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 combine them 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 within a conformant regression planner named $\mathcal{C}AltAlt$.

1 Introduction

Ever since CGP [Smith and Weld, 1998] a series of planners have been developed for tackling conformant planning problems – including GPT [Bonet and Geffner, 2000], C-Plan [Castellini *et al.*, 2001], PKSPlan [Bacchus, 2002], Frag-Plan [Kurien *et al.*, 2002], HSCP [Bertoli *et al.*, 2001], and KACMBP [Bertoli and Cimatti, 2002]. Some of these planners are extensions of heuristic state search planners, yet despite their success there is as yet little understanding on what the basis for heuristics should be.

Given the different search strategies used by the planners, it was hard to analyze the impact of various heuristics apart from the planning substrate that is used. Of the different strategies, only GPT and KACMBP use non-trivial reachability heuristics, however the former uses an explicit representation, and the latter uses a factored representation with BDDs. GPT has even been outperformed by HSCP (a precursor to KACMBP that relies only on belief state cardinality for guidance). This says little of the merit of the heuristics of the two approaches because while HSCP's search engine, based on binary decision diagrams, is quite sophisticated, its heuristics are quite primitive. KACMBP improves upon HSCP by combining an adjusted cardinality heuristic with a reachability heuristic. The forward chaining planner uses a reachability heuristic similar to FF [Hoffmann and Nebel, 2001], in that it takes a relaxed projection from the current belief state to the goal and sums over the costs of literals in the current belief state, or finds the maximum distance between any state in the current belief state and the goal state. The adjusted cardinality and reachability are combined by a weighted sum to get a heuristic value. Still, comparisons of the heuristic effectiveness of KACMBP cannot be decoupled from the effectiveness of the search substrate.

Thus, we will discuss the effectiveness of the heuristic techniques in a decoupled fashion to explore what measures may be beneficial for any substrate. We argue that the previous idea of using the maximum distance between any pair of states in the belief states in the distance computation is not the best means to assign heuristic values to a search belief state, nor does cardinality have a real meaning when costing distance. The reason that the maximum distance is used by forward-chaining planners like KACMBP and GPT is that there is a single goal state and each of the states in the current belief state must be able to reach the goal, otherwise the distance for the current belief state is infinite. However, when there are multiple goal states, then a maximum of all distances doesn't make sense because all of the states in the current belief state must reach *a* goal state, not *all* goal states. Also, cardinality has been seen as an effective heuristic for guiding search by leveraging the knowledge that there are multiple initial states and a single goal state. However, as search progresses or regresses, there can be an arbitrary number of states in any belief state. Cardinality just happens to work well when the domain is structured such that there is a monotonic increase or decrease in the size of a belief states during search. In other words, each action that is used in a conformant plan will either increase (in regression) or decrease (in progression) the size of the current belief state, and all other actions either decrease (in regression), increase (in progression), or maintain the size of the belief state. This occurs in many of the domains in conformant planning literature like BiT, $Cube$, and $Ring$. Cardinality can lead the search astray in the BiT domain, for instance, if there were many packages but in the initial state we specify that the bomb is in one of a subset of the packages. In regression, there will still be applicable actions for dunking each of the packages (possibly containing a bomb or not). Cardinality will direct the search to arbitrarily select dunk actions that may or may not be for packages that are relevant to the problem because regressing each action will increase the cardinality of a belief state.

Given these observations, our intent is to:

1. Describe, in general, what heuristic estimates for conformant planning should be measuring
2. Show how such heuristics can be computed with plan-

ning graphs

3. Provide empirical comparisons of the computation approaches.

To facilitate this discussion, we describe and evaluate the heuristics within a conformant planner called $\mathcal{C}AltAlt$.

$\mathcal{C}AltAlt$ does regression search in the space of "clausal states" (which are conjunctive representations for sets of states). The challenges in developing planning graph based heuristics for $\mathcal{C}AltAlt$ include: (i) handling the reachability cost of sets of states represented as clauses and (ii) handling uncertainty in the initial state by basing heuristics on multiple (rather than a single) planning graphs. 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 forward space combination heuristics used within KACMBP.

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.

2 State and Action Representation

State Representation: As discussed in [Bonet and Geffner, 2000], conformant planning can be seen as a search in the space of belief states.

A **belief state** BS_i is a set of multiple world states $\{S_1, ..., S_j, ..., S_n\}$.

We choose clausal states as a factored representation, in CNF, of belief states.

A **clausal state** CS_i, logically equivalent to BS_i, is a set of clauses $\{C_1, ..., C_l, ..., C_m\}$ where each C_l is a disjunction of a set of literals $\{c_1, ..., c_h, ..., c_p\}$. A *conjunctive clausal state* $CS_i^{\prime}$ is a set of unit clauses, when $\mid BS_i \mid = 1$. $CS_i^{\prime}$ also refers to a state S_j with a conjunctive set of literals $\{c_1, ..., c_h, ..., c_p\}$.

Using the $BiTC$[1] problem as a running example for this paper, the clausal state representation of $BiTC$'s initial state is:

$CS_I = [arm, \neg clog, (inP1 \vee inP2), (\neg inP1 \vee \neg inP2)]$

A clausal state CS_i is said to be **satisfied** by another clausal state $CS_{i'}$ if every clause C_l in CS_i is satisfied by $CS_{i'}$. More specifically, CS_i is satisfied by $CS_{i'}$ if

$\forall_{C_l \in CS_i} \exists_{C_{l'} \in CS_{i'}}$ s.t. $C_{l'} \subseteq C_l$.

For example, if $CS_I = [(inP1 \vee inP2), (\neg inP1 \vee \neg inP2)]$, and $CS_i = [(\neg clog \vee inP1 \vee inP2)]$, then CS_i is satisfied by CS_I.

A clausal state CS_i is said to be **inconsistent** with another state $CS_{i'}$, if there is a clause $C_l \in CS_i$ s.t. $\neg C_l$ is satisfied by $CS_{i'}$.

For example, if $CS_I = [\neg arm, \neg clog, (inP1 \vee inP2), (\neg inP1 \vee \neg inP2)]$, and $CS_i = [clog, arm, (inP1 \vee inP2)]$, then CS_i is inconsistent with CS_I because the negation of arm in CS_i is satisfied by $\neg arm$ in CS_I.

The set of **constituents** $\xi(CS_i)$ of a clausal state is all *minimal* conjunctive clausal states S_j that satisfy CS_i. $\xi(CS_i)$ is equivalent to a DNF representation of a clausal state's CNF representation. For example, $BiTC$'s initial state $CS_I = [arm, \neg clog, (inP1 \vee inP2), (\neg inP1 \vee \neg inP2)]$; the set of constituents $\xi(CS_I) = \{[arm, \neg clog, inP1, \neg inP2], [arm, \neg clog, inP2, \neg inP1]\}$.

Since we're dealing with partial regression states, $\xi(CS_i)$ may not represent all states in a belief state BS_i, so we define $\hat{\xi}(CS_i)$ as the *complete* set of states represented by BS_i. For example, if $BiTC$'s initial state were partial $CS_I = [arm, (inP1 \vee inP2), (\neg inP1 \vee \neg inP2)]$; the set of constituents $\xi(CS_I) = \{[arm, inP1, \neg inP2], [arm, inP2, \neg inP1]\}$, but $\hat{\xi}(CS_I) = \{[arm, clog, inP1, \neg inP2], [arm, \neg clog, inP2, \neg inP1], [arm, clog, inP1, \neg inP2], [arm, \neg clog, inP2, \neg inP1]\}$.

Action Representation: An action a_g, of the action set $A = \{a1, ..., a_g, ..., a_q\}$, is described in terms of (1) an executability precondition ρ_e, and (2) several conditional effects of the form ($\varphi_f : \rho_f \rightarrow \varepsilon_f$), where ρ_f and ε_f are, in general, clausal states[2]. The executability precondition ρ_e (a clausal state) of the action must hold for the action to be executable. Each conditional effect is of the form [antecedent (precondition ρ_f) $\rightarrow$ consequent (effect ε_f)]. The antecedent or consequent of the individual effects can be empty. In the first case, the action has a defined outcome in any state; and in the second case, the corresponding effects occur in all worlds. The conditional effects φ_f make up a set $\Phi_{a_g} = \{\varphi_1, ..., \varphi_f, ..., \varphi_r\}$.

As an example, the actions for $BiTC$ are expressed as:

$a_{DunkP1} : \{\rho_e : \neg clog, \rho_1 : inP1 \rightarrow \varepsilon_1 : \neg arm, \varepsilon_2 : clog\}$
$a_{DunkP2} : \{\rho_e : \neg clog, \rho_1 : inP2 \rightarrow \varepsilon_1 : \neg arm, \varepsilon_2 : clog\}$
$a_{Flush} : \{\varepsilon_1 : \neg clog\}$

Regression: Conformant planning by regression is just a search in the space of clausal states, starting with the goal state and regressing it non-deterministically over all relevant actions. Clausal states are regressed until finding a clausal state that is satisfied by the initial state. The main difference between regression search in conformant and classical planning is that because of disjunction in the initial state a conformant planner cannot split the disjunction in a regressed state[3] into the search space – and thus has to handle disjunctive clausal states directly.

Following [Pednault, 1987], regressing a clausal state CS_i over an action a_g involves taking the union of causation and preservation clauses of each $C_l \in CS_i$ w.r.t. each effect φ_f of a_g. Formally, the result $CS_{i'}$ of regressing the clausal state CS_i over the action a_g is defined as:

$CS_{i'} = Regress(CS_i, a_g) = \forall_{C_l \in CS_i} \forall_{\varphi_f \in \Phi_{a_g}} \rho_e \cup \Sigma_{a_g}^{\varphi_f}(C_l) \cup \Pi_{a_g}^{\varphi_f}(C_l)$

[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 (BiT)–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 $BiTC$ says that the toilet will clog unless it is "flushed" after each "dunking" action.

[2]ε_f is a conjunctive clausal state because we are only considering non-deterministic actions.

[3]To see this, consider that we have a goal $p \vee q$. Splitting the disjuncts into the search space treats the goal as $p \wedge q$, which will not be satisfi ed if the initial state has $p \vee q$.

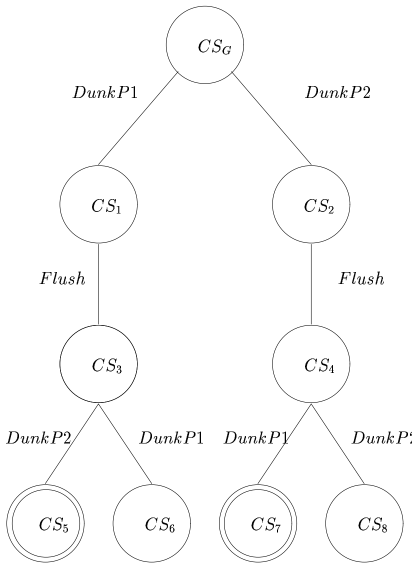

Figure 1: Regression path for $BiTC$ example.

Executability clause (ρ_e) is the executability precondition of a_g. This is what must hold in $CS_{i'}$ for a_g to have been applicable.

Causation clause for a clause C_l w.r.t an effect $\varphi_f : \rho_f \rightarrow \varepsilon_f$ of an action a_g (denoted by $\Sigma_{a_g}^{\varphi_f}(C_l)$) is defined as the weakest clause that must hold in the state before a_g such that φ_f causes C_l. Formally $\Sigma_{a_g}^{\varphi_f}(C_l)$ is defined as:
$\{C_l \vee \rho_f \mid \varphi_f : \rho_f \rightarrow \varepsilon_f \in \Phi_{a_g}$ and ε_f satisfies $C_l\}$.

Preservation clause of a clause C_l w.r.t. an effect $\varphi_f : \rho_f \rightarrow \varepsilon_f$ of action a_g (denoted by $\Pi_{a_g}^{\varphi_f}(C_l)$) is defined as the weakest clause that must be true before a_g such that C_l is not violated by the effect ε_f of φ_f. Formally $\Pi_{a_g}^{\varphi_f}(C_l)$ is defined:
$\{\neg\rho_f \mid \varphi_f : \rho_f \rightarrow \varepsilon_f \in \Phi_{a_g}$ and ε_f satisfies $\neg C_l\}$.

Example of Regression and Search: Since clausal state regression is not commonly discussed in planning literature, we will now give a complete example of clausal state regression search within $BiTC$. The search states are shown schematically in Figure 1. We start with initial and goal clausal states:
$CS_I = [arm, \neg clog, (inP1 \vee inP2), (\neg inP1 \vee \neg inP2)]$
$CS_G = [\neg arm]$

a_{DunkP1} and a_{DunkP2} are applicable for regression at CS_G because they both have $\neg arm$ as a conditional effect consequent. The regressed states CS_1 and CS_2 are constructed from the causation clause and the executability clause for the respective actions.
$CS_1 = Regress(CS_G, a_{DunkP1}) = [\neg clog, (inP1 \vee \neg arm)]$
$CS_2 = Regress(CS_G, a_{DunkP2}) = [\neg clog, (inP2 \vee \neg arm)]$

Notice that in classical planning, we would have said here that $Regress(CS_G, a_{DunkP1}) = [\neg clog, inP1]$ since $\neg arm$ is directly given by a_{DunkP1}. However, the weakest preconditions for $\neg arm$ to be true after a_{DunkP1} is $(inP1 \vee \neg arm)$ rather than just $\neg arm$.

a_{Flush} is applicable to both CS_1 and CS_2 because it has $\neg clog$ as an effect. The regressed states CS_3 and CS_4 are constructed based on the executability clause, but the precondition is always true, so the only change is the removal of $\neg clog$.
$CS_3 = Regress(CS_1, a_{Flush}) = [(inP1 \vee \neg arm)]$
$CS_4 = Regress(CS_2, a_{Flush}) = [(inP2 \vee \neg arm)]$

Finally, both $Dunk$ actions are applicable again for the same reason as the generation of CS_1 and CS_2. Choosing the unchosen $Dunk$ action, given the search path, will lead to a state with new information. After regressing all applicable $Dunk$ actions, we get CS_5, CS_6, CS_7, and CS_8.
$CS_5 = Regress(CS_3, a_{DunkP2}) = [\neg clog, (inP1 \vee inP2 \vee \neg arm)]$
$CS_6 = Regress(CS_3, a_{DunkP1}) = [\neg clog, (inP1 \vee \neg arm)]$
$CS_7 = Regress(CS_4, a_{DunkP1}) = [\neg clog, (inP1 \vee inP2 \vee \neg arm)]$
$CS_8 = Regress(CS_4, a_{DunkP2}) = [\neg clog, (inP2 \vee \neg arm)]$

CS_5 and CS_7 are both satisfied by CS_I because every clause in the clausal states is satisfied by a clause in the initial state, so either is a terminal search node, and the path of actions leading to it is a conformant plan.

3 Factored Belief State Distance Estimation

We will start by discussing what measures are worth estimating for providing heuristic guidance for conformant planning. Consider the example in Figure 2; 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 labeled 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 that a larger belief state has more probability of containing the states in 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 or estimates directly to get the d_{i-k} measures, and χ_2, which combines the d_{i-k} measures or estimates directly to get d_i. The applicable operations allowable

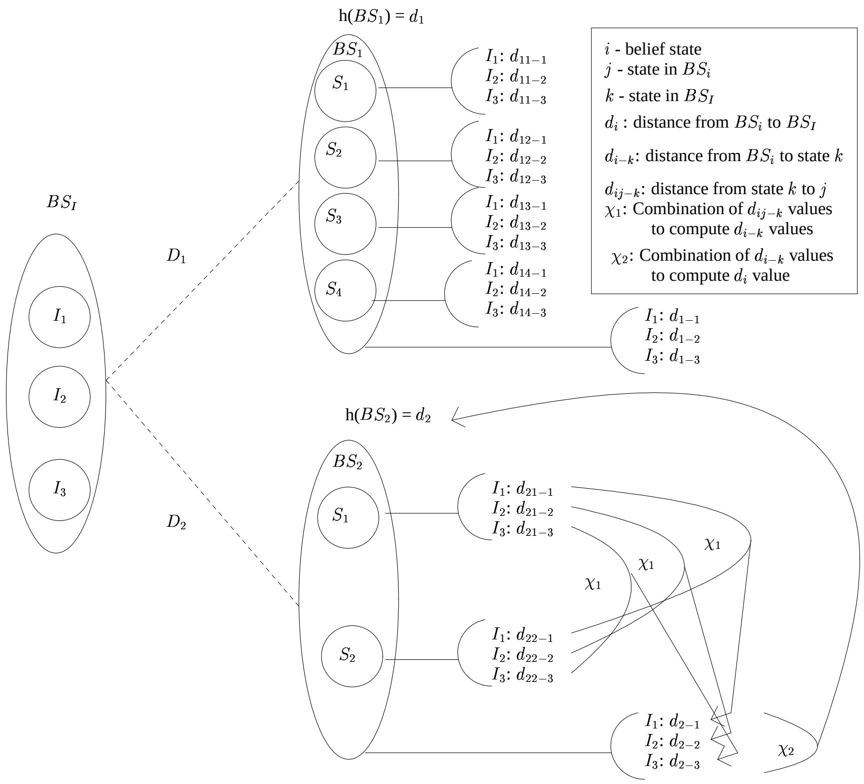

Figure 2: Example of Conformant Plan Distance Estimation in Belief Space

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, 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.

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 $\xi(BS_i)$, hence 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.

For example, in Figure 2, only using $\mid \hat{\xi}(BS_i) \mid$ to estimate d_i will tell us that BS_1 is a better choice because there are four states in it, opposed to two in BS_2. This may not be the best choice because it could be the case that $\forall_{j,k} d_{1j-k} = \infty$.

Alternatively, abandoning cardinality and using a heuristic similar to GPT, we'd set both χ_1 and χ_2 to take a max in order to find the $\max_{j,k} d_{ij-k}$ to for the estimate d_i. However, this is problematic for the same reason as $\mid \hat{\xi}(BS_i) \mid$ because maybe $\exists_{j,k} d_{1j-k} = \infty$. Then we'd eliminate BS_1 from consideration by assigning d_1 a cost of ∞, even though the other states in BS_1 may all have very low distances to BS_I, with respect to the distances of states in BS_2.

This brings up an important point for considering how to define χ_1 and χ_2 for costing a belief state. We would prefer χ_1 to be a minimization because a state $S_j \in BS_i$ may not even be relevant to achieving the state $S_k \in BS_I$ and taking the max would give $d_{i-k} = \infty$. The minimization is important because we need only one state $S_j \in BS_i$ to have a finite distance for each $S_k \in BS_I$. However, for χ_2 it is important to take a maximization because the distance from BS_i to BS_I is at least the largest distance from the minimum-cost *relevant* states of BS_i, and if for any state $S_k \in BS_I$ that the distance $d_{i-k} = \infty$ then $d_i = \infty$ because that initial state S_k is unreachable, from BS_i.

Another approach, that of KACMBP [Bertoli and Cimatti, 2002], is to consider a heuristic that combines reachability with relevance-based cardinality (or as they call truth percentage) for forward chaining search. The reachability measure is taken either similar to GPT as a $\max_{j,k} d_{ij-k}$, or a sum over the costs of goal literals in a projection from B_i to get d_i.

The lesson to be learned 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 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 min-cost distances from each of the 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.

4 Heuristics

This section provides, first, three sets of heuristics that estimate these distance computations, and second, approximations to the distance computations that improve performance.

All of the heuristics used in $\mathcal{C}AltAlt$ are within the context of greedy best first search (cf. [Bonet and Geffner, 1999]), where the reachability cost of a clausal state is $f(CS_i) = g(CS_i) + w * h(CS_i)$. The $g(CS_i)$ term is the number of actions regressed from the goal state to reach CS_i, w is the weight term, and $h(CS_i)$ is the heuristic estimate of how many actions are needed to reach the initial state from CS_i. The search is guided by expanding the clausal states with the lowest cost $f(CS_i)$.

For the remainder of this section, to illustrate the computation of each heuristic, we use an example from $BiTC$ called $CBiTC$,[4] 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 clausal state is $CS_I = [arm, \neg clog, (inP1 \vee inP2), (\neg inP1 \vee \neg inP2)]$, and the goal is $CS_G = [\neg clog, \neg arm]$. The optimal action sequences to reach G from I is: $\{DunkP1 \rightarrow Flush \rightarrow DunkP2 \rightarrow Flush\}$, or $\{DunkP2 \rightarrow Flush \rightarrow DunkP1 \rightarrow Flush\}$, thus the optimal heuristic estimate is $h^*(CS_G) = 4$ because in either plan there are four actions.

4.1 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 first means by which we make this measure is to take a belief state and find its set of constituents, $\xi(CS_i)$, to approximate $\hat{\xi}(CS_i)$. Using a clausal state CS_i directly, we expand CS_i into its set of constituent states $\xi(CS_i)$ and count them. Formally,

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

For instance in $CBiTC$, $h_{card}(CS_G) = 1$.

4.2 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 state clauses and insert each literal into the initial layer of the planning graph, ignoring interactions between possible worlds. Thus, for $CBiTC$, the initial level of the planning graph is expressed as $CS_I = [arm, \neg clog, inP1, inP2, \neg inP1, \neg inP2]$, ignoring the "xor" connective between $inP1$ and $inP2$. Once the planning graph is computed, the level $l(c_h)$ at which a particular literal appears in the planning graph is later used at its $cost$. Notice, $\chi_2 = \oslash$ 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

$h_{max}(CS_i) = \max\limits_{C_l \in CS_i} cost(C_l)$, where

$cost(C_l) = \min_{c_h \in C_l}(l(c_h))$.

Here we use χ_1 = estimate, and $\chi_2 = \oslash$ when constructing the cheapest set of literals and taking the max cost literal. This is approximate to GPT's heuristic because it takes the max distance to reach a literal of the goal state, which is an underestimate of the most distant state. Another heuristic is:

[4]Courteous BiTC.

$h_{sum}(CS_i) = \sum\limits_{C_l \in CS_i} cost(C_l)$

which sums costs of the literals of the closest estimated state in the belief state. It uses χ_1 = estimate, and $\chi_2 = \oslash$. 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.

4.3 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 $BiTC$, if $DunkP1$ was the only action, then $DunkP1$ has nothing to be mutex with. 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 fully support $\neg arm$), and there is no conformant plan[5].

To account for this and sharpen the heuristic estimate by accounting for support across all possible worlds, multiple planning graphs Γ are considered. Given the initial clausal state CS_I, we grow a planning graph $\gamma_k \in \Gamma$ for each conjunctive initial state $S_k \in \xi(CS_I)$. With multiple graphs, the achievability cost of a clausal state is computed in terms of its achievability in all the constituent graphs. In general we only build the minimal independent set of graphs for CS_I because $\xi(CS_I)$ is the set of minimally satisfying states. Hence, one disjunct is chosen from each clause to construct a graph, thus the independent set of graphs. We now can estimate many d_{i-k} measures and need χ_2 to combine them.

For example in $BiTC$, there would be two graphs built (Figure 3). They would have the respective conjunctive initial levels:

$S_{I_1} : [arm, \neg clog, inP1, \neg inP2]$
$S_{I_2} : [arm, \neg clog, \neg inP2, inP2]$

In the graph for the first world, S_{I_1}, $\neg arm$ comes in only through $DunkP1$ at level 1. In the graph for the second world, S_{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 clausal 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 $h_{sum_{max}}(CS_i)$ computes the sum of the cost of the clauses in CS_i for each graph $\gamma_k \in \Gamma$ and takes the maximum. Formally:

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

Here we use χ_1 = estimate, and χ_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

[5]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 $BiTC_2$).

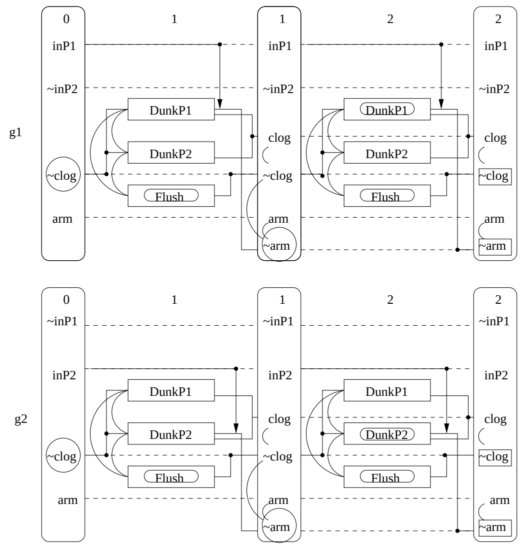

Figure 3: Multiple planning graphs for $CBiTC$, with facts used for $h_{sum_{max}}(CS_G)$ circled, facts used for $h_{level_{max}}(CS_G)$ in boxes, and actions for $h_{RP_{max}}(CS_G)$ and $h_{RP_{union}}(CS_G)$ in ovals.

difficult world to achieve the subgoals)[6]. 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 $CBiTC$, the goal is $CS_G = [\neg clog, \neg arm]$. Computing the $h_{sum_{max}}(CS_G)$ (Figure 3) 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 $\gamma_k \in \Gamma$, then the max of this value across the graphs is taken. $h_{level_{\gamma_k}}(CS_i)$ is computed by taking the minimum among the $S_j \in \xi(CS_i)$, of the first level ($lev(S_j)$) in the planning graph where no two literals in the constituent S_j are mutually exclusive. Formally:

$$h_{level_{\gamma_k}}(CS_i) = \min_{S_j \in \xi(CS_i)}(lev(S_j))$$
$$h_{level_{max}}(CS_i) = \max_{\gamma_k \in CS_G}(h_{level_{\gamma_k}}(CS_i))$$

Here we use χ_1 = minimum or estimate to get $h_{level_{\gamma_k}}$, then χ_2 = maximum for $h_{level_{max}}$. Note, this heuristic is admissible. By the same reasoning as in classical planning, the first level where all the subgoals are 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 state of CS_i and is thus a provable underestimate of h^*.

For the $CBiTC$ goal $CS_G = [\neg clog, \neg arm]$, computing the $h_{level_{max}}(CS_G)$ (Figure 3) 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, $h_{level_{max}}(CS_G) = 2$.

[6]This closely resembles the reachability heuristic used in KACMBP.

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 $S_j \in CS_i$ that contributes to the $h_{level_{\gamma_k}}(CS_i)$ for each $\gamma_k \in \Gamma$ and taking the max of the number of actions in the relaxed plan.

The relaxed plan for a clausal state CS_i is computed by a backward chaining search on the planning graph. We start at the constituent $S_j \in \xi(CS_i)$, such that S_j is the constituent at level b, computed in $h_{level_{\gamma_k}}(CS_i) = b$. From S_j at level b, for each subgoal $c_h \in S_j$, a supporting action is selected (ignoring mutexes) from the b^{th} action level. Once, a supporting set of actions ($step_b$) is determined, the support for the actions in $step_b$ is 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} =$\{step_{1,\gamma_k}$, ..., $step_{s,\gamma_k}$, ..., $step_{b,\gamma_k}\}$. Formally, when $h_{level_{\gamma_k}}(CS_i) = b$:

$$h_{RP_{max}}(CS_i) = \max_{\gamma_k \in \Gamma}\left(\sum_{s=1}^{b} \mid step_{s,\gamma_k} \mid\right)$$

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 $CBiTC$, the goal is $CS_G = [\neg clog, \neg arm]$. Computing the $h_{RP_{max}}(CS_G)$ (Figure 3) 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, $h_{RP_{max}}(CS_G) = 3$. Notice that this is the closest multiple graph estimate, so far, for $h^*(CS_G)$, but it can be improved.

RP-union ($h_{RP_{union}}$): Observing the relaxed plans computed in the $BiTC$ example given for $h_{RP_{max}}$, the relaxed plans extracted from each graph are different. This information can be leveraged to account for the interaction or overlap of the two worlds. Notice, that $step_2$ for both graphs contained a $Flush$ action which is not dependent on whether the bomb is in either package. 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.

A relaxed plan is computed for each graph $\gamma_k \in \Gamma$, as in $h_{RP_{max}}$. Then, starting from the last action level($h_{level_{max}}(CS_i)$) and repeating for each $step_s$ until the first level, we union the sets of actions for each relaxed plan at each level into another relaxed plan [$RP_{union}(CS_i) = \forall_{step_i} \cup_{\gamma_k \in \Gamma} step_{i,\gamma_k}$]. Notice, 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 $step_s$ in the RP_{union} is used as the heuristic value. Formally, when $h_{level_{max}}(CS_i) = b$:

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

Here $\chi_1 =$ minimum, and $\chi_2 =$ 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 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, or overlap. Thus the unioned relaxed plan contains a representative set of *overlapping* actions for achieving the *relevant* states in a clausal state in all worlds.

For the $CBiTC$ goal $CS_G = [\neg clog, \neg arm]$, computing the $h_{RP_{union}}(CS_G)$ (Figure 3) 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, $h_{RP_{union}}(CS_G) = 4$, which is equal to the optimum estimate $h^*(CS_G)$.

4.4 Reducing the Cost of Estimating Belief State Distance with Clausal States

Searching in the space of clausal 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 the set BS_i that is the easiest to reach, with respect to the number of actions needed of each γ_k in the set of n planning graphs to get d_i. When $n > 1$, we are using multiple planning graphs to represent different initial states, and need to perform some combination χ_2 of the d_{i-k} measures for each γ_k in the set of n planning graphs to get d_i. Otherwise when $n = 1$, d_i is simply the d_{i-1} measure. However, there still remains the issue of finding d_{i-k} from the d_{ij-k}'s. There are three main ways ,Θ, to get the cost d_{i-k} measures of a belief state BS_i, with respect to an initial state belief state BS_I.

- Θ_1 : *Expand* $\xi(CS_i)$ and get d_{ij-k}, for each of the states S_j on graph γ_k corresponding to $S_k \in BS_I$. Then we could combine all of the d_{ij-k} to get d_{i-k} by using χ_1.
- Θ_2 : Cost the cheapest minimal 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.
- Θ_3 : Cost the estimated cheapest minimal *state* 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, and complete, method Θ_1 for finding d_{i-k} is to explicitly expand a clausal state into the set of states that it represents (equated with converting CNF to DNF), by finding $\xi(CS_i)$, then getting a heuristic estimate for each state in $\xi(S)$ and taking the minimum, maximum, or average for χ_1. Clearly, a clausal state can represent an exponential number of states, so we desire a more efficient (less explicit) means of finding the d_{i-k}. Notice that if χ_1 were a summation that we would be able to adjust the heuristic to consider the cardinality of CS_i.

The second idea Θ_2 is to combine the cost of a set of literals that minimally satisfies a clausal state. This set of literals isn't checked for consistency, and may not even represent a valid state. It is found by assuming $\chi_1 =$ estimated minimum and taking the min cost literal from each clause (not checking the resulting set for consistency). Θ_2 is used for the single planning graph heuristics as well as $h_{sum_{max}}$. All other heuristics, besides h_{card} use either Θ_1 or Θ_3.

The third idea Θ_3 is construct only one state and use it for the estimate (enforcing $\chi_1 =$ minimum), thus avoiding the DNF expansion cost of Θ_1[7]. The state we construct is partially specified by all of the unit clauses in the clausal state. However there still remains the choice of an appropriate subset of the literals of the non-unit clauses to complete the state[8]. The appropriate subset is chosen such that the constructed state is the estimated easiest to reach of the set. This is similar to Θ_1 when $\chi_1 =$ min, but we avoid costing $n - 1$ states. 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 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 clausal state is set to infinity, otherwise the cost of the clausal state is the cost of the constructed state.

5 Empirical evaluation

This section presents the results[9] 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. The pieces of $\mathcal{C}AltAlt$ are the IPC parser for PDDL 2.1 , the HSP-r search engine [Bonet and Geffner, 1999], and the IPP planning graph [Koehler *et al.*, 1997]. The custom parts of the implementation include the action representation, clausal 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. The

[7]Notice, that using Θ_3 for a single planning graph doesn't make sense because consistent states cannot be identifi ed on a planning graph built from unioned initial states.

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

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

Problem	HCard	HMax	HSum	HSumMax	HLevelMax	HRPMax Θ_1	HRPMax Θ_3	HRPUnion
Rover1	0/0	0/0	0/0	0/0	**129/5**	147/5	145/5	145/5
2	0/0	3390/8	182272/8	13701/8	4751/8	420/9	420/9	**331/9**
3	0/0	23857/10	18185/10	13542/10	5985/10	494/11	**489/11**	636/11
4	0/0	0/0	0/0	736663/13	0/0	5173/15	5118/15	15530/15
Logistics1	0/0	1217/9	**147/9**	323/9	198/9	343/11	333/11	321/11
2	0/0	0/0	26438/15	24638/15	**2844/15**	11312/17	8979/17	6144/19
3	0/0	0/0	2611/14	10079/14	1973/14	7952/17	7536/17	**1723/17**
4	0/0	0/0	0/0	0/0	**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**	0/0	0/0	0/0	0/0	0/0	0/0	3380/20
30	0/0	0/0	0/0	0/0	0/0	0/0	0/0	**41114/30**
40	0/0	0/0	0/0	0/0	0/0	0/0	0/0	**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**	0/0	0/0	0/0	0/0	0/0	0/0	1987/29
20	0/0	0/0	0/0	0/0	0/0	0/0	0/0	**4077/39**
25	0/0	0/0	0/0	0/0	0/0	0/0	0/0	**8218/49**
30	0/0	0/0	0/0	0/0	0/0	0/0	0/0	**66287/59**
Cube2	24682/4	**10/3**	**10/3**	19/3	40/3	74/3	64/3	64/3
4	0/0	**434/9**	**434/9**	1486/9	2207/9	5021/9	2704/9	3671/9

Figure 4: This table shows the performance of the planning graph heuristics within $\mathcal{C}AltAlt$. All heuristics, unless otherwise indicated, use $w = 5$, and Θ_3, where applicable. Legend – [Search time (ms)/Plan length], "-": OoMemory or OoTime.

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 4 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 either based on single planning graphs and/or use Θ_2 to cost a belief state on a planning graph; 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	CGP	GPT	HSCP	KACMBP	HLevelMax	HRPUnion
Rover1	135/5	25960/5	3530/7	**100/5**	**129/5**	145/5
2	5788/7	-	10690/10	**700/13**	4751/8	**331/9**
3	-	-	10780/10	**710/13**	5985/10	**636/11**
4	-	-	30680/13	**800/22**	0/0	**15530/15**
Logistics1	64/8	69/9	40/26	**20/12**	**198/9**	321/11
2	1323/11	580/15	**120/26**	200/12	**2844/15**	6144/19
3	**168/10**	262/11	-	180/28	1973/14	**1723/17**
4	4559/14	4756/18	-	**210/28**	**9118/18**	164748/26
BiT2	**2/1**	42/2	20/2	10/2	**5/2**	11/2
10	63/1	234/10	30/10	**20/10**	5118/10	**462/10**
20	1032/1	-	**40/20**	**40/20**	0/0	**3380/20**
30	5492/1	-	**20/30**	50/30	0/0	**41114/30**
40	16005/1	-	**50/40**	80/40	0/0	**307590/40**
BiTC2	2/3	92/3	20/3	**10/3**	**11/3**	16/3
10	-	288/19	20/19	**10/19**	10016/19	**801/19**
15	-	34280/19	**30/19**	40/19	0/0	**1987/29**
20	-	-	**10/39**	80/39	0/0	**4077/39**
25	-	-	**20/49**	90/49	0/0	**8218/49**
30	-	-	**10/59**	140/59	0/0	**66287/59**
Cube2		42/3	**0/3**	10/3	**40/3**	64/3
4		63/9	**0/9**	**0/9**	**2207/9**	3671/9

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

However, h_{card} does perform better in the traditional conformant planning domains: BiT, $BiTC$, and $Cube$, 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 $S_j \in \xi(CS_i)$ in determining reachability estimates rather than the sum of unit costs of $\xi(CS_i)$. 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 4 also shows a comparison of $h_{RP_{max}}$ when using Θ_1 and Θ_3 to illustrate the speedup of not explicitly expanding $\xi(CS_i)$. The benefit of using Θ_3 appears in problems where there are many clauses in a clausal state, hence $\xi(CS_i)$ is large, such as in $Logistics$ and in $Cube$. 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), not all of which even use heuristics, it is hard to get a standardized comparison of heuristic effectiveness. Nevertheless, figure 5 compares CGP, GPT, HSCP, and KACMBP with $h_{level_{max}}$ and $h_{RP_{union}}$ 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 inoptimal with respect to plans generated by $\mathcal{C}AltAlt$ using $h_{level_{max}}$ and $h_{RP_{union}}$ in the $Rovers$ and $Logistics$ domains.

For $Rovers$, $h_{level_{max}}$ and $h_{RP_{union}}$ provide 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 more fertile means of comparison. The first lesson to be learned is that HSCP's cardinality heuristic, similar to h_{card}, does not scale well. Yet HCSP 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 planning graphs become less attractive. This issue is addressed in the following discussion.

The BiT and $BiTC$ 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.

For $Cube$, the planning graph heuristics seem to perform well with respect to the other planners in regards to search time, but have problems scaling to the level of other planners as the number of initial states increases exponentially. This problem is addressed in the followed discussion.

Heuristic Computation Cost: The advantage of our approach to computing heuristics for conformant planning with planning graphs is that we can give significant direction to conformant planners. Additionally, we've shown that conformant domains exist that necessitate such accurate estimates. However, in the current implementation, the cost of computing the multiple planning graph heuristics is still quite high. We are pursuing some very promising ideas for reducing this cost: (1) representing a subset of the initial states as planning graphs and (2) condensing the multiple planning graphs to one by using support labels on propositions. These improvements are based on the insight that since the planning graphs are being used as a basis for heuristics—rather than as a basis for search, as in the case of CGP—we can limit the amount of effort we expend in the heuristic computation, by trading off heuristic accuracy (c.f. [Nguyen *et al.*, 2002]). Choosing the right subset of initial states to use for building graphs seems tricky, but may be facilitated though identifying uncertainty dependencies between literals and only building graphs with the dependency sets. The multiple planning graphs can represent a large amount of redundancy when the possible worlds are quite similar. To avoid this, we are pursuing an idea called *labeled planning graphs*, which use 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. The labeled graphs would save time in planning graph construction and heuristic cost combination among worlds because we could have χ_2 as a direct estimate to get d_i. We are currently implementing and investigating these improvements to heuristic computation.

6 Related Work

The recent interest in conformant planning can be traced to CGP [Smith and Weld, 1998], a conformant version of Graphplan, where 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 *et al.*, 2001] and Frag-Plan [Kurien *et al.*, 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 and 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 and 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_{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 *et al.*, 2001] and KACMBP [Bertoli and Cimatti, 2002]. In comparison to $\mathcal{C}AltAlt$, the CMBP family of planners all represent belief states in terms of binary decision diagrams, and action application is modeled 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 pro-actively 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, validated by our current empirical study, 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 *et al.*, 2002], FF Hoffmann and Nebel [2001] and HSP-r Bonet and Geffner [1999].

7 Conclusion

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

1. Indicated the heuristic measures for conformant planning that should be estimated for accurate search control.
2. Shown how to compute such heuristic measures on planning graphs when using a clausal, factored representation of belief states.
3. Provided empirical comparisons of these measures.

We have also presented extensions to our work that are aimed at scaling conformant planning, through reduced heuristic computation, to consider even more complex domains.

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 NAG2-1461 and NCC-1225, and the NSF grant IRI-9801676.

References

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

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

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

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 *Artificial Intelligence Planning Systems*, pages 52–61, 2000.

Claudio Castellini, Enrico Giunchiglia, and Armando Tacchella. Improvements to sat-based conformant planning. In *6th European Conference on Planning*, 2001.

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

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

J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos. Extending planning graphs to an ADL subset. Technical Report report00088, IBM, 1, 1997.

James Kurien, P. Pandurang Nayak, and David E. Smith. Fragment-based conformant planning. In *Artificial Intelligence Planning Systems*, pages 153–162, 2002.

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. (submitted to the Journal of Artificial Intelligence).

David E. Smith and Daniel S. Weld. 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)*, pages 889–896, Menlo Park, July 26–30 1998. AAAI Press.