Time-Critical Planning and Scheduling Research at Brown University

Thomas L. Dean, Lloyd Greenwald,
Leslie Pack Kaelbling

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-94-41
October 1994

Time-Critical Planning and Scheduling Research at Brown University

Thomas L. Dean* Lloyd Greenwald* Leslie Pack Kaelbling*
Department of Computer Science
Brown University, Box 1910, Providence, RI 02912

Abstract

This report is a summary of recent work on time-critical planning and scheduling at Brown University. Much of our research over the last six years has been concerned with systems that are capable of managing computational resources for complex problem-solving tasks, in which the time spent in decision making affects the quality of the responses generated by a system. We developed an approach to designing algorithms and allocating computational resources to algorithms that is widely cited in the areas of real-time problem solving [8, 5]. We have provided a number of basic results regarding the design of systems that manage their computational resources by using expectations about the performance of decision-making procedures and preferences over the outcomes resulting from applying those procedures [4, 6]. We have also provided a basic framework for planning systems that employ decision theoretic techniques to evaluate plans [15, 16] and to perform probabilistic prediction [12, 14, 13].

The main thread of our recent research in time-critical planning and scheduling is concerned with decision making under uncertainty. Of particular interest are problems in which the notion of risk is complicated by limitations on time to plan and act. Our research has resulted in a family of algorithms [9, 11, 26, 21, 20] that extend techniques in operations research to efficiently handle stochastic processes with very large state and/or action spaces. Preliminary experimental results have helped to formulate a characterization of the class of problems for which our methods are well suited.

1 Introduction

In traditional transportation planning and scheduling, plans are made far in advance of execution with idealized expectations of the target situation; plans go wrong and are patched by humans with incomplete, local knowledge resulting in drastically sub-optimal performance. Our work focusses on providing technology to increase performance by developing

---

*This work was supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601 and by the Air Force and the Advanced Research Projects Agency of the Department of Defense under Contract No. F30602-91-C-0041.

conditional plans that take realistic failure scenarios into account and adapt on-line to unanticipated events.

We depart from the standard AI planning perspective, in which the world is assumed to be deterministic and planning is done entirely off-line. Traditional methods in operations research and combinatorial optimization either fail to account for uncertainty and time pressure or employ only simple predictive models. Even the best off-line methods using highly simplified models and relying on the latest linear programming and search methods cannot cope with problems of realistic size. In our work uncertainty and time pressure are given first-class treatment in solving real problems.

Problems such as transportation scheduling for large-scale strategic military airlift, air-traffic control, and factory machine scheduling require solving complex optimization problems in uncertain and changing environments. Systems designed to solve such problems must generate solutions in a timely manner and, if needed, revise those solutions quickly to accommodate new information. When the execution of plans and schedules must proceed in time-critical situations, the solutions must have flexibility built in. Flexible solutions to on-line planning and scheduling problems provide high payoff by using existing airfields, transportation assets, highways, and factory equipment more efficiently and by reducing the need for additional capital outlays.

This research adopts a stochastic-modeling framework and makes use of novel techniques for planning in unpredictable, dynamic environments with complex state and action spaces. Our primary interest is to extend existing planning and scheduling techniques to handle high-dimensional state and action spaces in a time-critical setting. We have identified general principles for dealing with large state and action spaces. Our iterative-refinement planning and scheduling methods rapidly assimilate new information and provide timely solutions in response to critical events. Planning and scheduling are tightly coupled with on-line execution. Our methods make the best possible use of available deliberation time. We have designed algorithms and a general architecture to support dynamic planning, scheduling, and execution. The architecture and algorithms are general enough to apply to a wide range of problems modeled as stochastic dynamical systems with very large state and action spaces.

2 Time-Critical Planning and Scheduling

Our approach to time-critical planning and scheduling is to construct flexible, on-line solutions to time-critical combinatorial problems; where flexible, on-line solutions correspond to contingent schedules, conditional plans, or policies. Our methods allocate computational resources to interruptible, iterative-refinement algorithms that provide better solutions the more time they are given. These algorithms exploit the structure of a particular problem instance or problem class to restrict attention to a small subset of the search space. Allocation involves trading solution quality against costs due to deliberation delay in an effort to maximize comprehensive value.

We have identified properties of planning and scheduling problems that allow us to make certain restrictions without loss of optimality or with low probability of significant degradation of performance. We use decision-theoretic methods to trade solution quality against the costs of delays to produce solutions in accord with the demands of evolving situations. Our methods combine operations research approaches for dealing with uncertainty with artificial intelligence techniques for steering on-line deliberation to solve the most relevant problems as they occur dynamically. Our contributions are in techniques to deal with the large state and action spaces typical of real problems.

In this section we identify the problems with existing solutions and provide technical arguments for four major approaches to these problems. We approach large state spaces with both state-space reduction techniques that iteratively focus state space exploration (Section 2.1) and state decomposition/aggregation techniques that prune out irrelevant state variables (Section 2.2). These two approaches complement each other and lead to effective state-based deliberation. Our use of time-based abstraction shifts deliberation focus dynamically as problem parameters change (Section 2.3). We approach large action spaces through the use of on-line stochastic simulation and bottleneck detection techniques to iteratively focus action space exploration and reduction (Section 2.4). All of our approaches must consider the problems inherent in modeling complex stochastic processes. We summarize here some technical detail to support the claims of this work.

2.1 State-Space Reduction

Our first approach to managing large state spaces in time-critical, goal-oriented planning under uncertainty is to take advantage of known information about the problem to iteratively expand the search space. Through the use of a known start state, and a model of the nature of the world's non-determinism, we can restrict the planner's attention to a set of world states that are likely to be encountered on the way to the goal. Furthermore, the planner can generate more or less complete plans depending on the time it has available. In this way, we provide efficient methods, based on existing techniques of finding optimal strategies, for planning under time constraints in non-deterministic domains.

Dynamic programming algorithms for computing the optimal policy given a stochastic model of the world are useful in small to medium-sized state-spaces, but become intractable on very large state-spaces. We address this difficulty by making some informal assumptions about the environment that allow us to generate approximate solutions efficiently. In particular, we assume that the environment has the following properties:

- *high solution density*: it is relatively easy to find plausible (not necessarily optimal) solutions
- *low dispersion rate*: from any given state, there are only a small number of states to which transitions can be made, given a specified action
- *continuity*: it is reasonable to estimate the values of states by considering the values of nearby states

Many large, realistic planning problems, such as those involving high-level navigation or scheduling, have these properties.

Following the work on Markov decision processes (MDP)[2, 3], we model the environment as a stochastic automaton. Let $\mathcal{S}$ be the finite set of world states and $\mathcal{A}$ be the finite set of actions. The transition model of the environment is a function mapping elements of $\mathcal{S} \times \mathcal{A}$

into discrete probability distributions over $\mathcal{S}$. We write $\Pr(s_1, a, s_2)$ for the probability that the world will make a transition from state s_1 to state s_2 when action a is taken.

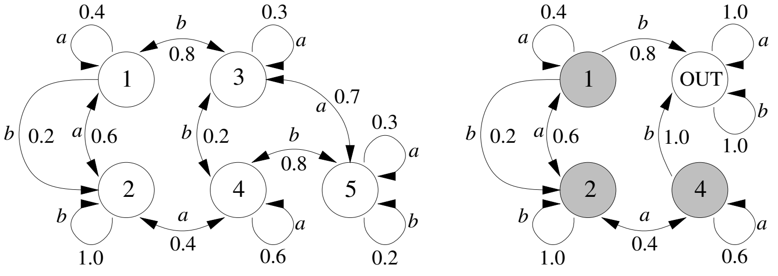
A *policy* π is a mapping from $\mathcal{S}$ to $\mathcal{A}$, specifying an action to be taken in each situation. An environment combined with a policy for choosing actions in that environment yields a Markov chain [25]. A *reward function* is a mapping from $\mathcal{S}$ to $\Re$, specifying the instantaneous reward that the system derives from being in each state. Given a policy π and a reward function R, the *value* of state $s \in \mathcal{S}$, $V_\pi(s)$, is the sum of the expected values of the rewards to be received at each future time step, discounted by how far into the future they occur. The *discounting factor*, $0 \leq \gamma < 1$, controls the influence of rewards in the distant future.

One of the most common goals is to achieve a certain condition p *as soon as possible*. If we define the reward function as $R(s) = 0$ if p holds in state s and $R(s) = -1$ otherwise, and represent all goal states as being absorbing, then the optimal policy will result in the system reaching a state satisfying p as soon as possible. A state is *absorbing* if all actions result in that same state with probability 1; that is, $\forall a \in \mathcal{A},\ \Pr(s, a, s) = 1$. The language of reward functions is quite rich, allowing us to specify much more complex goals, including the maintenance of properties of the world and prioritized combinations of primitive goals; this is explored in [10].

Given a state-transition model, a reward function, and a value for γ, it is possible to compute the optimal policy using either the policy iteration algorithm [23] or the value iteration algorithm [2]. However, for large state spaces, even a polynomial-time algorithm such as policy iteration becomes too inefficient. To compensate we consider a highly-restricted subset of the entire state space in our planning. A *partial policy* is a mapping from a subset of $\mathcal{S}$ into actions; the domain of a partial policy π is called its *envelope*, $\mathcal{E}_\pi$. The *fringe* of a partial policy, $\mathcal{F}_\pi$, is the set of states that are not in the envelope of the policy, but that may be reached in one step of policy execution from some state in the envelope. To construct a *restricted* stochastic automaton, we take an envelope $\mathcal{E}$ of states and add the distinguished state OUT, which is absorbing. The cost of falling out of the envelope is a parameter that depends on the domain. For example, if it is possible to re-invoke the planner when the system falls out of the envelope, then one approach is to assign $V(\text{OUT})$ to be the estimated value of the state into which the system fell minus some function of the time required to construct a new partial policy.

The basic algorithm starts with an initial policy and a restricted state space (or *envelope*), extends that envelope, and then computes a new policy. There are two basic types of operation on the restricted automaton. *Envelope alteration* serves to increase or decrease the number of states in the restricted automaton and *Policy generation* determines a policy for the system automaton using the restricted automaton. Note that, while the policy is constructed using the restricted automaton, it is a complete policy and applies to all of the states in the system automaton. For states outside of the envelope, the policy is defined by a set of *reflexes* that implement some default behavior for the system.

Figure 1.i shows an example system automaton consisting of five states. Suppose that the initial state is 1, and state 4 satisfies the goal. The path $1 \xrightarrow{a} 2 \xrightarrow{a} 4$ goes from the initial state to a state satisfying the goal and the corresponding envelope is $\{1, 2, 4\}$. Figure 1.ii shows the restricted automaton for that envelope. Let $\pi(s)$ be the action specified by the policy π to be taken in state s; the optimal policy for the restricted automaton shown in Figure 1.ii is defined by $\pi(1) = \pi(2) = \pi(4) = a$ on the states of the envelope and the reflexes



i. ii.

Figure 1: Stochastic process and a restricted version

by $\pi(\text{OUT}) = b$ (*i.e.,* $\forall s \notin \{1, 2, 4\}, \pi(s) = b$) (the reflex actions need not be the same in all states).

The basic algorithm is employed under various models of planning and execution. Two that have been considered are *precursor deliberation* and *recurrent deliberation*; they vary in the amount of concurrency between planning and execution. The details of the algorithms and models are given in [10]. We compared the performance of our planning algorithm with policy iteration optimizing the policy for the whole domain. Our results show that our planning algorithm supplies a good policy early, and typically converges to a policy that is close to optimal before the whole-domain policy iteration method does. Figure 2 shows average results from 630 runs, where a single run involves a particular start state and goal state.

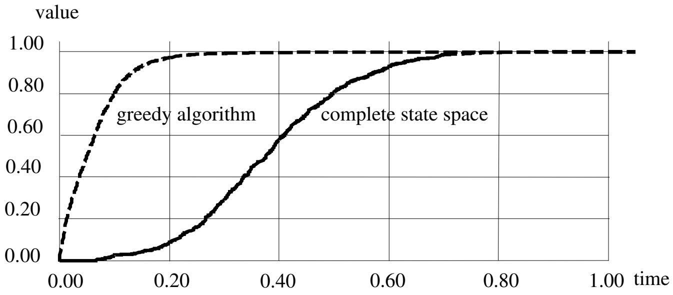


Figure 2: Comparison of planning algorithm using greedy deliberation strategy (dashed line) with the policy iteration optimization method (solid line): Average over 630 runs

2.2 State Decomposition/Aggregation

Complementary to our approach to state-space reduction are techniques for constructing abstract probabilistic world models that allow efficient approximate planning for very large stochastic domains. Domains are specified compositionally, in terms of state variables. The sensitivity of variables involved in the goal to other variables is analyzed, allowing approximate domain models at different levels of abstraction to be constructed. During the planning process, the approximation is refined until it is sufficient for the derivation of a plan at a specified level of reliability.

The models used in traditional AI planning, made up of operator descriptions, provide an important starting point for developing compact models of stochastic domains. They start by considering the world state as being composed of a number of state variables (typically propositional), each of which can take on values independently. The state transition function is given compositionally, by specifying, in different rules, how different aspects of the environment change given different actions. Thus, the color of walls need only be mentioned when describing change that occurs as a result of the paint operator.

A simplified version of the Bayesian network formalism [32] is well suited to specifying stochastic state transition and reward models. For each action, we use a two-slice network, in which nodes in the first slice represent values of state variables at time t and nodes in the second slice represent values of state variables at time $t+1$. In addition, there is a node in the second slice that represents instantaneous reward at time $t+1$. The value of this node depends deterministically on the values of the nodes to which it is connected; it plays the role of a value node in an influence diagram. Arcs in the diagram indicate probabilistic dependence between variables and are represented, in detail, by conditional probability distributions over values of the target variable given values of the source variables. Figure 3 shows a Bayesian network model for the results of performing the action GO and one for the turning actions (turn right, left and about all have the same network structure) in an example ground transport domain, whose states are characterized by the vehicle's location, direction, battery voltage, and the clutter of the current location.

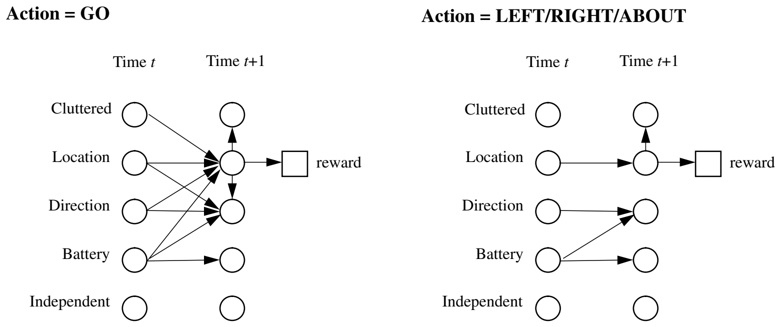


Figure 3: Bayesian network model of simple vehicle action

When there are no arcs between nodes in the second slice, the attributes of the state at time $t+1$ are *conditionally independent* given the state at time t. It is possible for attributes at time $t+1$ to be dependent on one another, as long as there is no cycle of dependency.

We assume that the state set $\mathcal{S}$ is describable as the cross product of a set of m sets of values V_i, one for each world attribute i. An individual state is described by the vector $\langle v_1, \ldots, v_m \rangle$. The state-transition function of the MDP can be derived from the Bayesian network in the conditionally-independent case as follows:

$$T(\langle x_1, \ldots, x_m \rangle, a, \langle y_1, \ldots, y_m \rangle) = \prod_i \Pr(y_i | a, x_1, \cdots, x_n) \ .$$

For any given y_i term, it is only necessary to condition on x_j if there is an arc from node j in the first slice to node i in the second slice. If the second-slice attributes are not conditionally independent, then because they are acyclic, it is possible to find an ordering of them that allows the following calculation:

$$T(\langle x_1,\ldots,x_m\rangle,a,\langle y_1,\ldots,y_m\rangle) = \Pr(y^1|a,x_1,\cdots,x_n)\prod_{i>1}\Pr(y^i|a,x_1,\cdots,x_n,y^1,\cdots,y^{i-1}) \enspace .$$

Even with a compact representation of the dynamics of the entire world, we will rarely want or need to work with the whole model. Given different goals, different time constraints, or different current world states, we might want to take very different *views of the world.* One way to construct different world views is by specifying only a subset of the possible attributes in the complete world model. In some cases, these abstract views will capture all of the world dynamics relevant to the problem at hand. In other cases, they will serve as tractable approximations to more complex models.

Even though the planning technique described in section 2.1 considers a potentially very small subset of the state space, it considers fully-specified world states, which may make distinctions that are not important to the planning task at hand. By restricting the world view before applying the envelope-based algorithm, we can potentially drastically reduce the number of states in the envelope. With an abstract world view, each of the states in the envelope stands for an equivalence class of world states that can be treated the same for the current purposes.

2.3 Time-Based Abstraction

Thus far we have focussed on managing large state spaces by considering restricted envelopes of states and decomposing the variables within any given state. In this section we discuss how to use the dynamic nature of planning and scheduling problems to shift deliberation focus over time. There are two components to this task. The first is to partition infinite horizon problems into problem instances over finite time intervals that take into account dynamically changing problem parameters and the second is to allocate on-line deliberation time to the corresponding problem instances. The planning approach of Section 2.1 takes a specific view of problem partitioning by using a restricted envelope and discount factor to focus attention on near (in time or space) states and considers two particular models for shifting deliberation focus over time. While this approach is satisfactory in some planning domains, in domains such as scheduling, the ability to find a reasonable envelope and discount factor is less certain; furthermore, as noted in [10] and [6], the interdependence of problem partitioning and allocation of on-line deliberation time must be carefully considered.

We have developed a planning and execution architecture that explicitly considers the problem of dynamically shifting deliberation focus. This architecture allows for concurrent planning and execution that is tailored to the dynamic nature of the environment. In addition, when possible, meta-scheduling may be performed off-line to aid time-critical tasks.

The architecture is depicted in Figure 4. The environment $g(x(t),u(t))$ enters a new state $x(t)$ at time t in response to the action $u(t)$ executed by the combination planning and execution component. The planning and execution component is divided into several on-line deliberative components and an on-line reactive component π_t to execute the policies generated by the deliberative components in response to the current state. Note that policies are non-stationary and are, therefore, indexed by time to explicitly account for a dynamic environment. The policies are generated on-line by deliberative decision procedures f. Due to the combinatorics involved in deliberative processing there is typically a delay Δ between the time a state triggers the deliberative process and the time the resulting policy is available for execution. The choice of decision procedure and appropriate calling parameters is determined by the strategy table Γ. When appropriate, the strategy table may be generated off-line (as depicted in the figure). A stochastic model of the environment is used both during on-line deliberation to predict future states and during strategy table construction to discern patterns of shifting dynamics. Detailed presentations of this architecture may be found in [19, 22].

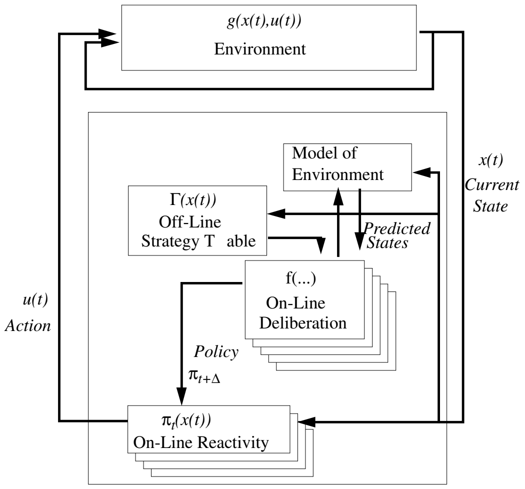


Figure 4: Planning and Execution Architecture

A typical instantiation of the planning and execution architecture proceeds as follows. At each time step t_j:

1. determine state $s_j = x(t_j)$ of environment g;
2. execute policy indicated by π_j to determine response to state s_j;
3. consult strategy table Γ to determine any new calls to decision procedures
4. if previous calls to decision procedures have terminated successfully since last time step, update reactive component π for those time steps by storing the constructed policies;

In the planning approach of Section 2.1, the decision procedures f correspond to envelope alteration and policy generation. The parameters used for these decision procedures determine the initial state from which to begin envelope creation. The precursor and recurrent deliberation models are two particular forms of strategy table for dispatching the decision procedures in response to changes in the state of the environment. In Section 2.4 we discuss an alternative form of decision procedure.

The incorporation of a strategy table and its associated deliberation scheduling and stochastic modelling allows us to explicitly reason about shifting deliberation focus over time and the interdependence of problem partitioning and allocation of on-line deliberation time. For example, we may consider the difference between partitioning time into disjoint or overlapping windows. Furthermore, if the windows are independent, a distinct policy π is generated for each window. For dependent windows, the policy is dependent on the output of other windows.

2.4 Action Space Reduction

In this section we discuss an approach to on-line scheduling that extends the approach of Section 2.1 to handle large action spaces. Given a large action space it becomes very expensive to consider all possible actions for any given state. Instead, we generate policies in two phases. We first employ Monte Carlo simulation and constrained dispatch scheduling using a stochastic model of the environment to sample the large state/action space and provide default reflexes to handle the remaining states. We then apply bottleneck-centered scheduling heuristics [1, 29] to improve initial policies. Deliberation scheduling is used to allocate computation time across these two phases. This work is described in detail in [21, 20].

In scheduling problems, each state consists of a number of available and unavailable resources and a number of activities that need to be scheduled on these resources. Actions consists of differing assignments of activities to resources. In the first phase of our approach, local assignments of activities to resources for a given state of a scheduling problem are determined by greedy dispatch scheduling rules such as first-come first-served, latest release date first or earliest deadline first. The greedy dispatch rules are controlled by applying constraints to the set of possible assignments (actions) on a state-by-state basis. Constrained dispatch scheduling solves the problem of scheduling in the face of a large action space. Given uncertainty, when executed, a given action may stochastically lead to a number of different states. Monte Carlo simulation is used to sample the state space reachable via the dispatch scheduling rules (actions) using a stochastic model of the effects of actions. The combined result is a policy executable under uncertainty.

The second phase analyzes the results of the first phase in order to detect areas for improvement (bottlenecks) within the policy and adds additional constraints on the action space in order to mitigate the bottlenecks and improve subsequent policies. Our technique for propagating constraints is based on bottleneck-centered heuristics [1, 29], a form of opportunistic scheduling in which critical points of resource contention are located and used to identify and remove potential for negative interactions between activities. Bottleneck-centered heuristics have demonstrated their effectiveness in deployed scheduling solutions [29]. We employ bottleneck-centered heuristics through two main steps. First, find a critical interaction (bottleneck) among activities; namely one which contributes directly to a policy with high expected cost. In practice [29] critical interactions occur during time intervals in which there is a high demand/supply ratio for resources. Second, identify an assignment of activities to resources that, if constrained, will both relieve the bottleneck and guarantee that an improved policy exists that may be found through dispatch scheduling.

Figure 5 depicts the mechanics of our two phase algorithm. From initial state 1 the dispatch scheduler chooses action C from among the unconstrained actions. Monte Carlo simulation is then executed to achieve three alternative next states reachable from state 1 through action C with varying probabilities. These states are then added to the list and the process is repeated by choosing a new state to expand from the list. From state 3 action I

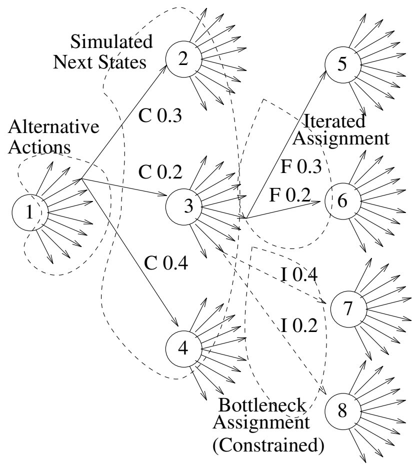


Figure 5: Scheduling algorithm applied to two step time window

is chosen and, through simulation, leads to states 7 and 8. After phase one has completed, bottleneck detection is invoked on the resulting policy and determines that action I in state 3 contains a bottleneck assignment. The action space for state 3 is constrained against action I (and other actions containing the bottleneck assignment). The two phases are then repeated with the new constraint added. This time the dispatch scheduler chooses action F to take from state 3 and, subsequently, states 5 and 6 are reached through simulation. In many cases the resulting policy will be an improvement (in expected value) on the previous policy.

This two-phase anytime algorithm for time-critical scheduling is most effective if constraints are only added when improved policies are expected. However, this method does not guarantee that a globally optimal schedule will be achieved. In order to guarantee global optimality without explicit exhaustive search over the action space we may need to retract as well as add constraints. This is an extension to our approach.

3 Related Research

Drummond and Bresina's *anytime synthetic projection* algorithm [17] incrementally constructs conditional plans for stochastic domains. Their work provided the initial motivation for our research into state-space reduction. Instead of building on Markov decision theory, Drummond and Bresina's work involves search in the space of situations given a set of operators that map situations to situations. Synthetic projection makes no effort to construct policies maximizing expected cumulative reward. Instead, synthetic projection seeks to maximize the probability of goal achievement. In particular, synthetic projection is not able to use expectations to predict and then avoid whole portions of the state space that are dangerous in the sense of having very low expected value. By adopting methods from Markov decision theory, we are able to identify relevant portions of the state space and then compute the optimal policy for the resulting restricted state space. We improve on the work of Drummond and Bresina by providing (i) coherent semantics for goals in stochastic domains, (ii) theoretically sound probabilistic foundations, (iii) and decision-theoretic methods for controlling inference.

Kabanza [24] describes a method for planning in nondeterministic environments that relies on exploring a small portion of the set of all possible action sequences and representing the resulting restricted automaton using a propositional branching time logic. Kabanza's approach does not make use of any probabilistic information regarding state transitions and makes no attempt to construct even an approximately optimal plan for any measure of performance. Thiebaux *et al.* [33] attempt to extend our work and that of Drummond and Bresina to use probabilistic logic [30] for planning under uncertainty. Probabilistic logic provides a more expressive representation than that offered by Markov process theory, but with the increased expressiveness comes increased computational overhead.

There have been a variety of other planning systems that are related to the problem. Georgeff's *procedural reasoning system* [18] was designed for on-line use in evolving situations, but it simply executes user-supplied procedures rather than constructing plans of action on its own. Systems for synthesizing plans in stochastic domains, such as those by Kushmerick, Hanks and Weld [27], do not directly address the problem of generating plans given time and quality constraints. Lansky [28] has developed planning systems for deterministic domains that exploit structural properties of the state space to expedite planning. Ow *et al.* [31] describe some initial efforts at building systems that modify plans incrementally. Neither Lansky nor Ow *et al.*'s systems deal with uncertainty nor provide options for user input and Lansky's system cannot handle concurrent planning and execution.

There is ongoing work at NASA in collaboration with the FAA on building systems for the automated management and control of air traffic [7]. We see our work as extending the basic capabilities of such systems to handle uncertainty and make the best use of the time available for planning in time-critical situations. Our work on large action spaces also borrows from and extends the operations research literature on bottleneck-centered heuristics developed by Adams *et al.* [1] and analyzed by Muscettola [29].

4 Application Areas

Our technology is directly applicable to problems in military and civilian air traffic control and fleet maintenance. We have also applied our techniques to problems of mobile robotics. We describe here other applications that we have investigated, all of which require the solution of highly complex problems in dynamic, incompletely-predictable domains.

Military and Civilian Air-Traffic Control and Fleet Maintenance With increasing air traffic, it is becoming desirable to coordinate arrivals and departures at airports for both military and civilian operations. Systems for global air-traffic control and terminal area traffic are faced with combinatorial problems that involve managing runway flow, scheduling landing and takeoff sequences, and assigning terminal gates. These systems are designed to ensure fuel-efficient and conflict free descents and to reduce delays at the gate and time spent waiting in holding patterns.

An example of our focus in this area is the problem of scheduling routine maintenance at distributed world-wide locations for an aircraft fleet. The input to this problem is a list of cities and their facilities, costs, maintenance histories for all aircraft, and the times at which facilities expect initial and amended schedules for specific intervals of time (usually during off-hours at an airport). In the current formulation, the system processes a steady stream of input including current aircraft location data. Takeoffs and landings are similarly logged. The system also processes time-stamped data regarding flight schedules and how they change over time. In this problem, the system cannot affect the schedules of aircraft but only the determination of which planes get maintenance in which cities. Periodically, the system issues maintenance schedules to particular cities; there are approximately 200 cities in about 15 time zones, but not all cities have maintenance facilities. There is a time when an initial schedule is due and a last time when the schedule can be amended. Approximately 30% of all flights do not fly their initially scheduled aircraft (this is industry wide) so there is a great deal of uncertainty in which planes will be in a given city for a given maintenance interval. The system that we are working on must decide when to allocate time to scheduling maintenance resources, how far in advance to schedule, and under what circumstances and for what cities to reschedule.

Dynamic Schedules for Health-Care Professionals Scheduling work assignments for nurses in a large hospital is a persistent problem and is related to crew scheduling problems in transportation domains. Typical constraints in the assignment of patients to nurses are that *primary* nurses are attached to each patient; nurses should be assigned patients that are geographically clustered, to avoid walking time; nurses must be assigned patient loads that are evenly balanced within a shift. In the assignment of nurses to shifts, an entirely different set of constraints applies: some nurses only work weekends; some generally work nights (but can be called in for day shifts in emergencies); nurses that have worked a night shift on one day should not work a day or evening shift the next; assignments should be chosen to minimize overtime pay during each week. The shift assignment problem is currently solved by one or two people with the aid of a computer program, making out a plan several weeks in advance. The patient assignment problem is worked out at the start of each shift. When staff illnesses, increased patient loads, or hospital emergencies disrupt the schedule, chaos ensues. Our stochastic modeling and prediction techniques could reduce the enormous time that is spent on the staff scheduling problem, introducing an important efficiency into a costly area of health care.

NASA Scheduling Problems NASA has a wealth of scheduling problems that have served to motivate our research. NASA's Data Information System for the Earth Observing System (EOSDIS) is concerned with analyzing, archiving, and retrieving scientific data for many present and future NASA projects. Scientists submit requests for data and analysis and EOSDIS schedules time on a world-wide network of computers and data archives to satisfy those requests. When the satellites of the Earth Observing System come on line, the National Space Science Data Center will be acquiring data at a rate of 2000 Gigabytes a day and responding to several thousand requests for both raw and analyzed data per year. Scheduling for these requests represents a formidable challenge.

NASA has a number of steerable, ground- and space-based automated telescopes. NASA needs a system that can take requests through the Internet from astronomers at their home institutions and then schedule observations and initiate control sequences at the remote telescopes to obtain the requested observations. The system needs to quickly respond to unforeseen events including hardware malfunctions and unanticipated opportunities to observe astronomical phenomena. This application requires concurrent scheduling and execution and an effective solution promises to reduce telescope operation costs and increases the number of requests that can be handled and their timeliness.

5 Conclusion

The work described in this report is a summary and synthesis of our recent work extending existing planning and scheduling techniques to handle high-dimensional state and action spaces in a time-critical setting. We have described algorithms and a general architecture to support dynamic planning, scheduling, and execution. The architecture and algorithms are general enough to apply to a wide range of problems modeled as stochastic dynamical systems with very large state and action spaces. We conclude this summary with an example strategic military airlift scenario that exemplifies the benefits of applying our technology to time-critical planning and scheduling problems.

5.1 Example Time-Critical Planning Scenario

It is early in the morning at Dover Air Force Base. A large-scale strategic airlift is in its second week, focusing on Dover as a point of embarkation for mobilization. Numerous force modules consisting of tanks, trucks, cargo, forces and supplies are converging at Dover from bases throughout the United States.

Currently, the Dover air traffic controller is monitoring several planes in their final descent. Earlier in the morning an operator working with the Air Traffic Management Advisory system set a landing sequence for these planes. Now the planes are following instructions from the Descent Advisor and Final Approach Spacing tools. The landings are almost entirely automatic; the air traffic controller and the pilots of the descending planes are checking visually for anomalies.

As is typical throughout the course of this mobilization, the local terminal traffic controller is preparing to assign staging areas for arriving and departing flights that require loading and unloading in the next two hours. Short-term aircraft flight schedules relevant to Dover have just been transmitted from the centralized scheduler of the Air Mobility Command at Scott Air Force Base. The local terminal traffic controller assisted by a local planning system uses the corresponding ETAs in conjunction with a local weather model to plan for the morning's traffic. Status at Dover (*e.g.,* information regarding severe weather or congestion) and the results of local planning decisions are transmitted back to the centralized flight scheduler at Scott. This procedure is repeated throughout the network of bases involved in this mobilization.

During the previous night, the flight scheduler at Scott simulated upcoming (daily and weekly) transportation activity throughout the network of bases. The simulations made use of movement models constructed from data from the OPLAN and TPFDD and weather forecasts throughout the country. Additionally, the flight scheduler took into account expected maintenance activities for aircraft throughout the network. Statistics gathered from these simulations were used to extract models of daily traffic patterns of the most relevant scenarios. The resulting statistics are now used to focus the on-line deliberations of the flight scheduler throughout the day. The profiles reveal a potential hot-spot at Dover around 1000 EST. The centralized scheduler allocates extra processing cycles to consider this bottleneck in order to plan for many possible contingencies. During the night, initial flight schedules for the day are precomputed based on projected conditions.

At 0800 EST a report is issued from overseas indicating that a returning C-141 requires routine maintenance. This situation was anticipated as a possible contingency in the initial flight schedules and the maintenance scheduler at Scott reacts to this information by re-routing this transport to McGuire rather than Dover in order to reduce parking congestion at Dover and take advantage of the special maintenance facilities of McGuire. Concurrently, the centralized scheduler reacts by re-sourcing a C-141 coming out of maintenance at Norton to Dover to satisfy the OPLAN. The replacement C-141 is scheduled from Norton through Travis to Dover. The aircraft is given an ETA of 2100 EST at Dover, which is time enough to satisfy the original OPLAN requirements.

Weather reports continually come in to Scott and are used to update the stochastic model for evaluating alternative flight schedules. Meanwhile, the centralized scheduler at Scott has constructed several contingent schedules for the 1000 EST period. These schedules are constructed by focusing on the states most likely to occur in that time period given the most recent weather, maintenance and other data. Reports from Dover specify the most recent capacity and delay information. Based on these reports Scott chooses one of the pre-constructed alternative flight schedules. The chosen schedule deals with congestion at Dover by holding low priority aircraft destined for Dover on the ground at McGuire. Scott transmits the updated flight schedules for 1000 EST activity to Dover and McGuire. The additional ground holds at McGuire are the least-cost solution to the congestion at Dover, taking into account all downstream constraints.

At 1300 EST severe weather reports come into Scott from Travis. The C-141 had to be rescheduled by local controllers to March Air Force Base and will not be able to make the 2100 EST rendezvous at Dover. Strategically, when re-sourcing the C-141 from Norton this flight leg was flagged by the centralized controller as a potential bottleneck and source of uncertainty given the weather conditions at Travis at 0800 EST and local stochastic model. Daily processing allocations were re-allocated at that time to focus on-line deliberation on constructing additional contingency flight schedules. The contingency plans resulting from this additional processing become available from the centralized scheduler at 1400 EST. The plans specify that a replacement for the C-141 grounded at March is available at Charleston. Unfortunately, this aircraft cannot make it to Dover until the next morning. The updated flight schedule takes this into account and grounds a personnel transport from McGuire so that the personnel may overnight at McGuire rather than further taxing the limited resources at Dover. Additionally, perishable supplies coming out of McChord are rescheduled to correspond with the arrival of the personnel. The additional parking space freed up by the missing C-141 allows a C-130 to overnight at Dover and undergo preventive maintenance.

As night falls the centralized scheduler begins to simulate airlift possibilities for the next day. A change in the OPLAN has been transmitted from Washington to the Air Mobility Command at Scott and is incorporated into the daily and weekly simulations. Some components of this strategic airlift application are depicted in Figure 6.

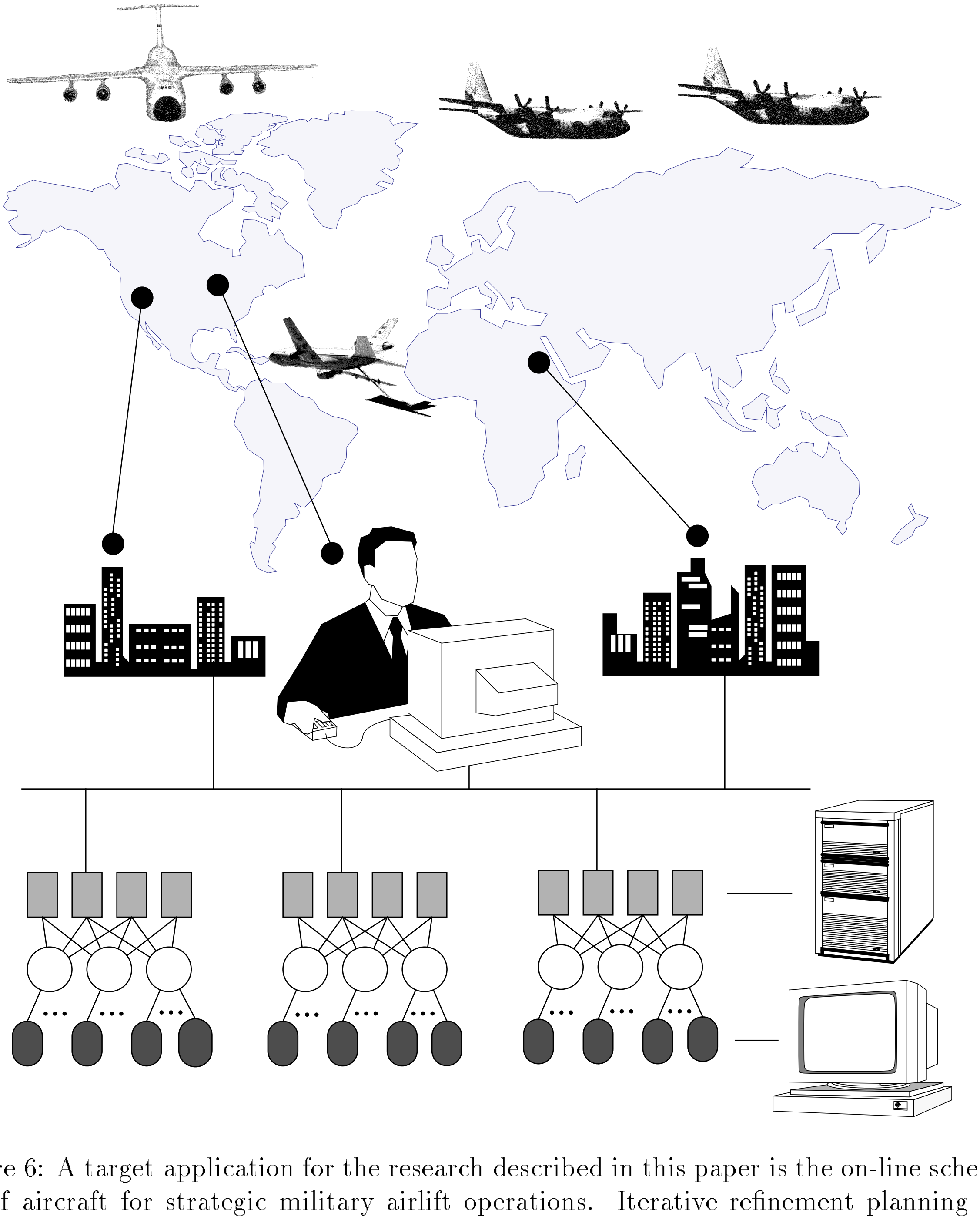


Figure 6: A target application for the research described in this paper is the on-line scheduling of aircraft for strategic military airlift operations. Iterative refinement planning and scheduling routines generate flight and maintenance schedules in a timely manner and continually revise them to account for delays due to weather and mechanical failure.

References

[1] J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3):391–401, 1988.

[2] Richard Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[3] Dimitri P. Bertsekas. *Dynamic Programming.* Prentice-Hall, Englewood Cliffs, N.J., 1987.

[4] Mark Boddy. Anytime problem solving using dynamic programming. In *Proceedings AAAI-91*, pages 738–743. AAAI, 1991.

[5] Mark Boddy and Thomas Dean. Solving time-dependent planning problems. In *Proceedings IJCAI 11*, pages 979–984. IJCAII, 1989.

[6] Mark Boddy and Thomas Dean. Decision-theoretic deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, to appear.

[7] Thomas Jr. Davis, H. Erzberger, S. M. Green, and W. Nedell. Design and evaluation of an air traffic control final approach spacing tool. *AIAA Journal of Guidance, Control, and Dynamics*, 14:848–854, 1991.

[8] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proceedings AAAI-88*, pages 49–54. AAAI, 1988.

[9] Thomas Dean, Leslie Kaelbling, Jak Kirman, and Ann Nicholson. Deliberation scheduling for time-critical sequential decision making. In *Proceedings of the Ninth Conference on Uncertainty in AI*, pages 309–316, 1993.

[10] Thomas Dean, Leslie Kaelbling, Jak Kirman, and Ann Nicholson. Planning under time constraints in stochastic domains. *submitted to Artificial Intelligence*, 1993.

[11] Thomas Dean, Leslie Kaelbling, Jak Kirman, and Ann Nicholson. Planning with deadlines in stochastic domains. In *Proceedings AAAI-93*, pages 574–579. AAAI, 1993.

[12] Thomas Dean and Keiji Kanazawa. Probabilistic temporal reasoning. In *Proceedings AAAI-88*, pages 524–528. AAAI, 1988.

[13] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[14] Thomas Dean and Keiji Kanazawa. Persistence and probabilistic inference. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(3):574–585, 1989.

[15] Thomas Dean and Michael Wellman. On the value of goals. In Josh Tenenberg, Jay Weber, and James Allen, editors, *Proceedings from the Rochester Planning Workshop: From Formal Systems to Practical Systems*, pages 129–140, 1989.

[16] Thomas Dean and Michael Wellman. *Planning and Control.* Morgan Kaufmann, San Mateo, California, 1991.

[17] Mark Drummond and John Bresina. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *Proceedings AAAI-90*, pages 138–144. AAAI, 1990.

[18] Michael P. Georgeff and Amy L. Lansky. Reactive reasoning and planning. In *Proceedings AAAI-87*, pages 677–682. AAAI, 1987.

[19] Lloyd Greenwald and Thomas Dean. Anticipating computational demands when solving time-critical decision-making problems. In *Workshop on the Algorithmic Foundations of Robotics*, 1994.

[20] Lloyd Greenwald and Thomas Dean. Deliberation scheduling for time-critical scheduling in stochastic domains. Technical Report CS-94-35, Brown University Department of Computer Science, Providence, RI, September 1994.

[21] Lloyd Greenwald and Thomas Dean. Monte carlo simulation and bottleneck-centered heuristics for time-critical scheduling in stochastic domains. In *ARPI Planning Initiative Workshop*, 1994.

[22] Lloyd Greenwald and Thomas Dean. Solving time-critical decision-making problems with predictable computational demands. In *Second International Conference on AI Planning Systems*, 1994.

[23] Ronald A. Howard. *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, Massachusetts, 1960.

[24] F. Kabanza. Synthesis of reactive plans for multi-path environments. In *Proceedings AAAI-90*, pages 164–169. AAAI, 1990.

[25] J. G. Kemeny and J. L. Snell. *Finite Markov Chains.* D. Van Nostrand, New York, 1960.

[26] Jak Kirman, Ann Nicholson, Moises Lejter, Thomas Dean, and Eugene Santos Jr. Using goals to find plans with high expected utility. In *Proceedings of the Second European Workshop on Planning*, 1993.

[27] N. Kushmerick, S. Hanks, and D. Weld. An Algorithm for Probabilistic Planning. Technical Report 93-06-03, University of Washington Department of Computer Science and Engineering, June 1993. To appear in *Artificial Intelligence.*

[28] Amy L. Lansky. Localized event-based reasoning for multiagent domains. *Computational Intelligence*, 4(4), 1988.

[29] Nicola Muscettola. An experimental analysis of bottleneck-centered opportunistic scheduling. In *Proceedings of the Second European Workshop on Planning*, 1993.

[30] Nils Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–88, 1986.

[31] P. S. Ow, S. F. Smith, and A. Thiriez. Reactive plan revision. In *Proceedings AAAI-88*. AAAI, 1988.

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

[33] Sylvie Thiebaux, Joachim Hertzberg, William Shoaf, and Moti Schneider. A stochastic model of actions and plans for for anytime planning under uncertainty. In E. Sandewall and C. Backstrom, editors, *Current Trends in AI Planning*. IOS Press, Amsterdam, 1994.