# Solving Time-Dependent Planning Problems

Mark Boddy and Thomas Dean*
Department of Computer Science
Brown University, Box 1910
Providence, RI 02912

## Abstract

A planning problem is *time-dependent,* if the time spent planning affects the utility of the system's performance. In [Dean and Boddy, 1988], we define a framework for constructing solutions to time-dependent planning problems, called *expectation-driven iterative refinement.* In this paper, we analyze and solve a moderately complex time-dependent planning problem involving path planning for a mobile robot, as a way of exploring a methodology for applying expectation-driven iterative refinement. The fact that we construct a solution to the proposed problem without appealing to luck or extraordinary inspiration provides evidence that expectation-driven iterative refinement is an appropriate framework for solving time-dependent planning problems.

## 1 Introduction

We are interested in the problem of controlling an autonomous agent embedded in the real world ( i.e., a robot). In order to act effectively in a complex environment, the agent will most likely need to reason about sequences of predicted events, some corresponding to its own actions, some the result of other processes. We call this reasoning process *planning.* In some situations, the time required to plan may be considerable. Predicted events may provide constraints on the amount of time available or otherwise affect the cost of time spent planning. We say that a planning problem is *tunc-dependent* if the time spent planning has a cost ( i.e., affects the utility of the agent's performance).

In [Dean and Boddy, 1988], we define a framework for constructing solutions to time-dependent planning problems that we call *expectation-driven iterative refinement.* This framework is intended to restrict the form of the solutions generated, in order to make their analysis easier. Planning using expectation-driven iterative refinement is accomplished using a set of decision procedures known as *anytime algorithms.* Anytime algorithms are defined as algorithms that return some answer for any allocation of computation time, and are expected to return better answers when given more time. A solution to a time-dependent planning problem using expectation-driven iterative refinement will consist of a set of anytime algorithms for planning, and a *deliberation-scheduling* algorithm that allocates computational resources to the set of anytime algorithms based on expectations regarding their performance.

*This work was supported by an IBM graduate fellowship, by the National Science Foundation under grant IR1-8612644, and by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-88-C-0132.

In this paper, we take the first steps in developing a methodology for applying expectation-driven iterative refinement to time-dependent planning problems. As an example of what such a methodology should consist of, consider the use of divide-and-conquer as a strategy for solving combinatorial problems. Divide-and-conquer is a useful strategy for problems that can be decomposed, such that the resulting subproblems have substantially easier solutions. The literature offers a great deal of advice on how to go about decomposing a particular problem ( e.g., keeping the size of the subproblems approximately equal [Aho *ct a*/., 1974]). Problems that are not completely decomposable may have approximate divide-and-conquer solutions based on the introduction of appropriate assumptions ( e.g., Karp's partition for TSP [1977]). Analogously, a methodology for expectation-driven iterative refinement will require some way of identifying planning problems as likely candidates for expectation-driven iterative refinement, and a set of methods for generating solutions.

We proceed with the development of a methodology for expectation-driven iterative refinement by applying it to a moderately complex robot planning problem. The process of constructing a solution provides some insight into what a methodology for expectation-driven iterative refinement will look like. It also helps validate our claim that the framework itself is useful in that we construct a useful solution without appealing to luck or extraordinary inspiration. In the next section, we present a robot planning problem, and discuss how to break the problem down into pieces suitable for implementation as anytime algorithms. The rest of the paper describes our solution in detail.

## 2 The robot courier

Suppose that we are in charge of designing the control program for a new model of robot courier for a delivery service in Manhattan. The function of these couriers is to pick up small parcels and deliver them to specified locations. Assuming that a courier completes all of its assigned deliveries, the main factor in evaluating the courier's performance will be the speed with which it accomplishes this task. The faster deliveries are made, the more deliveries a courier can make, and the more income it will generate.

In any world where the robot does not have complete information, the exact utility of a particular course of action cannot be determined *a priori.* If we can formulate expectations regarding the exact state of the world (possibly conditioned upon information that the robot has some way of obtaining) then it is possible to calculate the *expected* utility of a proposed course of action. This is the metric on the robot's performance that we seek to maximize. The utility of the robot's performance we define in terms of the time required to complete the entire set of deliveries ( i.e., the less time, the better).

How might a robot courier plan, in order to do its job better? The robot must fix upon a tour that visits all of locations on its current list of deliveries. We refer to this as *tour improvement* planning. Choosing a tour may have considerable impact on the length of time required to visit all the locations. Once the robot has an ordering for the locations, it may spend time determining how to get from one to another of them. We refer to this as *path* planning. We assume that path planning is accomplished by constructing an ordered set of *target points* between the two locations. Planning in any greater detail we ignore. Arguably, the tasks involved in navigating between target points are sufficiently routine that "planning" is not normally required.[1] In addition, whatever work is done on achieving tasks at lower levels will be local in scope (determining which door to go in, avoiding bumping into things, *etc.)* and should not greatly affect the expected utility of tour improvement or path planning.[2]

The benefit to be expected from either tour improvement or path planning will depend on the current situation. Finding a tour that avoids the area around the Lincoln Tunnel at rush hour is most likely time well spent. On the other hand, if the next location to visit is straight down 5th Avenue from the robot's current position, there may be only a minor benefit to refining a path to get there. In either case, time spent planning may hurt more than it helps if it causes the robot to delay too long. Deliberation scheduling for the robot courier consists of allocating time to algorithms for tour improvement and path planning based on the expected improvement in the robot's performance. An *optimal* deliberation schedule for a given situation is a deliberation schedule that maximizes the expected utility of the robot's performance in that situation. An optimal deliberation scheduling algorithm always generates the optimal schedule for the current situation. All the deliberation scheduling algorithms discussed in this paper are optimal.

In the next section, we describe an experimental domain within which we have implemented a solution to the robot courier problem.

[1]See, for example, [Brooks, 1985].

We are not ignoring the possibility that the robot may fail at low-level tasks so as to require additional planning. The possibility of failure is included in the expected utility of the plans generated.

## 3 The gridworld

One of the salient characteristics of solutions using expectation-driven iterative refinement is their reliance on expectations to support deliberation scheduling. Using expectations is necessary because we have no way of determining the precise behavior of the planning algorithms employed by the robot, apart from actually running them. Accordingly, an important part of constructing a solution to the robot-courier problem consists of gathering statistics on which to base expectations regarding the anytime algorithms for path planning and tour improvement. For the purposes of this exercise we have designed a simulated world in which the required statistics can be gathered easily. We call this simulated world the *gridworld.* The gridworld consists of a rectangular subset of the integer plane ($\mathcal{Z}^2$), where each point is a location that may be occupied by the robot, or by something else, but not both. The robot can move onto any of the 4 neighbors of the point it currently occupies, provided that neighbor is not already occupied. We define the distance between adjacent locations to be one unit. The robot has a map of the world, and is capable of finding its way from one point to another without a planned path, by keeping track of the heading of the destination as it performs a form of obstacle avoidance. Path planning helps because a planned path may be more direct.

The statistics we needed were gathered over randomly-generated instances of the gridworld with a predetermined size and a fixed probability that any given location is occupied (.2). We will refer to these instances as *standard* gridworlds.

## 4 Anytime algorithms

In this section, we present and analyze anytime algorithms for path planning and tour improvement in the gridworld. Before proceeding with our analysis, we introduce some terminology and notation:

- $\mathcal{L}$ is a set of *locations,* defined as points in the integer plane ($\mathcal{Z}^2$).
- $\mathcal{S} = \langle l_1, l_2, \ldots, l_m \rangle$ is a *tour;* an ordered sequence of locations in $\mathcal{L}$.
- *dij* is the distance between locations $l_i$ and $l_j$. In the gridworld, $d_{i,j}$ equals the sum of the difference in *x* and $y$ coordinates between $l_i$ and $l_j$.
- *T>s* is the length of the tour *S* ($\mathcal{D}_\mathcal{S} = \sum_{i=1}^{m-1} d_{i,i+1}$).
- $v$ is the mean speed of the robot, moving without incident or obstacle. We assume that the time re-

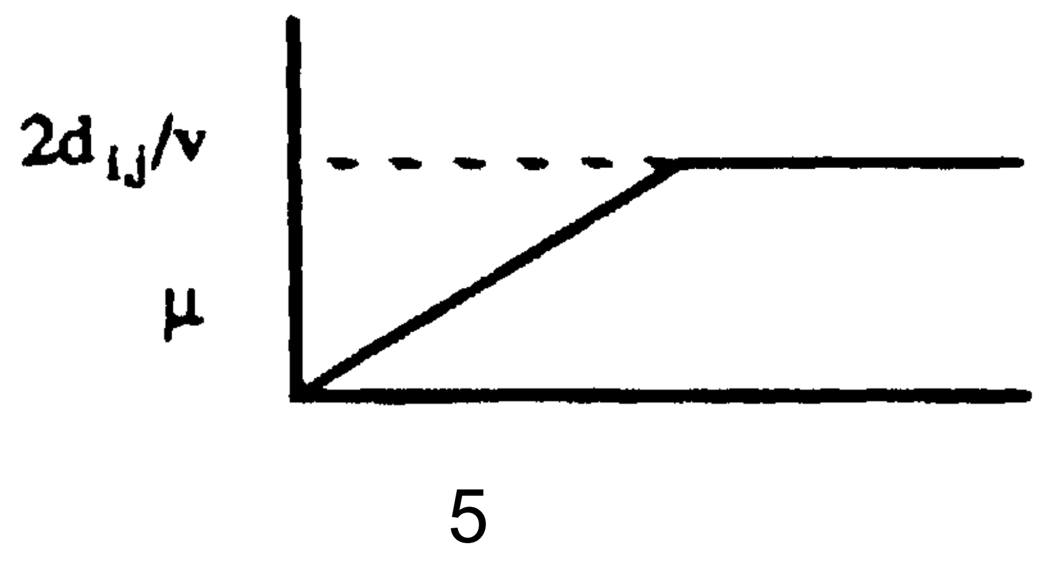

Figure 1: Performance profile for path planning

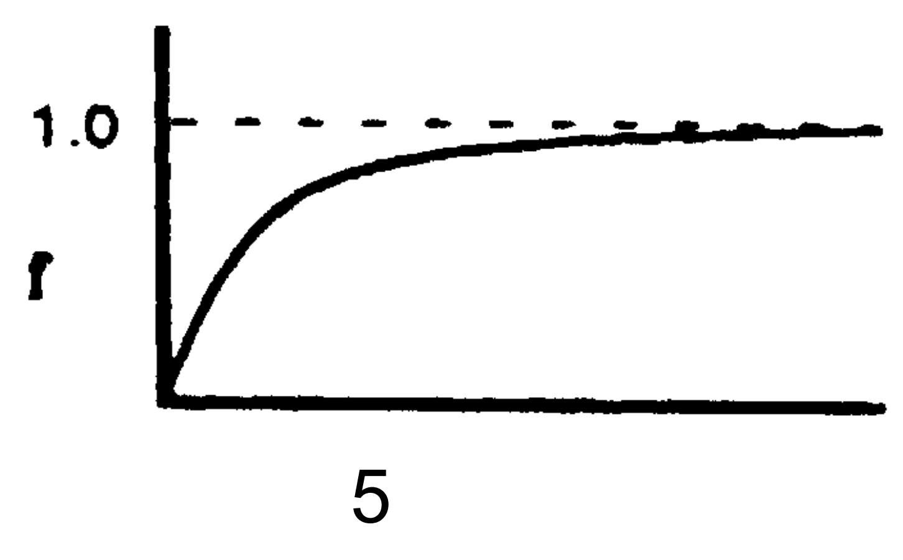

Figure 2: Performance profile for tour improvement

quired to attain this speed is small compared to the time the robot spends traveling between locations.

- *Tij(6)* is the expected amount of time required to go from $I_i$ to , given an amount of time *S* spent planning a path.
- *Tij* = *Tij(0)* is the expected amount of time required to go from /, to *lj*, without a path.

## 4.1 Path planning

The robot plans a path from one location to another by constructing an ordered set of adjacent locations (target points), starting at the initial location and working toward the destination. The algorithm used is a heuristic search similar to an algorithm descri bed in [Korf, 1987].[3] In the grid world, the algorithm when run to completion produces a path whose length averages 1.07 * the distance between the begin and end points, and examines a number of locations averaging 1.1 * the distance from one location to the other. Since it proceeds by searching from the initial point towards the destination, it can be interrupted at any time to return a partial path whose expected length is a linear function of the time spent planning, as is the expected distance remaining to the destination.

The robot can traverse a path at speed *v,* so that the expected time required to go from *l* to *l?* given a completely defined path is *a * (dij/v),* where *a* = 1.07. The mean time required to move from one location to another without a path ( i.e., working around obstacles as they are encountered) turns out to be a linear function of the distance between them. For standard gridworlds, the constant involved is approximately 3. Thus *Tij* = *b + dij/v,* where *b* « 3. In the interests of simplifying the analysis in the rest of this paper, we assume that *a* = 1 and 6 = 3.

We define /i(<5) to be *Tij* — *Tij (6),* the expected time saved by a partially defined path generated using an allocation of deliberation time *S.* Then /i(£) = ^ j , up to *6 = Tpfdij,* where $T_{pt}$ equals the expected amount of time required to add another target point to an existing path.[4] Figure 1 shows how the expected savings in travel

[3]Basically A*, using the distance to the goal as a heuristic evaluation function.

[4] We assume that the robot can work on a path between two locations only before it starts moving between them. This is a stronger assumption than we might like, but it can be relaxed fairly easily at the cost of complicating the deliberation scheduling algorithm (reschedule, breaking up the area to be traversed into planned and unplanned segments).

time varies with the time spent planning. We call such a function a *performance profile* for the anytime algorithm. The dependent variable for a performance profile may be any characteristic of the result of an algorithm that we find significant. One anytime algorithm could have several performance profiles tracking different attributes of the results it returns.

## 4.2 Tour improvement

We assume that the cost of a tour equals the total distance traversed. Though we make the assumption, it is not apparent that a minimum-distance tour will take minimal time to traverse, including planning time. In fact, it is possible to construct two tours such that the longer one is expected to take less time to traverse. However, our expectations regarding the result of tour improvement are in terms of the length of the entire tour, *not* the distribution of the distances between individual locations. Thus we can make no better assumption about the individual distances than that they are all equal. Given two tours of different lengths, both assumed to have all individual distances equal, the expected time to traverse the longer tour will be greater.

To construct a useful anytime algorithm, we need an initial answer whose cost is known, and some measure of how that cost is expected to improve with deliberation time. For tour improvement planning, the initial cost is the length of the initial tour (choose one at random, if necessary), and the optimal cost is the length of the optimal tour. The length of the optimal tour for n grid world locations can be approximated by 1.2d(n) , where *d* is the mean distance between the points on the tour, with a standard deviation of about 10% for tours of between 10 and 15 locations.

The next step is to find an anytime algorithm for tour improvement planning. Edge-exchange algorithms [Lin and Kernighan, 1973J produce tours that are progressively closer to optimal by exchanging small sets of edges (typically 2 or 3) such that the length of the overall tour decreases. It has been shown empirically that for large (100 city) tours, these algorithms average within about 8% of the optimal tour length when run to completion [Lawler *et* a/., 1985]. In the grid world, a tour-improvement algorithm that looks for pairs of edges to exchange averages within 2% of optimal when run to completion on considerably smaller tours (between 10 and 15 points). The mean number of exchanges necessary to obtain an almost-optimal tour for between 10 and 30 locations is approximately $(.61n - 2)^{1.1}$, with a mean error of less than 1% and a standard deviation of about 10%. The mean improvement in tour length after $k$ exchanges as a fraction of the maximum possible improvement can be approximated by a function of the form $f(k) = 1 - e^{-\lambda k}$, where $\lambda$ is a function of the size of the tour. For tours of 24 locations, a value for $\lambda$ of 0.29 results in a mean error and standard deviation both under 1%. Figure 2 gives a performance profile for tour improvement, showing the expected improvement in tour length as a fraction of the maximum possible, given an allocation of deliberation time $\delta$. Because the algorithm always looks for the best pairs of edges to exchange, the time required for any exchange is roughly constant for a given tour size.

Figure 2 is not a graph of the expected tour length over deliberation time. Given an initial tour length $\mathcal{D}_S$, an estimated optimal tour length $\mathcal{D}_{S^*}$, and an average length of time $t_e$ to make one exchange, the expected tour length as a function of time is $g(t) = \mathcal{D}_{S^*} + e^{-\lambda t/t_e}(\mathcal{D}_S - \mathcal{D}_{S^*})$. Given the assumption that the minimum-length tour will take the least time to plan for and traverse, minimizing expected tour length maximizes expected utility.

## 5 Deliberation scheduling

In this section, we discuss deliberation scheduling for three restricted versions of the robot-courier problem:

Pr-I. Deliberation scheduling for path planning for a fixed sequence of locations.

Pr-II. Deliberation scheduling for tour improvement followed by path planning for the improved tour.

Pr-III. Deliberation scheduling for a limited interleaving of tour improvement and path planning.

### 5.1 Pr-I

Assume a tour $\mathcal{S} = \langle l_0, l_1, \ldots, l_k \rangle$, where $l_0$ is the robot's current location. Let $\delta_i$ be the amount of time allocated to deliberating about the path from $l_{i-1}$ to $l_i$, for $1 \leq i \leq k$. Let $\mathsf{begin}(l_i)$ and $\mathsf{end}(l_i)$ be the begin and end points, respectively, of the period of time over which the robot is traveling from $l_{i-1}$ to $l_i$. This defines a time line with the following constraints:

- $\forall l_i : 1 \leq i \leq k, \mathsf{begin}(l_i) = \mathsf{end}(l_{i-1})$
- $\forall l_i : 1 \leq i \leq k, \mathsf{end}(l_i) = \mathsf{begin}(l_i) + T_{i-1,i}(\delta_i)$

We will also use the following definitions:

- Let $\delta_i^* = \tau_{pt} d_{i-1,i}$ – max. useful $\delta_i$.
- Let $c = \frac{2}{\tau_{pt} v}$ – slope of $\mu(\delta_i), 0 < \delta_i < \delta_i^*$.
- Using the results of Section 4.1:

$$\mu(\delta_i) = T_{i-1,i} - T_{i-1,i}(\delta_i) = \begin{cases} c\delta_i & \delta_i < \delta_i^* \\ 2(d/v) & \delta_i \geq \delta_i^* \end{cases}$$

The procedure given in Figure 3 is optimal in exactly the sense defined in Section 1. In other words, the algorithm generates a deliberation schedule for S such that

```
Procedure: Sched-1( $\mathcal{S}$ )
begin
  $t$ := end($l_k$)
  for $j$ = $k$ to 1, step $-1$
    begin
      if $t >$ begin($l_j$), then
         gap := max($c\delta_j^*$, $t-$ begin($l_j$))
         $t$ = begin($l_j$)
        else
         gap := 0
         $\delta_j$ := min($\delta_j^*$, $t-$ begin($l_1$), $\frac{1}{c+1}$($t-$ begin($l_1$)+ gap))
      end
      $t$ := $t - \delta_j$
      gap := max(0, gap $- c\delta_j$)
      if $t$ = begin($l_1$), goto exit
    end
 exit:: if $c > 1$ and gap $> 0$, then
      $\Delta$ := min($\delta_j^* - \delta_j$, $\frac{1}{c}*$ gap)
      $\delta_j$ := $\delta_j + \Delta$
      $\delta_i$ := 0, $1 \leq i < j$
      return $\Delta + \sum_1^k (T_{i-1,i}(\delta_i))$
end
```

Figure 3: An algorithm for Pr-I

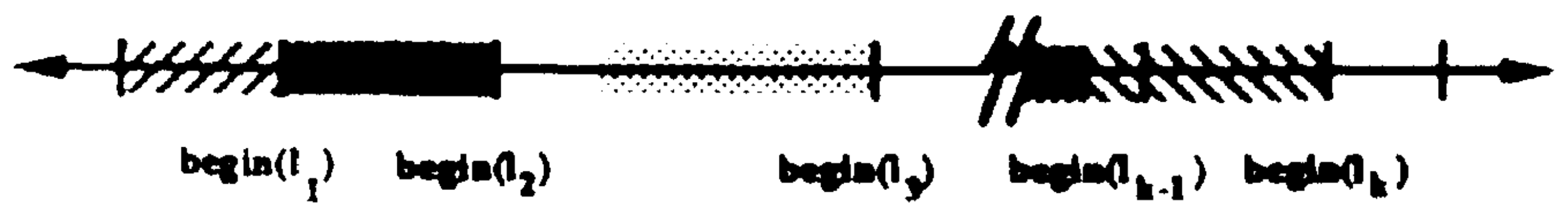

Figure 4: A deliberation schedule for path planning

there is no other schedule for which the robot is expected to complete the tour in less time. The procedure works backward from the last path to be traversed, ensuring that each path is allocated the maximum useful amount of time. There may be times where the robot is predicted to be traversing a path and all later paths have already been allocated as much time as will produce any improvement. The length of the earliest such predicted idle time is the value of the variable gap. The algorithm is simplified considerably by the fact that $c$ is the same for any path ( i.e., as long as $\delta_i < \delta_i^*$, it doesn't matter which $\delta_i$ gets allocated, for $1 < i < k$). That $c$ is a constant is a property of the grid world, not an assumption we make. Figure 4 shows a deliberation schedule constructed for an example tour. The shaded regions represent time allocated for deliberation.

### 5.2 Pr-II

Fr-II extends Pr-I by allowing S to be reordered, but only before any path planning gets done. In other words, if the initial tour is far from optimal, we expend some effort improving it before doing path planning. Given S, we can generate a deliberation schedule for path planning using Sched-1. Sched-1 returns the expected time to complete S, including path planning time. We also know the expected length of the tour $S'$ resulting from a given amount of time allocated to tour improvement. To calculate the expected utility of tour improvement, we need the expected time required, including path planning time, to complete $\mathcal{S}'$. We have no better information than the assumption that all the distances between locations in $S'$ are equal. So the path planning problem involves $k$ paths, all of length $d = \mathcal{D}_{S'}/k$.

In this case, the expected time to complete the tour given an optimal deliberation schedule can be expressed as a linear function of $d$ (and thus of $\mathcal{D}_{S'}$). There are four cases, depending on the value of c. The cases are distinguished by differing values of $c$ and $k$. In the first case, the expected benefit from path planning is so high that each path is allocated the maximum deliberation time. In the second case, the expected benefit is slightly less, so that setting $\delta_1$ to $\delta^*$ might make the tour take longer. In the third case, the expected benefit of path planning is low enough that any value of $\delta_1 > 0$ has a negative utility, and that only 6k is automatically allocated the maximum amount of time. In the last case, the improvement to be expected is so small that only the last path in the tour is worth planning for, and that only while traveling among the other locations.[5]

For a particular domain, $c$ and $v$ are constants. $k$ is fixed for a given set of locations. Let $f(d)$ be the appropriate linear function of $d$, which we will write as *ad.* Let $g(\mathcal{S}, \delta')$ be the expected length of the tour returned by the anytime algorithm for tour improvement, allocated $\delta'$ units of time. Then the following procedure is an optimal deliberation scheduling algorithm.

```
Procedure: Sched-2( S)
begin
  Choose $\delta'$ s.t. $\mathcal{T} = \delta' + (\alpha * g(\mathcal{S}, \delta')/k)$ is minimized.
  if Sched-1( $\mathcal{S}$) < $\mathcal{T}$, then $\delta' = 0$.
     return $\delta'$
end
```

Once the tour improvement algorithm has been run, we run Sched-1 on the resulting tour. That the required minimization can be done easily is guaranteed by the fact that $g'$ is always negative and $g''$ is always positive (see Section 4.2); there are no local minima.

### 5.3 Pr-III

Interleaving tour improvement and path planning arbitrarily can be complex; there are an exponential number of ways in which a given tour might be subdivided into pieces on which to run the tour improvement algorithm. There is a useful special case, however, for which we can construct a deliberation scheduling algorithm whose complexity is $O(n^2)$ in the number of locations. We define the scheduling problem Pr-111 to consist of determining some number of locations $l_0, \ldots, l_i$, for which we plan paths first, followed by the remaining locations, which are treated as an instance of problem Pr-11. An optimal scheduling algorithm for Pr-III is given in the longer paper.

[5]The exact form of the relation between distance and expected completion time for each of these cases is given in [Boddy and Dean, 1989], an extended version of this paper.

The algorithm works by assuming initially that all the travel time for the locations $l_0, \ldots, l_i$ will be used to deliberate about finding an improved tour for $l_i, \ldots, l_k$. Using a procedure almost identical to Sched-1, the algorithm then sweeps back from begin$(l_i)$, determining for each increment of time whether spending time on tour improvement or path planning is expected to save more time in the final schedule.

## 6 Related work

There is a growing interest in applying decision theory to AI problems. [Horvitz *et a/.*, 1988] sketches the current state of this effort, and points out some areas where further research might be most fruitful. In addition to our work on planning, the application of decision theory to the control of resource-bounded reasoning is currently being pursued in at least the areas of medical diagnosis [Horvitz, 1987], heuristic search [Hanson and Meyer, 1988], [Russell and Wefald, 1989], and computer vision [Levitt *et a/.*, 1988]. Breese and Fehling [1988] provide an abstract characterization for the decision-theoretic control of an autonomous agent, including reasoning about tradeoffs among planning, acting, and gathering information.

There is also some interesting work in other areas. In building his ping-pong playing robot, Andersson [1988] solves special cases of exactly the kind of problems we are interested in. It is not clear to what extent his techniques will generalize, but they are obviously quite successful for one particular domain. Wellman [1988] provides an extended analysis of the use of uncertain knowledge in planning, which is a problem that the control system for any reasonably complex agent must address. The work in [Ow *et al.,* 1988] deals with the problem of resource bounds by using the analysis of a specific domain to guide the construction of strategies for controlling reasoning in that domain.

## 7 Conclusions

Accounting for the time required for planning or other complex computations is a necessary part of constructing systems for controlling robots. In [Dean and Boddy, 1988], we define a framework for solutions to time-dependent planning problems called expectation-driven iterative refinement. This framework is intended to restrict the form of those solutions to facilitate decision-theoretic analysis. In the current paper, we are interested in a methodology for applying expectation-driven iterative refinement. Specifically, we want a way to identify problems as likely candidates and a body of techniques for generating solutions. We detail the process of solving a time-dependent planning problem involving a mobile robot courier, as a way of investigating such a methodology.

There are several lessons to be taken from this paper on how to go about solving a time-dependent planning problem. For one, we do not "solve" either of the planning problems facing the robot. We provide approximate solutions, and some measure of the errors to be expected. A problem where a precise answer is critical would probably use different algorithms. An interesting point that arises in the generation of performance profiles is that we need a fast estimate of how good an answer is obtainable. This estimation will of necessity introduce some error (otherwise the anytime algorithm would be superfluous) and we should know how much. Assuming that the anytime algorithms can be represented by continuous performance profiles does not for this problem introduce any large error—the increments in which planning is done are small enough that the approximation is a reasonable one. The form of the solution we provide is strongly determined by the structure of the environment within which the robot operates. The environment determines the performance profiles of the anytime algorithms the robot employs for planning, and to a lesser extent the algorithms themselves.

The process of generating a solution provides some indication of the characteristics'of the methodology we are developing. Good candidates for solution using expectation-driven iterative refinement will be problems that can be decomposed into subproblems that can be solved using anytime algorithms for which we can generate useful expectations in the form of performance profiles. There is an additional restriction, which is that the problem must be decomposed in such a way that we know how to combine the profiles for the subproblems so as to produce a performance profile for time spent on the problem as a whole.

Solving even a moderately complex time-dependent planning problem requires combining the results of several processes. In some problems, there may be several unrelated subproblems competing for processor time. For the robot courier, this would correspond to allocating planning time for the individual paths on a tour. Also, most useful anytime algorithms we know of apply to sufficiently simple problems that any interesting planning problem will require combining the results of several anytime algorithms. In order to plan for a tour, our robot courier needs to combine the results of path planning and tour improvement. These are not separate competing processes. The expected utility of time spent path planning depends on the result of tour improvement. In our example, combining expectations for the two planning algorithms is straightforward: they can be composed, given an assumption that is justified by the information available to the robot. Other problems and other decompositions will require combining expectations in different ways.

In this and earlier papers, we have analyzed a few special cases of time-dependent planning problems. In future work, we hope to provide a more general classification of time-dependent planning problems, and to analyze the various classes in terms of their solution using expectation-driven iterative refinement.

## References

[Aho *et al*, 1974] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Massachusetts, 1974.

[Andersson, 1988] Russell L. Andersson. *A Robot Ping-Pong Player: Experiment in Real-Time Intelligent Control.* MIT Press, Cambridge, Massachusetts, 1988.

[Boddy and Dean, 1989] Mark Boddy and Thomas Dean. Solving time-dependent planning problems. Technical Report CS-89-03, Brown University Department of Computer Science, 1989.

[Brccse and Fehling, 1988] John S. Breese and Michael R. Fehling. Decision-theoretic control of problem solving: Principles and architecture. In *Proceedings of the 1988 Workshop on Uncertainty in Artificial Intelligence,* 1988.

[Brooks, 1985] Rodney A. Brooks. A robust layered control system for a mobile robot. A.I. Memo No. 864, MIT Artificial Intelligence Laboratory, 1985.

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

[Hanson and Meyer, 1988] Othar Hanson and Andrew Meyer. The optimality of satisficing solutions. In *Proceedings of the 1988 Workshop on Uncertainty in Artificial Intelligence,* 1988.

[Horvitz *et al,* 1988] Eric J. Horvitz, John S. Breese, and Max Henrion. Decision theory in expert systems and artificial intelligence. *Journal of Approximate Reasoning,* 1988.

[Horvitz, 1987] Eric J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of the 1987 Workshop on Uncertainty in Artificial Intelligence,* 1987.

[Karp, 1977] R. Karp. Probabilistic analysis of partitioning algorithms for the traveling salesman problem in the plane. *Math. Oper. Res.,* 2:209-224, 1977.

[Korf, 1987] Richard E. Korf. Real-time path planning. In *Proceedings DA RPA Knowledge-Based Planning Workshop^* 1987.

[Lawler *el al,* 1985] E.L. L awler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. *The Travelling Salesman Problem.* Wiley, New York, 1985.

[Levitt *et al,* 1988] Tod Levitt, Tl lomas Bin ford, Gil Ettinger, and Patrice Gelband. Utility-based control for computer vision. In *Proceedings of the 1988 Workshop on Uncertainty in Artificial Intelligence,* 1988.

[Lin and Kernighan, 1973] S. Lin and B. W. Kernighan. An effective heuristic for the travelling salesman problem. *Operations Research,* 21:498 516, 1973.

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

[Russell and Wefald, 1989] Stuart Russell and Eric Wefald. Principles of metareasoning. In *First International Conference on Principles of Knowledge Representation and Reasoning,* 1989.

[Well man, 1988] Michael P. Well man. Formulation of tradeoffs in planning under uncertainty. Technical Report MIT/LCS/TR-427, MIT AI Laboratory, 1988.