Planning in Belief Space with a Labelled Uncertainty Graph

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

David E. Smith
NASA Ames Research Center
Comp. Sciences Division, MS 269-2
Moffett Field, CA 94035-1000
de2smith@email.arc.nasa.gov

Abstract

Planning in belief space with a Labelled Uncertainty Graph, LUG, is an approach that uses a very compact planning graph to guide search in the space of belief states to construct conformant and contingent plans. A conformant plan is a plan that transitions (without sensing) all possible initial states through possibly non-deterministic actions to a goal state. A contingent plan adds the ability to observe state variables and branch execution.

The LUG provides heuristics to guide a regression belief space planner, C*AltAlt*, and a progression belief space planner, $PBSP$. The key innovation of the LUG is to compactly represent the optimistic projection of several states (those in a belief state) within a single planning graph. Labels on the actions and literals denote the state projections that include them. We show the improvements of using a LUG by comparing it with a multiple planning graphs (one for each possible world) approach within C*AltAlt*. We also compare the approach to other conformant planners. Lastly, we outline an algorithm that can adapt the approach to stochastic planning.

Introduction

Planning with incomplete states is a well studied problem. It can be cast either as the computation of a policy for (PO)MDPs [Boutilier *et al.*, 1999] or as a directed search in the space of belief states [Bonet and Geffner, 2000]. The two views are coming closer in recent years [Bonet and Geffner, 2003]. One issue in the belief space search view is the need for effective heuristics to guide the search. We recently [Bryce and Kambhampati, 2004] showed that planning graph based heuristics, adapted from classical planning [Nguyen *et al.*, 2002], can be used to guide conformant planners. The heuristics were extracted from multiple planning graphs (MG) and proved to be useful in guiding C*AltAlt*, a conformant (non-observational) regression planner. However, the number of planning graphs needed was exponential in the number of uncertain initial state literals. Hence, the approach did not scale well as the number of possible initial states grew. The impediments involved either running out of space to build planning graphs or spending too much time computing heuristics across the multiple planning graphs.

In this paper, we describe a significant improvement that addresses these limitations. The idea is to condense the previously used multiple planning graphs to a single planning graph, a Labelled Uncertainty Graph (LUG). Loosely speaking, this single graph unions the causal support information present in the multiple graphs and pushes the disjunction, describing sets of possible worlds, into "labels". The graph elements are the same as those present in multiple graphs, but represented only once. For instance an action that was present in all of the multiple planning graphs would be present only once in the LUG and labelled to indicate that it is applicable in a projection from each possible world.

We have evaluated the LUG's effectiveness in both conformant and contingent planning. In conformant planning, we did regression and progression search. In contingent planning, we only evaluated progression search. For the evaluation we implemented two planners: C*AltAlt*, a conformant regression planner, and $PBSP$ a conformant and contingent progression planner.[1] Our results show that the LUG significantly improves C*AltAlt*'s performance in terms of heuristic computation time and scalability. Moreover, C*AltAlt* using the LUG is competitive with other state of the art conformant planners in CPU time, and provides high quality plans.

Although our focus to date has been on non-deterministic planning, we believe that these techniques will help in stochastic planning too. In particular, we will outline an approach that uses our techniques to solve non-deterministic relaxations of stochastic problems and uses the solution as a basis for stochastic plans.

The rest of the paper is organized as follows. First we briefly describe the search formulations for actions with conditional and non-deterministic effects. Then we describe how to build the LUG, how to use it for heuristic guidance, and show the LUG's improvement in C*AltAlt* over MG and compare with MBP [Bertoli *et al.*, 2001a], HSCP [Bertoli *et al.*, 2001b], KACMBP [Bertoli and Cimatti, 2002], GPT [Bonet and Geffner, 2000], CGP [Smith and Weld, 1998], and SGP [Weld *et al.*, 1998] on several domains for conformant planning. Finally, we discuss how this approach to non-deterministic planning could be extended to stochastic planning.

[1]For lack of space, we only describe the C*AltAlt* conformant planner here, and refer the reader to (http://verde.eas.asu.edu/belief-search/) for an extended version of the paper [Bryce *et al.*, 2004] that describes the $PBSP$ contingent planner in more detail.

Search

Our planning formulation uses regression search to find conformant plans. Search is in the space of belief states using actions with conditional and non-deterministic effects. The planning problem is $P = (D, BS_I, BS_G)$ and the domain is $D = (L, S, A)$, where L is the set of all literals l, S is the set of all states, and A is the set of actions. BS_I and BS_G are the respective initial and goal belief states.

Belief State Representation: As discussed in [Bonet and Geffner, 2000], conformant and contingent planning can be seen as a search in the space of belief states. Given a world state represented in terms of a set of boolean state variables, a belief state BS_i is an arbitrary propositional formula, representing a set of states (also referred to as possible worlds). Belief states can be seen as a uniform probability distribution over states. We consider two special canonical representations of BS_i – clausal representation $\kappa(BS_i)$, which is in CNF (clauses C over literals L), and constituent representation, $\xi(BS_i)$, which is in DNF (possible worlds S over literals L).

Using the bomb and toilet with clogging problem, BTC,[2] [McDermott, 1987] as a running example for this paper, the belief state representation of BTC's initial condition, in clausal representation, is: $\kappa(BS_I) = arm \wedge \neg clog \wedge (inP1 \vee inP2) \wedge (\neg inP1 \vee \neg inP2)$, or in constituent representation: $\xi(BS_I) = (arm \wedge \neg clog \wedge inP1 \wedge \neg inP2) \vee (arm \wedge \neg clog \wedge \neg inP1 \wedge inP2)$. BTC's goal state is partial, its clausal representation is: $\kappa(BS_G) = \neg arm$, and its constituent representation is: $\xi(BS_G) = \neg arm$.

Action Representation: A causative action a, of the action set A, is described in terms of (i) an executability precondition ρ_e, and (ii) several conditional effects φ_i of the form $(\rho_i \implies \varepsilon_i)$, where the antecedent ρ_i and the consequent ε_i are, in general, formulas. The executability precondition ρ_e, also a formula, of the action must hold for the action to be executable. We define φ_0 as the unconditional effect of an action where by convention $\rho_0 = \top$ and ε_0 is given.

As an example, the actions for BTC are:

$$\begin{aligned} DunkP1 : \{&\rho_e : \neg clog, \rho_0 : \top \implies \varepsilon_0 : clog,\\ &\rho_1 : inP1 \implies \varepsilon_1 : \neg arm\}\\ DunkP2 : \{&\rho_e : \neg clog, \rho_0 : \top \implies \varepsilon_0 : clog,\\ &\rho_1 : inP2 \implies \varepsilon_1 : \neg arm\}\\ Flush : \{&\rho_e : \top, \rho_0 : \top \implies \varepsilon_0 : \neg clog\} \end{aligned}$$

Regression: We start weighted A* regression search in the C*AltAlt* planner with the goal belief state and regress it non-deterministically over all relevant actions. An action is relevant for regressing a belief state if (i) its unconditional effect is consistent with the belief state and (ii) at least one effect consequent entails a literal that is present in the belief state.

Following Pednault [1987], regressing a belief state BS_i over an action a, with conditional effects, involves finding the executability, causation, and preservation formulas of BS_i w.r.t. a. We define regression in terms of clausal representation, but it can be generalized for arbitrary formulas. The regression of a belief state is a conjunction of the regression of clauses in $\kappa(BS_i)$. Formally, the result $BS_{i'}$ of regressing the belief state BS_i over the action a is defined as:[3]

$$BS_{i'} = Regress(BS_i, a) = \Pi_a \wedge \left(\bigwedge_{C \in \kappa(BS_i)} \bigvee_{l \in C} \left(\Sigma_a^l \wedge IP_a^l \right) \right) \quad (1)$$

Executability formula (Π_a) is the executability precondition ρ_e of a. This is what must hold in $BS_{i'}$ for a to have been applicable.

Causation formula (Σ_a^l) for a literal l w.r.t all effects φ_j of an action a is defined as the weakest formula that must hold before a such that l holds in BS_i. Formally Σ_a^l is defined as:

$$\Sigma_a^l = l \vee \bigvee_{j : \varepsilon_j \models l, j \neq 0} \rho_j \quad (2)$$

Preservation formula (IP_a^l) of a literal l w.r.t. all effects φ_j of action a is defined as the weakest formula that must be true before a such that l is not violated by the effect ε_j. Formally IP_a^l is defined as:

$$IP_a^l = \bigwedge_{j : \varepsilon_j \models \neg l, j \neq 0} \neg \rho_j \quad (3)$$

For example, in the BTC problem we have $BS_1 = Regress(BS_G, DunkP1) = \neg clog \wedge (\neg arm \vee inP1)$. The first clause is the executability formula and the second clause is the causation formula for $DunkP1$'s conditional effect and $\neg arm$. Regressing BS_1 with $Flush$ gives $BS_2 = (\neg arm \vee inP1)$ because the executability precondition of $Flush$ is $\top$, the causation formula is $\top \vee \neg clog = \top$ and $(\neg arm \vee inP1)$ comes through persistence. Finally, $BS_3 = Regress(BS_5, DunkP2) = \neg clog \wedge (\neg arm \vee inP1 \vee inP2)$.

Termination: Regression terminates when search node expansion generates a belief state BS_i which is entailed by the initial belief state BS_I. The plan is the sequence of actions regressed from BS_G to obtain BS_i.

From our example, we terminate at BS_3 because $BS_I \models BS_3$. The plan is $DunkP2, Flush, DunkP1$.

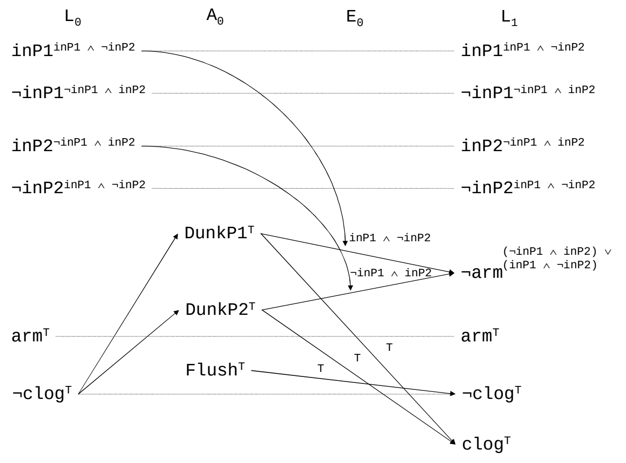
LUG Heuristic Guidance

Before search starts, we construct a data-structure (previously MG, now a LUG) to provide heuristics for search guidance. The data-structure approximates a progression tree, starting at BS_I. During search, each new belief state is evaluated with respect to the data-structure to give it a heuristic value – indicating its "goodness" or (i.e. relative number of actions to reach the initial belief state BS_I). Planner performance relies both on the time it takes to construct this data-structure and the structure's ability to give good heuristics (i.e. guide search through less of the search space).

The LUG is a generalization of the MG structure used in C*AltAlt* [Bryce and Kambhampati, 2004] to estimate the

[2]Bomb in the Toilet with Clogging.For the uninitiated, here are the arcana of the Bomb in the Toilet family of problems: Bomb in the Toilet (BT)–the problem includes two packages, one of which contains a bomb, and a toilet. The goal is to disarm the bomb and the only allowable actions are dunking a package in the toilet. The variation "bomb in the toilet with clogging" or BTC says that the toilet will clog unless it is "flushed" after each "dunking" action.

[3]Note that $BS_{i'}$ may not be in clausal form after regression (especially when an action has multiple conditional effects).



L_0
A_0
E_0
L_1
$inP1^{inP1 \wedge \neg inP2}$
$\neg inP1^{\neg inP1 \wedge inP2}$
$inP2^{\neg inP1 \wedge inP2}$
$\neg inP2^{inP1 \wedge \neg inP2}$
$DunkP1^{\top}$
$DunkP2^{\top}$
$Flush^{\top}$
$arm^{\top}$
$\neg clog^{\top}$
$inP1 \wedge \neg inP2$
$\neg inP1 \wedge inP2$
$\top$
$\neg arm^{(\neg inP1 \wedge inP2) \vee (inP1 \wedge \neg inP2)}$
$clog^{\top}$

Figure 1: LUG for BTC. The labels are the superscripts.

reachability of belief states. MG has the advantage of informative heuristics, but the disadvantages of computing redundant support information and looking at every graph to compute heuristics. A better approach condenses the multiple planning graphs to a single planning graph that retains the multiple possible world causal structure. Loosely speaking, this single graph unions the causal support information present in the multiple graphs and pushes the disjunction, describing sets of possible worlds, into "labels" (ℓ). The graph elements are the same as those present in multiple graphs, but represented only once. For instance an action that was present in all of the multiple planning graphs would be present only once in the LUG and labelled to indicate that it is applicable in a projection from each possible world of BS_I.

The LUG adds labels to graph elements to symbolically represent which projections through the graph relate to which constituents of BS_I. The labelled elements are each action a, each conditional and unconditional effect relation φ of an action, each literal l, and each mutex relation of the graph. In general, a label is an arbitrary propositional formula describing a set of possible worlds for which a graph element is reachable. The way the LUG is constructed is to label the single set of initial literals with the possible worlds where they hold and propagate these labels through actions as the graph is built. Construction ends when the goal belief state can be satisfied by literals present in a graph level and the literals are labelled to indicate that the goal belief state is supported by all possible worlds, i.e. the goal is fully-supported. Note, we are constructing the LUG in terms of literals, whereas the search is in terms of formulas. By using literals rather than formulas in the graph, we need labels to preserve the disjunction. Later, we describe how to determine if formulas are possibly supported by examining the labels of the formula's literals at levels in the LUG.

The LUG is based on IPP's [Koehler, 1999] planning graph, where there are three layers in a planning graph step: the action layer, effect layer, and literal layer. The extensions are to (1) keep sets of labelled action $\mathcal{A}$, effect relation $\mathcal{E}$, and literal $\mathcal{L}$ layers, and (2) keep sets of labelled binary mutexes for actions $\hat{\mathcal{A}}$, effect relations $\hat{\mathcal{E}}$, and literals $\hat{\mathcal{L}}$.[4] In the following subsections, we define the terms fully-supported and supported to aid in the discussion of the label propagation.

Label Propagation Recall that a label is a formula describing a set of possible worlds from which a labelled element is reachable. Labels are represented as arbitrary propositional formulas and efficient propagation of labels is handled using BDDs. The propagation of labels is based on the intuition that (i) actions and effects are applicable in the possible worlds specified by the conjunction of the precondition formula's labels and (ii) a literal is supported in possible worlds specified by the disjunction of labels of effect relations that support the literal.

We do not propagate labels to account for the non-deterministic outcomes of actions because there is no guarantee that the non-deterministic actions will be used to reach the goals, but it a requirement that each possible world needs to reach the goals. All effects, including non-deterministic effects, get labels to signify the possible worlds where they are supported, but possible worlds created by non-deterministic effects are not given labels. Within the LUG (and not in search), non-deterministic effects are treated as a conjunctive set of literals, where each literal is a literal appearing in some outcome of the non-deterministic effect. When an action with a non-deterministic effect is added to a level of the LUG, the set of literals is added to the subsequent literal layer.

During the LUG construction, a common operation is to determine if a formula is fully-supported (i.e. *all* possible world projections can reach a belief state that entails the formula), or supported (i.e. *there exists* a possible world whose projection can reach a belief state that entails the formula).[5]

A formula f is **fully-supported** (FSp(f, k)) at level k when for all possible worlds S entailing BS_I, f is supported by S.[6]

A formula f is **supported** (Sp(f,k,S)) at level k by a possible world S entailing BS_I when the labels of the literals in f indicate support by S. The labels of literals indicate support by S when the formula's literals are substituted with their labels and S entails the substituted formula. To ease the definition, we consider a canonical form for f, namely a CNF, $\mathcal{C}$, that is supported when:

$$S \models \left(\bigwedge_{C \in \mathcal{C}} \bigvee_{l \in C} \ell_k(l) \right) \qquad (4)$$

Here $\ell_k(l)$ is the label of the literal l at level k . Note that full-support is checked for all possible worlds at once by replacing S with BS_I.

[4]For a discussion of mutexes within the LUG, we refer the reader to (http://verde.eas.asu.edu/belief-search/) for an extended version of the paper [Bryce *et al.*, 2004] that describes them in more detail. The mutexes used in the evaluation of the LUG are equivalent to those used in the MG structures (i.e. the mutexes are computed for only same possible worlds).

[5]Note that the LUG is an approximation to the belief space projection from the belief state BS_I, so when we refer to something as supported we mean *possibly* supported.

[6]The notion of fully-supported is a generalization of the level heuristic for classical planning [Nguyen *et al.*, 2002]. It is also similar to the max heuristic used in GPT [Bonet and Geffner, 2000].

We now describe label propagation, first by showing how to construct the initial literal layer $\mathcal{L}_0$ of the graph, and then showing how a graph level $\{\mathcal{L}_k, \mathcal{A}_k, \mathcal{E}_k\}$ is built.

$\mathcal{L}_0 \leftarrow$ **insertInitialLiterals**(BS_I)**:** The initial literal layer $\mathcal{L}_0$ contains all literals l in the possible worlds of the belief state BS_I. Each literal l is labelled $\ell_0(l)$ to indicate the set of possible worlds where it holds. The label of literal l is found by the conjunction of l with the formula for BS_I.

BTC has the initial layer shown in Figure 1. The known literals (arm and $\neg clog$) are labelled $\top$[7], and the unknown literals ($inP1$ and $inP2$) are labelled to indicate the possible worlds that contain them. The labels in Figure 1 are the most general formulas to express the possible worlds (to conserve space), but in practice the labels involve all literals of the problem. For instance, the label for $inP1$ is denoted in Figure 1 as $inP1 \wedge \neg inP2$, but represents the possible world $inP1 \wedge \neg inP2 \wedge arm \wedge \neg clog$. The label for $inP1$ is found by taking the conjunction of $inP1$ with BS_I, which is $inP1 \wedge arm \wedge \neg clog \wedge (inP1 \vee inP2) \wedge (\neg inP1 \vee \neg inP2)$ that reduces to $inP1 \wedge \neg inP2 \wedge arm \wedge \neg clog$. The label for arm is found similarly by taking its conjunction with BS_I, which reduces to BS_I.

$\mathcal{A}_k \leftarrow$ **insertActions**$(\mathcal{L}_k)$**:** Once the literal layer $\mathcal{L}_k$ is computed, we compute the labelled action layer $\mathcal{A}_k$. $\mathcal{A}_k$ is defined as all applicable actions from the action set A, plus all literal persistence $\Diamond l$.[8] An action's executability precondition must be supported for some possible world at level k for the action to be applicable. If applicable, the action's label at level k, using a CNF for the formula of ρ_e, is:

$$\ell_k(a) = \bigwedge_{C \in \rho_e} \bigvee_{l \in C} \ell_k(l) \tag{5}$$

The labels of all the actions in BTC, Figure 1, are $\top$ since the enabling preconditions for all actions are either empty or labelled $\top$.

$\mathcal{E}_k \leftarrow$ **insertEffects**$(\mathcal{L}_k, \mathcal{A}_k)$ **:** The labelled effect relations $\mathcal{E}_k$ depend both on the literal layer $\mathcal{L}_k$ and action layer $\mathcal{A}_k$. An effect relation is applicable when the associated action is applicable and the antecedent of the effect is supported. The label of the effect is the conjunction of the label of the associated action with the label of the formula of the effect's antecedent. The label of an effect φ_i at level k, using a CNF for the formula of ρ_i, is:

$$\ell_k(\varphi_i) = \ell_k(a) \wedge \left(\bigwedge_{C \in \rho_i} \bigvee_{l \in C} \ell_k(l) \right) \tag{6}$$

The conditional effects of the $Dunk$ actions in BTC, Figure 1, have labels to indicate the possible worlds for which they will give $\neg arm$ because their antecedents do not hold in all possible worlds. The other effects of actions have labels $\top$ because they are unconditional and the associated action has label $\top$.

$\mathcal{L}_k \leftarrow$ **insertLiterals**$(\mathcal{E}_{k-1}), k > 0$**:** The literal layer at k is the set of labelled literals that are added by consequents of effects in $\mathcal{E}_{k-1}$. A literal is added to the literal layer if it is present in the formula of a consequent of an effect in the previous layer. The label of a literal, $\ell_k(l)$, is the disjunction of the labels of each effect that has the literal in its consequent's formula, using a CNF for the formula of ε_i, it is:

$$\ell_k(l) = \bigvee_{\substack{l \in C \\ C \in \varepsilon_i \\ \varphi_i \in \mathcal{E}_{k-1}}} \ell_{k-1}(\varphi_i) \tag{7}$$

The labels of the literals for level 1 of BTC, Figure 1, indicate that $\neg arm$ is fully-supported because its label is entailed by BS_I. Construction can stop here.

We change the usual notion of when planning graph expansion can cease. LUG construction stops when the formula for the goal belief state BS_G is fully-supported.

Heuristic Computation

The relaxed plan heuristic we extract from the LUG is similar to the h^{MG}_{RPU} heuristic [Bryce *et al.*, 2004]. The h^{MG}_{RPU} heuristic extracts a relaxed plan from each of the multiple planning graphs (one for each possible world) and unions the set of actions chosen at each level of each of the planning graphs, then takes the number of unioned actions as the heuristic value. The LUG relaxed plan heuristic h^{LUG}_{RP} is similar in that it counts actions that are applicable in multiple worlds only once and accounts for actions that are used in subsets of the possible worlds. The advantage is that we find these actions by looking at only one planning graph.

The relaxed plan is representative of a belief space plan because at each level of the planning graph we ensure that the chosen actions will support the subgoals from all possible worlds. In many cases the relaxed plan can use one action to support subgoals from several possible worlds. This is useful in guiding the search towards plans with lower overall plan length and higher world overlap in achieving the goal from all possible worlds. The relaxed plans extracted from the LUG assume independence between actions because mutex relations are ignored.

We extract the relaxed plan for a belief state BS_i by starting at level k, where FSp(BS_i, k) first holds, by supporting the formula for BS_i with sets of actions at $k-1$ (forming $step_{k-1}$), then support the conjunction of those actions' preconditions at the next lower level of the LUG, and so on. When supporting a formula, it is treated as a CNF. This means that for each clause we find a set of effect relations where (i) each effect gives at least one of the literals in the clause and (ii) the disjunction of labels of literals at k in the clause entails the disjunction of chosen effect relation labels at level $k-1$. This means the possible worlds that can reach the clause are covered by the set of chosen effects. For example, $\{\varphi_1, ..., \varphi_i, ...\}$ support a clause with literals $\{l_1, ..., l_j, ...\}$ if:

$$\left(\bigvee_j \ell_k(l_j) \right) \models \left(\bigvee_i \ell_{k-1}(\varphi_i) \right) \tag{8}$$

This is similar to how we find relaxed plans on normal planning graphs, but the differences are in how we determine that a formula is supported by a set of actions and that we prefer supporting with literal persistence before actions. Formally, the value of the relaxed plan heuristic for the LUG is:

$$h^{LUG}_{RP}(BS_i) = \sum_{j=0}^{k-1} \mid step_j \mid \tag{9}$$

[7]As an efficiency measure, without loss of generality, we replace the label of every element x with $\top$ if $BS_I \models \ell_k(x)$.

[8]Persistence for a literal l, denoted by $\Diamond l$, is represented as an action where $\rho_e = \varepsilon_0 = l$.

Problem	C*AltAlt* h^{MG}_{RPU}	C*AltAlt* h^{LUG}_{RP}	MBP	KACMBP	HSCP	GPT	CGP	SGP
Rovers1	185/5	16070/5	230/5	9293/5	-	3139/5	70/5	70/5
2	29285/9	10457/8	141/8	9289/15	-	4365/8	180/8	30/8
3	2244/11	10828/10	484/10	9293/16	-	5842/10	460/10	1750/10
4	3285/15	15279/13	3252/13	9371/18	-	7393/13	1860/13	-
5	-	64870/29	-	39773/40	-	399525/20	-	-
6	-	221051/25	727/32	-	-	-	-	-
Logistics1	1109/9	907/9	37/9	127/12	352/9	916/9	60/6	70/6
2	69818/19	2862/15	486/24	451/19	-	1297/15	290/6	510/6
3	70882/14	10810/15	408/14	1578/18	-	1711/11	400/8	4620/8
4	-	24862/19	2881/27	8865/22	-	9828/18	1170/8	447470/8
5	-	54726/34	-	226986/42	-	543865/28	-	-
BT2	21/2	16/2	6/2	10/2	8/2	487/2	20/1	0/1
20	2299/20	552/20	-	84/20	23/20	472174/20	3200/1	290/1
40	44741/40	7543/40	-	533/40	80/40	-	24630/1	3320/1
60	-	35983/60	-	2123/60	340/60	-	87970/1	83494/1
80	-	157655/80	-	-	-	-	-	-
BTC2	23/3	16/3	8/3	18/3	2/3	465/3	0/3	0/3
20	2652/39	651/39	98/39	211/39	98/39	-	-	-
40	51859/79	8009/79	615/79	1498/79	674/79	-	-	-
60	-	38393/119	2223/119	5506/119	5100/119	-	-	-

Figure 2: Results for C*AltAlt* using h^{MG}_{RPU} and $h^{LUG(D-S)}_{RP}$, MBP, KACMBP, HSCP, GPT, CGP, and SGP for conformant Rovers, Logistics, BT, and BTC. The data is Total Time / # Plan Steps, "-" indicates no solution.

From our example, $BS_I \models \ell_1(\neg arm)$, so FSp($BS_G$, 1) holds and $k = 1$. A relaxed plan to support BS_G is $DunkP1, DunkP2, Flush$. The first clause, $\neg arm$, is fully-supported through $DunkP1$ and $DunkP2$ because the disjunction of the labels of their conditional effects at level 0 entails the label of $\neg arm$ at level 1. Similarly, $\neg clog$ is fully-supported through $Flush$ because the label of $Flush$'s effect at level 0 entails the label of $\neg clog$ at level 1. Thus $h^{LUG}_{RP}(BS_G) = 3$.

Empirical Evaluation

The LUG's implementation takes advantage of several technologies, BDDs [Brace *et al.*, 1990] for labels, a model checker [Cimatti *et al.*, 2002] for checking entailment, and a planning graph [Koehler, 1999]. C*AltAlt* is based on the HSP-r [Bonet and Geffner, 1999] regression search engine.

We ran several test problems within C*AltAlt* using the LUG to compare it with the previously used MG. We also ran the same problems for MBP [Bertoli *et al.*, 2001a], KACMBP [Bertoli and Cimatti, 2002], HSCP [Bertoli *et al.*, 2001b], GPT [Bonet and Geffner, 2000], CGP [Smith and Weld, 1998], and SGP [Weld *et al.*, 1998]. All tests were run on a 2.66GHz Pentium 4 Linux machine with a memory limit of 1GB.

In addition to the standard bomb in toilet (BT), and bomb in the toilet with clogging (BTC), we also look at conformant versions of Logistics and Rovers. The Logistics and Rovers domains are similar to domains used in the International Planning Competition [IPC, 2003], and dissimilar to domains previously considered in conformant planning research. They present new challenges in reasoning about reachability for conformant planning because there are many possible plans of different lengths. Logistics has uncertainty over initial package locations and scales by adding packages, cities, and possible initial locations. Rovers has uncertainty over availability of collectable scientific data at various locations and scales by adding possible locations to get images, rocks, and soil. [9]

Figure 2 shows the results for running several domains with C*AltAlt* using h^{MG}_{RPU} and h^{LUG}_{RP}, MBP, KACMBP, HSCP, GPT, CGP, and SGP. The LUG does much better than MG in terms of scalability in all cases because of reduced graph memory requirements and heuristic computation time. Plan length is mostly better with the h^{LUG}_{RP} compared to h^{MG}_{RPU}. C*AltAlt* with h^{LUG}_{RP} tends to scale better than the optimal approaches of GPT, CGP, and SGP, without over sacrificing plan quality, and is comparable to the heuristic approaches of MBP, KACMBP, and HSCP (usually providing better plans and out-scaling).

Related Work

Of the many approaches to conformant and contingent planning this approach is most similar to the work of CGP and SGP ([Smith and Weld, 1998], [Weld *et al.*, 1998]), GPT [Bonet and Geffner, 1999], and the MBP-related planners ([Bertoli *et al.*, 2001b], [Bertoli and Cimatti, 2002], [Bertoli *et al.*, 2001a]).

CGP and SGP, like C*AltAlt*, use several planning graphs to represent a projection from each initial state. The difference is that CGP and SGP generalize GraphPlan [Blum and Furst, 1995] to perform a search on the multiple planning graphs, whereas we are using the planning graphs to provide heuristics for belief space search.

GPT, one of the first heuristic conformant and contingent planners, uses dynamic programming to compute reachability heuristics for progression search in belief space. The largest differences between this approach and GPT is that GPT uses an admissible max-type heuristic, whereas ours is inadmissible, and GPT explicitly represents the belief space where we use formulas.

The MBP-family of planners performs search in belief space through regression or progression guided by ad-hoc combinations of cardinality and reachability heuristics, as in KACMBP. The biggest difference between their approach and ours are the use of structured heuristics to improve plan quality.

Extension to Stochastic Planning

Given the positive results presented in this paper, we foresee use of the LUG in stochastic planning. There are two possible avenues: (i) extend planning graph heuristics to direct stochastic planning search, and (ii) solve a non-deterministic

[9]The problem and domain files can be found at http://verde.eas.asu.edu/belief-search/.

relaxation of a stochastic planning problem and use it as a seed stochastic plan. In the following discussion we describe the second approach.[10]

The basic approach is to (1) relax a stochastic planning problem to non-deterministic planning problem, (2) solve the non-deterministic planning problem with techniques presented herein, and (3) convert the non-deterministic plan to a stochastic plan and perform a local search on its structure to increase probability of goal satisfaction. The first step involves relaxing a stochastic plan to a non-deterministic plan entails ignoring probabilities and assuming uniform distributions. Some stochastic problems may not have any plan that will succeed in the non-deterministic relaxation; in such cases we may have to ignore some outcomes of actions or some of the initial states. The second step of solving the relaxed non-deterministic problem follows from the content of this paper and [Bryce *et al.*, 2004]. The quality of the non-deterministic plan, viewed as a stochastic plan may improve if some greediness is incorporated into the heuristics for the non-deterministic plan synthesis. One idea is to maintain probabilities for the effects of actions and the labels of possible worlds within the LUG (similar to PGraphPlan), and have relaxed plans prefer actions that have higher probability outcomes and support higher probability possible worlds. In the third step, with the plan, we can see with what probability certain actions support the conditions of other actions or goals. It may be possible to perform local search to add more actions to increase the probability of supporting conditions and goals.

The approach we outline is similar to that of Buridan [Kushmerick *et al.*, 1994] in that a seed stochastic plan is generated, and then actions to increase support for low probability conditions are added to the plan. The advantages of our version are in using planning graph heuristics and modelling non-determinism. Using planning graph heuristics allows us to tackle somewhat larger problems than Buridan, which uses no search heuristics. Relaxing the planning problem to a non-deterministic planning problem will provide a seed solution that is more faithful to the eventual solution than Buridan (which uses a relaxation to a deterministic planning problem).

Conclusion

In investigating the LUG's utility we have learned that the LUG is an effective data-structure to guide search in a conformant planner with competitive performance. The advantage of the LUG and the relaxed plan heuristic is that it helps generate high quality plans for difficult problems.

Acknowledgements This research is supported in part by the RIACS SSRP fellowship program, NASA grants NAG2-1461 and NCC-1225, and NSF grant IIS-0308139.

[10]Heuristics help in generating large non-deterministic plans; however, these heuristics may not be directly suitable for generating stochastic plans. Stochastic planning with planning graph heuristics is troublesome because planning graphs mainly provide lower bound estimates on the number of actions needed to reach states, but do not help in finding lower bound estimates for the probabilities of states. One approach that uses planning graphs for stochastic planning is PGraphPlan [Blum and Langford, 1999] where a planning graph is searched using forward dynamic programming, effectively propagating probabilities on the planning graph – a costly endeavor. In classical (deterministic) planning, GraphPlan [Blum and Furst, 1995] has been outperformed by many state-based search techniques that use planning graphs for heuristics [Hoffmann and Nebel, 2001], [Nguyen *et al.*, 2002]; the same may be true for stochastic planning, yet effective heuristics for state-based stochastic planning do not exist.

References

Piergiogio Bertoli and Alessandro Cimatti. Improving heuristics for planning as search in belief space. In *Artificial Intelligence Planning Systems*, pages 143–152, 2002.

Piergiorgio Bertoli, Alessandro Cimatti, Marco Roveri, and Paolo Traverso. Planning in nondeterministic domains under partial observability via symbolic model checking. In *IJCAI*, pages 473–478, 2001.

Piergorgio Bertoli, Alessandro Cimatti, and Marco Roveri. Heuristic search + symbolic model checking = efficient conformant planning. In IJCAI, 2001.

Avrim Blum and Merrick Furst. Fast planning through planning graph analysis. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95)*, pages 1636–1642, 1995.

Avrim Blum and John Langford. Probabilistic planning in the graphplan framework. In *ECP*, pages 319–332, 1999.

Blai Bonet and Hector Geffner. Planning as heuristic search: New results. In *ECP*, pages 360–372, 1999.

Blai Bonet and Hector Geffner. Planning with incomplete information as heuristic search in belief space. In *AIPS-2000*, pages 52–61, 2000.

Blai Bonet and Hector Geffner. Labeled rtdp: Improving the convergence of real-time dynamic programming. In ICAPS, 2003.

C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

Karl S. Brace, Richard L. Rudell, and Randal E. Bryant. Efficient implementation of a bdd package. In *Conference proceedings on 27th ACM/IEEE design automation conference*, 1990.

Daniel Bryce and Subbarao Kambhampati. Heuristic guidance measures for conformant planning. In *ICAPS'04*, June 2004.

Daniel Bryce, Subbarao Kambhampati, and David E. Smith. Planning graph heuristics for belief space search. Technical report, ASU, 2004. Forthcoming.

A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *CAV 2002*, July 2002.

Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

Volume 20, special issue on the 3rd international planning competition. *Journal of Artificial Intelligence Research*, 2003.

Jana Koehler. Handling of conditional effects and negative goals in IPP. Technical Report report00128, IBM, 17, 1999.

Nicholas Kushmerick, Steve Hanks, and Daniel Weld. An algorithm for probabilistic least-commitment planning. In *In AAAI-94*, 1994.

Drew McDermott. A critique of pure reason. *Computational Intelligence*, 3(3):151–237, 1987.

XuanLong Nguyen, Subbarao Kambhampati, and Romeo Sanchez Nigenda. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. *Artificial Intelligence*, 135(1-2):73–123, 2002.

Edwin P. D. Pednault. Synthesizing plans that contain actions with context-dependent effects. Technical Memorandum, AT&T Bell Laboratories, Murray Hill, NJ, 1987.

David E. Smith and Daniel S. Weld. Conformant graphplan. In *(AAAI-98) and (IAAI-98)*, pages 889–896, Menlo Park, July 26–30 1998. AAAI Press.

Daniel S. Weld, Corin Anderson, and David E Smith. Extending graphplan to handle uncertainty and sensing actions. In *In AAAI-98*, 1998.