Model-Lite Planning: Diverse Multi-Option Plans & Dynamic Objective Functions

Daniel Bryce*, **William Cushing, & Subbarao Kambhampati**
Department of Computer Science and Engineering
Arizona State University, Brickyard Suite 501
699 South Mill Avenue, Tempe, AZ 85281
{dan.bryce, wcushing, rao}@asu.edu

Abstract

Knowledge acquisition is one major bottle-neck in using planning systems. Model-lite planning reduces this burden by placing responsibility on the planning system to cope with partially specified models. For example, eliciting the planning objective can be difficult in applications where it is necessary to reason about multiple plan metrics, such as cost, time, risk, human life, etc. Traditional approaches, often require a (sometimes subjective) combination of these objectives into a single optimization metric. For example, decision theoretic planners combine plan cost and probability of goal satisfaction into a single reward metric. However, users may not know how to combine their metrics into a single objective without first exploring several diverse plan options.

To avoid premature objective function commitments at plan synthesis time (and even plan execution time), we develop the notion of multi-option plans. Much like conditional plans that branch to deal with execution-time observations, multi-option plans branch to deal with execution-time assessments of plan objectives. That is, a multi-option plan is a compact representation of the diverse Pareto set of plans, where at each step the user can execute one of several non-dominated options.

We formulate multi-option planning within the context of conditional probabilistic planning, where plans satisfy the goal with different probabilities and costs. Our approach is based on multi-objective dynamic programming in state space, where each plan node maintains a set of non-dominated sub-plan options, that are each a conditional plan.

Introduction

In contrast with traditional knowledge intensive planning models, model-lite planning (Kambhampati, 2007) seeks to ease the knowledge acquisition bottleneck. Users provide potentially incomplete domain models, and the planning system must still assist the user in their planning task. In this paper, we consider the issue of having a weakly defined planning objective function. That is, the user identifies the objectives of interest (e.g., cost, risk, and makespan, etc.), but does not or cannot commit to an objective function (e.g., the weights for a linear combination). We not only consider such user indecisiveness at plan synthesis time, but also during plan execution. Users may change their objective function during execution because uncertain events turn out (un)favorable, or factors not present in the model arise.

To support dynamic objective functions, we develop the notion of multi-option plans. A multi-option plan is a succinct representation of a diverse set of non-dominated plans, where each constituent plan optimizes the planning objectives in a unique way. To better understand the relationships between constituent plans (and to better support their synthesis), we represent the set much like a conditional plan. The correspondence can become explicit if we were to model the user's objective function as an uncertain action or state feature. However, instead of using a (potentially nonintuitive and large) transformation of the state and action model, we provide a new search space and plan synthesis algorithm.

Using a state space representation, we generalize the traditional mapping from each plan state to a *single best action* to a *set of best actions* (where each action is optimal for some subset of the non-dominated sub-plan options). Then in each state, the plan executor has multiple options, each satisfying objectives differently. We formulate a multi-objective Bellman equation to compute the set of best actions for each state and use a variant of the LAO* algorithm to compute the multi-option plan. We investigate conditional probabilistic planning, where the multi-option plan provides a range of choices for satisfying the goal with different probability at different costs. The motivation for a multi-option conditional plan is to allow the executor to adjust their plan cost budget as it proves more or less likely they will achieve the goal; consider the following example:

Example: *The multi-option conditional plan in Figure 1 starts with action a, which provides two observations and has an execution cost of one. In the branch corresponding to the first observation (whose probability is 0.2) it is possible to execute π_1 to achieve the goal with 1.0 probability and expected cost 50 or option π_2 with 0.5 probability and cost 10. In the branch for the second observation (whose probability is 0.8) it is possible to execute π_3 to achieve the goal with 0.75 probability and cost 30 or π_4 with 0.0 probability and cost 0. The result is a set of diverse plan options:*

*Current Affiliation: SRI International, 333 Ravenswood Ave., Menlo Park, CA, bryce@ai.sri.com.

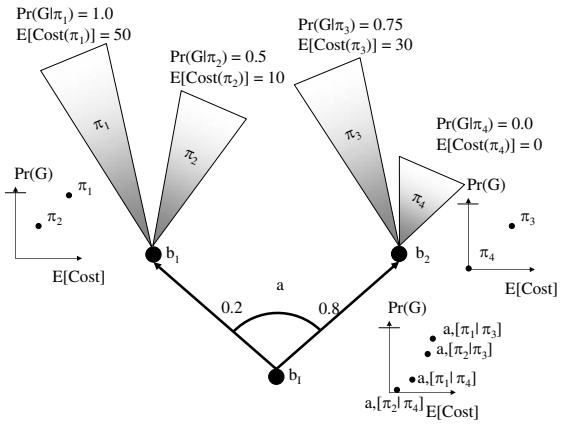

Figure 1: Example of multi-option conditional plan.

Options	E[Cost(π)]	Pr($G|\pi$)
$a, [\pi_1 \vert \pi_3]$	1+(.2)50+(.8)30 = 35	(.2)1+(.8).75 = .8
$a, [\pi_2 \vert \pi_3]$	1+(.2)10+(.8)30 = 27	(.2).5+(.8).75 = .7
$a, [\pi_1 \vert \pi_4]$	1+(.2)50+(.8)0 = 11	(.2)1+(.8)0 = .2
$a, [\pi_2 \vert \pi_4]$	1+(.2)10+(.8)0 = 3	(.2).5+(.8)0 = .1

By executing a, the executor does not need to commit to an objective function. After executing a and (by chance) following the second branch, the executor must decide between options π_3 and π_4. The executor may have been hoping for the first branch (to execute π_2), but at this point they can give up and select π_4 or expend more cost in π_3. In many cases the executor does not need to commit to a specific objective function, and in other cases they may have their choice of several options.

There are many ways to execute a multi-option plan. If the executor commits to an objective function, they can choose the option that most closely matches. If they do not commit to an objective function, they may want to choose actions that "keep their options open", preferring actions that are consistent with the most options, or the most diverse set of options. We define these two execution strategies and compare them with a plan executor that chooses random options to illustrate some of the benefits of multi-option plans. While we examine an execution strategy for keeping plan options diverse, we do not take steps to make the plans diverse, aside from enforcing option diversity through Pareto optimality. In our discussion of future work, we outline some of the issues in guiding plan synthesis toward finding diverse options.

In the next section, we formally define our planning model. We then show how our novel multi-objective formulation enables us to find multi-option plans in the successive section. The next section describes our empirical evaluation of multi-option plan synthesis, empirical evaluation of multi-option plan execution, a discussion of related work, and conclusions.

Background & Representation

We describe our planning models in terms of flat Markov decision processes (MDPs), but note that our underlying implementation is propositional (based on OBDDs (Bryant, 1986)). We are interested in partially observable planning, but to simplify later sections, we describe the partially observable model as an equivalent fully-observable belief state MDP.

Full Observability: The fully-observable model is given by (S, A, T, s_I, G), defined as a set of states S, a set of actions A, a state transition relation $T(s, a, s')$, an initial state s_I, and a goal state function $G(s)$. The state transition relation describes a probability distribution: $T(s, a, s') = Pr(s'|a, s)$. A subset of actions $A(s) \subseteq A \cup \perp$ is applicable in each state s. The $\perp$ action signifies no action is performed in s. The goal state function $G : S \rightarrow [0, 1]$ maps each state to a probability it satisfies the goal. In this model, $G(s) = 1$ or $G(s) = 0$ for every state (later we will see that $0 \leq G(s) \leq 1$ in belief state MDPs). Applying an action incurs a cost $c(a)$, which we will assume (without loss of generality) is uniform (with the exception that $c(\perp) = 0$).

Partial Observability: The partially-observable MDP model is given by $(S, A, T, b_I, G, O, \Omega)$, defined as before, and by an initial belief state b_I, an observation function $O(s, a, o)$, and a set of observations Ω. A belief state $b : S \rightarrow [0, 1]$ is a probability distribution that maps each state to a probability, such that $\sum_{s \in S} b(s) = 1.0$. We say that $s \in b$ for all states where $b(s) \geq 0$. The set $A(b) = \cap_{s \in b} A(s)$ contains all actions applicable in a belief state b. The observation function $O(s, a, o) = Pr(o|a, s)$ is the probability distribution over observations $o \in \Omega$ received upon transitioning to state s after executing action a.

We define the belief state b_a reached by applying a in belief state b as $b_a(s') = \sum_{s \in S} b(s) T(s, a, s')$. The belief state b_a^o is the belief state after receiving observation o in belief state b_a, defined as $b_a^o(s') = \alpha O(s', a, o) b_a(s')$, where α is a normalization factor. The probability of receiving observation o in belief state b_a is $T(b, a, o, b_a^o) = \alpha = \sum_{s \in S} b_a(s) O(s, a, o)$.

Belief state MDPs: To clarify the correspondence between the fully-observable and partially observable model, consider the following transformation, called a belief state MDP. The belief state MDP $(\tilde{S}, A, \tilde{T}, \tilde{s}_{b_I}, \tilde{G})$ for the partially-observable MDP $(S, A, T, b_I, G, O, \Omega)$ is given by several equivalences, where each state $\tilde{s}_b \in \tilde{S}$ corresponds to a belief state b, and $\tilde{s}_{b_I}$ is the initial (belief) state. A is an unchanged set of actions, where $A(\tilde{s}_b) = A(b)$ is the set of actions applicable in b. The transition relation $\tilde{T}(\tilde{s}_b, a, \tilde{s}_{b_a^o})$ is equivalent to $T(b, a, o, b_a^o)$. $\tilde{G}$ is the probability each (belief) state satisfies the goal, such that $\tilde{G}(\tilde{s}_b) = \sum_{s \in S} b(s) G(s)$. In the following, we ignore all distinctions between belief state MDPs and fully-observable MDPs, where possible, given the above equivalences.

Multi-objective Q-value Formulation

We introduce a multi-objective dynamic programming formulation for multi-option planning that is based on Q values. In traditional MDPs, Q values denote the expected value of executing an action in a state. The value V for

a state is simply the maximal Q value of actions applied in the state, which in turn defines the plan π. Many MDP algorithms, such as value iteration, start with an initial Q value for each state action pair, and at each iteration uses the Bellman equation to update the value. Our contribution redefines these V values as sets of non-dominated Q values, where each Q defines a unique vector of objective values for a state and action pair. We also define a multi-objective Bellman equation used to update these values. In the next section, we describe a variation of the LAO^* algorithm to compute values. We present a general multi-objective formulation and point out specific details for computing the expected cost and probability of goal satisfaction objectives that are needed in conditional probabilistic planning.

Our extension to the value function turns $V(s)$ into a Pareto set. Each option of the Pareto set $Q(s,a) \in V(s)$ represents a vector of n objectives, such that $Q(s,a) = (Q_0(s,a), Q_1(s,a), ..., Q_{n-1}(s,a))$, and $V(s) = \{Q(s,a) | \neg\exists_{Q'(s,a') \in V(s)} Q'(s,a') \prec Q(s,a)\}$, where $V(s)$ contains only non-dominated options. A option $Q'(s,a')$ dominates another $Q(s,a)$ if it is as good in all objectives, and strictly better in at least one:

$$Q'(s,a') \prec Q(s,a) \Leftrightarrow \quad \forall_i Q'_i(s,a') \leq Q_i(s,a) \wedge \\ \exists_i Q'_i(s,a') < Q_i(s,a)$$

Each option is mapped to a best action by π (i.e., $\pi(s, Q(s,a)) = a$), making it possible for two sub-plan options to start with the same action but have different values.

Each $Q(s,a) \in V(s)$ is computed in terms of the successors of applying a in s. Because each successor $(s', s'', ...)$ is associated with a Pareto set $(V(s'), V(s''), ...)$, it is possible to define a different $Q(s,a)$ from each element $(w = \{Q(s',a'), Q(s'',a''), ...\})$ of the cross product of the Pareto sets $(W = V(s') \times V(s'') \times ...)$. We saw this in the example (Figure 1), where it is possible to have $|V(\tilde{s}_{b_I})| = 4$ because $|V(\tilde{s}_{b_1})| = |V(\tilde{s}_{b_2})| = 2$. The cross product of $V(\tilde{s}_{b_1})$ and $V(\tilde{s}_{b_2})$ contains four elements, and each $w \in V(\tilde{s}_{b_1}) \times V(\tilde{s}_{b_2})$ is used to define an element of $V(\tilde{s}_{b_I})$. The actual number of elements in a Pareto set may be less because some elements will be dominated.

For each $w \in W$, we define expected cost and probability of *not* satisfying the goal objectives for probabilistic conditional planning, $Q(s,a) = (Q_0(s,a), Q_1(s,a))$, such that:

$$\begin{aligned} Q_0(s,a) &= c(a) + \sum_{Q'(s',a') \in w} T(s,a,s') Q'_0(s',a') \\ Q_1(s,a) &= \sum_{Q'(s',a') \in w} T(s,a,s') Q'_1(s',a') \end{aligned} \tag{1}$$

We also define $Q_0(s, \perp) = 0$ and $Q_1(s, \perp) = 1 - G(s)$ when no action is applied.

From the example, $V(\tilde{s}_{b_I}) = \{(35, 0.2), (27, 0.3), (11, 0.8), (3, 0.9)\}$. The first element of $V(\tilde{s}_{b_I})$ is defined by $(Q(\tilde{s}_{b_1}, (50, 0.0)), Q(\tilde{s}_{b_2}, (30, 0.25))) \in V(\tilde{s}_{b_1}) \times V(\tilde{s}_{b_2})$ as $Q_0(\tilde{s}_{b_I}, a) = 1 + (0.2)50 + (0.8)30 = 35$ and $Q_1(\tilde{s}_{b_I}, a) = (0.2)0.0 + (0.8)0.25 = 0.2$.

Computing the Multi-objective Bellman Equation: Computing the multi-objective Bellman equation *once* for a state s can take $O(|A(s)||V(s')|^{|S|})$ time because $|A(s)|$ actions are applicable in s, each results in at most $|S|$ successor states s', and there are $|V(s')|$ Pareto optimal sub-plan options for each state s'. In a later section, we identify ways to reduce this complexity by limiting the size of the Pareto sets $V(s)$. Limiting the size of $V(s)$ equates to a form of limited option plans, akin to limited contingency plans (Meuleau & Smith, 2003).

```
MOLAO*()
 1: i = 0
 2: repeat
 3:    Z = ExpandPlan(s_I, i++)
 4:    AddAncestors(Z)
 5:    VI(Z)
 6: until (ConvergenceTest(s_I) == T)

ExpandPlan(s, i)
 1: if expanded(s) < i then
 2:    Z = Z ∪ s
 3:    if expanded(s) == 0 then
 4:       expanded(s) = i
 5:       ExpandState(s)
 6:    else
 7:       expanded(s) = i
 8:       for Q(s,a) ∈ V(s) s.t. ¬solved(Q(s,a)) do
 9:          for s' ∈ S : T(s,a,s') > 0 do
10:             Z' = ExpandPlan(s', i)
11:             Z = Z ∪ Z'
12:          end for
13:       end for
14:    end if
15: end if
16: return Z
```

Figure 2: $MOLAO^*$ Search Algorithm.

Multi-option LAO^*

Hansen & Zilberstein (2001) introduce Looping AO* (LAO^*) search as a technique for solving stochastic shortest path and MDP problems. Unlike the traditional value iteration and policy iteration techniques for solving MDP problems, LAO^* generalizes AO* search (Nilsson, 1980) to handle loops in the search graph. The idea is to expand a reachable region of the state space, over which value iteration is used to identify a partial plan. In each iteration, LAO^* expands unexpanded fringe states of the partial plan, and performs value iteration.

The main generalization needed for a multi-option LAO^* ($MOLAO^*$) is to reinterpret the V-values as V-sets and compute them as such. This also means that there may be several non-dominated plan options that $MOLAO^*$ can expand each iteration. Figures 2 and 3 describe the $MOLAO^*$ search algorithm using V-sets to find a Pareto set of conditional plans. The $MOLAO^*$ function in Figure 2 contains the outer loop that calls `ExpandPlan` to compute a set of states Z that are reachable by the set of non-dominated partial plans. To Z, `AddAncestors` adds all states that can reach a state $s \in$ Z with a non-dominated plan. The set

VI(Z)

1: **while** error $> \epsilon$ **do**
2: **for** $s \in Z$ **do**
3: error(s) = `Backup(`s`)`
4: **end for**
5: error = $\max_{s \in Z}$error(s)
6: **end while**

`Backup`(s)

1: $V'(s) = \emptyset$
2: **for** $a \in A(s)$ **do**
3: $W = \times_{s':T(s,a,s')>0} V(s')$
4: **for** $w \in W$ **do**
5: $Q_0(s,a) = c(a) + \sum_{Q'(s',a') \in w} T(s,a,s')Q'_0(s',a')$
6: $Q_1(s,a) = \sum_{Q'(s',a') \in w} T(s,a,s')Q'_1(s',a')$
7: solved($Q(s,a)$) $= \bigwedge_{Q'(s',a') \in w}$ solved($Q'(s',a')$)
8: `UpdateDominance`($Q(s,a), V'(s)$)
9: **end for**
10: **end for**
11: error = `ComputeError`($V'(s), V(s)$)
12: $V(s) = V'(s)$
13: return error

Figure 3: $MOLAO^*$ `VI` and state value `Backup` algorithms.

of partial plans is revised by calling `VI`(Z) to update $V(s)$ for each $s \in$ Z. When `ConvergenceTest` indicates that $V(s_I)$ has converged (i.e, all non-dominated solutions are labeled solved, and upper and lower bounds on their values is below a threshold), it is possible to stop.

The `ExpandPlan` function recursively traces each partial option (lines 7-13) to find leaf states to expand (lines 4-5). The `ExpandState` function applies each action $a \in A(s)$ to generate successor states. Each successor state s' has its Pareto set $V(s')$ initialized and expanded(s') is set equal to zero. Initializing $V(s')$ involves adding an option $Q(s', \perp)$ where solved($Q(s', \perp)$) is true, indicating that it is possible to end the plan at s'. We can also add heuristic options $Q(s, *)$ to $V(s)$ that indicate heuristic estimates of non-dominated sub-plan options. Each heuristic option is marked unsolved. Through dynamic programming, each $V(s)$ will contain some heuristic options that indicate the value of partial sub-plan options rooted at s. Later, we discuss which and how many heuristic options we use.

The `VI` function performs value iteration on a set of states (i.e., iteratively calling `Backup`, Figure 3, for each state until the maximum change in some $V(s)$ falls below a threshold). The `Backup` function computes $V(s)$ by applying every action in the loop (lines 2-10). For each action, W is the cross product of Pareto sets of successor states (line 3). For each element $w \in W$, an option $Q(s,a)$ is computed (lines 5-6). Each new option is marked solved if each of the successor options is solved (line 7). The `UpdateDominance` function (line 8) maintains the Pareto set by checking if each new option is dominated, or dominates options already in the Pareto set.

We deliberatively leave `ConvergenceTest` and `ComputeError` undefined because space precludes a thorough analysis of the solution error. Instead, in the following, we introduce a variation on $MOLAO^*$ that we use to compute a subset of plan options. Many of the speed-ups involved will affect completeness and admissibility. In future work, we will analyze the value iteration error and conditions for optimality by using the notion of hyperarea difference (Wu & Azram, 2001).

$MOLAO^*$ Speedups

In the following, we describe four improvements to $MOLAO^*$ that help find feasible, but suboptimal, multi-option plans quickly. The first is a modified version of $MOLAO^*$, named m$MOLAO^*$. The second describes heuristics. The third involves Pareto set approximations. The fourth simulates the current partial plan to focus synthesis, similar to RTDP (Barto, Bradtke, & Singh, 1995).

m$MOLAO^*$**:** The variation of $MOLAO^*$ that we pursue is very similar to the "efficient" version of LAO^* presented by Hansen & Zilberstein (2001) (c.f. Table 7). The idea is to combine value iteration with solution expansion by calling the `Backup` function for each state after recursively calling `ExpandPlan` for each of child state reached by an action in $V(s)$. Figure 4 describes our version of the $MOLAO^*$ algorithm that combines value iteration with solution expansion. We omit the convergence test used in LAO*, and stop when there is at least one plan option where the probability of goal satisfaction is above a threshold τ. We use this stopping criterion because, following from the undecidability result for *partially observable* conditional probabilistic planning (Madani, Hanks, & Condon, 1999), it is (in general) impossible to know when we have found all possible plan options. In other words, in some cases it may always be possible to find plan options that achieve the goals with higher probability. Because it is also possible that we cannot find a plan that exceeds τ, we experiment with increasing values of τ.

$MOLAO^*$ **Heuristics:** Adding heuristic options increases the size of V-sets, which we previously noted as a major source of complexity in computing state backups. There are two techniques we use to mitigate the increase in V-set size. First, we use a single heuristic option that estimates the cost to satisfy the goal with probability 1. Second, we limit how solved and not solved options are combined in the state backups. Specifically, we only combine solved options with solved options, and not solved options with not solved options. Formally, we replace W in line 3 of `Backup` with W':

$$W' = W \backslash \{w | Q(s,a), Q'(s',a') \in w, w \in W, \text{solved}(Q(s,a)) \neq \text{solved}(Q'(s',a'))\}$$

If we let $V_{sol}(s)$ denote all solved options and $V_h(s)$ denote all not solved (heuristic) options for a state s, then restricting the combination of solved and not solved options will bring the complexity of computing the multi-objective Bellman equation from $O(|A(s)|(|V_{sol}(s)| + |V_h(s)|)^{|S|})$ to

```
mMOLAO*()
 1: i = 0
 2: repeat
 3:    mExpandPlan(s_I, i^++)
 4: until ∃_{Q(s_I,a)∈V(s_I)} feasible(Q(s_I, a))

mExpandPlan(s, i)
 1: if expanded(s) < i then
 2:    if expanded(s) == 0 then
 3:       expanded(s) = i
 4:       ExpandState(s)
 5:    else
 6:       expanded(s) = i
 7:       for Q(s, a) ∈ V(s) s.t. ¬solved(Q(s, a)) do
 8:          for s' ∈ S : T(s, a, s') > 0 do
 9:             mExpandPlan(s', i)
10:          end for
11:       end for
12:    end if
13:    Backup(s)
14: end if
```

Figure 4: Modified $MOLAO^*$ Search Algorithm.

$O(|A(s)|(|V_{sol}(s)|^{|S|}+|V_h(s)|^{|S|}))$. Notice that because we use a single heuristic option per Pareto set and combine options in this fashion, there will only be a single partial plan expanded in line 7 of `mExpandPlan` (i.e., $\forall s|V_h(s)| = 1$).

The specific heuristic that we use is extracted from the $\mathcal{M}cLUG$ (Bryce, Kambhampati, & Smith, 2006). It estimates the cost to satisfy the goal with at least a given probability (which we set to 1.0) by computing a relaxed plan. The $\mathcal{M}cLUG$ relaxed plan estimates the conformant probabilistic plan cost, which can be seen as an additional relaxation that ignores observations. We also compare with a non-heuristic strategy where we set the cost to reach the goal with 1.0 probability to zero.

Approximating Pareto Sets: As with most multi-objective optimization problems, a Pareto set can contain an exponential number of non-dominated options. There exists a range of techniques for computing these Pareto sets and we explore one idea to reduce their size. It relies on the notion of ϵ-domination (Papadimitriou & Yannakakis, 2003) to prune solutions. By inflating each objective of other options by a factor of $1 + \epsilon$, it is possible to approximate the Pareto to within a relative error of ϵ in each objective by using a different definition of domination:

$Q'(s, a') \prec^{\epsilon} Q(s, a) \Leftrightarrow \forall_i Q'_i(s, a') \leq (1 + \epsilon)Q_i(s, a)$

The resulting Pareto set is polynomial sized in the number of sub-plan options and $\frac{1}{\epsilon}$ (c.f. Theorem 1, Papadimitriou & Yannakakis, 2003), but may still take exponential time to compute. However, because the number of options tends to increase during bottom-up dynamic programming (i.e., a parent's Pareto set is exponential in its childrens' Pareto sets) pruning more options can have a large effect. In future work where we analyze the error in $MOLAO^*$, we intend to incorporate the error introduced by approximating the Pareto sets.

This strategy for pruning some of the plan options is one possible way to enforce diversity between options. In the conclusion and discussion of future work, we outline a few additional ways to enforce diversity (in conjunction with techniques for pursuing diverse options).

Randomized Expansions: Instead of expanding every leaf state of the current partial plan, it is possible to expand a single leaf state that is reached by simulating the partial plan. This variation of $MOLAO^*$ called $MOLAO^{*r}$ samples a single successor state s' from the transition relation in line 8 of `mExpandPlan` instead of iterating over every successor. The intuition for this variation is to concentrate plan synthesis on the most likely plan branches.

Empirical Evaluation: Plan Synthesis

We implemented $MOLAO^*$ within the $POND$ planner to take advantage of its reachability heuristics computed from the $\mathcal{M}cLUG$. In the following, each reference to $MOLAO^*$ is with respect to the modified version (described above).

We use two partially observable domains for evaluation: First-Responders (FR) and Grid. The following lists the number of Actions $|A|$, Propositions $|P|$, and observations $|O|$ in each problem, which is quite large for partially observable planning.

Problem	$\|A\|$	$\|P\|$	$\|O\|$
FR1	82	29	22
FR2	116	45	28
FR3	200	54	42
Grid	5	20	2

The First-Responders domain is inspired by the planning required of dispatchers in disaster scenarios. There may be multiple victims (of unknown health) that need to be treated on the scene (with a lower probability of improvement) or taken to the hospital by an ambulance (where their health will improve with high probability). There may also be fires at several locations that fire trucks can extinguish. Both fire trucks and ambulances can report what they observe at a location. The goal is to bring each victim to health and extinguish all fires. We use three instances: FR1 has two fires, two victims, and four locations; FR2 has two fires, four victims, and four locations; and FR3 has two fires, two victims, and nine locations.

The Grid domain is an adaptation of the grid problem first described by Hyafil & Bacchus (2004) to include an action with observations. Like the original domain, a robot is moving about a 10x10 grid. The robot starts in the center of the grid and must reach a given corner. Its actions move it in the intended direction with 0.8 probability and in an adjacent direction with 0.1 probability. A sensory action allows the robot to perfectly sense the presence of a wall.

To characterize how well our planner performs while finding multi-option plans, we report results for finding an option with an increasing probability of goal satisfaction (τ) – as higher probability options are typically more difficult to find. The results for the First-Responders and Grid domains are shown in Table 1. All the results are the average of 5 runs and correspond to a single plan option, whose probability of

		$MOLAO^{*r}$					$MOLAO^{*h}$					$MOLAO^{*hr}$				
	τ	T(s)	E[C]	PS	DS	TS	T(s)	E[C]	PS	DS	TS	T(s)	E[C]	PS	DS	TS
FR1	0.25	314.8	9.8	101.0	3.0	19.0	41.0	12.1	165.2	14.8	20.2	14.5	12.0	42.6	1.2	13.8
	0.50	352.7	19.1	185.0	13.0	24.0	56.8	21.9	335.2	31.8	31.0	30.9	22.5	85.8	4.4	17.0
	0.75	370.2	25.6	277.0	25.0	27.0	62.1	26.8	448.2	47.2	37.4	42.4	27.5	160.2	10.0	20.8
	0.95	-	-	-	-	-	81.7	30.1	533.6	65.8	31.8	145.8	29.0	330.8	29.5	30.8
FR2	0.25	-	-	-	-	-	291.9	15.5	447.2	65.4	39.0	160.3	18.0	86.6	2.6	17.2
	0.50	-	-	-	-	-	318.6	26.3	469.4	53.4	31.4	130.9	24.7	115.4	5.6	19.6
	0.75	-	-	-	-	-	370.1	32.8	585.4	77.8	37.0	231.9	31.1	218.6	17.4	25.4
	0.95	-	-	-	-	-	457.4	39.3	816.8	98.8	41.2	292.2	37.5	373.0	35.0	30.0
FR3	0.25	-	-	-	-	-	420.9	12.8	392.0	34.2	33.0	252.8	15.0	69.0	1.8	15.2
	0.50	-	-	-	-	-	468.2	23.5	553.3	50.0	39.3	265.0	27.0	141.6	4.4	20.2
	0.75	-	-	-	-	-	623.7	40.67	741.0	51.0	45.0	459.2	35.3	248.0	13.0	26.4
	0.95	-	-	-	-	-	624.6	39.2	865.0	70.0	49.0	-	-	-	-	-
Grid	0.25	-	-	-	-	-	32.3	10.8	21.2	0.0	6.8	21.7	13.2	19.8	0.0	6.8
	0.50	-	-	-	-	-	356.7	19.4	153.2	0.2	36.8	55.5	19.0	54.8	0.2	14.8
	0.75	-	-	-	-	-	954.5	23.6	537.6	0.8	104.8	161.3	21.5	150.6	0.0	29.8

Table 1: Results using $MOLAO^{*r}$, $MOLAO^{*h}$, and $MOLAO^{*hr}$.

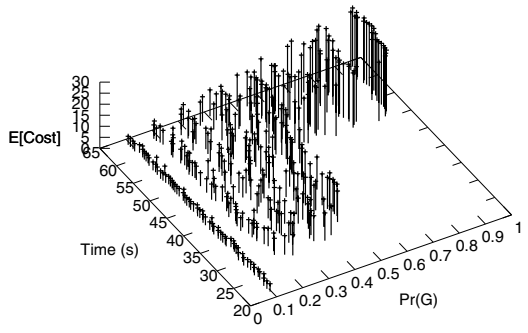

Figure 5: E[C] vs. τ over time on FR1, using $MOLAO^{*h}$.

goal satisfaction is maximal. We compare three versions of the modified $MOLAO^*$ where the superscript "r" corresponds to using randomized expansions, and the superscript "h" corresponds to using the relaxed plan heuristic (instead of zero) for heuristic options. The version of $MOLAO^*$ that does not use planning graph heuristics or randomized expansions was not able to solve any problems. In each version we use $\epsilon = 0.1$ and 10 samples within each $\mathcal{M}cLUG$ (Bryce, Kambhampati, & Smith, 2006). The table reports total time in seconds "T(s)", plan cost "E[C]", number of unique belief states in the plan where an action is executed "PS", number of belief states with more than one parent in the plan "DS", and the number of terminal belief states "TS".

We see that the probabilistic conformant planning graph heuristic does indeed improve scalability, as we hoped. $MOLAO^{*r}$ is only able to find a plan options for the smallest First-Responder instance up to a 0.75 probability of goal satisfaction. However, using the heuristic in $MOLAO^{*h}$ and $MOLAO^{*hr}$ allows $POND$ to find high probability plan options for every instance. While the expected cost is slightly better without using the heuristic, we see the benefit of using an inadmissible, but informative heuristic. In the instances requiring the heuristic, the number of plan belief states (PS) is quite large, testifying to the scale of the plans. The number of plan belief states with more than one parent (DS) shows the benefit of using $MOLAO^*$ to find more compact digraph plans. The number of terminal belief states (TS) attests to the number of branches in the plans (modulo the number of collapsed branches, captured by DS).

To visualize how our planner identifies plan options over time, consider Figure 5. The figure shows the values for the set of plan options starting in b_I over time (after each iteration in `ExpandPlan`) in the FR1 domain while using $MOLAO^{*h}$. Each line represents a solution satisfying the goal with the indicated probability and cost. The set of plan options steadily increases over time, suggesting how $MOLAO^*$ can be used in an anytime fashion.

Empirical Evaluation: Execution

To this point, we have mostly discussed multi-option plan synthesis, but now turn to execution strategies. Recall that in the absence of an objective function, the plan executor can try to "keep their options open". We explore three strategies for plan execution (i.e., option selection) that keep options open: preferring actions that appear in the most options, preferring actions that appear in the most diverse set of options, and preferring actions by sampling from the set of options.

Most Options: An executor may wish to keep as many options available during the entire execution, and in each state s execute the action a, where

$\text{argmax}_{a \in A(s)} |\{Q(s,a) | Q(s,a) \in V(s)\}|$

Most Diverse: An executor may be willing to sacrifice the sheer number of options for having a diverse set of options. The action that gives the most diverse options has the largest average distance between its options

$$\text{argmax}_{a \in A(s)} \frac{\sum_{Q(s,a),Q'(s,a) \in J(s)} dist(Q(s,a),Q'(s,a))}{|\{Q(s,a)|Q(s,a) \in J(s)\}|}$$

where, and $dist(Q(s,a),Q'(s,a))$ is the Euclidean distance between the points.

Random: As a baseline for comparison an executor may change their objective function at each step randomly,

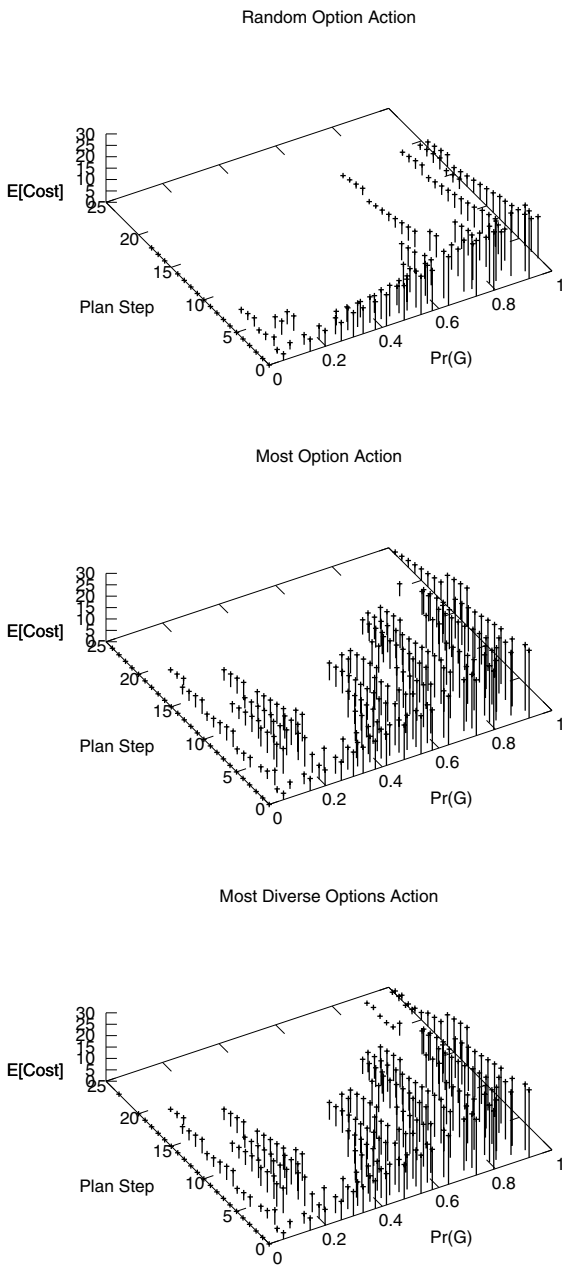

Figure 6: E[C] vs. Pr(G) .

choosing an action at random.

We compare these three strategies informally in the plots in Figure 6, showing a sample plan execution in terms of the options available at each plan step. At each plan step, there are a set of options that differ by their cost and probability of goal satisfaction. The choices made by the plan executor dictate the options available at the next plan step. While the results in the plots only compare a single execution of the same plan under three execution strategies, we see some interesting relationships. The random plan executor quickly runs out of options, where selecting actions in the most options or the most diverse options leads to more options throughout execution. Preferring the actions in the most diverse options leads to many options in more points in the plan. At the last few steps of the plan, choosing the diverse option action leads to more options than choosing the action with the most options. We hope to expand upon these results in future work and support diverse execution by exploring diverse plan synthesis.

Related Work & Discussion

Our formulation of multi-option conditional probabilistic planning is based on existing work techniques for vector-valued MDPs (Henig, 1983). We define cost and probability of goal satisfaction as values optimized in the MDP. Where previous work concentrates on value iteration, we extend the LAO^* algorithm to deal with multiple objectives.

Multi-option plans are also connected to work on finding sets of diverse plans (Myers & Lee, 1999; Srivastava *et al.*, 2007). A multi-option plan contains several options that are diverse in terms of the objectives that they optimize. We view this as an alternative notion of diversity that relies on quantitative plan metrics, as opposed to qualitative (Myers & Lee, 1999) or subjective plan distance functions (Srivastava *et al.*, 2007).

Using Pareto sets to represent multi-option plans gives us a foundation for representing diverse plan options – non-dominated plans are diverse because they satisfy the plan metrics in some unique way. However, our search algorithm does not actively pursue diverse plans. We intend to develop extensions of $MOLAO^*$ that guide the search to find a diverse set of plans through carefully crafting which heuristic points $Q(s, \star)$ we add to the Pareto sets. We can analyze Pareto sets to find regions that have relatively few options and insert heuristic points to estimate and guide the search for options in the regions.

In contrast with finding more options in specific regions of the plan metric space, we often have too many options. We presented ϵ-domination as a way to reduce the number of options, and increasing ϵ ensures that the options are increasingly diverse. However, like limited contingency planning (Meuleau & Smith, 2003), we may want to have a small number of options within some constant. It is reasonably straight-forward to constrain the number of options to a constant number, keeping those that are the most diverse or closest to a hypothesis of the user's objective function.

Conclusion & Future Work

To support model-lite planning, where users have ill defined and changing optimization criterion, we have presented multi-option plans. Multi-options plans give the user a set of diverse options relevant to their current state. By seeing the set of options, it should be easier for users to decide on the most meaningful optimization criterion.

Our multi-option plans can also be seen as a new type of conditional plan, where in addition nature selected "chance" contingencies, there are "choice" contingencies. These choice contingencies (options) allow the executor to delay

their choice of planning objective and/or change their objective multiple times.

In addition to this work on poorly defined objective functions, we see an important follow-on to this work in model-lite planners. When the user does not even specify the individual planning objectives (e.g., cost, makespan, etc.), it is up to the planner to identify objectives that make the plan options diverse. This can be seen as feature selection from a set of possible plan features, where the data (partial plans) are changing over the search episode.

Acknowledgements: We would like to thank Shlomo Zilberstein, Mausam, David E. Smith, David Musliner, and Sungwook Yoon for helpful discussions. This work was supported the NSF grant IIS-0308139, the ONR Grant N000140610058, the ARCS foundation, and an IBM Faculty Award.

References

Barto, A. G.; Bradtke, S.; and Singh, S. 1995. Learning to act using real-time dynamic programming. *AIJ* 72:81–138.

Bryant, R. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers* C-35(8):677–691.

Bryce, D.; Kambhampati, S.; and Smith, D. 2006. Sequential monte carlo in probabilistic planning reachability heuristics. In *Proceedings of ICAPS'06*.

Hansen, E., and Zilberstein, S. 2001. LAO: A heuristic-search algorithm that finds solutions with loops. *AIJ* 129(1–2):35–62.

Henig, M. 1983. Vector-valued dynamic programming. *Siam J. Cont.* 21:490–499.

Hyafil, N., and Bacchus, F. 2004. Utilizing structured representations and CSPs in conformant probabilistic planning. In *Proceedings of ECAI'04*, 1033–1034.

Kambhampati, S. 2007. Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain theories. In *Proceedings of AAAI'07*.

Madani, O.; Hanks, S.; and Condon, A. 1999. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proc. of AAAI'99*, 541–548.

Meuleau, N., and Smith, D. 2003. Optimal limited contingency planning. In *Proceedings of UAI-03*.

Myers, K., and Lee, T. 1999. Generating qualitatively different plans through metatheoretic biases. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. AAAI Press.

Nilsson, N. 1980. *Principles of Artificial Intelligence*. Morgan Kaufmann.

Papadimitriou, C. H., and Yannakakis, M. 2003. The complexity of tradeoffs, and optimal access of web sources. In *Proc. of FOCS-03*.

Srivastava, B.; Nguyen, T.; Gerevini, A.; Kambhampati, S.; Do, M. B.; and Serina, I. 2007. Domain independent approaches for finding diverse plans. In *Proc. of IJCAI-07*, 2016–2022. Hyderabad, India: IJCAI.

Wu, J., and Azram, S. 2001. Metrics for quality assessment of a multiobjective design optimization solution set. *Journal of Mechanical Design* 123:18–25.