 \tag{26}$$

subject to

$$y_i - (x_i, \boldsymbol{w}) - b \geq 1,\ i = 1, \ldots, l. \tag{27}$$

The support vectors lie on the supporting hyperplanes of the two classes. The support vector optimal hyperplane is the hyperplane which lies in the middle of the two parallel supporting hyperplanes (of the two classes) with maximum distance $d_{max} = \frac{2}{\|\boldsymbol{w}\|}$. We have the decision function,

$$\text{sign}(\boldsymbol{wx} + b). \tag{28}$$

which defines the division of different classes and also is used to classify plans under test.

The complexity of a function's representation by support vectors is independent of the dimensionality of the input space $\boldsymbol{X}$, and depends only on the number of support vectors.

The accuracy of an SVM model is largely dependent on the selection of model parameters. Some flexibility in separating the categories is needed. SVM implementations have a cost parameter *C*, which controls the tradeoff between generalization ability and fidelity to the training set. This parameter gives the model a soft margin that permits some misclassifications [22]. Increasing *C* increases the cost of misclassification of plans and

forces a more accurate model to be created. A search is used to find the optimal value of *C*.

Using a hyperplane we may separate the feature vectors into two classes when there are only two target categories [23], but how do we handle the case where we have more than two classes? The two most used methods are: (*i*) "*one against many*" where each category is split out and all the other categories are merged, and (*ii*) "*one against one*" where $k(k-1)/2$ models are constructed where k is the number of categories. In this work we use the second approach and we evaluate classes against each other.

7.7.1 Implementation of SVM

We study an experiment of 1000 evaluated plans that are clustered by Potts spin clustering into eleven different clusters based on their characteristics and outcomes. Each action of the plan holds a unique integer number representing the alternative performed for that action. A training matrix of the 1000 different plans of length 46 is normalized with respect to each action. The eleven clusters are represented as classes which in turn are represented by any integer between 1 and 11.

We use the LIBSVM library [24] in this work. Important in LIBSVM is the choice of its parameters. Parameter optimization is done by a full search out of a pre-defined parameter set. Cross validation is used for selection of best parameters for this training set, meaning that each combination of parameter choices is checked using cross validation, and the parameters with best cross-validation accuracy are chosen. Using the selected parameters the final model is trained on the whole training set. We use the optimal hyperplane defined by the SVM for determining the distance from any plan to the boundary of the classes for the other plans.

Since LIBSVM only delivers output for calculating the distance to the support vectors, the plans nearest the hyperplane of each class, we use an extra class for the plan under execution. This is (by definition) the only support vector of this class, and the distance from any specific plan of interest to the hyperplane can be calculated. Most interesting is how the distance for a specific plan under execution changes depending on how many of the actions have been performed. To be able to calculate this, the SVM needs to be re-trained for each new number of performed actions.

The primal variable $\boldsymbol{w}$ is not a direct output of LIBSVM. Instead we use the provided support vectors SVs and the coefficients for the support vectors sv_coef;

$$\boldsymbol{w} = \text{SVs} * \text{sv\_coef}. \tag{29}$$

The model is trained for twelve classes, eleven classes from pre-calculated Potts spin clustering and one class containing the plan under execution. Training is done 45 times for each investigation, each training with successive longer plans, from plan length of two actions to training on the full matrix with 43 actions (and *f*, *g*, *h*), see Table 4.

Table 4. Pseudo code for the investigation.

1. Select the plan to be investigated and put it in a separate class. Update input label vector.

2. For length of plan = 2 to 46:

    2.1 Select optimal parameters for training.

    2.2 Train the model.

    2.3 Calculate distances using (29) and (30).

3. Plot distances.

Note that this process normally is very sensitive to noise and outliers in the training data. SVM generally have problems with unbalanced problem where one of the classes has much more training examples than the other. For a balanced training set, the outliers from class A to end up in the middle of training examples from class B, and the algorithm can then identify them as outliers. Here we have an extreme case with only one training examples in one class, and thus the algorithm has not enough information to identify outliers in the other class. Our data are a selection of the best plans out of a much larger set of plans and carefully clustered before training. The probability of noisy data and outliers are low and should not be a problem.

For all points from the hyperplane $HP[(x_i, \boldsymbol{w}) + b = 0]$, the distance between the plan of interest and the hyperplane HP is

$$d = \frac{1}{\|\boldsymbol{w}\|}. \tag{30}$$

This is the distance measure we use for calculating the distances from the tested plan to the border of another class.

7.7.2 Using Hyperplanes as Decision Support

Single plans are tested against all the other plans and the result is plotted in Fig. 22–Fig. 24. The length of the plans is on the *x*-axis and the distances on the *y*-axis. The distances from the tested plan to the border of another class varies with the length of the plan. First, the best plan is chosen from all the other plans; the best plan is the one with lowest value of *h*. The distance from the best plan to nearest hyperplane of all other classes using successive longer plans is shown in Fig. 22. This figure shows ten curves, one for each class combined to the class 12 representing the single tested plan.

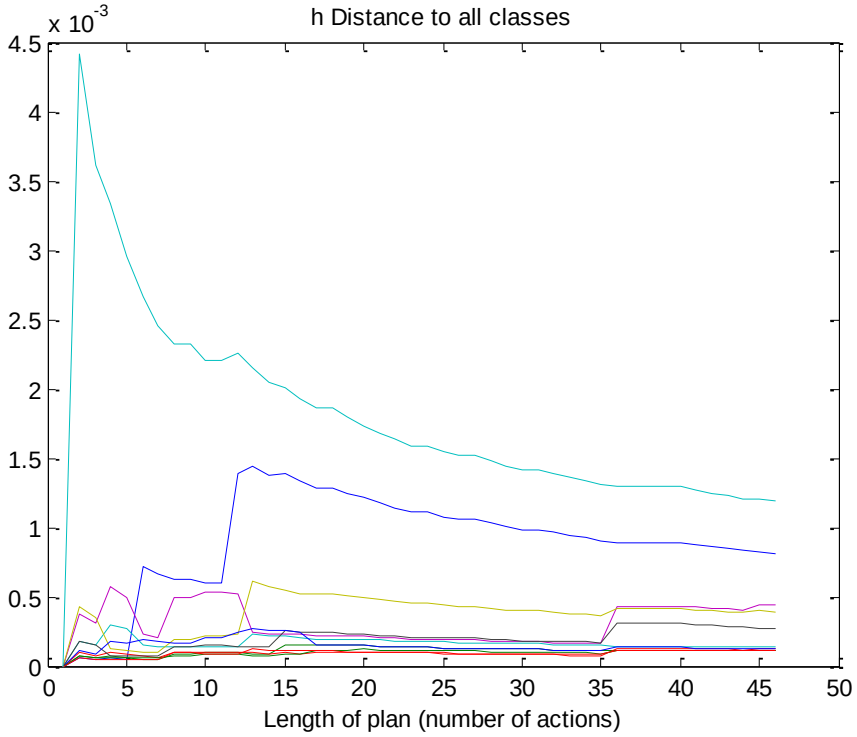


Fig. 22. Distance of the best plan during execution towards the eleven hyperplanes.

In Fig. 23 we show another view of the same result, by taking the minimum distance of all eleven classes in Fig. 22 at each length of plan.

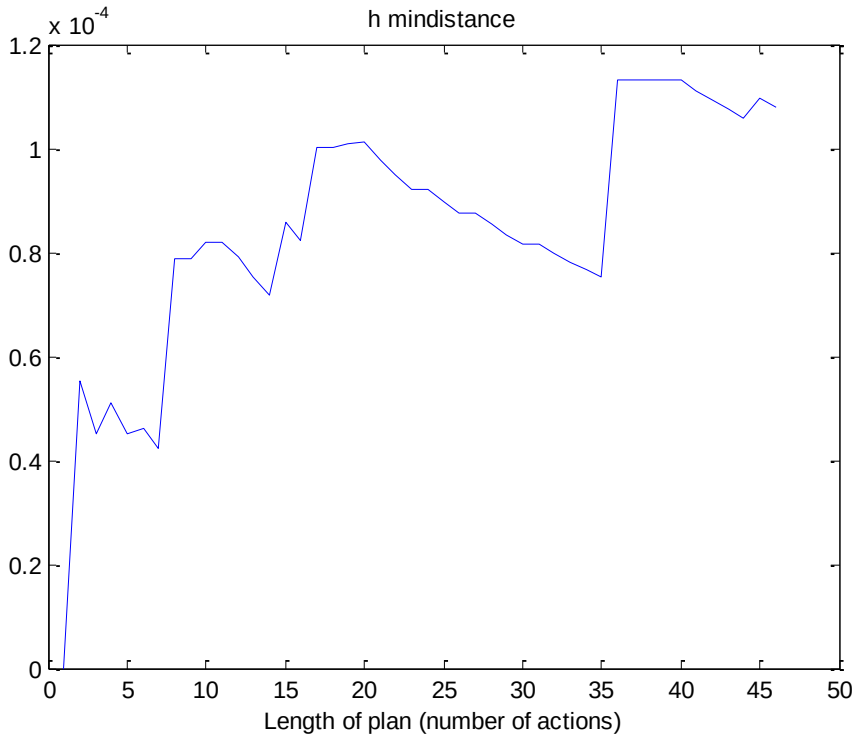


Fig. 23. Minimum distance of the best plan during execution to the closest hyperplane.

The eleven classes are designed unsupervised with respect to plans, structure and *f*-value in the preceding clustering stage. Each class is determined by its content. Since it is the 1000 best plans that are clustered, they are all relatively good, but a little different in character. It could be said that each class is determined by the quality of its best plan (min *h* value).

Since most of the plans are "good" we take a look at the ten best plans regarding *h*. In Fig. 24, minimum distances are created in the same way as in Fig. 23 are plotted for the ten best plans regarding *h*. The best plan is plotted in red for comparison. We can see that the best plan does not always have the largest minimum distance to neighboring classes.

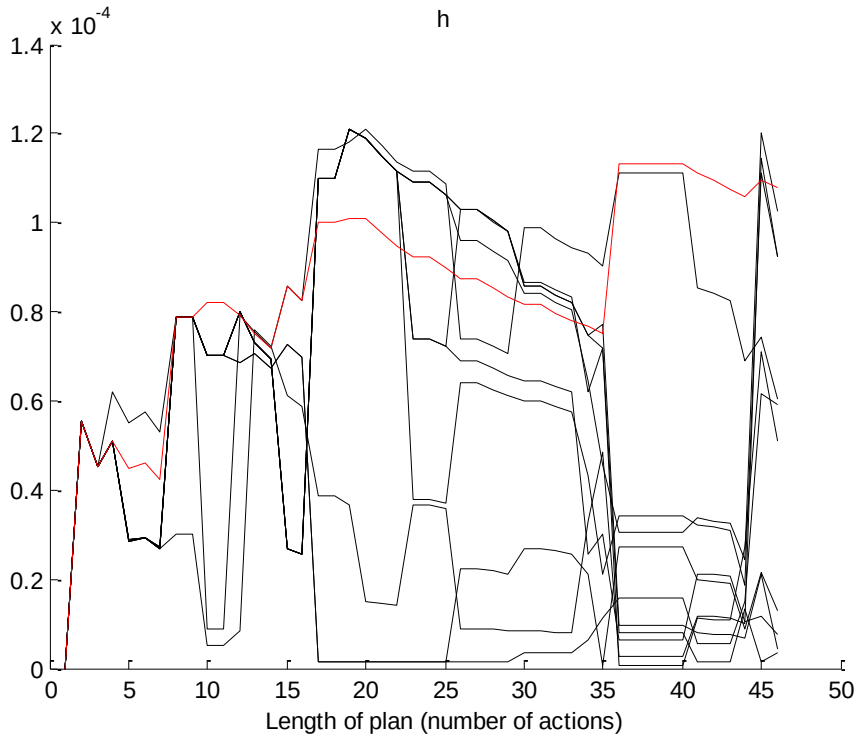


Fig. 24. Min 10 best plans, best plan in red.

The graphs show zero distance for some lengths of plans. This is natural since the class of origin for the investigated plan has zero distance to this plan as it is included in this class. Also there are mostly very small differences between the classes and, thus, their boundaries can lay tangent to each other.

By using views as in Fig. 23 we provide decision support during execution of a military operational plan. During the execution we observe in this figure the distance towards the closest (of eleven different) boarders for the plan under execution as we progress down the sequence of actions. The result shows that longer plans have larger margin to other classes.

In Fig. 22 the analyst observe a more refined view and may observe which other cluster of plans we might be approaching. The difference in outcomes by the current plan and the plans in the other cluster can then be observed by comparing with the best plan of that other cluster.

8 Simulation result analysis

8.1 Overall Description of the Bogaland Full-scale Simulation Experiment

For the purpose of our experiment we consider an extensive part of the Bogaland scenario that covers all actions performed by our own forces (BFOR) from day -70 to day +360. This scenario contains three operational phases; deployment, shaping and security support. During the deployment phase, actions such as securing the ports of disembarkation or establishing a No Fly Zone (NFZ) are being deployed. Shaping phase includes actions which require engagement with opposing forces, such as neutralizing irregular organizations' powerbases, enforcing embargo or restricting flow of irregular recruits and illegal arms.

Finally, actions in the security support phase are launched to ensure support and a correct handover of power to the local government. These actions include providing security support to the election process, supporting None Government Organizations (NGOs), and identifying and isolating maligned actors from Bogaland population.

These three operational phases are carried out through 43 different BFOR actions. Each action has between 1 to 8 alternative ways to proceed. A few of them may not be performed. Furthermore, some of these actions are divided into sub-actions, i.e., sub-actions that can only be launched as a consequence of which alternative is selected for an earlier action. In total our scenario contained 2.164×10^{23} alternative plans. Table 5 below lists the first thirty actions modeled in our scenario.

Table 5. Some actions in the Bogaland scenario.

Establish liasion	**Protect Oil/gas/key Infrastructure**
Establish liasion	**Deploy in Visby area in (some) force and Negotiate**
Establish NFZ/MEZ	**D-day Enforce cease fire by deploying in Visby area**
D-day Stabilise situation by deploying in between	**Deploy in Visby**
Deploy in secure parts	**Kill/Capture non compliant**
E. Kasuria and E.Mida Local Police Enforcing	**Negotiate and buy out Kasuria and/or Mida**
Support E.Kasuria and E.Mida Local police	**Handover to Bogaland authorities**
Neutralise MDCT By Negotiating	**Disarm compliant FWF regular units in On Gotland**
Liase with Kasuria and Neutralise MDCT	**Liase, Neutralise other Irregulas**
Neutralise MDCT	**Neutralise all Irregulas**
Liase and Kill/Capture MDCT	**Liase, Neutralise and Disarm all Irregulars**
Kill/Capture MDCT	**Neutralise and Disarm all Irregulars**
Survey Oil/gas/key infrastructure With BFOR Mil. Units (no protection)	**Stabilise situation by deploying in between West Kasuria and West Mida**
Protect Oil/gas/key Infrastructure With BFOR Mil. units	**Establish liasion**
Survey Oil/gas/key infrastructure	**Disarm FWF regular units in W Kasuria and W Mida Kill/Capture non compliant**

Altogether 40 actors were modeled in this scenario. These actors are listed in Table 6. The colors to the left of each actor indicate the initial roles of the actors in the scenario. Blue color represents BFOR and its allies, whereas red color shows the enemies. Green stands for neutral actors and yellow actors are those whose position or relation to us is not clear or yet to be determined.

Each of the 40 actors in the scenario is defined by 15 state variables, which together present the total ability of that actor and its internal state, as shown in Table 5. This sum-up to 600 (= 40 x 15) variables and 1.722×10^{361} (= 4^{600}) possible scenario states. For the purpose of our experiment we initialized these variables using data from SMEs. This also included actors' relations to each other. We also defined the desired goal state variables for each actor in cooperation with SMEs.

During the course of the scenario the actors are directly or indirectly affected by the actions carried out by BFOR. For each such action all involved actors are first pointed out and their roles in the action are defined, e.g., the actor that is enforcing the action is blue, the receiver of the action is red, etc.

For these involved actors the values of state variables are altered as a result of the action. This is what is meant by direct effect. All the other actors are affected based on their relationship with the directly involved actors, e.g., if my friend is being attacked by actor A_1 then my relationship with A_1 is being negatively affected.

Table 6. All actors in the Bogaland scenario.

	1 East Kasuria reg 1		21 East Mida reg 2
	2 East Kasuria reg 2		22 East Mida reg 3
	3 East Kasuria reg 3		23 East Mida reg 4
	4 East Kasuria reg 4		24 West Mida reg 1
	5 West Kasuria reg 1		25 West Mida reg 2
	6 West Kasuria reg 2		26 West Mida reg 3
	7 West Kasuria reg 3		27 West Mida reg 4
	8 West Kasuria reg 4		28 GM reg 1
	9 GK reg 1		29 GM reg 2
	10 GK reg 2		30 Mida Liberation Militia
	11 Death Star Division		31 Homeland Military Militia
	12 Movement of Delta Christian Tradition		32 GR
	13 East Kasuria civ Dn		33 East Mida civ E
	14 East Kasuria civ Di		34 West Mida civ Em
	15 West Kasuria civ E		35 West Mida civ Eh
	16 West Kasuria civ D		36 G civ E
	17 G civ D		37 Kasurian Special Police
	18 Westland		38 Freedom Front
	19 Northland		39 FFAB
	20 East Mida reg 1		40 BFOR

When executing the actions the simulation traverse through the action tree, which is our complete set of plans, using the A*-algorithm. Each action that is executed results in a set of reactions (i.e., actions conducted by other actors) as the actors' state variables and relations to other actors are being updated. These reactions might in turn result in new iteration of reactions. In our experiment we limit the number of reaction iterations to two.

As explained in section 6, we use Monte Carlo simulations to obtain a frequency function of the entire outcome space of our actions. The number of Monte Carlo simulations in our experiment was limited to 20.

The experiment was run on one Intel Xeon E5-2687W with 3.1 GHz and 64 GB RAM. The experiment terminated after the first 10 000 plans, i.e., the A*-algorithm terminated when it reached 10 000 leaves in our action tree. On average the time it took to generate, simulate and evaluate one plan alternative was 24 seconds. Each experiment run took 2.8 days to execute without any parallel computing.

8.2 The Simulation Output Data

For our scenario the simulator produced 10 000 rows of data which is a small fraction of the total theoretical amount of output. When analyzing the data we find different patterns which are visualized. Fig. 25 shows the parameter distribution for the 1000 first evaluated plans. Note that the parameters are distributed in different ways due to the fact that the scenario affected different parts of an actor during the entire execution.

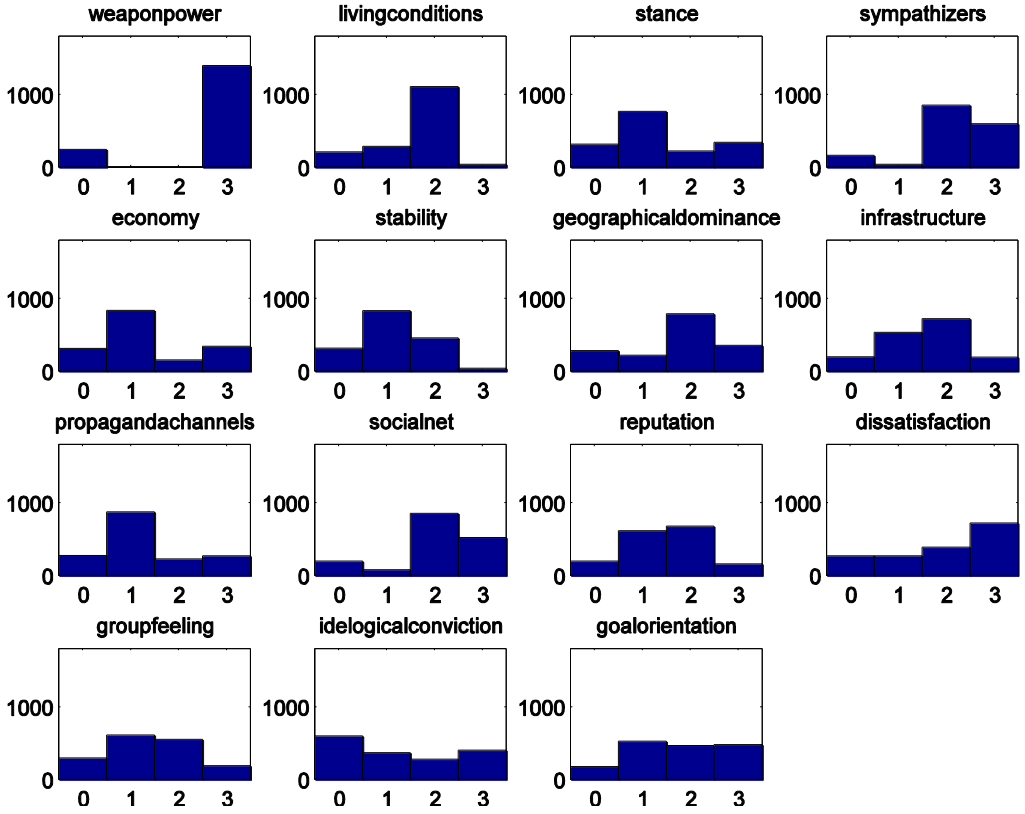


Fig. 25. Histograms of the distributions of the 15 different parameters. Each histogram bar is summed over all actors and actions for the best plan.

The plans that were generated were of different sizes since some of the 43 actions may not be performed in some plans. On average plans executed 39 actions, see Fig. 26. Note that a shorter plan may give a lesser *g*-value, but this does not guarantee that the *h*-value will also be small.

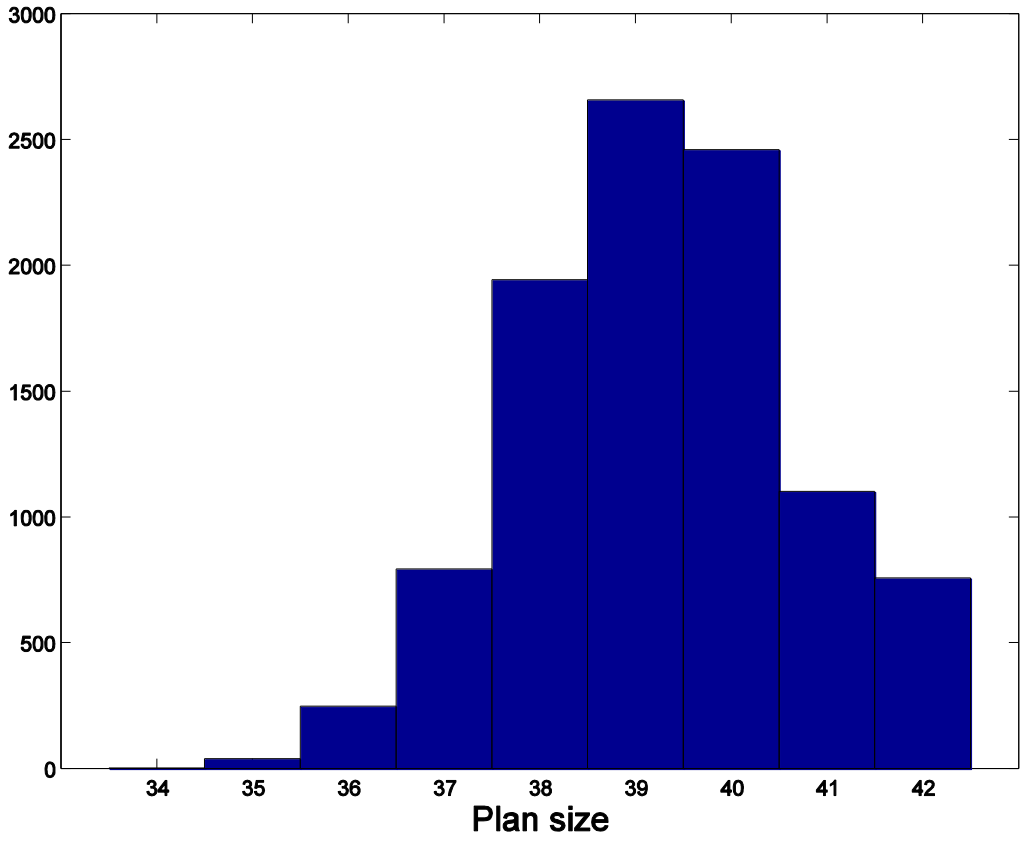


Fig. 26. The distribution of plan sizes for the 10 000 plans. The plan sizes differ from plan to plan since a varying number of actions are executed.

In Fig. 27, Fig. 28 and Fig. 29 we see the *f*, *g* and *h*-values for the actual plan sizes. For the *g*-value we see that the more actions a plan has, the higher the value becomes which is natural because more work has been done in the operation. This in turn will affect the *f*-value which will grow proportionally. What can be noted in Fig. 29 is that the *h*-value seems to drop as the plan sizes grow. However, there are some outliers with small *h*-values and fewer actions executed, e.g., one very effective plan was found with only 38 actions executed, Fig. 29.

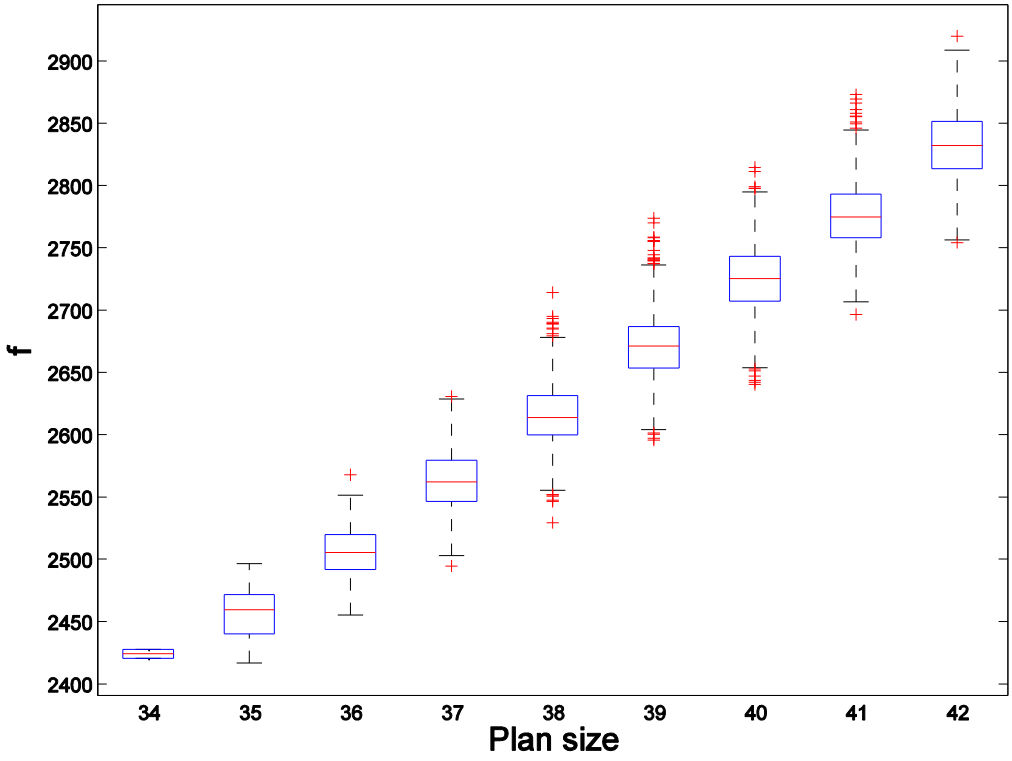


Fig. 27. Box diagrams over average *f*-value for different plan sizes.

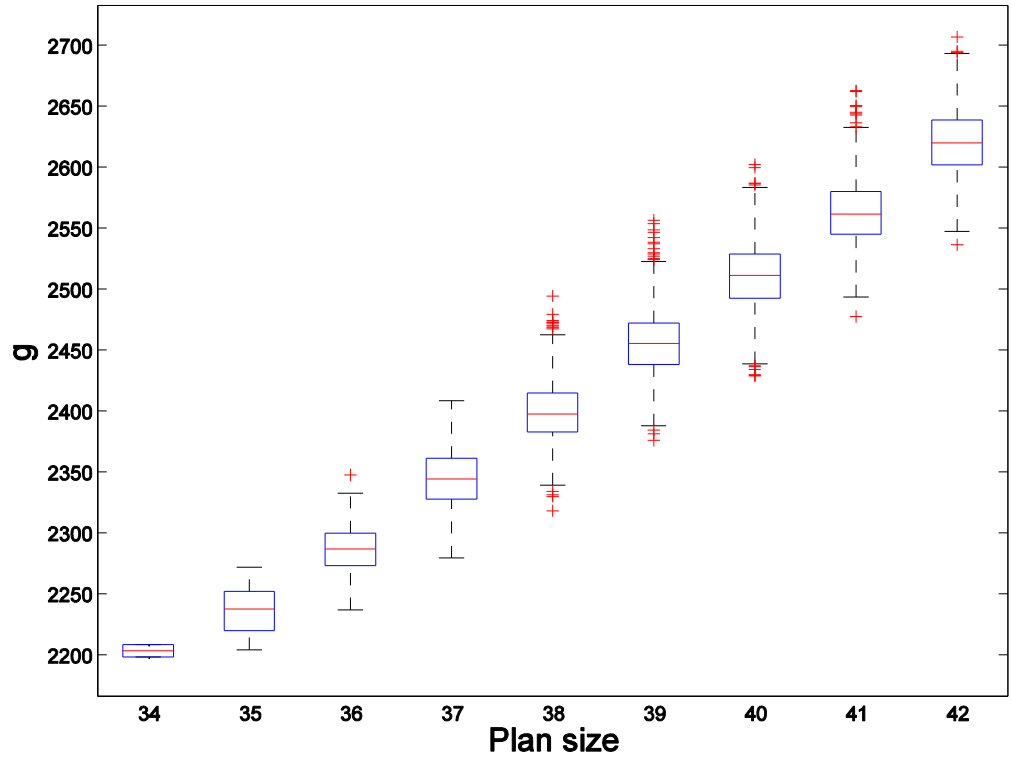


Fig. 28. Box diagrams over average *g*-value for different plan sizes.

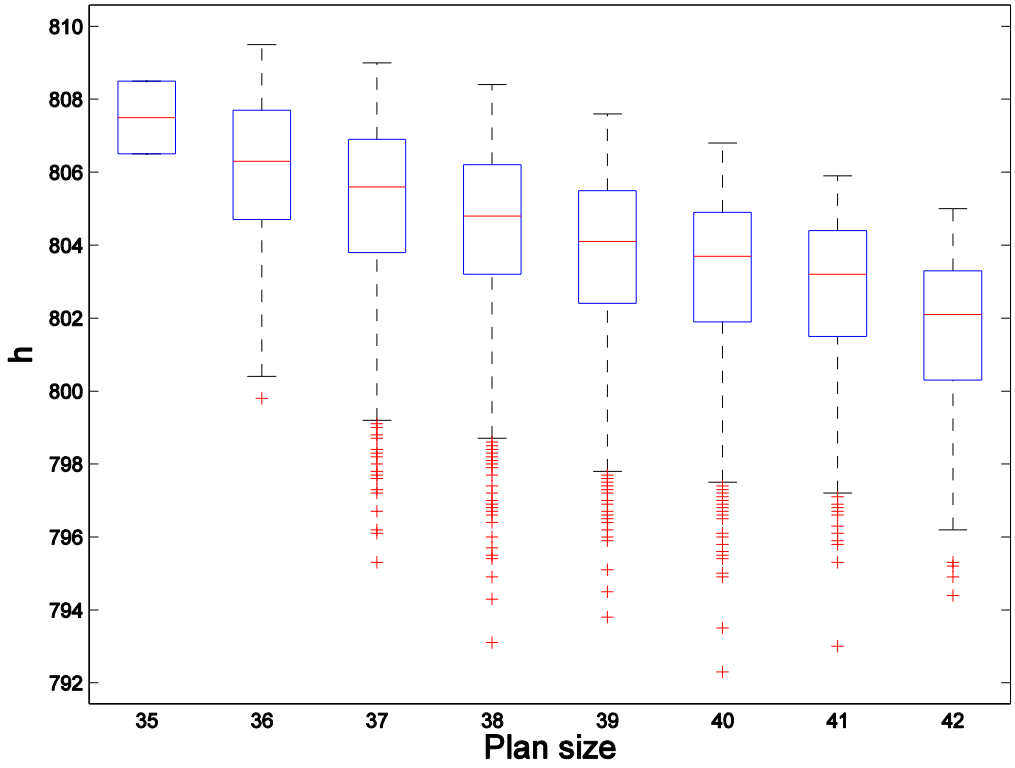


Fig. 29. Box diagrams over average *h*-value for different plan sizes.

After simulating 10 000 plans we plotted the *f*-values in ascending order, see Fig. 30. Here we can see that in practice the best plans (lowest *f*-values) were obtained after a few hundred simulations.

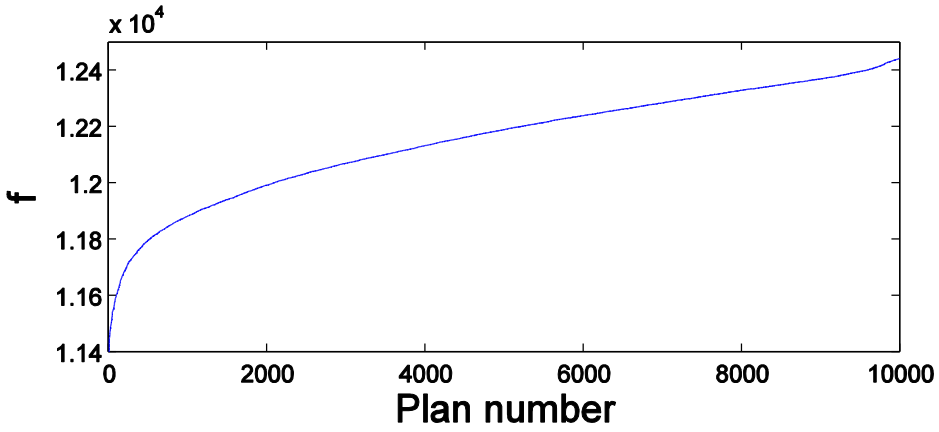


Fig. 30. The different *f*-values for 10 000 sorted plans.

9 Discussion

It is easy to observe from the analysis that progress is made by the best plan. A comparison between the two bubble charts of Fig. 15 and Fig. 16 demonstrates the progress made. However, from the time series of *h*-values in Fig. 14 we see that the progress made is far from the progress we try to attain. This analysis alone demonstrates to plan developers that only a small step is being taken in the right direction, i.e., *h* is lowered by approximately 5%. They need to develop better and many more actions to approach the end state. One interesting observation is that while there are 1.722×10^{361} different states to the scenario there are only 2.164×10^{23} possible plans in the experiment. As each plan will end up in one scenario state it is virtually impossible to exactly reach the end state which is a single state in the scenario.

From requirements of having a robust set of alternative plans it is necessary to alter the traditional A*-algorithm. First, we decide not to stop the algorithm when the first complete plan is evaluated, instead we continue to evaluate more plans to find a robust set of plans. Secondly, it is necessary to introduce a weight in the calculation of *f* as the plans under evaluation never reach the end state; we use $f = g + 80h$. This is domain dependent a may be altered. If plans evaluated are more successful the weight will be lowered.

While it is obvious that we may achieve that which we optimize for, we are still surprised of the small variations on the minor effects monitored (but not strived for) in Fig. 13 when there is much action taking place in the scenario as demonstrated by the bubble charts. These effects were developed independently of the end state, i.e., not as partitions of the end state. Thus, they may not lie directly in the path of optimization towards the end state.

In evaluating the impact and importance of different actions it is interesting to compare the sensitivity analysis box plots of Sec. 7.5 with the regression tree analysis of Sec. 7.6. We observe that the action A_{25}, with most negative impact in Fig. 18 is the first action to be split by the regression tree in Fig. 19, and A_7, with the highest 3rd quartile in the box plot is the second action split in the regression tree. Together these methods complement each other as the box plots provide the impact of all actions and the regression tree provides the importance of each action given the splits that are made on previous levels. On the other hand, the regression tree provides a partition of the alternatives for each action at each split.

In addition the regression tree is highly successful in making predictions on the outcome of the simulation on *g* and *h* with errors of circa 1–2%. This is achieved for each plan in milliseconds compared to 24 seconds for simulation of the plan. Thus, once trained the regression tree may act as decision support when many plans need to be evaluated in a short time span during re-planning of a plan under execution.

Finding the border of an operation is analyzed in Sec. 7.7. In Fig. 22 we observe the distance from the best plan towards the borders of other groups of plans during execution action-by-action of the best plan. As these borders are eleven dimensional hyperplanes in $(\mathbb{Z}^+)^{11}$ it is not possible to visualize them for decision makers. Instead we present time series of the distance from the best plan to all neighboring group of plans as actions are being executed step-by-step. If this type of presentation is combined with other information on the evaluated performance of the plans in other groups, this gives commander knowledge on which of the eleven borders should be monitored carefully during plan execution.

We believe that the analysis of borders together with the boxplots and actual monitoring of the progress of an operation are the most important components during plan execution. During the plan development process all methods of analysis present important views on the plan under development.

One additional observation that was made during the project is the need to provide computer system support to SMEs in scenario and plan development. A scenario as large as 40 agents where each agent is modeled by 15 parameters with their internal agendas and external relations, as well as a plan of 43 actions with 109 alternatives is too large to handle manually in an efficient manner. The behavior modeling discussed in Sec. 5.2 reduces the size of the problem by introducing an aggregated generic model. This is a step in the direction towards providing design support to SMEs. However, the development of the plan with all its alternatives was done manually by an SME. While direct design support was outside the scope of the project, it will be crucial to provide computer system support for SMEs developing plan and scenario, which prevents them from making logical errors in operational planning.

10 Conclusion

In this report we demonstrate that it is possible to draw important conclusions about the adequacy of a military operational plan in its ability to achieve a predetermined end state. By modeling alternative plans and a scenario we are able to analyze the best possible plans available within the bounds put forward by military planners through an extensive set of data analysis procedures. We conclude that this analysis will provide decision makers with information on how far the best plans advance towards the stated goal, if they are surrounded by a robust set of alternative plans, and which actions are most important. This gives planners early feed-back during plan development, and commanders information on where to focus their attention during plan execution.

References

[1] Schubert, J., Moradi, F., Asadi, H., Hörling, P. and Sjöberg, E., Simulation-based Decision Support for Effects-based Planning, in *Proceedings of the 2010 IEEE International Conference on Systems, Man and Cybernetics*, 2010, pp. 636−645.

[2] Smith, E. A., *Complexity, Networking, and Effects-based Approaches to Operations*. Washington, DC: Department of Defense CCRP, 2006.

[3] Hunerwadel, J. P., The effects-based approach to operations: Questions and answers, *Air & Space Power Journal* **20**:53–62, Spring 2006.

[4] *Effects-based Approach to multinational operations, Concept of operations (CONOPS) with implementation procedures*, Version 1.0. Suffolk, VA: Unites States Joint Forces Command, 2006.

[5] Farrell, P. S. E., New operations decision support requirements derived from a control theory model of effects-based thinking, in *Proceedings of the 13th International Command and Control Research and Technology Symposium*, 2008, paper 248, pp. 1−17.

[6] *Allied Command Operations Comprehensive Operations Planning Directive (COPD-Trial version)*. Brussels: Supreme Headquarters Allied Power Europe, NATO, 25 Feb. 2010.

[7] Schubert, J., Wallén, M. and Walter, J., Morphological refinement of effect-based planning, in *Stockholm Contributions to Military-Technology 2007*, M. Norsell, Ed. Stockholm: Swedish National Defence College, 2008, pp. 207−220.

[8] Schubert, J., Multi-level Subjective Effects-based Assessment, in *Proceedings of the 13th International Conference on Information Fusion*, 2010, paper We3.4.1, pp. 1−8.

[9] Duda, R. O., Hart, P. E. and Stork, D. G., *Pattern Classification* (2nd Edition). Wiley-Interscience, 2000.

[10] Huang, C. and Darwiche, A., Inference in belief networks: A procedural guide, *International Journal of Approximate Reasoning* **15**(3):225−236, Oct. 1996.

[11] Cozman, F. G., Generalizing variable elimination in Bayesian networks, in *Workshop on Probabilistic Reasoning in Artificial Intelligence*, 2000, pp. 27−32.

[12] Hamming, R. W., Error detecting and error correcting codes, *The Bell Systems Technical Journal* **29**(2):147−160, Apr. 1950.

[13] Wu, F. Y., The Potts model, *Reviews of Modern Physics* **54**(1):235−268, Jan. 1982.

[14] Peterson, C. and Söderberg, B., A new method for mapping optimization problems onto neural networks, *International Journal of Neural Systems* **1**(1):3−22, May 1989.

[15] Bengtsson, M. and Schubert, J., Dempster-Shafer clustering using Potts spin mean field theory, *Soft Computing* **5**(3):215−228, Jun. 2001.

[16] Schubert, J., Clustering belief functions based on attracting and conflicting metalevel evidence using Potts spin mean field theory, *Information Fusion* **5**(4):309−318, Dec. 2004.

[17] Ahlberg, A., Hörling, P., Johansson, K., Jöred, K., Kjellström, H., Mårtenson, C., Neider, G., Schubert, J., Svenson, P., Svensson, P. and Walter, J., An information fusion demonstrator for tactical intelligence processing in network-based defense, *Information Fusion* **8**(1):84−107, Jan. 2007.

[18] Rosling, H., Stats that reshape your worldview, *Technology, Entertainment, Design*, 2006. [Online] http://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen.html (January 2013)

[19] Schubert, J. and Hörling, P., Explaining the Impact of Actions, in *Proceedings of the 15th International Conference on Information Fusion*, 2012, pp. 354−360.

[20] Breiman, L., Friedman, J., Stone, C. J. and Olshen, R. A., *Classification and Regression Trees*, Boca Raton: Chapman and Hall, 1984.

[21] *Statistics Toolbox*. Natick, MA: The MathWorks Inc., 2012.

[22] Cortes, C. and Vapnik, V., Support-vector networks, *Machine Learning* **20**(3):273–297, Sep. 1995.

[23] Schubert, J. and Linderhed, A., Learning boundaries on military operational plans from simulation data, in *Proceedings of the 2011 IEEE International Conference on Systems, Man and Cybernetics*, 2011, pp. 1325−1332.

[24] Chang, C.-C. and Lin, C.-J., LIBSVM: a library for support vector machines, in *ACM Transactions on Intelligent Systems and Technology* **2**(3), article no. 27, Apr. 2011. [Online] http://www.csie.ntu.edu.tw/~cjlin/libsvm (January 2013)

FOI, Swedish Defence Research Agency, is a mainly assignment-funded agency under the Ministry of Defence. The core activities are research, method and technology development, as well as studies conducted in the interests of Swedish defence and the safety and security of society. The organisation employs approximately 1000 personnel of whom about 800 are scientists. This makes FOI Sweden's largest research institute. FOI gives its customers access to leading-edge expertise in a large number of fields such as security policy studies, defence and security related analyses, the assessment of various types of threat, systems for control and management of crises, protection against and management of hazardous substances, IT security and the potential offered by new sensors.

FOI
Defence Research Agency
SE-164 90 Stockholm

Phone: +46 8 555 030 00
Fax: +46 8 555 031 00

www.foi.se