Effective Course-of-Action Determination to Achieve Desired Effects

Sajjad Haider and Alexander H. Levis, *Life Fellow, IEEE*

***Abstract*—An evolutionary algorithm-based approach to identify effective courses of action (COAs) in dynamic uncertain situations is presented. The uncertain situation is modeled using timed influence nets, an instance of dynamic Bayesian networks. The approach makes significant enhancements to the current trial-and-error-based manual technique, which is not only labor intensive but also not capable of modeling constraints among actionable events. The proposed approach is an attempt to overcome these limitations. It automates the process of COA identification. It also allows a system analyst to capture certain types of constraints among actionable events. Because of its parallel search nature, the approach produces multiple COAs that have a similar fitness value. This feature not only gives more flexibility to a decision maker during mission planning, but it can also be used to generalize the COAs if there exists a pattern among them. This paper also discusses a heuristic that further enhances the performance of the approach.**

***Index Terms*—Course of action (COA), Dynamic Bayesian networks (DBNs), effects-based operations, evolutionary algorithms (EAs), optimization, timed influence nets (TINs).**

I. Introduction

FINDING effective courses of action (COAs) in uncertain situations has always been a challenging task for decision makers (DMs). The job becomes more challenging if the situation under consideration is also time dependent. Most of the time-critical activities in military, political, medical, and financial domains fall into this category. An effective COA, in such circumstances, is not only required to meet the objectives set by an organization, but it should also do it in a timely manner. To identify such COA(s), a system analyst has to analyze many alternatives and pick the (optimal) ones that satisfy the requirements. The existence of an extremely large number of possibilities, however, makes it a very difficult process. Another difficulty is the modeling of uncertain time-dependent cause and effect relationships among events in the problem domain. There exist several techniques in the literature to model such relationships, but probability theory, in general, and Bayesian networks (BNs) [28], [29], in particular, have become the tool of choice among practitioners for modeling uncertain causal relationships. Despite the rapid popularity of BN in the last two decades, most of the attention has remained focused on the enhancement of the knowledge acquisition interface of BN and the development of approximate/simulation-based probabilistic reasoning algorithms that can run in nonexponential time (the exact inference in BN is NP-complete [6]). Furthermore, most of the efforts were made to model static (time-independent) situations. It is only very recently that the focus has shifted to the integration of time and uncertainty, but even in that case, the focus remains on the development of efficient reasoning algorithms and the enhancements of knowledge acquisition interfaces. As a result, not much work has been reported in the literature that attempts to automate the process of effective COA determination in dynamic uncertain situations. The current practice among the community is to model a dynamic situation using variants of Dynamic BNs (DBNs) [26] and then use a trial-and-error method to identify effective COAs. The drawbacks of this manual trial-and-error process are rather obvious. First, it is a labor-intensive process to try several alternatives, and second, and more importantly, the chances of finding a reasonably good solution among millions (or even trillions) of possibilities can be very slim.

This paper is an attempt to automate the process of effective COA determination. It uses evolutionary algorithms (EAs) [8] as the search mechanism, whereas the underlying mathematical model is based on timed influence nets (TINs), a concise graphical–theoretical probabilistic approach to specify the cause and effect relationships among the variables of interest (actions, desired effects, and other uncertain events). This paper presents a set of metrics that can be used as measures of effectiveness during the process of COA selection. The approach also allows a system analyst to model certain types of constraints among actionable events. The complete methodology is named effective course-of-action determination using evolutionary algorithm (ECAD-EA). Because of the parallel search mechanism of the EA, the approach generates multiple solutions that have a similar fitness value. These multiple solutions can later be grouped together to obtain a generalized plan of action.

Very few efforts have been reported in the literature that attempt to automate the process of COA determination. Exceptions are the work by Wagenhals and Levis [39] and Haiying *et al.* [18]. Wagenhals and Levis converted the influence net (IN) with timing information to colored Petri nets [24] and then used state-space analysis to reveal all the possible sequences of a *probability profile*. The approach works well for small-size problems, but for moderate/large-size problems,

Manuscript received January 23, 2006; revised September 4, 2006. This work was supported in part by the Office of Naval Research under Grant N00014-06-1-0081 and in part by the Air Force Research Laboratory Information Directorate under Grant FA8750-05-C-0145. This paper was recommended by Associate Editor K. Hipel.

The authors are with the System Architectures Laboratory, Department of Electrical and Computer Engineering, Volgenau School of Information Technology and Engineering, George Mason University, Fairfax, VA 22030 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCA.2007.904771

it suffers from the combinatorial explosion of the state space. The work did not consider the modeling of temporal and causal constraints among actionable events. The approach developed by Haiying *et al.* [18] models an uncertain situation using DBNs and uses genetic algorithms [14] to search for a robust strategy. They use simulation to do probabilistic inference in DBN. Because of the time taken by simulation schemes, they have suggested their approach for offline planning. Although no run-time comparison has been made between their approach and ours, the approximate probabilistic inference scheme used by TINs, which is based on loopy belief propagation [11], [25], [27], [29], runs much faster than the simulation-based schemes and thus has a potential to have a smaller running time during the probabilistic inference phase. Furthermore, their approach does not consider the modeling of temporal/causal constraints among actionable events, as presented in this paper. Also, different metrics are needed for the kind of problems presented in this paper.

The rest of this paper is organized as follows. Sections II and III provide a brief overview of TINs and EAs. Section IV explains the ECAD-EA methodology with the help of few examples. A heuristic is suggested in Section V that further enhances the performance of the ECAD-EA methodology. Finally, Section VI presents conclusions and provides future research directions.

II. TINs

BNs were originally designed to capture time-independent cause and effect relationships among random variables in uncertain situations. They have been successfully applied in many areas, including medical diagnosis, human belief systems, forecasting, sensor fusion, system troubleshooting, etc. A detailed account of the real-world applications of BNs is given by Heckerman *et al.* [19]. Despite their popularity, BNs suffer from two major limitations. The first is the amount of data required to completely specify a BN, and the second is the intractability of exact belief propagation algorithms [6]. Work is reported in the literature that attempts to overcome these two limitations. Schemes have been proposed that reduce the amount of data that is required to completely specify a BN. Among them are BN with noisy-OR (BN2O) [1], [9] and the Causal Strength (CAST) logic [3], [30]. To address the intractability of belief propagation algorithms, several approximation-based and simulation-based algorithms have been proposed. These include the loopy belief propagation [11], [25], [27], [29], logic sampling [20], likelihood weighting [12], backward simulation [13], and importance sampling [2], [5]. A detailed comparison of different simulation schemes is given in [4] and [35].

INs [30]–[32] were developed with a similar aim of overcoming the aforementioned two limitations of BNs. They use the CAST logic for knowledge elicitation and a variant of loopy belief propagation for probabilistic inference. They are also closely related to another family of graphical models, namely influence diagrams (IDs) [21]–[23], [34]. An ID has three types of nodes: 1) decision nodes; 2) chance nodes; and 3) utility nodes. All the decisions are assumed to be taken in a sequential manner, and there is a certain precedence relationship among them. Mathematically, an ID can be considered as a BN extended with a utility function, whereas an IN can be considered as a special instance of BN using CAST logic for knowledge elicitation and loopy belief propagation for probabilistic inference.

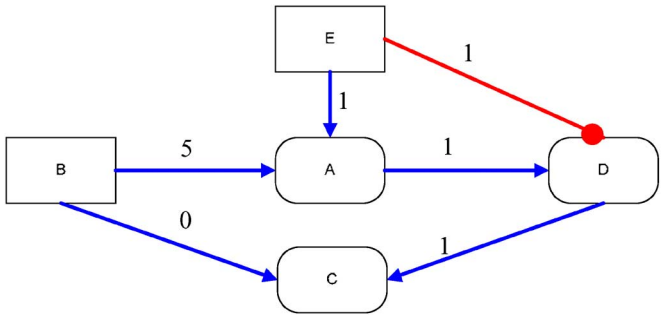


Fig. 1. Sample TIN.

Wagenhals *et al.* [42] have added a special set of temporal constructs to the basic formalism of INs. The INs with these additional temporal constructs are called TINs. TINs have been experimentally used in the area of effects-based operations for evaluating alternate COAs and their effectiveness in achieving mission objectives [39]–[41], [43]. The provision of time allows for the construction of alternate COAs as a timed sequence of actionable events.

The modeling of the causal relationships in TINs is accomplished by creating a series of cause-and-effect relationships between some desired effects and the set of actions that might impact their occurrence in the form of an acyclic graph. The actionable events in a TIN are drawn as root nodes (nodes without incoming edges). A desired effect, or an objective in which a DM is interested, is modeled as a leaf node (node without outgoing edges). Typically, the root nodes are drawn as rectangles, whereas the nonroot nodes are drawn as rounded rectangles. Fig. 1 shows a partially specified TIN. Nodes B and E represent the actionable events (root nodes), whereas node C represents the objective node (leaf node). The directed edge with an arrowhead between two nodes shows that the parent node has a positive influence on the chances of the child node being true, whereas the roundhead edge shows that the parent node has a negative influence on the chances of the child node being true. The inscription associated with each arc shows the time delay it takes for a parent node to influence a child node. For instance, event B in Fig. 1 influences the occurrence of event A after 5 time units.

The purpose of building a TIN is to evaluate and compare the performance of alternative COAs. The impact of a selected COA on the desired effect is analyzed with the help of a *probability profile*. Consider the TIN shown in Fig. 1. Suppose the following COA is decided: actions B and E are taken at times 1 and 7, respectively. Because of the propagation delay associated with each arc, the influences of these actions impact event C over a period of time. As a result, the probability of C changes at different time instants. A probability profile draws these probabilities against the corresponding time line. The probability profile of event C is shown in Fig. 2.

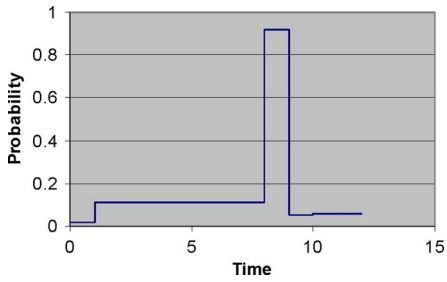


Fig. 2. Probability profile for node C.

The following items characterize a TIN, whereas a formal definition is given in Definition 1.

1) A set of random variables that makes up the nodes of a TIN. For semantic purposes, the root nodes are referred to as actions, whereas the leaf nodes are referred to as desired effects. All the variables in a TIN have binary states.
2) A set of directed links that connect pairs of nodes.
3) Each link has associated with it a pair of parameters that shows the causal strength of the link (usually denoted as g and h values). A positive/negative value of h means that if the parent node is true, it has a positive/negative influence on the child node being true. A positive/negative value of g means that if the parent node is false, it has a positive/negative influence on the child node being true. For visualization purposes, if h is positive, the corresponding link is drawn as an arrowhead, whereas if h is negative, the link is drawn as a roundhead.
4) Each nonroot node has an associated baseline probability, whereas a prior probability is associated with each root node. The baseline probability is similar to the "leak probability" in the BN2O approach, which suggests that the variable can still be true with some probability even if none of the causes are present.
5) Each link may have a corresponding delay d (where $d \geq 0$) that represents the communication delay.
6) Each node may have a corresponding delay e (where $e \geq 0$) that represents the information processing delay.
7) A pair (p, t) for each root node, where p is a list of n real numbers representing probability values, and t is a set of n time intervals. For each probability value, a corresponding time interval is defined in t. In general, (p, t) is defined as

$$([p_1, p_2, \ldots, p_n], [[t_{11}, t_{12}], [t_{21}, t_{22}], \ldots, [t_{n1}, t_{n2}]]),$$
$$\text{where } t_{i1} < t_{i2} \text{ and } t_{ij} \geq 0 \quad \forall i = 1, 2, \ldots, n \text{ and } j = 1, 2.$$

The last item in the aforementioned list is referred to as an input scenario or, sometimes (informally), as a COA.

Definition 1: TIN: A TIN is a tuple $(\mathbf{V}, \mathbf{E}, \mathbf{C}, \mathbf{B}, \mathbf{D_E}, \mathbf{D_V}, \mathbf{A})$, where

- $\mathbf{V}$: set of nodes;
- $\mathbf{E}$: set of edges,
- $\mathbf{C}$ represents causal strengths: $\mathbf{E} \rightarrow \{(h, g)$ such that $-1 < h, g < 1\}$;
- $\mathbf{B}$ represents the baseline/prior probability: $\mathbf{V} \rightarrow [0, 1]$;
- $\mathbf{D_E}$ represents delays on edges: $\mathbf{E} \rightarrow \mathbf{N}$;
- $\mathbf{D_V}$ represents delays on nodes: $\mathbf{V} \rightarrow \mathbf{N}$;
- $\mathbf{A}$ (input scenario) represents the probabilities associated with the sequence of actions and the time associated with them

$$\mathbf{R} \rightarrow \{([p_1, p_2, \ldots, p_n], [[t_{11}, t_{12}], [t_{21}, t_{22}], \ldots, [t_{n1}, t_{n2}]]),$$
$$\text{such that } p_i = [0, 1],\ t_{ij} \rightarrow \mathbf{Z},\ \text{and } t_{i1} < t_{i2},$$
$$\forall i = 1, 2, \ldots, n \text{ and } j = 1, 2, \text{ where } \mathbf{R} \subset \mathbf{V}\}.$$

III. EAs

EAs can be interpreted as a parallel adaptive search procedure that mimics the metaphor of natural biological evolution. They have been applied to a wide variety of application areas, including optimization, search, learning, automated programming, adaptation, design, control, etc. Their popularity lies in the fact that they make little assumptions about the landscape and work for problems where classical optimization schemes fail, such as when the derivative of a function does not exist or the function is discontinuous. The current EAs have their roots in three distinct efforts that were initiated in parallel almost four decades ago: 1) evolution strategies [36], [37]; 2) evolutionary programming [10]; and 3) genetic algorithms [7], [14].

An EA consists of a *population of individual solutions* that are selected and modified to discover overall better solutions in the search space. An individual in the population is referred to as *genome* or *chromosome*. The representation of chromosome can be divided into two categories: *genotype* and *phenotype*. When a representation is stated in its natural state, it is said to be a phenotype. For example, a real-valued optimization problem would be represented as a set of real-valued coordinates. On the other hand, it may be necessary to map a phenotype representation into another structure to make it easier for the algorithm to modify and exchange information. In many cases, the phenotype can be mapped into a representation that resembles, at a very high level, a sequence of local structures or building blocks. In this case, the representation is referred to as a genotype. A chromosome consists of a number of *genes*, whereas each gene represents one characteristic of an individual. A *fitness* function evaluates the quality of the chromosome in the context of a given problem.

During the search for a better solution, one (or more) solution(s) is (are) modified with the help of reproduction operators to produce new *offspring* in the population. *Crossover* and *mutation* are the main reproductive operators used in an EA. In crossover, two or more individuals exchange information to create one or more new individuals. Mutation, on the other hand, is a genetic operator by which small modifications are made to a single genotype or phenotype of a selected individual. For real-valued phenotype approaches, the mutation operator is usually of the form of a small Gaussian change to the phenotype. For binary-coded genotypes, a mutation is a random bit flip.

Selection in EAs is the process of choosing which individuals reproduce offspring and which individuals survive to the next

TABLE I
CANONICAL EA

```
Randomly generate the initial population
Do until a stopping criterion is met
    Select parent(s) using a selection procedure.
    Create new offspring(s) by applying the variation operators on the parents.
    Compute the fitness of the new offspring(s).
    Select member(s) of the population to die using a selection procedure.
```

generation. When selection is used to choose which individuals reproduce, the process is referred to as preselection (*parent selection*). When it is used to select the individuals that survive to the next generation, it is called postselection (*survival selection*). Selection can further be categorized as deterministic or probabilistic. Deterministic selection tends to behave more like greedy hill-climbing algorithms and exploits the nearest areas with promising solutions. Probabilistic selection schemes are more exploratory and search the landscape. Some of the popular selection schemes are given in the list that follows, whereas the major steps involved in a canonical EA are shown in Table I [8].

- Fitness proportional: Individuals are selected based on their fitness in proportion to the other individuals in the population.
- Rank selection: Individuals in the population are first ranked by fitness and then selected for reproduction based on a probability proportional to rank.
- Binary tournament: Two individuals are randomly selected from the population and compared. The one with the highest fitness is selected for reproduction. Then, another two individuals are randomly selected, and the best fit is kept as the mate to the first parent.
- Truncation: This scheme is used during postselection. Populations of parents and offspring are merged together, and the top k fittest individuals survive to the next generation.
- Uniform stochastic: Individuals are selected with uniform probabilities.

IV. ECAD-EA

Once a system analyst captures an uncertain situation using TIN, the next task the analyst confronts is the identification of COAs that would maximize the likelihood of achieving a desired effect. The task is sometimes, rather informally, referred to as the best COA determination. This paper presents a methodology, named ECAD-EA, to accomplish this task. The methodology uses EAs to produces several alternative COAs that have a *similar* likelihood of producing the desired effect. The advantage of having alternative COAs is that they not only provide more options to a DM, but they can also be grouped together to generalize temporal relationships between actionable events at a qualitative level. The generalized relationships provide the sequence in which action should take place, thus giving more flexibility to a DM during the planning stage of a mission.

The ECAD-EA methodology also allows a system analyst to capture certain types of temporal and causal constraints that could exist among actionable events. For instance, it is possible that, due to temporal requirements, two actions need to be executed in a particular sequence. Conversely, some situations may require two actions to be taken at the same time to have a maximum impact on a desired effect. Currently, TINs (or even DBNs) do not have a mechanism to specify these constraints. This paper develops a constraint specification language to capture several types of temporal and causal relationships. The ECAD-EA methodology takes these constraints into consideration while identifying an effective strategy in a given situation. Another important contribution of this paper is the development of a set of metrics that can be used for the automatic evaluation and performance comparison of alternative COAs. The explanation of the set of metrics and the constraint specification language are given in Sections IV-A and B, which are followed by the specification of the EA (Section IV-C).

A. Identification of a Set of Metrics

In the TIN-based formalism, the impact of a selected COA on a desired effect is analyzed with the help of a probability profile. This paper suggests several metrics that evaluate a COA's performance as a function of the desired effect's probability profile produced by that particular COA. For instance, consider the two probability profiles shown in Fig. 3. The profiles are produced by two different COAs. Each profile represents the probability of achieving a desired effect during time interval 0 to 12. There are several features of a probability profile that can be considered as potential metrics. For instance, a DM in some situations might be interested in having a profile that has an overall high probability value during the course of a campaign. Conversely, there might be situations when the DM is interested in only those profiles that have a higher value during a particular interval, say during 2 to 8 time units, and does not care about the other times. Similarly, in some situations, the DM might be interested in only particular time instances, say time instances 4, 7, and 9, whereas the other instances are not really important to the DM. Thus, there are several features of a probability profile that can be considered as potential metrics, and the selection of a particular COA really depends on the metric that is used to analyze it. The following is a preliminary list of features that are used by the ECAD-EA

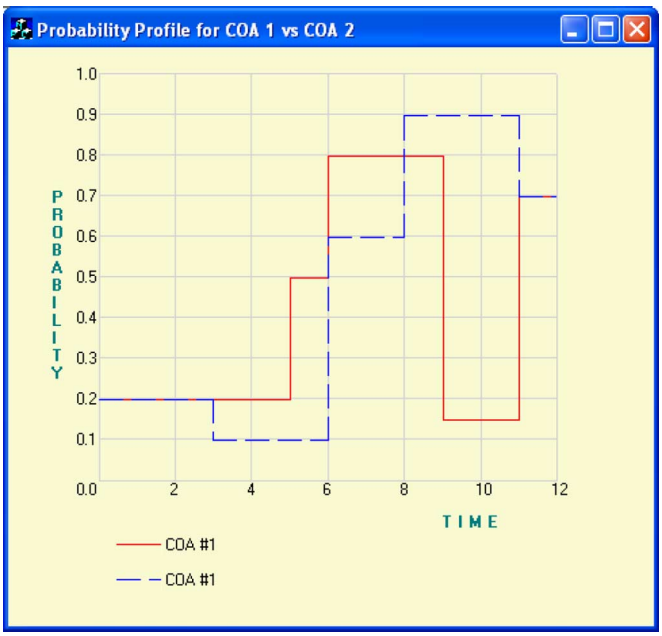


Fig. 3. Two competing COAs.

methodology as the set of metrics. These metrics are explained with the help of the probability profiles of Fig. 3.

1) *Maximum probability achieved:* This metric specifies the highest probability achieved by the probability profile of a desired effect. For instance, in the profiles of Fig. 3, the highest probability produced by COA #2 is 0.9 in contrast to 0.8 produced by COA #1.
2) *Time to reach the maximum probability:* This metric is related to the first one, and it specifies the time at which the maximum probability is achieved. For instance, COA #2 in Fig. 3 has produced a better probability value (i.e., 0.9), but the value is achieved at time 8. COA #1, on the other hand, reaches the probability of 0.8 at time 6. If metric 1 is considered alone, then COA #2 would be considered better, but if a DM is interested in getting the probability of the desired effect above a certain threshold in a timely manner, then COA #1 might be considered better if it satisfies the threshold requirement.
3) *Area under the curve (AUC):* This metric represents the area under a probability profile. The metric prefers those COAs whose overall performance is better than the others. Visual comparison of COAs of Fig. 3 suggests that both COAs perform equally well except during the time interval 9 to 11, when COA #1 has a lower probability of achieving the desired effect. The computation of the metric for both COAs also confirms this and identifies COA #2 as better than COA #1.
4) *Probabilities at specific points/intervals:* A DM might be interested in profiles that have probabilities above (below) a certain threshold or highest (lowest) at specific time instances or intervals. Considering the profiles of Fig. 3, suppose a DM is interested in the profile that has a higher probability during the time interval 4 to 8 (AUC during the interval 4 to 8). COA #1, in this case, is superior to COA #2. On the other hand, if the interval under consideration is between times 8 and 11, then COA #2 would be considered better. The time instances/intervals could be a list of values; that is, the DM may require the probability to be above (below) certain thresholds at times 3, 9, and 11.

A COA is evaluated using either a simple or a composite metric. A composite metric is a weighted combination of several simple metrics. Weights assigned to each of the individual metric may vary from situation to situation and depend on the discretion of a DM. Once a set of metrics is specified, along with their corresponding weights, the fitness of a particular COA is computed as

$$\text{fitness}(\text{COA } x) = w_1 m_1 + w_2 m_2 + \cdots + w_n m_n$$

where w_i is the weight corresponding to the metric m_i, and each metric is normalized on the scale of 0 to 1. For instance, metric 2 is normalized using the expression $(1 - t_x/\max\_t)$, where t_x is the time COA x reaches the given threshold probability, whereas $\max\_t$ is the total duration of the campaign (identified as *observation window* and is further explained later in this paper). If $\max\_t$ is set as 12 for the COAs of Fig. 3, then the normalized metric for COA #1 is 0.5 $[1 - (6/12)]$, whereas for COA #2, it is 0.33 $[1 - (8/12)]$. Similarly, the AUC criteria of metric 3 can be normalized after dividing it by the maximum AUC. This maximum area is the rectangular region between $(0, 0)$ and $(\max\_t, 1)$ and is equal to $\max\_t$.

B. Constraint Specification Language

Several temporal and logical constraints may exist among the actionable events that cannot be directly modeled using TIN. This limitation raises the need for a constraint specification language that can aid a system analyst in modeling certain types of constraints. This paper suggests one such language that assists a system analyst in specifying several types of temporal/causal constraints among actionable events. The consistency of these constraints can be checked by propositional and temporal logic-based systems depending on their nature. Table II describes the constructs of the suggested constraint specification language. Constraints 1 and 2 describe the cases where it is required that some actions remain in a specific state. Constraint 3 requires that A1 and A2 should have the same state (whether "True" or "False"), whereas Constraint 4 requires two actions to have different states. Constraint 5 specifies the time "t" at which action A must take place, whereas Constraint 6 shows that action A1 must happen before action A2. The parameter v represents a time interval measured in integer time units. If the value of v is zero, then A1 should happen at any time earlier than A2, but if v is nonzero (positive), then A1 must happen at least v time units before A2. Constraint 7 specifies that two actions A1 and A2 should happen at the same time.

In addition to providing the aforementioned set of constraints, a system analyst must also provide two additional parameters to the ECAD-EA methodology. These parameters create a boundary around the solution space. The first parameter is the duration in which all the actions must take place. This duration is referred to as either *window of opportunity* or *action*

TABLE II
CONSTRUCTS OF THE CONSTRAINT SPECIFICATION LANGUAGE

1. True	A			// Factual Constraint
2. False	A			// Factual Constraint
3. Same_State	A1	A2		// Causal Constraint
4. Different_State	A1	A2		// Causal Constraint
5. At_Time	A	t		// Temporal or Factual Constraint
6. Before	A1	A2	v	// Temporal or Causal Constraint
7. Equal_Time	A1	A2		// Temporal or Conceptual Constraint

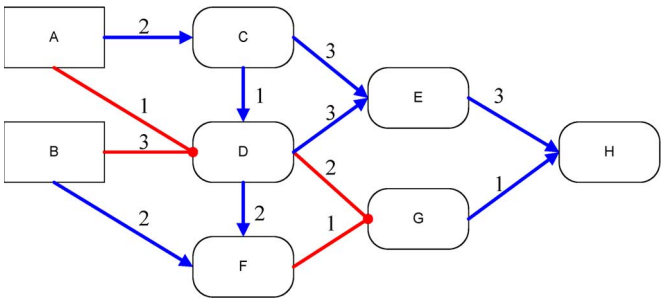


Fig. 4. TINs with two actionable events.

window. The second parameter is the time period in which a DM is interested in getting the desired results. This duration is referred to as *observation window*. This parameter becomes important when assigning a fitness value to a COA.

C. Specifics of the EA

The population of the EA in the ECAD-EA methodology consists of candidate COAs. Each individual in the population consists of actions and the times at which the actions are executed. Actionable events in a TIN have binary states; thus, a bit representation is used for the actionable events, whereas an integer value representation is used for the time at which an action is executed. An individual in the population of solutions has the following structure:

$$\langle \text{Action } 1, \text{time}(\text{Action } 1), \text{Action } 2, \text{time}(\text{Action } 2), \ldots, \text{Action } n, \text{time}(\text{Action } n) \rangle .$$

The other parameters of the EA, such as parent/survival selection schemes, mutation rate, crossover, and population/generation sizes, may vary from problem to problem and can be specified by a system analyst. The software implementation of the ECAD-EA methodology *Pythia*[1] has a default value for each of these parameters.

Once an EA is completely specified and all the other inputs, such as windows of observation and opportunity, set of constraints, and set of metrics, are provided to the ECAD-EA methodology, the EA searches the probability profile solution space to determine effective COAs. Consider the TIN shown in Fig. 4. There are two actionable events A and B and one desired effect H. Suppose a DM is interested in selecting a COA that maximizes the probability of H over a given time interval. Hence, AUC is selected as a measure of COA's performance. The window of opportunity is set as an interval between times 1 and 10, whereas the observation window is between times 1 and 20. Thus, the solution space consists of $2^2 \times 10^2 = 400$ COAs. For such a small number of possibilities, it is easy to compare the performance of the EA with the exhaustive search. The results obtained through the exhaustive search provide the solution space of the problem. Fig. 5(a) shows a region of the solution space where both A and B are true. There exist three other such regions for the remaining combinations of A and B: 1) A true, B false; 2) A false, B true; and 3) A false, B false. Since the global maximum lies in the region where both A and B are true, therefore we would focus on the solution space of Fig. 5(a). It can be seen in the figure that the global maximum lies in the space where action B is taken before action A.

To analyze the performance of the EA, we run the ECAD-EA methodology over event H's probability profile solution space. The parameters of the EA are shown in Table III. For instance, the parent selection scheme is uniform stochastic, whereas the survival selection scheme is binary tournament. Since all the actions have binary states, a bit-flip mutation operator is used for them, whereas a delta x mutation operator is used for the times at which actions are executed. The population and the generation sizes are set to 10 and 20, respectively. During its search for the best COA, the EA found many solutions that have a very close fitness value. The top four ones are shown in Table IV. The solution shown in the first row is read as follows: "to maximize AUC, action A is made True at time 9, whereas action B is made True at time 5." The other rows in the table can be read in a similar manner. All the solutions are centered on the region where the global maximum lies in Fig. 5(a). The results agree with the intuition developed earlier by looking at the solution space that the global maximum lies in the region when B is taken before A. Thus, the advantage of generating multiple solutions with a very close fitness value is that the solutions could be generalized to produce a plan that describes time at a qualitative level or at a more general quantitative level. The generalization is useful for sequencing the actions in a mission. Haider and Levis [15] describe the steps involved in generalizing temporal relationships between actionable events from multiple solutions.

The solution space of Fig. 5(a) was produced without considering any user-defined constraints. Suppose it is required that a feasible solution must satisfy the constraint that A must

[1]Pythia is a software tool developed by the System Architectures Laboratory, Department of Electrical and Computer Engineering, Volgenau School of Information Technology and Engineering, George Mason University, to build graph-based decision theoretic models and to perform several analyses on them.

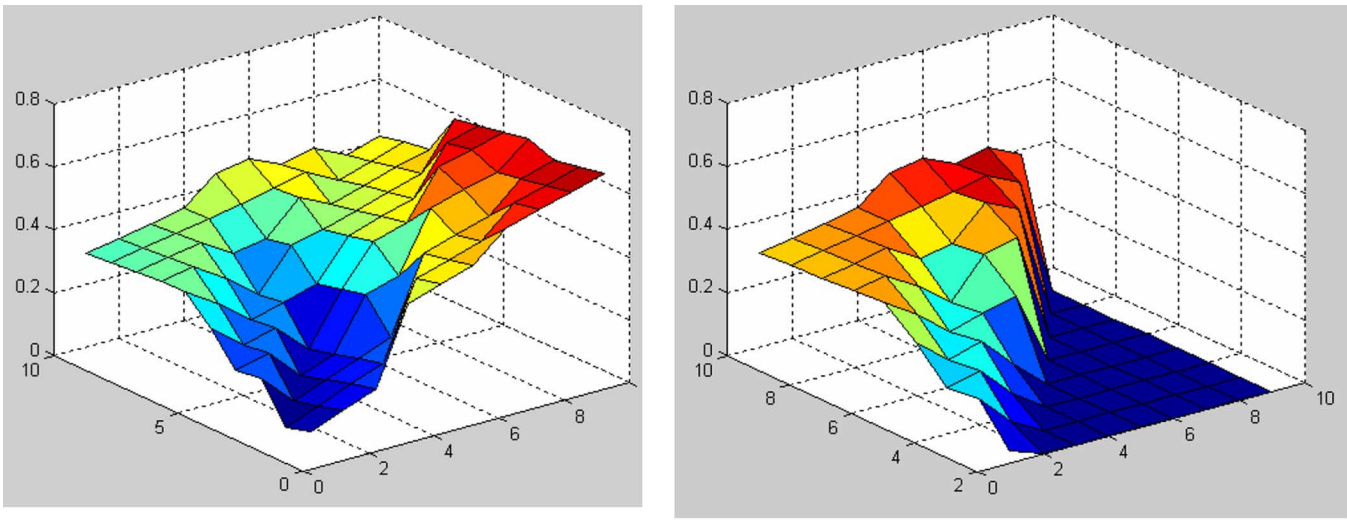


Fig. 5. Solution space of the TIN shown in Fig. 4. (a) Original solution space. (b) Constrained solution space.

TABLE III
PARAMETERS OF THE EA

Representation: Phenotype
Parent Selection: Uniform Stochastic
Survival Selection: Binary Tournament
Mutation: Bitflip / Delta x
Mutation Rate: 0.2
Crossover: one point
Population Size: 10
Number of Generations: 20

TABLE IV
FOUR BEST SOLUTIONS FOR TIN OF FIG. 4

A	time(A)	B	time(B)
True	9	True	5
True	9	True	6
True	8	True	5
True	9	True	4

be executed before B. Using the constructs of the constraint specification language, the constraint can be written as

$$\text{Before} \quad A \quad B \quad 0.$$

The reduced solution space (based on the AUC metric) is shown in Fig. 5(b). When the ECAD-EA methodology is run on this modified solution space, it generates COAs that maximize the AUC metric for event H's probability profile under the given constraint. The top four COAs are shown in Table V. Similar to the previous case, the solutions produced by the EA are centered around the global maximum of Fig. 5(b).

The previously discussed toy example reveals many of the characteristics of the ECAD-EA methodology. It should be mentioned, however, that for large TIN models, the ECAD-EA methodology takes some amount of time before it converges to a solution. Furthermore, there is always a possibility that the EA may converge to local optima. It is therefore desirable

TABLE V
FOUR BEST SOLUTIONS WITH CONSTRAINED SOLUTION SPACE

A	time(A)	B	time(B)
True	8	True	10
True	7	True	9
True	6	True	9
True	4	True	9

to identify schemes that can increase the performance of the ECAD-EA methodology. This objective can be achieved if, instead of initializing the EA randomly, some prior knowledge is provided to the algorithm. One such heuristic is presented in this paper that attempts to solve the problem of the best or close-to-best sets of actions in ordinary INs. The solutions obtained through the heuristics are then provided as an input to the ECAD-EA methodology, which enhances its performance. Section V briefly explains the heuristics, named set of actions finder (SAF), and how it guides the search of the EA.

V. SAF ALGORITHM AS A PRESTEP TO THE ECAD-EA METHODOLOGY

The SAF algorithm [16] was originally proposed as a heuristic approach to solve the problem of the best set of actions determination in ordinary INs. It requires polynomial (in terms of the number of actionable evens) searches that are lower than the exponential number of searches required by the exhaustive search but higher than the linear number of searches required by sensitivity analysis (SA) [31], [33], [41]. The SA looks at how sensitive an effect is with respect to the actionable events when actions are considered one at a time. The problem with this approach is that the interaction among the actionable event is not always synergistic because of the nonlinearities encoded in an IN. Thus, any decision that is based on the individual impact of the actionable events on the effect might be misleading. The other alternative that is computationally expensive

TABLE VI
SAF ALGORITHM

Given ***A, E, S, t***
where ***A*** = Set of Actions
S = Set of Selected Actions
E = Desired Effect
t = Threshold

1. Initialize ***S*** = null.
2. $\forall$ x Set P(x) = 0, where x $\in$ ***A***.
3. Compute P(E). Set P'(E) = P(E).
4. Set P(I) = 1 where I $\in$ ***A***
   a. $\forall$ J Set P(J) = 0 where J $\in$ ***A*** \ {I}
   b. Compute P(E).
   c. Set Diff(I) = P'(E) – P(E).
   Iterate for all members of A.
5. Select the highest Diff(I) obtained in Step 4.
   a. If Diff(I) > 0 OR if the corresponding P(E) is greater than the threshold t
      i. Remove I from ***A***.
      ii. Insert I into ***S***.
      iii. Set P'(E) = P(E).
      iv. Go to Step 4.
   b. Else Stop.

requires doing an exhaustive search over the possible states of the actionable events. If there are n actionable events, then it would require running the model an exponential number (2^n) of times. Even for INs having a moderate number of actionable events, this is not a desirable alternative. The SAF algorithm overcomes the limitation of both the SA and the exhaustive search by capturing the combined impact of actionable events on a desired effect in polynomial time. The proposed algorithm uses a greedy approach to identify the best (or close-to-best) sets of actions. The algorithm starts with a single action that, when considered individually, causes the highest increase in the probability of the objective being true. This is followed by the selection of a second action from the remaining set of actions that, together with the first action, cause the highest increase in the probability of the objective node. Other actions are added iteratively in a similar manner. The process stops at a point where the inclusion of an action causes the probability of the objective node to decrease and to fall below the given probability threshold. The algorithm is presented in Table VI.

It is worth mentioning that because of its similarity to the hill-climbing search, there is always a likelihood that SAF may stuck at a local optimum. Haider *et al.* [17] have presented an empirical study to analyze the performance of the SAF algorithm over thousands of IN models. They have also shown that the problem of the best set of actions determination can be formulated as an instance of *mixed-integer nonlinearly constraint optimization* (or sometimes termed mixed-integer nonlinear programming, MINLP) problems. Their empirical study also compares the performance of the SAF algorithm to that of MINLP and SA.

A. SAF and ECAD-EA

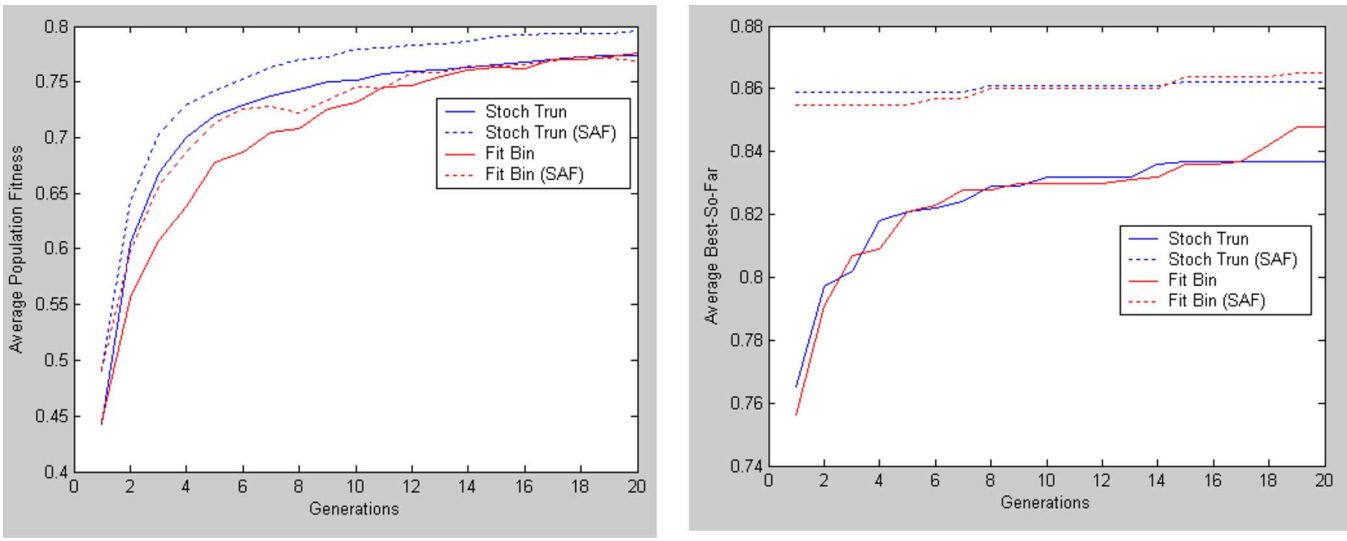
This section discusses how the performance of the ECAD-EA methodology is improved when it is initialized with the solutions obtained by first running the SAF algorithm. Two comparatively large INs are used for the study. The first TIN was originally discussed in [41] and was further analyzed in [44]. It describes the political crisis that occurred in East Timor during the final years of the previous decade. The TIN attempts to answer the question of how rebels in that part of the world can be persuaded to adopt peaceful means to achieve their demands. The model was developed as a prototype for the Decision Support System for Coalition Operations developed by SPAWAR Systems Center-San Diego to support the Operations Planning Team of the Commander in Chief, U.S. Pacific Command.[2] There are 82 nodes in the model, out of which 39 nodes represent actionable events. Suppose that each action has binary states and that all the actions need to be taken within a window of 1 to 20 time units (window of opportunity). Assuming integer time, there are $2^{39} \times 20^{39}$ (1 with 62 zeros after it) possible COAs. It is quite evident that manual trial-and-error-based methods could not exhaust all the possibilities and thus are likely to result in selecting a suboptimal COA, which is only the best of few possibilities explored by an analyst. The second TIN used in this paper is described by Haider and Levis [15]. The model was developed to capture the events and the associated uncertainties in combating insider threat. There are ten nodes in the TIN that represent actionable events, and overall, there are 21 nodes in the network.

The impact of SAF on the performance of ECAD-EA is analyzed by running the ECAD-EA methodology with and without SAF input on both models. For instance, when the SAF algorithm is applied to the untimed East Timor IN, the top 20 solutions produced by the algorithm are recorded. The solutions are then provided to ECAD-EA for initializing the population of EA. The number 20 is simply selected because the population size of the EA in this experiment is set to 20; for a different population size, this number can be adjusted accordingly. If there are fewer than 20 solutions (or whatever the population size of the EA is) produced by the SAF algorithm, then the remaining solutions are initialized randomly. As described earlier, a particular individual in a population has the following form:

$$\langle \text{Action } 1, \text{time}(\text{Action } 1), \text{Action } 2, \ \text{time}(\text{Action } 2), \ldots, \text{Action } n, \text{time}(\text{Action } n) \rangle .$$

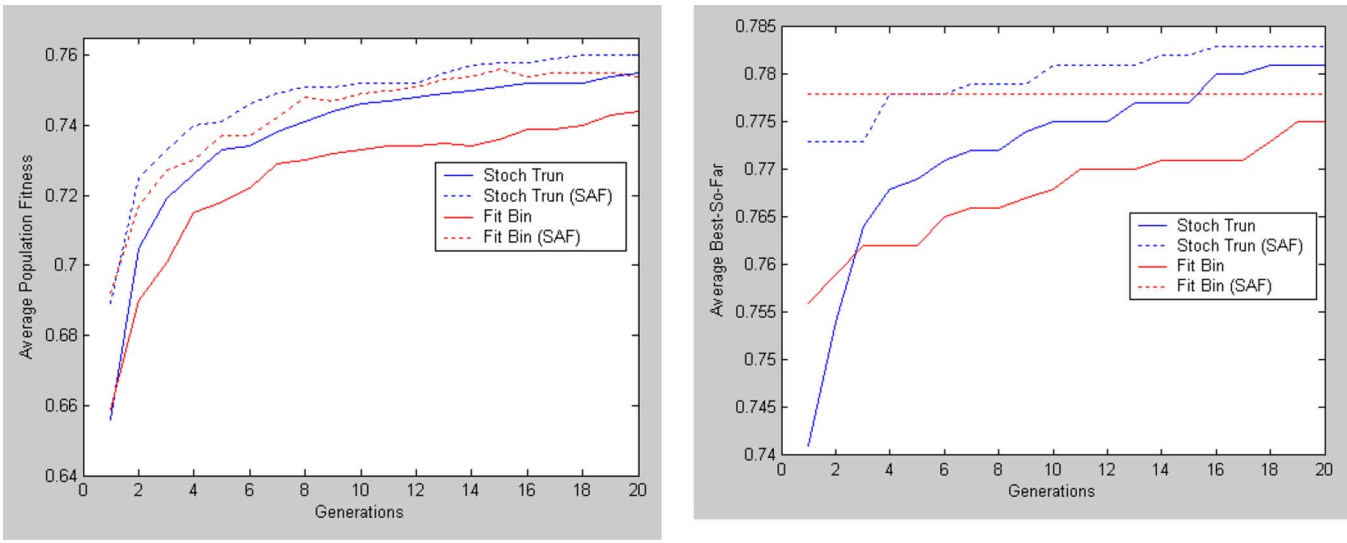
The odd elements (or genes) represent the state of actions, whereas the even elements (genes) represent the time at which the actions are made in that particular state. Thus, the solutions obtained from the SAF algorithm only initialize the state genes; genes representing the time are still randomly initialized. If there are causal constraints among actionable events, as specified by a system analyst, then only those solutions are provided as an input that are consistent with the given constraints. For this experiment, however, it is assumed that there are no causal constraints specified by an analyst. Two different combinations of parent and survival selection schemes are used in this experiment. Combination 1 has *fitness proportional* as parent selection and *binary tournament* as survival selection, whereas

[2]A detailed description and the graphical representation of the model can be found in [41].



(a) (b)

Fig. 6. (a) Average population fitness and (b) best-so-far curves of EAs for the East Timor model.



(a) (b)

Fig. 7. (a) Average population fitness and (b) best-so-far curves of EAs for the insider threat model.

combination 2 has *uniform stochastic* as parent selection and *truncation* as survival selection. Each of these combinations is run with and without SAF input on both TIN models. The population and the generation size are both set to 20.

Fig. 6(a) shows the average population fitness of different EAs for the East Timor model. Each point in the curve represents a mean value obtained after repeating the experiment ten times. It can be seen in the figure that when SAF inputs are provided to both combinations, there is a significant improvement in the average population fitness value. The *uniform stochastic* + *truncation* combination with SAF input has performed better than all the other combinations. Fig. 6(b) shows the average best-so-far curve of different EAs. The figure shows a big improvement in the average best-so-far curve when SAF input is provided. In fact, the average best-so-far value produced by the EAs with the SAF input after the first generation is considerably better than the one produced by the EAs (without the SAF input) after 20 generations. This ability to quickly converge to local optima can be very handy in time-critical conditions. When a DM does not have the luxury of running the EA for a large number of generations, he can follow the two-step process of first running the SAF algorithm and then providing the solutions to the EA and run it for very few generations.

Fig. 7(a) and (b) shows the average population fitness curves and the average best-so-far curves, respectively, for the insider threat model. As is the case with the East Timor model, the EAs with the SAF input perform much better than the ones without the SAF input. It is possible, as shown in the figure, that a different combination of EA without the SAF input can perform

better than some other combination of EA with the SAF input. For instance, in Fig. 7(b), the combination *uniform stochastic + truncation* without the SAF input has a better best-so-far curve than the *fitness proportional + binary tournament* combination after the 16th generation. However, all the experiments have suggested that when the comparison is made between EAs having similar combinations of parent and survival selection schemes, then EAs with the SAF input perform better than the ones without the SAF input.

VI. Conclusion

This paper has presented a methodology to identify effective COAs in a dynamic uncertain situation. Several metrics are suggested that can be used for automatic evaluation and comparison of COAs. A constraint specification language is also presented that aids a system analyst in specifying certain types of temporal and causal constraints among the actionable events. The presented methodology not only provides a single best solution but also provides several alternate solutions that are close enough to the best solution. In addition to providing more options, these alternate solutions aid a DM in understanding the impact of actions' time of executions on the desired effect. To further improve the performance of the ECAD-EA methodology, the use of a heuristic has also been suggested in this paper.

There are several research directions that stem from this research. A TIN has nodes that represent uncertain events. They are called uncertain because not enough information is available about these events at the start of a campaign. In many situations, it is often the case that the states of these uncertain events become known with certainty during the course of a campaign. This new information may be either favorable to the mission's objective or unfavorable to it. In both cases, it is desirable to revise the current strategy after incorporating this additional information and the set of actions taken so far in the campaign. In the Bayesian terminology, the incorporation of new information into a probabilistic network is termed belief updating. As mentioned earlier, belief updating in a general class of static probabilistic network is NP-hard. For probabilistic networks that deal with time, the problem becomes more complex. One of the future research directions is to develop/apply approximate belief updating techniques for dynamic probabilistic networks that can run in real time.

The ECAD-EA methodology, as explained in this paper, produces a single time stamp associated with an actionable event, i.e., at what time the decision regarding the state of an action needs to be taken. Thus, in solutions produced by the EA, all the actions are only taken once. There may be situations, however, when an action is required to be repeated several times. The repetition may mean that the state of an action keeps toggling during different time intervals or the action goes back to an uncertain state after each instantiation. Another future research direction is to extend the ECAD-EA methodology so that it can produce multiple time stamps associated with a single action. This would require having variable-length genes in contrast to the fixed-length genes presented in this paper. Some additional constraints, such as how long an action can stay in a particular state, also need to be provided to the ECAD-EA methodology.

This paper has demonstrated the application of the ECAD-EA methodology for a single objective/desired effect. Another future research direction is to extend the methodology for multiple objectives/desired effects. An extensive body of literature is available for obtaining a pareto-optimal solution. A careful analysis of available techniques needs to be done before a particular method is chosen.

Acknowledgment

The authors would like to thank the anonymous reviewers, whose comments substantially improved the quality of this paper.

References

[1] J. M. Agosta, "Conditional inter-causally independent node distributions, a property of noisy-OR models," in *Proc. 7th Conf. Uncertainty Artif. Intell.*, 1991, pp. 9–16.

[2] J. Cano, L. Hernández, and S. Moral, "Importance sampling algorithms for the propagation of probabilities in belief networks," *Int. J. Approx. Reason.*, vol. 15, no. 1, pp. 77–92, Jul. 1996.

[3] K. C. Chang, P. E. Lehner, A. H. Levis, S. A. K. Zaidi, and X. Zhao, *On Causal Influence Logic*. Fairfax, VA: George Mason Univ., Dec. 1994.

[4] J. Cheng, "Efficient stochastic sampling algorithms for Bayesian networks," Ph.D. dissertation, Univ. Pittsburgh, Pittsburgh, PA, 2001.

[5] J. Cheng and M. J. Drudzel, "An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks," *J. Artif. Intell. Res.*, vol. 13, pp. 155–188, 2000.

[6] G. F. Cooper, "The computation complexity of probabilistic inference using Bayesian belief networks," *Artif. Intell.*, vol. 42, no. 2/3, pp. 393–405, Mar. 1990.

[7] K. A. DeJong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, MI, 1975.

[8] K. A. DeJong, *Evolutionary Computing*. Cambridge, MA: MIT Press, 2005.

[9] M. J. Drudzel and M. Henrion, "Intercausal reasoning with uninstantiated ancestor nodes," in *Proc. 9th Conf. Uncertainty AI*, 1993, pp. 317–325.

[10] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. Chichester, U.K.: Wiley, 1966.

[11] W. T. Freeman and Y. Weiss, "On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 736–744, Feb. 2001.

[12] R. Fung and K. C. Chang, "Weighting and integrating evidence for stochastic simulation in bayesian networks," in *Proc. 5th Conf. Uncertainty Artif. Intell.*, 1990, pp. 209–219.

[13] R. Fung and B. D. Favero, "Backward simulation in Bayesian networks," in *Proc. 10th Conf. Uncertainty Artif. Intell.*, San Francisco, CA, 1994, pp. 227–234.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[15] S. Haider and A. H. Levis, "On finding effective courses of action in a complex situation using evolutionary algorithms," in *Proc. 10th Int. Command Control Res. Technol. Symp.*, Washington, DC, 2005.

[16] S. Haider, A. K. Zaidi, and A. H. Levis, "A heuristic approach for best set of actions determination in influence nets," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, Las Vegas, NV, 2004, pp. 600–605.

[17] S. Haider, A. K. Zaidi, and A. H. Levis, *Identification of Best Sets of Actions in Influence Nets*. Forth Coming.

[18] T. Haiying, Y. N. Levchuk, and K. R. Pattipati, "Robust action strategies to induce desired effects," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 34, no. 5, pp. 664–680, Sep. 2004.

[19] D. E. Heckerman, A. Mamdani, and M. P. Wellman, "Real-world applications of Bayesian networks," *Commun. ACM*, vol. 38, no. 3, pp. 24–26, Mar. 1995.

[20] M. Henrion, "Propagation of uncertainty by probabilistic logic sampling in Bayesian networks," in *Proc. Uncertainty Artif. Intell.*, 1988.

[21] R. Howard and J. Matheson, "Influence diagrams," in *Readings on the Principles and Applications of Decision Analysis*, vol. 2, R. Howard and

J. Matheson, Eds. Menlo Park, CA: Strategic Decisions Group, 1981, pp. 721–762.
[22] R. A. Howard and J. E. Matheson, "Influence diagram retrospective," *Decis. Anal.*, vol. 2, no. 3, pp. 144–147, Sep. 2005.
[23] R. A. Howard, J. E. Matheson, M. W. Merkhofer, A. C. Miller, and D. W. North, "Comment on influence diagram retrospective," *Decis. Anal.*, vol. 3, no. 2, pp. 117–119, Jun. 2006.
[24] K. Jensen, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Volumes 1, 2, and 3. Basic Concepts, Monographs in Theoretical Computer Science*. Berlin, Germany: Springer-Verlag, 1997.
[25] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 219–230, Feb. 1998.
[26] K. P. Murphy, "Dynamic Bayesian networks: Representation, inference and learning," Ph.D. dissertation, Univ. California, Berkeley, Berkeley, CA, 2000.
[27] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proc. 15th Conf. Uncertainty Artif. Intell.*, 1999, pp. 467–475.
[28] R. E. Neapolitan, *Learning Bayesian Networks*. Englewood Cliffs, NJ: Prentice-Hall, 2003.
[29] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1987.
[30] J. A. Rosen and W. L. Smith, "Influence net modeling with causal strengths: An evolutionary approach," in *Proc. Command Control Res. Technol. Symp.*, Monterey, CA, 1996, pp. 699–708.
[31] J. A. Rosen and W. L. Smith, *Influencing Global Situations: A Collaborative Approach*. US Air Force Air Chronicles, 1996.
[32] J. A. Rosen and W. L. Smith, "Influence net modeling for strategic planning: A structured approach for information operations," *Phalanx*, vol. 33, no. 4, pp. 6–7, Dec. 2000.
[33] J. A. Rosen, W. L. Smith, E. S. Smith, and M. A. Maldony, "Planning with influence net modeling: An architecture for distributed, real-time collaboration," in *Proc. MORS*, 1998.
[34] R. D. Shachter, "Evaluating influence diagrams," *Oper. Res.*, vol. 34, no. 6, pp. 871–882, Nov./Dec. 1986.
[35] R. D. Shachter and M. Poet, "Simulation approaches weighting and integrating evidence for stochastic simulation in Bayesian networks," in *Uncertainty in Artificial Intelligence*, vol. 5, M. Henrion, D. Shachter, L. N. Kanal, and J. F. Lemmer, Eds., 1990.
[36] H.-P. Schwefel, *Evolution and Optimum Seeking*. Hoboken, NJ: Wiley, 1995.
[37] H.-P. Schwefel, "Evolutionsstrategie und numerische Optimierung," Ph.D. dissertation, Dept. Process Eng., Tech. Univ. Berlin, Berlin, Germany, 1975.
[38] L. W. Wagenhals, "Course of action development and evaluation using discrete event system models of influence nets," Ph.D. dissertation, School Inf. Technol. and Eng., George Mason Univ., Fairfax, VA, 2000.
[39] L. W. Wagenhals and A. H. Levis, "Course of action development and evaluation," in *Proc. Command Control Res. Technol. Symp.*, 2000.
[40] L. W. Wagenhals and A. H. Levis, "Modeling effects based operations in support of war games," in *Proc. 15th Int. Symp. Aerosp./Defense Sens.*, 2001.
[41] L. W. Wagenhals, T. J. Reid, R. J. Smillie, and A. H. Levis, "Course of action analysis for coalition operations," in *Proc. 6th Int. Command Control Res. Technol. Symp.*, 2001.
[42] L. W. Wagenhals, I. Shin, and A. H. Levis, "Creating executable models of influence nets with coloured Petri nets," *Int. J. Softw. Tools Technol. Transf.*, vol. 2, no. 2, pp. 168–181, 1998.
[43] L. W. Wagenhals and L. K. Wentz, "New effects-based operations models in war games," in *Proc. 8th Int. Command Control Res. Technol. Symp.*, Washington, DC, 2003.
[44] A. K. Zaidi, L. W. Wagenhals, and S. Haider, "Assessment of effects based operations using temporal logic," in *Proc. 10th Int. Command Control Res. Technol. Symp.*, Washington, DC, 2005.

Sajjad Haider received the B.Sc. (Hons.) degree in statistics and the M.Sc. degree in computer science from the University of Karachi, Karachi, Pakistan, in 1997 and 1999, respectively, and the M.S. degree in information systems and the Ph.D. degree in information technology from George Mason University, Fairfax, VA, in 2002 and 2005, respectively.

He is currently a Postdoctoral Fellow with the System Architectures Laboratory, Department of Electrical and Computer Engineering, Volgenau School of Information Technology and Engineering, George Mason University. He is also a Consultant with the Predictive Analytics Group, Federal National Mortgage Association (Fannie Mae). His areas of interest include decision sciences, probabilistic belief networks, integration of time and uncertainty, evolutionary algorithms, and optimization.

Dr. Haider was a recipient of the 2004 Gary F. Wheatley Best Student Award at the Command and Control Research and Technology Symposium and the Best Paper Award from the International Council of Systems Engineering for a paper published in the *Systems Engineering Journal* over the six-year period of 1998–2003.

Alexander H. Levis (S'67–M'68–SM'80–F'87–LF'06) received the A.B. degree in mathematics and physics from Ripon College, Ripon, WI, in 1963 and the B.S., M.S., M.E., and Sc.D. degrees in mechanical engineering, with a specialization in the area of control systems, from the Massachusetts Institute of Technology (MIT), Cambridge, in 1965, 1965, 1967, and 1968, respectively,

He is currently a Professor of electrical, computer, and systems engineering and the Head of the System Architectures Laboratory, Department of Electrical and Computer Engineering, Volgenau School of Information Technology and Engineering, George Mason University, Fairfax, VA. From 2001 to 2004, he served as the Chief Scientist of the U.S. Air Force. He taught at the Polytechnic Institute of Brooklyn, Brooklyn, NY (1968–1973), headed the Systems Research Department, Systems Control, Inc., Palo Alto, CA (1973–1979), was a Senior Research Scientist with the Laboratory for Information and Decision Systems, MIT (1979–1990), and moved to George Mason University in 1990, where he headed twice the Department of Systems Engineering. For the last 15 years, his areas of research have been system architectures, including organization architecture design and evaluation, adaptive architectures for command and control, and methodologies for architecture comparison and evaluation.

Dr. Levis is a Fellow of the American Association for the Advancement of Science, a Fellow of the International Council on Systems Engineering, and an Associate Fellow of the American Institute of Aeronautics and Astronautics. He is a past President of the IEEE Control Systems Society. He was thrice a recipient of the Exceptional Civilian Service Medal from the Air Force (1994, 2001, and 2004) for his contributions as a member of the Air Force Scientific Advisory Board and as the Chief Scientist. He was also a recipient of the Third Millennium Medal from IEEE.