The Causal Graph Heuristic is the Additive Heuristic plus Context

Héctor Geffner
ICREA & Universitat Pompeu Fabra
Passeig de Circumvalació 8
08003 Barcelona Spain
hector.geffner@upf.edu

Introduction

The causal graph heuristic introduced by Helmert in (Helmert 2004; 2006b) is one of the most interesting recent developments in the heuristic search approach to domain-independent planning. Unlike the additive heuristic used early in the HSP planner (Bonet & Geffner 2001) and the relaxed planning graph heuristic used in FF (Hoffmann & Nebel 2001), the causal graph heuristic is not based on the *delete-relaxation* but on a deeper analysis of the problem structure as captured by its underlying *causal graph*. The causal graph is a directed graph where the nodes stand for the variables in the problem and links express the dependencies among them. The causal graph heuristic is defined for problems with *acyclic* causal graphs as the sum of the costs of plans for subproblems that include a variable and its parents in the graph. The local costs are not optimal (else would be intractable) and are defined *procedurally*.

In this note we introduce an alternative, declarative formulation of the causal graph heuristic that we believe is simpler and more general. The new heuristic reduces to Helmert's heuristic when the causal graph is acyclic, but *does not require either acyclicity nor the causal graph itself.* Like the additive heuristic, the new heuristic is defined mathematically by means of a functional equation, which translates into a shortest path-problem over a poly-size graph that can be solved by standard algorithms. Indeed, the only difference between this account of the causal graph heuristic and the normal additive heuristic is that the nodes in this graph, that stand for the atoms in the problem, are labeled with contextual information. The new formulation of the causal graph heuristic suggests a number of extensions, all of which have to do with the exploitation of implicit or explicit precedences among the actions preconditions in order to capture side-effects in the computation of the heuristic.

Multivalued Planning Tasks

The causal graph heuristic is defined over a planning language with multivalued variables based on the SAS+ language (Bäckström & Nebel 1995), where the basic atoms are of the form $v = d$ where v is a variable and $d \in D_v$ is a value in v's domain D_v.

Formally, a *multivalued-planning task* (MPTs) is a tuple $\Pi = \langle V, s_0, s_\star, O, \rangle$ where V is a set of variables v with associated finite discrete domains D_v, s_0 is a state over V characterizing the initial situation, $s_\star$ is a partial state over V characterizing the goal situation, and O is a set of operators that map one state into a possibly different state.

A state is a function s that maps each variable $v \in V$ into a value $s(v)$ in D_v. A partial state is one such function but restricted to a subset $V' \subseteq V$ of variables. As it is common in the boolean setting, we often represent and treat such functions as the *set of atoms* $v = d$ that they make true, while keeping in mind that the set of atoms that represent the state $s/v = d'$ obtained from s by changing the value of variable v from d to d', contains the atom $v = d'$ but not $v = d$.

An operator a has a precondition $pre(a)$ that is a partial state, and a set of effects or rules $z \rightarrow v = d$, written also as $a : z \rightarrow v = d$, where the condition z is a partial state, $v \in V$ is a variable, and d is a value in D_v.

An action a is executable in a state s if $pre(a) \subseteq s$ and the result is a state s' that is like s except that variables v are mapped into values d when $a : z \rightarrow v = d$ is an effect of a and $z \subseteq s$. A *plan* is a sequence of applicable actions that maps the initial state s_0 into a final state s_G where $s_\star$ holds.

These definitions follow the ones in (Helmert 2006b) with a few simplifications (e.g., axioms and derived atoms are not considered). In addition, for simplicity and without loss of generality, we make two assumptions. First, we will assume that *action preconditions* $pre(a)$ *are empty*. This is because, the value of the causal graph heuristic does not change when preconditions $p \in pre(a)$ are moved into the body z of all effects $z \rightarrow v = d$. Second, we will assume that the variable v that appears in the head of a rule $z \rightarrow v = d$ also appears in the body z. Effects $z \rightarrow v = d$ for which this is not true are to be replaced by a collection of effects $v = d', z \rightarrow v = d$, one for each value $d' \in D_v$ different than d; a transformation that preserves the semantics and complies with the above condition. Effects $z \rightarrow v = d$ can thus all be written as

$$v = d', z' \rightarrow v = d\ .$$

While this language is not the standard in planning, an automatic translator from PDDL into MPT's is described in (Helmert 2006a).

The Causal Graph Heuristic

The **causal graph heuristic** $h_{cg}(s)$ provides an estimate of the number of actions needed to reach the goal from a state s in terms of the estimated costs of changing the value of each variable v that appears in the goal from its value v_s in s to its value $v_\star$ in the goal:

$$h_{cg}(s) = \sum_{v \in s_\star} cost_v(v_s, v_\star) \quad (1)$$

The **costs** $cost_v(d, d')$ are defined with the help of two structures: the *domain transition graphs* $DTG(v)$, that reveal the structure of the domain D_v associated with each variable v, and the *causal graph* $CG(\Pi)$ that reveals the relation among the variables v in the problem Π.

The **domain transition graph** $DTG(v)$ for a variable $v \in V$, is a labelled directed graph with vertex set D_v and edges (d, d') labelled with the condition z for rules $v = d, z \rightarrow v = d'$ in Π.

The **causal graph** $CG(\Pi)$ for $\Pi = \langle V, s_0, s_\star, O\rangle$, is the directed graph with vertex set V and arcs (v, v') for $v \neq v'$ such that v appears in the label of some arc in $DTG(v')$ or some action a affects both v and v' (i.e., Π contains effects $a : z \rightarrow v = d$ and $a : z' \rightarrow v' = d'$ for an action a).

The costs $cost_v(d, d')$ that determine the heuristic $h_{cg}(s)$ in Equation 1 are defined in terms of the causal graph $CG(\Pi)$ and the domains transition graphs $DTG(v)$.

The definition assumes that $CG(\Pi)$ is *acyclic*. When this is not so, Helmert's planner 'relaxes' the causal graph by deleting some edges, defining the costs and the resulting heuristic over the resulting acyclic graph.

The measures $cost_v(d, d')$ stand for the cost of *a plan* π that solves the subproblem $\Pi_{v,d,d'}$ with initial state $s/v = d$, goal $v = d'$, that involves only the variable v and its parent variables in the causal graph. The measures $cost_v(d, d')$ however do not stand for the *optimal costs* of these subproblems which are not tractable (Helmert 2006b), and are defined *procedurally* using a slightly modified Dikjstra's algorithm (Cormen, Leiserson, & Rivest 1989; Bertsekas 1991), in topological order, starting with the variables with no parents (root variables) in the causal graph.[1]

I will not repeat the exact procedure for computing these costs, that can be found in Figure 18, (Helmert 2006b), but rather I will explain the procedure in a way that will make the relationship between the causal graph and the additive heuristics more direct.

Let us recall first that in **Dijkstra's algorithm**, a label $c(i)$ is associated with all nodes i in the graph, initialized to $c(i) = 0$ if i is the source node and $c(i) = \infty$ otherwise. In addition, an OPEN list is initialized with all nodes. The algorithm then picks and removes the node i from OPEN with least cost $c(i)$ iteratively, updating the values $c(j)$ of all the nodes j still in OPEN to $c(j) = \min(c(j), c(i) + c(i, j))$, where $c(i, j)$ is the cost of the edge connecting node i to j in the graph. This is called the *expansion* of node i.

[1]Actually Helmert's procedure does not compute the costs $cost_v(d, d')$ for all $v \in V$ and all d, d' in D_v but this optimization is not relevant here.

The algorithm finishes in a number of iterations bounded by the number of nodes in the graph when OPEN is empty. The label $c(i)$ of a node is optimal when selected for expansion and remains so until termination.

The cost $c(i, j)$ of the directed edges (i, j) is assumed to be non-negative and is used in the computation *only* right after node i is expanded. Helmert's procedure takes advantage of this fact for *setting the cost of such edges dynamically, right after node i is selected for expansion.*

Actually, when v is a root variable in $CG(\Pi)$, **Helmert's procedure** for solving the subproblem $\Pi_{v,d,d'}$, for a given $d \in D_v$ and all $d' \in D_v$, resulting in the costs $cost_v(d, d')$, is exactly Dikjstra's: the graph is $DTG(v)$, the source node is d, the cost of all edges is set to 1, and upon completion, $cost_v(d, d')$ is set to $c(d')$ for all $d' \in D_v$. For such variables, $cost_v(d, d')$ is indeed optimal for $\Pi_{v,d,d'}$.

For variables v with a non-empty set of parent variables v_i in the causal graph with costs $cost_{v_i}(e_i, e'_i)$ already computed for all $e_i, e'_i \in D_{v_i}$, Helmert's procedure for solving the subproblem $\Pi_{v,d,d'}$ for a given d and all $d' \in D_v$, modifies Dijkstra's slightly by setting the cost of the edges $(d_1, d_2) \in DTG(v)$ labelled with condition $z : v_1 = e_1, \ldots, v_n = e_n$ to

$$1 + \sum_{i=1,\ldots,n} cost_{v_i}(e'_i, e_i) \quad (2)$$

right after node d_1 has been selected for expansion, where e'_i is the value of variable v_i in the state s_{d_1} associated with node d_1. The state associated with a node d is defined as follows. The state s_d associated with the source node d is s, while if s_d is the state associated with the node d just selected for expansion, and the expansion of d causes $c(d')$ to decrease for some $d' \in D_v$ due to an edge (d, d') labelled with z, then $s_{d'}$ is set to s_d/z.

This procedure for solving the subproblems $\Pi_{v,d,d'}$ is not optimal nor is it complete. Indeed, it is possible for the costs $cost_v(d, d')$ to be infinite while the costs of $\Pi_{v,d,d'}$ are finite. This is because the procedure achieves the intermediate values d'' *greedily*, carrying the *side effects* $s_{d''}$ but ignoring their impact in the 'future' costs to be paid in going from d'' to d'.

These limitations of the causal graph heuristic are known, what we seek here is an understanding of the heuristic in terms of the simpler and declarative additive heuristic.

Additive Heuristic

For the language of the MPT's $\Pi = \langle V, s_0, s_\star, O\rangle$, the additive heuristic (Bonet, Loerincs, & Geffner 1997; Bonet & Geffner 2001) can be expressed as:

$$h_a(s) \stackrel{\text{def}}{=} \sum_{x \in s_\star} h(x|s) \quad (3)$$

where x stands for the atoms $v = d$ for $v \in V$ and $d \in D_v$, and $h(x|s)$ is an estimate of the cost of achieving x from s given as:

$$h(x|s) \stackrel{\text{def}}{=} \begin{cases} 0 \text{ if } x \in s, \text{ else} \\ \min_{a:z \rightarrow x} [1 + \sum_{x_i \in z} h(x_i|s)] \end{cases} \quad (4)$$

This functional equation approximates the true cost function by assuming that the cost of joint conditions (in goals and effects) is additive. Such sums go away, however, if the goal is a single atom and no condition z features more than one atom. In such a case, the additive heuristic $h_a(s)$ coincides with the max heuristic $h_{max}(s)$ (Bonet & Geffner 2001), and both are optimal provided that all these conditions are mutex.

It follows from this that if $x_1, \ldots, x_n$ represent the atoms $v = d_1, \ldots, v = d_n$ for a root variable v in the causal graph of Π, the values $h(x_i|s)$ that follow from (4), for $i = 1, \ldots, n$, are all optimal, and thus in correspondence with the costs $cost_v(d_k, d_i)$ computed by Helmert's procedure when d_k is the value of v in s. For other values d_j of v, the costs $cost_v(d_j, d_i)$ are equivalent to the estimates $h(x_i|s_j)$ with $s_j = s/[v = d_j]$.

This correspondence between the costs $cost_v(d', d)$ and the heuristics $h(x|s')$ when v is a root variable in the causal graph, x is $v = d$, and s' is $s/v = d'$, raises the question of whether such costs can be chacterized in the style of the additive heuristic also when v is *not* a root variable.

Notice first that Equation 2 used in Helmert's procedure, is additive. At the same time, the procedure keeps track of *side effects* in a way that is not captured by (4) where the costs $h(x_i|s)$ of all conditions x_i in the body of the rules $a : z \rightarrow x$ are evaluated with respect to the same state s. This however suggests to look at variations of (4) where these conditions x_i are evaluated in different states s_i rendering the general pattern

$$h(x|s) \overset{\text{def}}{=} \begin{cases} 0 \;\text{ if } x \in s, \text{else} \\ \min_{\,a:z\rightarrow x} [1 + \sum_{x_i \in z} h(x_i|s_i)] \end{cases} \qquad (5)$$

where the states s_i over which some of the conditions x_i in a rule are evaluated may be different than the seed state s where the value of the heuristic needs to be assessed. We will refer to such states s_i that may be different than the seed state as *contexts*.

Additive h with Contexts

The use of (5) in place of (4) for defining the additive heuristic raises two questions:

1. how to choose the *contexts* s_i needed for the evaluating the conditions x_i in a given rule $a : z \rightarrow x$, and
2. how to restrict the number of total contexts s_i needed for computing the heuristic value of s.

We answer these two questions in line with Helmert's formulation. Later on we will consider some generalizations.

Let us recall first that, without loss of generality, we are assuming that all the rules have the form $a : x, z \rightarrow x'$ where x and x' are atoms referring to the same (multi)variable v in the problem (Section 2). An atom x is a v-atom when it has the form $v = d$ for some $d \in D_v$, and refers to the same variable as an atom x' when the two are v-atoms for some v. We also say that $x' : v = d'$ is the *value* of x in the state s' as an abbreviation for saying that x' and x refer to the same variable v, and d' is the value of v in s'. As before, s/s' when s is a state and s' is partial state, refers to the state that is like s except over the variables mentioned in s' where it is equal to s'.

The answer to the first question that follows from Helmert's formulation is that in the rules $a : x', z \rightarrow x$ *the condition x' that refers to the same variable as x, is achieved first, and the rest of the conditions x_i in z are evaluated in the state s' that results* (**Assumption 1**).

The answer to the second question is that in the computation of the heuristic for a state s, the costs $h(x_i|s')$ for a context s' are mapped into costs $h(x_i|s/x'_i)$ where x'_i is the value of x_i in s', meaning that information in the state s' about other variables is discarded (**Assumption 2**). We will write $h(x_i|s/x'_i)$ then as $h(x_i|x'_i)$.

Provided with these two assumptions, the idea underlying (5) can be formalized as follows:

$$h(x''|x') \overset{\text{def}}{=} \begin{cases} 0 \quad \text{ if } x'' = x', \text{else} \\ \min_{a:x,z\rightarrow x''} [1 + h(x|x') + \sum_{x_i \in z} h(x_i|x'_i)] \end{cases} \qquad (6)$$

where x'_i is the value of x_i in the state that results from achieving x from x', written as $s(x|x')$ and obtained from

$$s(x''|x') \overset{\text{def}}{=} \begin{cases} s/x' \quad \text{ if } x'' = x', \text{else} \\ s(x|x')/z, x'', y_1, \ldots, y_n \end{cases} \qquad (7)$$

where $a : x, z \rightarrow x''$ is the rule that yields the minimum in (6), and $a : x, z \rightarrow y_i$, $i = 1, \ldots, n$, are other rules in the problem with the same action and body (thus when the former is applied, the latter are applied as well).

In words, when $a : x, z \rightarrow x''$ is the best (min) support for atom x'' from x' according to (6), and $s(x|x')$ is the state that is deemed to result for achieving x from x' (i.e., the 'side-effect' of achieving x from x'), then $s(x|x')/z, x'', y_1, \ldots, y_n$ is the state that is deemed to result for achieving x'' from x'.

Equations 6 and 7 define an heuristic that is very much like the additive heuristic except that a) the heuristic values $h(x''|s)$ are computed not only for the seed state s but for all the states $s' = s/x'$ when x' is an atom that refers to the same variable as x'', and b) during the computation, the preconditions other than x in the rules $a : x, z \rightarrow x''$ are evaluated in the state $s(x|x')$ that results from achieving the 'pivot' condition x. producing then the side effect $s(x''|x')$ associated with x''. This recursion starts with $x'' = x'$ when $h(x''|x') = 0$ and $s(x''|x') = s/x'$ (the seed state s updated with x').

The *context-enhanced* additive heuristic $h^c_a(s)$ is defined then as

$$h^c_a(s) \overset{\text{def}}{=} \sum_{x \in s_\star} h(x|x_s) \qquad (8)$$

where x_s is the value of x in s and $h(x|x')$ is defined by (6)–(7).

Example

As an illustration, consider a problem with a boolean variable Y and a multivalued variable $X \in [0 \ldots n]$, represented by the booleans y, $\neg y$, and x_i, standing for the assigments

$Y = true$, $Y = false$, and $X = i$ for $i = 0, \ldots n$, and actions a and b_i, $i = 0, \ldots, n$ with rules

$$a : \neg y \rightarrow y \quad , \quad b_i : x_i, y \rightarrow x_{i+1} \wedge \neg y$$

where $z \rightarrow x \wedge y$ stands for $a : z \rightarrow x$ and $a : z \rightarrow y$. We want to determine the value of $h_a^c(s)$ when x_0 and $\neg y$ hold in s and the goal is x_n. From (8), $h_a^c(s) = h(x_n|x_0)$. The optimal plan for the problem is the action sequence $a, b_0, a, b_1, \ldots, a, b_{n-1}$ for a cost of $2n$. The plans must increase the value of X one by one, but before each step the value of Y that is made false by each increase in X, must be restored to true.

From (6), it follows that

$$h(x_{i+1}|x_0) = 1 + h(x_i|x_0) + h(y|y') \tag{9}$$

for $i = 0, \ldots, n-1$, where y' represents the value of Y in the state $s(x_i|x_0)$ that results from achieving x_i from x_0 characterized as:

$$s(x_{i+1}|x_0) = s(x_i|x_0)/x_{i+1}, \neg y \; .$$

The relevant 'border' conditions are $h(x_0|x_0) = 0$ and $s(x_0|x_0) = s$. Clearly y' above is $\neg y$, as $\neg y$ holds in $s(x_i|x_0)$ for all $i = 0, 1, \ldots$, so that (9) becomes:

$$h(x_{i+1}|x_0) = 1 + h(x_i|x_0) + 1$$

for all $i > 0$ as $h(\neg y|y) = 1$. Thus, $h(x_{i+1}|x_0) = 2 + h(x_i|x_0)$ with $h(x_0|x_0) = 0$. So $h(x_n|x_0) = 2n$ and thus $h_a^c(s)$ is optimal. The plain additive heuristic, on the other hand, is $h_a(s) = n+1$, which is optimal only for the delete-relaxation.

Causal Graph and Additive Heuristics

The causal graph underlying the problem above involves a cycle between the two variables X and Y. In the absence of such cycles the following correspondence can be shown:[2]

Theorem 1 (Causal and Additive Heuristics) *If the causal graph $CG(\Pi)$ is* acyclic, *then the causal graph heuristic h_{cg} and the context-enhanced additive heuristic h_a^c are equivalent, i.e., for every state s, $h_{cg}(s) = h_a^c(s)$.*

The sketch of the proof proceeds as follows. When $CG(\Pi)$ is acyclic, a correspondence can be established between the costs $cost_v(d, d')$ defined by Helmert's procedure and the costs $h(x'|x)$ defined by Equations 6–7 for $x : v = d$ and $x' : v = d'$, and between the states $s_{d'}$ associated with the node d' and the states $s(x'|x)$ defined by 6 and 7.

These correspondences must be proved inductively: first on the root variables of the causal graph, and then on the execution of the modified Dijkstra's procedure.

The first part is straightforward and was mentioned above: if v is a root variable, $cost_v(d, d')$ is the optimal cost of the subproblem $\Pi_{v,d,d'}$ which involves no other variables, and the costs $h(x'|x)$ remain optimal as well, as there is a single condition in every rule and hence no sums or additivity assumptions, and all these conditions are mutex. At the same time, the states $s_{d'}$ and $s(x'|x)$ remain equivalent too.

If v is not a root variable, the correspondence between $cost_v(d, d')$ and s'_d on the one hand, and $h(x'|x)$ and $s(x'|x)$ on the other, must hold as well for $d' = d$ before any node is expanded in Helmert's procedure. Assuming inductively that the correspondence holds also for all values $e_i, e'_i \in D_{v_i}$ of all ancestors v_i of v in the causal graph, and for all values $cost_v(d, d')$ and states $s_{d'}$ after the first i-nodes have been expanded, it must be shown that the correspondence holds after the $i+1$-node is expanded as well.

[2]The correspondence assumes that edges in domain transition grapahs and rules in the problem are ordered statically in the same way, so that ties in Helmert's procedure and in (6-7) are broken in the same way. Note that there is a direct correspondence between edges (d, d') labelled with conditions z in $DTG(v)$ and rules $v = d, z \rightarrow v = d'$.

Computing the Heuristic for Cyclic Graphs

The extended additive heuristic h_a^c reduces to the causal graph heuristic h_{cg} in MPT's with acyclic causal graphs, but does not require acyclicity, and indeed, does not use the notion of causal graphs or domain transition graphs. In this sense, the induction over the causal graph for defining the costs $cost_v(d, d')$, from variables to their descendants, is replaced in h_a^c by an implicit induction over costs.

Indeed, in the presence of a cyclic graph, the costs $h(x|x')$ in (6)–(7) can be computed using a modified Dijkstra's algorithm, similar to Helmert's but that works over all domain transition graphs $DTG(v)$ and possible sources $d \in D_v$ at the same time. More precisely, the nodes in the graph would correspond to the transitions $x|x'$, for all atom pairs x and x' referring to the same variable. Initially the costs $h(x|x)$ are zero and all other transition costs are infinite, and in each step the transition from OPEN with least cost is selected and its cost and state are propagated as in the modified Dijkstra's procedure. The algorithm remains polynomial, and indeed, Dijkstra's algorithm can be used for computing the *normal additive heuristics*, whether the causal graph is cyclic or not, although in practice the Bellman-Ford algorithm is preferred (Liu, Koenig, & Furcy 2002).

Generalizations

The context-enhanced heuristic h_a^c is more general than the causal graph heuristic as it applies to problems with cyclic causal graphs. The heuristic can be generalized further, however, while remaing polynomial by relaxing the two assumptions that led us to it from the more general form (5) where each condition x_i is evaluated in a potentially different context s_i in $h(x_i|s_i)$. The two assumptions were 1) that the contexts s_i for all the conditions x_i in a rule $a : x', x_1, \ldots, x_n \rightarrow x$ are all the same and correspond to the state s' resulting from achieving the first condition x', and 2) that $h(x_i|s')$ is reduced to $h(x_i|s/x'_i)$ where x'_i is the value of x_i in s', thus effectively throwing away all the information in s' that does not pertain to the variable v associated with x_i. Both of these assumptions can be relaxed, leading to potentially more informed heuristics, still expressable in the style of (5) and computable in polynomial time. Some possibilities follow.

The formulation that follows from Assumptions 1 and 2 above presumes that in every rule $z \rightarrow x$ in the problem,

a) there is a condition in the body of the rule that must be achieved first; call it the *pivot* condition, b) that this pivot condition involves the same variable as the head, and c) that no *precedence information* involving the rest of the conditions in z is available or usable.

Condition a) is not particularly restrictive as it is always possible to add a dummy pivot condition. Condition b) on the other hand, is restrictive, and often, unnecessarily so. Consider for example the following rule for 'unlocking a door at a location'

$$have\_key(D), at = L \rightarrow unlocked(D) \qquad (10)$$

and say that the key is at a location L' different than L, while L_s is the current agent location. The cost of applying such a rule should include the cost of going from L_s to L' to pick up the key, as well as the cost of going from L' to L to unlock to the door. However, this won't follow from the formulation above or from the causal graph heuristic, as the variable in the head of the rule is boolean. We can actually cast this rule in the format assumed by the formulation as

$$\neg unlocked(D), have\_key(D), at = L \rightarrow unlocked(D)$$

where for boolean variables v, v is used as an abbreviation of $v = true$ and $\neg v$ of $v = false$, yet one can show that the context-mechanism that the causal graph heuristic adds to the normal additive heuristic, has no effect on rules $z \rightarrow v = d$ with *boolean variables* v in the head. This is because, either $v = d$ is true in s for a cost of 0, or $v = d'$ is true in s for $d' \neq d$, with no side-effect. For a rule such as (10), it makes sense to treat the boolean $have\_key(D)$ as the *pivot condition*, even if it involves a variable that is different than the one mentioned in the rule head. Actually the generalization of the heuristic for accounting for arbitrary pivot conditions in rules, as opposed to pivot conditions involving the head variable, is easy to accommodate. In the example above, the side effect of achieving the pivot condition $have\_key(D)$ involves $at = L'$, so that the second condition in the rule $at = L$ should be evaluated in that context, accounting for the cost of getting to the key location, and from there to the door.

A second generalization can be obtained by making use of further *precedence constraints* among the rule conditions. In the extreme case, if we have a *total ordering among all of the rule conditions*, then we should evaluate the second condition in the context that results from the achievement of the first, the third condition in the context that results from the achievement of the second, and so on. Indeed, such an ordering among conditions or *subtasks* is one of the key elements that distinguishes *HTN planning* (Erol, Hendler, & Nau 1994) from 'classical' planning. An heuristic of the type proposed can be used to provide an estimator capable of taking such precedence constraints into account.

The two generalizations above follow from relaxing Assumption 1 above. Assumption 2, that maps the heuristics $h(x_i|s')$ for contexts s' into the estimates $h(x_i|s/x_i')$ where x_i' is the value of x_i in s', throws away information in the context s' that does not pertain directly to the multivalued variable associated with x_i but which may be relevant to it. In the causal graph heuristic, this assumption translates in the exclusion of the non-parent variables v' from the subproblems $\Pi_{v,d,d'}$. There may be a bounded number of such variables v' however that one may want to keep in all such subproblems. Such an extension would cause polynomial complexity growth, and can be accommodated simply by changing the assumption $h(x_i|s') = h(x_i|s/x_i')$ implicit in (6), where x_i' is the value of x_i in s, by an explicit assumption $h(x_i|s') = h(x_i|s/x_i', y')$ where y' stands for the values of such 'core variables' to be preserved in all contexts.

It must be said that all these generalizations inherit certain commitments from the additive and causal graph heuristics, in particular, the additivity of the former, and the greedy (min) rule selection of the latter.

Discussion

Defining heuristics mathematically rather than procedurally seems to pay off as mathematical definitions are often clearer and more concise, and force us to express the assumptions explicitly, separating *what* is to be computed from *how* it is computed. Also, the functional equation form used in definition of the additive heuristic, that is common in dynamic programming, appears to be quite general and flexible. It has been used to define the max heuristic h_{max}, the h^m heuristics (Haslum & Geffner 2000), and more recently, the set-additive heuristic (Keyder & Geffner 2007) and the cost-sharing heuristics (Mirkis & Domshlak 2007).

The resulting formulation of the causal graph heuristic does not require acyclicity or causal graphs, and admits some interesting generalizations all of which have to do with *the use of ordering information among action precondition to capture side-effects in the computation of the heuristic.* There are some interesting connections with HTN planning, where precedence constraints among preconditions or subtasks are common, that are worth exploring too.

We have not explored the practical ramifications of the new formulation, e.g., on the performance of domain-independent planners, but hope that some of these ideas will turn out to be helpful in domains where the delete-relaxation heuristics, or relaxations that render a causal graph acyclic, are not appropriate. Actually, the causal graph heuristic is weak even in acyclic graphs where multivalued variables have many children. This happens for example when different values of such variables are needed by many other variables in the graph, as in a problem where many packages p_i must be picked from different locations l_i. The causal graph heuristic will behave in such a case as the normal additive heuristic, ignoring the side effects on the agent location that follow from getting the other packages. A refinement of the additive heuristic that is able to approximate the TSP cost in such cases is given in (Keyder & Geffner 2007).

Acknowledgements. H.Geffner is partially supported by Grant TIN2006-15387-C03-03 from MEC, Spain.

References

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS$^+$ planning. *Computational Intelligence* 11(4):625–655.

Bertsekas, D. 1991. *Linear Network Optimization: Algorithms and Codes*. MIT Press.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A robust and fast action selection mechanism for planning. In *Proceedings of AAAI-97*, 714–719. MIT Press.

Cormen, T. H.; Leiserson, C. E.; and Rivest, R. L. 1989. *Introduction to Algorithms*. The MIT Press.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *Proc. AAAI-94*.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *Proc. of the Fifth International Conference on AI Planning Systems (AIPS-2000)*, 70–82.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proc. ICAPS-04*, 161–170.

Helmert, M. 2006a. *Solving Planning Tasks in Theory and Practice*. Ph.D. Dissertation, Freiburg University, Germany.

Helmert, M. 2006b. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

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

Keyder, E., and Geffner, H. 2007. Set-Additive and TSP Heuristics for planning with action costs and soft goals. Technical report, Proc. 2007 ICAPS Workshop on Heuristics for Domain-Independent Planning.

Liu, Y.; Koenig, S.; and Furcy, D. 2002. Speeding up the calculation of heuristics for heuristic search-based planning. In *Proc. AAAI-02*, 484–491.

Mirkis, V., and Domshlak, C. 2007. Cost-sharing approximations for h^+. In *Proc. ICAPS-07*.