

# Approaches to macro decompositions of large Markov decision process planning problems

Terran Lane and Leslie Pack Kaelbling

MIT Artificial Intelligence Laboratory, Cambridge, MA, 02139

## ABSTRACT

Mobile robot navigation tasks are subject to motion stochasticity arising from the robot’s local controllers, which casts the navigational task into a Markov decision process framework. The MDP may, however, be intractably large; in this work we consider the prioritized package delivery problem which yields an exponentially large state space. We demonstrate that the bulk of this state space is tied to a sub-problem that is an instance of the traveling salesdroid problem and that exponential improvements in solution time for the MDP can be achieved by addressing the TSP sub-problem separately. This process produces a suboptimal solution, but we show that the degree of suboptimality can be controlled by employing more effective TSP approximators. The key contribution is the demonstration that MDP solution techniques can substantially benefit from careful application of well-understood deterministic optimization techniques.

**Keywords:** Mobile Robotics, Navigation, Stochastic Planning, Markov Decision Processes, Traveling Salesdroid Problem, Prize-Collecting Minimum Latency TSP

## 1. INTRODUCTION

Markov decision processes are an appealing formalism for modeling planning and learning tasks in stochastic domains, including mobile robotics. Unfortunately, solution techniques for MDPs often require more computational overhead than is practical in many systems with runtime constraints. Furthermore, even intuitively simple tasks may yield extremely complex decision processes (e.g., exponentially large) whose solution requires resources that are infeasible even for dedicated, non-realtime systems. A number of researchers have recently proposed approaches to working with such complex models by attempting to exploit various classes of special structure in the model<sup>1-3</sup> but, in general, there is no guarantee that compactly representable MDPs possess a compactly representable optimal plan. Additionally, it may be difficult to generalize the solution of a single MDP to related models; by default, even a small change in the problem’s dynamics or reward model requires complete re-computation of the plan from scratch. In this paper, we attempt to address these issues for a special class of Markov decision processes. Many important MDPs contain embedded deterministic computational problems with known properties and we can achieve significant improvements in solution speed and plan reusability by extracting the known sub-problem from the MDP model, solving it in isolation, and recombining its solution with the solution of the “stochastic” component of the MDP to yield a final plan. The resulting plan is likely to be suboptimal, but in the domains of interest it is utterly infeasible to obtain an optimal solution.

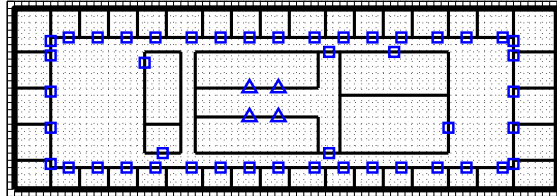
While the notion of working with embedded known problems is quite general, in practice we must address particular instances by recognizing such problems manually and developing specific algorithmic approaches to each. In this paper, we demonstrate such a development for a class of *package delivery* MDPs which contain the traveling salesdroid problem (TSP)<sup>4</sup> or the less well understood minimum latency TSP (MLP).<sup>5,6</sup> This work is an extension of an earlier study of extracting shortest-path solutions from MDPs.<sup>7</sup> In that work, we were also concerned with “spatial” decompositions of the navigational component of the MDP; this paper does not consider that form of hierarchical decomposition, but we note that it could be applied in conjunction with the techniques we demonstrate here.

---

Author email addresses: terran@ai.mit.edu; lpk@ai.mit.edu

## 2. PROBLEM DESCRIPTION

We begin by introducing the package delivery problem that motivates this branch of work. It turns out that there are no personal mailboxes in the MIT AI lab; instead, packages are delivered directly to the recipient’s office. Some packages have higher priority (e.g., FedEx deliveries) or higher cost of delivery (say, large computer components). We would like a mobile robotic agent to be able to carry out this task, delivering packages in a timely fashion while attempting to respect the relative package priorities. The particular mix of packages and priorities may change from day to day, though the world map remains fixed, and we would like to be able to take advantage of this by amortizing some planning effort across these problem-solving *episodes* (the execution time from picking up a single set of packages with given priorities through delivery of the final package in that set). We will describe the generation of fixed spatial navigation macros that can be reused across episodes and the use of a combinatorial optimization process to select between those macros within a single episode.



**Figure 1.** Schematic depiction of the target domain: one floor of the MIT AI lab. Dark lines denote walls, grid lines indicate location quantization, squares depict doorways, and the triangles note the elevator locations.

A coarse model of this domain is displayed in Figure 1. This is a representation of one floor of the building housing the AI lab; the thick lines denote walls, the squares mark doorways, and the triangles indicate the elevators. The floor is divided into 1875 physical locations, as indicated by the light grid, and there is a delivery location in each office (a total of 45 “delivery” spots, including the mail pickup location near the elevators). We assume, in this work, that the robot’s location is known exactly (localized to a single grid cell), but that its actions have stochastic effects—atomic actions in this model are movements in the four cardinal directions, but they may fail or deposit the robot in an adjacent square with some probability. In the full model that we’re currently studying, there are 10 floors, similar to this one, and we account for factors such as lockable doors, battery power, and the robot’s orientation. For this paper, though, we restrict our attention to the simpler model.

Clearly, the pure “navigational” task is not challenging—MDPs with 1875 states are easily within the reach of well-known solution methods.<sup>8</sup> The difficult part of this domain is that optimal action depends on knowledge of which packages have already been delivered, so the agent’s state representation must include a set of bits indicating which are still outstanding (45 in this case). This boosts the state space of this domain to roughly  $2^{45}$  states—now well beyond straightforward methods. We note, though, that the raw size of the state space does not reflect the true difficulty of the problem. The bulk of the state space is in the indicator bits which are, in some sense, independent and non-stochastic—the delivery of one package does not affect which other packages are outstanding, and package delivery occurs automatically and with probability 1 when the appropriate geographic location is reached. Furthermore, the dynamics of geographic navigation remained fixed across problem-solving episodes.

It seems intuitively plausible that the optimal solution to this problem is some form of a TSP tour and that one could approach this domain by approximating the TSP rather than solving the MDP directly. In fact, we show that the simplest model of utility for this domain *does* yield a TSP, but that when we consider package priorities, we are instead faced with a minimum latency TSP—a considerably less well understood and practically more difficult problem, but one whose approximate solution is still far more tractable than the direct solution of the full MDP.

## 3. FORMAL MODELS

In this section, we give a brief background on Markov decision processes and present the notation we use throughout this paper. For a more thorough treatment of the subject, we refer the reader to Puterman’s text.<sup>8</sup> We then give a formal specification of our package delivery problem.

### 3.1. Markov Decision Processes

A Markov decision process,  $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R \rangle$ , is a description of a synchronous control domain specified by four components: a *state space*,  $\mathcal{S} = \{s_1, s_2, \dots, s_N\}$ , of cardinality  $|\mathcal{S}| = N$ ; a set of primitive *actions*,  $\mathcal{A} = \{a_1, a_2, \dots, a_k\}$ , of cardinality  $|\mathcal{A}| = k$ ; a *transition function*,  $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ ; and a *reward function*,  $R : \mathcal{S} \rightarrow \mathbb{R}$ . The transition function, written  $T(s'|s, a)$ , determines the probability of arriving in state  $s'$  upon taking action  $a$  from state  $s$  and must represent a valid probability distribution:  $\forall s, a \sum_{s' \in \mathcal{S}} T(s'|s, a) = 1$ . An agent acting in a given MDP is, at any time step, located at a single state  $s \in \mathcal{S}$ . The agent chooses an action  $a \in \mathcal{A}$  and is relocated to a new state,  $s'$ , determined by the transition probability distribution  $T(s'|s, a)$ , whereupon it receives reward  $R(s')$ . The goal of the agent is to maximize its aggregated reward over time. Typically, we wish the agent to attain some sense of the “shortest” path necessary to achieve the specified goals, so we include a temporal penalty in the reward aggregation function. In this work, we will address two types of temporal penalty: *additive* rewards, in which each step that the agent takes in the world incurs some small cost,  $\epsilon > 0$ , and *multiplicative* penalties in which the value of rewards is discounted by a factor of  $0 \leq \gamma < 1$  for each step taken in reaching it. In the first case, the total aggregated reward received by an agent is

$$r_{\text{tot}} = \sum_{t=0}^T (R(s_t) - \epsilon)$$

and in the second, it is

$$r_{\text{tot}} = \sum_{t=0}^T R(s_t) \gamma^t$$

where  $s_t$  is the state reached by the agent at timestep  $t$ . Both cases are widely used in practice and we will show that they lead to a common solution framework using our methods. In general,  $\epsilon$  or  $\gamma$  may also depend on the state; reformulating the above equations to reflect this is straightforward, though we do not address that extension in this work. These equations reflect a finite operational lifetime; we can model an unterminating process simply in the multiplicative case but we require a zero-penalty absorbing state in the additive case to prevent divergence of the sum.

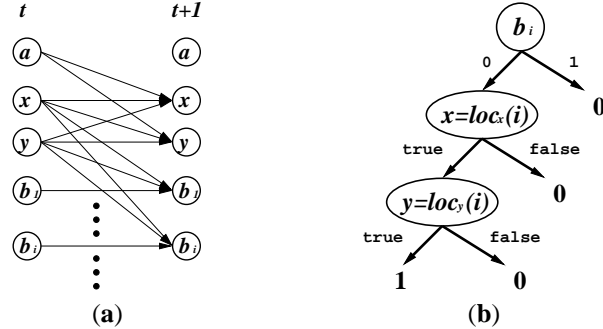
The goal of planning in an MDP context is to locate a *policy*,  $\pi : \mathcal{S} \rightarrow \mathcal{A}$ , that determines an action for the agent for any possible state. In general,  $\pi$  can specify a distribution over actions but an important theorem of MDP planning says that, for a fixed MDP,  $\mathcal{M}$ , there is always a *deterministic* policy,  $\pi_{\mathcal{M}}^*$ , that maximizes our expected aggregate reward,  $E[r_{\text{tot}}]$ .<sup>8</sup> Associated with a given MDP and policy is a *value function*,  $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$ , which records the total expected aggregate reward achieved by an agent that starts in state  $s$  and acts according to  $\pi$  for its operational lifetime. The value function for a particular MDP and policy can be found exactly by solving the *Bellman equations*, requiring  $O(N^3)$  time and  $O(N^2)$  space for a model with  $N$  states. The difficulty is that the state space is often best expressed as a Cartesian product of *state variables* and is, therefore, exponentially large in the number of variables.

### 3.2. Domain Model

We model the package delivery problem with an MDP that represents the world described in Section 2. The state space is defined by two spatial variables,  $x$  and  $y$ , and  $k = 45$  bits,  $b_1 \dots b_k$  where  $b_i = 0$  iff package  $i$  has not yet been delivered. We assume that local controllers are available that can take the robot between adjacent cells (except where blocked by walls) so the primitive action space is  $\{\text{NORTH, SOUTH, EAST, WEST}\}$ .<sup>\*</sup> The transition function is compactly represented by a dynamic Bayes network<sup>2</sup> whose structure is given in Figure 2 (a). This structure encodes the intuitive notion that the robot’s position,  $\langle x, y \rangle$ , at time  $t+1$  depends only on its position at time  $t$  and the action taken,  $a$ , but not on which packages have been delivered. The delivery state of package  $i$ , in turn, depends only on the previous state of the package and the robot’s location but not on the state of any other packages. (It is also independent of the action, though this is not relevant to our current development.) The probability tables on the  $\langle x, y \rangle$  arcs model simple local movement controllers and are defined to allow the robot to move by single cells in the indicated directions. On each movement, there is some small chance of failure which deposits the robot in one of the orthogonal cells, chosen uniformly at random. The probability of passing through a wall is 0. The probability tables for indicator bit  $b_i$  sets the bit to 1 when the robot reaches the delivery location for package  $i$ , denoted  $\text{loc}(i)$ , and prevents the bit from ever being set back to 0 (i.e., the package can be delivered only once).

While the world dynamic, as defined by the state space, action space, and transition function, is fixed across episodes, the mix of packages and priority, as expressed by the reward function, differs between episodes (within a single episode, all

<sup>\*</sup>Handling non-holonomic robots, more finely quantized location, or even continuous location variables makes the solution of the geographic problem more difficult, but does not fundamentally change the nature of our approach.



**Figure 2.** Sketches of the (a) dynamic Bayes net representation of transition function and (b) decision tree representation of the reward function component for package  $i$ .

components of the MDP remain fixed). We capture flexibility of priorities by writing the reward function as a linear combination of reward basis functions, or components, each corresponding to the delivery of a single package:

$$R(s) = \sum_{i=0}^k w_i R_i(s) . \quad (1)$$

The per-episode package weight,  $w_i$ , expresses the current priority of package  $i$  with  $w_i = 0$  indicating that there is no package for location  $i$  in this episode. For this work, we restrict the  $w_i$ 's to be non-negative (i.e., it is never detrimental to deliver a package). Each reward component is fixed across all episodes and encodes the intuitive notion that the robot is rewarded for successfully delivering a package. Such a component can be expressed, for example, by the decision tree given in Figure 2 (b). Note that the combination of this tree with the structure of the transition function ensures that the robot receives reward for each package only once. This particular tree is appropriate for a multiplicative discounting framework; for an additive discount, the rule is simply  $R_i(s) = -\epsilon$  when  $b_i = 0$  and  $R_i(s) = 0$  when  $b_i = 1$ . We will also discuss the degenerate case in which the robot receives a single reward only when *all* packages have been delivered. In this case, the per-episode weights are irrelevant as there is no notion of relative package priority.

#### 4. ATTACKING THE EXPONENTIAL STATE SPACE

It is clear that the problem described in the previous section possesses a considerable degree of special structure. In particular, although the state space contains  $O(|x||y|2^k)$  states, the “spatial navigation” sub-problem, existing in the  $\langle x, y \rangle$  subspace, is independent of the value of the indicator bits and is relatively straightforward to solve.<sup>†</sup> For our target problem, at least, the spatial subdomain is tractable for known solution methods. (When this subspace is intractably large itself, we can resort to spatial hierarchical decompositions.<sup>9-13</sup>) The *difficult* part of this domain is in the factor of  $2^k$  arising from the indicator bits, but these bits are independent in the sense that the setting of one bit doesn't influence the optimal plan for changing a different bit in any way. We will exploit these properties in this section to demonstrate how to extract a known combinatorial optimization problem (the minimum latency TSP) from the MDP and use its (approximate) solution to craft a global plan.

##### 4.1. The Combinatorial Optimization Sub-problem

We begin by examining the simplest case in which the robot receives reward *only* for delivering *all* packages under the additive cost model. We will model this with the reward function,  $R(s) = -\epsilon$  if any  $b_i = 0$ . Because the values of the indicator bits are coupled only to the location variables, the condition  $b_i = 1$  for all  $i$  will be achieved when all the locations  $\text{loc}(i)$  have been reached at least once.<sup>‡</sup> The location variables, in turn, are coupled only to each other and the action, so the number of atomic action steps necessary to pass between them is independent of the settings of the  $b_i$ . Suppose that the policy  $\pi$  tours all locations in some order. Let the random variable  $n_\pi(i, j)$  be the number of steps taken by an agent executing policy  $\pi$  when

<sup>†</sup>By “solve” we mean determine an optimal policy for reaching a particular spatial location from any other. For an approach to solving the all-pairs policy construction problem for spatial domains, see the work by Moore et al.<sup>9</sup>

<sup>‡</sup>We assume, without loss of generality, that the initial state is  $b_i = 0$  for all  $i$ .

passing between the  $i$ th and  $j$ th location on the path, and let the “starting” location be 0. We can then write the value of  $\pi$  as:

$$\begin{aligned}
V^\pi(s) &= \mathbb{E}[-\epsilon(\# \text{ steps from start to delivery of final package})] \\
&= \mathbb{E}\left[-\epsilon \sum_{i=1}^k n_\pi(i-1, i)\right] \\
&= -\epsilon \sum_{i=1}^k \mathbb{E}[n_\pi(i-1, i)] \quad .
\end{aligned} \tag{2}$$

The sum in the last line is precisely the tour length for a traveling salesdroid following the tour defined by the  $l_0 \dots l_k$  according to the costs  $n$ . There are two components to minimize this sum: the number of steps between each pair of locations and the order in which the locations are toured. Because the spatial transitions are independent of the bits, the first quantity can be minimized without regard to bit settings, while the ordering problem is precisely the TSP. While the TSP is NP-hard, very effective heuristic search methods and polynomial-time approximation techniques are known for many practical special cases.<sup>14</sup> We will show how to use this insight to construct a final plan in the next section.

For the multiplicative cost case, the reward function is  $R(s) = 1$  when  $b_i = 1$  for all  $i$  (followed by a transition to a zero reward absorbing state), and the value function becomes

$$V^\pi(s) = \mathbb{E}[\gamma^{\sum_{i=1}^k n_\pi(i-1, i)}] \quad .$$

If  $\gamma$  is very close to 1 (i.e.,  $\gamma^{\text{total \# of steps}}$  is still close to 1), then we can use the approximation that  $\gamma^t = (1 - \epsilon)^t \approx 1 - \epsilon t$  and we have

$$V^\pi(s) \approx 1 - \epsilon \sum_{i=1}^k \mathbb{E}[n_\pi(i-1, i)] \quad .$$

This is effectively the same form encountered in Equation 2 and indicates that the TSP solution is also appropriate for this case.

The more interesting case occurs when we wish to account for package priorities, but this turns out to complicate the picture considerably. Starting again with the additive costs case and given the priority factors,  $w_i$ , introduced in Equation 1, the value function becomes:

$$\begin{aligned}
V^\pi(s) &= \mathbb{E}\left[\sum_{i=1}^k -\epsilon w_i (\# \text{ steps from start to location } i)\right] \\
&= -\epsilon \sum_{i=1}^k w_i \sum_{j=1}^k \mathbb{E}[n_\pi(j-1, j)] \quad .
\end{aligned} \tag{3}$$

The double sum in Equation 3 is related to the *minimum latency TSP* (a.k.a., traveling repairdroid problem or MLP).<sup>5,6</sup> The goal of this problem is to produce a tour over locations that minimizes the total latency, or delay from start to *each* location. (Picture a repairperson servicing  $k$  customers; each customer’s distress is related to the length of time that individual had to wait and the amount of grief the repairperson accumulates is related to the total delay.) While still formally NP-hard, this problem turns out to be more difficult in practice and much less is known about effective heuristics for this case. Nonetheless, it seems likely that we can locate a reasonable approximate minimization of this double sum far more easily than we could directly solve for the value function on the full, exponentially large, state space. Our formulation includes a factor of  $w_i$  as well, which makes this a *weighted* or *prize-collecting minimum latency TSP*. We are still studying the relationship of this variant to the MLP, but it seems to be a minor extension to the known MLP.

Again, the multiplicative cost is similar. If we invoke the near-unity  $\gamma$  approximation, we find that

$$V^\pi(s) \approx \sum_{i=1}^k w_i - \epsilon \sum_{i=1}^k w_i \sum_{j=1}^k \mathbb{E}[n_\pi(j-1, j)] \quad .$$

## 4.2. Planning Algorithm

Having found known combinatoric optimization problems embedded in plausible MDPs, we can exploit this to quickly approximate the MDP solution. The key insights are that the only quantities we need for the TSP solution are the expected distances between package delivery locations,  $\mathbb{E}[n(i, j)]$ , and that because the indicator bits are independent, we can find these quantities

in a much lower-dimensional state space. In particular,  $n$  depends only on  $\langle x, y \rangle$ , so we can work with only the navigational subspace rather than the full ordering problem. Specifically, we define the “reduced MDPs”,  $\widetilde{\mathcal{M}}_i = \langle \widetilde{\mathcal{S}}_i, \mathcal{A}, \widetilde{T}_i, R_i \rangle$ , where  $\widetilde{\mathcal{S}}_i$  is the state space formed by the variables  $\langle x, y, b_i \rangle$ ,  $\widetilde{T}_i$  is the transition function formed by dropping references to  $b_j$  for  $j \neq i$ , and  $R_i$  is the reward function defined in Section 3.2 for the additive discount case. This structure is an exponentially smaller MDP than the original model and we can solve for an optimal policy on it directly via, for example, policy iteration. The resulting policy,  $\widetilde{\pi}_i$ , expresses how to reach goal state  $\text{loc}(i)$  in the fewest steps from any  $\langle x, y \rangle$  location in the world. The quantity  $E[n_{\pi_i}(s, i)]$  for all  $s \in \widetilde{\mathcal{S}}_i$  is available from this policy—this is the *first passage time*<sup>15</sup> and can be calculated in from the Markov chain that results from coupling  $\widetilde{\pi}_i$  to  $\widetilde{\mathcal{M}}_i$  in time polynomial in the size of  $\widetilde{\mathcal{S}}_i$ .

We can simply project  $\widetilde{\pi}_i$  back into the global state space by reintroducing the other indicator bits without regard to their values,  $\pi_i(\langle x, y, b_1 \dots b_i \dots b_k \rangle) = \widetilde{\pi}_i(\langle x, y, b_i \rangle)$ . Because navigation is independent of indicator bits, these policies individually remain optimal with respect to reaching locations in the map. We treat the group of all such policies as a set of macros that express the shortest paths to each delivery location. The question is in what order to invoke them so as to optimize our global aggregate reward. Our macros are closest in structure to *options* as defined by Sutton, Precup, and Singh<sup>16,17</sup> and, indeed, can be easily written as options. It has been shown that a set of options defines a semi-Markov decision process over the original state space and that optimal planning with options is equivalent to solving this SMDP. This approach is still exponentially difficult but we *can* attack it in polynomial time via the TSP approximation.

Define the auxiliary graph  $G$  with vertices  $G_V = \{s, 1 \dots k\}$  and edges  $G_E = \{E[n_{\pi_j}(i, j)]\}$ , where  $s$  is the unique “starting state” for the episode. For the simple TSP case (single reward at termination, as specified in Equation 2), this is sufficient to define a TSP instance which can be approximated via known techniques. The particular approximation bounds and running times achievable depend on specific properties of  $G$ , but reasonable approximations are available for many cases of interest. When  $G$  is Euclidean, for example, Christofides’s algorithm produces a path guaranteed to be within a factor of 1.5 of the optimal and much better bounds can be achieved in practice. Our interest here is not with the details of TSP solution methods, so we will assume the existence of a polynomial-time TSP approximator,  $TSP(G)$ , that produces a *tour* or permutation,  $\tau$ , on  $G_V$  that specifies the order in which macros are to be invoked:  $\tau(i) = j$  indicates that macro  $j$  (for delivering package  $j$ ) should be the  $i$ th invoked during execution. We note that the model formulation and design of our macros are such that each macro terminates at its target location with probability 1 (i.e., there are no ergodic sets in the model other than the terminal one where all packages are delivered). Thus, the geographic package locations are traversed in a deterministic order given by  $\tau$  (modulo prematurely encountering a package delivery state—a case that we will describe shortly). We have exchanged the “spatial uncertainty” in the atomic actions for spatial determinism at the cost of introducing *temporal* uncertainty in the duration of the macros, expressed by  $n$ .

The global plan, then, is defined to be:

$$\pi(s) = \widetilde{\pi}_{\tau(i)}(s) \quad \text{for the minimum } i \text{ s.t. } b_{\tau(i)} = 0 \text{ and } b_{\tau(j)} = 1 \text{ for all } j < i \quad (4)$$

In the prioritized packages case, defined by Equation 3, we assume the existence of a weighted-MLP approximator,  $MLP(G, W)$ , where  $W = \langle w_1 \dots w_k \rangle$  is the vector of package weights.  $MLP$  produces a tour permutation,  $\tau$ , which is used directly in Equation 4 to produce a global policy that accounts for package priorities.

The policy given by Equation 4 is universal, in that it specifies an action for every state, but can be implicitly represented with a polynomial amount of space and constructed on demand during runtime with linear work per robotic action. It is, however, suboptimal in at least two ways:

1. The quality of the permutation  $\tau$  is determined by the accuracy of the TSP approximation. A poor TSP tour will yield a poor final policy.
2. This rule assumes that the robot will only find itself in areas close to the “intended” path between goals  $\tau(i)$  and  $\tau(i+1)$ . If the robot should happen to deviate very far from this path, it might be necessary to reevaluate the TSP tour and reconstruct  $\pi$ .

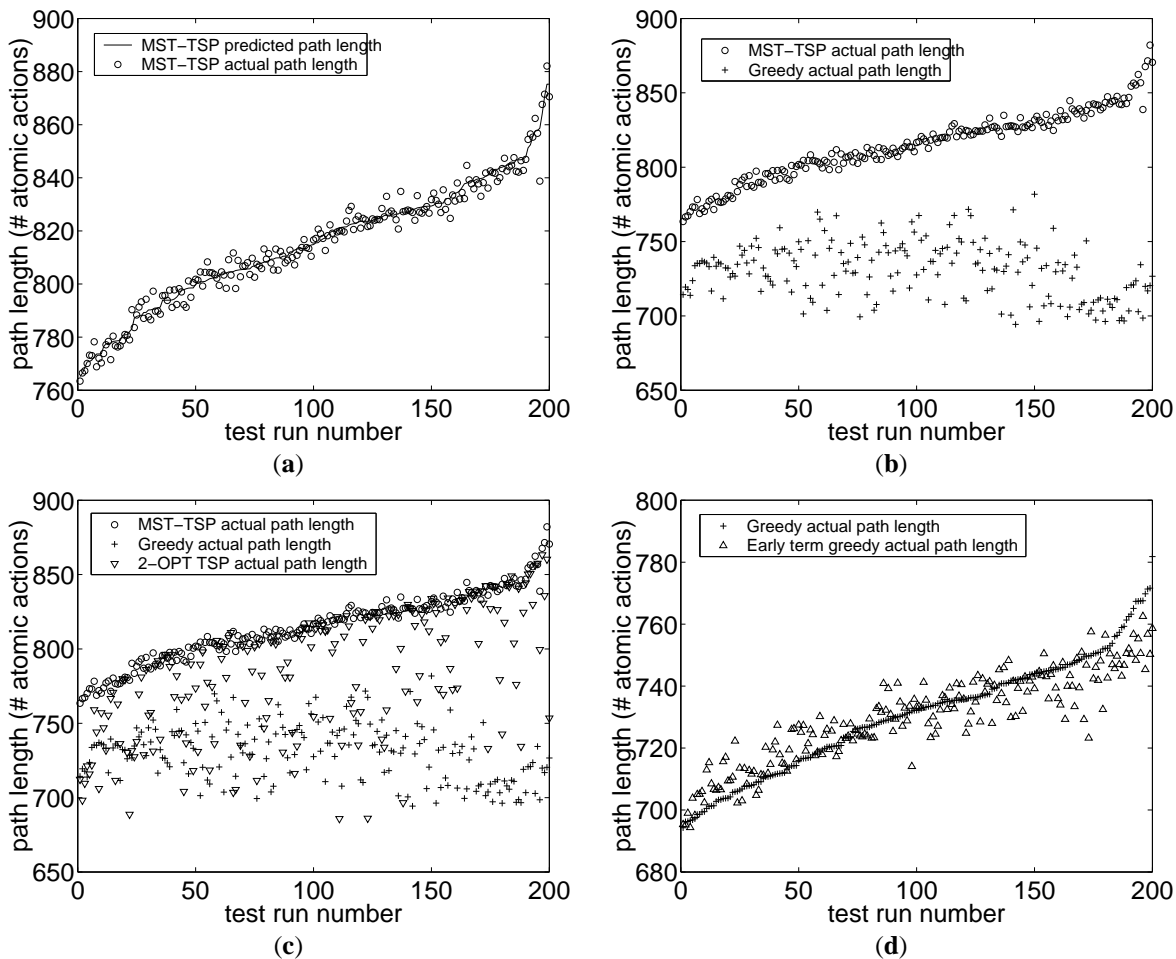
Furthermore, for the multiplicative discounting case, there is an additional approximation factor related to the proximity of  $\gamma$  to 1.

The second point is, perhaps, the most troubling. Depending on the nature of the navigational problem and the proximity of goal states (in geographic space), the chances of this might be high. We are currently evaluating possible solutions, but in the

worst case we could certainly reevaluate the entire TSP solution (in polynomial time) after each action step. This is the *early termination* process described by Precup for the options framework<sup>16</sup> and is guaranteed to improve (or at least not decrease) the quality of the final plan.

## 5. EVALUATION

Our initial evaluations in this domain have focused on the single ultimate reward additive case which yields the standard TSP optimization problem (Equation 2). The prioritized packages case, which yields the prize-collecting minimum latency optimization problem (Equation 3), requires more complex algorithms that are currently under development. The simple TSP case suffices to demonstrate the principle of applying deterministic optimization techniques to MDP models. We apply a number of simple known TSP heuristics for this case; while our purpose in this paper is not to fully explore the breadth of TSP optimization methods, examining a few simple heuristics will help demonstrate the relative impact of the two sources of suboptimality described in Section 4.2.



**Figure 3.** Mean path length results for a number of heuristic TSP solvers in the domain of Section 3.2. Y-axes give mean path length in number of atomic actions from start to termination; different starting locations are arranged horizontally. (a) Predicted and actual path lengths for the spanning tree TSP heuristic. (b) Comparison of MST-TSP and greedy heuristics. (c) Comparison of MST-TSP, greedy, and the 2-OPT improvement to MST-TSP. (d) Effects of early termination on the greedy heuristic.

Possibly the simplest reasonable approach is to generate a tour by shortcutting a traversal of the minimum spanning tree over the TSP graph (MST-TSP). This heuristic is guaranteed to produce a tour within a factor of 2 of the optimal when the TSP graph obeys the triangle inequality.<sup>18</sup> § We can easily improve on these solutions with the 2-OPT search, which greedily

§In general, passage times produced by MDPs need not be metric (we have, in fact, detected small asymmetries in the mean passage times

selects pairs of edges to swap within the tour, terminating when no more improving swaps are available or an iteration limit is exceeded. We have also examined greedy tour generation, which incrementally grows a tour by adding the nearest as-yet unvisited location to the end. Greedy selection is known to fail catastrophically for some simple TSP instances, but is simple and executes in linear time (in fact, the tour need not be generated in advance at all).

Figure 3 displays the path length results for these TSP heuristics (for this case, total reward is simply  $-\epsilon$  times the path length). By hypothesis, this domain is far too large to evaluate the exact value of every state, so we sampled initial geographic locations with all packages outstanding. We chose 200 initial (geographic) states in the model described in Section 3.2, distributed uniformly at random, and generated tours from each starting location. We estimated the value of each tour by executing its policy to termination (i.e., delivery of the final package) 25 times and averaging the result. In Figure 3 (a), we show the results for the MST heuristic method. The vertical axis gives tour length and the tours are laid out horizontally by starting state. For clarity of presentation, we have sorted the starting locations to arrange the tour values in monotonic order. The solid line shows the value estimated for the tour by the TSP solution (i.e., by Equation 2) and the circles give the actual value achieved by sampling the policy. The proximity of the two data sets indicates that Equation 2 is a good estimate and that, for example, the TSP solution is not distorted by the variance of passage time or other higher-order terms.

In Figure 3 (b), we show the sampled performance values achieved by the greedy heuristic compared with the MST solution. The predicted tour lengths are again close to achieved and we omit them henceforth. Clearly, the greedy heuristic substantially outperforms the MST approach in this domain. In general, this need not be so, but the world we are dealing with is fairly regular and it is, perhaps, not surprising that a greedy strategy would perform reasonably. Note, however, that the greedy instances do not form a smooth curve on this plot; the instances for which the MST heuristic performs strongly are not necessarily the same as those for which the greedy method is most successful.

Figure 3 (c) demonstrates that we can use 2-OPT (initialized with the MST tours) to improve on MST tours; this has the advantage of often locating much more effective tours while maintaining the MST's worst-case bound of a factor of 2; a bound that the greedy search cannot promise. In some cases, these tours outperform even those of the greedy selection system. In general, it is clear that there is a wide performance range, closely coupled to the quality of the TSP solution (point 1 in Section 4.2). The TSP is an extremely well studied and well understood problem and many practically effective techniques exist for it; the power of our approach is that all of that knowledge is now immediately applicable to these types of navigational MDPs.

There remains point 2 raised in Section 4.2: all of the methods discussed so far operate strictly on the navigational macros and don't consider atomic actions. In principle, macro-based plans can be improved through early termination<sup>16</sup> which allows the agent to reevaluate the plan on each step. For computational reasons, we have not yet implemented this procedure for the MST or 2-OPT heuristics, but the greedy approach is cheap enough to practically reevaluate on each step. (In practice, the robot simply heads for the currently closest outstanding location on every time step.) We show the results of this approach in Figure 3 (d), from which it is apparent that early termination produces little improvement over the unterminating macro greedy approach in this domain. Intuitively, this is appealing; in this domain, the goal locations (in the offices) are well separated and the chances of navigational stochasticity carrying the robot far off its intended path (relative to inter-goal distances) is fairly small. Thus, it is likely that a macro will continue to remain appropriate for its entire duration and plan reevaluation will not be necessary. We expect this to be the case, as well, for the other TSP heuristics, but we intend to examine this point in future work.

## 6. CONCLUSIONS AND FUTURE WORK

We have demonstrated that for a plausible robot navigation domain, it is possible to achieve exponential improvements in planning time by extracting a known and well-understood combinatorial optimization problem (the TSP) from a reasonable Markov decision process model and using an approximate TSP solution to construct a full plan within the MDP. The key issue here is not that solution of a particular domain or even the use of the TSP per-se (clearly, package-delivery problems are a fairly constrained class of MDPs), but that it can be useful to incorporate known non-MDP optimization techniques to the solution of stochastic planning problems. We believe that there are many MDPs of practical interest that contain sub-problems that can be productively attacked in this way. Shortest paths, bin packing, and resource scheduling are all examples of plausible sub-problems that would benefit from our approach.

Having gone to these efforts, the reader may wonder why it is even useful to introduce the MDP model at all. Why not just measure the expected passage times directly in the world and plan directly with the TSP? Our answer is twofold: for in our TSP graphs), but this domain is fairly close to Euclidean and we consider this a reasonable approximation.

the package delivery problem in specific, the MDP allows us to model long-distance uncertainties (i.e., passage times between distant locations) by using purely local stochastic information. We can easily measure the dynamic properties of the robot's local controllers in a relatively confined region (e.g., a single room or hallway) and extend that knowledge to the entire navigational domain through the MDP model. Directly measuring the expected distances between distant locations is much more difficult, as it requires actually measuring the robot's actions over those distances which, in turn, requires a global controller. But finding a global controller is precisely the point of this entire exercise.

Our second answer is more widely applicable: MDPs can express extremely general dynamical models that may contain *many* sub-problems that interact in complex and stochastic ways. For example, some of our extended goals for robotic planning include handling domains with resource constraints (e.g., battery power and maximum load), timing constraints (monitoring locations), and cooperation of multiple agents. While it may be possible to solve the various sub-problems via deterministic methods in isolation, the MDP is necessary to express the relationships between them as well as the noise in atomic actions.

In general, we believe that there is a spectrum of problems from those that are best attacked purely in the stochastic planning domain of MDPs to those that are best handled purely with deterministic optimization methods. A great deal of effort has been spent on the ends of this spectrum, but we are interested in examining the region in the middle. We are exploring the tradeoffs between the stochastic and deterministic systems, when different models are appropriate, and how we can exploit the strengths of both classes simultaneously.

In the shorter term, we are working in conjunction with the MIT computer science theory group to develop and implement appropriate algorithms for the prize-collecting minimum latency TSP that arises from the prioritized packages case. We are also extending our simulation domains to a larger physical space (an entire building or campus, rather than a single floor) and to handle more factors (battery power, load limits, non-fixed delivery locations, "target of opportunity" goals, etc.) The smallest of the latter models contain roughly  $2^{500}$  states; techniques like those presented here are utterly critical to attacking such systems. We also intend to extend the investigations of Section 5 to examine early termination for TSP results and by comparing the TSP heuristics to truly optimal solutions in very small domains where direct solution is still feasible. Finally, our ultimate goals are to implement our techniques on physical robotic systems and test them on real problem domains such as package delivery or search and rescue.

## ACKNOWLEDGMENTS

We would like to thank David Karger and Maria Minkoff for their insights into the prize-collecting minimum latency case and for their ongoing assistance in developing algorithms for that situation. We are also grateful to Yu-Han Chang and Mike Benjamin for commenting on drafts of this paper and to the Learning in Intelligent Systems group in general for their helpful discussions on early phases of this work. This research is supported by DARPA contract #DABT63-99-1-0012 and by NASA award #NCC2-1237.

## REFERENCES

1. D. Koller and R. Parr, "Policy iteration for factored MDPs," in *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference (UAI 2000)*, Morgan Kaufmann, 2000.
2. C. Boutilier, R. Dearden, and M. Goldszmidt, "Stochastic dynamic programming with factored representations," *Artificial Intelligence* **121**(1–2), pp. 49–107, 2000.
3. T. G. Dietterich, "An overview of MAXQ hierarchical reinforcement learning," in *Proceedings of the Symposium on Abstraction, Reformulation and Approximation SARA 2000*, B. Y. Choueiry and T. Walsh, eds., Lecture Notes in Artificial Intelligence, Springer Verlag, New York, 2000.
4. M. R. Garey and D. S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, W. H. Freeman and Company, NY, 1979.
5. M. Goemans and J. Kleinberg, "An improved approximation ratio for the minimum latency problem," in *SODA: Proceedings of the Seventh ACM-SIAM Symposium on Discrete Algorithms*, 1996.
6. S. Arora and G. Karakostas, "Approximation schemes for minimum latency problems," in *ACM Symposium on Theory of Computing*, pp. 688–693, 1999.
7. T. Lane and L. P. Kaelbling, "Toward hierarchical decomposition for planning in uncertain environments," in *Proceedings of the IJCAI-01 Workshop on Planning Under Uncertainty and Incomplete Information*, pp. 1–7, International Joint Conferences on Artificial Intelligence, Inc (IJCAI), (Seattle, WA), Aug. 2001.

8. M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons, New York, 1994.
9. A. W. Moore, L. C. Baird, and L. Kaelbling, "Multi-value-functions: Efficient automatic action hierarchies for multiple goal MDPs," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, T. Dean, ed., Morgan Kaufmann, (Stockholm, Sweden), 1999.
10. L. P. Kaelbling, "Hierarchical learning in stochastic domains: Preliminary results," in *Proceedings of the Tenth International Conference on Machine Learning*, pp. 167–173, 1993.
11. M. Hauskrecht, N. Meuleau, C. Boutilier, L. P. Kaelbling, and T. Dean, "Hierarchical solution of Markov decision processes using macro-actions," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, G. F. Cooper and S. Moral, eds., Morgan Kaufmann, 1998.
12. R. Parr, "Flexible decomposition algorithms for weakly coupled Markov decision problems," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, G. F. Cooper and S. Moral, eds., Morgan Kaufmann, 1998.
13. C. Guestrin and D. Ormoneit, "Robust combination of local controllers," in *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-01)*, J. Breese and D. Koller, eds., pp. 178–185, Morgan Kaufmann, (Seattle, WA), Aug. 2001.
14. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, eds., *The Traveling Salesman Problem*, John Wiley & Sons, New York, 1985.
15. J. G. Kemeny and J. L. Snell, *Finite Markov Chains*, Undergraduate Texts in Mathematics, Springer-Verlag, New York, 1976.
16. D. Precup, *Temporal Abstraction in Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, Department of Computer Science, 2000.
17. R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence* **112**, pp. 181–211, 1999.
18. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.