Heuristic Guidance Measures for Conformant Planning

Daniel Bryce & Subbarao Kambhampati
Department of Computer Science and Engineering
Arizona State University, Tempe AZ 85287-5406
{dan.bryce,rao}@asu.edu

Abstract

Scaling conformant planning is a problem that has received much attention of late. Many planners solve the problem as a search in the space of belief states, and some heuristic guidance techniques have been developed to estimate the distance between belief states. We claim that heuristic techniques in the past involved an *ad hoc* combination of classical planning heuristics and cardinality measures. We discuss how to derive heuristics systematically, with the help of planning graphs, such that the measures reflect the reachability of relevant states within belief states. To demonstrate these ideas we show how distances between belief states can be estimated by a set of reachability heuristics. We empirically evaluate their effectiveness within a conformant regression planner named $\mathcal{C}AltAlt$.

Introduction

Ever since CGP (Smith & Weld 1998) a series of planners have been developed for tackling conformant planning problems – including GPT (Bonet & Geffner 2000), C-Plan (Castellini, Giunchiglia, & Tacchella 2001), PKSPlan (Bacchus 2002), Frag-Plan (Kurien, Nayak, & Smith 2002), HSCP (Bertoli, Cimatti, & Roveri 2001), and KACMBP (Bertoli & Cimatti 2002). Several of these planners are extensions of heuristic state search planners that search in the space of "belief states" (where a belief state is a set of states, one of which the agent "believes" it is currently in). Although heuristic search conformant planners are currently among the best, the question of what should constitute heuristic estimates for a conformant planner has not yet been adequately investigated.

Intuitively, it can be argued that the heuristic merit of a belief state depends on at least two factors–the size of the belief state (i.e., the uncertainty in the belief state), and the distance of the individual states in the belief state from the goal (belief) state. The question of course is how to compute these measures and which are most effective. We argue that existing planners do not adequately address this question, concentrating instead exclusively on one measure or another. For example, in regression search, HSCP picks belief states with high cardinality for expansion, with the intuition that the larger the belief state, the larger the uncertainty. KACMBP adds to HSCP's heuristic by also measuring reachability of literals. In contrast to these methods, GPT only considers only admissible reachability information.

A related issue in evaluating the effectiveness of heuristics is the many architectural differences between planners. It is quite hard to pinpoint the global effect of the assumptions underlying their heuristics on performance. For example, GPT is outperformed by HSCP and KACMBP–but it is questionable as to whether the credit for this efficiency is attributable to the differences in heuristics, or differences in search engines (HSCP and KACMBP use a BDD-based search).

Our interest in this paper is to systematically evaluate a spectrum of approaches for computing heuristics for conformant planning. We will start by discussing, in general terms, what the heuristic estimates *should be* evaluating in conformant planning. This involves formalizing the notions of distances between belief states. Once we figure out what information the heuristics should be computing, we will then turn our attention to efficient ways of computing that information. We will show that planning graph structures continue to provide a good substrate for the heuristic computation. To improve the informedness of the heuristics, we need to track multiple planning graphs, each corresponding to one of the possible initial states. We will describe how several powerful planning graph-based heuristics from classical planning, including "level" and "relaxed plan" (Nguyen, Kambhampati, & Nigenda 2002) can be generalized to the case of multiple planning graphs. Finally, we will evaluate the relative advantages of these heuristics in a normalized setting. For this, we use C*AltAlt*, a regression search planner that searches in the space of belief states. Our results show that relaxed plan heuristics derived from multiple planning graphs are often the best.

Although our main interest in this paper is to evaluate the relative advantages of a spectrum of conformant planning heuristics in a normalized setting, we also compare the best heuristics from this work to existing conformant planners. Our empirical studies show that planning graph based heuristics provide accurate guidance compared to cardinality heuristics as well as the reachability heuristic used by GPT, and are competitive with BDD-based conformant planners such as HSCP or KACMBP and GraphPlan-based ones such as CGP.

The rest of this paper is organized as follows. We present our work by first explaining the state and action representation used within $\mathcal{C}AltAlt$, then discuss appropriate heuristic measures for conformant planning, followed by the set of heuristics used within $\mathcal{C}AltAlt$ for search control, followed by empirical evaluation, related work, and concluding remarks.

State and Action Representation

Our conformant planning formulation uses A* regression search in the space of belief states over 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 & Geffner 2000), conformant planning can be seen as a search in the space of belief states. Given a world represented in terms of a set of boolean state variables, a belief state BS_i is an arbitrary propositional formula. We consider two special canonical representations of BS_i – clausal representation $\kappa(BS_i)$, which is in CNF, and constituent representation, $\xi(BS_i)$, which is in DNF.

Since we're dealing with partial regression states, BS_i may not explicitly represent all states in a belief state, so we define $\hat{\xi}(BS_i)$ as the *complete* set of states represented by BS_i.

Using the bomb and toilet with clogging problem, BTC,[1] (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: $\xi(BS_I) = (arm \wedge \neg clog \wedge inP1 \wedge \neg inP2) \vee (arm \wedge \neg clog \wedge inP1 \wedge \neg inP2)$. BTC's goal state is partial, its clausal representation is: $\kappa(BS_G) = \neg arm$, and its constituent representation is: $\xi(BS_G) = \neg arm$. However, its complete set of states: $\hat{\xi}(BS_G) = (\neg arm \wedge clog \wedge inP1 \wedge \neg inP2) \vee (\neg arm \wedge clog \wedge \neg inP1 \wedge inP2) \vee (\neg arm \wedge \neg clog \wedge inP1 \wedge \neg inP2) \vee (\neg arm \wedge \neg clog \wedge \neg inP1 \wedge inP2)$.

[1]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.

Action Representation: An action a, of the action set A, is described in terms of (1) an executability precondition ρ_e, (2) an unconditional effect φ_0, and (3) several conditional effects φ of the form ($\rho \implies \varepsilon$), where the antecedent ρ and the consequent ε 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 $\rho_0 = \top$ 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 pose conformant planning by regression as a search in the space of belief states, starting with the goal state and regressing it non-deterministically over all relevant actions. An action is relevant for regressing a belief state if (1) its unconditional effect is not inconsistent with the belief state and (2) at least one effect consequent gives 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:[2]

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

where

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 in the state before a such that l holds in BS_i. Formally Σ_a^l is defined as:

$$\left\{ l \vee \bigvee_j \rho_j \;\middle|\; \varepsilon_j \models l \right\} \quad (2)$$

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.

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

Formally IP^l_a is defined as:

$$\left\{ \bigwedge_j \neg\rho_j \;\middle|\; \varepsilon_j \models \neg l \right\} \tag{3}$$

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, $BS_3 = regress(BS_2, DunkP2) = \neg clog \wedge (\neg arm \vee inP1 \vee inP2)$.

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 $DunkP2, Flush, DunkP1$.

Belief State Distance Estimation

We will start by discussing what measures are worth estimating for providing heuristic guidance for conformant planning through regression. Consider the example in Figure 1; there are two belief states BS_1 and BS_2 that we are trying to assign heuristic measures for the difficulty of reaching the initial belief state BS_I. We would like to estimate D_1 and D_2, the actual lengths of conformant plans from BS_I to BS_1 and BS_2, respectively. The arcs on BS_2 labelled χ_1 and χ_2 are showing how state distance measures are combined.

There are several factors to consider and leverage in making this estimation of D_1 and D_2:

1: $\hat{\xi}(BS_i)$, the set of states in the belief state.

2: Reachability measures between pairs of individual states, d_{ij-k}, where each pair is a state S_k from BS_I and S_j from BS_i, as well as χ_1 and χ_2, the combination techniques for the distances of individual states to obtain d_i, a distance estimate to D_i.

3: The *overlap* of independent plans that reach the relevant states of BS_i from states in BS_I.

The cardinality of a belief state may be used as a cheap heuristic that assumes, in regression, that a larger belief state has more probability of containing the states of the initial belief state. However, this can be misleading because even though a belief state is large, we may not be able to extend it to include the initial states, during regression.

The reachability measures of pairs of states (d_{ij-k}) or pairs of belief states and states (d_{i-k}) also reflect how difficult a conformant plan will be to construct. These d_{ij-k} and d_{i-k} measures can be handled as either numbers estimating the plan length or sets of actions estimating a plan. Also important is how to combine the d_{ij-k} and d_{i-k} measures to ultimately get the estimate d_i. We define two combinations: χ_1, which uses the d_{ij-k}'s to get the d_{i-k} measures, and χ_2, which combines the d_{i-k} measures to get d_i. The applicable operations allowable in χ_1 and χ_2 for numerical estimates are minimum, maximum, and average; and for estimated sets of actions we can take the minimum cardinality set, maximum cardinality set, or the union of sets. Note that the sets of actions can be turned into numerical estimates by taking the cardinalities of the sets; this necessarily happens before we get a final number for d_i.

An important point to note is that in regression not all of the states $S_j \in BS_i$ need to be costed with respect to each of the initial states, only the min-cost reachable S_j for each $S_k \in BS_I$. Whereas, in progression, if there is a single goal state, then each of the states in the current belief state must have finite distance to the goal to be useful. However, the same argument for regression holds in progression when there are multiple goal states; we only care that each of the states in the current belief state has finite distance to one of the goal states. So we would like to take the max of the distances from the min-cost states to one of the goal states because some of the states in the current state may not be able to reach all of the goal states.

Furthermore, of the reachability measures for $S_j \in \hat{\xi}(BS_i)$ there can be much redundancy because the same actions may be used in many of the individual plans that map the initial states BS_I into BS_i, hence plans have high overlap. As we will show, keeping sets of actions instead of numerical estimates for the d measures can allow us to reason about overlap. Planning graphs can aid in finding the sets of actions that improve the measure of individual plan overlap.

Heuristics

This section provides three sets of heuristics that estimate these distance computations, alluded to in the previous section.

To illustrate the computation of each heuristic, we use an example from BTC called $CBTC$,[3] where a courteous package dunker has to disarm the bomb and leave the toilet unclogged. This problem is used because the goal state has two conjuncts, allowing better illustration of heuristic computation that combines the costs of individual subgoals. The initial belief state in clausal representation is $arm \wedge \neg clog \wedge (inP1 \vee inP2) \wedge (\neg inP1 \vee \neg inP2)$, and the goal is $\neg clog \wedge \neg arm$. The optimal action sequences to reach BS_G from BS_I is: $DunkP1, Flush, DunkP2, Flush$, or $DunkP2, Flush, DunkP1, Flush$, thus the optimal heuristic estimate is $h^*(BS_G) = 4$ because in either plan there are four actions.

[3]Courteous BTC.

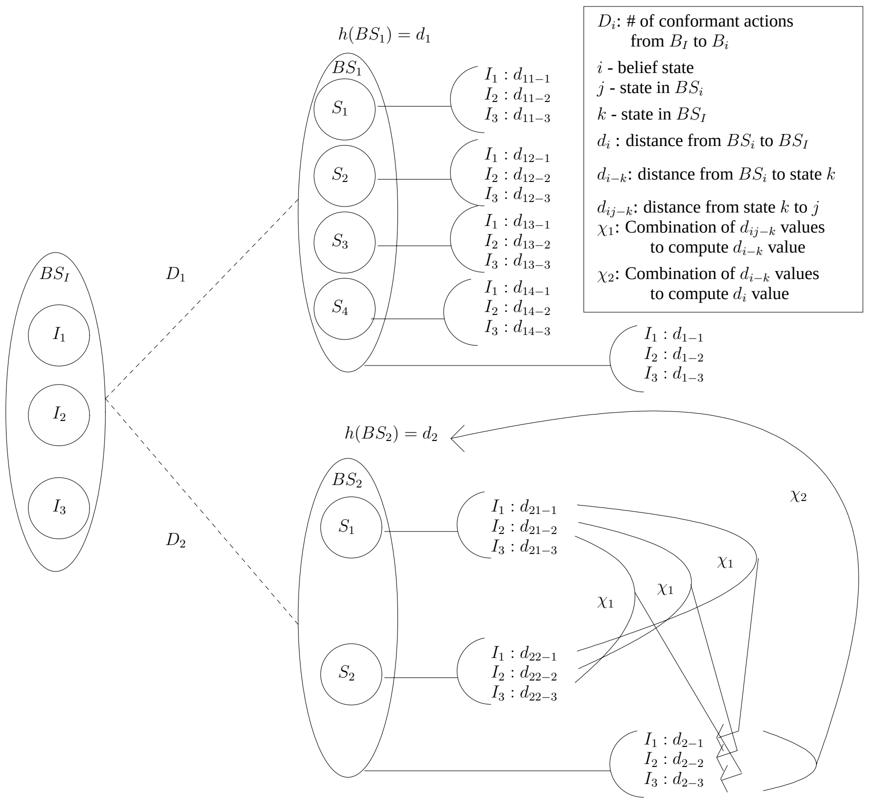

Figure 1: Conformant Plan Distance Estimation in Belief Space

We shall assume that by default all of the heuristics used in $\mathcal{C}AltAlt$ are within the context of weighted A* search (cf. (Bonet & Geffner 1999)).

Cardinality

The idea behind cardinality is to count the number of states that are represented by a belief state. This can be useful in regression because the more states that are in a belief state the better chance that the initial states are in the belief state. The means by which we make this measure is to take a belief state and find its set of constituents, $\xi(BS_i)$, approximating $\hat{\xi}(BS_i)$, and count them. Formally,

$h_{card}(BS_i) = \mid \xi(BS_i) \mid$.

For instance in $CBTC$, $h_{card}(BS_G) = 1$.

Single planning graph heuristics

The base approach for using planning graphs for conformant planning heuristics is to just take all the literals in the initial belief state and insert each literal into the initial layer of the planning graph, ignoring interactions between possible worlds. Thus, for $CBTC$, the initial level of the planning graph is $\{arm, \neg clog, inP1, inP2, \neg inP1, \neg inP2\}$, ignoring the “xor” connective between $inP1$ and $inP2$. Once the planning graph is computed, the level $lev(l)$ at which a particular literal appears in the planning graph is later used at its $cost$. Notice, χ_2 is not applicable because there is only one d_{i-k} value estimated by a single planning graph.

The most simple conformant planning heuristic to compute on a planning graph is

$h_{max}(BS_i) = \max_{C \in \kappa(BS_i)} cost(C)$, where

$cost(C) = \min_{l \in C}(lev(l))$.

Here we estimate χ_1 when constructing the cheapest set of literals and taking the max cost literal. This is an underestimate of the most distant state. Another heuristic is:

$h_{sum}(BS_i) = \sum_{C \in \kappa(BS_i)} cost(C)$

It sums the costs of the literals of the closest estimated state in the belief state to estimate χ_1. Other heuristics considering mutex information can be computed on a single graph, and we have investigated several of them. They are not discussed here for lack of space.

The main disadvantages of single planning graph heuristics is that they make it hard to reason about the *overlap* of independent plans from the initial states, and make it difficult to identify consistent states because the graph is built from an inconsistent union of literals.

Multiple planning graph heuristics

Single graph heuristics are mostly uninformed because the initial belief state corresponds to multiple possible states.

The lack of accuracy is because single graphs are often not able to capture propagation of world specific support information. Consider, in BTC, if $DunkP1$ was the only action we could say that $\neg arm$ is reachable in level 1, but in fact the cost of $\neg arm$ is infinite (since there is no $DunkP2$ to support $\neg arm$ from all initial worlds), and there is no conformant plan.[4]

To account for this and sharpen the heuristic estimate by considering support across all possible worlds, multiple planning graphs Γ are considered. Given the initial belief state BS_I, we grow a planning graph $\gamma_k \in \Gamma$ for each constituent of $\xi(BS_I)$. With multiple graphs, the achievability cost of a belief state is computed in terms of its achievability in all the constituent graphs. We now can estimate many d_{i-k} measures and need to define χ_2 methods to combine them.

For example in BTC, there would be two graphs built (Figure 2). They would have the respective conjunctive initial levels:

$I_1 = \{arm, \neg clog, inP1, \neg inP2\}$

$I_2 = \{arm, \neg clog, \neg inP2, inP2\}$

In the graph for the first world, I_1, $\neg arm$ comes in only through $DunkP1$ at level 1. In the graph for the second world, I_2, $\neg arm$ comes in only through $DunkP2$ at level 1. Thus, the multiple graphs show which actions in the different worlds contribute to the same fact's support.

There are several ways to compute the achievability cost of a belief state with multiple graphs, as follows:

Sum-max ($h_{sum_{max}}$): The easiest heuristic to compute with multiple planning graphs is $h_{sum_{max}}$. The $h_{sum_{max}}(BS_i)$ computes the sum of the cost of the clauses in $\kappa(BS_i)$ for each graph $\gamma_k \in \Gamma$ and takes the maximum. Formally:

$$h_{sum_{\max}}(BS_i) = \max_{\gamma_k \in \Gamma} \left(h_{sum_{\gamma_k}}(BS_i) \right)$$

Here we estimate χ_1, and $\chi_2 =$ maximum. $h_{sum_{max}}$ considers the minimum cost, relevant literals of a belief state (those that are reachable given an initial state for each graph γ_k) to get d_{i-k} measures. The max is taken because the estimate accounts for the worst (i.e., the plan needed in the most difficult world to achieve the subgoals). This max nullifies the chance of getting any overlap information between the worlds, but taking an average or sum wouldn't help either because there is no way to tell overlap by looking at the numerical estimates for each world.

From the $CBTC$, the goal is $BS_G = \neg clog \wedge \neg arm$. Computing the $h_{sum_{max}}(BS_G)$ (Figure 2) finds $h_{sum_{\gamma_1}} = 1$ (denoted by circled facts in the top graph), $h_{sum_{\gamma_2}} = 1$ (denoted by the circled facts in the bottom graph), and the max, $h_{sum_{max}}(G) = 1$.

Level-max ($h_{level_{max}}$): Similar to $h_{sum_{\max}}$, $h_{level_{max}}$ is found by first finding $h_{level_{\gamma_k}}$ to get d_{i-k} for each graph

[4]If any of the planning graphs does not "reach" all of the goals, then this is an indication that a conformant plan does not exist (as would be the case with only the $DunkP1$ action in BTC_2).

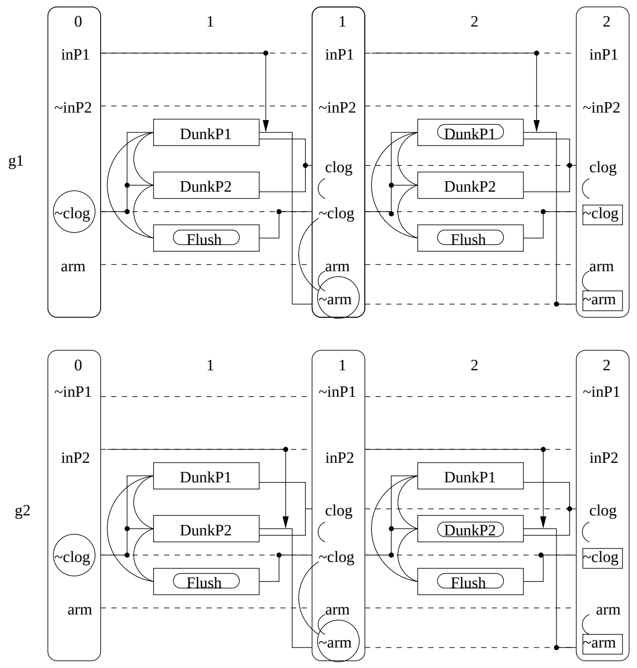

Figure 2: Multiple planning graphs for $CBTC$, with facts used for $h_{sum_{max}}(BS_G)$ circled, facts used for $h_{level_{max}}(BS_G)$ in boxes, and actions for $h_{RP_{max}}(BS_G)$ and $h_{RP_{union}}(BS_G)$ in ovals.

$\gamma_k \in \Gamma$, and then taking the max of this value across the graphs. $h_{level_{\gamma_k}}(BS_i)$ is computed by taking the minimum among the $s \in \xi(BS_i)$, of the first level ($lev(s)$) in the planning graph where literals of s are present with none of them marked mutex. Formally:

$$h_{level_{\gamma_k}}(BS_i) = \min_{s \in \xi(BS_i)} (lev(s))$$

$$h_{level_{max}}(BS_i) = \max_{\gamma_k \in \Gamma} (h_{level_{\gamma_k}}(BS_i))$$

Here we use $\chi_1 =$ minimum to get $h_{level_{\gamma_k}}$, then $\chi_2 =$ maximum for $h_{level_{max}}$. Note that this heuristic is admissible. By the same reasoning as in classical planning, the first level where all the subgoals are present and non-mutex is an underestimate of the true cost of a state. This holds for each of the graphs. Taking the max accounts for the most difficult world in which to achieve a constituent of BS_i and is thus a provable underestimate of h^*. Note, GPT's heuristic (Bonet & Geffner 2000) can be shown to be similar to $h_{level_{max}}$.

For the $CBTC$ goal $BS_G = \neg clog \wedge \neg arm$, computing the $h_{level_{max}}(BS_G)$ (Figure 2) finds $h_{level_{\gamma_1}} = 2$ (denoted by level containing facts inside boxed for the top graph), $h_{level_{\gamma_2}} = 2$ (denoted by level containing facts inside boxed for the top graph), and the max, $h_{level_{max}}(BS_G) = 2$.

RP-max ($h_{RP_{max}}$): Following the same maximization logic as the $h_{sum_{max}}$ and $h_{level_{max}}$ heuristics for χ_2, but to account for the actual number of actions used, $h_{RP_{max}}$ is computed by finding the relaxed plan from the constituent $s \in BS_i$ that contributes to the $h_{level_{\gamma_k}}(BS_i)$ for each

$\gamma_k \in \Gamma$ and taking the max of the number of actions in the relaxed plan.

The relaxed plan for a belief state BS_i is computed by a backward chaining search on the planning graph. We start at the constituent $s \in \xi(BS_i)$, such that s is the constituent at level b, computed in $h_{level_{\gamma_k}}(BS_i) = b$. From s at level b, for each subgoal $l \in s$, a supporting action is selected (ignoring mutexes) from the b^{th} action level. Once, a supporting set of actions ($step_b$) is determined, the needed preconditions for the actions in $step_b$ are added to the list of subgoals to support for level $b-1$. Then, we look at level $b-1$. The algorithm repeats and continues until the initial level is reached. Thus, a relaxed plan is the set RP_{γ_k} =$\{step_{0,\gamma_k}$, ..., $step_{s,\gamma_k}$, ..., $step_{b,\gamma_k}\}$. Formally, when $h_{level_{\gamma_k}}(BS_i) = b$:

$$h_{RP_{max}}(BS_i) = \max_{\gamma_k \in \Gamma}\left(\sum_{s=0}^{b} \mid step_{s,\gamma_k} \mid\right)$$

Here χ_1 = minimum is used to get the cheapest estimated relaxed plan for each initial state, then χ_2 = maximum is used to get d_i. This gives an inadmissible estimate for the number of actions to reach the easiest constituent state in the most difficult world.

For $CBTC$, the goal is $BS_G = \neg clog \wedge \neg arm$. Computing the $h_{RP_{max}}(BS_G)$ (Figure 2) finds $h_{RP_{\gamma_1}} = 3$ ($step_1 = Flush, step_2 = \{DunkP1, Flush\}$; actions in ovals for the top graph), $h_{RP_{\gamma_2}} = 3$ ($step_1 = Flush, step_2 = \{DunkP2, Flush\}$; actions in ovals for the bottom graph), and the max, $h_{RP_{max}}(BS_G) = 3$. Notice that this is the closest multiple graph estimate, so far, for $h^*(BS_G)$, but it can be improved.

RP-union ($h_{RP_{union}}$): Observing the relaxed plans computed by $h_{RP_{max}}$ in the BTC example, we see that the relaxed plans extracted from each graph are different. This information can be leveraged to account for the interaction or overlap of the two initial worlds. Notice, that $step_2$ for both graphs contained a $Flush$ action irrespective of which package the bomb is in. Also, $step_2$ contains a $DunkP1$ for the first graph, and $DunkP2$ for the second graph. Now, taking the union of the two relaxed plans, would give $step_2 = \{DunkP1, DunkP2, Flush\}$, thus accounting for the action that is the same between possible worlds and the actions that differ.

In order to get the union plan, first a relaxed plan is computed for each graph $\gamma_k \in \Gamma$, as in $h_{RP_{max}}$. Then, starting from the last action level ($h_{level_{max}}(BS_i)$) and repeating for each step until the first level, we union the sets of actions for each relaxed plan at each level into another relaxed plan. The relaxed plans are *right-aligned*, hence the unioning of steps proceeds from the last step of each relaxed plan to create the last step of $step_{b,union}$, then the second to last step for each relaxed plan is unioned for $step_{b-1,union}$ and so on. Then the sum of the numbers of actions of the each step in the RP_{union} is used as the heuristic value. Formally, when $h_{level_{max}}(BS_i) = b$:

$$h_{RP_{union}}(BS_i) = \sum_{s=0}^{b} \mid step_{s,union} \mid$$

Here χ_1 = minimum, and χ_2 = union.

$h_{RP_{union}}$ doesn't follow the same form as the rest of the techniques, rather it estimates d_i by finding the relaxed plans corresponding to $min_j(d_{ij})$ for each graph γ_k, then unions the relaxed plans to get the overlap of plans for relevant states.

The insight of this heuristic is that taking the union of action levels of relaxed plans between graphs will account for the same action being used at the same level in multiple worlds. Thus the unioned relaxed plan contains a representative set of *overlapping* actions for achieving the *relevant* states in a belief state for all initial states.

For the $CBTC$ goal $BS_G = \neg clog \wedge \neg arm$, computing the $h_{RP_{union}}(BS_G)$ (Figure 2) finds $RP_{\gamma_1} = \{step_1 = Flush, step_2 = \{DunkP1, Flush\}\}$, $RP_{\gamma_2} = \{step_1 = Flush, step_2 = \{DunkP2, Flush\}\}$, and $RP_{union} = \{step_1 = Flush, step_2 = \{DunkP1, DunkP2, Flush\}\}$. Thus, $h_{RP_{union}}(BS_G) = 4$, which is equal to the optimum estimate $h^*(BS_G)$.

Reducing Heuristic Cost

Searching in the space of belief states complicates heuristic computation by us having to reason about the reachability of sets of states from sets of states. We want to find a state within BS_i that is the easiest to reach, with respect to each initial state. In doing this, an efficiency issue is how to find the belief state to state measure (d_{i-k}) as the minimum among the state to state measures (d_{ij-k}). Up to this point, we've enumerated all of the d_{ij-k} values to find the minimum, but this is costly, so we present two techniques for reducing the cost in finding d_{i-k}.

- Cost the cheapest *set of literals* to get only one d_{ij-k} that is used for d_{i-k}. This corresponds to χ_1 being an estimate.
- Cost the estimated cheapest *constituent* s to get only one d_{ij-k} that is used for d_{i-k}. This also corresponds to χ_1 being an estimate.

The first idea is to combine the cost of a set of literals that satisfies the clauses in the clausal representation of a belief state. This set of literals isn't checked for consistency, and may not even represent a valid state. It is found by assuming χ_1 is an estimated minimum and taking the min cost literal from each clause of $\kappa(BS_i)$ (not checking the resulting set for consistency). This is used for the single planning graph heuristics as well as $h_{sum_{max}}$.

The second idea is to construct only one state, thus avoiding the DNF expansion cost of the enumerative technique.[5] The state we construct is partially specified by all of the unit

[5]Notice, that using this technique for a single planning graph doesn't make sense because consistent states cannot be identified on a planning graph built from unioned initial states.

clauses in the clausal representation of the belief state. However there still remains the choice of an appropriate subset of the literals of the non-unit clauses to complete the state.[6] The appropriate subset is chosen such that the constructed state is the estimated easiest to reach of the set. Here we avoid costing $\mid \xi(BS_i) \mid -1$ constituents. The greedy approach to selecting the subset of literals from the non-unit clauses is to take the single literal from each clause that appears at the lowest level in the planning graph. The algorithm for this selection is as follows:

(1) Sort the literals in each non-unit clause by increasing level of first appearance.

(2) Sort the set of non-unit clauses in decreasing order, using the level of the first element of the clause as the key.

(3) While the set of non-unit clauses is non-empty and the current partial state is a consistent state (i.e. the literals of the partial state appear non-exclusive at some level in the graph).

(a) Insert the first literal of the first non-unit clause into the partial state.

(b) Remove all clauses from the list of non-unit clauses that contain the literal from (a).

(4) If the complete constructed state or partial state is not consistent, then the cost of the belief state is set to infinity, otherwise the cost of the belief state is the cost of the constructed state.

Empirical evaluation

This section presents the results[7] of our experimentation with the heuristics within $\mathcal{C}AltAlt$. We also compare with the competing approaches (CGP, GPT, HSCP, and KACMBP) for several domains and problems.

The implementation of $\mathcal{C}AltAlt$ uses several off the shelf planning software packages. Figure 4 shows a diagram of the system architecture. The pieces of $\mathcal{C}AltAlt$ are the IPC parser for PDDL 2.1 , the HSP-r search engine (Bonet & Geffner 1999), and the IPP planning graph (Koehler *et al.* 1997). The custom parts of the implementation include the action representation, belief state representation and regression operator, not to mention the heuristic calculation.

In addition to the standard domains used in conformant planning–such as Bomb-in-the-Toilet variants, we also developed two new domains. We chose these domains because they demonstrate higher difficulty in the attainment of subgoals, having many plans of varying length.

The $Rovers$ domain is a conformant adaptation of the analogous domain of the IPC. The added uncertainty to the initial state is conditions that rule whether an image objective is visible from various vantage points due to weather.

[6]Only one literal from each clause is taken because selecting more literals will only increase the cost of the state.

[7]All tests were run in Linux on a Pentium 4 1.6GHz w/512MB RAM.

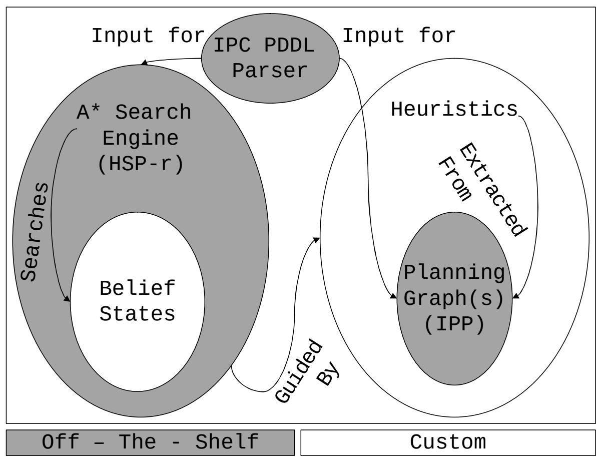

Figure 4: Architecture diagram of C*AltAlt*.

The goal is to upload an image of an objective, thus a conformant plan requires visiting all of the possible vantage points and taking a picture. Significant negative interaction of actions comes in through having to calibrate the camera on an objective before taking the picture, but navigating the rover will de-calibrate the camera.

The $Logistics$ domain is a conformant adaptation of the classical $Logistics$ domain where trucks and airplanes move packages. The uncertainty is the initial locations of packages. The problems scale by adding packages and cities. $Logistics$ shows that conformant planning problems can require reachability heuristics, but the cost of computing the heuristic must be considered in relation to the benefit of the search.

Figure 3 shows the performance of the heuristics within $\mathcal{C}AltAlt$ where the heuristic weight is 5 for all heuristics. For $Rovers$, $h_{RP_{union}}$ performs well, but the unioning approach for χ_2 may be a bit more costly than simply taking a maximum, as in $h_{RP_{max}}$. However, in $Logistics$ the unioning is worth the effort because it reduces the overall search time significantly over maximization. Furthermore, using sets of actions as the d values proves to be more informed than simply using numerical estimates. With the exception of $h_{level_{max}}$, the heuristics based on numerical estimates are largely uninformed on the $Logistics$ and $Rovers$ domains. The reason these other heuristics do not provide as much relevant information is that they are based on single planning graphs; this means that we're not considering overlap of individual world plans or mutex interactions of literals. Notice, that h_{card} doesn't solve any of the problems in these two domains because, as we indicated in section 3, cardinality does not provide accurate reachability information that is necessary in more complex domains.

Problem	HCard	HMax	HSum	HSumMax	HLevelMax	HRPMax	HRPMax*	HRPUnion
Rover1	-	-	-	-	129/5	147/5	145/5	145/5
2	-	3390/8	182272/8	13701/8	4751/8	420/9	420/9	331/9
3	-	23857/10	18185/10	13542/10	5985/10	494/11	489/11	636/11
4	-	-	-	736663/13	-	5173/15	5118/15	15530/15
Logistics1	-	1217/9	147/9	323/9	198/9	343/11	333/11	321/11
2	-	-	26438/15	24638/15	2844/15	11312/17	8979/17	6144/19
3	-	-	2611/14	10079/14	1973/14	7952/17	7536/17	1723/17
4	-	-	-	-	9118/18	306615/22	241311/19	164748/26
BiT2	17/2	3/2	3/2	3/2	5/2	10/2	11/2	11/2
10	78/10	3920/10	3921/10	2737/10	5118/10	10560/10	10253/10	462/10
20	364/20	-	-	-	-	-	-	3380/20
30	-	-	-	-	-	-	-	41114/30
40	-	-	-	-	-	-	-	307590/40
BiTC2	6/3	4/3	4/3	5/3	11/3	16/3	16/3	16/3
10	106/19	9798/19	10072/19	5001/19	10016/19	197191/19	179249/19	801/19
15	353/29	-	-	-	-	-	-	1987/29
20	-	-	-	-	-	-	-	4077/39
25	-	-	-	-	-	-	-	8218/49
30	-	-	-	-	-	-	-	66287/59

Figure 3: This table shows the performance of the planning graph heuristics within $\mathcal{C}AltAlt$. Legend – [Search time (ms)/Plan length], “-”: Out of Memory or Out of Time.

However, h_{card} does perform better in the traditional conformant planning domains: BT, BTC, as expected. The surprising thing is that h_{card}, while outperforming in easier problems, does not scale as well as $h_{RP_{union}}$. $h_{RP_{union}}$'s advantage over h_{card} is that it considers the union of non-unit costs of the minimum-cost constituents $s \in \xi(BS_i)$ in determining reachability estimates rather than the sum of unit costs of $\xi(BS_i)$. The advantage of $h_{RP_{union}}$ over the other planning graph heuristics in these domains is that it actually counts the number of actions that are needed among all worlds. Simply considering a max among worlds gives no discernible information about reachability because among individual worlds the initial states are equidistant.

Figure 3 also shows a comparison of $h_{RP_{max}}$ when using the enumerative and the estimated minimum state technique ($h_{RP_{max}}*$) to illustrate the speedup of not explicitly expanding $\xi(BS_i)$. The benefit of estimating the minimum cost constituent state appears in problems where there are many clauses in the clausal representation of a belief state, hence $\hat{\xi}(BS_i)$ is large, such as in $Logistics$. However, the gain is small in problems were uncertainty is more limited.

Comparisons to other planners: Although this work is aimed at giving a general comparison of heuristics for conformant planning, we also present a comparison of two heuristics within $\mathcal{C}AltAlt$ to some of the other leading approaches to conformant planning. Note, since each approach uses a different planning representation (BDDs, GraphPlan, or explicit state space), and not all of which even use heuristics, it is hard to get a standardized comparison of heuristic effectiveness. The problems for each planner are expressed with binary variables. Although, some of the planners support multi-valued variable encodings, we chose to examine only binary to get a levelled comparison among the competing approaches. Figure 5 compares CGP, GPT, HSCP, and KACMBP with $h_{RP_{union}}$ with respect to run time and plan length.

An observation independent of the planning substrate is the optimality of plans. Optimality can be ensured by using admissible heuristics, but of the heuristic approaches that are inadmissible it is interesting to note that HSCP and KACMBP tend to generate plans that are highly in-optimal with respect to plans generated by $\mathcal{C}AltAlt$ using $h_{RP_{union}}$ in the $Rovers$ and $Logistics$ domains.

For $Rovers$, $h_{RP_{union}}$ provides the best guidance by outperforming CGP, GPT, HSCP, and KACMBP (on some problems). GPT builds a model of over 10000 states for $Rovers_1$, and cannot scale for the other versions of $Rovers$. CGP has trouble constructing its planning graphs as the conformant depth of the goal increases. The bi-level planning graphs in $\mathcal{C}AltAlt$ can handle large domains better than CGP's planning graphs, and thus scale much better.

$Logistics$ provides a better means of comparison. The first thing we see is that HSCP's cardinality heuristic, similar to h_{card}, does not scale well. Yet HSCP does better than h_{card}, indicating that the planning substrate, opposed to the heuristic, may be responsible for the performance. Second, $Logistics$ can have relatively complex planning graphs and as problems scale to include more initial states, multiple

Problem	CGP	GPT	HSCP	KACMBP	HRPUnion
Rover1	135/5	25960/5	4395/5	100/5	145/5
2	5788/7	-	-	14609/8	331/9
3	-	-	-	14916/12	636/11
4	-	-	-	14525/26	15530/15
Logistics1	64/8	69/9	561/9	188/10	321/11
2	1323/11	580/15	-	863/33	6144/19
3	168/10	262/11	-	2576/35	1723/17
4	4559/14	4756/18	-	15225/86	164748/26
BT2	2/1	42/2	8/2	16/2	11/2
10	63/1	234/10	16/10	25/10	462/10
20	1032/1	-	35/20	127/20	3380/20
30	5492/1	-	70/30	373/30	41114/30
40	16005/1	-	137/40	877/40	307590/40
BTC2	2/3	92/3	14/3	21/3	16/3
10	-	288/19	39/19	72/19	801/19
15	-	34280/29	162/29	336/29	1987/29
20	-	-	160/39	330/39	4077/39
25	-	-	479/49	1018/49	8218/49
30	-	-	484/59	1020/59	66287/59

Figure 5: This table shows the performance of CGP, GPT, HSCP, KACMBP in comparison with $h_{RP_{union}}$. Legend – [Search time (ms)/Plan length], "-": Out of Memory or Out of Time.

planning graphs become less attractive. This issue is addressed in the following discussion.

The BT and BTC domains show that $\mathcal{C}AltAlt$ is competitive with CGP and GPT, but is dominated by HSCP and KACMBP with respect to handling common structure of problems.

Related Work

The recent interest in conformant planning can be traced to CGP (Smith & Weld 1998), a conformant version of GraphPlan, here the graph search is conducted on several planning graphs, each constructed from one of the possible initial states. More recent work on C-plan (Castellini, Giunchiglia, & Tacchella 2001) and Frag-Plan (Kurien, Nayak, & Smith 2002) generalize the CGP approach by ordering the searches in the different worlds such that the plan for the hardest to satisfy world is found first, and is then extended to the other worlds. Although $\mathcal{C}AltAlt$ utilizes planning graphs similar to CGP and Frag-plan, in contrast to them, it only uses them to compute reachability estimates. The search itself is conducted in the space of belief states.

Another strand of work models conformant planning as a search in the space of belief states. This started with Genesereth & Nourbakhsh (1993), who concentrated on formulating a set of admissible pruning conditions for controlling search. There were no heuristics for choosing among unpruned nodes. GPT (Bonet & Geffner 2000) extended this idea to consider a simple form of reachability heuristic. Specifically, in computing the estimated cost of a belief state, GPT assumes that the initial state is fully observable. The cost estimate itself is done in terms of reachability (with relaxed dynamic programming rather than planning graphs). GPT's reachability heuristic is similar to our $h_{level_{max}}$ heuristic because they both underestimate the cost of the farthest (max distance) state by looking at a deterministic relaxation of the problem. In comparison to GPT, $\mathcal{C}AltAlt$ can be seen as using heuristics that do a better job of considering the cost of the belief state across the various possible worlds.

A sub-strand of search in belief states is the MBP-family of planners—CMBP, HSCP (Bertoli, Cimatti, & Roveri 2001) and KACMBP (Bertoli & Cimatti 2002). In comparison to $\mathcal{C}AltAlt$, the MBP-family of planners all represent belief states in terms of binary decision diagrams, and action application is modelled as modifications to the BDDs. CMBP and HSCP support both progression and regression in the space of belief states, while KACMBP is a progression planner. While CMBP concentrated on efficient BDD manipulations, HSCP employs cardinality heuristic in addition. Before computing heuristic estimates, KACMBP proactively reduces the uncertainty (disjunction) in the belief state by taking actions that effectively force the agent into states with reduced uncertainty. The motivation for this approach is that applying heuristics to belief states containing multiple states may not give accurate enough direction to the search. While reducing the uncertainty seems to be an effective idea, we note that (a) not all domains may contain actions that reduce belief state uncertainty and (b) the need for uncertainty reduction may be reduced when we have heuristics that effectively reason about the multiple worlds (viz., our multiple planning graph heuristics). Nevertheless, it would be very fruitful to integrate knowledge goal ideas of KACMBP and the reachability heuristics of $\mathcal{C}AltAlt$ to handle domains that contain both high uncertainty and costly goals.

In contrast to these domain-independent approaches that only require models of the domain physics, PKSPlan (Bacchus 2002) is a forward-chaining *knowledge-based planner* that requires richer domain knowledge.

Finally, $\mathcal{C}AltAlt$ is also related to, and an adaptation of the work on reachability heuristics for classical planning, including $AltAlt$ (Nguyen, Kambhampati, & Nigenda 2002), FF (Hoffmann & Nebel 2001) and HSP-r (Bonet & Geffner 1999).

Conclusion

With the intent of scaling conformant planning to domains where reachability of subgoals is a non-trivial search problem, we have:

1. Discussed what the heuristic measures for conformant planning that should be estimating.
2. Shown how to compute such heuristic measures on planning graphs when using a factored representation of belief states.
3. Provided empirical comparisons of these measures.

The best heuristics reported in this paper are based on tracking multiple planning graphs. Although effective, the multiple graph heuristics do suffer from certain shortcomings: we need to track one graph for each potential initial state (although we do have the option of ignoring some of the initial states at the expense of lower quality of heuristic). The multiple planning graphs also have a large amount of redundancy when the possible worlds are quite similar. To overcome these disadvantages, we have recently pursued an idea called *labeled planning graphs*, which uses a single condensed graph. The literals and actions of the condensed graph would be labeled to indicate the worlds in which they are supported. When computing heuristic costs, a literal's level is not the first level where it appears, but the first level where its label indicates it has full support in every world. A forthcoming companion paper (Bryce, Kambhampati, & Smith 2004) reports on the results from this investigation.

Acknowledgements: We thank Minh B. Do, Romeo Sanchez, Terry Zimmermam, and Satish Kumar Thittamaranahalli for helpful discussions and feedback. We also thank David Smith and Piergiorgio Bertoli for help with the CGP and MBP planners. This research is supported in part by the NASA grants NCC2-1225 and NAG2-1461, and the NSF grant IIS-0308139.

References

Bacchus, R. P. P. F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In *Artificial Intelligence Planning Systems*, 212–221.

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

Bertoli, P.; Cimatti, A.; and Roveri, M. 2001. Heuristic search + symbolic model checking = efficient conformant planning. In *ijcai*.

Bonet, B., and Geffner, H. 1999. Planning as heuristic search: New results. In *ECP*, 360–372.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Artificial Intelligence Planning Systems*, 52–61.

Bryce, D.; Kambhampati, S.; and Smith, D. 2004. Conformant planning with a labelled uncertainty graph. Submitted for publication.

Castellini, C.; Giunchiglia, E.; and Tacchella, A. 2001. Improvements to sat-based conformant planning. In *6th European Conference on Planning*.

Genesereth, M. R., and Nourbakhsh, I. R. 1993. Time-saving tips for problem solving with incomplete information. In *Proceedings of the 11th National Conference on Artificial Intelligence*, 724–730. Menlo Park, CA, USA: AAAI Press.

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

Koehler, J.; Nebel, B.; Hoffmann, J.; and Dimopoulos, Y. 1997. Extending planning graphs to an ADL subset. Technical Report report00088, IBM.

Kurien, J.; Nayak, P. P.; and Smith, D. E. 2002. Fragment-based conformant planning. In *Artificial Intelligence Planning Systems*, 153–162.

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

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

Pednault, E. P. D. 1987. Synthesizing plans that contain actions with context-dependent effects. Technical Memorandum, AT&T Bell Laboratories, Murray Hill, NJ. (submitted to the Journal of Artificial Intelligence).

Smith, D. E., and Weld, D. S. 1998. Conformant graphplan. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, 889–896. Menlo Park: AAAI Press.