Operations Research Perspectives 2 (2015) 36–56

Contents lists available at ScienceDirect

Operations Research Perspectives

journal homepage: www.elsevier.com/locate/orp

Simulation-based decision support for evaluating operational plans☆

Johan Schubert[a,*], Farshad Moradi[a], Hirad Asadi[a], Linus Luotsinen[a], Eric Sjöberg[b], Pontus Hörling[a], Anna Linderhed[c], Daniel Oskarsson[d]

[a] Department of Decision Support Systems, Division of Information and Aeronautical Systems, Swedish Defence Research Agency, SE-164 90 Stockholm, Sweden
[b] Department of Strategy and Policy, Division of Defence Analysis, Swedish Defence Research Agency, SE-164 90 Stockholm, Sweden
[c] Department of Sensor Informatics, Division of Sensor and EW Systems, Swedish Defence Research Agency, P.O. Box 1165, SE-581 11 Linköping, Sweden
[d] Department of Capabilities and Concepts, Division of Defence Analysis, Swedish Defence Research Agency, SE-164 90 Stockholm, Sweden

ARTICLE INFO

Article history:
Available online 4 March 2015

Keywords:
Computer simulation
Decision support systems
Data analysis
Operational planning
Effects-based planning
Effects-based approach to operations

ABSTRACT

In this article, we describe simulation-based decision support techniques for evaluation of operational plans within effects-based planning. Using 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 objective of this study is to examine the potential of a decision support system that helps operational analysts understand the consequences of numerous alternative 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 examine several different alternative ways to perform the action. We use a representation where a plan consists of several actions that should be performed. Each action may be performed in one of several different alternative ways. Together these action alternatives make up all possible plan instances, 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 several alternatives for these actions, as well as a scenario of 40 group actors. Decision support for planners is provided by several methods that analyze the impact of a plan on the 40 actors, e.g., by visualizing time series of plan performance. Detailed decision support for finding the most influential actions of a plan is presented by using sensitivity analysis and regression tree analysis. Finally, a decision maker may use the tool to determine the boundaries of an operation that it must not move beyond without risk of drastic failure. The significant contribution of this study is the presentation of an integrated approach for evaluation of operational plans.

1. Introduction

In this article, we describe the development of a simulation-based decision support using an event-based simulation that models military operational plans according to effects-based planning (EBP). The developed methods can be used in an incremental manner to test plans as they are developed step-by-step and new activities are added. This is the problem faced by planners of operations planning. To support the planners we develop a simulation-based decision support system within the effects-based approach to operations (EBAO). Chandrasekaran [1,2] suggests that in situations with high degree of uncertainty, the aim of simulation-based decision support systems should not be to find optimal options but rather options that are robust, i.e., less sensitive to uncertainties. In such cases, it is impossible or highly improbable to build models that give an exact prediction of outcomes for running different options. The main reason for this observation is that the modeler must make many assumptions and guesses during the development of the models in order to keep the size and complexity of models reasonable. In such cases, the models should support the decision-makers, e.g., military planners, to explore different options based on their sensitiveness to uncertainties. Bankes [3] refers to these types of models as exploratory models (EM). An EM is a model that displays the behavior of a real system given certain assumptions, and by running several such models, the planner can explore the significance of those assumptions. According to Bankes [3] there

☆ This work was supported by the FOI research project "Real-time Simulation Supporting Effects-based Planning", which is funded by the R&D program of the Swedish Armed Forces and represents a major extension to preliminary work presented at IEEE SMC 2010 (Schubert et al., 2010).
* Corresponding author. Tel.: +46 8 55503702.
E-mail address: johan.schubert@foi.se (J. Schubert).

http://dx.doi.org/10.1016/j.orp.2015.02.002

are three different types of EM: Data-Driven EM, Model-Driven EM, and Question-Driven EM. The models presented in this work are Question-Driven EM, and the decision maker's aim is to investigate alternative plans and highlight consequences of different operational courses of action.

How planners model a phenomenon depends on the purpose of the model and the questions that they want to answer. Because our simulation system aims to support decision-making within an effects-based approach to operations [4,5], the modeling is based on the concepts of EBAO as a set of effects and actions that together will lead to a desired military end state. The EBAO concepts will be explained in Section 2. 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 such a desired military end state. The objective is to evaluate plans against a large set of actors and understand their consequences by simulating the events and producing outcomes that result from making alternative decisions with respect to those actions. Each plan consists of many actions, many of which can be performed in a number of alternative ways. Providing information for probable consequences of the choice of different options by the decision-makers is known as Option Awareness (OA) [6]. Pfaff [6] states that OA augments situational awareness by illuminating the components of the decision space via exploration of different options using, e.g., simulation models. We model the plan and evaluate alternative plan instances based on how well they are able to drive the entire state of a simulation model that simulates a large set of actors towards a predetermined military end state. These plan instances are evaluated based on their performance and are clustered into subsets in which all plan instances have both common characteristics and outcomes. The idea is that these clusters, if they contain plan instances of good performance, present a robust set of alternative plans that can be used for minor dynamic re-planning whenever necessary. However, an important issue to consider in deploying these models is that they are only representations of the real world, and a gap will always exist between the model and the reality it represents. The size of this gap depends on a number of errors that are primarily caused by various assumptions with respect to the real world, e.g., if critical information is missing or inaccessible. Consequently, running these models using computer simulation will result in uncertain outcomes. A good example of such situations is military planning with a significant amount of uncertainties. We try to manage this issue by using stochastic models where uncertainty is large and using a Monte Carlo approach within the simulation where each instance is simulated 20 times.

Actors and actions are modeled in 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, their reactions, and possible follow-on interactions. Because the actions may have several different alternatives with which they can be carried out, these alternatives together span an action tree. In this searchable tree, each level in the tree corresponds to an action, and each node in the tree represents an alternative for that action. As the action tree is searched, each node is evaluated using the simulator and the results found are stored. In using the search to guide the tasks of the simulator, we allow the simulator to operate 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 (that arise from all possible combinations of action alternatives). Simulated plans that are similar in both their structures 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 that adds new actions or new alternatives to existing actions become necessary as the situation evolves.

Decision support is achieved through a series of statistical analyses, information fusion, machine learning, and information visualization techniques. For example, we develop methods for effects and end-state time series visualization for easy overview of the time development of several alternative plans as action-by-action progress is 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 that moves the state of away from the end-state, leaving the decision-maker free to focus his attention on the set of remaining actions. By systematically varying one action at a time and 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 plan. This sensitivity analysis displays the relative level of importance of making the correct selection of alternatives for each action. Using the explanation function, a decision-maker can be informed as to which actions of the plan are crucial to its success.

In addition to the planning process before the execution of a plan, we may have re-planning (see Appendix) during execution if the plan goes astray. We differentiate between minor and major re-planning, where minor re-planning is when only a few existing actions need adjustment and major re-planning is when the decision maker needs to restart the planning process from scratch. To differentiate between minor re-planning and situations in which major re-planning becomes necessary to avoid drastic negative consequences of plans that begin to deviate substantially from the initial planning, we adopt indicators as warning bells. An indicator represents the boundary between two clusters beyond which drastic changes can occur. We determine the boundaries using simulated data from alternative plan instances 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 in a step-by-step manner as the actions are executed. By visualizing the change in distance during simulation execution a commander can observe whether the operation is approaching a boundary beyond which outcomes may be uncertain. If minor re-planning is not sufficient to avoid approaching a boundary, major re-planning must take place that may include developing entirely new actions.

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

The goal of the article is, after first describing the planning approach and the type of situations we might be encountering ("the Bogaland scenario"), to thoroughly describe the components of a simulator designed to find which plans, synthesized from a set of modeled actions could best solve the problems in Bogaland. In Section 2, we describe the effects-based planning approach. Section 3 presents an overview of the Bogaland scenario used for experimentation. Section 4 describes a simulation control approach in which a decision-maker steers the computational focus of the simulator. Section 5 subsequently model actors and actions, and a simulation methodology is developed in Section 6. A decision support methodology is presented in Section 7, and the simulation results are analyzed in Section 8. Finally, Section 9 provides a discussion of the approach, and Section 10 presents the conclusions.

In this article, we use many concepts that characterize the studied field. Instead of introducing and exemplifying these concepts where they first appear in the text, we have collected them all in the Appendix.

2. Effects-based planning

The modeling is based on EBAO and the concepts used within it, i.e., plan, action, effect, end state, etc. The EBAO is a military approach to management and implementation of efforts at the operational level. According to the United States Joint Forces Command

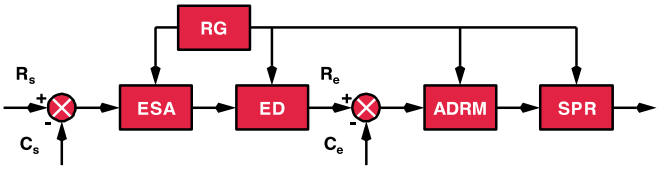


Fig. 1. Effects-based planning, including sub-processes End State Analysis (ESA), Effects Development (ED), Action Development and Resource Management (ADRM), Synchronization and Plan Refinement (SPR), and Red Green Teaming (RG).

(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" [7].

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 and that are conceptually developed starting top-down from a desired end state. The methodology in EBP is iterative in nature, and the development of the plan is constructed step-by-step and tested as it gradually emerges. To provide decision support for this planning work, we develop methods that can be used iteratively during successive modeling of different elements of the plan and testing via simulation and evaluation against a scenario with operator models that react to the execution of plan elements. It is thus possible to measure the change in state of all the actors relative to the desired end state.

A control theory model of EBP [8] is shown in Fig. 1. As input, we use the required situation R_s, which is compared with the current situation C_s received from the assessment. The first process is an end state analysis (ESA) followed by effects development (ED). Initially, if 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 effect C_e, and is also received from assessment.

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

For the Allied Command Operations Comprehensive Operations Planning Directive (COPD) [9], we focus the simulation-based decision support for generation and testing of alternatives at the Joint Force Command (JFC) Operational Concept Development (JFC Phase 4a). This approach does not exclude the use of these methods at an earlier strategic level.

The similar concept of comprehensive approach (CA) has recently received additional focus. This approach differs to a certain extent and is allowed to do so between nations and organizations such that we actually address several comprehensive approaches. However, these approaches all have a common goal of coordinating own and friendly forces and allies in a conflict area, both military and civilian, in combined and joint efforts intended to reach a stable peaceful situation. The British MoD has formulated CA as "*Commonly understood principles and collaborative processes that enhance the likelihood of favourable and enduring outcomes within a particular situation*" [10]. NATO describes CA as "*an orchestration of communication of all activities in a country, coming to a well defined and well understood end state*" [11]. More specifically, the idea is that all actors should work in a coordinated manner with a common agreed end state as the final goal in the *political*, *military*, *economic*, *social*, *infrastructure*, and *information* (PMESII) situations viewed as an overall system within a country.

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

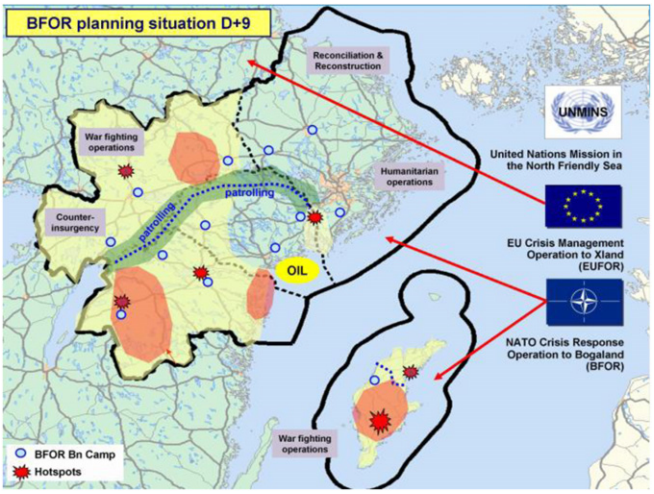


Fig. 2. The Bogaland test scenario.

"Viking" Exercises. The scenario is composed of several fictitious countries, two of which have been described in-depth, i.e., Xland and Bogaland. 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 (see Fig. 2).

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 regions of the country.

In Bogaland, which is a newly industrialized country, a civil war broke out ten years ago when discontent within the minority ethnic-religious group reached notably 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 rather low levels. The country was split into two regions, roughly along ethnic lines, with each region forming 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 known as BFOR is present to support the implementation of the agreement. Irregular groups (i.e., non-government combat 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 that we analyze corresponds to selecting a sequence of actions from sets of alternatives. Most actions contain between two and eight alternative methods of execution and one alternative is chosen for each action. Thus, a chosen sequence of alternative actions constitutes a plan for attempting to reach a peaceful end-state in Bogaland. The number of possible plans can theoretically grow extremely large because each combination of alternative actions for the different actions will constitute a separate plan. In our planning problem which finds some of the best

plans to solve the problems in Bogaland by simulation, we have designed 43 actions and a total of 109 alternatives for these actions. This theoretically constitutes 2.164×10^{23} possible plans if all possible combinations are counted. Of course, in practice, many of these plans can be ruled out because they start out with a sequence of action alternatives that lead to early failure, i.e., status quo or even moving away from the end-state we try to reach.

The simulator is free to choose how to sequentially combine action alternatives to plans to evaluate in its search for the best plans, an incremental process which is further described in Section 6. However, it is also possible to give the simulator instructions on how to select the combinations of action alternatives it should prefer during simulation. In this way, it is possible to guide the simulator to plans, among all possible plans, that meet the decision-makers requirements. There are three complimentary approaches to managing the simulator's efforts towards plans with actions that:

- are executed within a specified geographical area,
- are executed within a specified timeframe, or
- may lie outside the specified area and timeframe if they strongly influence actions within the area and timeframe [12,13], as described by a cross-impact matrix (CIM) [14,15].

The first two approaches direct the focus of the simulation on actions that are spatiotemporally limited to a preferred frame.

The third approach uses the CIM. It is a matrix set up for all actions that specifies how much the actions counteract or support each other during execution due to resource conflict or the extent to which one action lays the foundation for another, etc. With this technique, all phenomena are placed against each other in a matrix, and their constrained probabilities are calculated, i.e., what is the probability of a particular phenomenon given that another has just occurred? Since Gordon and Hayward's original article in 1968, research in the area of cross-impact analysis has developed in several directions, and the two main themes are statistical and qualitative. The statistical line begins with Gordon and Hayward [14]. Under this approach, Turoff [16] presented an alternate cross-impact analysis based on the impact one event has on a different event. Dalkey [17] derived the requirement for a CIM to be consistent, and Duperrin and Godet [18] developed a new method based on cross-impact analysis to improve the decision-making process. Recently, Bañuls and Turoff [19] extended the original method of Turoff to allow the user to work in an incremental manner and set an own stopping point when analyzing a scenario. Additionally, Bañuls et al. [20] extended the methodology to collaborative group scenario generation. In this article, we follow the qualitative line of research. Within this framework, Jeong and Kim [21] developed a qualitative cross-impact model built on fuzzy relationships. The method allows definition of the impact in linguistic terms or directly with fuzzy relationships. Asan et al. [22] also proposed a fuzzy methodology for cross-impact analysis and used four different analysis methods in parallel to each research question followed by comparison of the results. Parashar et al. [23] use a fuzzy cross-impact simulation in which the interaction in a studied system is represented by a CIM with linguistic terms. The methodology is used to visualize the dynamic development of the system in question. Among other developments, Enzer [24] combined cross-impact analysis with the Delphi methodology [25], Helmer [26] studied non-commutative problems, and Amara [27] studied the impact between events in which the sequential order of events is crucial. Bloom [28] went one step further and developed a time-dependent cross-impact model.

In this article, we combine the idea of multiple alternatives from morphological analysis [29] and the methodology of a qualitative cross-impact analysis to analyze an operational plan. Because we study our own operational plan in which we make decisions, we will not model different alternative outcomes of a particular action but instead examine the alternative specific ways of implementing the action within the operational plan. We combine this approach with the idea of a qualitative analysis of a CIM in which we take into account how all of the alternatives between two different plan elements affect each other. Because our use of this method is qualitative, it is not a question of conditional probabilities between different outcomes but the impact (whether positive or negative) between the different plan elements. We use a scale of integers from -9 to $+9$, which provides sufficient ability to describe the strength of influences among all plan elements. These impacts are used in a clustering process [30] to find sets of actions that influence each other, i.e., the blue and red action groups in the lower left of Fig. 3.

Shown in Fig. 3 is the graphical user interface where we make each of these three (spatial, temporal and CIM connections) 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 influences given to other actions and received from other actions 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 items for each action. The final weight for an action, which will be used for its importance in the simulation, is the product of these three weights. The simulator will be focused to find plans relative to these weights. In Fig. 3, six action alternatives out of the twelve zoomed got highest priority. In the simulation there are 43 actions with 109 alternatives altogether, but we have zoomed in on the figure for clarity.

5. Modeling actors and actions

In this work, we employ an actor modeling approach targeted towards modeling of aggregate entities of human groups and organizations, i.e., 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 in this work 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 that are known to the actor in question (including reference to itself) are represented using separate sets of the above mentioned state variables. The purpose of the state variables is to provide a common knowledge representation that can be used in 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. For instance the actor BFOR has a set of state variables that represent its physical capabilities, such as weapon power, and state variables that represent its mental capabilities such as group feeling. There is also a set of variables that define its relationship to other actors, e.g., it is friendly towards the civilian population and the local government, but it is suspicious towards irregular groups.

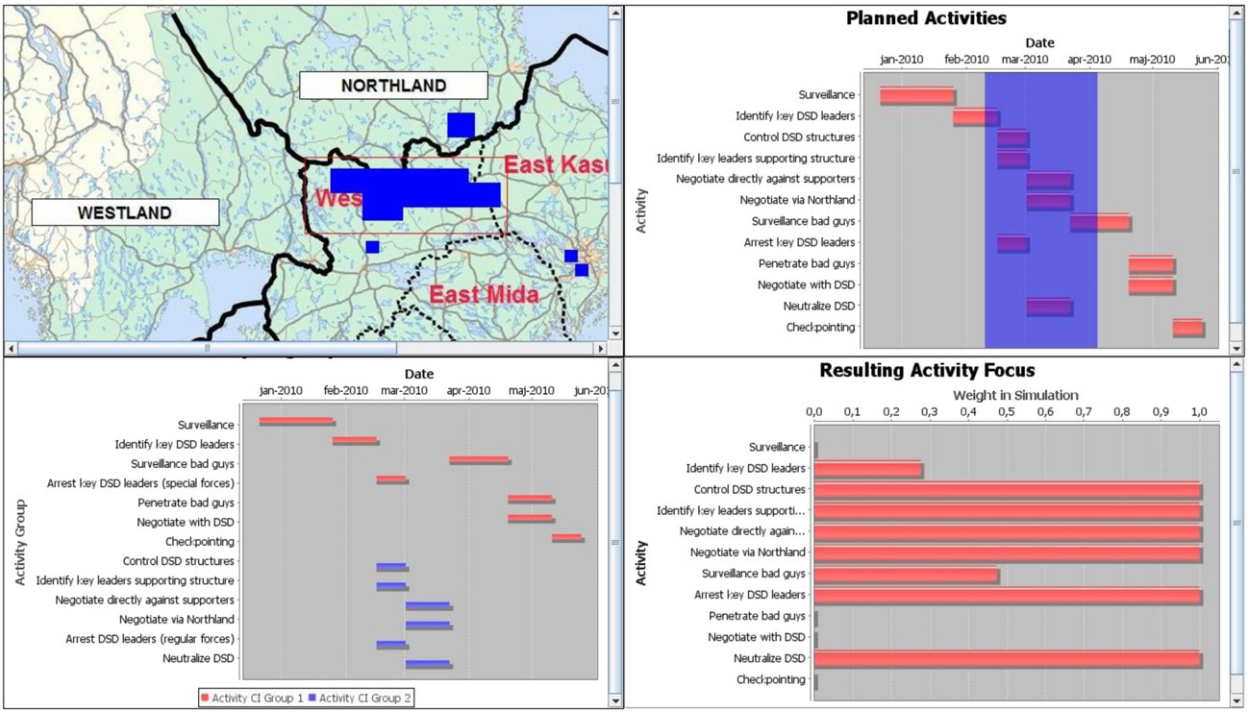


Fig. 3. Simulation control tab of the GUI: *Upper left:* Each action alternative has an area in which it is executed in Bogaland (blue filled rectangles). Selection of the geographical focus area is performed with a red rectangle. Upper right: Similarly, selection of focus timeframe (purple). *Lower 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 action alternatives that give their importance in the simulation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Internal state variables (parameters).

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, 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	1000–10 000 times GNP/capita	10 000–100 000 times GNP/capita	$>$100 000 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, cell-phones	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	Wants to see responsibility 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

The internal state variables, which originate from previous work [31], are represented in this work 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 are presented in Table 1. The variables in the internal state vector are limited to four integer values $[0, \ldots, 3]$. For instance, an actor with regard to its "Reputation" can be either 0 (Despised), 1 (Light-weight), 2 (Recognized) or 3 (Highly regarded). We believe that this design decision ensures that the modeling efforts remain pragmatic and not excessively time-consuming. Additionally, 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 article.

Similar to the internal state of an actor, the actor's 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 reason is that not all actors are known to each other either because of geographical proximity or the fact that some actors intentionally strive to be unknown. The relationship variables take on four integer values $[0, \ldots, 3]$ that 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 article. Another actor a_i known to the actor is represented by $\omega_i = \{I_{ij},\ R_{in}\}$, where $j = \{A,\ B, \ldots, O\}$ and $n = \{1,\ 2, \ldots, N\}$. In other words, I_{ij} represents actor a_i's internal state variable j, and R_{in} represents actor a_i's relationship to actor a_n. Given that the actor knows about N actors (including itself), its complete knowledge space is $\Omega = \{\omega_1, \ldots, \omega_N\}$.

Let us also introduce the concept of roles that is used 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 whose 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 uses the actor's state variables Ω as input and generates the alternative α to execute as output, as shown in (1). For instance, if actor A attacks actor B then B might respond differently depending on its weapon power. If B feels that it is inferior to A with regard to weapon power it is more probable that it surrenders or retreats.

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

In this work, a Bayesian network (BN) [32] approach is adapted to model f using subject matter experts. We have chosen to use BNs due to the following properties:

- Ability to graphically represent actor behavior using directed acyclic graphs (DAGs), which ultimately improve the general understanding of the model both for the developer and the user with regard to validity and reliability of the model,
- Ability 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 [32]. We have,

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

Using the Bayes rule, a probability value known as the posterior is calculated for each action available to the actor. Typically, the action with the maximum posterior is selected by the actor; however, this is 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. In other words, 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 sufficient data to model the likelihood function due to the high dimensional state variable vector Ω in our case. In this case, the BNs come to the rescue by introducing conditional independence between variables and hence simplifying the likelihood estimation process.

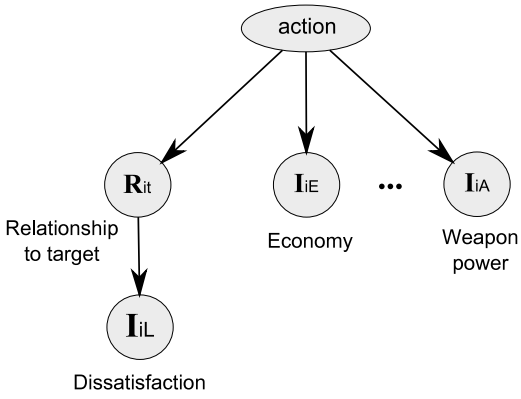


Fig. 4. Naïve Bayes classifier in which I_{iL} is conditionally dependent on R_{it}. All other state variables are assumed to be independent of each other.

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),

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

where K is the dimensionality of Ω. The DAG of an example naïve Bayes classifier BN is presented in Fig. 4. As it is depicted in the figure the value of an action is based on the value of internal state variables and relationship state variables.

However, not all variables in Ω are conditionally independent of each other. As an example, an actor a_i's dissatisfaction I_{iL} with another actor a_t is conditionally dependent on its perceived relationship R_{it} to a_t, Fig. 4. Links between any two variables in the DAG indicate that a conditional dependency exists 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 [33]. In this study, we chose to use the algorithm presented in [34]. It is important to understand that the time required to infer probabilities varies depending on the structure of the BN as well as the amount of evidence (or knowledge) known prior to inference.

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

- Maximum 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

While behavior modeling presents a method for selecting proper actions for an actor based on the values of its state variables, the action modeling explains how the state of an actor is changed when an action is performed. Hence, actions are the means by which an actor may alter its state Ω. An action is represented in this work by a set of rules, each consisting of a condition (i.e., the

if component) and a list of effects (i.e., the *then* component) such that if the condition is *true*, then the list of effects will execute, ultimately resulting in state variable changes. However, if the condition is *false*, then none of the effects will execute.

In this work, subject matter experts have developed hundreds of rules to model 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 to model certain phenomena, i.e., 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* components) of the rules are described using Boolean expressions. The effects (*then* components) of the rules are described using a function notation in which *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.

6. Simulation methodology

The simulation contains the plan that is to be evaluated, the environmental data, and models of participating actors and their initial states, including a probability distribution for different actions that they are capable of carrying out. In this context, a plan 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 that 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.

To describe the simulation process, we define the system state S_n as the combination of all actors' state variables and all environment parameters. Next, consider action A_n, which 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 actions, 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 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 in which the outcome of an action depends on a number of random variables drawn according to selected distributions. The disadvantage of this model 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. 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 are absolute values at the beginning, 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 theater, 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 the end state. 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) must 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.

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 t_n:*
  - *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. 5. Monte Carlo algorithm.

Our Monte Carlo simulation is therefore structured as shown in Fig. 5.

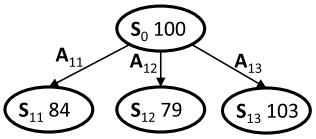
During the actual time interval $[t_{n-1}, t_n]$, our action A_n is initiated. Probable external events are chosen in the same way and placed in the event list according to their given distributions. The action A_n is observed by the other actors immediately or eventually via an information channel. 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 with histograms and serve as an approximation for the respective output distribution.

6.1. A*-search

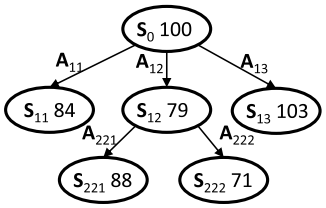
The purpose of our simulation system is to search for a sequence of actions that best suits the decision-maker's desired end state. However, we also want the simulation to be capable of suggesting an alternative solution at any moment in time. Hence, such a simulation system can neither be designed according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

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

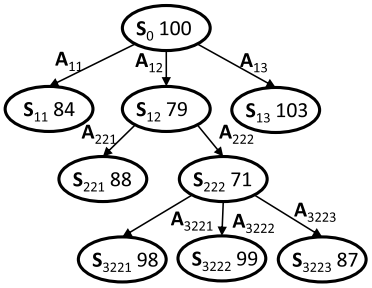
where $g(n)$ is the total distance from the starting position to the current location, and $h'(n)$ is the estimated remaining distance



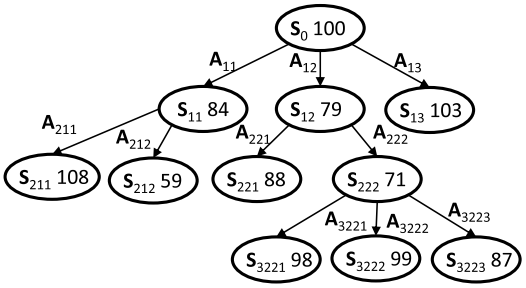
Step 1: From the initial state all available alternatives are simulated. S_{12} appears to be "closest" to the target.



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



Step 3: 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. 6. An example illustrating the four first steps in a simulation of a plan beginning with initial system state S_0 with a distance of 100 to the desired end state. The available action alternatives A_x are executed successively in the currently most favorable plan option.

from the current position to the goal (end) state. A heuristic function is used to create this estimate of the distance to the goal state. The function $f'(n)$ is the sum of $g(n)$ and $h'(n)$, i.e., the current estimated shortest path. The true shortest path $f(n)$ is not discovered until the A*-algorithm is finished. This means that based on a given system state, we simulate the effect of each alternative action in our plan but only one step at a time. Thus, for every alternative, we obtain a new system state whose distance to the desired end state can be 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 shown by Fig. 6, step 3.

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 had been passed over in favor of a better sequence of alternative actions. The best sequence thus becomes the basis for the next simulation step. At any time, the user can ask for the sequence which at that time appears to be the best, i.e., the sequence of alternative actions that leads to a simulated state that 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 sufficient execution time, all options will eventually be investigated. For the tool to function in this manner, 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 defining a credible distance function. The solution 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", 0.5 is 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 known as $S_{0,0}$, and the end state is referred to as 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

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

During the simulation, an assessment is carried out for how well each action is performed. This process is performed 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 [31]. 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, the current state is

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

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

7. Decision support methodology

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

We observe the difference in consequences between two plans, and we compare the difference in the incremental changes of g and h, referred to as 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. For each A_k, we obtain,

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

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

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

where 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, which takes on an integer value that is the number of the alternative chosen for this action, e.g., $P_1.A_3 = 41$ implies that action number 3 of plan number 1 executes alternative number 41.

In addition, we must measure the structural distance between two plans, and this is accomplished using the Hamming [35] 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,

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

In this work, 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" if all three measures are at minimum.

We partition the set of all simulated plans into clusters using the Potts spin model [36] in such a manner as to minimize the overall sum of all interactions J_{ij}^{-} within each cluster.

The Potts spin problem consists of minimizing an energy function

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

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 if plan P_i and P_j are in the same cluster.

For computational reasons, we use a mean field model in which spins are deterministic with $V_{ia} = \langle W_{ia}\rangle$, $V_{ia} \in [0, 1]$ to find the minimum of the energy function. The Potts mean field equations are formulated [37] as

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

where

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

and T is a parameter known as the temperature used to control the influence of the interaction; this is a system parameter initialized to

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

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

$$M_{ij} = J_{ij}^{-} - \gamma\delta_{ij}. \tag{20}$$

To minimize the energy function, (17) and (18) are iterated until a stationary equilibrium state has been reached for each temperature. Next, the temperature is lowered in a step-by-step manner by a constant factor until $\forall i, a.\ V_{ia} = \{0, 1\}$ in the stationary equilibrium state, as shown in Fig. 7 [30,38].

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 section, 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 [39].

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

Klein [40] demonstrated that the ability of decision-makers to identify more robust options increases substantially through visual comparison. Instead of performing mental simulations of different options, which is a highly error-prone process, Klein [40] allowed the decision-makers to obtain a clear picture of the relationship between available options using graphical visualization of the consequences of performing those options using simulations. However,

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) $= \frac{1}{K} \cdot \max(-\lambda_{\min}, \lambda_{\max})$, 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:

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

  UNTIL-2

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

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

UNTIL

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

RETURN

$\{\chi_a | \forall S_i \in \chi_a, \forall b \neq a. V_{ia}^{s} > V_{ib}^{s}\}$.

Fig. 7. Potts spin clustering of simulated plans partitions the set of simulated plans into clusters of similar plans.

the authors believe that presenting the decision space such that the decision-makers receive efficient support in viewing synergies and patterns is an ongoing challenge.

In Sections 7.1–7.7, we present a number of different analysis and decision support methodologies that combine analytical and visualization methodologies. The purpose of many of these visualization methodologies is to provide the analyst with an overview of the consequences of the plans in a data exploration mode.

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 encountered a leaf node in the search tree. These recorded paths can later be visualized using our tree visualization GUI, which means that a decision-maker will be able to browse the complete tree and view which nodes were included in the best path in addition to other nodes that were almost included as well as those that were discarded.

In Fig. 8, we are presented with an example of paths that were visualized with the tree visualization GUI. In this figure, 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 located on the next level in the tree, he will continue down the best path; however, if he clicks on node 42 (yellow color), a path of lower quality will be selected.

If the user clicks on node 61, its children on the next level are shown (see Fig. 9). In this figure, 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.

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

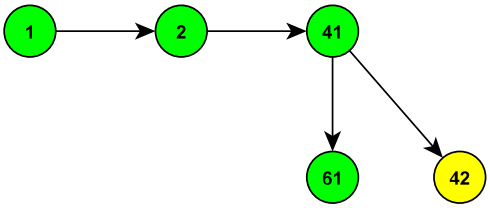


Fig. 8. Tree visualization of a plan (level 4). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

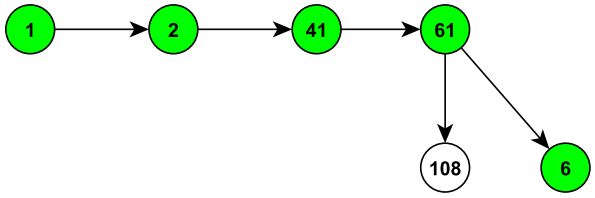


Fig. 9. Tree visualization of a plan (level 5).

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

7.2. Visualizing the best plan effects time series

During plan execution, it is valuable to analyze whether the entire operation is approaching the desired *end state*, and one way to determine whether this is the case is to define a set of advantageous key states that must be achieved. Achieving such a state (hence, moving from a present worse state) is known as obtaining an *effect*. These actions are often stated on a high semantic level. A typical example effect in the Bogaland scenario could be "Establish order and stability in East Kasuria".

Because effects-based planning will typically involve designating a number of effects described in natural language whose fulfilment 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 steered in the direction of a desired end state. Because the end state is a point in parameter space that the simulation attempts to reach, the effects should be viewed as partitions of the parameter space that the simulation increasingly occupies, and the intersection or center of gravity of these partitions should be located close to the end state. Visualizing the fulfilment of effects along the progression of simulation for the best found plan (i.e., action numbers on the x-axis) should determine whether the best course of action does indeed correlate with stepwise achievement of the designated effects or whether success is best achieved through other paths.

Conversely, monitoring the degree of fulfilment 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 fulfilment of critical effects).

The total actor state is defined at each time by the matrix of 15 parameters with values 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 representation is equivalent to the 15 sub-intervals of the allowed parameter values 0, 1, 2, 3 (or a union of such). The distance from the present collective actors' state to an effect is the sum of the Manhattan (L_1 metric) distances from all actors' current parameter values to the closest points (corners, sides, or hyperplane) of the effect as measured from each actor.

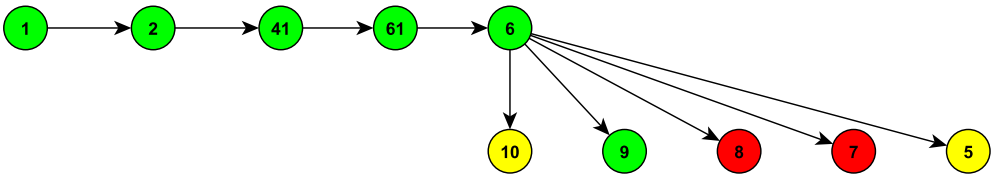


Fig. 10. Tree visualization of a plan (level 6). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

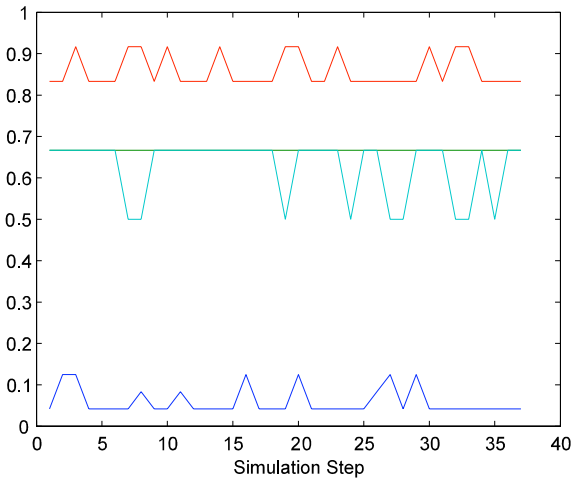


Fig. 11. Time series of supporting effects and decisive points as measured by effect fulfilment.

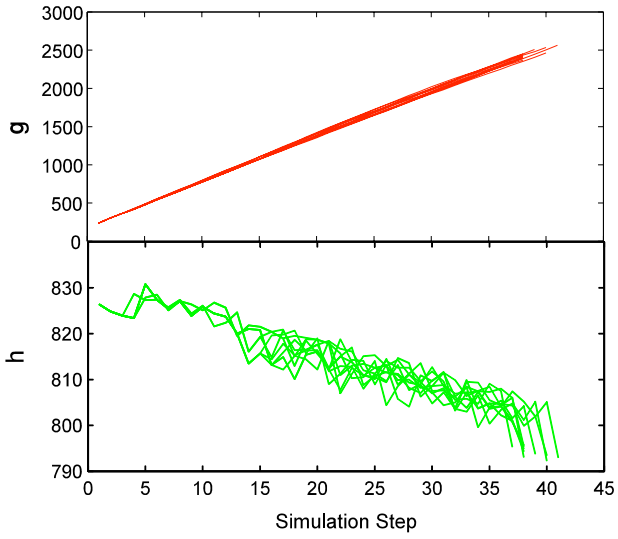


Fig. 12. Functions f, g, and h plotted vs. simulation step for the 10 best plans.

Fig. 11 shows an example of monitoring four effects during the execution of the best plan. In this example, we observe no trends but note that three out of four effects are mostly achieved. This observation is explained by the fact that we are actively striving towards the end state (point) and not towards an effect (volume) in parameter space because effects are only monitored but not actively aimed at in this work. Hence, in this example, we are neither approaching nor distancing ourselves from any of the effects in any major way (as measured by a degree of effect fulfilment calculated from a distance measured in parameter space).

Another method used to assess the probability for achievements of the effects (and finally, the end state) is to use the CIM. In the CIM, the expected impacts of actions on effects and effects on the end-state are stated. Depending on the observed progress of actions, and together with the commanders' own observations of the situation on any effect-level, the probability of achieving these higher effects can be assessed [13].

7.3. Visualizing the multiple plan end-state time series

During the A*-search, the simulator attempts to find a way to traverse the search tree that minimizes f, which is the sum of the distance traveled so far (g) plus the expected distance to the end state (h), i.e., to minimize the expected total effort of the operation required 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 that we are continuously approaching the end state. Function h has an analytical definition according to Section 6.2, i.e., 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 traveled thus far, i.e., the sum of the L_1 parameter value changes for each actor over all actions executed thus 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 might increase for an action that shifts these parameters away from the end state.

We can plot the measures vs. action number for a subset of the best plans to obtain an estimate of how well the simulation manages to approach the end state, as shown in Fig. 12. In the figure, it appears that each plan step roughly moves us nearly the same distance (nearly linear development of f and g) but is not reflected as well in the development of h, i.e., 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 displayed in the noisy behavior of h, certain actions tend to take us farther from the end state, producing increases of every second step in h. For the example plan under investigation, we begin at simulation step 0 with a distance $h = 744.0$ to the end state (not shown in Fig. 12). The situation deteriorates for all 10 best simulations and turns favorable after the fifth action. The quick deterioration is due to a direct change of most of the parameters when BFOR enters Bogaland and initiates its first action. After the fifth action, the general trend presented (as well as for all 10 000 simulations) produces a slowly decreasing h. The best result reached is $h = 792.3$, as shown in the figure. Hence, the plans currently under investigation do 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 require additional development work. It should be pointed out that the relatively similar result for the 10 000 simulated plans is a direct consequence of the A*-search methodology that searches through a tree of action alternatives. Because each simulation corresponds to a path through this tree and many paths found by the search algorithm are necessarily similar, they will also have similar characteristics. This outcome is deliberate as we search for a set of good plans (for the sake of robustness), but if a greater variance in the result is sought, we can use other approaches instead that do not include any search [41].

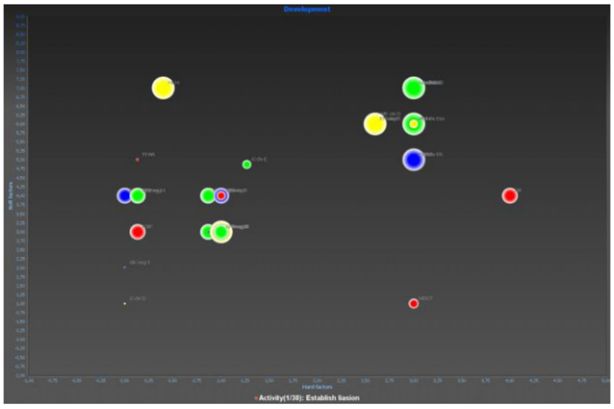


Fig. 13. Animated bubble chart visualization of the best plan (at the start of operations).

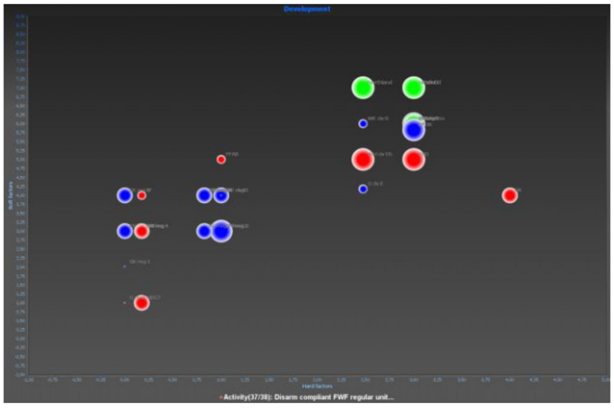


Fig. 14. Animated bubble chart visualization of the best plan (at the end of operations).

7.4. Actors' time development

The simulation engine gives us different paths for which each contains a solution to the problem, i.e., a chain of actions that must 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 note how the actors progress during the entire execution of all actions.

Different approaches exist for how to visualize multiple variables at once. In our case, we prefer to visualize the following rather heterogeneous entities:

- Actors' parameters,
- Actions,
- Actors' relationship to the blue forces,
- The importance of an actor based on economy, stability, and dominance in the region,
- Temporal changes.

Visualizing multivariate heterogeneous data is not easy. Often one must use different projection or transformation techniques or *coordinate systems* with odd shapes in which the perpendicular dimensions greater than three are transformed in different ways to increase interpretability at the cost of losing a portion of the original information. User intervention techniques can be applied, such as that used in Yi et al. [42], which presents the idea of *dust* (multi-attributed data points) and *magnets* (attractors, one for each attribute). Magnets are placed and moved on a 2D surface using the mouse on the screen and attract the data points originally gathered in the center to move towards the magnets depending on their attribute values. Hur and Yi [43] presented a tool known as *SimulSort* for sorting multivariate data simultaneously. Different attribute columns are sorted individually, and therefore, records end up split between different rows. Records are identified by highlighting the attribute values that belong to the same record by moving the cursor over one of them. Several records can be highlighted in different colors and thus compared simultaneously.

For visualization of the entities in the bullet list above, we choose to use a bubble chart combined with a subset of the animation effects demonstrated by Rosling during a lecture at TED [44]; see Figs. 13 and 14.

We group the parameters into three different groups. One group is *soft factors*, which consist of parameters related to an actor's social status, i.e., social network, feeling as a group, etc., and this parameter 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. and is plotted on the x-axis. A higher value indicates a more militarily advanced actor. The third group consists of parameters such as economy, geographical dominance, and stability. This group is represented using the size of the bubble, and a larger bubble represents a more important actor.

The bubbles also have colors that represent the relationships between the actor and the blue force BFOR. Green indicates a neutral actor, blue indicates an ally, red indicates an enemy, and yellow indicates 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 effects are animated bubbles that represent four-dimensional (X, Y, size, color) system changes that occur with respect to time (i.e., execution of action). Furthermore, the visualization software also contains different tool features, i.e., fast-forwarding and filtering, based on visible actors in the GUI. This option makes it possible to see changes (correlated or anti-correlated) that occurs simultaneously in one or several dimensions for several plotted entities.

7.5. Explaining the impact of actions

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

As we work with plans consisting of several actions A_k, we must measure 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 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,

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

where $n_{ik} = |\{f_{ikl}\,(P_i.A_k = l)\}_l|$ is the number of alternatives for $P_i.A_k$ [45].

Because the variance in this measure can be large between different plans P_i, we might choose to use box plots for a small number

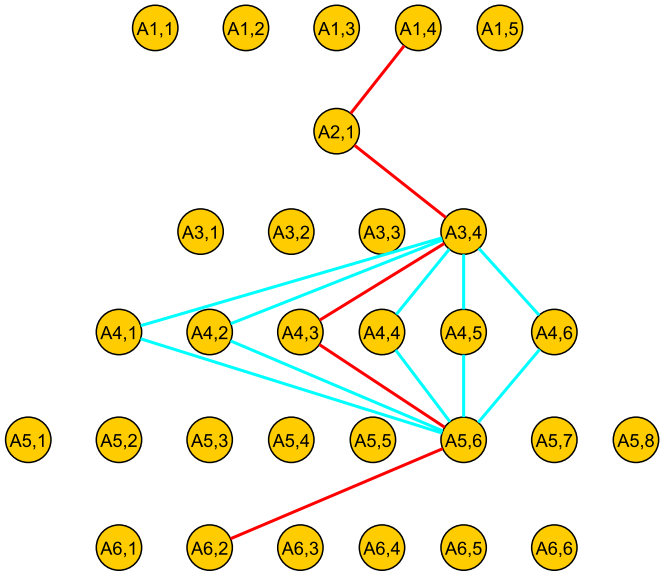


Fig. 15. Conceptually, a plan P_i is a choice of alternatives for a sequence of actions, one for each consecutive action to be executed, i.e., the red-colored path. Each cyan-colored path in this six-action planning problem corresponds to one neighboring plan with $P_i.A_4 = \{1, 2, 4, 5, 6\}$ for action 4. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of good plans for each action A_k. For example, we will examine box plots for the five best plans over all alternatives for the averages of all actions A_k,

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

To find the impact of the actions, we must perform additional simulations. The A*-search algorithm is intended to deliver the best plans that it finds relative to success in reaching the end state, as reflected in the distance f from the start state to the end state (i.e., the lower, the better). Each of these plans consists of a sequence of actions for which the actions have several alternative methods of execution, and a plan must choose one alternative from each of these actions. Certain actions in the simulation turn out to be more important than others for plan success. To determine 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 certain respects. This process can be carried out by comparing P_i with neighboring plans that only differ from P_i in the selection of alternatives for one single multi-alternative action, as shown in Fig. 15. Thus, we have

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

We simulate all neighbors for each good plan P_i already found with a variation compared with P_i of exactly one action alternative at 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. The result is the set P_{ik} consisting of $|P_i.A_k| - 1$ neighboring plans where

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

After working through all actions with alternatives and changing only one action at a time, we obtain as many neighboring plans for P_i as the total number of additional alternative actions, excluding the alternatives that are components 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

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

neighboring plans that must be compared with P_i. In our analysis, we use g and h instead of f as the refined quality measures of a plan and investigate how they are 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, as shown in Fig. 16.

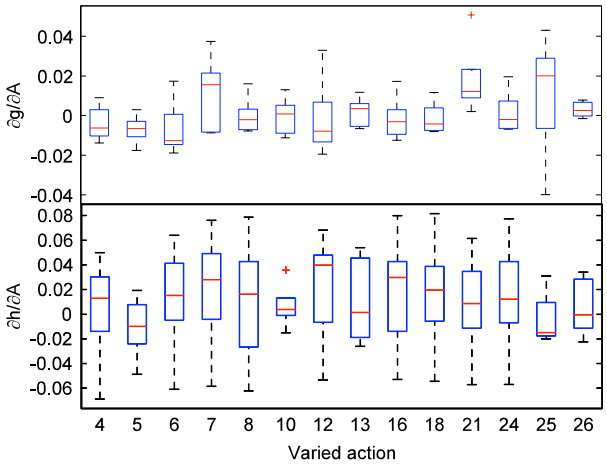


Fig. 16. Spread in sensitivity of the five best plans for which each action's sensitivity is computed for g and h, similar to (22) for f.

The sensitivity of g does not appear pronounced except for action 7, 21, and 25 based on the highest median value (read as the line in each blue box) over the five plans, where a selection of an alternative other than the one present in the main plan appears 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, based on the highest median value (red line) over the five plans.

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 [46] can be used to hierarchically find the importance of a set of input variables in a dependent continuous output variable. After simulation and traversal of the A*-search tree, the 10 000 best plans were obtained. As described above, each plan consists of a set of input actions in which certain actions have several discrete alternatives for which each plan produced from A*-search consists of 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 provide a good statistical foundation for building a regression tree on these data to find the most important actions, as shown in Fig. 17. It appears that 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.

Once a regression tree has been built, it can be used as a rough prediction tool for the dependent variable, given a new plan. A stop condition must be chosen for the further splitting into branches; eventually the tree will split into 10 000 leaves, one per simulated plan, but its predictive power will decrease the deeper the tree is traversed and will be based more on random noise from the Monte Carlo process than the major statistical tendencies that are of interest. Given a set of alternatives that a planner has chosen, one can

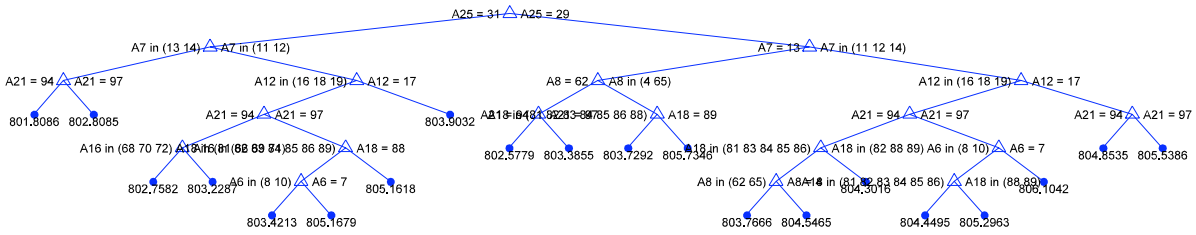


Fig. 17. A regression tree of h based on 10 000 simulations. Only the 18 most important *bifurcations* or branches on 5 levels are shown; the full tree with the default Matlab statistical toolbox setting produces nearly 1500 branching points on 22 levels (overlapping texts are due to the *view* plot-out function for regression trees in the toolbox and cannot easily be overcome).

follow the path that 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 process has been performed for the 100 best plans resulting from a second set of 10 000 simulations with a new seed, and the paths of these plans are followed in the tree to find the values of g and h that the tree proposes. A comparison with the real values of g and h for those 100 plans can be found in Fig. 18. The traceability 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 the worst (highest) f, the situation is actually reversed. Perhaps this outcome is not difficult to understand; it is easier for the regression analysis to find the systematic progress for how far we have traveled (g) than for how far it is to the goal (h) because the previous value lies inherently in the effect of the conducted actions, whereas the latter is not as easy to assess. It may be possible to compute the distance from the final actor parameter state to the end state in parameter space, but this task is not as easy.

Two figures of merit used to estimate this progress are the *re-substitution* error and the *cross-validation* error of a tree. With the re-substitution error, we use the root mean square of the g and h values predicted by the tree compared with the true values 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" [47]. In the case of g and h for the 100 best plans shown in Fig. 18, these errors are approximately 1%–2%.

In a real situation, the planning process may not begin at the top of the regression tree because the actions are ordered by execution and not by importance, but the planning procedure is often performed in a certain order. However, in this paper, actions for different plans are executed in the same order, and only the choice of alternative for each action differs between the plans. This approach places a strong constraint on the planning procedure. In addition, improvisation could be important for success, especially for handling the consequences of extreme events [48]. To increase the number of analyzed options available for improvisation, one could perform simulations in which two or more activities are swapped or selected activities not executed at all (i.e., swapping two rows or removing one row in Fig. 15). This loosening of planning constraints would expand the already huge A* search space and has therefore not been considered.

As an alternative to regression trees, Martignon et al. [49] investigated fast-and-frugal trees, which are unbalanced trees that have at least one exit node at each level, i.e., at least one branch from each node directly terminates at a leaf. Regression trees are usually not fast and frugal, e.g., the regression tree in Fig. 17 is not fast and frugal because neither branch from A_{25} terminates at a leaf. The authors claim that fast-and-frugal trees are more *psychologically plausible* for decision-makers and may be applied more rapidly in time-critical situations. In a test using the UC Irvine Machine Learning Repository, researchers found the predictive accuracy of fast-and-frugal trees to be 1%–7% worse than that of regression trees, whereas the standard deviation for one type of fast-and-frugal tree, known as *Zig*, was lower than that of the regression trees.

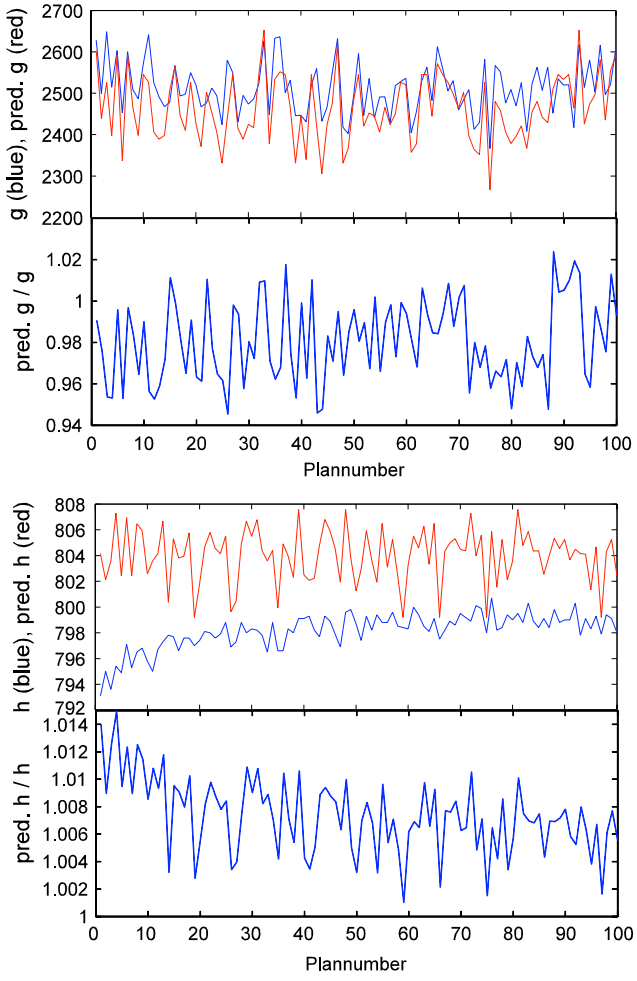


Fig. 18. Plots of real (blue) values of g and h for the best 100 plans from a simulation with a new seed as well as the predicted (red) values of g and h from the regression tree in Fig. 17 trained with the 10 000 plans from the first simulations. The ratio of predicted and real values is shown in the respective lower plot. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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 this result. First, we need to find the best way to represent the training data for use in the SVM, and this includes normalization. Second, 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 a 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 for a separating hyperplane. Selecting this hyperplane maximizes the SVM's capability to calculate the correct classification of previously 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 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 be represented as a vector of real numbers. Let a plan contain R actions that 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 $\boldsymbol{x}_i = \{x_{i1}, \ldots, x_{il}\}$. The plan vectors $\boldsymbol{x}_i$ are all normalized. Scaling them before applying the SVM is very important. This is done to avoid having 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 a linear function 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 $\boldsymbol{w} \in \boldsymbol{X}$, $b \in \boldsymbol{R}$ and where $(,)$ denotes the dot product. We want to minimize the norm $\|\boldsymbol{w}\|^2 = (\boldsymbol{w}, \boldsymbol{w})$, as shown in Fig. 19. This can be formulated as a convex optimization problem.

Minimize

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

subject to

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

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

$$\text{sign}(\boldsymbol{w}\boldsymbol{x} + b) \tag{28}$$

which defines the division of different classes and is also 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 [50]. 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.

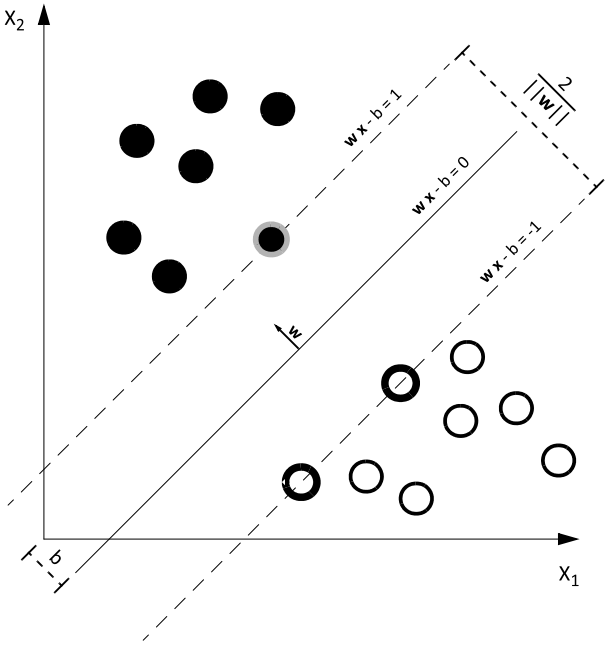


Fig. 19. Optimal linear divider of two separate classes.

Using a hyperplane, we may separate the feature vectors into two classes when there are only two target categories [51], 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 and all of the other categories are merged, and (ii) "*one against one*", where $k(k-1)/2$ models are constructed and 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 perform 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 [52] in this work. Important in LIBSVM is the choice of its parameters. Parameter optimization is performed by a full search of a pre-defined parameter set. Cross validation is used for selection of the best parameters for this training set, meaning that each combination of parameter choices is checked using cross validation, and the parameters with the best cross-validation accuracy are chosen. Using the selected parameters, the final model is trained on the entire 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.

Because 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.

Table 3
Pseudo code for the investigation.

1. Select the plan to be investigated and put it in a separate class. Update the input label vector.
2. For the length of the plan $=$ 2 to 46:
   2.1 Select the optimal parameters for training.
   2.2 Train the model.
   2.3 Calculate the distances using (29) and (30).
3. Plot the distances.

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);

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

The model is trained for 12 classes: 11 classes from pre-calculated Potts spin clustering and one class containing the plan under execution. Training is performed 45 times for each investigation, each training with successively longer plans, from a plan length of two actions to training on the full matrix with 43 actions (and f, g, and h); see Table 3.

Note that this process is normally very sensitive to noise and outliers in the training data. SVMs generally have problems with unbalanced problems where one of the classes has many more training examples than the other. For a balanced training set, the outliers from class A 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 example in one class, and thus the algorithm does not have 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 are carefully clustered before training. The probability of noisy data and outliers is 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 of the other plans, and the result is plotted in Figs. 20–22. 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 vary with the length of the plan. First, the best plan is chosen from all of the other plans; the best plan is the one with the lowest value of h. The distance from the best plan to the nearest hyperplane of all other classes using successive longer plans is shown in Fig. 20. This figure shows ten curves, one for each class combined to class 12 representing the single tested plan.

In Fig. 21, we show another view of the same result by taking the minimum distance of all eleven classes in Fig. 20 at each length of the plan.

The eleven classes are designed unsupervised with respect to plans, structure and g value in the preceding clustering stage. Each class is determined by its content. Because 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).

Because most of the plans are "good" we take a look at the ten best plans with regard to h. In Fig. 22, minimum distances, created in the same way as in Fig. 21, are plotted for the ten best plans with regard to 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.

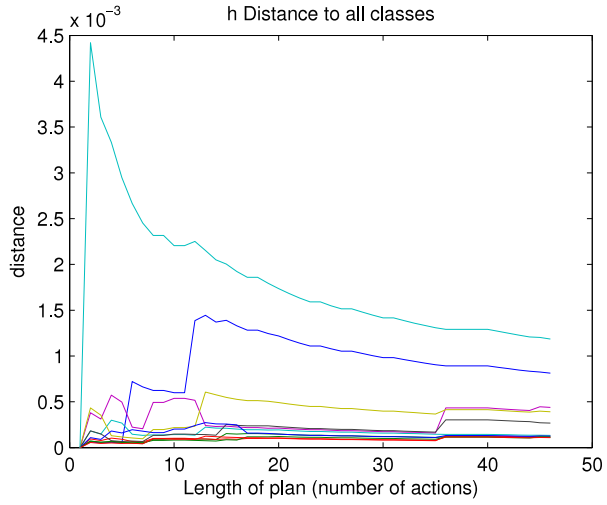


Fig. 20. Distance d (30) of the best plan during execution towards the eleven hyperplanes.

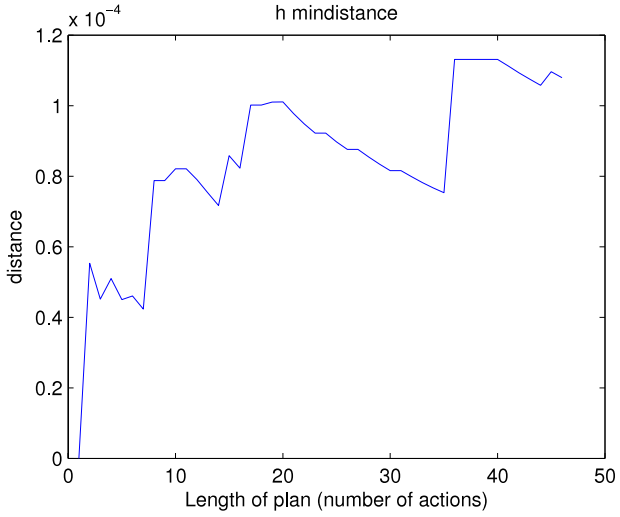


Fig. 21. Minimum distance d of the best plan during execution to the closest hyperplane.

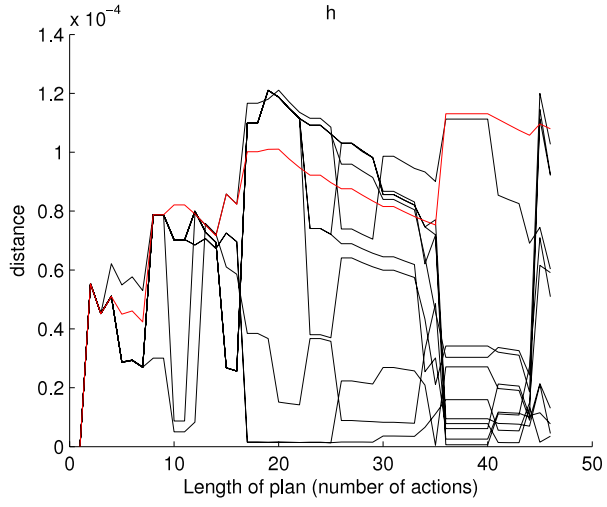


Fig. 22. Minimum distance d of the ten best plans (the best plan is in red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Some actions in the Bogaland scenario.

Establish liaison	Protect Oil/gas/key infrastructure
Establish liaison	Deploy in Visby area in (some) force and negotiate
Establish NFZ/MEZ	D-day enforce cease fire by deploying in Visby area
D-day stabilize situation by deploying in between	Deploy in Visby
Deploy in secure parts	Kill/capture non-compliant
E. Kasuria 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
Neutralize MDCT by negotiating	Disarm compliant FWF regular units on Gotland
Liaise with Kasuria and neutralize MDCT	Liaise, neutralize other irregulars
Neutralize MDCT	Neutralize all irregulars
Liaise and kill/capture MDCT	Liaise, neutralize and disarm all irregulars
Kill/capture MDCT	Neutralize and disarm all irregulars
Survey oil/gas/key infrastructure with BFOR mil. units (no protection)	Stabilize situation by deploying in between West Kasuria and West Mida
Protect oil/gas/key infrastructure with BFOR mil. units	Establish liaison
Survey oil/gas/key infrastructure	Disarm FWF regular units in W. Kasuria and W. Mida kill/capture non-compliant

The graphs show zero distance for some lengths of plans. This is natural because 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. 21, we can 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. 20, the analyst observes 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 deployed. The shaping phase includes actions that require engagement with opposing forces, such as neutralizing irregular organizations' powerbases, enforcing embargoes 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 Non-Government Organizations (NGOs), and identifying and isolating harmful actors from the Bogaland population.

These three operational phases are carried out through 43 different BFOR actions. Each action has between 1 and 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., 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 4 lists the first thirty actions modeled in our scenario.

Altogether, 40 actors were modeled in this scenario. These actors are listed in Table 5. The colors to the left of each actor indicate the initial roles of the actors in the scenario. The color blue represents BFOR and its allies, whereas the color red shows the enemies. Green stands for neutral actors, and yellow actors are those actors whose position or relation to us is not clear or yet to be determined.

Table 5
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

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 1. This sums to 600(=40 × 15) variables and $1.722 \times 10^{361}(=4^{600})$ possible scenario states. For the purpose of our experiment, we initialized these variables using data from SMEs. This also includes actors' relations to each other. We define 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 noted, 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 of 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.

When executing the actions, the simulation performs graph traversal of 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 a 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.

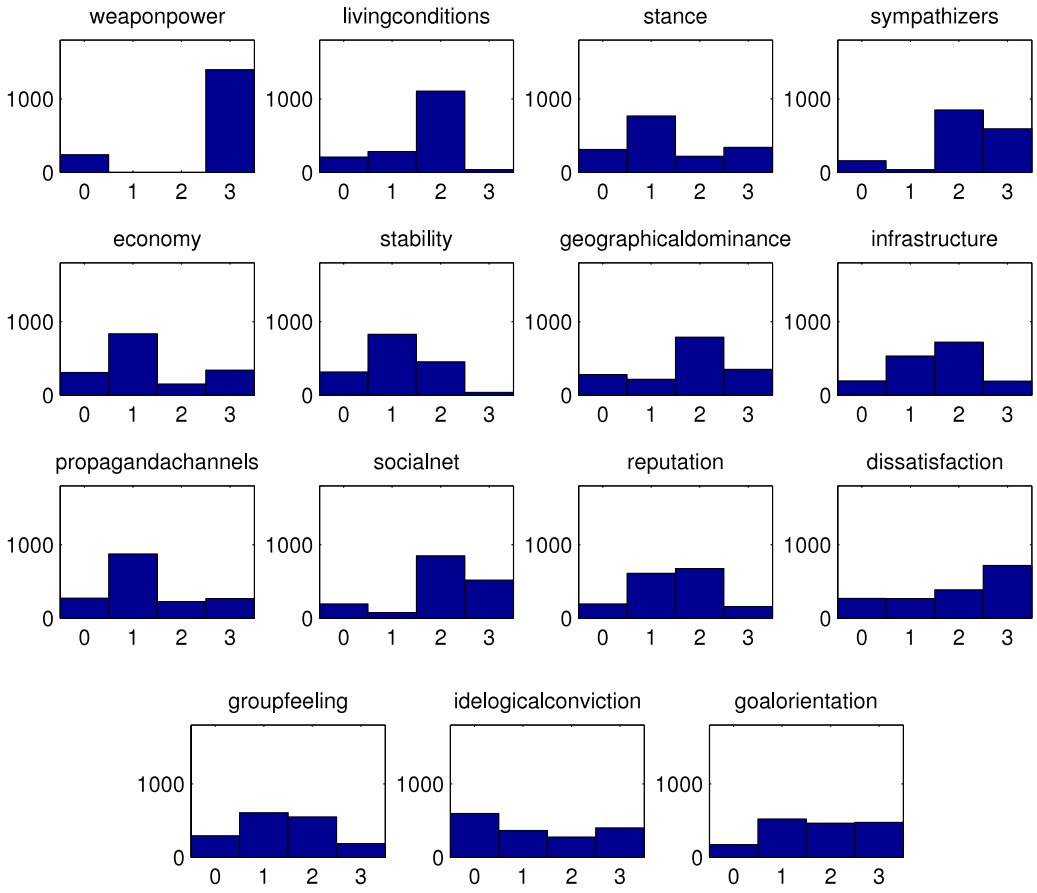


Fig. 23. Histograms of the distributions of the 15 different parameters (the same as the internal state variables in Table 1). Each histogram bar is summed over all actors and actions for the best plan.

The experiment was run on one 3.1 GHz Intel Xeon E5-2687W CPU with 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 instance was 24 s. Each experiment run took 2.8 days to execute without any parallel computing. A description of the simulation system is available in [53].

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 found different patterns, which have been visualized. Fig. 23 shows the parameter distribution for the best plan. Note that the parameters are distributed in different ways because the scenario affected different parts of an actor during the entire execution. For an explanation of each value (0–4), see Table 1. For example, in the best plan, we observe several skewed distributions, e.g., *weaponpower* and *livingconditions*. We notice that for most actors, the *livingconditions* are *Not suffering, limited resources* (value 2, Table 1), while a few actors are worse off.

The plans that were generated were of different sizes because some of the 43 actions may not be performed in some plans. On average, plans executed 39 actions; see Fig. 24. Note that a shorter plan may give a lesser g-value, but this does not guarantee that the h-value will also be small.

There are several alternative methods to visualize the dispersion of measurement data. Dispersion of data can be due to measurement errors or underlying stochastic behavior, such as the Monte Carlo process in our simulation. Edwards et al. [54] discuss different ways to visualize dispersions of measurement data; the uncertainty dispersion implies how people tend to interpret the uncertainties depending on how they are visually represented, e.g., as box plots, error bars, scatterplots or various probability density distributions. In general, box plots are easy to understand and seem to give a better match between real dispersion and the interpretation the viewer gets. In Figs. 25–27, we see the f, g and h-values for the actual plan sizes. For the f-value, we observe a constant f, as g and h values balance each other. For the g-value, we see that, the more actions a plan have, the higher the value becomes, which is natural because more work has been done in the operation. What can be noted in Fig. 27 is that the h-value seems to drop as the plan

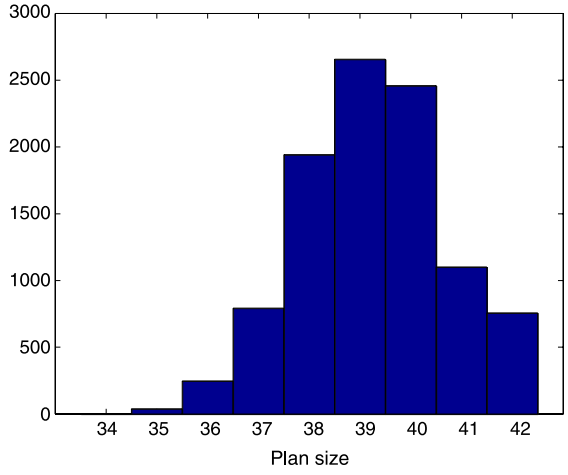


Fig. 24. The distribution of plan sizes for the 10 000 plans. The plan sizes differ from plan to plan because a varying number of actions are executed.

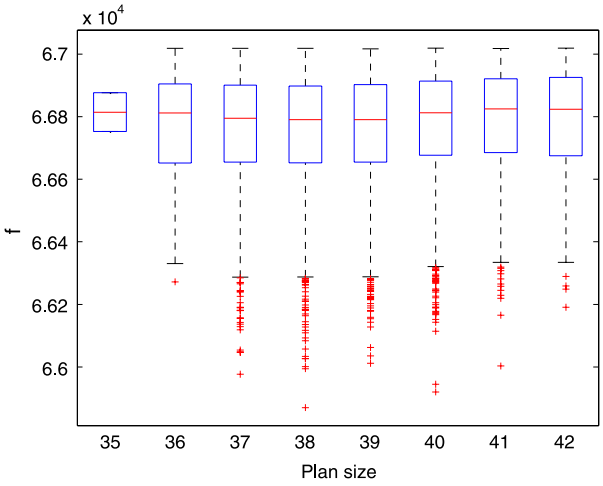


Fig. 25. Box plots over the average f-value for different plan sizes.

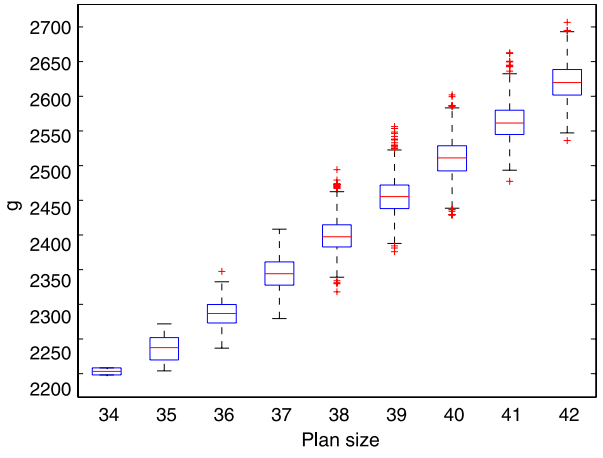


Fig. 26. Box plots over the average g-value for different plan sizes.

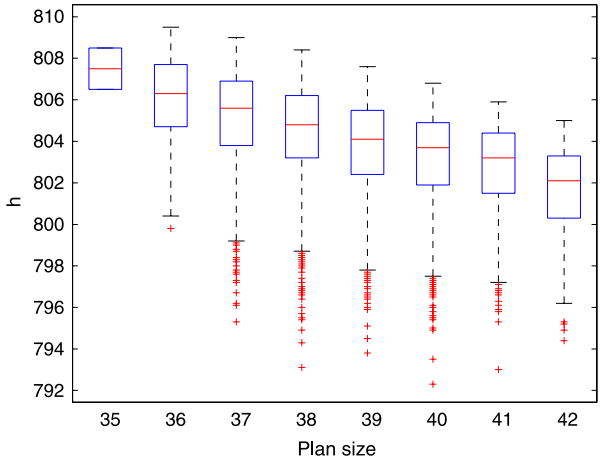


Fig. 27. Box plots over the average h-value for different plan sizes.

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, shown in Fig. 27.

After simulating 1000 plans, we plotted the f-values in ascending order; see Fig. 28. Here we can see that, in practice, the best plans (lowest f-values) were obtained after a few hundred simulations.

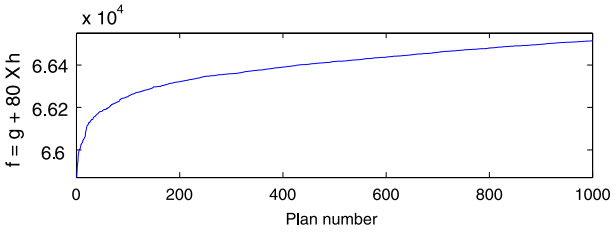


Fig. 28. The different f-values for 1000 sorted plans.

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 Figs. 14 and 15 demonstrates the progress made. However, from the time series of h-values in Fig. 13, 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 actions and many more of them 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 the requirement 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 and may be altered. If the plans evaluated are more successful, the weight will be lowered. Because $g \gg h$ and the variance of g is much greater than the variance of h, it is obvious that the weight of h must be much higher than that of g (because we are never able to reach $h = 0$ in the A*-search). The value "80" was found by extensive experimentation such that the impact from h on the minimization of f was slightly greater than the impact from g.

While it is obvious that we may achieve that which we optimize for, we are still surprised by the small variations in the minor effects monitored (but not strived for) in Fig. 12 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 Section 7.5 with the regression tree analysis of Section 7.6. We observe that the action A_{25}, with the most negative impact in Fig. 17, is the first action to be split by the regression tree in Fig. 18 and that 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. Alternately, the regression tree also 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 approximately 1%–2%. This is achieved for each plan in milliseconds compared to 24 s 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 timespan during re-planning of a plan under execution.

Finding the border of an operation is analyzed in Section 7.7. In Fig. 21, 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 a time series of the distance from the best plan to all neighboring groups 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 the commander knowledge of which of the eleven borders should be monitored carefully during plan execution.

We believe that the analyses of borders, together with the box plots 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 of 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 own 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 Section 5 reduces the size of the problem by introducing an aggregated generic model. This is a step in the right direction towards providing design support to SMEs. However, the development of the plan with all its alternatives was performed 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 plans and scenarios that prevent them from making logical errors in operational planning.

10. Conclusion

In this article, 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 about 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 feedback during plan development and gives commanders information on where to focus their attention during plan execution.

Acknowledgment

We wish to express our thanks to Frida Hinshaw for her technical contribution to data analysis.

Appendix

- **Action** is an activity well-bounded in space and time that is intended to result in some form of advantageous change of local or regional state. Example: "Arrest key DSD leaders", "Take control over suburb Hallunda".
- **Action alternatives** are different ways to conduct an *action* in space and time that may use different resources. There is one alternative that needs to be chosen for each action of the *plan*. This preferred alternative (chosen and evaluated by the simulator) is the alternative that implies a change of state that takes largest step towards the *end-state*. All alternatives are intended to obtain the same result, but can be more or less costly or prone to failure depending on the overall situation. Example: Alternative tactical ways to "Arrest key DSD leaders" such as attacking their headquarters in order to make them to give up, or alternatively, to show strength of force and convince them to give up after negotiation.
- **Action sequence** is a set of several *action alternatives* chosen and evaluated by the simulator, one per *action* that are conducted in a sequence. Some of them might have alternatives, and the sequence consists of one alternative from each *action*. Example: "Secure sea port of own troop debarkation", then "Debark own forces in the operational area", and finally "Establish camps and headquarters".
- **Actors** are entities that execute *actions* in an operation. In a *simulation*, actors are often more or less autonomous software agents that interact. Each actor has its own set of resources, an action repertoire, an internal state and relationship with other actors. Actors can be groups of people, who somehow have a common identity and purpose.
- **CIM** Cross Impact Matrix is a matrix where alternatives for different *actions* are evaluated as to their compatibility. With cross impact analysis it is possible to detect some internal inconsistencies within a *plan* before *simulation*.
- **Effect** is somewhat abstract; the joint consequences of a set of state-changes due to executed *actions* are expected to lead to one or several pre-defined and measurable effects such as "Establish order and stability in East Kasuria". It should be noted that one or several *actions* can contribute to one or several different effects. Mathematically an effect is a set of possible states.
- **End-state** is the ultimate goal of the whole operation we are conducting and the ultimate goal that we try to reach by executing a *plan* with an appropriate *sequence of actions*. The end-state can be seen as an intersection of *effects* in the operational area (often nation, or at least region-wide). Mathematically the end-state is a single point in parameter space.
- **Evaluated action** is an *action* which is *simulated* and where the resulting change of state is evaluated to see how much it moves the current state towards the *end-state*.
- **Event** is an occurrence that causes a change of state in a simulation.
- **Execution** of an *action* or a *plan* is the process of trying to obtain the objectives of the *action* (obtaining the action's state-change) or the *plan* (obtaining the *end-state*).
- **Plan** is a combination of all *actions* that have to be *executed* in order to come closer to or, ideally, reach the *end-state*. Some of the *actions* have two or more *action alternatives* out of which one and only one has to be chosen for each *action*. Example: A combination of *action alternatives* such as securing a sea port for own troop debarkation, debarking own forces in the operational area, establish camps and headquarters, establish interdicted corridors for enemy troops and vessels, establish order in liberated areas, etc.
- **Planning** is the process of selecting among available *action alternatives* to find a suitable *plan* in order to achieve a desirable *end-state*.
- **Re-planning** is the repeated process of setting up a new plan for the rest of the operation, normally after some actions have already been *executed*. Usually, a plan does not hold perfectly from start to beginning due to unforeseen *events* or conditions, so a re-planning has to be done to get back "on track" towards the *end-state*. Re-planning is just like planning but limited to future actions when some subset of the action sequence has already been performed.
- **Simulation** is the method for implementing a model over time, where a model is a physical, mathematical or otherwise logical representation of a system, entity, phenomenon, or process.
- **Situation** is the total state of all actors and objects in a system, including ongoing actions.

References

[1] Chandrasekaran B. From optimal to robust COAs: challenges in providing integrated decision support for simulation-based COA planning. Ohio State University; 2005. http://web.cse.ohio-state.edu/~chandra/From-Optimal-to-Robust-COAs-white-paper.pdf [Online].

[2] Chandrasekaran B, Goldman M. Exploring robustness of plans for simulation-based course of action planning: a framework and an example. In: Proceedings of the IEEE symposium on computational intelligence in multicriteria decision making; 2007. p. 185–92.

[3] Bankes S. Exploratory modeling for policy analysis. Oper. Res. 1993;41(3):435–49.

[4] Smith EA. Complexity, networking, and effects-based approaches to operations. Washington, DC: Department of Defense CCRP; 2006.

[5] Hunerwadel JP. The effects-based approach to operations: questions and answers. Air Space Power J. 2006;20:53–62.

[6] Pfaff MS, Klein GL, Drury JL, Moon SP, Liu Y, Entezari SO. Supporting complex decision making through option awareness. J. Cogn. Eng. Decis. Mak. 2013;7(2):155–78.

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

[8] Farrell PSE. 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. p. 1–17.

[9] Allied command operations comprehensive operations planning directive (COPD-Trial version). Brussels: supreme headquarters allied power Europe, NATO; 25 February 2010.

[10] The comprehensive approach. Joint discussion note 4/05. Swindon: Ministry of Defence; 2006.

[11] Dijk G. Comprehensive approach: why it is a big NATO issue. In: Proceedings of the CIOR symposium on "NATO's comprehensive approach and the role of reservists"; 2010.

[12] Schubert J, Wallén M, Walter J. Morphological refinement of effect-based planning. In: Norsell M, editor. Stockholm contributions to military-technology 2007. Stockholm: Swedish National Defence College; 2008. p. 207–20.

[13] Schubert J. Multi-level subjective effects-based assessment. In: Proceedings of the 13th international conference on information fusion, paper We3.4.1; 2010. p. 1–8.

[14] Gordon TJ, Hayward H. Initial experiments with the cross impact matrix method of forecasting. Futures 1968;1(2):100–16.

[15] Glenn JC, Gordon TJ. Futures research methodology—version 2.0. Washington, DC: American Council for the United Nations University; 2003.

[16] Turoff M. An alternative approach to cross impact analysis. Technol. Forecast. Soc. Change 1972;3(3):309–39.

[17] Dalkey NC. An elementary cross impact model. Technol. Forecast. Soc. Change 1972;3(3):341–51.

[18] Duperrin JC, Godet M. SMIC 74—a method for constructing and ranking scenarios. Futures 1975;7(4):302–12.

[19] Bañuls VA, Turoff M. Scenario construction via Delphi and cross-impact analysis. Technol. Forecast. Soc. Change 2011;78(9):1579–602.

[20] Bañuls VA, Turoff M, Hiltz SR. Collaborative scenario modeling in emergency management through cross-impact. Technol. Forecast. Soc. Change 2013;80(9):1756–74.

[21] Jeong GH, Kim SH. A qualitative cross-impact approach to find the key technology. Technol. Forecast. Soc. Change 1997;55(3):203–14.

[22] Asan U, Bozdag CE, Polat S. A fuzzy approach to qualitative cross-impact analysis. Omega 2004;32(6):443–58.

[23] Parashar A, Paliwal R, Rambabu P. Utility of fuzzy cross-impact simulation in environmental assessment. Environ. Impact Assess. Rev. 1997;17(6):427–47.

[24] Enzer S. Delphi and cross-impact techniques: an effective combination for systematic futures analysis. Futures 1971;3(1):48–61.

[25] Helmer O. Analysis of the future: the Delphi method. P-3558. Santa Monica, CA: The RAND Corporation; 1967.

[26] Helmer O. Reassessment of cross-impact analysis. Futures 1981;13(5):389–400.

[27] Amara RC. A note on cross-impact analysis: a calculus for sequence-dependent events. Futures 1972;4(3):267–71.

[28] Bloom MF. Time-dependent event cross-impact analysis: results from a new model. Technol. Forecast. Soc. Change 1977;10(2):181–201.

[29] Zwicky F. Discovery, invention, research through the morphological approach. New York: Macmillan; 1969.

[30] Schubert J. Clustering belief functions based on attracting and conflicting metalevel evidence using Potts spin mean field theory. Inf. Fusion 2004;5(4):309–18.

[31] Schubert J, Moradi F, Asadi H, Hörling P, Sjöberg E. Simulation-based decision support for effects-based planning. In: Proceedings of the 2010 IEEE international conference on systems, man and cybernetics; 2010. p. 636–45.

[32] Duda RO, Hart PE, Stork DG. Pattern classification. second ed. Hoboken, NJ: Wiley-Interscience; 2000.

[33] Huang C, Darwiche A. Inference in belief networks: a procedural guide. Internat. J. Approx. Reason. 1996;15(3):225–36.

[34] Cozman FG. Generalizing variable elimination in Bayesian networks. In: Workshop on probabilistic reasoning in artificial intelligence; 2000. p. 27–32.

[35] Hamming RW. Error detecting and error correcting codes. Bell Syst. Tech. J. 1950;29(2):147–60.

[36] Wu FY. The Potts model. Rev. Modern Phys. 1982;54(1):235–68.

[37] Peterson C, Söderberg B. A new method for mapping optimization problems onto neural networks. Int. J. Neural Syst. 1989;1(1):3–22.

[38] Bengtsson M, Schubert J. Dempster–Shafer clustering using Potts spin mean field theory. Soft Comput. 2001;5(3):215–28.

[39] Ahlberg S, Hörling P, Johansson K, Jöred K, Kjellström H, Mårtenson C, et al. An information fusion demonstrator for tactical intelligence processing in network-based defense. Inf. Fusion 2007;8(1):84–107.

[40] Klein GL, Drury JL, Pfaff MS. Providing an option awareness basis for naturalistic decision making. Cogn. Technol. 2011;16(2):10–9.

[41] Schubert J, Hörling P. A comparative study between using A*-search, Latin hypercube and genetic algorithms in design of experiment for simulation of military operational plans. In: Proceedings of the 2014 IEEE international conference on systems, man and cybernetics; 2014. p. 372–79.

[42] Yi JS, Melton R, Stasko J, Jacko JA. Dust & magnet: multivariate information visualization using a magnet metaphor. Inf. Vis. 2005;4(4):239–56.

[43] Hur I, Yi JS. SimulSort: multivariate data exploration through an enhanced sorting technique. In: Jacko JA, editor. Human–computer interaction. Novel interaction methods and techniques. LNCS, vol. 5611. Berlin: Springer; 2009. p. 684–93.

[44] Rosling H. Stats that reshape your worldview. Technology, entertainment, design; 2006. http://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen.html [Online].

[45] Schubert J, Hörling P. Explaining the impact of actions. In: Proceedings of the 15th international conference on information fusion; 2012. p. 354–60.

[46] Breiman L, Friedman J, Stone CJ, Olshen RA. Classification and regression trees. Boca Raton, FL: Chapman and Hall; 1984.

[47] Statistics toolbox. Natick, MA: The MathWorks Inc.; 2012.

[48] Mendonça D. Decision support for improvisation in response to extreme events: learning from the response to the 2001 World Trade Center attack. Decis. Support Syst. 2007;43(3):952–67.

[49] Martignon L, Katsikopoulos KV, Woike JK. Categorization with limited resources: a family of simple heuristics. J. Math. Psych. 2008;52(6):352–61.

[50] Cortes C, Vapnik V. Support-vector networks. Mach. Learn. 1995;20(3):273–97.

[51] Schubert J, 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. p. 1325–1332.

[52] Chang C-C, Lin C-J. LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. 2011;2(3):27.

[53] Asadi H, Schubert J. A Stochastic discrete event simulator for effects-based planning. In: Proceedings of the 2013 winter simulation conference; 2013. p. 2842–53.

[54] Edwards JA, Snyder FJ, Allen PM, Makinson KA, Hamby DM. Decision making for risk management: a comparison of graphical methods for presenting quantitative uncertainty. Risk Anal. 2012;32(12):2055–70.