2014 IEEE International Conference on Systems, Man, and Cybernetics
October 5-8, 2014, San Diego, CA, USA

A Comparative Study between Using A*-search, Latin Hypercube and Genetic Algorithms in Design of Experiment for Simulation of Military Operational Plans*

Johan Schubert, Pontus Hörling
Department of Decision Support Systems
Division of Information and Aeronautical Systems
Swedish Defence Research Agency
SE-164 90 Stockholm, Sweden
schubert@foi.se
http://www.foi.se/fusion

***Abstract*—In this paper we compare three alternative ways of designing a data farming experiment for a military operational planning problem. In this multi-objective optimization problem the goal is to minimize two different measures of effectiveness. Military plans under consideration are evaluated by an event based simulation system. We compare three different approaches for selecting which plans should be simulated; A*-search, genetic algorithms and Latin Hypercube. The more robust Latin Hypercube approach is found to be the better approach for this application compared to the other two more focused approaches.**

Keywords—Data farming; design of experiment; computer simulation; data analysis; decision support systems; operational planning; effects-based approach to operations; A*; nearly orthogonal nearly balanced hypercube; genetic algorithms.

I. INTRODUCTION

How can we support decision makers when facing a new problem domain where little a priori domain knowledge is available? For each new problem we face, we probably know about the factors of influence, but we usually have little knowledge of the internal system dynamics. We might not be able to say anything about the way in which the various factors have impact on an outcome, or which internal relationships that exist between the factors, i.e., how they interact and what effect this has on the outcome. By gaining an understanding of what impact the effects have on the outcome, we can provide better and more accurate predictions than we could otherwise achieve.

A process developed to handle this kind of problem is data farming – a process that can be said to be a combination of already existing processes and technologies that make up a tool suite to maximize the information available; see for example [1]. Data farming manipulates simulation models to their advantage through detailed experiment design since simulation of every value and combination of values is not possible to compute in spite of the availability of high performance computing. The choice of experiment design limits the information that can be extracted from the model which stresses the importance of filling up the parameter space as efficiently as possible. The focus is on trying to produce a sufficiently complete landscape of potential outcomes, and identify areas of special significance, rather than identifying an individual response. In addition to identifying significant effects and relationships between the factors, importance is also placed on detecting possible anomalies and includes them in the decision.

In this paper we compare three alternative ways of designing a data farming experiment for a military operational planning problem for an expeditionary operation [2, 3]. In this planning problem an operational plan is described as a sequence of actions. For each action we may have several different alternative ways to perform the action. Together they make up all possible plans, which can be represented as a tree of action alternatives that may be searched for the most effective action alternative sequences. Alternatively, plans may be selected by design of experiment where each plan corresponds to a design point, i.e., input data to the simulator, or learnt by evolutionary methods. For all three approaches the task is to find plans that are both effective and efficient. These plans need to be effective in reaching the sought after end state with minimal effort and efficient in minimizing any possible remaining distance to the end state after the last action of the plan has been performed. Within this multi-objective optimization problem our goal is to minimize both a distance g from the initial state of the system under consideration to the final state reached of this system, and a distance h from the final state after the last action of the plan has been performed to the desired end state of the system. The system in this application is a description with 40 actors each with 15 parameters describing the state of the actor; in total 600 parameters describe the entire state. All alternative plans under consideration by any of the three alternative approaches are evaluated by simulation in a discrete event simulator [4] based on their g and h values.

*This work was supported by the FOI research project "Data Farming", which is funded by the R&D programme of the Swedish Armed Forces.

978-1-4799-3840-7/14/$31.00 ©2014 IEEE

Within the design of experiment approach we derive a *Latin hypercube* (LH) [5] design called a *Nearly Orthogonal Nearly Balanced* (NONB) mixed design [6] by genetic evolution of 1000 design points corresponding to 1000 alternative military operational plans using a genetic algorithm that optimize the orthogonality ρ of the NONB design. No attempt is in this approach given to optimization of the *g* and *h* output parameters. These design points will be used for simulation of the corresponding military plans that will be evaluated based on their *g* and *h* values.

A LH is an experimental design that focuses on complex models with many factors. In exploratory data analysis we want designs that can suit a variety of simulation models. For such situations, LH proves to be an invaluable technique. This is the dominant design for experiments with computer simulations. An important reason for this is that they come with minimal restrictions on the number of factors.

We investigate the performance of the NONB created by the genetic evaluation and compare it with two other approaches in two previous studies; (*i*) where design points are derived by A*-search within a decision tree of plan action alternatives [2, 3], where a full sequence of action alternatives plan items make up one design point for the simulation system [4], and (*ii*) by genetic algorithm evolution of full plan sequences [7] that optimize the performance (*g* and *h*) of the plan itself.

The purpose of A*-search is to search for a sequence of actions alternatives that best suits the decision maker's desired end state. Such a search method can neither be designed according to the principle of "breadth first search" nor "depth first search". In the former case it will take too much time before we reach a reasonably correct prediction. In the latter case we get stuck with just one suboptimal plan. A more suitable approach is to apply an A*-search algorithm, represented by $f = g + h$, where *g* is the total distance from the starting position to the current location, and *h* is a underestimation of the remaining distance from the current position to the goal state, and *f* is minimized by the method. A heuristic function is used to create this estimate on how far it is to the goal state. This is the current estimated shortest path *f*. The true shortest path is not discovered until the A*-algorithm is finished.

In an earlier publication [7] we represented the military planning problem as a bi-objective optimization problem and solved it using a Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [8] and evaluated alternative plans by the same simulator. We will in addition to the main comparison between NONB and A* also shortly compare the results of NONB with the results obtained using GA.

In Sec. II we describe the planning problem at hand. Focus is here on finding the best possible plan by selecting alternatives for each action within the plan to maximize the degree of success. We continue by introducing the Bogaland scenario that we use as a test case (Sec. III). In Sec. IV we develop a data farming approach where design points are assigned by a NONB LH [6, 9] derived by genetic algorithms in off-line pre-processing. In Sec. V we describe the data farming approach where simulation use design points that are selected by A*-search in a tree of sequentially ordered actions [2, 3]. In Sec. VI we compare the two approaches as to their ability to find good plan instances, and we study the differences in ability to explain the results through extensive data analysis. These two methods are here also compared to a third approach where genetic algorithms are used [7]. Finally, conclusions are drawn (Sec. VII).

II. The Planning Problem

How we model a phenomenon depends on the purpose of the model and the questions we want to answer. Since our simulation system aims to support decision-making within an effects-based approach to operations (EBAO) [10, 11] the modeling has to be based on EBAO and the concepts used within it, such as plan, action, effect, end state, etc. EBAO is a military approach to the management and implementation of efforts at the operational level.

To provide decision support for this planning work, we develop methods that can be used iteratively when successively modeling different elements of the plan and testing them by simulation and evaluation against a scenario with operators' models that reacts to the execution of plan elements. It is possible to measure the change in state of all the actors in relation to the desired end state.

A plan as it is defined in the context of EBAO is a sequence of actions that together leads to a desired end state which is set by a military force.

A typical plan instance P_1 is

[1 2 41 61 43 108 20 7 12 4 64 50 51 52 16 63 53 55 56 66 67 81 91 79 80 97 95 96 102 107 29 30 57 46 47 66975.9 2255.9 809]

where all but the last three numbers in this sequence are the numbers of the selected alternative for each action in this plan instance. For example, action number 3 (i.e., position 3 in the sequence) takes alternative number 41. Note that alternatives for different actions are numbered with running numbers in no particular order; they do not restart at 1 for each new action. A full plan instance can be represented as a path from the root of a tree down to one particular leaf. Obviously, the depth of the tree is the length of the sequence minus three (i.e., not counting the *f*, *g* and *h* estimates). Plan instance P_1 above corresponds to a sequence of 35 specific simulations where the actions take the numbered alternative listed in the sequence as its input parameter.

The last three parameters are different evaluation measures called *f*, *g* and *h* ($f = g + 80h$). They are distance measures calculated from changes in the scenario state and used in the A*-search algorithm. The "80" in the function *f* was found by experimentation. This factor gives the best balance between minimizing *g* and *h*.

Plan simulation is performed by the simulation engine. The engine basically contains an implementation of sequential evaluation of action alternatives which uses the Monte Carlo principle for event based simulation.

In addition to the simulation approach taken in this paper we have also analyzed operational level plans by

morphological [12, 13] and statistical methods [14, 15]. With morphological methods it is possible to quickly find inconsistences within plans, and with statistical methods we can find the higher level effects of success or failure based on evaluation of all action alternatives of the plan.

III. The Scenario

We make use of the same scenario that has regularly been used by the Swedish Armed Forces in the Combined Joint Staff and "Viking" Exercises. The scenario comprises several fictitious countries, two of which, Xland and Bogaland, have been described in-depth. Background histories offer explanations to why and how sentiments, stances, identities, loyalties, economic dependencies and inequalities have evolved over time, occasionally resulting in shifts of power. Phenomena that are commonly found in conflict areas and post conflict areas have been embedded in scenario contexts that make the origins of the phenomena plausible, Fig. 1.

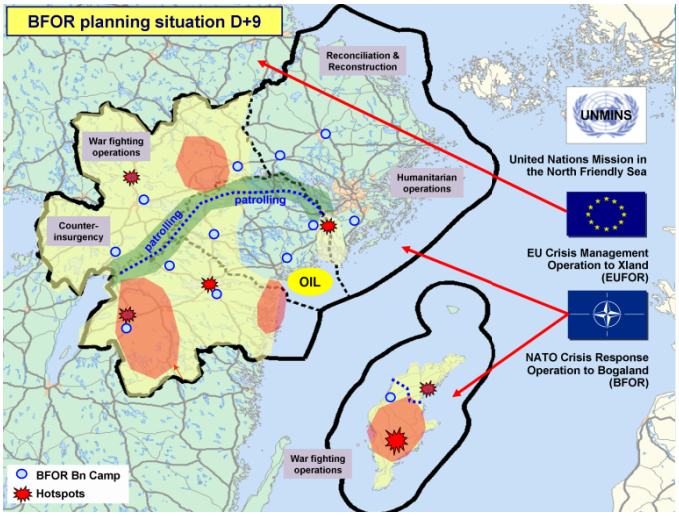

Fig. 1. The Bogaland test scenario.

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

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

The incumbent presidents have signed a peace-agreement, and an international force, BFOR, is present to support the implementation of the agreement. Irregular groups in Bogaland seek to preserve or increase their influence by undermining the efforts of BFOR, the governments or competing irregulars. Two of the neighboring countries have much at stake in the conflict, because of economic interests and shared identities with parties within Bogaland. Actors within these neighboring countries support irregulars in Bogaland.

For details on scenario modelling and calculation of *f*, *g* and *h* parameters we refer to [2, 3].

IV. The Latin Hypercube Approach

In this section we investigate and compare data farming methodologies using LH design [5] for the same planning problem as was solved using simulation and A*-search [1, 3]. The strength of using Latin hypercube design is that the design works especially well when a priori knowledge regarding the factors response is low. Some other strength is its effectiveness, space filling properties, and design and analysis flexibility.

If the simulation model contains discrete [6] and categorical factors [16] which may have different number of levels per factor we use the NONB design.

When designing a NONB we measure its correlation between every pair of input vectors. Let $\mathbf{M}$ be a $n \times j$ matrix where n is the number of rows (i.e., number of design points) and j is the number of columns (i.e., number of parameters). $\mathbf{M}$ has elements a_{rc} where $1 \le r \le n$ and $1 \le c \le j$. The pairwise correlation ρ_{xy} between any two columns x and y in $\mathbf{M}$ is

$$\rho_{xy} = \frac{1}{(n-1)S_x S_y} \sum_{r=1}^{n} (a_{rx} - \mu_x)(a_{ry} - \mu_y) \quad (1)$$

where

$$\mu_x = \frac{1}{n} \sum_{r=1}^{n} a_{rx}, \quad (2)$$

$$\mu_y = \frac{1}{n} \sum_{r=1}^{n} a_{ry} \quad (3)$$

and

$$S_x = \sqrt{\frac{1}{n-1} \sum_{r=1}^{n} (a_{rx} - \mu_x)^2}, \quad (4)$$

$$S_y = \sqrt{\frac{1}{n-1} \sum_{r=1}^{n} (a_{ry} - \mu_y)^2}. \quad (5)$$

Our military planning case has several input parameters with few parameter values, making it difficult to design a very large NONB. For comparison with the A*-approach we want to simulate 1000 plans. To achieve this we design a stacked

NONB design with 100 NONBs, each with 10 columns. Together they make up a complete set of design points for the simulator.

We measure the orthogonality of the NONB design by the average pairwise correlation between any two columns over all 100 NONBs. We get,

$$\bar{\rho} = \frac{1}{100}\sum_{i=1}^{100}\rho^i = \frac{1}{4500}\sum_{i=1}^{100}\sum_{x=1}^{9}\sum_{y=x+1}^{10}\rho_{xy}^i \qquad (6)$$

where i is the index of the i^{th} NONB, and x and y are column indices in the NONBs. We also measure the average maximum pairwise correlation over all 100 NONBs;

$$\overline{\rho_{max}} = \frac{1}{100}\sum_{i=1}^{100}\max_{xy}\rho_{xy}^i. \qquad (7)$$

The 100 NONBs of the stacked design is each evolved by a genetic algorithm that minimizes the average correlation ρ^i between all columns for the i^{th} NONB.

The genetic algorithm uses a random initial population of 20 integer vectors corresponding to the selected alternatives for all actions in 20 alternative military plans. We use the genetic algorithm in the MATLAB Optimization Tool 100 times for each of the NONBs of the stacked design. We use a genetic algorithm with stochastic uniform selection, rank scaling of the fitness function, 90% use of a scattered crossover and 10% elitist copy of the best individuals from generation to generation, no mutation, and ρ^i as fitness function for the i^{th} NONB.

The average pairwise correlations $\bar{\rho}$ over all 100 NONBs obtained (6) is 1.49%, with an average maximum correlation $\overline{\rho_{max}}$ over all NONBs obtained (7) of 9.66%.

The 1000 design points of the stacked design of these 100 NONBs are used as the design of experiment for the LH approach.

V. The A*-search Approach

The A* approach can best be described as a combination of tree search using A* and a series of sequential simulations at each node in the three, where nodes correspond to action alternatives in the military plan.

This means that, on the basis of a given system state, we simulate the effect of each action alternative in the plan, but only one step at a time. Doing so, for every alternative, we get a new system state whose distance h to the desired end state is calculated. Given the alternative that is best, i.e., closest to our end state, we simulate the possible subsequent alternative actions that are provided. One of these alternatives leads to a condition that is closer than the others, Fig. 2.

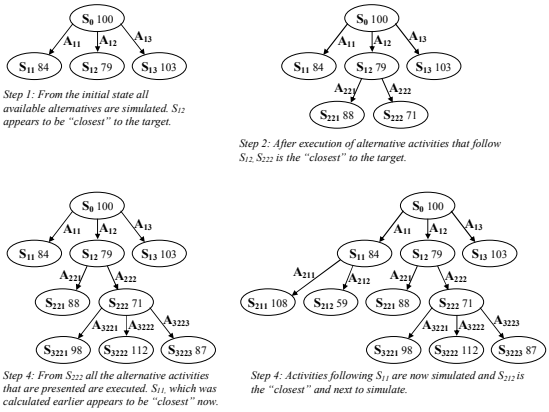

Fig. 2. An example illustrating the four first steps in a simulation of a plan starting with initial system state S_0 with the *distance* of 100 to the desired end state.

Therefore, we must also compare the new distance with the best of the distances that have been simulated and recorded in the previous simulation steps, but then had opted out in favor of a better sequence of alternative actions. The best sequence now becomes the basis for the next simulation step.

We employ two alternatives to standard A* search. First we change the distance function into $f = g + 80h$ in order to obtain a good balance between the impact of g and h. This is necessary since A* will never be able to reach the goal state. Secondly, we let A* continue to run after a best plan has been found, in order to develop a larger set of 1000 alternative plans.

VI. The Comparison of Approaches

In this section we will compare the three approaches A*, NONB and GA based on different methods of analyzing input and output data from the simulation system for the different approaches. We will compare the three approaches based on their ability in minimizing f, g and h. We ignore issues of computational complexity, execution time, etc. since almost all computation time is in running the simulation with very little overhead for the three methods.

A. Comparing Simulation Input Data

Before we turn to the performance of the three approaches of selecting design points for simulation, let us observe the input data distribution [17]. In Fig. 3 we observe series of averages and normalized input distributions over all action alternatives. For example, if an action has two alternatives and the frequency over these alternatives over the two alternatives are 667 and 333 they will be plotted as 1.33 and 0.67, respectively. From these plots we notice the relative high space filling of NONB compared to A* and GA. This may be an advantage or disadvantage depending on the difficult of the problem. A difficult energy landscape may favor NONB or GA, but a simple landscape may favor A*.

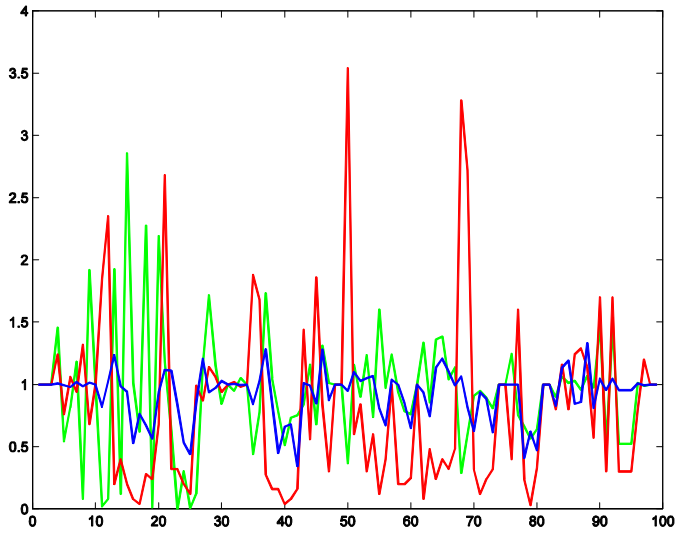

Fig. 3. Input distribution over all action alternatives for A* (green), NONB (blue) and GA (red).

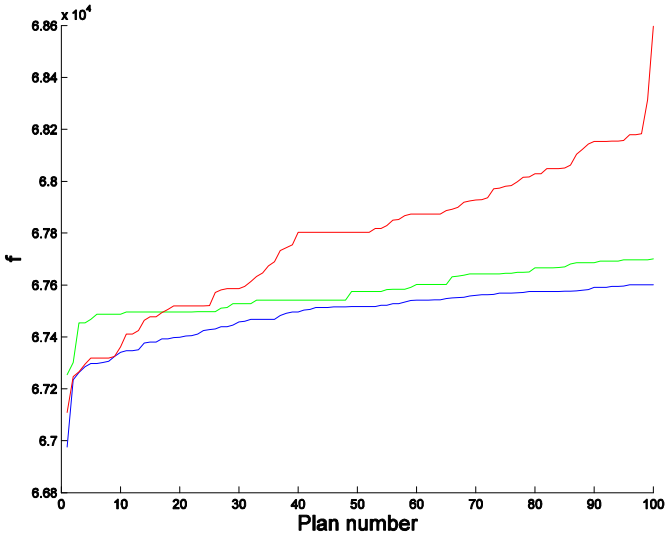

Fig. 4. Comparing measure *f* for A* (green), NONB (blue) and GA (red) for the 100 *f*-best plans.

B. *Comparing Simulation Output Data*

In TABLE I we look at the single best plan found by each of the three different approaches. While we try to optimize both *g* and *h* as measures of plan effectiveness we consider *h*, the remaining distance to the end state after the performance of the last action as the more important measure. While we observe that all three approaches work reasonable fine we notice that NONB performs 0.5‰ and 2.7‰ better than A* and GA, respectively, in minimizing *h*. On the other hand GA is 2.6% and 6.6% superior to NONB and A*, respectively, in minimizing *g*.

TABLE I. COMPARING EVALUATION FOR A*, NONB AND GA.

Best	A*	NONB	GA
$\min_i f(P_i)$	67 254.1	66 975.9	67 109.4
$\text{Min}_j g(P_j)$	2353.8	2255.9	2197.4
$\min_k h(P_k)$	809.3	808.9	811.1

NB. P_i, P_j and P_k may be different plans. Thus, *g* and *h* does not sum to *f* for different plans.

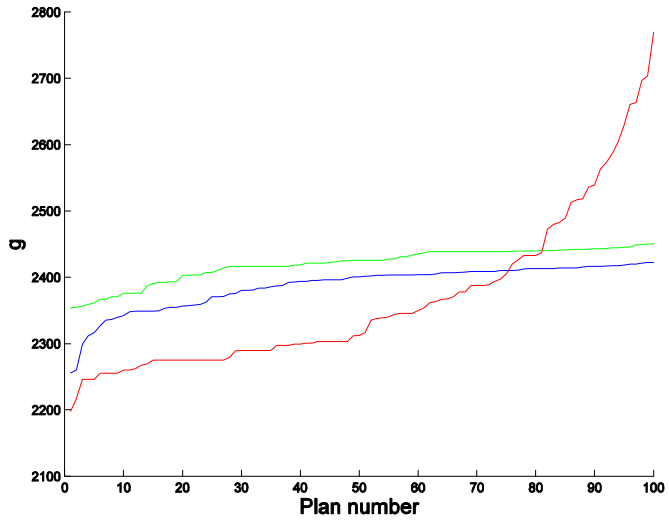

Fig. 5. Comparing measure *g* for the 100 *g*-best plans; A* (green), NONB (blue) and GA (red).

Below, we show 3 figures (Fig. 4, Fig. 5, Fig. 6) where *f*, *g*, and *h* are plotted vs. the plan number for the 100 best plans (i.e., with lowest *f*, *g* and *h*) where plan 1 is the best plan for that graph from A*, NONB and GA. This means that a plan with a certain number can very well be different plans for the three graphs in a figure. We plot it like this simply to compare the qualities of the plans found using the different methods.

For the overall measure *f*, Fig. 4, we find that NONB and GA perform better than A* for the top 15 best plans. NONB has the overall best result for the top 100 plans. For the first 10 plans, GA and NONB perform approximately equal. Maybe somewhat confusing since NONB is not an approach that tries to find the best plan, while A*, continuously, during simulation tries to find the path through a sequence of action alternatives that eventually forms a plan that minimize *f*. On the other hand, it can be prone to get stuck in local minima of the "*f*-energy landscape". Note that A* only guarantees to find the global minima when the evaluation of *g* and *h* are deterministic and

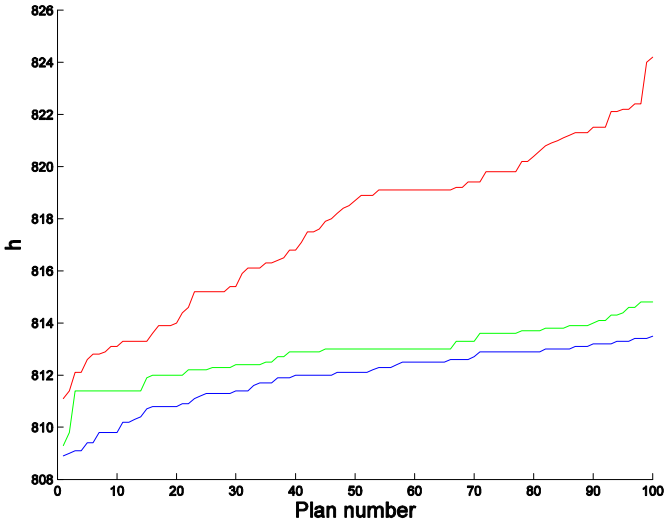

Fig. 6. Comparing measure *h* for the 100 *h*-best plans; A* (green), NONB (blue) and GA (red).

the goal state is possible to reach. None of which is true here. NONB, as explained in section IV, distributes − yet very sparsely, but as effective as possible − the initial parameters more randomly within the search space.

For measure *g*, GA is clearly better than both NONB and A*, however, GA quickly deteriorates beyond the 50 *g*-best plans due to a relative small population size of 150 used [7]. Had a larger population sized been used results may have been better.

Measure *h* might be regarded the most important; the distance left to the goal after the execution of the last action in the plans. NONB is here slightly better than A*, while GA perform worse than both.

We also note that none of the three approaches A*, NONB or GA manages to reach the goal state ($h = 0$). The goal in a military conflict might be a theoretically ideal state that is very difficult to reach in practice. It may also be noticed that the number of states of the system far outnumbers the number of possible plans, which makes the task of reaching the goal state close to impossible. We have 2.164×10^{23} possible plans and 1.722×10^{361} different states. Obviously, for each initial state it is only possible to reach at most 2.164×10^{23} of all states. Note that A* is guaranteed optimal only if the problem is deterministic and if the goal state is possible. Thus with these conditions A* cannot be guaranteed to be optimal and more robust methods may be of interest.

C. Comparing Sequential Simulation Performance

In Fig. 7, we see the development of *g* and *h* during simulation of action-by-action of the 10 best plans found for A*, NONB and GA, respectively. As mentioned in section V, small values of measures *g* and *h* are considered to be good plans, i.e., we want to minimize the total effort *g* as well as the remaining distance *h* after the last action is performed to the end state. It lies inherently in the A* algorithm that it outputs the plans in order by minimum *f*. For NONB and GA, the plans are sorted based on minimum *f* after simulation.

We also note that since different plans can contain different numbers of actions, the number of simulation steps, one step per simulated action, can vary.

For *g*, the total distance during the simulation of a plan, we see that most actions tend to add about the same distance which gives a rather smooth linear rise for *g*. Measure *h* varies much more; several of the actions executed actually takes us away from the end state (increasing *h*). On average, however, we approach the goal slightly in a "two steps forward, one step backward" manner.

It seems that all three methods perform approximately equal in minimizing *h*.

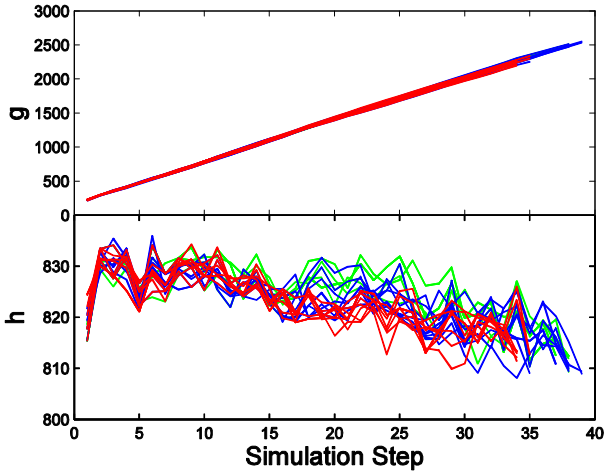

Fig. 7. The stepwise development of *g* and *h* during simulation for the 10 best plans for A* (green), NONB (blue), GA (red).

D. Comparing Regression Tree Analysis

Regression Analysis Trees [18] can be used to hierarchically find the influences of a set of input variables on a dependent continuous output variable. As described above, each plan consists of a set of input actions where some have several discrete alternatives. Each plan produced from, e.g., A*, NONB or GA, is a certain combination of action alternatives, and the continuous *g* or *h* value may be chosen as the dependent output for each plan. Especially, the 1000 plans produced by A* and NONB make good statistical basis for training a regression tree on these data to find the most important actions concerning their influence on *g* and *h*, see Fig. 8. Before doing the analysis, every plan set is sorted by increasing *f* so the best plan have number 1, the second best number 2, etc. However, every tenth plan in each respective plan set from A*, NONB and GA is first removed from the

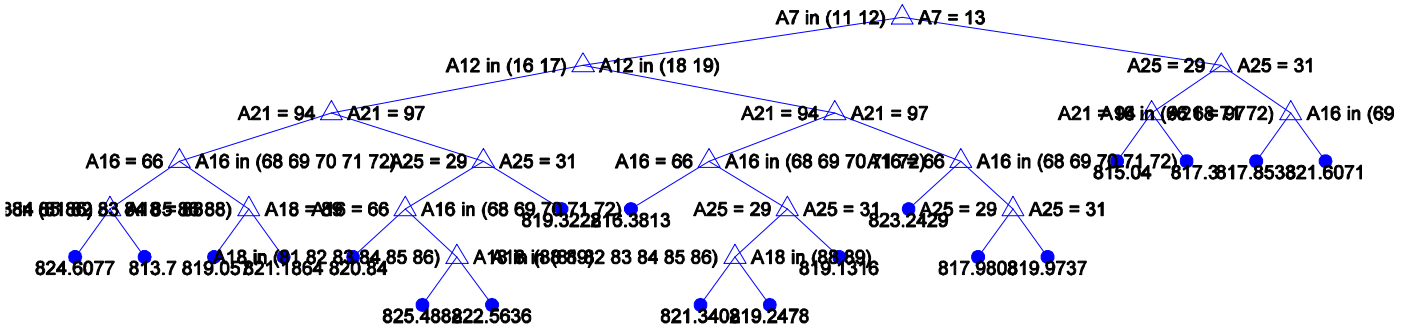

Fig. 8. A regression tree of *h* based on 900 simulations using A* search. Only the 18 most important branching points on 6 levels ending in 19 leafs are shown. The full tree with the default MATLAB Statistics Toolbox setting gets 176 branching points on 16 levels ending in 177 leafs.

training set before training the respective tree; these will instead be used to test the predictive accuracy of the trees as discussed below.

The tree shown in Fig. 8 is analyzed from top to bottom. As an example, if alternative 11 of the possible {11, 12, 13} is chosen for action A_7, action A_{12} is second most important. Eventually we reach a leaf that predicts a value of h for the chosen sequence of alternatives.

The average, μ, value for g and h is calculated as

$$\mu_g^x = \frac{1}{n}\sum_{i=1}^{n} g(P_i) \tag{8}$$

and

$$\mu_h^x = \frac{1}{n}\sum_{i=1}^{n} h(P_i) \tag{9}$$

where $n = 100$ and $x = \{A^*, \text{NONB}\}$. Standard deviation, σ, is calculated as

$$\sigma_g^x = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(g(P_i) - \mu_g^x\right)^2} \tag{10}$$

and

$$\sigma_h^x = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(h(P_i) - \mu_h^x)^2}. \tag{11}$$

TABLE II. COMPARING TRACKING ERROR FOR REGRESSION TREES TRAINED ON 900 DESIGN POINTS BY A* AND NONB.

Best 100	A*	NONB
pred. g / g (μ_g^x)	1.0013	1.0035
pred. g / g (σ_g^x)	0.0132	0.0178
pred. h / h (μ_h^x)	0.99996	1.0002
pred. h / h (σ_h^x)	0.0047	0.0053

NB. A* and NONB values are based on 100 out of 1000 design points (in minimizing f).

As the average tracking error for each method can easily be compensated for, we consider the standard deviation to be the more important measure. We find (TABLE II) that A* standard deviation compared to NONB are 74% for g, and 89% of for h. Thus, we consider a regression tree trained on design points derived by A* to be a slightly better predictor of both g and h for a new plan.

For GA which was analyzed in another study [7] we have only the 100 best design points available. Since GA used a population of 150 design points in evolution we have to focus on the top 10 best design points as any higher number would include examples that are not fair representatives of the method. It is of course possible to use a larger population in GA which would have allowed us to make a direct comparison of the best 100 as with A* and NONB in TABLE III but this was not done in that study. We recalculate (8)–(11) with $n = 10$ and $x = \{A^*, \text{NONB, GA}\}$, see TABLE III.

TABLE III. COMPARING TRACKING ERROR FOR REGRESSION TREES TRAINED ON 90 DESIGN POINTS BY A*, NONB AND GA.

Best 10	A*	NONB	GA
pred. g / g (avg.)	1.0088	1.0014	0.9955
pred. g / g (std.)	0.0148	0.0182	0.0175
pred. h / h (avg.)	1.0059	1.0070	1.0014
pred. h / h (std.)	0.0044	0.0040	0.0044

NB. A*, NONB and GA values are here based on 10 out of 100 design points (in minimizing f).

From TABLE III we observe a standard deviation for A* of 81% and 84% for g compared to NONB and GA, respectively, and a standard deviation for NONB of 91% for h compared to both A* and GA. We find that a regression tree trained on design points derived by A* is slightly better than a regression tree trained on NONB design points in prediction g (with GA in second place). For h, NONB turns out to be slightly better than both A* and GA.

In Fig. 9 shows the predictive ability when comparing g of every tenth of the 1000 plans from A*, with what is predicted by the tree in Fig. 8, trained from the remaining 900 plans. The upper section of the figure shows the real g output data from the simulator (blue) vs. the predicted g by the regression tree (red). The tracking is very good, and implies that the trained tree is good in predicting the total length for the combination of actions in a plan found by A*. The lower section shows their quotient with an average tracking error of 1.1‰ and a standard deviation of 8.1‰.

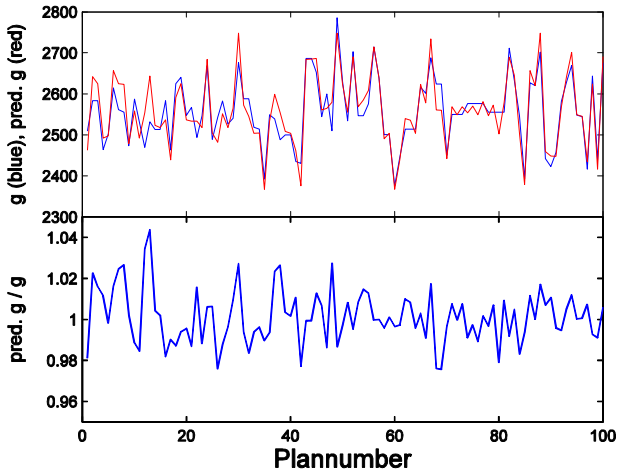

Fig. 9. This figure show the tracking accuracy of g for a regression tree learnt using 900 design points from A*-search.

In Fig. 10 the test for the NONB regression tree for h is shown. Here the tracking is worse. This might not be difficult to understand since it is harder to predict the distance left to the end state. We also see a drift upwards for simulated h which is just very vaguely seen in the trend for the predicted values of h for the corresponding plans. The increasing trend of h is not difficult to understand since the plans are sorted according to f where $f = g + h$ and we minimize f.

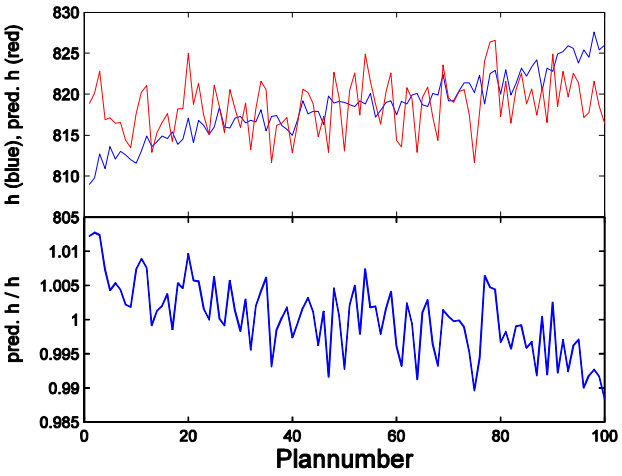

Fig. 10. This figure show the tracking accuracy of *h* for a regression tree learnt using 900 design points from NONB.

VII. Conclusions

In this paper we compare the three approaches of NONB, GA, and A* for assessing military plans for evaluation by simulation. We observe that all three approaches performed relatively well. However, for non-deterministic problems with very difficult search space's (like evaluation of military plans) we conclude that the more robust approach of NONB performs 0.5‰ and 2.7‰ better than A* and GA, respectively, in minimizing the difference between the final state reached after the last action of the plan and the sought after end state (*h*, TABLE I). For problems with more than two criteria it is often useful to handle the problem as a Multi-criteria Decision Making problem as it will be increasingly difficult for decision makers to find an appropriate weighting of the criteria. Instead, decision makers may assign preferences regarding the importance of different criteria [19, 20].

References

[1] G. Horne, et al., "Data farming in Support of NATO – Final report of task group MSG-088," NATO Research and Technology Organisation, Neuilly-sur-Seine, France, STO Tech. Rep. STO-TR-MSG-088, March 2014. [Online]. Available: http://ftp.rta.nato.int/public//PubFullText/RTO/TR/STO-TR-MSG-088///$$TR-MSG-088-ALL.pdf

[2] J. Schubert, et al., "Simulation-based decision support evaluating operational plans – Final report," Swedish Defence Research Agency, Stockholm, Sweden, Tech. Rep. FOI-R--3635--SE, January 2013.

[3] J. Schubert, F. Moradi, H. Asadi, P. Hörling, and E. Sjöberg, "Simulation-based decision support for effects-based planning," in Proceedings of the 2010 IEEE International Conference on Systems, Man and Cybernetics, October 2010. Piscataway, NJ: IEEE, 2010, pp. 636−645.

[4] H. Asadi and J. Schubert, "A stochastic discrete event simulator for effects-based planning," in Proceedings of the 2013 Winter Simulation Conference, December 2013. Piscataway, NJ: IEEE, 2013, pp. 2842–2853.

[5] F. A. C. Viana, G. Venter, and V. Balabanov, "An algorithm for fast optimal Latin hypercube design of experiments," International Journal for Numerical Methods in Engineering, vol. 82, no. 2, pp. 135–156, April 2010.

[6] H. Vieira Jr., S. Sanchez, K. H. Kienitz, and M. C. N. Belderrain, "Generating and improving orthogonal designs by using mixed integer programming," European Journal of Operational Research, vol. 215, no. 3, pp. 629−638, December 2011.

[7] I. Younas, R. Ayani, J. Schubert, and H. Asadi, "Using genetic algorithms in effects-based planning," in Proceedings of the 2013 IEEE International Conference on Systems, Man and Cybernetics, October 2013. Piscataway, NJ: IEEE, 2013, pp. 438−443.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, April 2002.

[9] H. Vieira Jr., S. M. Sanchez, K. H. Kienitz, and M. C. N. Belderrain, "Efficient, nearly orthogonal-and-balanced, mixed designs: an effective way to conduct trade-off analyses via simulation," Journal of Simulation, vol. 7, no. 4, pp. 264–275, November 2013.

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

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

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

[13] J. Schubert, "Analysis and assessment of effects-based plans," in Proceedings of the NATO Symposium on Analytical Support to Defence Transformation (SAS-081), April 2010. Neuilly-sur-Seine: NATO Research and Technology Organisation, 2010, Paper 33, pp. 1−18.

[14] J. Schubert, "Subjective effects-based assessment," in Proceedings of the Eleventh International Conference on Information Fusion, July 2008. Piscataway, NJ: IEEE, 2008, pp. 987−994.

[15] J. Schubert, "Multi-level subjective effects-based assessment," in Proceedings of the 13th International Conference on Information Fusion, July 2010. Piscataway, NJ: IEEE, 2010, Paper We3.4.1, pp. 1−8.

[16] H. Vieira Jr., S. M. Sanchez, K. H. Kienitz, "Improved efficient, nearly orthogonal, nearly balanced mixed designs," in Proceedings of the 2011 Winter Simulation Conference, December 2011. Piscataway, NJ: IEEE, 2011, pp. 3605–3616.

[17] J. Schubert and P. Hörling, "Explaining the impact of actions," in Proceedings of the 15th International Conference on Information Fusion, July 2012. Piscataway, NJ: IEEE, 2012, pp. 354−360.

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

[19] J. Schubert and P. Hörling, "Preference-based Monte Carlo weight assignment for multiple-criteria decision making in defense planning," in Proceedings of the 17th International Conference on Information Fusion, July 2014. Piscataway, NJ: IEEE, 2014, in press.

[20] J. Schubert, "Partial ranking by incomplete pairwise comparisons using preference subsets," in Proceedings of the 3rd International Conference on Belief Functions, September 2014. Berlin: Springer, 2014, in press.