

Planning Graph Heuristics for Incomplete and Non-Deterministic Domains

Daniel Bryce

Department of Computer Science and Engineering
Ira A. Fulton School of Engineering
Arizona State University
Brickyard Suite 501
699 South Mill Avenue
Tempe, AZ 85281
dan.bryce@asu.edu

Abstract

This doctoral work centers on developing domain-independent heuristics for planning problems characterized by state incompleteness and action non-determinism. The key means by which the heuristics are computed is through planning graph analysis. The approach has been used to construct conformant and contingent plans in two different search techniques. The initial focus of the research was investigating useful single and multiple planning graph heuristic measures to estimate the conformant distance between belief states. The work showed that multiple planning graphs can accurately estimate belief state distance, but are costly. A subsequent contribution reduces the cost of multiple planning graph heuristics but maintains the accuracy through using a Labelled Uncertainty Graph (*LUG*). The *LUG* uses a single planning graph where graph elements are labelled with propositional formulas to indicate the states of the projected belief state for which the element is reachable. This innovation has proven to place the conformant planner *CAItAlt* in competition with current techniques in conformant planning. More recently, the *LUG* is showing its worth in deriving heuristics for a contingent planner *PBSP*. This abstract will describe the steps that I have taken in completing the aforementioned work and outline future prospects for my thesis.

Introduction

Recently there has been a lot of interest in improving the performance of planners that reason with incompleteness and non-determinism, such as the work exemplified by the following planners: *SGP* [Weld *et al.*, 1998], *GPT* [Bonet and Geffner, 2000], *MBP* [Bertoli and Cimatti, 2002], *CAItAlt* [Bryce and Kambhampati, 2004], and *CFP* [Brafman and Hoffmann, 2004]. Few of these modern approaches use the type of planning graph based heuristic analysis that has been leveraged so well in classical planners like *FF* [Hoffmann and Nebel, 2001] and *AltAlt* [Nguyen *et al.*, 2002].

My initial doctoral work has concentrated on adapting classical planning heuristics, derived from planning graphs, to the conformant planning setting. Conformant planning assumes state incompleteness, action non-determinism, and no observability. The approach taken was to compare the use of a single classical planning graph, multiple planning graphs

(one for each initial state, as with *CGP* [Smith and Weld, 1998]), and a labelled planning graph in deriving heuristics for regression search. The graph(s) were built once from the initial belief state and used to guide the regression search. The single planning graph proved to give little information for heuristics because it assumes too much independence between the states of the initial belief state. The multiple planning graphs proved to work better because they can help reason about achieving the goal from each of the initial states. The best heuristic derived from the multiple planning graphs extracted a relaxed plan from each graph and took into account the overlap (common actions) of the different relaxed plans. The work to this point is described in [Bryce and Kambhampati, 2004].

While the multiple graphs provided useful guidance to the search, they limited scalability because the number of graphs needed is exponential in the number of uncertain initial state variables. Thus, my work considered a new planning graph data structure, a labelled graph (*LUG*). The *LUG* captures the information of the multiple planning graphs within a single planning graph by pushing the disjunction of causal support into propositional labels. For instance, an action present in half of the multiple planning graphs would be represented once in the *LUG*, but have a label indicating which subset of the multiple planning graphs contain it. This helps reduce memory requirements and computation time for heuristics. Using a heuristic similar to the best multiple planning graph heuristic, but extracted from the *LUG*, places the *CAItAlt* planner in competition with the current best conformant planning approaches. This work is described in a pending publication [Bryce *et al.*, 2004].

The remainder of this paper will discuss in more detail the regression and progression search formulation, the planning graph heuristics, empirical results, and future work.

Search

Our formulation uses *A** regression search for conformant planning in *CAItAlt* and a modified *LAO** progression search for conformant and contingent planning in *PBSP* 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 and 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.

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.

Observation Representation: An observational action a , of the action set A , is described in terms of (i) an executability precondition ρ_e and (ii) a set of observational partition formulas O . Like causative actions, an observational action is only executable when its executability precondition is entailed by the current belief state. Each observational partition formula $o \in O$ defines the properties of a distinct outcome of the observation.

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:¹

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

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:

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

¹Note that $BS_{i'}$ may not be in clausal form after regression (especially when an action has multiple conditional effects).

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

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

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 .

Progression: We use progression in the *PBSP* planner to generate conformant and contingent plans with a modified LAO* search. The progression function, $\text{Progress}(BS_i, a)$, returns \perp if a is not applicable to BS_i , or a set of belief states B if a is applicable. For a result of \perp we do not add a new edge and node to the search graph. Otherwise, we add an And-edge for every node that represents a belief state $BS_i \in B$. “And”-edges, as in And-Or search, relate to hyperedges of a graph that must all be satisfied in a feasible solution. In our case, all “And”-edges of a node correspond to different outcomes of an observation; each outcome leads to belief state for which a path to a goal state must exist.

Since actions are either strictly causative or strictly observational, the *Progress* function either generates a single successor or partitions the states of BS_i into several belief states. When a is causative, we apply a to all states $S \in BS_i$ and take the disjunction of all resulting formulas as the result. The result of applying a to a state S is constructed by taking the conjunction of the consequents ε_j of effects φ_j of a where $S \models \rho_j$, then conjoining it with the persistence literals of S . When a is observational, *Progress* returns a set of successors B , where each is the conjunction of a distinct observational partition $o \in O$ with BS_i .

Progression planning in *PBSP* uses an algorithm similar to LAO* to find contingent plans with loops. LAO* [Hansen and Zilberstein, 2001] is a search algorithm that generalizes AO* to find plans with loops – corresponding to MDP policies. It has a major advantage over value or policy iteration, in that it visits only a subset of the state space. We would also like to find solutions with branches and loops, so we choose to build upon LAO*. Our modifications to LAO* are that we assume no stochastic actions (only non-deterministic), partial-observability and dead-end nodes (i.e. belief states that cannot reach goal belief states). With this modified LAO* we can find contingent plans that are general graphs, or policies for partially observable domains.

With partial-observability we cannot assume every state is fully-observable at every step of the plan, hence belief states (as opposed to states) are nodes in the search graph. Next, observational actions (as opposed to non-deterministic actions, as in LAO*) are associated with the “And”-edges in the search graph because we do not assume full observability. Lastly, non-deterministic actions generate belief states instead of splitting the resulting states into the search graph because we do not assume that we have full observational knowledge at all times.

LAO* operates on the assumption that every state can eventually reach a goal state, or that there exists at least one

proper policy. The domains of interest to us, and existing in real-life, rarely exhibit this property, so we need to use a different algorithm. Our approach uses a fix-point computation to label nodes in the solution graph as *not solved*, *partially solved*, or *solved* based on their successors, then if necessary checks the best solution graph to see if it represents a complete solution.

Initially all nodes are marked not solved, except for the node of the goal belief state. A node is solved if all successors of the best action’s “And”-edges are solved, and likewise for not solved nodes. A node is partially solved if (i) any successor of the best action’s “And”-edges are partially solved, or (ii) there exists a successor of the best action’s “And”-edges that is solved and another successor that is not solved. Essentially, a node is solved if all outcomes of the best action *will* lead to a goal state, it is partially solved if an outcome of the best action *can* reach a goal state, and not solved otherwise. The fix-point computation is initiated after expanding a new node as part of the backup procedure. If at the end of the fix-point computation the initial belief state’s node is labelled solved, then we can terminate with a correct solution, and if it is labelled as partially solved then we need to check the best solution graph to make sure it is correct.

The reason for checking the solution when the initial belief state node is partially solved is that the solution may have an unsolved node (unacceptable), or a partially solved node that returns through a cycle to another partially solved node (acceptable). The solution checking phase recursively walks the best solution graph, starting at the initial belief state node. The recursion terminates with success if every path reaches a solved node, or re-encounters another partially solved node. The recursion returns unsuccessfully when an unsolved node is encountered.

Planning Graph Heuristics

Single Planning Graph (SG) 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. 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 prevent identification of consistent states because the graph is built from an inconsistent union of literals.

Multiple Planning Graphs (MG) 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. 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 $\xi(BS_I)$.

Here I mention the best performing heuristic, see [Bryce and Kambhampati, 2004] for the full set.

RP-union (h_{RP}^{MG}): In order to get the union relaxed plan, first a relaxed plan is computed for each graph $\gamma_k \in \Gamma$. Re-

laxed plans extracted from each of the multiple graphs are different, but hold some similarities. This information can be leveraged to account for the interaction or overlap. Starting from the last action level 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. 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.

Labelled Uncertainty Graph (LUG) We improve the multiple graph approach by addressing its limitations. The idea is to condense the previously used multiple planning graphs to a single planning graph. Loosely speaking, this single graph unions the causal support information present in the multiple graphs and pushes the disjunction, describing sets of worlds, into “labels” (ℓ). The graph elements are the same as those present in multiple graphs, but represented only once. For instance an action that was present in all of the multiple planning graphs would be present only once in the *LUG* and labelled to indicate if it is applicable in a projection from each initial world.

This single Labelled Uncertainty Graph, *LUG*, adds labels to graph elements to symbolically represent which projections through the graph relate to which initial states. The labelled elements are each action a , each conditional and unconditional effect relation φ of an action, each literal l , and each mutex relation² of the graph. In general, a label is an arbitrary propositional formula describing a set of initial states for which a graph element is reachable. The way the *LUG* is constructed to contain the multiple world causal support information, as present in *MG*, is to label the single set of initial literals with their initial worlds and propagate these labels through actions as the graph is built. Construction ends when the goal belief state can be satisfied by literals present in a graph level and the literals are labelled to indicate that the goal belief state is reachable from all possible worlds, i.e. the goal is fully-supported. A variety of planning graph heuristics can be adapted for the *LUG*.

The propagation of labels is based on the intuition that (i) actions and effects are applicable in the possible worlds specified by the conjunction of the precondition formula’s labels and (ii) a literal is supported in possible worlds specified by the disjunction of labels of effect relations that support the literal.

²Mutexes are not described here for lack of space, but are instrumental in the performance of the *LUG*. We have investigated several schemes for computing mutexes; the most expressive is similar to CGP [Smith and Weld, 1998]. The scheme presented in the results, (D-S), refers to computing only same world mutexes.

Problem	CAltAlt h_{RPU}^{MG}	CAltAlt $h_{RP}^{LUG(D-S)}$	PBSP h_{RP-ha}^{LUG}	MBP	KACMBP	HSCP	GPT	CGP	SGP
Rov1	185/5	16070/5	230/5	66/5	9293/5	-	3139/5	70/5	70/5
2	29285/9	10457/8	1370/12	141/8	9289/15	-	4365/8	180/8	30/8
3	2244/11	10828/10	12920/17	484/10	9293/16	-	5842/10	460/10	1750/10
4	3285/15	15279/13	81280/24	3252/13	9371/18	-	7393/13	1860/13	-
5	-	64870/29	-	-	39773/40	-	399525/20	-	-
6	-	221051/25	-	727/32	-	-	-	-	-
Log1	1109/9	907/9	610/11	37/9	127/12	352/9	916/9	60/6	70/6
2	69818/19	2862/15	8570/21	486/24	451/19	-	1297/15	290/6	510/6
3	70882/14	10810/15	24780/20	408/14	1578/18	-	1711/11	400/8	4620/8
4	-	24862/19	-	2881/27	8865/22	-	9828/18	1170/8	44740/8
5	-	54726/34	-	-	226986/42	-	543865/28	-	-
BT2	21/2	16/2	0/2	6/2	10/2	8/2	487/2	20/1	0/1
10	342/10	71/10	130/10	119/10	16/10	6/10	627/10	520/1	30/1
20	2299/20	552/20	1550/20	-	84/20	23/20	472174/20	3200/1	290/1
30	9116/30	2415/30	7170/30	-	244/30	47/30	-	10330/1	1170/1
40	44741/40	7543/40	21040/40	-	533/40	80/40	-	24630/1	3320/1
50	-	17573/50	54160/50	-	1090/50	148/50	-	49329/1	7550/1
60	-	35983/60	122110/60	-	2123/60	340/60	-	87970/1	83494/1
70	-	67690/70	-	-	3529/70	-	-	145270/1	114340/1
80	-	157655/80	-	-	-	-	-	-	-
BTC2	23/3	16/3	0/3	8/3	18/3	2/3	465/3	0/3	0/3
10	614/19	89/19	230/19	504/19	45/19	25/19	715/19	39370/19	-
20	2652/39	651/39	2840/39	98/39	211/39	98/39	-	-	-
30	9352/59	2721/59	12180/59	268/59	635/59	293/59	-	-	-
40	51859/79	8009/79	36750/79	615/79	1498/79	674/79	-	-	-
50	-	19074/99	95130/99	1287/99	10821/99	1352/99	-	-	-
60	-	38393/119	-	2223/119	5506/119	5100/119	-	-	-
70	-	-	-	3625/139	9334/139	-	-	-	-

Figure 1: Results for CAltAlt using h_{RPU}^{MG} and $h_{RP}^{LUG(D-S)}$, PBSP using h_{RP-ha}^{LUG} , MBP, KACMBP, HSCP, GPT, CGP, and SGP for conformant Rovers, Logistics, BT, and BTC. The data is Total Time / # Plan Steps, “-” indicates no solution.

During the *LUG* construction, a common operation is to determine if a formula is fully-supported (i.e. *all* possible world projections can achieve a belief state that entails the formula), or supported (i.e. *there exists* a possible world whose projection can achieve a belief state that entails the formula). A formula f is **fully-supported** (FSp(f, k)) at level k when for all possible worlds S of BS_I f is supported by S .³ A formula f is **supported** (Sp(f, k, S)) at level k by a possible world S of BS_I when the labels of the literals in f indicate support by S . The labels of literals indicate support by S when the formulas’ literals are substituted with their labels and S entails the substituted formula. To ease the definition, we consider a canonical form for f , namely a CNF, \mathcal{C} , that is supported when:

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

Here $\ell_k(l)$ is the label of the literal l at level k . Please note, full-support can be easily checked for all possible worlds at once by replacing S with BS_I . Also note, the *LUG* is an approximation to the belief space projection from the belief state BS_I , so when we refer to something as supported we mean *possibly* supported.

We now describe label propagation, first by showing how to construct the initial literal layer \mathcal{L}_0 of the graph, and then showing how a graph level $\{\mathcal{A}_k, \mathcal{E}_k, \mathcal{L}_{k+1}\}$ is built.

$\mathcal{L}_0 \leftarrow \text{insertInitialLiterals}(BS_I)$:

The initial literal layer \mathcal{L}_0 contains all literals l in the possible worlds of the belief state BS_I . Each literal l is labelled $\ell_0(l)$ to indicate the set of possible worlds where it holds. The

³The notion of fully-supported is a generalization of the level heuristic for classical planning [Nguyen *et al.*, 2002]. It is also similar to the max heuristic used in GPT [Bonet and Geffner, 2000].

label of literal l is found by conjoining l with the formula for BS_I .

$\mathcal{A}_k \leftarrow \text{insertActions}(\mathcal{L}_k)$:

Once the previous literal layer \mathcal{L}_k is computed, we compute the labelled action layer \mathcal{A}_k . \mathcal{A}_k is defined as all applicable actions from the action set A , plus all literal persistence $\diamond l$.⁴ An action’s executability precondition must be supported for some possible world at level k for the action to be applicable. If applicable, the action’s label at level k , using a CNF for the formula of ρ_e , is:

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

$\mathcal{E}_k \leftarrow \text{insertEffects}(\mathcal{L}_k, \mathcal{A}_k)$:

The labelled effect relations \mathcal{E}_k depend both on the literal layer \mathcal{L}_k and action layer \mathcal{A}_k . \mathcal{E}_k is the set of applicable labelled effect relations at level k . An effect relation is applicable when the associated action is applicable and the antecedent of the effect is supported. The label is the conjunction of the label of the associated action with the labels of the formula of the effect’s antecedent. The label of an effect i at level k , using a CNF for the formula of ρ_i , is:

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

$\mathcal{L}_k \leftarrow \text{insertLiterals}(\mathcal{E}_{k-1}), k > 0$:

The literal layer at k is the set of labelled literals that are added by consequents of effects in \mathcal{E}_{k-1} .

A literal is added to the literal layer if it is present in the formula of a consequent of an effect in the previous layer. The label of a literal, $\ell_k(l)$, is the disjunction of the labels

⁴Persistence for a literal l is represented as an action where $\rho_e = \varepsilon_0 = l$.

Problem	PBSP h_{RP-ha}^{LUG}	MBP	GPT	SGP
Rov1	240/5	3328/11	3148/5	70/5
2	620/7	7066/57	5334/7	760/7
3	990/8	4818/61	7434/8	-
4	1490/10	4939/57	11430/10	-
5	38640/20	1389/81	-	-
6	-	3145/158	-	-
Log1	220/7	31/17	1023/7	5490/6
2	2460/13	-	5348/12	-
3	2110/12	2120/45	2010/8	-
4	26820/26	1086/650	-	-
5	-	5768/1977	-	-
BT2	0/2	8/3	510/2	0/1
10	200/10	12/19	155314/10	70/1
20	2090/20	78/39	-	950/1
30	9060/30	305/59	-	4470/1
40	26420/40	887/79	-	13420/1
50	66320/50	2350/99	-	32160/1
60	-	4479/119	-	90407/1
70	-	-	-	120010/1
80	-	-	-	-
BTC2	0/2	10/3	529/2	10/2
10	310/10	53/19	213277/10	-
20	3400/20	518/39	-	-
30	14100/30	2551/59	-	-
40	42580/40	8475/79	-	-
50	-	20684/99	-	-
60	-	47375/119	-	-
70	-	-	-	-
Omelette1	130	-	250	-
2	-	-	4297	-
3	-	-	12866	-
Medical1	80/1	12/3	506/1	0/1
2	-	12/3	514/3	20/3
3	-	6/3	541/5	70/3
4	-	12/4	531/5	630/3
5	-	16/4	547/5	4550/3

Figure 2: Results for *PBSP* using h_{RP-ha}^{LUG} , MBP, GPT, and SGP for contingent Rovers, Logistics, BT, and BTC. The data is Total Time / # Max possible steps in a execution, “-” indicates no solution.

of each effect that has the literal in its consequent’s formula, using a CNF for the formula of ε_i , it is:

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

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

Label Relaxed Plan h_{RP}^{LUG} : The heuristic we extract from the *LUG* is similar to the h_{RPU}^{MG} heuristic for multiple planning graphs. This heuristic is similar in that it counts actions that are applicable in multiple worlds only once per step. The advantage is that we find these actions by looking at only one labelled planning graph. The relaxed plan is representative of a conformant plan because at each level of the graph we ensure that the chosen actions will support the subgoals from all possible worlds. This is useful in guiding the search towards plans with lower overall plan length and higher world overlap. We can also use helpful actions [Hoffmann and Nebel, 2001] from the relaxed plan for the h_{RP-ha}^{LUG} heuristic search.

Empirical Results

Figure 1 shows results for using h_{RPU}^{MG} and $h_{RP}^{LUG(D-S)}$ in *CAItAlt* and h_{RP-ha}^{LUG} in *PBSP* in comparison to six other

conformant planners: MBP, KACMBP, HSCP, GPT, CGP, and SGP. The *LUG* with same-world mutexes is used in *CAItAlt* to derive a relaxed plan heuristic. The results show that the *LUG* improves scalability over *MG*, the *LUG* is most beneficial in regression search because it is built only once, and the *LUG* places *CAItAlt* in competition with several state of the art conformant planners.

Figure 2 shows the results for *PBSP* in comparison to three other state of the art contingent planners: MBP, GPT, and SGP. The *PBSP* planner using h_{RP-ha}^{LUG} is competitive with optimal planners (GPT and SGP) by finding good quality plans faster, and a heuristic planner (MBP) by finding much better quality plans in comparable time.

Future Work

In the longer term, I intend to consider how the *LUG* heuristics can be adapted to scaling stochastic planning. The basis for scaling stochastic planning will rely on using the *LUG* heuristics to create a seed non-deterministic plan that abstracts probabilities, then performing local search on the seed plan, accounting for probabilities, to improve the plan’s probability of success. My doctoral thesis will tie together all of these results to present a heuristic approach to planning in incomplete, non-deterministic, and stochastic domains.

References

- Piergiorgio Bertoli and Alessandro Cimatti. Improving heuristics for planning as search in belief space. In *Artificial Intelligence Planning Systems*, pages 143–152, 2002.
- Blai Bonet and Hector Geffner. Planning with incomplete information as heuristic search in belief space. In *Artificial Intelligence Planning Systems*, pages 52–61, 2000.
- Ronen Brafman and Joerg Hoffmann. Conformant planning via heuristic forward search: A new approach. In *ICAPS04*, 2004.
- Daniel Bryce and Subbarao Kambhampati. Heuristic guidance measures for conformant planning. In *ICAPS-04*, 2004.
- Daniel Bryce, Subbarao Kambhampati, and David Smith. Planning graph heuristics for belief space search. *Journal of Artificial Intelligence Research*, 2004. Forthcoming.
- Eric A. Hansen and Shlomo Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62, 2001.
- Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.
- 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.
- Daniel S. Weld, Corin Anderson, and David E. Smith. Extending graphplan to handle uncertainty and sensing actions. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-98)*. AAAI Press, 1998.