

Solving Time-Critical Decision-Making Problems with Predictable Computational Demands

Lloyd Greenwald*

lgg@cs.brown.edu

(401) 863-7662

Thomas Dean*

tld@cs.brown.edu

(401) 863-7645

Department of Computer Science

Brown University, Box 1910, Providence, RI 02912

Abstract

In this work we present an approach to solving time-critical decision-making problems by taking advantage of domain structure to expand the amount of time available for processing difficult combinatorial tasks. Our approach uses predictable variability in computational demands to allocate on-line deliberation time and exploits problem regularity and stochastic models of environmental dynamics to restrict attention to small subsets of the state space. This approach demonstrates how slow, high-level systems (*e.g.*, for planning and scheduling) might interact with faster, more reactive systems (*e.g.*, for real-time execution and monitoring) and enables us to generate timely solutions to difficult combinatorial planning and scheduling problems such as air traffic control.

Introduction

We are interested in the design of systems that make the best use of the time available for decision-making by explicitly accounting for the costs and benefits of computational delays. Applications such as air traffic control involve solving high-complexity planning and scheduling problems in dynamic situations. Dynamic situations imply that there are events outside the planner's control. These applications have time-critical properties that require timely responses to these events to ensure the safety of personnel and equipment and to avoid costly delays. They can be modeled as stochastic processes with very large state spaces and uncertainty in predicting future states. Such problems cannot be handled by search techniques that require enumerating states in the state space or state trajectories in the corresponding phase space (the product of the state space and the set of times considered).

*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.

The traditional approach to handling problems like this is to avoid exact combinatorial algorithms and expend effort constructing approximate algorithms and heuristics that are as fast as possible [Davis *et al.*, 1991]. Our work takes an orthogonal approach by concentrating on taking advantage of domain structure to expand the amount of time available for processing difficult combinatorial tasks. Additionally, we are able to restrict attention to a small subset of the state space without loss of optimality or with low probability of significant degradation of performance.

In the local air traffic control problem we are interested in scheduling runways and gates for aircraft at a busy airport such as Chicago's O'Hare. At any given time there are planes approaching the airport, landing on runways, loading and unloading passengers and cargo at gates, waiting for departure, waiting for repair, and taxiing down runways for take-off. The air traffic controller must take into account all this activity in order to schedule the runways and gates. The controller transmits speeds, headings and altitudes to the planes to satisfy the scheduling. As situations change these instructions may be modified. The objective is primarily to maintain the safety of passengers and equipment and secondarily satisfy performance measures such as minimizing fuel cost and delays and maximizing runway throughput.

The controller's task is most difficult during peak periods when the number of planes and possible varieties of situations is very large. During non-peak hours tasks can be solved with considerably less time and effort. Furthermore, the pattern of peak and non-peak hours is fairly regular throughout the week. For example, at 4am there are few planes arriving at O'Hare. The controller can construct optimal schedules for these planes and can modify them on demand based on weather changes and other uncontrollable events. At 6pm the situation is drastically different. The controller cannot hope to construct optimal schedules on demand. Some form of advanced planning is necessary. Advanced planning requires a model of upcoming traffic and uncontrollable events in order to get some idea of the possible situations that may require responses at

6pm. While 4am and 6pm vary greatly in both problem complexity and problem parameters, 6pm on Tuesday is of the same difficulty level as 6pm on Wednesday even if the actual combination of planes and other parameters varies. This regularity can be exploited in allocating on-line deliberation time between the 4am and 6pm task.

As can be seen in the above example, problem difficulty is not constant. The number of planes in the air varies throughout the day. Few planes in the air lead to relatively simple runway and gate assignment tasks. Brute force algorithms may be used to compute optimal assignments fast enough to provide real-time response to changing conditions. When the number of planes in the air grows, such as during traditional rush-hour periods, brute force algorithms can no longer provide fast enough response to avoid dangerous situations.

We make use of domain structure in two principal ways. First, we use a stochastic model of the domain and on-line knowledge of the state of the environment in order to predict future states. The ability to predict future states allows us to begin processing in advance of the actual states. Furthermore, careful prediction allows us to restrict our planning to a small subset of the state space. Prediction alone is insufficient to make progress with these types of problems. Second, we take advantage of large variabilities in computational demand across time. By removing the restriction of equal processor allocation for each state, we gain flexibility in allocating processing time. Additionally, some forms of regularity allow us to meta-reason off-line about the allocation of on-line deliberation time.

Response Planning Approach

We define *embedded planning* to be the problem of determining actions for an agent embedded in an uncertain environment with dynamics outside of the agent's control. A salient feature of this problem is that the agent must react to changes in the environment in *real-time*, *i.e.*, the agent must sense the state of the environment and act appropriately in a timely manner. There is rarely the luxury of waiting for a deliberative planner to construct the optimal action in response to environmental dynamics. There are hard deadlines imposed by the environment that, if violated, can lead to disastrous conclusions for the agent. In this work we provide an overview of the response planning approach to solving embedded planning problems. See [Greenwald and Dean, 1994] for more detailed data structures and algorithms.

In the case of local air traffic control the agent is the controller. The dynamics outside the agent's control include weather changes, plane arrivals in the controller's air space and delays due to mechanical and other failures. For each new set of problem parameters the controller must respond appropriately in a limited amount of time or risk the safety of passengers and

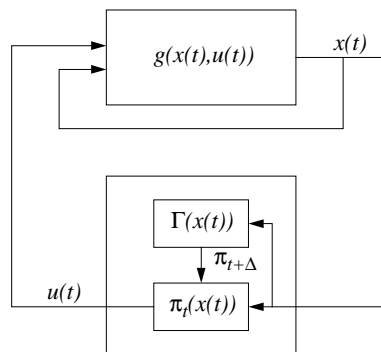


Figure 1: Model for embedded planning and control

equipment.

Each set of problem parameters may be considered a unique state. Response time is the minimum time between sensing a state and the necessity of response. Necessity of response is domain specific and is directly related to the rate change of the underlying dynamics. We assume for these purposes that response time is a fixed constant. We can then divide time into discrete time steps where a state is sensed at the beginning of each time step (the state may be the same as in the previous time step) and a response is required by the end of the time step (in some situations a null response may be satisfactory).

A solution that provides a response at each time step is called a *policy*. A policy π is a mapping from states to responses (actions). An air traffic controller may use a policy as part of a real-time control system that responds to changes in the environment. The focus of the response planning approach is to determine how to allocate on-line deliberation time to construct policies that satisfy particular performance criteria. Additionally, response planning provides an interface between the combinatorial process of constructing the policy and the reactive execution of the policy.

Figure 1 captures the essential properties of embedded planning and control systems; $x(t)$ is the state of the system (including the embedded agent) at time t , $u(t)$ is the action taken at time t by the composite planning and control system, and the function $g(x(t), u(t))$ determines the dynamics of the system. In this work we assume that we are dealing with a discrete time system. The action is determined by a policy that can be executed by the real-time control system. The policy π_t has a temporal index to indicate that it may change under the control of the planning component Γ . Due to the combinatorics involved in planning there is typically a delay Δ between when a state triggers the planning process and when the resulting policy is available for execution. We distinguish between state and control in keeping with standard practice in the control literature. For a description of the motivation and use of this formalism for modeling dynamical

systems see [Dean and Wellman, 1991, Gopal, 1985, Kalman *et al.*, 1969].

There are many options for designing the composite planning and control system signified by Γ and π . For example, a rigid solution to this problem is to design Γ as an off-line process that generates a policy for π that is executed sequentially without regard to the ensuing dynamics of the environment. In the air traffic control problem this is equivalent to having deterministic knowledge of the weather and plane arrivals and constructing the proper scheduling of all aircraft ahead of time. This is clearly not a useful solution in nondeterministic domains.

A similar solution is to design Γ as an off-line process that generates a table for π that specifies an optimal action (defined by some cost function) for every possible world state. This solution is only feasible in general given unlimited storage and computation power. Deliberative strategies similar to this are referred to as *universal plans* (see for example [Schoppers, 1987]). The huge number of possible combinations of planes, weather, delays and other parameters make this solution intractable for the air traffic control problem.

More appropriate is to design Γ as an on-line process that can adapt to the underlying dynamics. In a purely reactive design, Γ would be required to construct a policy at the onset of each time step (in response to a state) to be executed by π by the end of the time step. This would require that the delay, Δ , characterizing Γ 's behavior be less than the duration of a single time step. This method is useful in computationally non-intensive problems or slowly evolving systems. However, if the planning problem is combinatorial then the delay introduced in computing the policy often makes this solution infeasible.

Actually, the usefulness of this method may differ for different states within the same problem. For difficult states, such as during rush periods, in which many planes are in the air and may interact in numerous ways, the combinatorial nature of scheduling algorithms may preclude deriving a solution within a single time step. In other states, such as during night, Δ may be sufficiently small to allow this technique. This variability of Δ is the main motivation behind the response planning approach.

One further way to improve the employment of Γ on-line is to make use of predictive knowledge of the underlying domain. If we can predict the state of the environment for time t , $x(t)$, at some point in the past, $t' < t$, then Γ may begin constructing a policy for time t at time t' (in response to state $(x(t'))$). This gives an additional $t - t'$ time units for processing, if no other processing conflicts. Given a stochastic model of the environment we can predict a set of reachable states for time t . Thus, a complete policy must be able to handle more than one state. If prediction is poor or too far in advance the policy must handle the entire state space and is no better than a universal plan. However, if

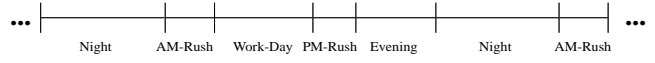


Figure 2: Temporal division of typical day into periods of equivalent difficulty

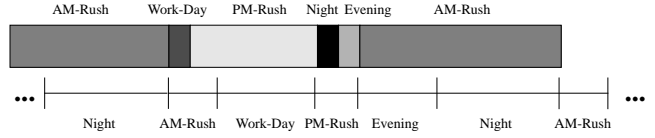


Figure 3: Example planner processor allocation for local air traffic control

prediction is sufficiently accurate we can restrict the planner's attention to a small subset of the states that could possibly occur at time t . How to use domain structure to determine an appropriate t' at which to plan for time t is one focus of response planning.

We have been discussing the case of computing a new policy for each time step. In general it may be advisable to compute a single policy (or set of policies) for a period of contiguous time steps. This takes advantage of similarities among reachable states in the period. For example, during the hours of 6am through 8am the distribution of planes in the sky and the expected delays remains relatively constant. Therefore, a comprehensive policy constructed to handle all reachable states at 6am may be applied to handle states at 8am and all times in between. Bear in mind that a policy need not explicitly and independently handle each possible combination of problem parameters. Rather, it must have a response to any given set of problem parameters (though, not necessarily unique or optimal). The grouping of contiguous time steps in this way corresponds to constructing temporal equivalence classes. We can hope to partition a problem into a sequence of equivalence classes.

As an example, consider the pattern of rush and calm in local air traffic control. We can divide the problem into periods similar to Figure 2. Each period corresponds to a different level of problem difficulty. We depict discrete periods here though in general the periods may not have strict boundaries. For any given period the controller must execute a response to each change in problem parameters. To simplify discussion, assume that Γ constructs the policy for each period prior to that time period, to be executed throughout that time period. For example, consider Figure 3. The policy for the PM-Rush period is computed during the AM-Rush and Work-Day periods and is available to the controller at the start of the PM-Rush. During the PM-Rush the controller (π) executes the given policy while the planner (Γ) computes policies for the Night and Evening periods.

As seen in the above example, the response planning approach requires two separate processors. One for

planning and one for execution. There is no attempt to merge these into a single thread of control.

Response planning is concerned with making the best possible use of on-line decision-making time by dividing a problem temporally and exploiting domain structure to borrow processing time from easy time periods to provide to difficult time periods. Combined with a stochastic model of the environment that provides for prediction under uncertainty, this enables us to provide combinatorial processing in advance of anticipated states and thereby achieve timely responses.

Γ is an on-line process that performs three disparate functions. It performs domain specific processing to compute the responses to be taken for sensed states (*i.e.*, constructs policies); it uses the underlying environment model g to predict reachable states in upcoming time periods; and, it reasons about its use of time based on processing requirements, predictions and cost measures. We divide Γ conceptually into two components. One we call f that performs prediction of reachable states and construction of policies and the other, that maintains the name Γ , that reasons about known properties of f and a given cost function and allocates deliberation (processing) time to f in order to provide the best possible performance.

The relationship between Γ , π and f is made explicit in Figure 4 and is summarized in the following procedure. At each time step t_j :

1. determine state s_j of environment g ;
2. execute policy indicated by π_j to determine response to state s_j ;
3. consult Γ to determine any new calls to f
4. if previous calls to f have terminated successfully since last time step, update π for those time steps by storing the constructed policies;

Note that new calls to f include parameters such as target time period and processing time allocated. This procedure is exemplified at the bottom of Figure 4. In this figure we show a new call to Γ at the beginning of each time period for simplicity. In general, Γ can instigate a new call to f in response to any state. In the figure we see Γ calling f during the Night to construct policies for AM-Rush. We further note that just prior to AM-Rush, f updates π with its constructed policy. This policy is then in effect until the end of AM-Rush at which point the policy for Work-Day takes effect.

As described, Γ senses the state at each time step and determines the optimal way to call f (allocate on-line deliberation time) to satisfy a given cost measure. For example, as in Figure 3, it may be optimal to solve the PM-Rush problem beginning at 7am and then solve the Night problem prior to the Evening problem. However, the process of allocating on-line deliberation time, often called *deliberation scheduling*, can itself be a combinatorial scheduling problem. If this were to be performed on-line it would compete with f for processing time. Furthermore, if it were to be performed on-line

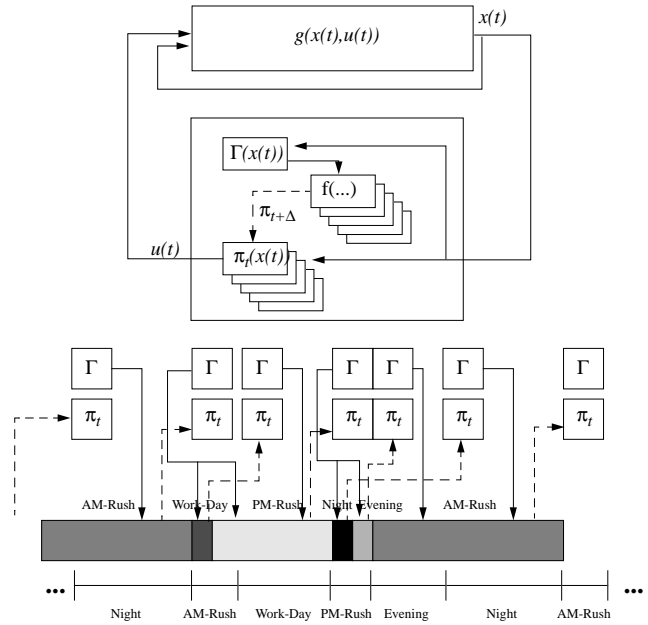


Figure 4: Response Planning model and example

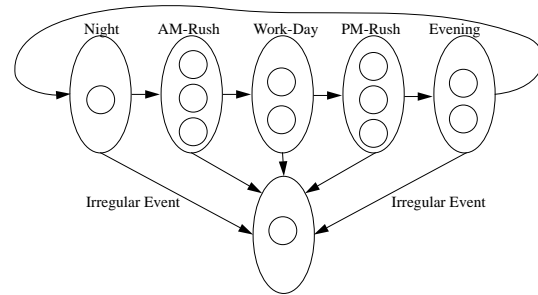


Figure 5: Regularity demonstrated by repeating sequence of equivalence classes; catastrophic events disrupt regularity

it may be subject to the same delays as f . We avoid this situation by using domain structure to construct Γ off-line into a *strategy table*.

One particular domain structure we are interested in is the *regularity* displayed in problems such as local air traffic control. As described earlier the problem can be partitioned into a sequence of equivalence classes of states corresponding to time periods. For example, regardless of which actual state occurs during AM-Rush it will be shortly followed by a state indicative of the Work-Day period. This regularity can be disrupted only by catastrophic events such as complete shutdown of the airport. As shown in Figure 5, as long as regularity is maintained the process will proceed in a fixed, possibly repeating sequence of equivalence classes of states. Catastrophic events force the process into a unique state that must be handled separately.

While we only have stochastic knowledge of the ac-

tual state trajectory, we have deterministic knowledge of this sequence of equivalence classes. We can use this knowledge in deliberation scheduling. In particular, we may have domain knowledge about the level of difficulty of each equivalence class as a function of when processing and prediction is commenced for the class. In cases for which the regularity is periodic we can attempt to determine cyclic schedules for the strategy table. In an example fixed cyclic schedule we may solve the PM-Rush problem beginning at 7am every day. In a non-cyclic schedule this may not be the case. More general models can be constructed in which problem regularity takes the form of a stochastic model or in smaller units that do not repeat in a single period. Strategy tables for these problems may still be handled off-line but they take on a more complicated form and require additional motivation beyond the scope of this presentation. Additionally, the problem becomes more involved if the responses constructed by f can alter the sequence of classes visited.

Related Research

This work continues our research into 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. It differs most from the approach used by Boddy and Dean on anytime algorithms [Dean and Boddy, 1988, Boddy and Dean, 1989] in that processing time and quality of on-line deliberation are a function of the time of commencement of processing and not just the time allocated on the processor. Furthermore, domain structure is employed to allow deliberation scheduling to take place off-line. Zilberstein [Zilberstein, 1993] has addressed similar issues while discussing dependent anytime components but our problem is more precisely stated and our results more concrete.

There have been a variety of planning systems that are related to the problem. Georgeff's *procedural reasoning system* [Georgeff and Lansky, 1987] 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 Drummond and Bresina [Drummond and Bresina, 1990], and Kushmerick, Hanks and Weld [Kushmerick *et al.*, 1993] do not address the problem of generating plans given time and quality constraints. Lansky [Lansky, 1988] has developed planning systems for deterministic domains that exploit structural properties of the state space to expedite planning. Ow *et al.* [Ow *et al.*, 1988] describe some initial efforts at building systems that modify plans incrementally. Neither Lansky or Ow *et al.*'s systems deal with uncertainty and Lansky's system cannot handle concurrent planning and execution.

Discussion and Conclusion

We have presented an approach to solving time-critical decision-making problems that takes advantage of domain structure to expand the amount of time available for processing difficult combinatorial tasks. We do this by using problem regularity to construct off-line a strategy table that allocates on-line deliberation time according to some cost measure. This approach is only possible in the face of variability in computational demands across time periods and the ability to use a stochastic model of the underlying environment to predict future reachable states. Judicious use of prediction and sequencing of processing tasks allows us to restrict attention to a small subset of the state space without loss of optimality or with low probability of significant degradation of performance.

Given that our stated goal is to expand the time available for processing difficult combinatorial tasks, one may be tempted to suggest that we instead throw more computing power at the problem. This solution is not feasible for the following reasons. Without prediction computation must be done in response to a each state. This allows a single time step to process difficult combinatorial tasks. In air traffic control this corresponds to rescheduling all planes in response to changing problem parameters. Even if rescheduling can be turned into an iterative process it is still combinatorial and must deal with a very large state space. We achieve large multiples of processing time with minimal prediction. Furthermore, throwing more computing power at the problem to handle the difficult tasks quickly becomes cost-ineffective. It has been mentioned that variability in processing difficulty is a key feature of applicable domains. In these cases the additional computing power would remain idle for a good portion of the time. Our response planning approach achieves optimal utilization of available resources.

The amount of processing time our technique affords to difficult periods must be much larger than a single time step. In order for this to be possible the increase in difficulty caused by additional prediction (starting processing earlier) must not increase rapidly for difficult periods. In particular, it must increase more slowly than the amount of time gained in prediction. Furthermore, the variability in processing difficulty must be great. A problem characterized by few difficult periods and many easy periods is ideal. The ability to trade quality for processing time by employing *anytime algorithms* as decision procedures (f) adds more flexibility to Γ and brings out the full power of this model. Finally, a cost measure that gives equal weight to difficult and easy periods, such that grossly suboptimal solutions during the difficult periods are acceptable, makes our technique inapplicable. Our techniques are especially applicable in domains in which increasing processing power is not an alternative due to power and space limitations, such as embedded plan-

ning in robotics.

The response planning approach we have outlined follows the lead of the real-time community by providing a formal model of computation for time-critical decision-making problems. The details of this research method are presented in [Greenwald and Dean, 1994].

References

- Boddy, Mark and Dean, Thomas 1989. Solving time-dependent planning problems. In *Proceedings IJCAI 11*. IJCAII. 979–984.
- Davis, Thomas Jr.; Erzberger, H.; Green, S. M.; and Nedell, W. 1991. Design and evaluation of an air traffic control final approach spacing tool. *AIAA Journal of Guidance, Control, and Dynamics* 14:848–854.
- Dean, Thomas and Boddy, Mark 1988. An analysis of time-dependent planning. In *Proceedings AAAI-88*. AAAI. 49–54.
- Dean, Thomas and Wellman, Michael 1991. *Planning and Control*. Morgan Kaufmann, San Mateo, California.
- Drummond, Mark and Bresina, John 1990. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *Proceedings AAAI-90*. AAAI. 138–144.
- Georgeff, Michael P. and Lansky, Amy L. 1987. Reactive reasoning and planning. In *Proceedings AAAI-87*. AAAI. 677–682.
- Gopal, M. 1985. *Modern Control System Theory*. Halsted Press, New York.
- Greenwald, Lloyd and Dean, Thomas 1994. Anticipating computational demands when solving time-critical decision-making problems. In *First Workshop on the Algorithmic Foundations of Robotics*.
- Kalman, R. E.; Falb, P. L.; and Arbib, M. A. 1969. *Topics in Mathematical System Theory*. McGraw-Hill, New York.
- Kushmerick, Nicholas; Hanks, Steve; and Weld, Daniel 1993. An algorithm for probabilistic planning. Unpublished Manuscript.
- Lansky, Amy L. 1988. Localized event-based reasoning for multiagent domains. *Computational Intelligence* 4(4).
- Ow, P. S.; Smith, S. F.; and Thiriez, A. 1988. Reactive plan revision. In *Proceedings AAAI-88*. AAAI.
- Schoppers, Marcel J. 1987. Universal plans for reactive robots in unpredictable environments. In *Proceedings IJCAI 10*. IJCAII. 1039–1046.
- Zilberstein, Shlomo 1993. *Operational Rationality through Compilation of Anytime Algorithms*. Ph.D. Dissertation, University of California at Berkeley.