Plan Simulation Using Bayesian Networks*

Moisés Goldszmidt and Adnan Darwiche
Rockwell Science Center
444 High Street, Suite 400
Palo Alto, CA 94301, U.S.A.
{moises, darwiche}@rpal.rockwell.com

Abstract

This paper describes part of Rockwell's contribution to Phase I of ARPA's Planning Initiative (ARPI). Specifically, we describe the representation language and a set of algorithms that constitute the core of a Plan Simulation and Analysis tool (PSA). The main objective of the PSA is to provide capabilities for the testing and evaluation of sequences of actions in domains characterized by unavoidable uncertainties and difficult trade-offs between resources and objectives. The representation language, called action networks, is a semantically well founded framework for reasoning about actions and change under uncertainty based on probabilistic Bayesian networks. Action networks add primitives to Bayesian networks to represent canonical models of time-dependencies, and controllable variables to represent agents manipulations of the domain. In addition, action networks allow different methods for quantifying the uncertainty in causal relationships, which go beyond traditional probabilistic quantification. Inferences are performed via a set of algorithms for approximate computation of belief update. These algorithms allow the user to trade-off computational time for accuracy of the answer, and can be also applied in an *anytime* fashion.

1 Introduction: Plan Simulation

The work we report in this paper is directed towards creating a tool, called Plan Simulator and Analyzer (PSA), for supporting the functionality portrayed in Figure 1. Here, a human planner is confronted with a situation, an objective, and a set of courses of action or plans which are believed to achieve the objective. The human planner, is looking for assistance in simulating the behavior of each of these plans. This simulation entails testing the plans against different conditions in the domain (where the plans will be executed) in order to establish their degree of success, compute costs of different portions of each plan, and identify dependencies upon the relevant parameters. The final objective is to make choices regarding which plan to adopt according to pre-established criterion.

There are mainly three reasons as of why such a tool becomes indispensable in domains such as crisis management operations (both military and civilian), economic forecasting and business planning:

1. In these domains assuming a worst case scenario in order to assess the impact of an adverse outcome on the plan is not acceptable. Such a strategy leads to inflexible plans with possible unacceptable economical and political costs.
2. These domains are characterized by unavoidable uncertainties and difficult trade-offs about resources and objectives.
3. Finally, the situations in these domains are dynamic, constantly changing and furthermore, there is little hope of gathering enough information to be able to formulate a complete picture. Hence, the situation and uncertainties at the time of generating a plan, will be very different than those at the time of deploying the plan.

Example. Consider a non-combatant evacuation operation. The objective is to transport civilians from an assembly area in one city to the airport in another city (in a hostile country). One course of action establishes that if air support is provided, then there will be no need to send additional troops to protect the transportation of civilians. Additional troops imply additional logistics and support and may become expensive. Suppose that the transportation command guarantees air support during the second day of the operation, but with 70% can guarantee air support for the rest of the operation.

Given this information, a military planner may wish to find out the probability that additional troops may be needed, compute the expected cost, and the impact that the availability of troops will have on the success of the operation. Success in this case being defined as the probability of safe arrival of the civilian contingent to the target city.

In order to provide this type of functionality it is clear that the tool should be able to represent the dependencies among events in the domain of interest, and their uncertainty.[1] To this end, we propose

*This paper is a slightly modified version of a paper of the same title ©1995 IEEE. Reprinted, with permission, from *Proceedings of The Eleventh Conference on Artificial Intelligence for Applications*, Los Angeles, CA, February 20–22, 1995, pages 155-161.

[1] This should be contrasted with the requirements on a tool for plan generation, for example, where the emphasis is on action indexing.

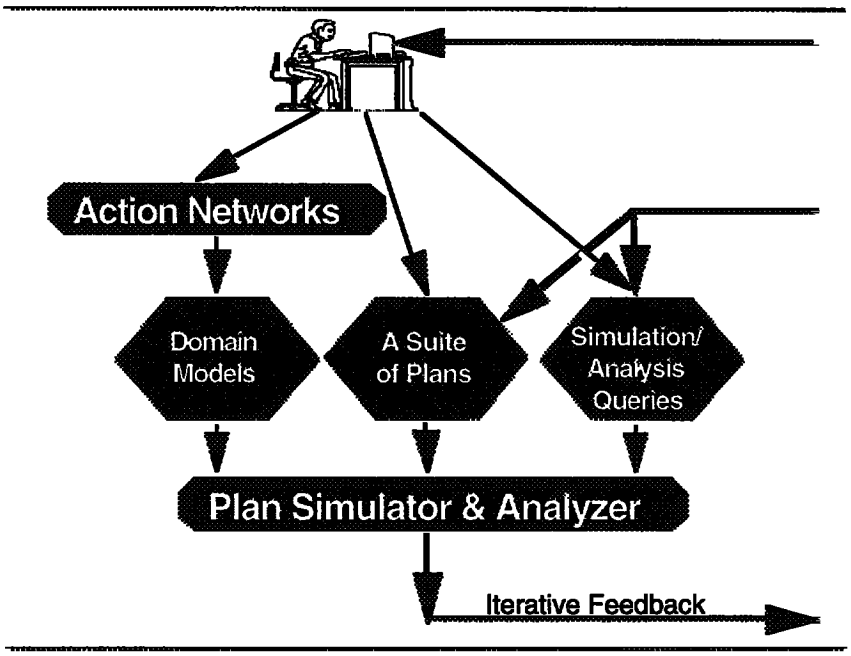

Figure 1: *Schematic view of the information flow between a planner and the plan simulation and analyzer tool.*

a modeling language based on probabilistic Bayesian networks (BN) [17]. The language, called action networks, extend the basic representation mechanisms in BN with primitives to represent time, actions, and to quantify uncertainty with different levels of abstraction, including numerical probabilities, order of magnitude probabilities or rankings [13, 14, 9], and logical statements [4, 3]. This last capability, which was initially used for representational purposes only, turned out to be instrumental in some of the algorithms in the inference engine of the tool [12]. Action networks are formally described in [8] and are reviewed in Section 2.

With respect to the inference capabilities of PSA we have concentrated our efforts on tasks of belief update. That is, given a set of initial conditions and sequence of actions, compute the updated degree of belief in the events and propositions of interest. In order to deal with issues of scalability and speed, we have devised a set of approximate algorithms for belief update that trade-off time for accuracy in the computation of uncertainty. The approximation is in terms of a lower bound on the uncertainty with precise guarantees on the error. The idea is that in a large number of instances, approximate answers provide enough information for decision making, and the user is willing to deal with these approximate answers specially if it translates into a major speed up in performance. Thus, for example, a lower bound of 35% on the event that additional troops will have to be deployed (in order to guarantee the success of the operation), may be enough to force a modification of the original plans in order to take these additional troops into consideration. In PSA, this trade-off between time and accuracy can be controlled via a parameter ϵ that sets a threshold for approximating numerical probabilities and ignoring events with small probability mass. We remark that it is easy to modify these algorithms to exhibit an anytime behavior [1, 12]. These algorithms are formally explained in a set of papers [7, 6, 12], and are summarized in Section 3. We point out that the parameter ϵ controls the level of abstraction in the representation of the uncertainty (from numerical probabilities to order of magnitude probabilities or rankings).

Section 4 describes the current status of the system and future work. We remark that the same capabilities needed for simulation can be used to monitor the execution of a plan once it is deployed.

2 Modeling the Domain and Representing Plans

The emphasis in the representation component of PSA is on modeling the relations between different variables, standing for events and persistences in the domain where different plans are to be deployed. The reason is that it is against changes in these relations and their uncertainties that plans are going to be tested and simulated. PSA uses action networks [8] as a language to represent these relations, their uncertainty, and an agent ability to manipulate this domain through actions.

The specification of an action network requires three components: (1) the causal relations in the domain with a quantification of the uncertainty in these relations, (2) the set of variables that are "directly" controllable by actions (with the variables that constitute

their respective preconditions), and (3) variables that persist over time, which we call *persistences* in this paper, and variables that do not persist over time, which we call *events*. An example of the structure of an action network modeling a situation for the evacuation of civilians in a noncombatant operation is shown in Figure 2.

A directed acyclic graph is used to specify the causal relations in the domain and their interdependencies. A directed arc between two nodes in these networks denotes direct influence. Thus, for example, in Figure 2, the nodes *rebel threat in Delta* and *civilians on their way to Delta* have a direct influence on *successful arrival to Delta*. To quantify the uncertainty in these relations, the user must provide a set of local tables or matrices attached to every node in the graph. In this respect action networks are no different than BN [17] except for the fact that action networks bring the ability to perform symboiic and qualitative reasoning under uncertainty. BN have been probabilistic in that the quantification of the relations and interactions must be done by providing numerical probabilities. Action networks allow the quantification of these relations and interactions using, in addition to probabilistic numbers, orders-of-magnitude probabilities (also known as epsilon probabilities and κ-rankings of belief [13, 14, 9]), and logical arguments [4, 10]. From the knowledge representation point of view, this ability provides additional flexibility to the user when building a model of the domain: even if the exact probabilistic parameters are not known, knowledge about relative rankings of likelihood can be encoded and sound inferences can be performed. Similarly, whenever uncertainty is not an issue, logical information can be used, and network structure can be utilized to reason about the effect of actions [10]. Further details on the representation of a domain using networks are provided in Section 2.1.

In addition to providing extra flexibility in the representation of a domain, the capability of using order-of-magnitude probability translates into faster algorithms for performing inferences (see Section 3).

Actions are represented as direct manipulations of variables in the domain. Thus, for example, the ability to perform the action of sending troops from one city to another, is represented as the ability to directly control the value of the *troops location* variable of Figure 2 by setting it to the appropriate value. Section 2.2 describes the semantics of this representation in terms of changes in the probability distribution representing the interactions in the domain (also see [13, 18, 19] for further details of this representation). More complex actions, such as those requiring preconditions, or those that are conditional on other events, are easily built on top of the representation described above. Graphically (and also in the graphical interface of PSA), controllable variables have a control nodes "▼" pointing into them (e.g, *troops location* in Figure 2). As mentioned above, the specification of which variables are controllable in the domain is part of the input to an action network.

In this framework a plan is represented as a sequence of actions; that is, a specification of which variables are being controlled and the *time instances* when this manipulation occurs. Note that the same representation

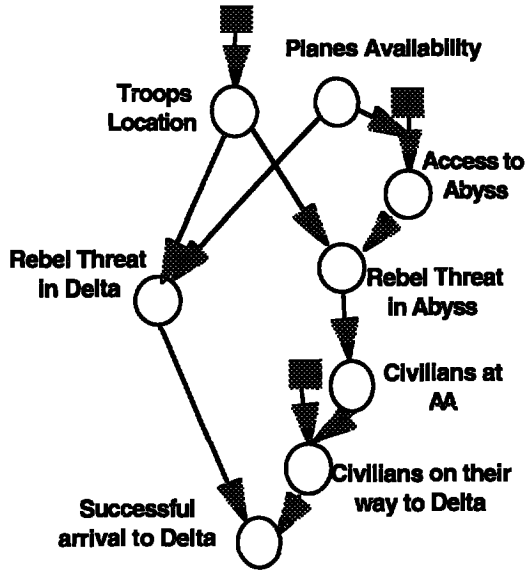

Figure 2: *An example of the structure of an action network. This network is expanded over time assuming canonical models of time behavior, and special constructs are added for the controllable variables.*

of the domain, that is, the same action network, can then be used to represent different plans as different sequences of actions are specified.

Once sequences of actions are introduced, we are forced to model some notion of time. There are basically two options: we either require the user to specify the behavior of the domain over time, or we assume canonical models of time behavior and allow the specification of a static domain, that can be then automatically expanded. We follow the second strategy.

In order to expand a static network into a temporal network, action networks appeal to three assumptions. First, time is regarded to be discrete. The second assumption states that causal relations between variables at a specific time point are similar to those explicated in the action network. That is, if e has causes $c_1, \ldots, c_n$ in the action network, it will have these causes at every time point. The third assumption relates to the fact that most of the temporal behavior needed in the simulation can be captured through a set of canonical models.

One of these models is temporal persistence. It says that the state of the system modeled by an action network persists over time (with a certain degree of uncertainty) in the absence of external intervention. Thus, for example, according to Figure 2, once the troops are sent to a particular city, they are assumed to remain there until a new action is specified. The details of an expansion based on assumptions of persistence can be found in [8]. Other models of time behavior such as variables whose uncertainty decay (or increase) over time, have already been investigated by the authors (and others, see for example [15]) but have not yet

been implemented in our framework.[2]

Sections 2.1 and 2.2 describe in further details on the representation of a domain, and the representation of actions using networks.

2.1 Network Representations

We briefly review some of the key concepts behind representing uncertain knowledge using networks. A network consists of a directed-acyclic graph Γ and a quantification Q over Γ. Nodes in Γ correspond to domain variables, and the directed edges correspond to the causal relations among these variables. The quantification Q encodes the uncertainty in these relations.

We denote the set of parents of a node e in a network by $\pi(e)$. $\vec{\pi}(e)$ will denote a state of the propositions that constitute the parent set of e. The set conformed by e and its parents $\pi(e)$ is usually referred to as the causal family of e. In Figure 2 the nodes *rebel threat in Delta* and *civilians on their way to Delta* represent the causal family of *successful arrival to Delta*. This network in conjunction with its quantification represents the belief in a successful arrival given the danger in the destination, and the amount of civilians on their way.

The quantification of Γ is over the families in the network representing the uncertainty in the causal influences between $\pi(e)$ and e. The user provides a local specification of the uncertainty[3] from which a global and coherent state of belief can be reconstructed using the Markovian assumptions of independence embedded in the network structure (see [17] for details). Thus, networks provide a compact specification of a state of belief. Consider a network with nodes $x_1, \ldots, x_n$, and assume that the quantification provided is in terms of numerical probabilities. The user provides the set of conditional probabilities $Pr(x_i|\pi(x_i))$[4] for each x_i $(1 \leq i \leq n)$, and a state of belief $Pr(\mathbf{x_1}, \ldots, \mathbf{x_n})$[5] is computed using the following equation:

$$Pr(\mathbf{x_1}, \ldots, \mathbf{x_n}) = \prod_{1 \leq i \leq n} Pr(\mathbf{x_i}|\vec{\pi}(x_i)) \qquad (1)$$

In addition to numerical probabilities, action networks allow other ways to encode uncertainty that do not require an exact and complete probability distribution. Two recent approaches are the κ-calculus where uncertainty is represented in terms of plain beliefs and degrees of surprise [13],[6] and argument calculus where uncertainty is represented using logical sentences as arguments [4]. These approaches are regarded as abstractions of probability theory since they retain the main properties of probability including Bayesian conditioning. Similar equations to Eq. 1 can be obtained for the κ-calculus and for the argument calculus, when the quantification of the network is done through ranks of belief and logical arguments.

In addition to providing an efficient and compact way of representing uncertainty, networks provide a guide for inference algorithms that compute changes in the state of belief that arise as the consequence of both actions and observations. The case of observations is handled through Bayesian conditioning (and its equivalents in the κ-calculus and the argument calculus). For the case of actions we follow the proposal in [13] which is briefly reviewed in the next section.

2.2 Actions

The proposal in [13] treats primitive actions as external direct interventions that deterministically set the value of a specific proposition in the domain. In probabilities, this is accomplished by a simple modification of the probability distribution $Pr(e|\pi(e))$ quantifying the causal relation between a controlled node e, and its causes $\pi(e)$. We include an additional parent do_e which takes the same values as e in addition to the value *idle*.[7] Thus, the new parent set of e will be $\pi(e) \cup \{do_e\}$. The new probability distribution $Pr'(\mathbf{e}|\vec{\pi}(e) \wedge \mathbf{do_e})$ is (see [13, 19] for further details)

$$Pr'(\mathbf{e}|\vec{\pi}(e) \wedge \mathbf{do_e}) = \begin{cases} Pr(\mathbf{e}|\vec{\pi}(e)) & \text{if } \mathbf{do_e} = idle \\ 0 & \text{if } \mathbf{do_e} \neq \mathbf{e} \\ 1 & \text{if } \mathbf{do_e} = \mathbf{e} \end{cases} \qquad (2)$$

Similar expressions can be obtained for the κ-calculus and the argument calculus (see [13, 10]).

More complicated actions, including those requiring preconditions and those that are conditional on other events in the domain can be built from this representation by simple modifications of Eq. 2. To represent preconditions we specify additional cases to Eq. 2 to reflect the intuition that the action do_e will be effective only if the precondition is true; otherwise, the state of a node e is decided completely by the state of its natural causes, that is, excluding do_e and the preconditions of do_e (see also [19]).

Graphically, actions are specified by indicating which nodes in the causal network are controllable and under what preconditions using the node "☛" denoting controllability.[8] These nodes are place-holders for the do_e parents. In Figure 2 for example, *troops location* and *access to Abyss* are controllable propositions. The precondition for controlling the access to Abyss is dependent on the availability of planes.

One advantage of using this proposal as opposed to others, such as STRIPS, is that the approach based on direct intervention relies on the network representation for dealing with the indirect consequences of actions and the related frame problem. In specifying the action regarding the location of the troops, for example, the user need not worry about how this action will affect the state of other related consequences such as the probability of a successful arrival to Delta.

[2] For example, the user may want to specify that the certainty of a *rebel threat* increases over time (Figure 2), or a decaying model such as the one suggested in [15].

[3] Local to each family.

[4] Note that the number of probabilities that need to be specified is much less than the 2^n required for $Pr(x_1, \ldots, x_n)$ (see [17]).

[5] We will represent an instantiated proposition x in boldface $\mathbf{x}$.

[6] Formally, these rankings of beliefs can be viewed as order-of-magnitude abstraction of the probabilistic numbers [14, 9].

[7] Thus, if e is binary $\mathbf{do_e} \in \{true, false, idle\}$.

[8] The square node representation is reminiscent of decision nodes in influence diagrams. The exact relation between controllable nodes and decision nodes is explored in [19] and it is beyond the scope of this paper.

3 Inference

In this approach, plan simulation reduces computationally to reasoning about belief change with a network representation. Although there are a number of well known algorithms for this purpose in the literature [17], these algorithms will perform poorly on networks generated for plan evaluation. The basic problem is the existence of undirected cycles (loops) in the network. If no loops exist, the network is singly-connected (also called a poly-tree) and the computational complexity is linear in the number of nodes and arcs. Otherwise the problem of belief update is known to be NP-complete [2]. Networks generated for plan evaluation tend to have many loops because of the dependencies between variables at one time step and the same variables at previous time steps. Thus, for example, a network with one loop can have more than twenty loops when expanded over three time steps.

There are essentially two methods to deal with networks containing loops: *clustering and conditioning*. In clustering, the network is aggregated and transformed into a singly-connected network of aggregated states. The root of exponential behavior in the clustering method can be ascribed to the computations in the aggregate states. This method is due to Lauritzen and Spiegelhalter [16]. In conditioning, certain variables (forming a cutset of the graph) are instantiated, and due to the set of conditional independences that hold, the network becomes again singly-connected. In essence the method of conditioning corresponds to reasoning by cases, where each case corresponds to a singly-connected network. The root of exponential behavior in this method is the number of cases that have to be considered. The conditioning algorithm is due to Pearl [17].

Our commitment to conditioning methods stems from two factors: (1) Given the results in [3], we know that we can apply conditioning methods to any network independently of its quantification, and (2) we have identified a number of computational techniques for approximating the degree of belief of propositions that are best embedded in the context of a conditioning method.

These techniques are comprised in an algorithm called ϵ-bounded conditioning, which allows a user to trade efficiency for accuracy of computation [7]. The main concept behind ϵ-bounded conditioning is that we can approximate the computation of the degree of belief of a proposition by considering only a subset of all the cases that are necessary under the conditioning method. Furthermore, the answer will be a lower bound on the actual degree of belief, and we can compute the quality of this approximation, by computing the probabilistic mass of the cases ignored. Obviously the quality of the approximate answer computed using this method will increase as we consider more of the relevant cases (those that are highly probable) and ignore the irrelevant ones (those that are very unlikely). We explain this approximation further.

The method of cutset conditioning computes the following sum [20], $Pr(x) = \sum_s Pr(x \wedge s)$, where s ranges over all possible states of a loop cutset. ϵ-bounded conditioning splits members of the above sum into two classes, those exceeding a particular value ϵ and those having a lesser or equal value to ϵ. It then ignores the latter elements of the sum. Thus, for some ϵ, ϵ-bounded conditioning splits the conditioning sum as follows:

$$Pr(x) = \sum_{s, Pr(s) > \epsilon} Pr(x \wedge s) + \sum_{s, Pr(s) \leq \epsilon} Pr(x \wedge s). \quad (3)$$

The algorithm then ignores $\sum_{s, Pr(s) \leq \epsilon} Pr(x \wedge s)$, thus computing:

$$Pr'(x) = \sum_{s, Pr(s) > \epsilon} Pr(x \wedge s). \quad (4)$$

Note that we can judge the quality of its approximations without knowing what the chosen value of ϵ because it provides an upper and a lower bound on the exact probability.

To completely specify ϵ-bounded conditioning, we need to show how to identify cutset states that have probability no more than ϵ. Also, this computation must be faster than the final computation of the uncertainty. Suppose that we have an algorithm such that for each variable X and value x it will tell us whether $Pr(x) \leq \epsilon$. Using such an algorithm, call it ϵ-algorithm, we can identify all cutset states that have probabilities no more than an ϵ. One such algorithm is *Predict* [12]. *Predict* is an approximate but polynomial ($O(E)$ where E is the number of edges in the network) algorithm for computing which variable in the network have extreme degrees of belief. Thus, it is used to compute which of the cases in the conditioning strategy can be ignored because they are very unlikely. It assumes that the uncertainty is represented as rankings of belief or as order of magnitude approximations of the real probability numbers. It then takes advantage of the properties of order-of-magnitude reasoning to perform polynomial-time computations on networks. If the uncertainty in the model is represented with numerical probabilities, we can use the transformation method proposed in [9] to approximate these numbers into rankings or order-of-magnitude probabilities. The *Predict* algorithm and its properties are described in detail in [12].

In addition, ϵ-bounded conditioning takes advantage of another algorithm, named dynamic conditioning, to reduce the effective time needed for computing belief change. Dynamic conditioning is an exact algorithm for believe update that exploits the structure of the network to discover *local* and *relevant* cutsets (for conditioning). Relevant cutsets characterize cutset variables that affect the value of each message passed by the polytree algorithm, therefore, identifying whether two conditioned networks lead to the same value of a polytree message so that the message will be computed only once. Local cutsets, on the other hand, eliminate the need for conditioning on a full loop cutset, therefore, reducing the number of polytree computations. The algorithm of dynamic conditioning and some experimental results are described in [6].[9]

[9] These experiments show that dynamic conditioning performs comparably to the Lauritzen and Spiegelhalter algorithm as implemented in IDEAL [11, 21].

In conclusion, ϵ-bounded conditioning provides approximate but fast answers to queries about belief change. These approximate answers are in terms of lower bounds to the actual degrees of belief in a proposition and can be calculated easily. Furthermore, a slight modification of the algorithm will yield an anytime procedure [12]. Further details on ϵ-bounded conditioning can be found in [7].

4 Final Remarks

PSA is currently in the experimental-prototype phase. Its functionality was briefly demonstrated at Rome Laboratories (Air Force) in September 1994. The domain was evacuation operations of civilians in noncombatant situations. Figure 3 shows the required inputs and the current output of the PSA.

Efforts to put together the PSA started in February of 1994, although research in the representation of uncertainty through order-of-magnitude probability and arguments are part of the respective dissertations by the authors. Most of these efforts were concentrated on the representation of time using networks and on providing algorithms for inference that could scale well as these networks grew bigger and became more complex. The main contribution of this work, apart from the extensions to BN for modeling uncertainty and time, is precisely the ability to control the time required in computations by trading-off time versus accuracy. The approximation in the answers is controlled through a parameter ϵ (Section 3), since the closer ϵ is to zero, the better the quality of the answer. As an example, we ran ϵ-bounded conditioning on the network in Figure 2. We were interested in the uncertainty (probabilities in this case) of all the nodes after 3 time steps given a set of initial conditions and some actions. The expanded network contained approximately 80 nodes, and the cutsets found by the algorithms contained approximately 20 nodes. The number of states (cases) to be considered for an exact computation of the uncertainties was approximately 20×10^6 states. For $\epsilon = 0.2$, the algorithm computed an answer in approximately 2 minutes. Yet, the quality of the answer was not very good, 50% of the probabilistic mass was lost. For $\epsilon = 0.1$ the computation took nearly 4 minutes, yet the accuracy of the answers increased considerably, with an error of less than 0.05. For, $\epsilon = 0.01$, the computation took approximately 6 minutes, with an error of less than 0.002. The exact computation without using ϵ-bounded conditioning took 9 hours.[10] Yet, in most cases (except when strict optimality is required) an approximate answer, in the form of a lower bound (as returned by ϵ-bounded conditioning) is all what is needed. Thus, for example, a lower bound of 30% on the probability of an un-safe arrival (by the civilian contingent) may be all that is needed to dismiss a certain course of action. All the algorithms were implemented in C-NETS [5] which is a Lisp-based environment for representing and computing with generalized networks [3]. We expect substantial improvements in performance as the tool evolves out of the experimental-prototype phase. Other portions of the PSA including a CLIM interface were also implemented in LISP.

Future research and enhancements for the PSA include:

1. The implementation of additional canonical models of time behavior, such as decaying functions.
2. The addition of capabilities for mixing different quantifications of uncertainty in one action network.
3. The investigation of operational definitions of complex plan-evaluation criteria such as flexibility and robustness.

[10] The exact computation was performed using dynamic conditioning.

Acknowledgments

Jim White was instrumental in building the CLIM interface for the PSA. This work was partially supported by ARPA contract F30602-91-C-0031, and by IR&D funds from Rockwell Science Center.

References

[1] M. Boddy and T. Dean. Decision-theoretic deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67:245–286, 1994.

[2] G. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990. Research note.

[3] A. Darwiche. *A symbolic generalization of probability theory*. PhD thesis, Computer Science Dept., Stanford University, 1992.

[4] A. Darwiche. Argument calculus and networks. In *Proc. of the* 9^{th} *Conference on Uncertainty in AI*, 420–421, 1993.

[5] A. Darwiche. CNETS: A computational environment for causal networks. Technical memorandum, Rockwell International, Palo Alto Labs., 1994.

[6] A. Darwiche. Conditioning algorithms for exact and approximate inference in causal networks. In *Proc. of the* 11^{th} *Conference on Uncertainty in AI*, 99–107, 1995.

[7] A. Darwiche. ϵ-bounded conditioning: A method for the approximate updating of causal networks. Technical Memorandum 94-135, Rockwell Science Center, Palo Alto, 1994.

[8] A. Darwiche and M. Goldszmidt. Action networks: A framework for reasoning about actions and change under uncertainty. In *Proc. of the* 10^{th} *Conference on Uncertainty in AI*, 136–144, 1994.

[9] A. Darwiche and M. Goldszmidt. On the relation between kappa calculus and probabilistic reasoning. In *Proc. of the* 10^{th} *Conference on Uncertainty in AI*, 145–153, 1994.

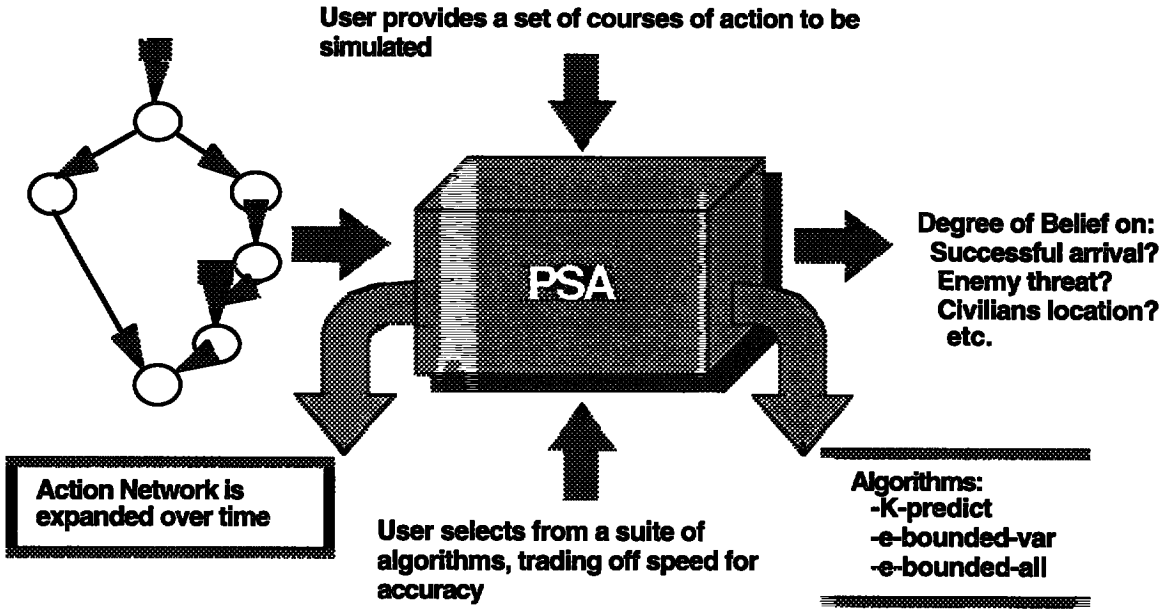

Figure 3: *Input/output schematics of the Plan Simulator and Analyzer (PSA) tool.*

[10] A. Darwiche and J. Pearl. Symbolic causal networks. In *Proc. of the* 12^{th} *National Conference on AI*, 238–244, 1994.

[11] F.Jensen, S. Lauritzen, and K. Olesen. Bayesian updating in recursive graphical models by local computations. Technical Report R 89-15, Institute for Electronic Systems, Dept. of Mathematics and Computer Science, University of Aalborg, Denmark, 1989.

[12] M. Goldszmidt. Fast belief update using order-of-magnitude probabilities. In *Proc. of the* 11^{th} *Conference on Uncertainty in AI*, 208-216, 1995.

[13] M. Goldszmidt and J. Pearl. Rank-based systems: A simple approach to belief revision, belief update, and reasoning about evidence and actions. In *Principles of Knowledge Representation and Reasoning: Proc. of the Third International Conference*, 661–672, 1992.

[14] M. Goldszmidt and J. Pearl. Reasoning with qualitative probabilities can be tractable. In *Proc. of the* 8^{th} *Conference on Uncertainty in AI*, 112–120, 1992.

[15] K. Kanazawa and T. Dean. A model for projection and action. In *Proc. of the International Joint Conference on AI*, 985-990, 1989.

[16] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems. *Royal Statistical Society*, 50:154–227, 1988.

[17] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[18] J. Pearl. From conditional oughts to qualitative decision theory. In *Proc. of the* 9^{th} *Conference on Uncertainty in AI*, 12–20, 1993.

[19] J. Pearl. A probabilistic calculus of action. In *Proc. of the* 10^{th} *Conference on Uncertainty in AI*, 454–462, 1994.

[20] M. Peot and R. Shachter. Fusion and propagation with multiple observations in belief networks. *Artificial Intelligence*, 48(3):299–318, 1991.

[21] S. Srinivas and J. Breese. Ideal: A software package for analysis of influence diagrams. In *Proc. of the* 5^{th} *Conference on Uncertainty in AI*, 1990.