A Switching Planner for Combined Task and Observation Planning

anonymous

Abstract

Realistic robot planning problems in uncertain environments often require achieving tasks while also finding out about the world. Because the world state cannot be observed directly, these problems are naturally represented as partially observable Markov decision problems (POMDPs). However, these are typically intractable for realistic problems. We present a *switching planner* that employs fast sequential planning to decide on the overall strategy, and uses a decision-theoretic planner to solve the subproblems where partial observability will significantly impact the quality of the plan. We demonstrate the approach in a realistic robot exploration domain.

1 Introduction

Planning for agents acting in partially observable environments with stochastic actions is extremely computationally expensive [Mundhenk *et al.*, 2000]. Even relatively small partially observable Markov decision problems (POMDPs), represented by tens of propositional variables, have state spaces far larger than existing optimal or close to optimal planners can handle. For the easier problem of planning in completely observable stochastic domains, an alternative has been to use classical sequential planners and replan when actions have unplanned-for outcomes [Yoon *et al.*, 2007]. Execution monitoring tracks properties of the state that determine whether further planning is required. For partially observable domains, only probabilistic information about such properties will, in general, be available.

This paper describes the planning component for a robotic system that continuously deliberates in a stochastic dynamic environment in order to achieve objectives set by the user, and acquire knowledge about its surroundings. This domain features partial observability, particularly because the state is not perfectly known, and state information is gained by the robot through the use of unreliable sensing actions. For interesting sized tasks the corresponding POMDP model is too large for a POMDP solver to be applied directly. Inspired by the replanning approaches used successfully for MDP planning [Yoon *et al.*, 2007; 2008], we propose a continual planning approach that uses a classical planning system to compute a reasonably valuable trace in the model, and then uses a decision-theoretic planner on small subproblems where reasoning about observations might be useful. We refer to the complete system as a *switching continual planner*. The approach is not optimal, particularly as it relies on the results of sequential planning directly. It does nevertheless perform better than a purely sequential replanner, and is fast enough to be used for real-time decision-making on a mobile robot.

Our system is domain independent, taking domain and problem descriptions in a first-order declarative language we have developed, called the decision-theoretic planning domain definition language (DTPDDL). From these descriptions of POMDPs, we automatically construct a deterministic model for the sequential planner. That model includes actions which correspond to making assumptions about the values of imperfectly known state variables. Assumptions scheduled by the sequential system are used to propose a pragmatic abstract belief-state space to the decision-theoretic system, and to modify the reward function, so that system might pursue sensing related to those assumptions.

The switching planner provides important advantages in our mobile robot domain. First, the significant source of uncertainty in the domain is the unreliability of observations and this approach is particularly suited to problems that combine a deterministic task planning problem with observations because the decision-theoretic planner is only invoked to deal with state uncertainty. Secondly, replanning makes the system robust to changing objectives and discoveries about the world, both of which feature in our domain.

The remainder of the paper proceeds as follows. We begin by describing POMDPs with a propositionally factored representation of states and observations. We also outline the relationship between this and a flat POMDP representation. Next we describe our DTPDDL language with example declarations from a mobile robot exploration task. Then we describe the switching planner, including how the sequential domain is produced and a plan is generated, and how this then provides the input for the decision-theoretic planner. Finally we provide an evaluation of the system on the example robot domain, and finish with a short survey of related work and some concluding remarks.

2 Propositional Decision-Theoretic Planning

We describe the partially observable propositional probabilistic planning problem, with costs and rewards. We model a process state s as the set of propositions that are true of the state. Notationally, $\mathcal{P}$ is the set of propositions, p is an element from that set, and we have $s \subseteq \mathcal{P}$. The underlying process dynamics are modelled in terms of a finite set of *probabilistic STRIPS operators* [Boutilier and Dearden, 1994] $\mathcal{A}$ over state-characterising propositions $\mathcal{P}$. We say an action $a \in \mathcal{A}$ is applicable if its precondition $\texttt{pre}(a)$, a set of propositions, are satisfied in the current state – i.e., $\texttt{pre}(a) \subseteq s$. We denote by $\mu_a(a_i^d)$ the probability that nature chooses a deterministic STRIPS effect a_i^d, and for all a we require $\sum_{a_i^d \in \mathcal{A}^d(a)} \mu_a(a_i^d) = 1$.

We are concerned with problems that feature partial observability. Although we could invoke *extended probabilistic STRIPS operators* [Rintanen, 2001] to model actions and observations propositionally, we find it convenient for presentation and computation to separate sensing and action. Therefore, we suppose a POMDP has a perceptual model given in terms of a finite set of stochastic *senses* K, deterministic sensing outcomes K^d, and perceptual propositions Π, called *percepts*. In detail, we take an observation o to be a set of percepts $o \subseteq \Pi$, and denote by O the set of reachable observations. The underlying state of the process cannot be observed directly, rather, senses $\kappa \in K$ effect an observation $o \in O$ that informs what should be believed about the state the process is in. If a is applied effecting a transition to a successor state s', then an observation occurs according to the active senses $K(a, s') \subseteq K$. A sense κ is active, written $\kappa \in K(a, s')$, if the senses' action-precondition, $\texttt{pre}_{\mathcal{A}}(\kappa)$, is equal to a, and the state-precondition $\texttt{pre}_{\mathcal{S}}(\kappa) \subseteq \mathcal{P}$ is satisfied by the state s' – In the usual sense that $\texttt{pre}_{\mathcal{S}}(\kappa) \subseteq s'$. When a sense is active, nature must choose exactly one outcome amongst a small set of deterministic choices $K^d(\kappa) \equiv \{\kappa_1^d, \ldots, \kappa_k^d\}$, so that for each i we have $\kappa_i^d \subseteq \Pi$. The probability of the i^{th} element being chosen is given by $\psi_\kappa(\kappa_i^d)$, where $\sum_{\kappa_i^d \in K^d(\kappa)} \psi_\kappa(\kappa_i^d) = 1$. The observation received by the agent corresponds to the union of perceptual propositions from the chosen elements of active senses.

A POMDP has a starting configuration that corresponds to a Bayesian belief-state. Intuitively, this is the robot's subjective belief about its environment. Formally, a belief-state b is a probability distribution over process states. We write $b(s)$ to denote the probability that the process is in s according to b, and b_0 when discussing the starting configuration.

Finally, we make some fairly standard assumptions. First, that action execution and sensing occur instantaneously, and that only one action can be applied per plan-step. Secondly, it can arise that in some states an action is applicable, and in others it is not. Moreover, a belief-state can assign non-zero probability to states where that applicability holds, and states where it does not. Unlike Hoffmann and Brafman (2006), we incorporate a PPDDL-like default semantics, treating an action a executed at s_i as if it has no effect at states s if $\texttt{pre}(a) \not\subseteq s$.

2.1 Costs, Rewards, and Belief Revision

Until now we have discussed the POMDP in terms of propositions and percepts. In order to address belief revision and utility it is convenient to consider the underlying decision process in a flat format. This is given by the tuple $\langle \mathcal{S}, b_0, \mathcal{A}, \mathsf{Pr}, \mathsf{R}, O, v \rangle$. Here, b_0 is the initial belief-state, $\mathcal{S}$ is the finite set of reachable propositional states, $\mathcal{A}$ is the finite set of actions, and O is the finite set of reachable observations (i.e., perceptual states). Where $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$, from μ we have a state transition function $\mathsf{Pr}(s, a, s')$ giving the probability of a transition from state s to s' if a is applied. For any s and a we have $\sum_{s' \in \mathcal{S}} \mathsf{Pr}(s, a, s') = 1$. The function $\mathsf{R} : \mathcal{S} \times \mathcal{A} \to \Re$ is a bounded real valued reward function. Therefore a finite positive constant c exists so that for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, $|\mathsf{R}(s, a)| < c$. We model costs as negative rewards. From ψ we have that for each $s' \in \mathcal{S}$ and action $a \in \mathcal{A}$, an observation $o \in O$ is generated independently according to a probability distribution $v(s', a)$. We denote by $v_o(s', a)$ the probability of getting observation o in state s'. For s' and a we have $\sum_{o \in O} v_o(s', a) = 1$.

Successive state estimation is by application of Bayes' rule. Taking the current belief b as the *prior*, and supposing action a is executed with perceptive outcome o, the probability that we are in s' in the successive belief-state b' is:

$$b'(s') = \frac{v_o(s', a) \sum_{s \in \mathcal{S}} \mathsf{Pr}(s, a, s') b(s)}{\texttt{Pr}(o|a, b)} \qquad (1)$$

where $\texttt{Pr}(o|a, b)$ is the normalising factor, that is, the probability of getting observation o given we execute a in b.

2.2 Plan Evaluation

An optimal solution to a finite-horizon POMDP is a contingent plan, and can be expressed as a mapping from observation histories to actions. Although suboptimal in general, useful plans can also take a classical sequential format. This is the case in *conformant* planning, where the objective is to find a sequence of actions that achieves a goal —i.e., reaches a state that satisfies a given Boolean condition— with probability 1. Generally, whatever the structure of a plan, its value corresponds to the expected cumulative reward:

$$V_{\texttt{PLAN}}(b) = \mathsf{E}\left[\sum_{t=0}^{N-1} \mathsf{R}(b_t, \texttt{PLAN}_t) \mid \texttt{PLAN}, b_0 = b \right] \qquad (2)$$

Where b_t is the belief-state at step t, $\texttt{PLAN}_t$ is the action prescribed at step t, and

$$\mathsf{R}(b, a) = \sum_{s \in \mathcal{S}} b(s) \mathsf{R}(s, a).$$

3 Decision-Theoretic PDDL

The modelling language of choice for planning in probabilistic problems is the Probabilistic Planning Domain Definition Language [Younes and Littman, 2004; Younes *et al.*, 2005]. PPDDL has been used in all three of the International Planning Competitions since 2004. A variation on PDDL for describing domains with stochastic actions and uncertain starting configurations, PPDDL is a declarative first-order language that facilitates factored descriptions of domains and

problems. There are straightforward compilations from problems expressed in PPDDL to propositional representations amenable to state-of-the-art planners.

Because PPDDL cannot model domains that feature partial observability, we develop an extension we call *Decision-Theoretic (DT)PDDL*. This can express probabilistic models of the sensing consequences of acting, to quantitatively capture unreliability in perception. That expressive power is achieved by incorporating perceptual analogues of fluent, predicate, and action definitions. In detail, we have declarations of state characterising predicate and fluent symbols according to the PPDDL syntax. In addition, we allow two other declarations, for perceptual predicates and fluents respectively. For example, suppose our robot is tasked with exploring *locations* in order to identify the whereabouts of a *visual-object*. We must describe state and perceptual facts that model the *true*, resp. perceived, locations of objects. In DTPDDL, these declarations appear as:

```
(:functions ;; state fluents
 (is-in ?v - visual-object) - location )
(:observable-functions ;; perceptual fluents
 (o-is-in ?v - visual-object) - location )
```

To model the sensing capabilities of the agent, we have operator-like declarations, with preconditions expressed using state predicate and action function symbols, and uniformly positive effects over perceptual predicates. A sense declaration corresponds to a lifted description of proposition *senses*. Whereas the effects of an *operator* schema describe how states change under application of actions, the effects of a *sense* schema are perceptual, specifying the composition of an observation following the execution of an action.

Making the above ideas concrete with an example, suppose a robot called DORA is able to look for a visual-object, such as a box, at a given place. We can model that deterministic action using the following operator schema:

```
(:action look-for-object
 :parameters (?r - robot ?v - visual-object
              ?l - location)
 :precondition (and (= (is-in ?r) ?l) )
 :effect (and (assign (reward) -3) ) )
```

In a slight departure from PPDDL, we suppose that look-for-object has an effect on the state. That is, its execution incurs an instantaneous penalty, 3, that corresponds to the *cost* of performing the visual search.[1]

Completing the example, we model the sensing that results from instantiating the look-for-object operator:

```
(:sense vision
 :parameters (?r - robot ?v - visual-object
              ?l - location)
 :execution  (look-for-object ?r ?v ?l)
 :precondition (and (= (is-in ?r) ?l) )
 :effect     (and (when (= (is-in ?v) ?l)
                  (probabilistic 0.8
                  (= (o-is-in ?v) ?l)))
              (when (not (= (is-in ?v) ?l))
                  (probabilistic 0.1
                  (= (o-is-in ?v) ?l)))))
```

Here, the *execution* declaration is a lifted description of the sense action-precondition – i.e, (look-for-object DORA box office) is equal to $\mathtt{pre}_{\mathcal{A}}(\mathtt{vision\ DORA\ box\ office})$. Above, we include a redundant state-precondition, $(= (\mathtt{is\text{-}in}\ ?r) ?l)$ in order to fully demonstrate the syntax. Interpreting the above schema, if action (look-for-object DORA box office) is executed, there is a 0.8 chance of perceiving the box if it is in the office, and otherwise a 0.1 chance of perceiving it.

The syntax and semantics for describing initial state distributions in DTPDDL is taken verbatim from PPDDL. It is useful to present that factored tree-like structure here, in order to aid us when discussing our switching continual planner. The distribution is expressed in a tree-like structure of terms. Each term is either: (1) atomic, e.g., a state proposition such as $(= (\mathtt{is\text{-}in}\ \mathtt{box})\ \mathtt{office})$, (2) probabilistic, e.g., $(\mathtt{probabilistic}\ \rho_1(T_1)..\rho_n(T_n))$ where T_i are conjunctive, or (3) a conjunct over probabilistic and atomic terms. The root term is always conjunctive, and the leaves are atomic. For example, the starting distribution for DORA might be described as follows:[2]

```
(:init (= (is-in DORA) kitchen)
 (probabilistic .8 (= (is-in box) office)
                .2 (= (is-in box) kitchen))
 (probabilistic .3 (= (is-in cup) office)
                .7 (= (is-in cup) kitchen)))
```

The interpretation of such an expression can be given by a *visitation* of terms: An atom is *visited* iff its conjunctive parent is visited, and a conjunctive term is visited iff all its immediate subterms are visited. A probabilistic term is visited iff its conjunctive parent is visited, and exactly one of its subterms, T_i, is visited. Each visitation of the root term according to this recursive definition encapsulates a starting state, along with the probability that it occurs. The former corresponds to the union of all visited atoms, and the latter corresponds to the product of ρ_i entries on the visited subterms of probabilistic elements. Making these ideas concrete, our example yields the following flat state distribution:

Probability	(is-in DORA)	(is-in box)	(is-in cup)
.24	kitchen	office	office
.06	kitchen	kitchen	office
.56	kitchen	office	kitchen
.14	kitchen	kitchen	kitchen

From hereon, to simplify the discussion (and implementation) we shall restrict our attention to POMDPs with deterministic actions. We note that POMDPs with stochastic actions can be compiled into equivalent deterministic-action POMDPs, where all the original action uncertainty is expressed in the starting-state distribution [Ng and Jordan,

[1] In PPDDL *reward* is a *reserved word*, and occurs in *increased* and *decreased* terms in operator schemata. In that setting, reward is accumulated, whereas in DTPDDL it is instantaneous.

[2] In PDDL, $(\mathtt{:init}\ T_1..T_n)$ expresses the conjunctive root of the tree – i.e., the root node $(\mathtt{and}\ T_1..T_n)$. Also, we shall write p, rather than $(\mathtt{and}\ p)$, for conjunctive terms that contain a single atomic subterm.

2000]. In our setting, that of finite-horizon planning, such a compilation yields a finite flat representation of the original POMDP, only with deterministic actions.

4 Switching Continual Planner

We describe our approach that switches between sequential and contingent sessions. As a continual planning approach, it proceeds by interleaving planning and execution in a deterministic-action POMDP described in DTPDDL. During a sequential session, planning is performed by a "classical" system,[3] and execution proceeds according to the *trace* computed by that system. Taking the form of a classical plan, the trace specifies a sequence of POMDP actions that achieve the agent's objectives in a deterministic approximation, i.e., *determinisation*, of the problem at hand. More precisely, the trace is an interleaved sequence of POMDP actions and *assumptive* actions. The latter correspond to assumptions the planner makes about the truth value of propositions – e.g. that a box of cornflakes is located in the kitchen at the third plan step. They are called *applicability* assumptions if the trace includes an action a that is not applicable with probability 1 at the belief-state b that the system is projected to be in when a is scheduled for execution, i.e., $\exists s \in \mathcal{S}\ b(s) > 0$ and $\texttt{pre}(a) \not\subseteq s$. By scheduling a, the serial planner makes an assumption about the observability of the precondition $\texttt{pre}(a)$.

Our approach always begins with a sequential session. Non-assumptive actions from the trace are executed in sequence until the applicability of the next scheduled action is uncertain. We denote that action by $\tilde{a}$. A contingent session then begins that tailors sensory processing by the agent to determine whether the assumptions made in the trace hold. We add *disconfirm* and *confirm* actions to the model, so the session is encouraged to judge the assumptions made in the trace. On execution of one of those actions, control is returned to the sequential session that continues at the currentbelief-state. Because contingent planning is only practical in relatively small POMDPs, contingent sessions plan in an abstract decision process determined by the current trace. This abstraction is constructed by first omitting all propositions that do not feature in the trace, and by then iteratively refining the model while the result is of a practicable size.

Finally, whether proceeding in a sequential or contingent session, our continual planner maintains a factored representation of successive belief-states by performing belief revision according to Eq 1. Moreover, our internal representation of the underlying belief-state corresponds more-or-less directly to an (:init) declaration. Our approach uses that distribution: (1) as the source of candidate determinisations for sequential planning, (2) to determine when to switch, and (3) as a mechanism to guide construction of an abstract process for contingent sessions.

4.1 Sequential Planning in POMDPs

A sequential session uses a classical planner to compute a trace. The latter encapsulates assumptions about: (1) the true underlying state, (2) how execution will progress, and (3) the possibility of the agent eventually holding strong beliefs about the truth values of specific state propositions. Here we describe the deterministic planning problem, derived from the DTPDDL model, that admits plans which correspond to traces.

In a deterministic-action POMDP all the uncertainty in state and action is expressed in the (:init) declaration. Our approach uses the structure of that, as it occurs in the problem description, to define a set of state-assumptions available to sequential planning. Writing # if the value of a proposition is unspecified, for DORA we have the following assumptions:

Probability	(is-in DORA)	(is-in box)	(is-in cup)
.24	kitchen	office	office
.06	kitchen	kitchen	office
.56	kitchen	office	kitchen
.14	kitchen	kitchen	kitchen
.7	kitchen	#	kitchen
.3	kitchen	#	office
.8	kitchen	office	#
.2	kitchen	kitchen	#
1.0	kitchen	#	#

An assumption corresponds to a *relaxed* visitation of the root term of (:init). In this relaxed case, a conjunctive term is visited iff its atomic subterms are visited, and zero or more of its immediate probabilistic subterms are visited. The starting state, s_0, for sequential planning is an *abstract* state that captures the unique assumption which has probability 1. In DORA, that is:

$$s_0 \equiv \{(= (\texttt{is-in DORA})\ \texttt{kitchen}),\\ (= (\texttt{is-in box})\ \#), (= (\texttt{is-in cup})\ \#)\}.$$

In order to make assumptions available to the sequential planner, we add one *assumptive* action $\mathcal{A}^\circ(\rho_i; T_i)$ to the problem for each element, $\rho_i(T_i)$, of each probabilistic term from (:init). The physics of these actions is as follows: $\mathcal{A}^\circ(\rho_i; T_i)$ can be executed if no $\mathcal{A}^\circ(\rho_j; T_j)$, $j \neq i$, has been executed from the same probabilistic term, and, either (probabilistic ..ρ_i (T_i)..) is in the root conjunct, or it occurs in T_k for some executed $\mathcal{A}^\circ(\rho_k; T_k)$. Also, an assumption about a proposition p can only be made once, and must be sequenced before any POMDP actions that have p in their precondition or effect. In other words, an assumptive action must be applied to make a truth assignment to p, and then that truth assignment can be modified by the DTPDDL domain operators. Executing $\mathcal{A}^\circ(\rho_i; T_i)$ in s effects a transition to a successor state s^{T_i} with probability ρ_i, and $s^\perp$ with probability $1 - \rho_i$. Here, s^{T_i} is the union of s with atomic terms from T_i with the proposition denoting the undefined fact is deleted. For example, if the action adds (= (is-in box) office), then it must delete (= (is-in box) #). State $s^\perp$ is an added sink, where $\forall a\ \mathsf{R}(s^\perp, a) = 0$ and $\mathsf{Pr}(s^\perp, a, s^\perp) = 1$. Lastly, we have that if a has a precondition that is not true with probability θ, a given threshold, in the underlying belief b, then immediately before it is executed blindly the system switches to a contingent session (described below).

We now describe the optimisation criteria given to the classical planner. Where ρ_i is the probability that the i^{th} sequenced action a_i does not transition to $s^\perp$, we define the value of a trace $s_0, a_0, s_1, a_1, .., s_N$ from the deterministic model to equal:

[3] That is, a planner designed to solve fully observable deterministic tasks.

$$V(s_0, a_0, s_1, a_1, .., s_N) = \prod_{i=1..N-1} \rho_i \sum_{i=1..N-1} \mathsf{R}(s_i, a_i) \quad (3)$$

The optimal trace given a plan, i.e., the sequence of non-assumptive actions from the trace, therefore has value:

$$V^* = \max_{\rho_1,..,\rho_n} \prod_{i=1..N-1} \rho_i \sum_{i=1..N-1} \mathsf{R}(s_i, a_i),$$

which is equal to the maximal additive contribution a trace of the plan can make to Eq 2. In goal directed problems, where the only non-zero reward is received at the first transition to a goal state, the Eq 3 criteria gives us the behaviour sought by FFR$_a$ [Yoon *et al.*, 2007]. From the perspective of gradient-based reinforcement learning systems, such as William's REINFORCE and GPOMDP variants for planning [Buffet and Aberdeen, 2009], under reasonable assumptions the optimal trace identifies a maximal gradient step from a uniformly random soft-max policy. Finally, it is worth clarifying that although the above discussion has been in terms of a trace of length N, there is nothing in our approach that artificially limits the length of candidate sequential plans.

4.2 Decision-Theoretic Planning in Abstractions

In a contingent session our switching planner solves a sensing problem in an abstract processes defined in terms of the assumptive actions in the current trace, and the (`:init`) declaration that models the underlying belief-state.

The focus for contingent planning is on sensing, we therefore give the process in this session an augmented reward model. This reflects the value of performing sensing actions in the context of the trace proposed by the preceding sequential deliberation. First, all rewards from the original POMDP are retained. Then, for each $\mathcal{A}^\circ(\rho_i; T_i)$ action scheduled by the current trace, we have a *dual* $\mathcal{A}^\bullet(\rho_i; T_i)$ so that $\forall s \in \mathcal{S}$:

$$\mathsf{R}(s, \mathcal{A}^\bullet(\rho_i; T_i)) = \begin{cases} \$(T_i) & \texttt{if } \ T_i \not\subseteq s \\ \hat{\$}(T_i) & \texttt{otherwise} \end{cases}$$

where $\$(T_i)$ (resp. $\hat{\$}(T_i)$) is a positive (negative) numeric quantity which captures the utility the agent receives for correctly (incorrectly) rejecting an assumption. The added dual actions are propositionally trivial – i.e., $\forall s \in \mathcal{S}$ $\texttt{pre}(\mathcal{A}^\bullet(\rho_i; T_i)) \subseteq s$ and $\mathcal{A}^\bullet(\rho_i; T_i)(s) \equiv s$. In order to reduce the number of possible rejections we omit assumptions that are not *active* with respect to the action, $\tilde{a}$, whose scheduled execution switched the system into the contingent session. An assumption $\mathcal{A}^\circ(\rho_i; T_i)$ is active if $\tilde{a}$ is not applicable in the sequential model unless $\mathcal{A}^\circ(\rho_i; T_i)$ is scheduled in the trace prefix to $\tilde{a}$. For example, if DORA's trace is:

$\mathcal{A}^\circ(.8; (= (\texttt{is-in box})\texttt{office}))$;
$\mathcal{A}^\circ(.3; (= (\texttt{is-in cup})\texttt{kitchen}))$;
(`look-for-object` DORA `box office`);
(`look-for-object` DORA `cup kitchen`);
(`report-is-in box office`);
(`report-is-in cup kitchen`)

Taking the switching action $\tilde{a}$ to be (`look-for-object DORA box office`), we have that $\mathcal{A}^\circ(.3; (= (\texttt{is-in cup})\texttt{kitchen}))$ is not active, and therefore exclude $\mathcal{A}^\bullet(.3; (= (\texttt{is-in cup})\texttt{kitchen}))$ from the POMDP posed to the contingent session.

We also include another propositionally trivial action $\mathcal{A}.\texttt{pre}(\tilde{a})$ with the reward property:

$$\mathsf{R}(s, \mathcal{A}.\texttt{pre}(\tilde{a})) = \begin{cases} \$(\texttt{pre}(\tilde{a})) & \texttt{if } \ \texttt{pre}(\tilde{a}) \subseteq s \\ \hat{\$}(\texttt{pre}(\tilde{a})) & \texttt{otherwise} \end{cases}$$

For the purposes of successive switching, execution of either a dual action of the form $\mathcal{A}^\bullet(\rho_i; T_i)$, or the $\mathcal{A}.\texttt{pre}(\tilde{a})$, returns control to a sequential session. If a dual is executed, then the sequential planner must replan with b_0 equal to the underlying belief-state. Otherwise, if $\mathcal{A}.\texttt{pre}(\tilde{a})$ is executed, then the current sequential plan is executed until further sensing is scheduled, or to completion. Lastly, we note that in our project the rewards, i.e., functions $\$, \hat{\$} : 2^{\mathcal{P}} \rightarrow \Re$, for the POMDP posed to the contingent session are sourced from a motivational component of our robotic architecture. In this paper, for $\mathcal{A}^\bullet(\rho_i; T_i)$ actions we set $\$(x)$ to be a positive constant, and have $\hat{\$}(x) = -\$(x)(1-\rho)/\rho$ where ρ is the probability that x is true. For $\mathcal{A}.\texttt{pre}(\tilde{a})$ actions we have $\hat{\$}(x) = -\$(x)\rho/(1-\rho)$.[4]

The starting belief-state for the abstract process is given in terms of a *relaxed* (`:init`) declaration. We first construct a tree which only includes terms from the *relaxed* visitation characterised by the active *assumptive* actions in the current sequential plan. Figure **??** shows the belief-state of a more detailed version of our previous example. The diamonds represent probabilistic terms of the state description while circles represent conjunctive or atomic terms.

In the first step, we remove all facts that are not part of an assumption (Fig. **??**). At this point, the session would proceed in an abstraction of the environment that does not contain `place1`, the `cup` or a `kitchen`. In a second step, we iteratively refine the relaxed declaration by adding terms from the original statement of (`:init`) while the number of abstract states in b_0 that occur with non-zero probability according to that refined declaration remains of a practicable size. In detail, for each excluded atomic term, we compute the *entropy* of the active assumptions, *conditional* on the corresponding proposition. Loosely speaking, the lower the *conditional entropy* of an atom, the more information about the assumptions the contingent planner expect to derive from it. Thus, propositions that in the first place were excluded are iteratively added to the problem posed to the conditional session in increasing order according to that measure (Fig. **??**).

5 Evaluation

We have implemented our switching continual approach in the MAPSIM environment [Brenner and Nebel, 2009]. We have extended MAPSIM so that it can parse DTPDDL, and perform successive estimation of the underlying belief-state using DLIB-ML [King, 2009] for inference. In this evaluation we use our own version of Fast Downward [Helmert, 2006] for sequential sessions. We have extended that system to support actions with success probabilities. In our evaluation Fast

[4] These values were chosen to encourage sensing.

Downward is run with the cyclic causal graph heuristic using an A* search or weighted A* (with weight 5). We use A* for easy problems and WA* for more difficult problems where A* is ineffective. Contingent sessions use our own forward search procedure.

In order to evaluate our system, we have also implemented a baseline approach in MAPSIM. Rather than invoking a contingent session when a switching action $\tilde{a}$ is scheduled for execution, the baseline simply executes one action that can trigger a sensing outcome determined by the precondition of $\tilde{a}$, and replans in the resulting belief-state.

To test our approach we use a robot exploration domain based on a scenario from our physical robotic system. Here, a robot is exploring an office or living environment, and trying to report the locations of objects to their owner. Basic types are `rooms`, `places` and `objects`. Places are topological map nodes that occur in rooms. Objects, and also the robot, are always located at a place. The robot can move around the rooms via connections between places given by a `connected` predicate. Each room has a, possibly unknown, *category* (e.g. kitchen, office, living room). Certain objects are more likely to be located in rooms of a particular category. For example, a box of cornflakes is more likely to be located in the kitchen than the office. The robot can look for an object at a place by executing a `look-for-object` action. Executing `look-for-object` results in a noisy sensing outcome, sometimes indicating a positive perception if the object is there. We have that some objects are harder to detect than others. Finally, if the robot is in the presence of a human, it can ask what type of room it is currently in, however conducting a dialogue is more costly than running a vision algorithm (cost of 8 vs costs of 3).

We evaluated our switching planner against the baseline in several tasks with the number of rooms ranging from 3 to 6. We examine the impact of sensing reliability, experimenting using sensor models that are: (*reliable*) 0.9 chance the object is perceived if it is present, (*semi-reliable*) chance is 0.65, and (*noisy*) chance is 0.4. The continual planning times are reported in Figure 1, and the quality data in Figure 2. Here, in switching runs we examine the performance of our system where b_0 allocates non-zero probability to between 20 and 200 abstract states during contingent sessions. For Item a-e the objective is to find one or more objects and report their position to the user. In some of the tasks we configured the environment so that no plan exists, by excluding the desired object from the environment. Item f reports for tasks requiring indirect sensing, where the robot must relocate to a particular room category. For Item f there is no human in the environment, and therefore the baseline cannot be used, because there are no sensing schemata that directly sense room categories. In our evaluation we run 50 simulations in each configuration, and have a timeout on each simulation of 30 minutes (1800 seconds)[5]. In our domain, rewards encapsulate a goal condition, and therefore we find it intuitive to report average costs of the plans and the success rate in problems that admit a solution (i.e., positive reward scaled by a constant factor).

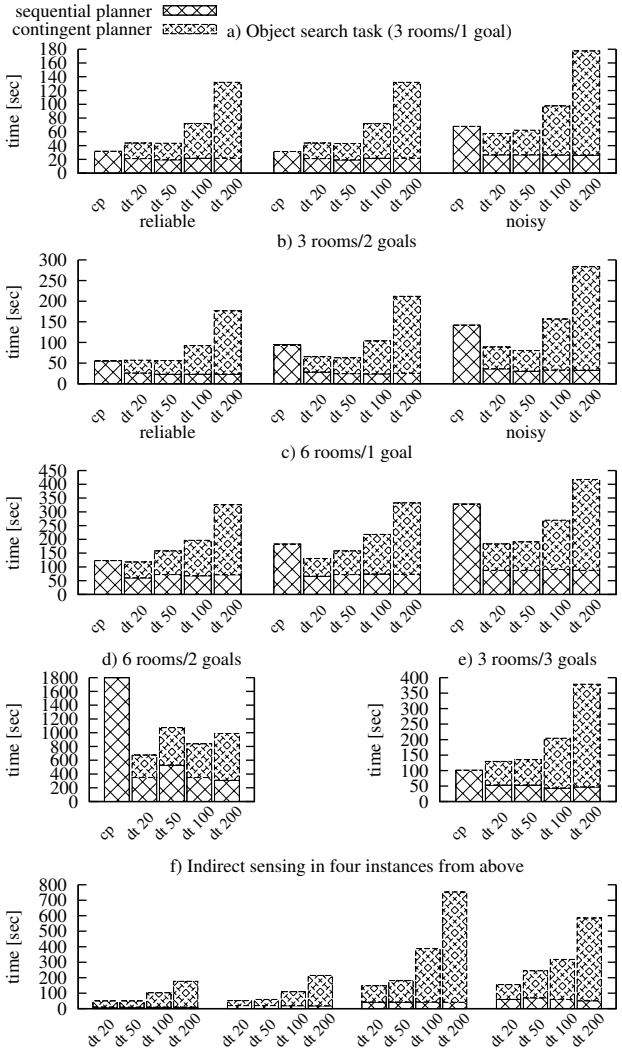

Figure 1: Average runtime

Examining success ratios and plan costs, where sensing is reliable there is little to be gained by using the contingent planner, as the greedy sensing approach of the baseline is sufficient. Not surprisingly, as sensing degrades contingent planning pays off. Also, we find that time spent in contingent planning increases steeply as the abstraction b_0 becomes more refined. That refinement seems to be paying off in terms of the success ratio, particularly for tasks d and e, where we had sequential sessions using weighted A^* (rather than A^*). For less refined initial configurations, the increase cost of contingent planning is compensated by a decrease in Fast Downward planning times. The relatively high success rate irrespective of the level of refinement in the initial configuration indicates the effectiveness of using conditional entropy to guide abstraction refinement in our setting.

[5]All experiments were conducted on a 2.3GHz AMD Opteron using one CPU core.

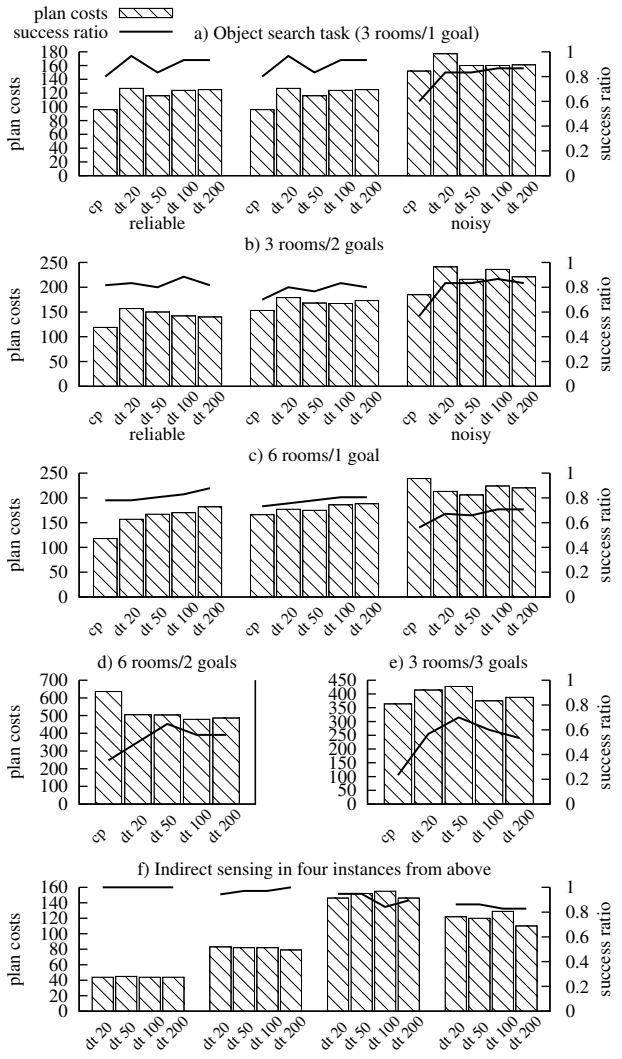

Figure 2: Average plan costs and number of successful runs.

6 Related Work

There have been a number of recent papers on planning under uncertainty using systems that were intended for sequential planning in deterministic problems, for example, FFR$_a$ [Yoon *et al.*, 2007], the winning entry from the probabilistic track of the 2004 International Planning Competition. In the continual paradigm, FFR$_a$ uses the fast satisficing procedure FF [Hoffmann and Nebel, 2001] to compute sequential plans and corresponding execution traces. More computationally expensive approaches in this vein combine sampling strategies on valuations over *runtime variables* with deterministic planning procedures. The outcome is typically a more robust sequential plan [Yoon *et al.*, 2008], or contingent plan [Majercik, 2006].

Also leveraging deterministic planners in problems that feature uncertainty, CONFORMANT-FF [Hoffmann and Brafman, 2006] and T_0 [Palacios and Geffner, 2009] demonstrate how conformant planning —i.e., sequential planning in unobservable worlds— can be modelled as a deterministic problem, and therefore solved using sequential systems. In this conformant setting, advances have been towards compact representations of beliefs amenable to existing best-first search planning procedures, and lazy evaluations of beliefs. Most recently this research thread has been extended to contingent planning in fully observable non-deterministic environments [Albore *et al.*, 2009]. The continual planning system that motivated our project [Brenner and Nebel, 2009] also has this characteristic, and has been applied in completely observable domains, particularly those featuring multiple communicating agents.

Our work is motivated by domains that contain a mixture of task planning and observation planning. There have been a number of recent papers representing observation planning problems as POMDPs and using various techniques to manage the large state space. Sridharan *et al.* (2010) take this approach in a vision algorithm selection problem. In their case there is a natural hierarchical decomposition of the problem which allows them to solve large problems by breaking them into a set of small POMDPs. Doshi and Roy (2008) represent a preference elicitation problem as a POMDP and take advantage of symmetry in the belief space —essentially the idea that it does not matter what the value of the variable you are trying to observe is— to exponentially shrink the state space. Although we have been actively exploring the Doshi and Roy approach, we have yet to find those exploitable structures in our domain due to the task planning requirement.

7 Concluding Remarks

From an automated planning perspective, the problem of practical mobile robot control poses important and contrary challenges. On the one hand, planning and execution monitoring must be lightweight, robust, timely, and should span the lifetime of the robot. Those processes must seamlessly accommodate exogenous events, changing objectives, and the underlying unpredictability of the environment. On the other hand, robot planning should perform computationally expensive reasoning about contingencies, and possible revisions of subjective belief according to quantitatively modelled uncertainty in acting and sensing.

In this paper we address these challenges, developing a continual planner that switches between fast sequential and expensive decision-theoretic planning. Given a POMDP model of the environment, sequential planning is used to compute an initial deterministic sequential plan and complementary runtime evolution of the decision process. That plan is executed until testing of an assumption about the runtime state is scheduled. At that point a contingent session performs some sensing which determines whether or not the sequential session should be allowed to continue, or otherwise acts to achieve the overall objectives.

The most pressing item for future research, is to develop a scheme whereby the serial planner can relax *executability* assumptions, so that conformant (or semi-conformant) plans can be executed without interruption by a contingent session. A more general criticism of switching continual planning concerns interleaved sequential and contingent sessions failing to make any progress towards the objectives. For ex-

ample, in the worst case we can have each sequential session producing an identical plan, and each decision-theoretic session rejecting it without further sensing. Although this has not been an issue in our work so far, it must be dealt with rigorously in the future. We suggest that a good way to mitigate this problem is by developing a *motivational* component that maintains a *dynamic* reward model whose limiting behaviour prevents that switching deadlock.

References

[Albore *et al.*, 2009] Alexandre Albore, Héctor Palacios, and Héctor Geffner. A translation-based approach to contingent planning. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1623–1628, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[Boutilier and Dearden, 1994] Craig Boutilier and Richard Dearden. Using abstractions for decision-theoretic planning with time constraints. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1016–1022, Seattle, 1994.

[Brenner and Nebel, 2009] Michael Brenner and Bernhard Nebel. Continual planning and acting in dynamic multiagent environments. *Journal of Autonomous Agents and Multiagent Systems*, 19(3):297–331, 2009.

[Buffet and Aberdeen, 2009] Olivier Buffet and Douglas Aberdeen. The factored policy-gradient planner. *Artificial Inteligence*, 173(5-6):722–747, 2009.

[Doshi and Roy, 2008] Finale Doshi and Nicholas Roy. The permutable POMDP: Fast solutions to POMDPs for preference elicitation. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems*, 2008.

[Helmert, 2006] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Hoffmann and Brafman, 2006] Jörg Hoffmann and Ronen I. Brafman. Conformant planning via heuristic forward search: a new approach. *Artif. Intell.*, 170:507–541, May 2006.

[Hoffmann and Nebel, 2001] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14, 2001.

[King, 2009] Davis E. King. Dlib-ml: A machine learning toolkit. *J. Mach. Learn. Res.*, 10:1755–1758, December 2009.

[Majercik, 2006] Stephen M. Majercik. APPSSAT: Approximate probabilistic planning using stochastic satisfiability. *International Journal of Approximate Reasoning*, 2006.

[Mundhenk *et al.*, 2000] M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM*, 47(4):681–720, 2000.

[Ng and Jordan, 2000] Andrew Y. Ng and Michael I. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 406–415. Morgan Kaufmann Publishers Inc., 2000.

[Palacios and Geffner, 2009] Hector Palacios and Hector Geffner. Compiling uncertainty away in conformant planning problems with bounded width. *J. Artif. Int. Res.*, 35:623–675, August 2009.

[Rintanen, 2001] Jussi Rintanen. Complexity of probabilistic planning under average rewards. In *IJCAI*, pages 503–508, 2001.

[Sridharan *et al.*, 2010] Mohan Sridharan, Jeremy Wyatt, and Richard Dearden. Planning to see: Hierarchical POMDPs for planning visual actions on a robot. *Artificial Intelligence Journal*, 174(11):704–725, 2010.

[Yoon *et al.*, 2007] Sung-Wook Yoon, Alan Fern, and Robert Givan. FF-replan: A baseline for probabilistic planning. In Mark S. Boddy, Maria Fox, and Sylvie Thibaux, editors, *ICAPS*, pages 352–. AAAI, 2007.

[Yoon *et al.*, 2008] Sungwook Yoon, Alan Fern, Robert Givan, and Subbarao Kambhampati. Probabilistic planning via determinization in hindsight. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, pages 1010–1016. AAAI Press, 2008.

[Younes and Littman, 2004] H. L. S. Younes and M. Littman. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-167, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2004.

[Younes *et al.*, 2005] Håkan L. S. Younes, Michael L. Littman, David Weissman, and John Asmuth. The first probabilistic track of the international planning competition. *J. Artif. Intell. Res. (JAIR)*, 24:851–887, 2005.