Decision-Theory for Crisis Management
— Final Report —

Moisés Goldszmidt
(Principal Investigator and Program Manager)
Adnan Darwiche
(Principal Investigator)
Tom Chávez
David Smith
James White

Additional Participants:
John Breese, Denise Draper, Max Henrion, and Mark Peot

Rockwell Science Center
Palo Alto, CA 94301
(415) 325 7145

PIIN:
F30602-91-C-0031
PR NO.
B-1-3356
ARPA Order Number:
7741
Effective Date of Contract:
February-01-1991
Contract Expiration Date:
September-30-1994
Amount of Contract:
$1,808,626.00

Contents

List of Figures

List of Tables

1 Executive Summary

The Rockwell Science Center's main contribution to the ARPA Planning Initiative (ARPI) has been the development and application of decision theoretic concepts and tools to the problem of military transportation and crisis planning. Real world planning, like most complex problems, involves multiple competing objectives and uncertainty about actual outcomes. Thus, effective operational tools for planning must be able to efficiently manage uncertainty and trade-offs between competing objectives. Rockwell's contributions span three main areas of the planning process: generation, evaluation, and simulation. Contributions range from basic research results to application oriented software tools for military transportation planning. These contributions are briefly enumerated in the sections below, and are summarized in Tables 1 and 2.

Planning Subtask	Basic Advances	Demonstrations & Deliverables
Generation	Conditional planning (First CNLP). Control of planning search (Op. Graphs). Evaluation of partial plans.	Briefing at Rome Labs 1993.
Evaluation	Transportation Trade-off Analyzer Tool (TTA). Course of Action Trade-off Analyzer Tool (COATA). New Algorithms for Sensitivity Analysis.	IFD-3 Sept 1993. ARPI Workshops 1992, 1993, and 1994. Software training sessions for Rome Labs Manager, Sept. 1993 TTA, IDEAL, IDEAL-DEMOS May 1992.
Simulation	Uncertainty modeling over time. Suite of approximate and exact algorithms for inference.	Site visit (Palo Alto) by Rome Labs Manager, Sept 1994.

Table 1: *Summary of Rockwell's contributions to ARPI.*

1.1 Plan Generation Results

- **Conditional planning.** Developed and implemented the first algorithm for conditional partial order planning. The algorithm accepts a goal, initial conditions, and Strips-like operators, extended to allow uncertain outcomes. It produces partially-ordered plans with conditional branches for observed uncertainties. Details of the algorithm are in [37]. The ideas from this algorithm are now used in several other planners and algorithms including [18, 20, 8]. See Section 3.1.

Planning Subtask	Impact	Future Plans
Generation	Techniques and methods influenced other planners such as [18, 20, 8, 27, 34, 29].	Techniques being extended in ARPA–MADE program. Part of Rockwell IR&D.
Evaluation	Decision theoretical component in TARGET.	Graphical interfaces being extended for in-house tools.
Simulation	Uncertainty modeling techniques being used by other researchers [25, 1].	Extend simulation techniques to generation. Applications to diagnosis for Rockwell business divisions.

Table 2: *Impact on scientific and applications communities and future plans.*

- **Control of planning search.** Developed, implemented, and evaluated a number of techniques to help control the combinatorial explosion that occurs in generative planning. The control techniques include smart ordering of open conditions, recognition and control of recursion, variable binding analysis, and smart resolution of threats between operators. Descriptions of these techniques can be found in [42, 44, 38, 41]. Several of these ideas are now being investigated in other generative planners [27, 34, 29]. See Section 3.2.

- **Evaluation of partial plans.** Developed a technique for estimating the value of partially completed plans. This technique provides estimates of the cost and probability of success of achieving different possible subgoals that could occur for a problem. This information is then used to compute an estimate of the cost and probability of success for a partial plan [41]. We are currently investigating this technique further for the generation of assembly plans in the ARPA MADE project. See Section 3.3.

1.2 Plan Evaluation Results

- **Software tools.** During the course of the program Rockwell produced and made available to the ARPI community a set of tools for utility modeling and decision theoretic evaluation of plans. These include:

  - **IDEAL, IDEAL-DEMOS.** Domain independent decision theoretic software tools, which enable construction and evaluation of utility models (made available to CPE in May 1992). Delivered to the Common Prototyping Environment (CPE) (1992). See Section 4.2.
  - **Transportation Trade-off Analyzer (TTA).** Utility models and inference machinery for computing trade-offs between consumption of resources and

the expected time for the achievements of goals under uncertainty in the transportation domain (delivered in May 1992). See Section 4.1.

  - **Course Of Action Trade-off Analyzer (COATA).** Utility models and inference machinery for the evaluation of courses of action in the noncombatant evacuation domain. COATA constituted the decision theoretic component of the "Theater-level Analysis, Replanning and Graphical Execution Toolbox" (TARGET) for the third Integrated Feasibility Demonstration (IFD-3) (May and September 1993). See Section 4.3.

- **Sensitivity analysis.** Developed a new approximation algorithm for uncovering relevant data with respect to a given set of decisions and a course of actions. This functionality has proven to be essential for building utility models in both COATA and the TTA. See Section 4.4.

1.3 Plan Simulation Results

- **Modeling uncertain domains.** Developed a graphical language, called Action Networks, for modeling uncertain situations for the purpose of simulating their dynamics under different courses of action. Action networks are based on extending causal networks with explicit constructs for time and action. See Section 5.1.

- **Algorithms for uncertain inference.** Developed a collection of algorithms for simulating plans under uncertain conditions. The algorithms allow the user to trade-off accuracy for response time in the simulation process, and they can be used in general forecasting of uncertainty models:

  - **ϵ-bounded conditioning.** A method for the approximate simulation of plans that allows the user to trade the exactness of the simulation with the time it takes to compute the simulation. See Section 5.2.5.
  - **Dynamic Conditioning.** A method for the exact simulation of plans based on hypothetical reasoning (reasoning by cases). Dynamic conditioning provides an effective infra-structure for realizing computational techniques that are based on reasoning by cases. ϵ-bounded conditioning makes use of dynamic conditioning. See Section 5.2.2.
  - **Kappa Predict.** An algorithm for predicting the effect of plans with respect to order-of-magnitude action networks. The algorithm has a polynomial computational complexity but may be unable to provide complete answers in certain cases. Although the answers it provides may be incomplete, they may be enough to judge the quality of certain plans. It is a major building block of ϵ-bounded conditioning. Section 5.2.4.
  - **Action Network Predict.** An algorithm for predicting the effect of plans with respect to a subclass of action networks. The algorithm has a polynomial computational complexity on this subclass. See Section 5.2.3.

2 Introduction: Decision-Theory for Crisis Management

Military planning for crisis management and transportation scheduling is characterized by unavoidable uncertainties and difficult trade-offs about resources and objectives. For example, reliability and rapidity of deployment often come at the cost of increased resource requirements, including transport crew, vehicles, fuel, port facilities, and so on. Under limited resources and in crisis conditions, prompt delivery of force A to location X may require delays in delivery for unit B to location Y. Greater transport volumes may be possible for given resources, but only at the cost of higher risks of delay or loss. Thus, any successful tool for planning in this domain must be able to represent the inherent uncertainty, and formulate, weight and resolve the different trade-offs in order to compute and evaluate feasible courses of actions. Rockwell's main objective in this program has been to develop computational models and systems that incorporate notions of rational decision making and reasoning under uncertainty in order to enable the generation, evaluation and simulation of military plans that effectively meet these demands.

An examination of the state of the art at the time this program began reveals that traditional approaches to planning and scheduling, including the current military systems, employ goal-oriented and constraint satisfaction notions that deal primarily with feasibility and goal satisfaction. These approaches include a specification of a set of Boolean goals, a deterministic world model, and a set of action descriptions specifying how an action deterministically transforms the world from one state to another. Traditional AI planning shares this world view — it is normally defined as generating a sequence or program of actions that will allow the agent to transform the current world into a world where the goal is true. This view of planning is too narrow as a general notion of planning for several reasons.

The first problem is that goals are rarely expressible as a set of Boolean sentences. Invariably, there are partially achievable goals and trade-offs involved. We wish to achieve security for American citizens in some region, but security is rarely absolute and it may only be possible to secure major population centers. Given these types of trade-offs, we believe it is necessary to employ a richer representation of objectives, namely, a utility function, that allows one to score various combinations of outcomes (e.g., casualties, fiscal expenditures, achievement of mission objectives). As soon as one adopts a utility or objective function (or even some partial order on world states) as a representation of preferences, planning is transformed from a notion of constraint satisfaction and goal achievement into an optimization problem.

The second problem is that most automated planning and scheduling systems, including techniques arising in both AI and operations research (e.g. linear programming), do not include any notion of uncertainty. AI planning systems typically assume a deterministic world model, where all actions have deterministic effects, and any changes to the world model are due to the actions of the agent. These assumptions are clearly untenable

for military planning and scheduling. Action effects are not guaranteed because planes can break down, ports can be bombed, and weather can be adverse, all in unpredictable ways. Other agents, such as the enemy or some other threat, can interfere with ongoing plans or actions in a non-deterministic manner. Therefore, mechanisms for addressing uncertainty are necessary in a planning situation. To deal with this problem, existing DOD planning exercises often assume a worst-case scenario in order to assess the impact of an adverse outcome on the plan. Unfortunately, this strategy yields inflexible plans with sometimes unacceptable economical costs. In this project, we adopt probability as a mechanism for modeling uncertainty. This provides a theoretically sound means for incorporating uncertainty, and allows us to utilize a number of well-established algorithms for reasoning.

Rockwell activities in this program cover three main areas in planning technology:

1. **Plan Generation.** The objective behind this area of planning has been to research and develop general purpose planning systems suitable for military transportation planning and crisis management based on generalizing classical generative planning techniques to incorporate utility and probability metrics. Several significant advances were made that led to scholarly papers. Upon publication these advances were adopted by other researchers in their automated generative planners.

2. **Plan Evaluation.** The work in this area concentrated on developing approaches and software tools for the construction, analysis, and explanation of utility models in the domain of military transportation planning and crisis management. Some of these models were developed in close collaboration with military personnel.

3. **Plan Simulation.** Our involvement on plan simulation has been directed towards creating a tool for supporting and assisting a military planner in simulating the behavior of each possible course of action to enable a choice of which one to adopt. This work, under the name Action Networks, has significantly extended the power of the probabilistic reasoning methods with the addition of capabilities for qualitative reasoning, the explicit representation of time and actions, and the increased ease of end-user domain modeling.

A summary of the technical approach and main results in each of these areas can be found in Sections 3, 4, and 5 respectively. The details can be found in the set of papers we include in the Appendix.

3 Plan Generation

Our work on plan generation has focused on three areas:

1. Extension of generative planning techniques to allow the generation of conditional plans and plans under uncertainty.

2. Controlling the search required to generate plans.

3. Using utility information to help guide a planner towards better plans.

We have made significant progress in all three of these areas and give brief descriptions below.

3.1 Conditional Planning

Classical goal based planners assume that 1) the effects of operators are deterministic and completely specified, and 2) no exogenous events or actions take place during plan execution. Because of these assumptions, classical planners generate unconditional deterministic plans.

A conditional plan is necessary when uncertainties in the environment or in the effects of actions preclude the selection of a single course of action to accomplish a goal. A conditional plan contains tests and branches that result in different courses of action depending on the outcome of each test.

Under this contract we developed the first algorithm for conditional partial order planning (CNLP). The algorithm is based on the Systematic Non-Linear Planning algorithm (SNLP) of McAllester and Rosenblitt [31]. Like SNLP, CNLP accepts a goal, initial conditions, and Strips-like operators. However, unlike, SNLP, operators can have several different sets of effects. Each set of effects denotes a different possible outcome for the operator. For example the operator:

Drive(x, y)		
	Pre:	$\{\text{At}(x), \text{Route}(x, y)\}$
	Effects:	$\{\text{At}(y), \neg\text{At}(x)\}$
		$\{\text{Stuck}(x, y), \neg\text{At}(x)\}$

has two possible outcomes, one where the destination is attained, and one where the vehicle gets stuck enroute. (One can imagine other possible outcomes.)

Like the SNLP algorithm, CNLP works backwards from open conditions (goals and preconditions) and inserts new steps and causal links into the plan to achieve those open conditions. For CNLP, an action can be inserted into the plan if any one of its effect sets would achieve the open condition. When a step is introduced with multiple outcomes, CNLP needs to consider how to achieve the goal if any of the alternative (contingent) outcomes were to occur.

To do this, CNLP must keep track of two additional kinds of information:

1. Context: the set of contingencies under which each action in the plan will be executed.

2. Reason: the ultimate purpose for each action and open condition in the plan.

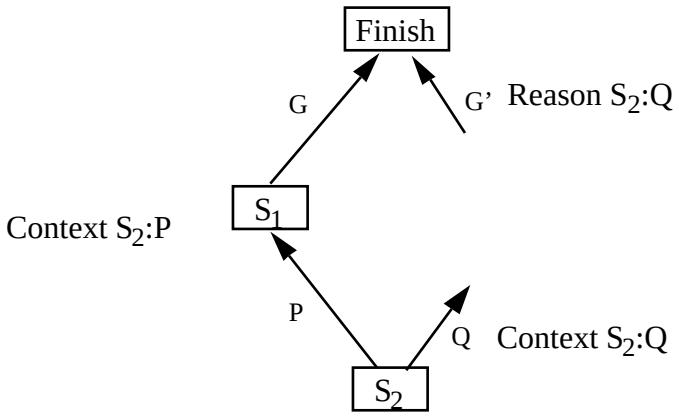

Figure 1: *A partial conditional plan.*

When a step with multiple outcomes is introduced, each outcome is tagged with a different context label. This context indicates the conditions under which any dependent steps will be executed, so any action supported by that effect inherits the context label.

To plan for contingent outcomes, the planner reintroduces the original goal as an open condition, but gives it a reason label that corresponds to the contingent context being considered. Planning then proceeds as before, except that steps in the original context cannot be used to achieve open conditions with the new reason label.

As an example, consider the partial plan in Figure 1. The original goal is the single clause G, and the step S_1 is being used to achieve it. The step S_2 is being used to achieve the precondition P of S_1. However, S_2 has another possible outcome, Q. The step S_1 is therefore labelled with the context $S_2 : P$. CNLP also adds a new open condition G' (a copy of the goal G) with reason $S_2 : Q$. Because $S_2 : P$ and $S_2 : Q$ are incompatible, the step S_1 cannot be used to achieve G'.

Threat resolution for CNLP is also more complex than for SNLP. For example, threats cannot occur between incompatible contexts. *Conditioning* an action on a contingent outcome can therefore be used to remove a threat from a plan.

Detailed descriptions of CNLP can be found in [37] and [36]. Techniques developed for CNLP are now being used in several other planners including [18], [20], and [8].

3.2 Control of Planning Search

Control of search is a critical problem in generative planning. The trouble is that there are too many partial plans, most of which either fail, or are very costly. Unfortunately, the introduction of uncertainty and conditionals only makes the problem worse; there are more circumstances that can occur and therefore more possibilities that must be considered.

We have developed and investigated a number of techniques to help control search including:

1. Recognition of operator relevance,
2. Variable binding analysis,
3. Smart open condition ordering,
4. Recursion analysis,
5. Conflict postponement,
6. Smart conflict resolution strategies.

Most of these techniques make use of a structure called an operator graph (also developed under this contract). Operator graphs facilitate meta-level reasoning about the way in which relevant operators can interact. In the next section we briefly describe operator graphs. In subsequent sections we briefly describe each of the above control techniques. Several of these control techniques are now being investigated by other researchers including Draper, Joslin [27], Kambhampati and Weld.

3.2.1 Operator graphs

An operator graph is a directed graph consisting of operators and their preconditions. Edges in the graph indicate which preconditions belong to which operators, and which operators can potentially be used to satisfy which preconditions. To illustrate, consider the simple set of operators in the table below:

Operator	**Precondition**	**Effects**
O_1	R	A
O_2	S	A
O_3	T	B
O_4	U, V	B
O_5	W	T

Table 3: *Operator definitions.*

Suppose that the goal is A and B and the initial conditions are R, S, U, V, and W. The operator graph for this problem is shown in Figure 2. Intuitively the operator graph is a schematic that shows the basic relationships between the goals, subgoals and operators for this problem. Among other things, it shows that both O_1 and O_2 can potentially be used to achieve A, and that O_3 and O_4 can potentially be used to achieve B.

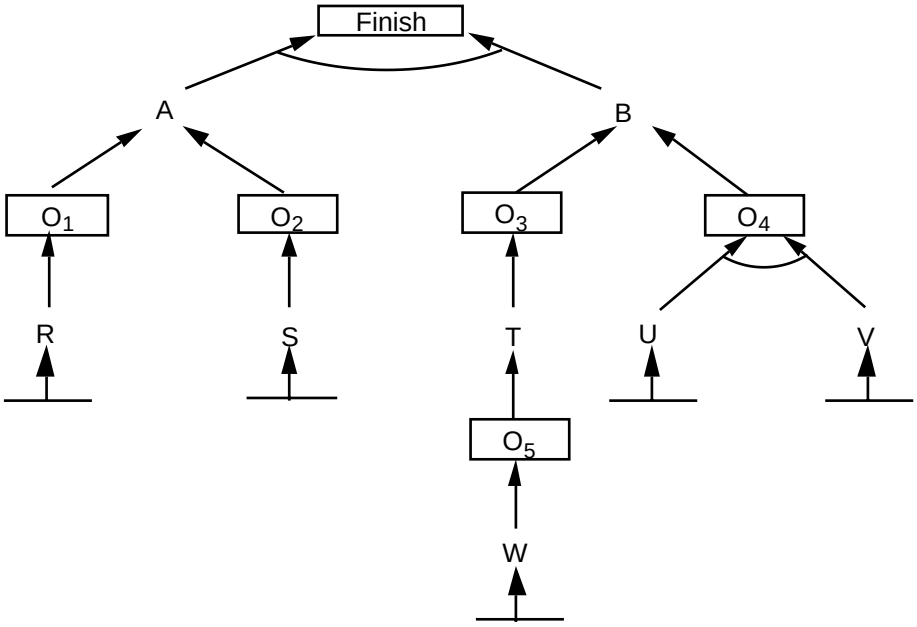

Figure 2: *Operator graph for a simple propositional problem.*

While there may be many potential plans for a problem there is only one operator graph, and each operator appears only once in the graph. As a result the operator graph has bounded size and is quick and easy to construct. A more complete description of operator graphs and their construction can be found in [44].

3.2.2 Operator relevance

One important advantage of constructing an operator graph for a problem is that it allows us to easily determine which operators are relevant to each open condition in a planning problem. For the open condition B we can readily determine that O_3 and O_4 are the only relevant operators.

We can take additional advantage of this property by performing some analysis on the operator graph ahead of time. Suppose that in the example above, W were not in the open conditions. In this case there would be no way of achieving the precondition of O_5, so O_5 can be deleted from the graph. Having done this, there is no way of achieving T, the precondition of O_3. As a result, O_3 can also be deleted from the graph. More generally, if any precondition of an operator in the operator graph cannot be achieved, the operator can be deleted from the graph.This can be applied repeatedly to eliminate all irrelevant operators from the graph.

Doing this analysis reduces the number of blind alleys that the planner must investigate.

3.2.3 Variable analysis

The example used above was purely propositional, but many planning problems involve variables. Figure 3 shows an operator graph for a first-order version of the previous problem. As indicated in the graph, there are many initial conditions that satisfy each of the predicates $R(x)$, $S(x)$, $W(x)$, $U(x,y)$ and $V(y)$. Now suppose that our planner first chooses to work on the open condition $A(x)$, and chooses the plan where O_1 is used to achieve $A(x)$. The planner might then try different possible bindings for x that achieve $R(x)$, and then try to achieve $B(x)$. Unfortunately, none of the possible bindings for $W(x)$ are compatible with the bindings for $R(x)$. Similarly, none of the possible bindings for $U(x,y)$ are compatible with the bindings for $V(y)$. As a result, all of these plans will ultimately fail.

Rather than rediscovering this for each plan, this variable binding analysis can be performed once in the operator graph. We start at the bottom of the graph and work upwards, figuring out the possible bindings for each open condition in the graph. So, in our example the possible bindings for $A(x)$ would be $x \in 0, 2, 4, 6, 8, 10 \ldots$, and for $T(x)$ would be $x \in 1, 3, 5, \ldots$. The bindings for $U(x,y)$ and $V(y)$ are incompatible, so O_4 can be immediately removed from the graph, and the possible bindings for $B(x)$ become $x \in 1, 3, 5, \ldots$. Many of the bindings for $A(x)$ and $B(x)$ are incompatible. Taking the intersection, we get the set $x \in 5, 15, 25, \ldots$.

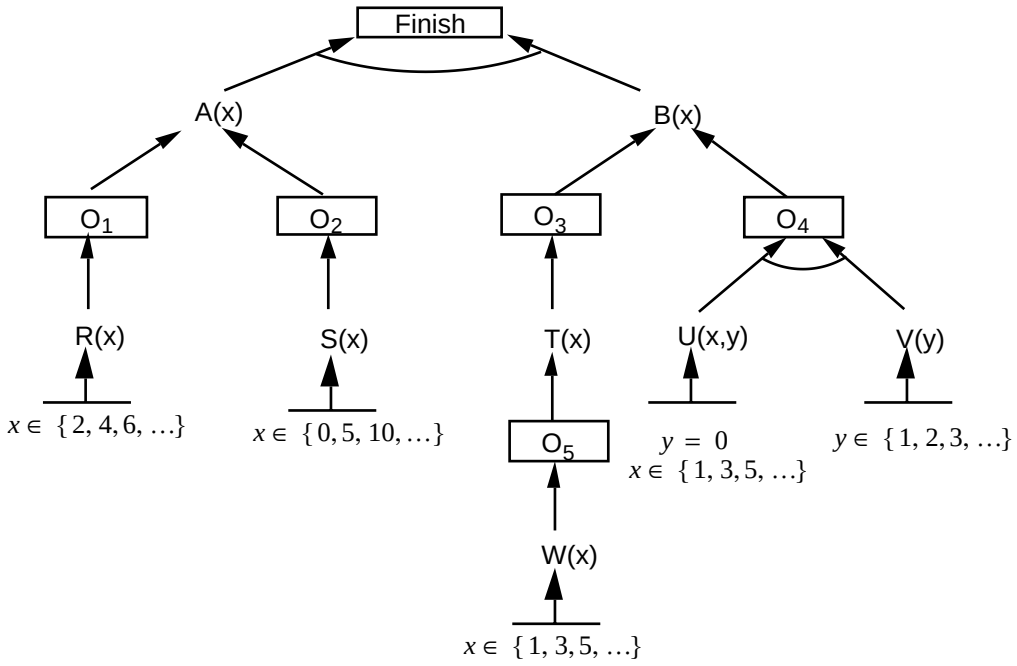

Figure 3: *First-order Operator graph.*

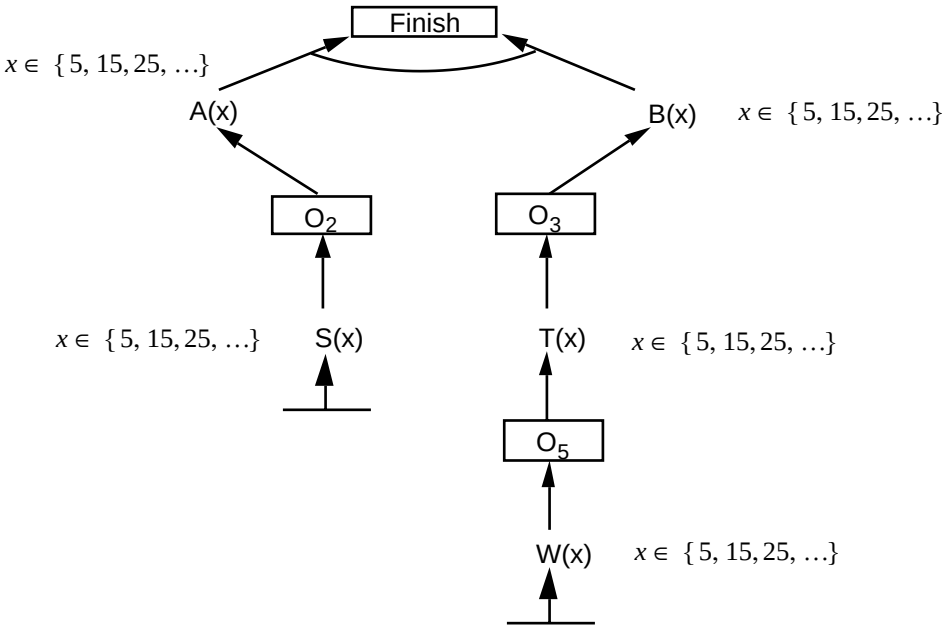

Figure 4: *Operator graph after variable analysis.*

The second part of the process is to propagate these limitations back down the tree. When we do this for $A(x)$ we find that the remaining bindings are incompatible with the bindings for $R(x)$, so O_1 can be deleted from the graph. When the process is complete, the resulting operator graph is as shown in Figure 4. This operator graph dramatically limits the space of plans that will have to be explored by the planner.

In general, variable analysis in the operator graph can be expensive because the sets of variable bindings can become large. However it is always possible to simplify the binding sets by removing those variables that have too many possible solutions. Removing the variable means that we are assuming it can take on any possible value. This may reduce the amount of pruning that can be done, but doesn't affect the completeness or soundness of the planning process.

3.2.4 Open condition ordering

The choice of which open condition to work on next often results in orders of magnitude difference in the amount of search required. The most important factor in deciding which open condition to work on next is the number of different possible ways of achieving the open condition. If there are few potential plans for an open condition, then choosing and expanding that open condition does not significantly increase the number of partial plans. Alternatively, expanding an open condition with many options leads to many

possible plans.

Counting the number of possible expansions for each open condition does add overhead to the planning process. However, the operator graph can be used to help speed this process. First, it provides an upper bound on the number of possibilities for each open condition. Second, it allows the planner to rapidly find and count the relevant expansions for each open condition. As a result, the overhead for this technique seems to be minimal. We have evaluated this technique in several standard planning test domains and found it to be extremely successful. These experimental results can be found in [38]. Joslin [27] has done further evaluation of this technique and has confirmed its significance.

3.2.5 Recursion analysis

Many operators in planning problems allow recursion. For example, a transportation operator might be used to go from A to B, but might also be used to go back from B to A. This means that a plan can be constructed that involves going back and forth between A and B any number of times. Sometimes this may be necessary. For example, several trips might be necessary to transport all the supplies to B. Unfortunately, such operators also allow planners to construct and investigate lots of ridiculous plans.

We've developed a technique that helps to avoid useless recursion. The basic idea is to recognize when the planner has generated a repeated subgoal and suspend work on that subgoal until there is some other reason for exploring that repeated subgoal. To illustrate, consider the partial plan in Figure 5a. The goal is to achieve P and R, and the step S_1 is used to achieve P. S_2 is used to achieve the precondition Q of S_1, but S_2 has the precondition P, which is one of the goals the planner was trying to achieve in the first place. As a result, the open condition P of S_2 can be suspended. This means that the planner should not work on this open condition unless some other open condition is linked into either S_1 or S_2. In Figure 5b, the open condition R is now being established by S_2. As a result, the condition P must be reenabled. In contrast, if no other open condition is ever linked to S_1 or S_2 (as in Figure 5c), the plan is discarded.

There is considerable overhead involved in recognizing recursive open conditions, and in checking to see when they should be re-enabled. Operator graphs help considerably in this process because they make it clear which open conditions are susceptible to recursion, and also make it clear which operators could potentially link into a recursion and cause open conditions to be re-enabled. (If there is no possibility of such links, the plan can be discarded early.)

There are a number of subtleties involved in the general technique.

1. The presence of variables makes suspension and enablement conditions more complex.

2. Certain conflicts between operators may also require re-enablement of suspended open conditions.

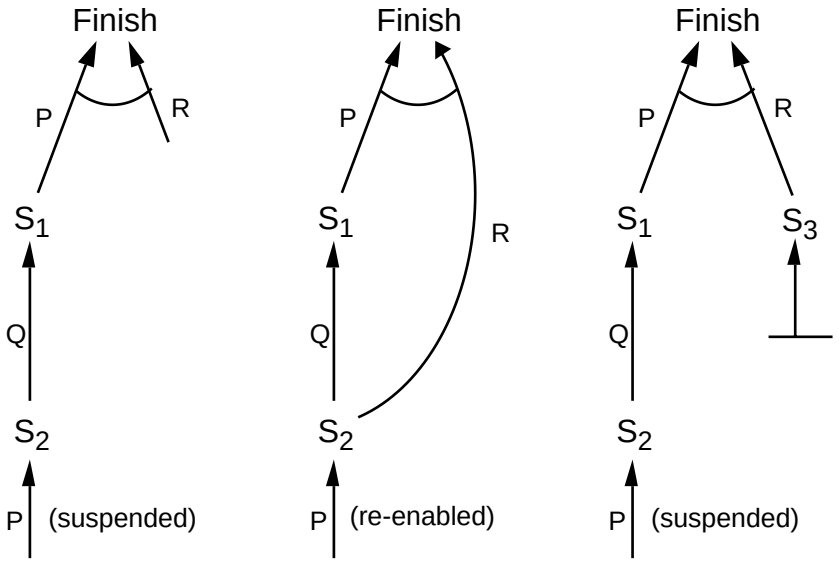

Figure 5: *Partial plans with recursion.*

A more complete description of this technique can be found in [40].

3.2.6 Postponing Conflicts

During the planning process, conflicts often arise between operators used to achieve different parts of a goal. For example, one operator may consume a resource needed for another operator, or may change some condition in the world that is a prerequisite for another operator. Resolving these conflicts is therefore a critical part of the planning process.

Several planners [39], [49], [50] ignore conflicts until planning is otherwise complete, and then try to fix the plan to eliminate conflicts. If the subgoals in the problem interact only loosely, this can be an efficient strategy. However, if there are complex interactions between the subgoals and operators for a problem, this approach results in extensive backtracking by the planner.

An alternative approach used in recent planning systems [31], [2], [52], [8] is to resolve each conflict as it arises during the planning process. Using this approach a planner notices unresolvable conflicts early in the planning process. The problem is that there are often many possible ways to resolve each conflict, thus contributing to an explosion in the search space. More bluntly, resolving conflicts during planning multiplies the exponential task of conflict resolution by the exponential task of planning.

Both of these alternatives are too extreme. Many conflicts are simple to resolve and can be delayed until the end. However, those that are tightly interconnected need to be

resolved during the planning process (once the entire group of conflicts is generated). We have developed techniques for automatically deciding which conflicts should be resolved during the planning process, and which conflicts should be delayed until planning is otherwise complete. The basic idea is to analyze the potential conflicts between operators in the operator graph. Roughly, a set of conflicts can be postponed if we can show in the operator graph that there will always be a way of eliminating the conflicts by imposing partial ordering constraints among the operators. Under these circumstances, the postponed conflicts can be ignored during planning, and can always be resolved by imposing additional ordering constraints on the otherwise complete plan. The technique is also recursive; postponing one set of conflicts may then allow postponing more conflicts.

In loosely coupled domains, many of the conflicts can usually be postponed. This has the effect of decoupling the planning problem into two pieces: selection of the operators for achieving the goals, and ordering the operators to eliminate conflicts between them. This can result in dramatic reductions in the amount of search required to find a plan.

More complete descriptions of this technique can be found in [43] and [44].

3.2.7 Conflict resolution strategies

Not all conflicts can be postponed. In this case the question arises as to when conflicts should be resolved during the planning process. The two options, resolve conflicts immediately, and resolve conflicts at the end, represent two extreme positions. There are several other interesting options:

DSEP : Wait to resolve a conflict until the variable bindings guarantee that the conflict will occur.

DUNF : Wait to resolve a conflict until there is only a single way of resolving it.

DRES : Continue checking each conflict to make sure it can be resolved, but don't resolve conflicts until the end.

In [38], we give a detailed description of each of these three strategies, and show that DSEP always generates a smaller search space than the usual strategy of resolving conflicts as soon as they occur. We also show that DUNF generates a smaller search space than DRES, which generates a smaller search space than the strategy of delaying consideration of conflicts until the very end. Furthermore, strategies DSEP and DUNF are not comparable: DSEP is better in some cases, and DUNF is better in others. These theoretical results are summarized in Figure 6

These results have led to a more complex strategy (DMIN) [45] that is a combination of both DSEP and DUNF. The basic idea is:

1. Do not consider any threat that is still separable.

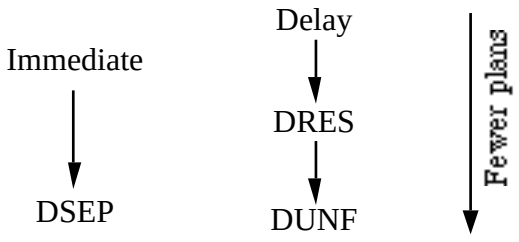

Figure 6: *Search space relationships for five threat removal strategies.*

2. Resolve any threat that has only one possible resolution.

3. Verify that there is a way of resolving any remaining threats (not covered by 1 and 2), but do not commit to any particular resolution of these threats.

We have shown, theoretically, that this combination strategy always generates a search space that is the same size or smaller than that of either DSEP or DUNF. We have verified all of these theoretical results by testing these strategies on several standard planning test domains. A complete description of these experiments can be found in [38]. Joslin, [27], and Kambhampati, [28], have also confirmed several of these experimental results.

3.3 Partial Plan Evaluation

In general we want planners that find not just any plan, but a good plan. Given utility models for a planning domain, we can evaluate and compare completed plans. However, it is computationally intractable to generate and compare all possible plans for a significant domain. Instead, we need to be able to evaluate and compare partial plans during the planning process, so that planning effort can be focused on the most "promising" plans. The tough part here is being able to estimate the cost and probability of success for those parts of the plan that are not yet complete. More precisely, we want to know how much it will cost, and how likely it is that we can achieve the remaining open conditions in the plan. This, together with the actions already in the plan, allows an estimate of the overall worth of the partial plan.

Operator graphs can be used to help estimate costs and probabilities for unfinished parts of the plan. To see how this works, consider the operator graph in Figure 7, where C_1 through C_5 are the costs of operators O_1 through O_5 respectively. According to the operator graph, the conditions R, S, W, U, and V can all be satisfied by the initial conditions. As a result, they would all have cost 0. Next consider the condition A. There are two possible operators that can be used to achieve A, O_1, and O_2. If O_1 is used, a cost of C_1 will be incurred, plus the cost of achieving O_1's preconditions. If O_2 is

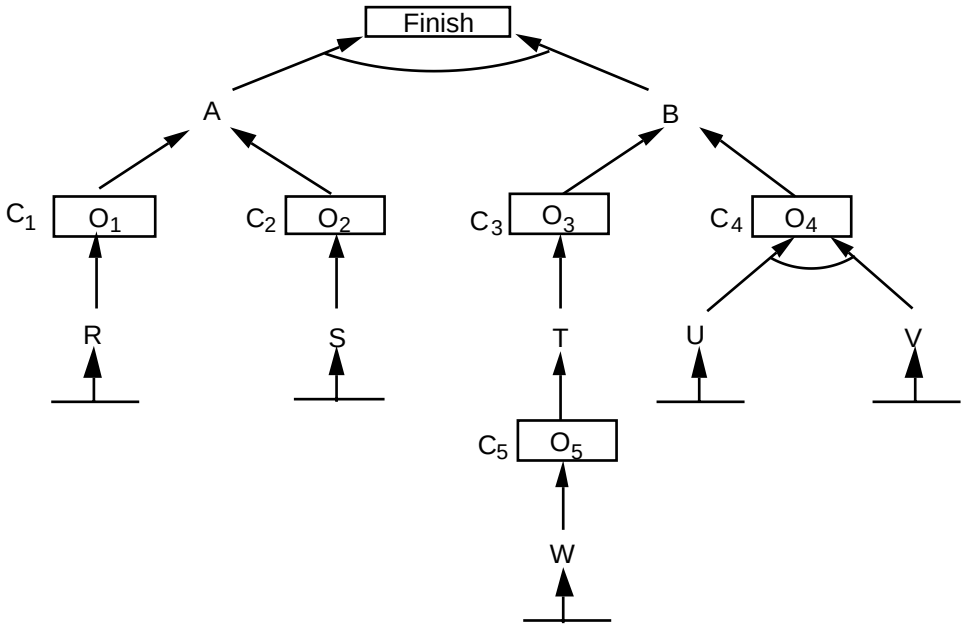

Figure 7: *Propositional operator graph with operator costs.*

used, a cost of C_2 will be incurred, plus the cost of achieving O_2's preconditions. Thus the best we can expect for achieving A is the minimum of these two costs, $\min(C_1, C_2)$. Similarly, the best we can expect for achieving B is $\min(C_3 + C_5, C_4)$.

Now consider the partial plan shown in Figure 8. It contains one step, an instance of O_2, and two open conditions, S, and B. The cost of O_2 is C_2, the cost for S is 0, and the best cost for B is $\min(C_3 + C_5, C_4)$. As a result, the best cost for this partial plan is $C_2 + \min(C_3 + C_5, C_4)$. Using this cost, this plan can be compared with other candidate partial plans.

In the above example, our calculations were particularly simple because the operators are all propositional (no variables) and there is no recursion possible. When variables and/or recursion are present, the analysis becomes much more complex. For example, when variables are present, the cost of achieving a particular open condition may depend heavily on the variable bindings. To take this into account we must use variable analysis like that described in Section 3.2.3 to find out the possible relevant bindings for variables. We then compute minimum costs for these different possible variable bindings. Recursion among the operators poses similar problems.

Although we only considered cost in this example, the same kind of calculations can be done for probability of success and utility. The key is that the operator graph allows us to consider the alternative ways of achieving each potential subgoal, to facilitate some

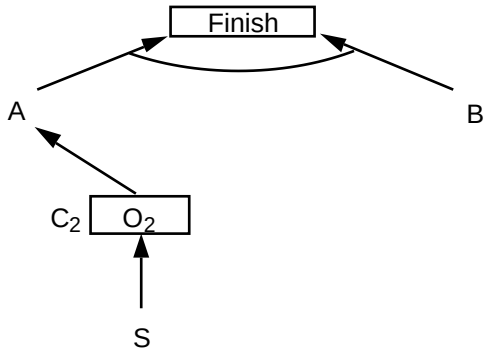

Figure 8: *Partial plan with two open conditions.*

estimate of how costly it will be to achieve that subgoal.

A more detailed description of this ongoing work can be found in [41].

4 Plan Evaluation

For plan evaluation, Rockwell efforts concentrated on the building of utility models, and the use of both in-house tools (IDEAL and IDEAL-DEMOS [48, 46]) and commercial tools (DEMOS [51]) for decision making and inference. The first significant product of this effort was the Transportation Trade-off Analyzer (TTA). This was a decision-theoretic based tool that demonstrated substantial capability enhancements over traditional transportation schedulers and feasibility estimators. The TTA is described in Section 4.1 and in previous annual reports to ARPI [4, 3].

The TTA was demonstrated at the end of the first year of the program, and as a result, Rockwell was invited to apply these decision-theoretic tools to the crisis action planning domain for the Third Integrated Feasibility Demonstration (IFD-3). Rockwell, along with ISX, developed the Course Of Action Trade-off Analyzer (COATA) for Non-combatant Evacuation Operations. The functionality provided by COATA was well received by the Operational Planning Team at the U.S. Pacific Command, and COATA was showcased in the original version of TARGET developed for IFD-3 as one of only a handful of Planning Initiative technologies then present in TARGET. On the basis of the work on COATA, Rockwell was invited to be a member of BBN's successful bid for the Deployable Joint Task Force Advanced Technology Program. COATA is described in Section 4.3.

Rockwell made available two decision-theoretic tools to the Planning Initiative com-

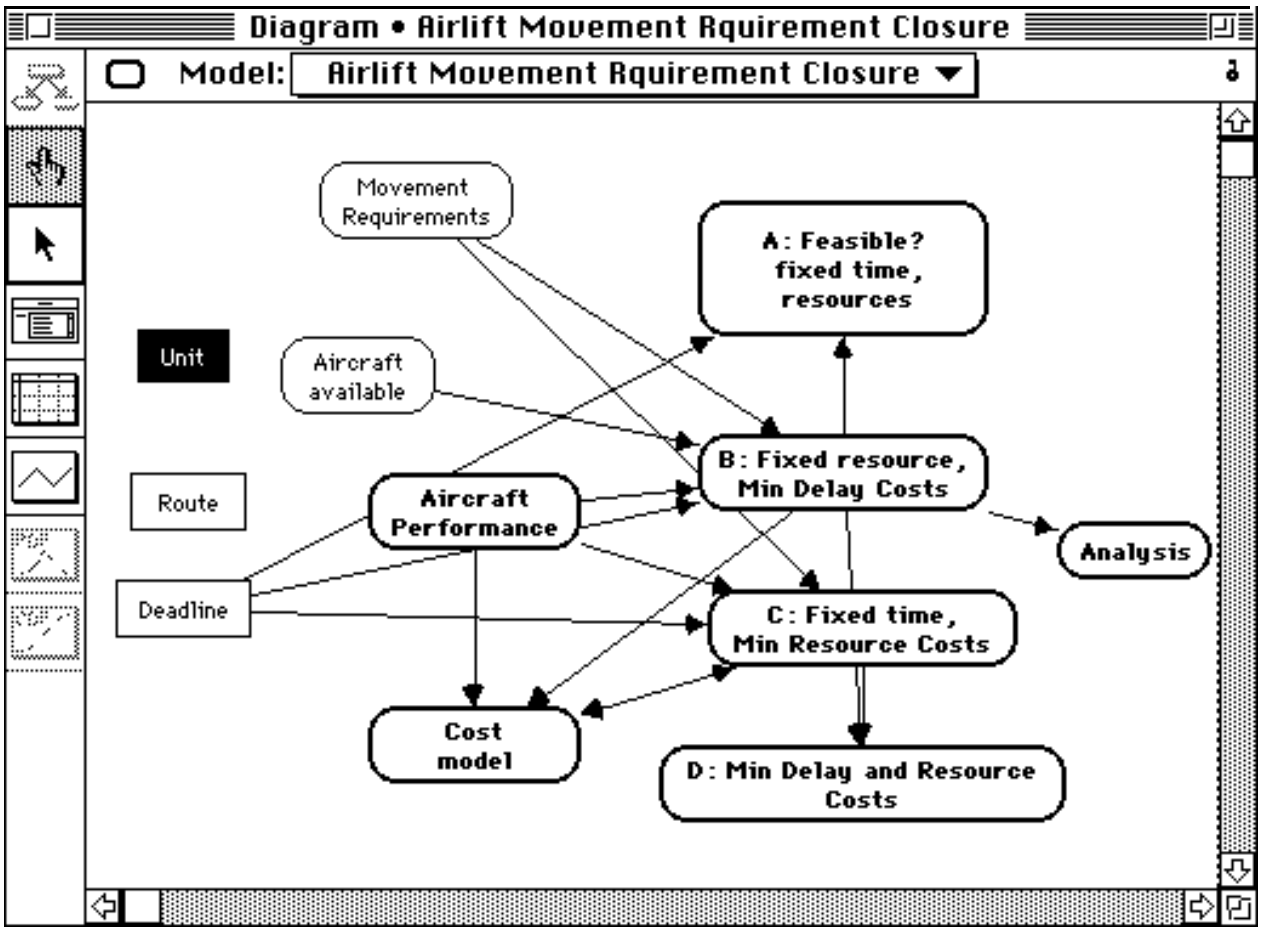

Figure 9: *Model structure used in the Transportation Trade-Off Analyzer.*

munity through the CPE.[1] These tools, IDEAL and IDEAL-DEMOS, are general decision-theoretic modeling tools that can be applied to a wide range of tasks. IDEAL is currently in use by hundreds of researchers within the decision theory and probabilistic reasoning community. Section 4.2 briefly describes the capabilities of IDEAL and IDEAL-DEMOS; further information can be found in [47, 46].

Finally, the development of utility models for crisis management pointed to the need for new techniques on sensitivity analysis, specifically on the computation of expected value of information. The new techniques and algorithms developed by Rockwell for this program are described in Section 4.4.

[1]Rockwell was in fact one of the first PI members to provide tools for other members.

4.1 Transportation Trade-off Analyzer

As the ARPI began in 1991, the focus was on increasing the capabilities of DOD tools for military transportation planning and scheduling. Our first goal was to explore the domain of military transportation planning and to capture the essential uncertainties and trade-offs inherent in this domain. The strategy was to develop utility and value models that express trade-offs between consumption of resources and achievement of goals under uncertainty, in order to express the preferences, objectives, and knowledge of DOD planning commands. Once this was done, we encoded this information in a decision-theoretic model which provided help with such capabilities as generating alternative plans, guiding search, and controlling computational resource allocation.

The domain study included training at the Armed Forces Staff College. The resulting tool was named "Transportation Trade-off Analyzer (TTA)". Uncertainty was modeled explicitly using belief networks and influence diagrams [32], and efficient mechanisms for reasoning with probabilistic information were employed. By explicitly modeling costs, the effects of delay, and uncertainties in the transportation process, TTA is able to optimize the transportation plan for changing requirements and under the changing conditions of the real world. In its most basic configuration, holding costs and resources fixed, the TTA model performs the same functionality as RAPIDSIM or DART in measuring plan feasibility. By relaxing these artificial constraints, TTA is able to provide significantly enhanced information. The characteristics of the TTA model are described in depth in Rockwell's first annual report [4]. A picture of the top level TTA screen is shown in Figure 9.

4.2 Decision-Theoretic Modeling Tools in the CPE

After the first year of the PI program, Rockwell made two general decision-theoretic modeling tools available to the other members of the PI community. These tools, IDEAL and IDEAL-DEMOS [48, 47, 46], were placed in the Common Prototyping Environment (CPE), and were some of the first researcher-based tools to be made available there.

IDEAL is a LISP-based test bed for work in influence diagrams and Bayesian networks. It contains various inference algorithms for belief networks and evaluation algorithms for influence diagrams. It contains facilities for creating and editing influence diagrams and belief networks both programmatically and via a CLIM-based graphical editor called IDEAL-EDIT [17]. An IDEAL-EDIT window displaying a basic IDEAL influence diagram is shown below in Figure 10.

IDEAL-DEMOS is a LISP reimplementation of the core of the DEMOS [51] modeling language. The objective was to complement the IDEAL inference system with some of the inference capabilities present in DEMOS.[2]

[2]DEMOS is a commercially available system for decision modeling.

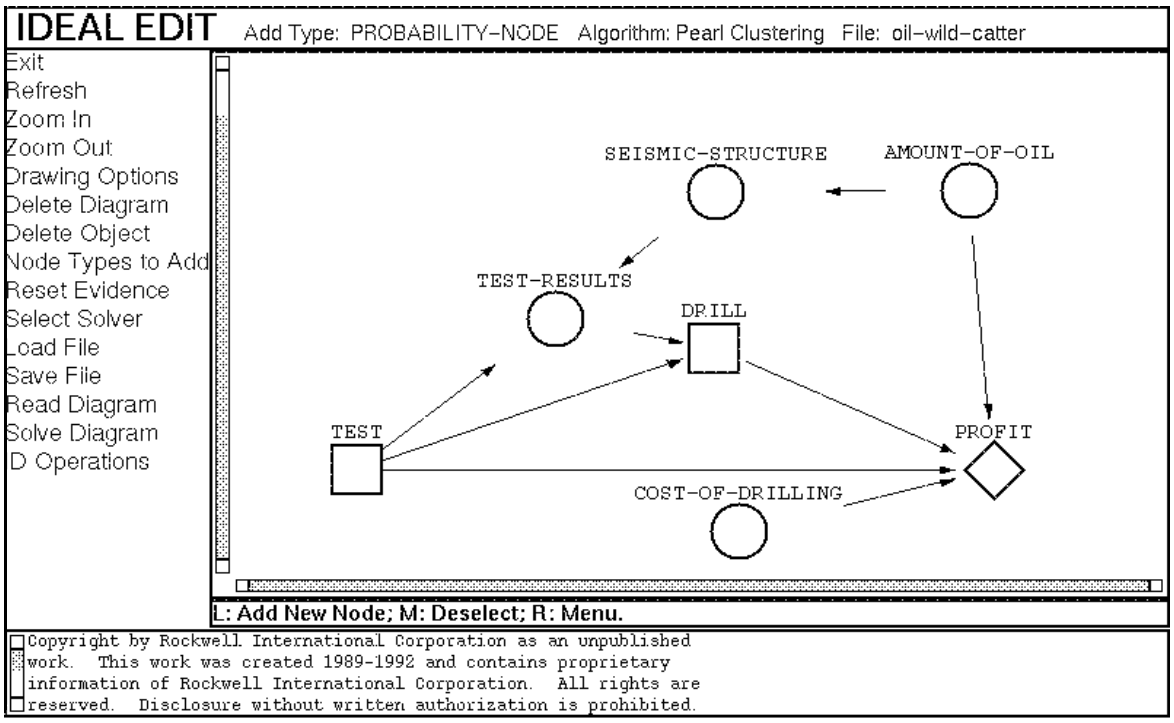

Figure 10: *An example of an influence diagram in IDEAL-EDIT.*

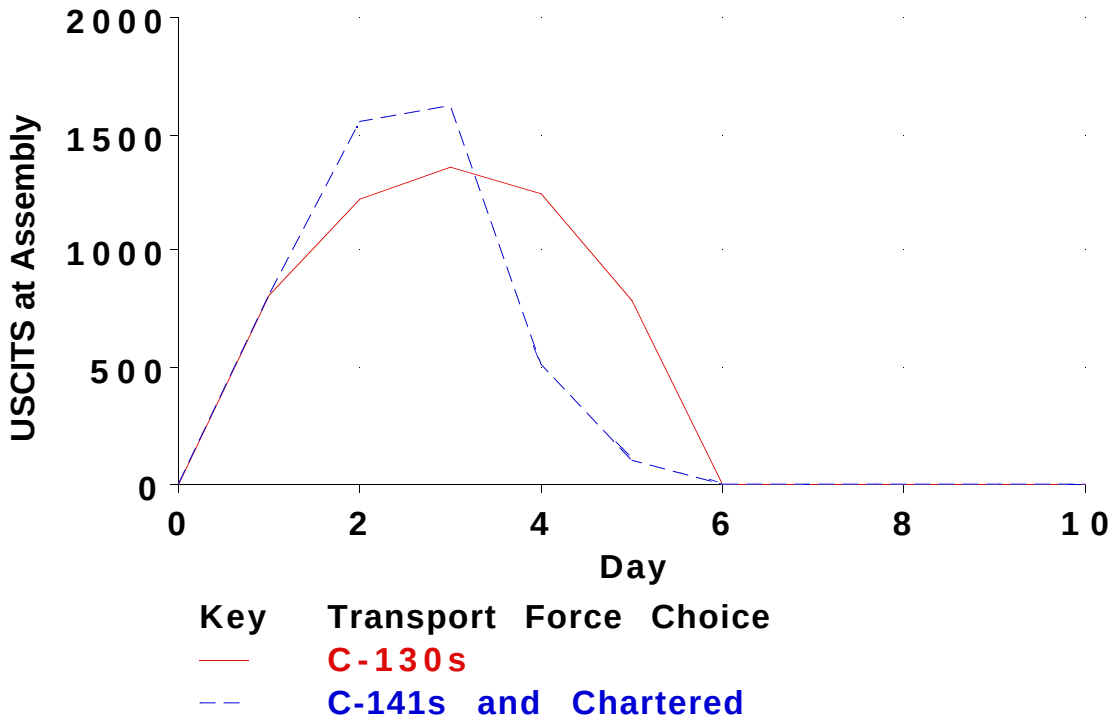

Figure 11: *The graph describes the buildup of U.S. citizens at an assembly area.*

4.3 Course Of Action Trade-off Analyzer

The success of the TTA model and the knowledge learned from its construction was beneficial in several ways. As well as providing feedback that was useful in other research efforts, Rockwell was invited to apply the same techniques used in the TTA model to the domain of crisis action planning for the Third Integrated Feasibility Demonstration (IFD-3) of the ARPA/Rome Labs Planning Initiative. Rockwell provided the decision-theoretic plan evaluation capabilities for the Theater-level Analysis, Replanning and Graphical Execution Toolbox (TARGET) during IFD-3. These capabilities included the means of estimating the feasibility, costs, casualties, and times associated with various courses of actions (COA). The targeted user was the executive officer of the Operational Planning Team (OPT). The objective was to provide a computational tool, called "Course of Action Trade-off Analyzer" (COATA), for capturing the assumptions and the rationale behind the planning process. The output of COATA is a set of estimates and entries that can support the development of a COA selection matrix. Figure 11 shows an example of these outputs.

COATA consists of three components: a decision model, an inference engine, and a graphical interface. The decision model was developed by Rockwell personnel in conjunction with U.S. Pacific Command to support COA analysis for Non-combatant Evac-

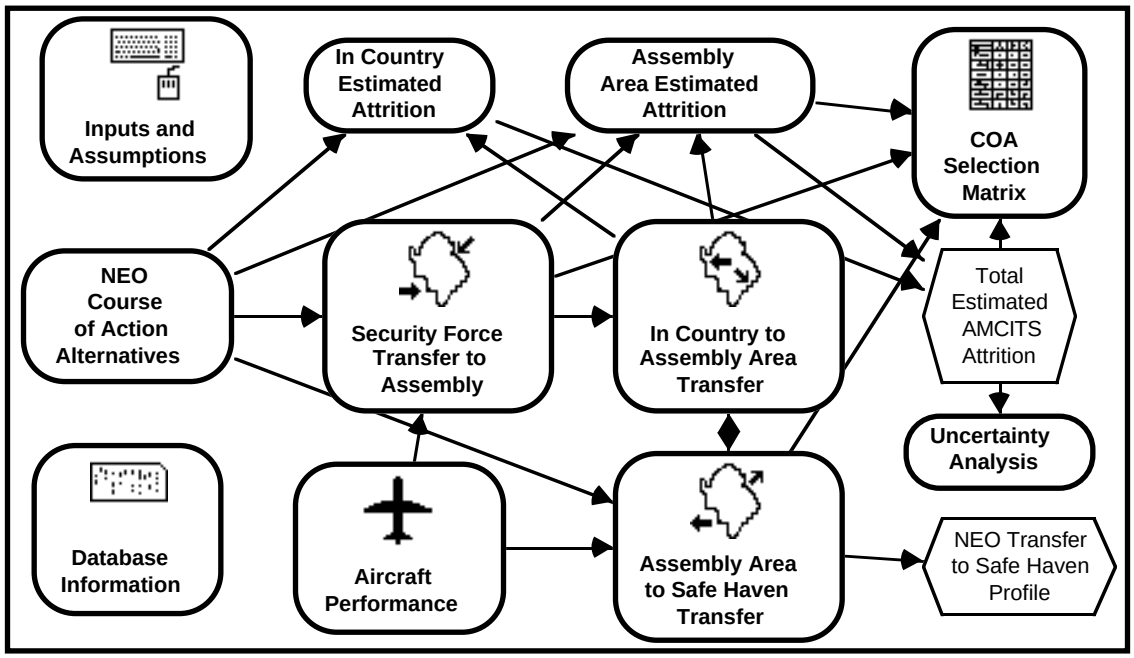

Figure 12: *Top level model for the NEO course of action analysis tool. The tool generates graphs describing risks for U.S. citizens and estimates of transportation times.*

uation Operations (NEO). The inference engine is part of a Rockwell in-house tool for decision making under uncertainty, and the run-time graphical interface was developed primarily by ISX.

The model assumes the following scenario: A situation arises which requires the evacuation of U.S. civilians from a foreign country. These U.S. citizens, located in one or more regions in the host country, are told to move from their current locations to one or more assembly areas in the country. Simultaneously, U.S. military forces are deployed from their current locations in the world to the assembly areas. Upon arrival, the military forces secure the assembly areas for civilian transit. Transportation assets are utilized to move the U.S. citizens from the assembly areas to safe haven(s), which are generally located outside of the country. The top level model (in the form of an influence diagram) is shown in Figure 12.

The COATA model allows the user to instantiate a generic NEO plan with specific locations, forces, and destinations, along with the (possible) uncertainties associated with this information (represented by probability distributions), and then to perform trade-off analyses for different courses of action. The user or some other automated planner must specify the specific data regarding country locations and number of citizens, the assembly areas, safe havens, security forces, and transportation assets. In addition, estimates on the risk to the U.S. citizens and the military of attrition (death) at various stages of the operation are also needed. The different COAs are defined by the choice of security forces (including their original location, date of availability and capabilities), safe havens, and transportation assets. Note that the use of probability distributions allows the specification of uncertain and incomplete data that can be refined during the various stages of the planning process.

Once the necessary information is specified, the model performs a dynamic simulation of the flow of U.S. citizens from their initial locations to assembly areas and then to the safe havens. Some of the more important outputs of the model include: time to completion, total of U.S. civilian and military casualties, speed of first response, and maximum build-up of U.S. citizens at assembly areas. The model also includes risk factors associated with both U.S. citizens and U.S. military personnel as a function of time. These risk factors are used to calculate expected casualties for various options. Additionally, since the uncertainty of any input is represented in the model, the tool is capable of generating an expected value for the parameter as well as a full uncertainty distribution for any of the output parameters.

The COATA tool provided a successful demonstration of technology transfer for IFD-3 at PACOM in May and September of 1993. The senior officer for the OPT at PACOM, who helped to develop the model, gave high marks to the capabilities of the tool. Currently, the Rockwell Palo Alto Laboratory is a member of an on-going project to extend the TARGET capabilities to the deployable Joint Task Force. The second annual report contains more specific information about COATA [3].

4.4 Sensitivity Analysis in Planning

The experience in building computer-based utility models for evaluating and simulating crisis scenarios pointed towards the need for more sophisticated techniques and methods for sensitivity analysis. It is necessary to decide which aspects of a scenario are essential for planning and need to be represented in the model, and which aspects can be ignored. The reason is simple: given that information is an expensive commodity, every model must balance the desire for maximum precision and detail against the need to get the job done with limited time, data, and computational resources. This tension is particularly acute in military crisis planning, where the stakes, risks and complexities are typically high, and where the deadlines are always tight.

Decision theory, and the techniques of decision analysis that are based on it, provide a number of powerful methods for analyzing these trade-offs between the precision of a model and the resources required to analyze it. The explicit representation of uncertainty in the form of probability distributions allows comparison of the effects of the different sources of uncertainty, and analysis of the costs and benefits of obtaining more detailed information to reduce the uncertainty. A planner needs to know which uncertainties matter, but he or she also needs to know how to allocate economic resources to resolve or reduce those uncertainties.

We have therefore focused on giving planners tools for **sensitivity analysis and scenario management**. Faced with dozens, possibly hundreds, of variables, with complex interdependencies and interactions among them, planners need principled, efficient techniques for identifying the most relevant or essential factors. They need to know where their strategies are most likely to "break," and, in particular, what uncertainties contribute most to the fragility of their plans. Such knowledge, in turn, helps them allocate resources (e.g., deployment of intelligence assets, consultation with other experts) for **information-gathering** and **uncertainty/risk reduction**.

For example, consider the NEO-COA model, developed by the Rockwell Palo Alto Laboratory, in collaboration with planners from PACOM, the Pacific Crisis Planning team introduced in Section 4.3, and described in [3]. NEO-COA was designed to help military planners analyze alternate courses of action for the evacuation of U.S. citizens from crisis areas. Examples of uncertainties explicitly represented in NEO-COA are the numbers of U.S. citizens, their locations in particular crisis regions, the accumulated risks they face over time, and the rates at which they move to assembly areas and points of embarcation. The number of citizens in the capital might be a probability distribution with median 500, and a 90% probability that the total is between 300 and 1000. Should planners go ahead now and schedule air transports with capacity for, say, 1000 evacuees? Should they wait until intelligence provides a more precise estimate? Or should they schedule air transports to carry 500 evacuees, with further capacity 500 to be held in reserve? NEO-COA can help planners compare these alternatives, and assess the expected value of better information relative to the costs of waiting for it.

4.4.1 The Expected Value of Information

The Expected Value of Information (EVI) is the best sensitivity measure because it analyzes a variable's importance in terms of the recommended action, and it expresses that importance in the value units of the problem. Other methods, such as rank-order correlation, measure a variable's contribution to the overall uncertainty in the model's outputs, expressed as a probability correlation. EVI, in contrast, measures a variable's importance in terms of what matters — in the case of NEO-COA, for example, civilian lives.

Certain methods, such as deterministic perturbation, measure a variable's importance in terms of what matters; yet they can still mislead. For example, suppose the only available evacuation transport is a ship that can hold 1500 passengers, and the key plan option is the choice of safe haven. The number of U.S. citizens at risk in the crisis region may affect the overall outcome of the operation, but will probably not affect the plan choice. EVI would reveal uncertainty about the number of evacuees to be irrelevant to the plan choice, while deterministic perturbation might show high sensitivity to this variable.

The use of less powerful sensitivity measures can mislead planners into thinking that it is worthwhile to wait for information about variables which are, in fact, irrelevant. Such sensitivities reflect the potential of new information to change beliefs or values on outcomes. EVI, in contrast, measures the potential of new information to change decisions.

4.4.2 Computing the Value of Information in Large Decision Models

To date, calculation of EVI has remained largely intractable for models of any reasonable size. In models with discrete variables (i.e., models that admit solutions using decision trees), EVI has computational complexity that is exponential in the number of state variables. For models with continuous variables such as NEO-COA, there were, to our knowledge, no algorithms — exponential or otherwise — for calculating EVI. Thus, an important part of our work has been to develop efficient methods for calculating EVI in probabilistic models for planning.

The innovative approach we developed is based on three features: The synthesis of an initial high dimensional problem into a summarizing function, the introduction of special information measures, and an efficient pre-posterior analysis. The algorithmic techniques are conceptualized in Figure 13, and a precise description of the algorithm and the results of its application to NEO can be found in [6, 7].

D denotes a set of m mutually exclusive actions; X is a vector of n state variables, all of which may be defined as probability distributions. The set-up alone makes clear why the calculation of EVI is difficult: it is a high-dimensional problem; worse, each dimension is itself a random variable. We assume a value function $V(d, X)$, which specifies value to the decision maker when X assumes a particular value and action d

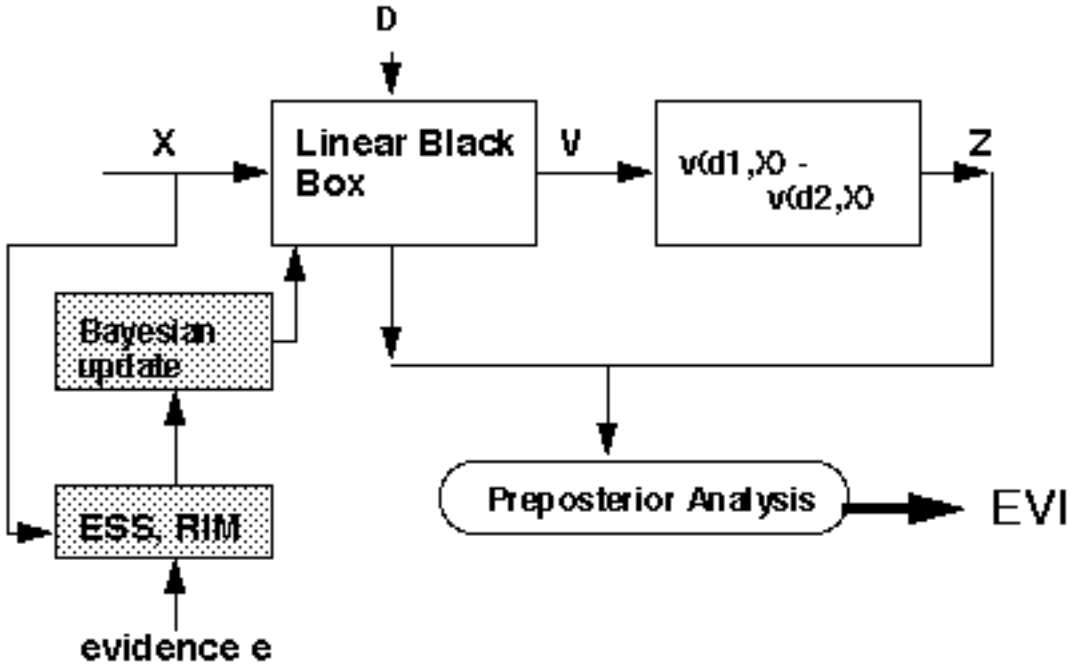

Figure 13: *Conceptual overview of the algorithm for estimating EVI.*

has been taken. Formally, the definition of EVI is as follows:

$$EVI = \max_d E[V(d,X)|X,s] - \max_d E[V(d,X)|s] \tag{1}$$

where E denotes expectation and s stands for the decision maker's prior state of information. (We include s simply to emphasize that all the quantities in the model have been assessed relative to the decision maker's prior state of information.) That is, EVI is the difference in expected value achieved by taking the optimal action with information on X, and the optimal action without information on X. For simplicity, in what follows we assume that D consists of just two discrete actions, d_1 and d_2.

Observing Figure 13, we see that X and D are applied as inputs to a linearizing method to be described shortly; the output is a variable we call Z. Z is defined as the difference in value between d_2 and d_1, assuming that d_2 is the preferred action (i.e., d_2 yields higher expected value than d_1).

$$Z = V(X,d_2) - V(X,d_1). \tag{2}$$

In Figure 14, we show a graph of the probability distribution for Z.

Z is the pivotal element of our analysis because it essentially collapses the dimensionality of our original problem. We begin with n probability distributions for each of the variables of X, a separate set of actions D, and a value function $V(X,d)$. The

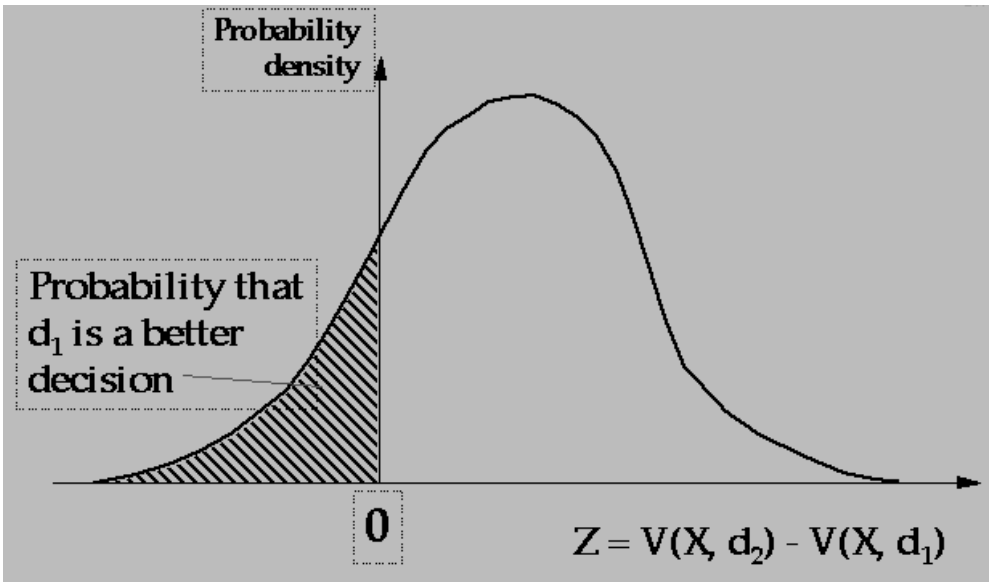

Figure 14: *Probability distribution on Z.*

probability distribution on Z — a simpler, two-dimensional quantity — encodes all the information from these elements that is vital for calculating EVI.

For the next portion of the analysis we consider the impact of evidence or information e on any of the state variables in X. We use two measures, ESS (equivalent sample size) and RIM (relative information multiple) to describe the magnitude of new information. ESS is commonly used in statistical decision theory; RIM is a new measure we have introduced. If X is the variable about which we expect to receive new information, then ESS is an equivalent number of observations on X's value, while RIM specifies a reduction factor in the variance of the probability distribution for X. The advantage of a RIM is that it is defined in terms of a parameter already familiar to the decision maker — the variance of the probability distribution for X. In essence, these measures allow the user to calculate the value of gathering more information about a certain variable, where "more" is formalized as a multiplicative factor of the initial state of uncertainty: e.g., what would be the value of decreasing by a factor of 2 our uncertainty on the exact number of citizens in the capital?

The final step of the analysis requires preposterior analysis on Z to calculate the effects of e, expressed in RIM's or ESS, on combinations of state variables. We rely on our linearizing method, probabilistic analysis, and numerical techniques to solve for EVI.

We have applied the algorithm to a large-scale decision model for planning NEO-COA introduced in Section 4.3. The results are described in [6, 7], and include:

- an operational framework for calculating EVI based on ideas from statistical deci-

sion theory;

- an efficient approximation algorithm for EVI;
- a formal analysis which guarantees the convergence of the algorithm, using a stopping rule which tests a simple condition involving pre-specified error parameters and the algorithm's current estimate; in addition, a guarantee that the algorithm's estimate is "probably approximately correct" — that is, with probability at least $1 - \delta$, the algorithm's output is within relative error ε of the "true" answer.
- a knowledge representation for evidence or information that accommodates perfect/partial EVI;
- an application of the algorithm to the NEO-COA model; and
- implementation of the algorithm in detachable software modules using DEMOS [51], a commercially available software tool.

In addition, we have recently completed a set of experiments demonstrating that the algorithm is **stable** in the sense that, as its running time increases, its estimates converge at roughly the same rate as answers calculated using exact techniques. We have verified that, while the linearizing assumption introduces error, its errors are at least well-behaved compared to the errors introduced by any estimation algorithm. This is important, because we want to insure that the errors in the algorithm's outputs are at least systematic, and that they decrease monotonically as we increase the algorithm's running time.

Our immediate goal is to apply the resulting algorithm to other large probabilistic models for planning. We are especially interested to see how well it "scales up" in large, temporal models for planning under uncertainty, though we do not anticipate difficulties with such an application.

In sum, we have developed, implemented and tested an **efficient algorithm for anytime approximation of EVI** in large probabilistic models, for perfect and/or partial information on any combination of variables, and with no limiting assumptions about the distributions of the variables, the nature of the value function, or the number of possible actions.

5 Plan Simulation

Our work on plan simulation was directed towards creating a tool for supporting the functionality portrayed in Figure 15. Here, a human planner is confronted with a situation, an objective and some courses of action (plans) that are believed to achieve the objective. The human planner, however, is looking for assistance in simulating the behavior of each of these plans to make a choice on which one to adopt. The purpose of

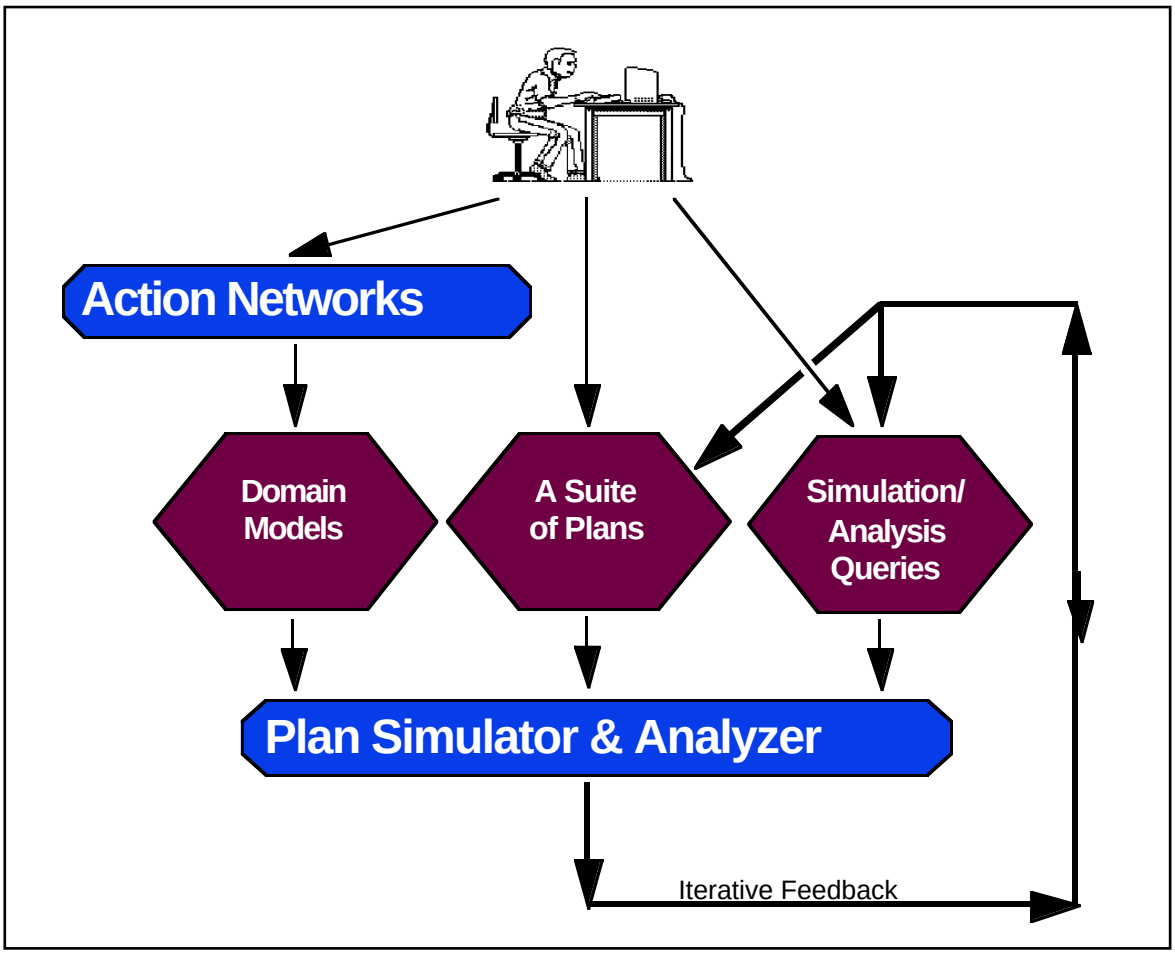

Figure 15: *Schematic view of a plan simulation scenario.*

our work under this part of the contract is to develop such a tool, which is comprised of two major components:

1. A language for modeling the situation at hand.
2. A set of algorithms that operate on the model in order to simulate a plan.

These two components are described in the following two sections (Sections 5.1 and 5.2.

5.1 Action Networks

The first step in simulating and analyzing a plan is to describe the domain in which this plan will be executed. According to the terminology used in SWAT team B report, this amounts to creating a *situation model.* In our research program, we proposed the

formalism of action networks to construct a situation model and used them to simulate military plans (see Figure 16). In the remainder of this section, we will focus on the following:

1. Provide a brief introduction into action networks as a domain modeling language
2. Summarize the current status of action networks based on research conducted under this contract

5.1.1 Introduction into action networks

An action network is a modeling tool oriented mainly towards dynamic domains that can be controlled by agents. Action networks are very closely connected to what is known as causal networks, which are the most sophisticated and effective tools for modeling domains under uncertainty as demonstrated by many real-world applications [5, 32].

The difference between (the recently developed) action networks and (the by now matured) causal networks is that causal networks do not include explicit constructs for modeling time and action, which are the key primitives for generating plans in dynamic domains. Our work on action networks was aimed mainly on introducing these two primitives to causal networks. We emphasize here that each action network is equivalent to a causal network. Therefore, action networks are supported by all the theoretical and practical tools that have been developed for causal networks, including the theory developed in [32] and the various commercial tools available from a number of vendors.

5.1.2 Progress on Action Networks

The first publication on action networks appeared in [14], which describes the following additions that action networks brought to causal networks:

1. *Controllable variables* (action), which are variables in a causal structure that can be controlled by agents. With respect to a controllable variable, an action network may designate a number of other variables that determine the conditions of controllability. These are called precondition variables and are connected to the controllable variables using a precondition arc.
2. *Persistent variables* (facts), which are variables in a causal structure the values of which persists over time unless acted upon by agents.

Variables that are neither actions nor facts are called events. These are variables that cannot be controlled but at the same time will retain their values over time.

In terms of implementational details, action networks and a number of corresponding inference algorithms are implemented in C-NETS [11], a common lisp system of which we are constructing a C version.

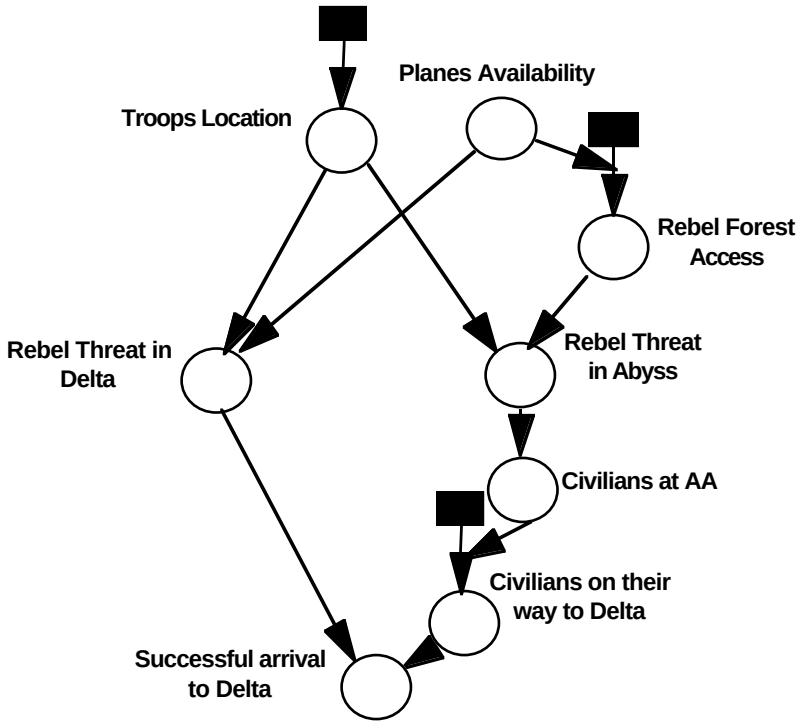

Figure 16: *An example of an action network.*

In addition to controllable and persistent variables, action networks have brought to traditional causal networks the ability do symbolic and qualitative reasoning under uncertainty. Traditionally, causal networks have been probabilistic in the sense that cause–effect interactions are quantified by providing probabilities of effects given their causes. But action networks are not necessarily probabilistic. Instead, a causal network will consist of two parts: a directed graph , representing a "blueprint" of the causal relationships in the domain and a quantification $\mathcal{Q}$ of these relationships. The quantification $\mathcal{Q}$ introduces a representation of the uncertainty in the domain because it specifies the degree to which causes will bring about their effects. Action networks allow uncertainty to be specified at different levels of abstraction:

1. Point probabilities, which is the common practice in causal networks [32].

2. Order–of–magnitude probabilities, also known as epsilon probabilities [23, 24, 15].

One should emphasize here that causal networks are usually associated with probabilistic reasoning and therefore are perceived to inherit the problems associated with this mode of reasoning, especially the infamous where-do-the-numbers-come-from? problem. We stress, however, that causal networks upon which action networks are based could be purely symbolic if the user so desires [16]. The principal investigators in this program have been leading the efforts for going beyond point probabilities in quantifying causal networks; their work on this can be found elsewhere [9, 23].

Although action networks support order-of-magnitude probabilities, they do not support pure symbolic modeling yet. We plan to address this ability, however, in future research.

5.2 Algorithms

5.2.1 Introduction

In our approach, plan simulation reduces computationally to reasoning with a belief network. There is a number of algorithms in the literature for this purpose [32], but they do perform poorly on networks generated for plan evaluation. The basic problem is the existence of undirected cycles (loops) in the network, which complicates the computation process.[3] Networks generated for plan evaluation tend to have many loops given their temporal nature. For example, a network with one loop could have more than twenty loops when expanded over three time steps.

To deal with networks containing loops, researchers have appealed to two concepts: clustering and conditioning. In clustering, the network is aggregated to allow the decomposition of computations into simpler ones. In conditioning, certain variables are instantiated to allow for conditional independences to hold, thus leading to more efficient

[3]If no loops exist, the network is singly–connected and the computational complexity is linear in the number of nodes and arcs.

computations. The method of clustering is due to Lauritzen and Spiegelhalter [30], and the conditioning algorithm is due to Pearl [32].

As was observed by Pearl [32], the use of conditioning to facilitate the computation of beliefs is not foreign to human reasoning. When we find it hard to assess our belief in a proposition, we often make hypothetical assumptions that render the assessment simpler. This correspondence to human reasoning is probably the reason why cutset conditioning was first expected to be more efficient than other methods for evaluating belief networks. But as observed later in [33], practice has shown that conditioning turned out to be computationally less attractive than the Lauritzen and Spiegelhalter algorithm [30].

Our commitment to conditioning methods stems from two factors: 1) We know that we can apply these methods to any network independently of its quantification [9],[4] and 2) we have identified a number of computational techniques that are best embedded in the context of a conditioning method. To provide a set-up for implementing these techniques, we developed the method of *dynamic conditioning,* which is a refinement of cutset conditioning that makes it comparable in computational performance to clustering methods. Based on dynamic conditioning, we developed the method of ϵ-bounded conditioning, which is an approximate method that allows one to trade efficiency for accuracy of computation. We also investigated two specialized algorithms. The first is a prediction algorithm that applies to action networks only by taking advantage of the recurrent structure of the temporal expansion of the action network. The second algorithm is also for prediction and is applied to order-of-magnitude networks only by taking advantage of approximating probabilistic uncertainty to orders-of-magnitude uncertainty. We will discuss each of these algorithms in the following sections. But we first summarize the algorithms according to the properties that make them of interest to plan simulation:

Dynamic Conditioning: A method for the exact simulation of plans and is based on hypothetical reasoning (reasoning by cases). The significance of the algorithm stems from: (1) It is the basis for the algorithm of ϵ-bounded conditioning to be discussed next; and (2) It is an infra-structure for realizing computational techniques that are based on reasoning by cases.

ϵ-bounded conditioning: A method for the approximate simulation of plans which allows one to trade the exactness of the simulation with the time it takes to compute the simulation. The significance of this algorithm stems from the following observation: Although the results of a particular simulation may not be exact, they may be enough to support a specific judgment about some plan.

Action Network Predict: An algorithm for predicting the effect of plans with respect to a subclass of action networks. The algorithm has a polynomial computational complexity on this subclass.

[4]To the best of our knowledge, there are no current results (or implementation) that suggests that clustering methods can work for quantifications other than probabilistic.

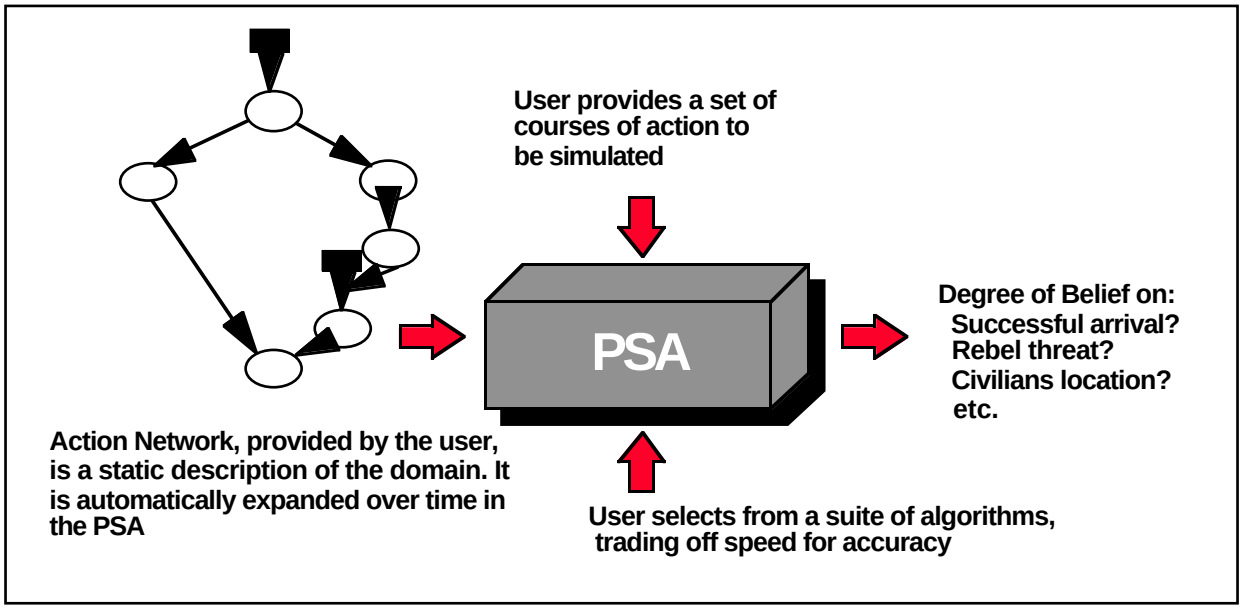

Figure 17: *Current realization of the Plan Simulator and Analyzer (PSA) tool.*

Kappa Predict: An algorithm for predicting the effect of plans with respect to order-of-magnitude action networks. The algorithm has a polynomial computational complexity but may be unable to provide complete answers in certain cases. The significance of this algorithm stems from at least two factors: (1) Although the answers it provides may be incomplete, they may be enough to judge the quality of given plans; and (2) It is a major building block of ϵ-bounded conditioning that we discussed earlier.

Figure 17 shows a refinement of the schematics in Figure 15, with a precise characterization of inputs and outputs including a choice of the algorithms described above for computation.

5.2.2 Dynamic Conditioning

As a first step in our research on algorithms, we have attempted to show that conditioning methods can be as efficient as clustering methods, thus setting the ground for our further improvements on this approach. In particular, we refined cutset conditioning into the method of *dynamic conditionining* [12], which was shown experimentally to perform comparably to the Lauritzen and Spiegelhalter clustering method.

The breakthrough that permitted our results is the discovery of two important notions in connection to conditioning methods: *local* and *relevant* cutsets, which are subsets of a *loop* cutset on which the method of conditioning is based. Relevant cutsets can be

identified in linear time and they usually lead to exponential savings when computing beliefs in the light of assumptions. Local cutsets, on the other hand, eliminate the need for considering an exponential number of assumptions because they identify assumptions that lead to the same beliefs. Local cutsets can be computed in polynomial time from relevant cutsets. The algorithm of dynamic conditioning is a refinement of cutset conditioning utilizing the notions of relevant and local cutsets. We provided experimental results in [12] showing that dynamic conditioning performs comparably to the Lauritzen and Spiegelhalter algorithm as implemented in IDEAL [19, 48]. We also discussed network structures on which dynamic conditioning has a linear computational complexity, while cutset conditioning leads to an exponential behavior.

Dynamic conditioning has been implemented in C-NETS [11][5] and tested for correctness against IDEAL [48] over hundreds of randomly generated networks.

5.2.3 Prediction Algorithm: Specific to action networks

Temporal causal networks that result from expanding atemporal networks (using the suppressor model) [14] tend to have a large number of undirected cycles. These cycles lead to networks that are difficult to manage computationally; in fact, inference in such networks is known to be NP-hard.

To deal with such networks, we have been exploring a number of approaches. One of these approaches tries to prove independence properties about temporal networks that are then utilized by algorithms in decomposing complex computations into simpler ones that can be performed in parallel. A second approach explores task-specific procedures with computation of approximate answers sacrificing accuracy in order to gain speed.

A key result obtained under this investigation is that temporal networks that result from expanding singly connected atemporal networks (without cycles) have some strong independences properties. In particular, it can be shown that the parents $\pi(N_t)$ of any node N_t at time t become independent given the images of N_t and $\pi(N_t)$ at time $t-1$. This independence is strong enough to justify the derivation of a *sound* and *complete* algorithm for prediction that is linear in the number of time steps and exponential only in twice the number of parents per node. The algorithm is valid for inference in probabilistic, kappa, and symbolic causal networks.

We stress here that although the original atemporal network may be singly connected, its temporal expansion may have many cycles, which makes it very difficult for general purpose inference algorithms. But by virtue of the expansion method (according to the suppressor model), the resulting multiply connected network is guaranteed to satisfy some strong independence properties.

[5] C-NETS is an environment for representing and computing with generalized causal networks [9, 13]. Therefore, dynamic conditioning is not restricted to reasoning with probabilistic causal networks, but is also applicable to other classes of networks, such as kappa causal networks [21] and symbolic causal networks [10].

5.2.4 Prediction Algorithm: Specific to order-of-magnitude networks

The second approach we investigated is an algorithm for prediction that takes as input a network quantified with kappas (degrees of plausibility based on probabilities) and returns as output the most plausible state of each variable.

The algorithm has the following properties:

1. The asymptotic complexity is $O(\max[E, N \times \mathrm{LU}])$ where E is the number of edges in the network, N is the number of nodes, and LU are table look-up operations.[6]

2. The algorithm is *sound*: If the algorithm predicts that x is the most plausible state of X, then the probability of $X = x$ is arbitrarily high.

3. The algorithm is *complete* only in any of the following cases:

   - The matrices representing plausibility of nodes never leaves that plausibility undetermined. That is for every state of the parents of the node in the network there is a unique most plausible state.
   - Nodes whose most plausible state remain uncommitted are not involved in cycles where all other nodes are also uncommitted.

   In the context of the algorithm, completeness means that if the probability of $X = x$ is high, then the algorithm should return x as the most plausible state of X. The lack of completeness simply means that some times the algorithm will remain uncommitted regarding to the most plausible state of a variable X, even though this state could be determined by exact (but time consuming) computations. We point out that checking whether the result of a run of the algorithm for a specific query is complete, is $O(E)$ where E is the number of edges in the network.

This algorithm can be used to perform prediction, given a course of action, in a domain represented using default rules, or as an approximate prediction algorithm in probabilistic representation. The approximation is done as proposed in [15], by transforming exact probabilistic statements into order-of-magnitude probabilistic statements. Details about the algorithm can be found in [22].

This algorithm is implemented on top of C-NETS [11].

5.2.5 Bounded Conditioning

ϵ-bounded conditioning is a method for the approximate updating of causal networks. It is a special case of bounded conditioning [26], which ranks the conditioning cases in cutset conditioning according to their probabilities and then considers the more likely cases first. The contribution of ϵ-bounded conditioning is in:

[6] The uncertainty in the domain is stored in matrices associated with the nodes in a network. A table look-up operation implies, in the worst case, examining all the entries of a particular matrix. The size of these matrices is nevertheless, bounded.

1. proposing a particular method for ranking the conditioning cases and

2. providing some guarantees about the approximate degrees of belief it computes.

To simplify the discussion, we will ignore the existence of evidence, therefore focusing only on the computation of $Pr(x)$ where X is a variable in the causal network.

Cutset conditioning computes the following sum [35],

$$Pr(x) = \sum_{s} Pr(x \wedge s),$$

where s ranges over all possible states of a loop cutset. The problem with cutset conditioning, however, is that the number of cutset states is exponential in the size of the cutset. To deal with this difficulty, bounded conditioning [26] orders the elements of the above sum according to their values so that elements of a bigger value are considered first.

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.

ϵ-bounded conditioning applies to both probabilistic and kappa causal networks, but we restrict the discussion here to probabilistic networks.

The Probabilistic Case

For some ϵ, ϵ-bounded conditioning splits the conditioning sum as follows:

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

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

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

An important property of ϵ-bounded conditioning is 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, which is discussed next.

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 an ϵ-algorithm for prediction is the one we discussed earlier for order-of-magnitude networks. This algorithm, which is linear in the number of arcs, is sound but incomplete, where incompleteness means that the algorithm may return "unknown" for whether $Pr(x) \leq \epsilon$.

This incompleteness, however, does not matter for ϵ-bounded conditioning: it only means that some cutset states which have probability greater than ϵ are being considered by ϵ-bounded conditioning although they should be ignored. Of course, however, the closer to being complete the ϵ-algorithm is, the better the performance of ϵ-bounded conditioning will be.

6 Concluding Remarks

Military planning activities, like most complex problems, must respond to multiple competing objectives with uncertainty about actual outcomes. The approach pursued at the Rockwell Science Center is centered on well founded normative notions of rational decision making and probability theory, and is based on the expertise of the researchers of the laboratory in this area.

The results of this program, as summarized in Tables 1 and 2, have produced important advances in the state of the art in the generation, evaluation, and simulation of plans, with recognized impact in both the scientific and application oriented community as well as in ARPA. Moreover, we are currently investigating the application (and transition) of the methods and technology developed under this program to Rockwell business divisions in the areas of automatic diagnosis, optimal repair, and resource allocation.

References

[1] A. Balke and J. Pearl. Probabilistic evaluation of counterfactual queries. In *Proceedings of the* 10^{th} *Conference on Uncertainty in Artificial, Intelligence*, 1994. In press.

[2] Anthony Barrett and Daniel S. Weld. Partial-order planning: evaluating possible efficiency gains. *Artificial Intelligence*, 67(1):71–112, 1994.

[3] John Breese, Michael Buckley, Tom Chávez, Max Henrion, Eric Horvitz, David Smith, Mark Peot, and James White. Decision-theory for crisis management – annual report II. Period of performance: February 1992 – January 1993, Contract F30602-91-C-0031, 1993.

[4] John Breese, Max Henrion, David Smith, and Mark Peot. Decision-theory for crisis management – annual report I. Period of performance: February 1991 – January 1992, Contract F30602-91-C-0031, 1992.

[5] Eugene Charniak. Bayesian networks without tears. *The AI Magazine*, 12(4):50–63, 1991.

[6] Tom Chávez and Max Henrion. Efficient estimation of value of information in Monte Carlo models. In *Proceedings of the* 10^{th} *Conference on Uncertainty in Artificial, Intelligence*, pages 119–127, 1994.

[7] Tom Chávez and Max Henrion. Focusing on what matters in plan evaluation: Efficiently estimating the value of information. In *Proceedings of the Workshop of Knowledge Base Planning and Scheduling Initiative (ARPA–Rome Labs.)*, pages 387–400, 1994.

[8] Greg Collins and Louise Pryor. Achieving the functionality of filter conditions in a partial order planner. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 375–380, 1992.

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

[10] Adnan Darwiche. Argument calculus and networks. In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pages 420–421, Washington DC, 1993.

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

[12] Adnan Darwiche. Dynamic conditioning: An efficient refinement of cutset conditioning using local and relevant cutsets. Technical memorandum, Rockwell Science Center, Palo Alto Labs., 1994.

[13] Adnan Darwiche and Matthew Ginsberg. A symbolic generalization of probability theory. In *Proceedings of the American Association for Artificial Intelligence Conference*, pages 622–627, San Jose, CA, 1992.

[14] Adnan Darwiche and Moisés Goldszmidt. Action networks: A framework for reasoning about actions and change under uncertainty. In *Proceedings of the* 10^{th} *Conference on Uncertainty in Artificial, Intelligence*, pages 136–144, 1994.

[15] Adnan Darwiche and Moisés Goldszmidt. On the relation between kappa calculus and probabilistic reasoning. In *Proceedings of the* 10^{th} *Conference on Uncertainty in Artificial, Intelligence*, pages 145–153, 1994.

[16] Adnan Darwiche and Judea Pearl. Symbolic causal networks. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 238–244, 1994.

[17] Denise Draper and John Breese. *IDEAL-EDIT: Documentation and user's guide.* Rockwell Science Center, Palo Alto Laboratory, 1992. Technical Memorandum # 77.

[18] Denise Draper, Steve Hanks, and Daniel Weld. Probabilistic planning with information gathering and contingent execution. In *Proceedings of the Second International Conference on AI Planning Systems*, pages 31–36, 1994.

[19] 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, Department of Mathematics and Computer Science, University of Aalborg, Denmark, 1989.

[20] Robert P Goldman and Mark S. Boddy. Conditional linear planning. In *Proceedings of the Second International Conference on AI Planning Systems*, pages 80–85, 1994.

[21] Moisés Goldszmidt. *Qualitative Probabilities: A Normative Framework for Commonsense Reasoning*. PhD thesis, Computer Science Dept., University of California Los Angeles, 1992. Technical Report TR-190, Cognitive Systems Laboratory.

[22] Moisés Goldszmidt. Belief-based irrelevance and networks: Toward faster algorithms for prediction. Working notes: AAAI Fall Symposium on Relevance, 1994.

[23] Moisés Goldszmidt and Judea 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: Proceedings of the Third International Conference*, pages 661–672, Boston, 1992.

[24] Moisés Goldszmidt and Judea Pearl. Reasoning with qualitative probabilities can be tractable. In *Proceedings of the* 8^{th} *Conference on Uncertainty in AI*, pages 112–120, Stanford, 1992.

[25] D. Heckerman, J. Breese, and K. Rommelse. Sequential troubleshooting under uncertainty. Submitted to CACM, 1994.

[26] Eric Horvitz, Gregory Cooper, and H. Jacques Suerdmont. Bounded conditioning: Flexible inference for decisions under scarce resources. Technical Report KSL-89-42, Knowledge Systems Laboratory, Stanford University, 1989.

[27] David Joslin and Martha E. Pollack. Least-cost flaw repair: a plan refinement strategy for partial-order planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1004–1009, 1994.

[28] Subbarao Kamhampati, 1993. Personal communication.

[29] Subbarao Kamhampati, Craig Knoblock, and Qiang Yang. Planning as refinement search: a unified framework for evaluating design tradeoffs in partial order planning. *Artificial Intelligence*, 1994. To appear.

[30] 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.

[31] David McAllester and David Rosenblitt. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 634–639, 1991.

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

[33] Judea Pearl. Belief networks revisited. *Artificial Intelligence*, Special Issue, "AI in Perspective," (59):49–56, 1993.

[34] J. Scott Penberthy and Daniel S. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of the Third International Conference on Knowledge Representation and Reasoning*, 1992.

[35] Mark Peot and Ross Shachter. Fusion and propagation with multiple observations in belief networks. *Journal of Artificial Intelligence*, 48(3):299–318, 1991.

[36] Mark A. Peot. *Decision-Theoretic Planning*. PhD thesis, Stanford University, 1995. In preparation.

[37] Mark A. Peot and David E. Smith. Conditional nonlinear planning. In *Proceedings of the First International Conference on AI Planning Systems*, pages 189–197, 1992.

[38] Mark A. Peot and David E. Smith. Threat-removal strategies for partial-order planning. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 492–499, 1993.

[39] Earl Sacerdoti. *A structure for plans and behavior*. North Holland, 1977.

[40] David E. Smith. Controlling recursion in planning. In preparation, 1994.

[41] David E. Smith. Evaluating partial plans. In preparation, 1994.

[42] David E. Smith and Mark A. Peot. A critical look at knoblock's hierachy mechanism. In *Proceedings of the First International Conference on AI Planning Systems*, pages 307–308, 1992.

[43] David E. Smith and Mark A. Peot. Postponing simple conflicts in nonlinear planning. In *Working Notes of the AAAI Spring Symposium on Foundations of Automatic Planning: The Classical Approach and Beyond*, pages 132–136, 1993.

[44] David E. Smith and Mark A. Peot. Postponing threats in partial-order planning. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 500–506, 1993.

[45] David E. Smith and Mark A. Peot. A note on the dmin strategy. Available by request, 1994.

[46] Sampath Srinivas. *IDEAL-DEMOS: Documentation and user's guide.* Rockwell Science Center, Palo Alto Laboratory, 1992. Technical Memorandum # 76.

[47] Sampath Srinivas and John Breese. *IDEAL: Influence diagram evaluation and analysis in Lisp. Documentation and user's guide.* Rockwell Science Center, Palo Alto Laboratory, 1989. Technical Memorandum # 23.

[48] Sampath Srinivas and John Breese. IDEAL: a software package for analysis of influence diagrams. In *Proceedings of the* 6^{th} *Conference on Uncertainty in Artificial Intelligence*, Cambridge, MA, 1990.

[49] Gerald Sussman. A computational model of skill acquisition. Technical report, MIT AI Laboratory, 1973.

[50] David E. Wilkins. *Practical Pplanning: Extending the Classical AI Planning Paradigm.* Morgan Kauffman, 1988.

[51] Neil Wishbow and Max Henrion. *Demos User's Manual.* Department of Engineering and Public Policy, Carnegie-Mellon University, 1987.

[52] Quang Yang. A theory of conflict resolution in planning. *Artificial Intelligence*, 58:361–392, 1992.

A Appendix: Relevant Papers

In the appendix we include five papers by project team members:

- **Tom Chávez and Max Henrion.** "Focusing on what matters in plan evaluation: Efficiently estimating the value of information." In *Proceedings of the Workshop of Knowledge Base Planning and Scheduling Initiative (ARPA–Rome Labs.)*, pages 387–400, 1994.
- **Adnan Darwiche and Moisés Goldszmidt.** "Action networks: A framework for reasoning about actions and change under uncertainty." In *Proceedings of the* 10^{th} *Conference on Uncertainty in Artificial, Intelligence*, pages 136–144, 1994.
- **Mark A. Peot and David E. Smith.** "Conditional nonlinear planning." In *Proceedings of the First International Conference on AI Planning Systems*, pages 189–197, 1992.
- **Mark A. Peot and David E. Smith.** "Threat-removal strategies for partial-order planning." In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 492–499, 1993.
- **David E. Smith and Mark A. Peot.** "Postponing simple conflicts in nonlinear planning." In *Working Notes of the AAAI Spring Symposium on Foundations of Automatic Planning: The Classical Approach and Beyond*, pages 132–136, 1993.