Unifying the Causal Graph and Additive Heuristics

Malte Helmert
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Georges-Köhler-Allee 52
79110 Freiburg, Germany
helmert@informatik.uni-freiburg.de

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

Abstract

Many current heuristics for domain-independent planning, such as Bonet and Geffner's *additive heuristic* and Hoffmann and Nebel's *FF heuristic*, are based on delete relaxations. They estimate the goal distance of a search state by approximating the solution cost in a relaxed task where negative consequences of operator applications are ignored. Helmert's *causal graph heuristic*, on the other hand, approximates goal distances by solving a hierarchy of "local" planning problems that only involve a single state variable and the variables it depends on directly.

Superficially, the causal graph heuristic appears quite unrelated to heuristics based on delete relaxation. In this contribution, we show that the opposite is true. Using a novel, declarative formulation of the causal graph heuristic, we show that *the causal graph heuristic is the additive heuristic plus context*. Unlike the original heuristic, our formulation does not require the causal graph to be acyclic, and thus leads to a proper generalization of both the causal graph and additive heuristics. Empirical results show that the new heuristic is significantly better informed than both Helmert's original causal graph heuristic and the additive heuristic and outperforms them across a wide range of standard benchmarks.

Introduction

The causal graph heuristic introduced by Helmert (2004; 2006) represents one of the few informative heuristics for domain-independent planning that takes into account negative operator interactions. Unlike the additive heuristic used early in the HSP planner (Bonet and Geffner 2001) and the relaxed planning graph heuristic used in FF (Hoffmann and 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 the heuristic would be intractable) and are defined *procedurally*.

In this paper 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 *requires neither 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, which 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 an operator's preconditions in order to capture side effects in the computation of the heuristic. Even without such extensions, we experimentally show that the new heuristic delivers considerably better heuristic guidance than both the causal graph heuristic and additive heuristic across a large suite of standard benchmarks.

Multi-valued Planning Tasks

The causal graph heuristic is defined over a planning language with multi-valued variables based on the SAS$^+$ language (Bäckström and 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 *multi-valued planning task* (MPT) 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 goal situations, 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 s' is a state restricted to a subset $V' \subseteq V$ of variables. We write $\mathrm{dom}(s')$ for the subset of variables on which s' is defined. 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. For an atom x, we write $\mathrm{var}(x)$ for the variable associated with x. For example, if x is the atom $v = d$, then $\mathrm{var}(x) = v$.

Given a state s and an atom $v = d$, $s[v = d]$ denotes the state that is like s except for variable v, which it maps to d. We will also use similar notations like $s[s']$ where s' is a partial state, to denote the state that is like s except for the

variables in $\text{dom}(s')$, where it is like s'.

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

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

These definitions follow the ones by Helmert (2006), apart from the omission of axioms and derived variables. (This is for simplicity of presentation; all results in this paper readily generalize to the full formalism.) In addition, for simplicity and without loss of generality, we make two assumptions. Firstly, we assume that *operator preconditions* $pre(o)$ *are empty*. This is because the value of none of the heuristics considered in this paper changes when preconditions $p \in pre(o)$ are moved into the body z of all effects $z \rightarrow v = d$. Secondly, we 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 from d; a transformation that preserves the semantics and complies with the above condition. Effects $z \rightarrow v = d$ can thus all be written as

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

While this language is not standard in planning, an automatic translation algorithm from PDDL into MPTs exists (Helmert 2008).

Causal Graph Heuristic

The **causal graph heuristic** $h^{\text{CG}}(s)$ provides an estimate of the number of operators 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 $s(v)$ in s to its value $s_\star(v)$ in the goal:

$$h^{\text{CG}}(s) \stackrel{\text{def}}{=} \sum_{v \in \text{dom}(s_\star)} cost_v(s(v), s_\star(v)) \tag{1}$$

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

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

The **causal graph** $\text{CG}(\Pi)$ 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 $\text{DTG}(v')$ or some operator o affects both v and v' (i. e., Π contains effects $o : z \rightarrow v = d$ and $o : z' \rightarrow v' = d'$ for some operator o).

The costs $cost_v(d, d')$ that determine the heuristic $h^{\text{CG}}(s)$ in Eq. 1 are defined in terms of the causal graph $\text{CG}(\Pi)$ and the domain transition graphs $\text{DTG}(v)$.

The definition assumes that $\text{CG}(\Pi)$ 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.[1]

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 $s[v = d]$ and 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, whose computation is intractable (Helmert 2004), and are defined *procedurally* using a slight modification of Dijkstra's algorithm (Cormen, Leiserson, and Rivest 1990; Bertsekas 2000). Costs are computed in topological causal order, starting with the variables with no parents in the causal graph. (In practice, not all cost values need to be computed, but this optimization is not relevant here.)

We will not repeat the exact procedure for computing these costs (found in Fig. 18, Helmert 2006). Instead, we will explain the procedure in a way that makes the relationship between causal graph heuristic and additive heuristic more direct.

Let us recall first that in **Dijkstra's algorithm**, a cost label $c(i)$ is associated with each node 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.

The algorithm finishes when *open* is empty, after a number of iterations bounded by the number of nodes in the graph. The cost 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 by *setting the cost of such edges dynamically, right after node i is selected for expansion.*

When v is a root variable in $\text{CG}(\Pi)$, **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 Dijkstra's: the graph is $\text{DTG}(v)$, 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 algorithm slightly by setting the cost of the edges $(d_1, d_2) \in \text{DTG}(v)$ labelled with condition z to

$$1 + \sum_{(v_i = e_i) \in z} cost_{v_i}(e_i', e_i) \tag{2}$$

---

[1] The original paper on the causal graph heuristic (Helmert 2004) also mentions, in passing, a generalization that does *not* require acyclicity. This generalization is very similar to the heuristic we introduce in this paper.

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 neither optimal nor 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 MPTs $\Pi = \langle V, s_0, s_\star, O \rangle$, the additive heuristic (Bonet, Loerincs, and Geffner 1997; Bonet and Geffner 2001) can be expressed as:

$$h^{\mathrm{add}}(s) \overset{\text{def}}{=} \sum_{x \in s_\star} h^{\mathrm{add}}(x|s) \tag{3}$$

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

$$h^{\mathrm{add}}(x|s) \overset{\text{def}}{=} \begin{cases} 0 & \text{if } x \in s \\ \min\limits_{o:z \to x} \big(1 + \sum\limits_{x_i \in z} h^{\mathrm{add}}(x_i|s)\big) & \text{if } x \notin s \end{cases} \tag{4}$$

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^{add} coincides with the max heuristic h^{max} (Bonet and Geffner 2001) and both heuristics are optimal.

It follows from this that if x represents any atom $v = d$ for some root variable v in the causal graph of Π, the value $h^{\mathrm{add}}(x|s)$ that follows from Eq. 4 is optimal, and thus in correspondence with the costs $cost_v(s(v), d)$ computed by Helmert's procedure. For other values d' of v the costs $cost_v(d', d)$ are equivalent to the estimates $h^{\mathrm{add}}(x|s')$ with $s' = s[v = d']$.

This correspondence between the costs $cost_v(d', d)$ and the estimates $h^{\mathrm{add}}(x|s')$ for root variables v raises the question of whether such costs can be characterized in the style of the additive heuristic also when v is *not* a root variable.

Notice first that Eq. 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 Eq. 4, which evaluates the costs $h^{\mathrm{add}}(x_i|s)$ of all conditions x_i in the body of the rules $o: z \to x$ with respect to the same state s. This however suggests considering variations of Eq. 4 where these conditions x_i are evaluated in *different* states s_i, rendering the general pattern

$$h(x|s) \overset{\text{def}}{=} \begin{cases} 0 & \text{if } x \in s \\ \min\limits_{o:z \to x} \big(1 + \sum\limits_{x_i \in z} h(x_i|s_i)\big) & \text{if } x \notin s \end{cases} \tag{5}$$

where the states s_i over which some of the conditions x_i in a rule are evaluated may be different from 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 from the seed state as *contexts*.

Context-enhanced Additive Heuristic

The use of Eq. 5 in place of Eq. 4 for defining a variation of the additive heuristic raises two questions:

1. how to choose the *contexts* s_i needed for evaluating the conditions x_i in a given rule $o: z \to x$, 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 of the causal graph heuristic, thus obtaining a heuristic that is closely related to both the causal graph and additive heuristics. We call the resulting heuristic the *context-enhanced additive heuristic* h^{cea}. We remark that there are other reasonable answers to these two questions, and we will discuss some generalizations later.

Let us recall first that, without loss of generality, we assume that all the rules have the form $o: x', z \to x$ where x and x' are atoms that *refer to the same variable*, i.e., $\mathrm{var}(x) = \mathrm{var}(x')$. Also recall that $s[s']$ for state s and partial state s' refers to the state that is like s except for the variables mentioned in s', where it is equal to s'.

The answer to the first question implied by Helmert's formulation is that for rules $o: x', z \to x$ the condition x' *that refers to the same variable as x is achieved first, and the conditions in z are evaluated in the state s' that results* (**Assumption 1**).

The answer to the second question is that in the heuristic computation for a seed state s, the costs $h^{\mathrm{cea}}(x_i|s_i)$ for a context s_i are mapped into costs $h^{\mathrm{cea}}(x_i|s[x_i'])$ where x_i' is the value of $\mathrm{var}(x_i)$ in s_i, meaning that information in the state s_i about other variables is discarded (**Assumption 2**). We will write $h^{\mathrm{cea}}(x|s[x])$ as $h^{\mathrm{cea}}(x|x)$.

These assumptions lead to a heuristic that is very much like the additive heuristic except that a) the heuristic values $h^{\mathrm{cea}}(x|s)$ are computed not only for the seed state s but for all states $s' = s[x']$ where 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 $o: x'', z \to x$ are evaluated in the state $s(x''|x')$ that results from achieving the "pivot" condition x'', producing then the context $s(x|x')$ associated with x. Formally, we get the following equations:

$$h^{\mathrm{cea}}(x|x') \overset{\text{def}}{=} \begin{cases} 0 & \text{if } x = x' \\ \min\limits_{o:x'', z \to x} \big(1 + h^{\mathrm{cea}}(x''|x') \\ \qquad + \sum\limits_{x_i \in z} h^{\mathrm{cea}}(x_i|x_i')\big) & \text{if } x \neq x' \end{cases} \tag{6}$$

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

$$s(x|x') \stackrel{\text{def}}{=} \begin{cases} s[x'] & \text{if } x = x' \\ s(x''|x')[z,x][y_1,\ldots,y_n] & \text{if } x \neq x' \end{cases} \quad (7)$$

where x'' is the atom for which $o : x'', z \to x$ is the rule that yields the minimum[2] in Eq. 6, and $y_1, \ldots, y_n$ are the heads of all rules that must trigger simultaneously with this rule (i. e., $o : x'', z_i \to y_i$ for some $z_i \subseteq z$ for all $i = 1, \ldots, n$, for the same operator o). In words, when $o : x'', z \to x$ is the best (cost-minimizing) achiever for atom x from x' according to Eq. 6, and $s(x''|x')$ is the state that is deemed to result from 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 from achieving x from x'.

The *context-enhanced* additive heuristic $h^{\text{cea}}(s)$ is then defined as

$$h^{\text{cea}}(s) \stackrel{\text{def}}{=} \sum_{x \in s_\star} h^{\text{cea}}(x|x_s) \quad (8)$$

where x_s is the atom that refers to $\mathrm{var}(x)$ in s, and $h^{\text{cea}}(x|x')$ is defined by Eqs. 6 and 7.

Example

As an illustration, consider a problem with a Boolean variable Y and a multi-valued variable $X \in \{0, \ldots, n\}$ represented by the atoms y, $\neg y$, and x_i, standing for the assignments $Y = \textit{true}$, $Y = \textit{false}$, and $X = i$ for $i = 0, \ldots, n$, and operators a and b_i, $i = 0, \ldots, n-1$ with rules

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

(note how the operators b_i have two effects). We want to determine the value of $h^{\text{cea}}(s)$ when x_0 and y hold in s and the goal is x_n. From Eq. 8, $h^{\text{cea}}(s) = h^{\text{cea}}(x_n|x_0)$. The optimal plan for the problem is the operator sequence $b_0, a, b_1, \ldots, a, b_{n-1}$ for a cost of $2n - 1$. The plans must increase the value of X one by one, and in between any two steps the value of Y that is made false by each increase in X must be restored to true.

From Eq. 6, it follows that

$$h^{\text{cea}}(x_0|x_0) = 0$$
$$h^{\text{cea}}(x_{i+1}|x_0) = 1 + h^{\text{cea}}(x_i|x_0) + h^{\text{cea}}(y|y') \quad (9)$$

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. By Eq. 7, this state is characterized as:

$$s(x_0|x_0) = s$$
$$s(x_{i+1}|x_0) = s(x_i|x_0)[x_{i+1}, \neg y] \quad (10)$$

From Eq. 10, we see that $\neg y$ holds in all states $s(x_i|x_0)$ for $i \geq 1$, so that $y' = \neg y$ in Eq. 9 for $i \geq 1$. However,

$y' = y$ in Eq. 9 for $i = 0$, since $s(x_0|x_0) = s$, and y is true in s. Clearly, we have $h^{\text{cea}}(y|y) = 0$ and $h^{\text{cea}}(y|\neg y) = 1$, so that Eq. 9 becomes:

$$h^{\text{cea}}(x_1|x_0) = 1 + h^{\text{cea}}(x_0|x_0) + 0$$
$$h^{\text{cea}}(x_{i+1}|x_0) = 1 + h^{\text{cea}}(x_i|x_0) + 1 \quad \text{for } i \geq 1$$

With $h^{\text{cea}}(x_0|x_0) = 0$, we thus get $h^{\text{cea}}(x_i|x_0) = 2i - 1$ for all $i \geq 1$, and therefore $h^{\text{cea}}(s) = h^{\text{cea}}(x_n|x_0) = 2n - 1$, so that $h^{\text{cea}}(s)$ is optimal. The plain additive heuristic, on the other hand, is $h^{\text{add}}(s) = n$, which is optimal only for the delete relaxation.

Relationship to Causal Graph Heuristic

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:[3]

Theorem 1 (h^{cea} vs. h^{CG})
If the causal graph $CG(\Pi)$ is acyclic, then the causal graph heuristic h^{CG} and the context-enhanced additive heuristic h^{cea} are equivalent, i. e., for every state s, $h^{CG}(s) = h^{cea}(s)$.

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 $h^{\text{cea}}(x'|x)$ defined by Eq. 6 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 Eq. 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 procedure.

The first part is straightforward and was mentioned earlier: 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 $h^{\text{cea}}(x'|x)$ are 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 $h^{\text{cea}}(x'|x)$ 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 can be shown that the correspondence holds after $i + 1$ node expansions as well.

The above result suggests that the context-enhanced additive heuristic might be more informative than the causal graph heuristic: in problems with acyclic causal graphs, it is equally informative, and in problems with cycles in the causal graph, it does not need to ignore operator preconditions in order to break these cycles. We will later investigate this potential advantage empirically.

---

[2]If there are several rules minimizing the expression, some form of tie-breaking is needed to make the equations well-defined, for example by imposing an arbitrary strict order on rules. Different tie-breaking can lead to different heuristic estimates. This is not unusual; e. g., the causal graph heuristic and FF heuristic also depend on how ties between achievers of a proposition are broken.

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

Relationship to Additive Heuristic

As stated earlier, there is also a close relationship between the context-enhanced additive heuristic and the plain additive heuristic. Indeed, it is easy to prove the following result:

Theorem 2 (h^{cea} vs. h^{add})
In problems where all state variables have Boolean domains, the additive heuristic h^{add} and the context-enhanced additive heuristic h^{cea} are equivalent, i.e., for every state s, $h^{add}(s) = h^{cea}(s)$.

To see that this is true, observe that Eq. 6 defining $h^{\text{cea}}(x|x')$ simplifies significantly for Boolean domains. For the case where $x \neq x'$, we must have $x'' = x'$, since x'' must be different from x in a rule $o : x'', z \rightarrow x$ and there are only two values in the domain of $\text{var}(x)$. This means that the conditions in z are evaluated in the context of $s(x'|x')$, i.e., the same state in which $h^{\text{cea}}(x|x')$ is evaluated. Using this observation, it is easy to prove inductively that all costs are evaluated in the context of the seed state s, so that Eq. 6 for h^{cea} defines the same cost values as Eq. 4 for h^{add}.

Again, the result suggests that the context-enhanced additive heuristic might be more informative than the additive heuristic: in the case of all-Boolean state variables, the two heuristics are the same, while in general, h^{cea} may be able to use the context information to improve its estimates.

Computing the Heuristic for Cyclic Graphs

The context-enhanced additive heuristic h^{cea} reduces to the causal graph heuristic h^{CG} in MPTs 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^{cea} by an implicit induction over costs.

Despite the presence of a cyclic graph, the costs $h^{\text{cea}}(x|x')$ in Eqs. 6 and 7 can be computed using a modification of Dijkstra's algorithm, similar to Helmert's algorithm for h^{CG}. Indeed, Dijkstra's algorithm can be used for computing the *normal additive heuristics* as well, whether the causal graph is cyclic or not, although in practice the Bellman-Ford algorithm is preferred (Liu, Koenig, and Furcy 2002).

For h^{cea}, the Dijkstra-like algorithm operates on a graph containing one copy of $\text{DTG}(\text{var}(x'))$ for each atom x' of the task. In the copy for atom x', the distance from x' to x corresponds to the heuristic value $h^{\text{cea}}(x|x')$ from Eq. 6. The algorithm initially assigns cost 0 to nodes corresponding to values $h^{\text{cea}}(x'|x')$ and cost ∞ to other nodes. It then expands nodes in order of increasing cost, propagating costs and contexts in a similar way to Helmert's procedure for h^{CG}.

However, while this approach leads to polynomial evaluation time, we have found it prohibitively expensive for use in a heuristic planner supposed to provide competitive results with h^{CG} and h^{add}. A critical speed-up can be obtained by separating the *cost* of a node in the Dijkstra algorithm (the associated $h^{\text{cea}}(x|x')$ value) from its *priority* (the time during the execution of the Dijkstra algorithm where the value is required). In particular, many nodes with low cost are never needed for the computation of $h^{\text{cea}}(s)$ because they refer to contexts that never arise in the computation of the relevant cost estimates. By expanding nodes in order of increasing priority, rather than cost, we can compute $h^{\text{cea}}(s)$ much more efficiently. Due to space restrictions, we do not discuss these algorithmic issues in detail.

Experiments

We have seen that the context-enhanced additive heuristic is a natural generalization of both the additive heuristic and the (acyclic) causal graph heuristic. In this section, we show that the unified heuristic is not just a theoretical concept, but can actually be put to good use in practice. For use in a heuristic planning system, a heuristic estimator must be *informative*, i.e., provide heuristic estimates that guide the search algorithm to a goal state without expanding too many search nodes, and it must be *efficiently computable*, i.e., require as little time as possible for each heuristic evaluation. Since h^{cea} clearly requires more computational effort than either of the heuristics it generalizes, the question is whether there are cases where it provides significantly better guidance, enough to overcome the higher per-node overhead.

We conducted two experiments to address this question. In the first experiment we investigate overall performance, while in the second we look more closely at the informativeness of the heuristics. As a search algorithm, we use greedy best-first search with deferred evaluation and preferred operators, as implemented in the Fast Downward planning system (Helmert 2006). All experiments were conducted with a 30 minute timeout and 2 GB memory limit on a computer with a 2.66 GHz Intel Xeon CPU.

As all heuristics were implemented within the same planning system and we want to compare the heuristics, not overall system performance, all runtimes reported in the following refer to search time (i.e., time spent by the search algorithm and heuristic evaluators), excluding overhead for parsing and grounding that is identical for all heuristics.

Domain	Helmert, 2004			New Results			
	h^{CG}	h^{add}	h^{FF}	h^{CG}	h^{add}	h^{FF}	h^{cea}
BLOCKSWORLD (35)	0	0	0	0	0	0	0
DEPOT (22)	14	11	10	10	7	5	5
DRIVERLOG (20)	3	3	2	0	0	1	2
FREECELL (80)	2	11	9	9	0	1	2
GRID (5)	1	2	1	1	2	1	1
GRIPPER (20)	0	0	0	0	0	0	0
LOGISTICS (63)	0	0	0	0	6	1	0
MICONIC (150)	0	0	0	0	0	0	0
MOVIE (30)	0	0	0	0	0	0	0
MPRIME (35)	1	6	6	0	1	4	0
MYSTERY (19)	1	0	1	2	1	2	0
ROVERS (20)	3	5	3	0	0	0	0
SATELLITE (20)	0	0	2	0	0	0	0
ZENOTRAVEL (20)	0	0	0	0	0	0	0
Total (539)	**25**	**38**	**34**	**22**	**17**	**15**	**10**

Table 1: Number of unsolved problem instances by domain, for the benchmark suite considered by Helmert (2004). Eleven unsolvable MYSTERY instances excluded. First column shows number of problem instances in parentheses.
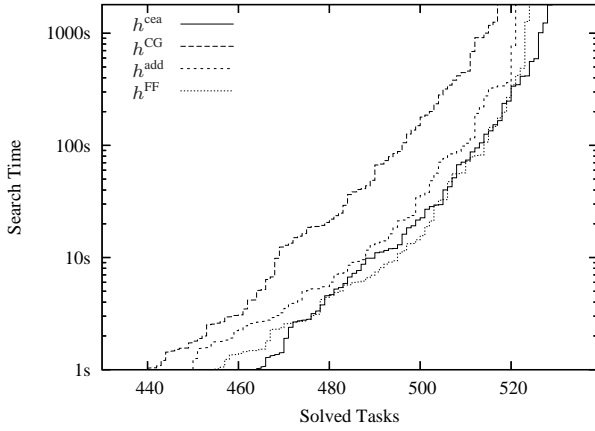
Figure 1: Number of tasks solved by each heuristic within a given time, for Helmert's (2004) benchmark suite.

First Experiment

The first experiment, in which we investigate the overall performance of the context-enhanced additive heuristic, is modelled after the main experiment of the original paper on the causal graph heuristic (Helmert 2004). We evaluate on the same benchmarks (namely, the STRIPS benchmarks of IPC 1–3), we report the same data (number of problem instances solved in each domain and overall runtime results), and we compare the same heuristic estimators (additive heuristic, acyclic causal graph heuristic, FF heuristic). In addition, we provide results for the context-enhanced additive heuristic.

Table 1 reports the overall success rates of all heuristics, showing the number of problem instances *not solved* with each heuristic in each domain. For reference, we also include the results from Helmert's original experiment.

We first observe that for each of the three reference heuristics, we report better results than in the original experiment. This is partly due to differences in the experimental setup (e. g., Helmert only used a 5 minute timeout), but also due to the search enhancements used. The improvement is much more pronounced for the additive and FF heuristics than for the causal graph heuristic because the latter benefits less from the use of preferred operators. (For all three heuristics, disabling preferred operators degrades results, but the effect is least pronounced for the causal graph heuristic.)

Secondly, the table shows that the results for the context-enhanced additive heuristics are markedly better than for the other heuristics. While the closely related causal graph and additive heuristics leave 22 (h^{CG}) and 17 (h^{add}) out of 539 tasks unsolved, the context-enhanced additive heuristic solves all but 10 tasks, roughly halving the number of failures. The FF heuristic achieves the next best result by leaving 15 out of 539 tasks unsolved, 50% more than h^{cea}.

Aggregate runtime results are shown in Fig. 1. The graph for h^{cea} is consistently below h^{CG} and h^{add}, showing that the context-enhanced additive heuristic solves more problems than the two heuristics it generalizes for any timeout between one second and 30 minutes. The results for h^{cea} and

Domain	Percent	Median	Average
ASSEMBLY	100%	2034	2670
DEPOT	100%	15.487	16.042
FREECELL	94%	10.615	11.643
MPRIME	100%	10.308	21.076
TRUCKS	91%	8.377	5.758
GRID	100%	4.735	4.117
DRIVERLOG	94%	3.891	3.502
PIPESWORLD-NOTANKAGE	82%	3.057	4.814
ZENOTRAVEL	92%	2.186	2.261
TPP	84%	1.848	2.134
MYSTERY	85%	1.846	2.210
BLOCKSWORLD	71%	1.843	2.692
PIPESWORLD-TANKAGE	80%	1.818	2.592
AIRPORT	78%	1.706	4.393
PATHWAYS	80%	1.700	8.215
LOGISTICS	96%	1.628	1.690
PSR-SMALL	86%	1.364	1.436
PSR-LARGE	76%	1.354	1.264
PSR-MIDDLE	64%	1.050	1.077
ROVERS	52%	1.038	1.334
OPENSTACKS	52%	1.030	2.403
SCHEDULE	35%	0.999	0.720
SATELLITE	38%	0.966	0.583
STORAGE	41%	0.939	0.569
OPTICALTELEGRAPHS	0%	0.923	0.923
MICONIC-FULLADL	35%	0.917	0.846
PHILOSOPHERS	10%	0.612	0.617

Table 2: Comparison of node expansions of h^{cea} and h^{CG}: percentage of tasks where h^{cea} expands fewer nodes than h^{CG}, median reduction in node expansions, (geometric) average reduction in node expansions. Domains are ordered by decreasing rate of (median) improvement. Above the line, h^{cea} is more informed than h^{CG}.

h^{FF} are comparable. Their graphs intersect occasionally, so either heuristic outperforms the other for some timeouts.

Second Experiment

In the second experiment, we focus on the informedness of the context-enhanced additive heuristic, compared to the causal graph and additive heuristics. In this experiment, we consider all domains from the previous International Planning Competitions (IPC 1–5), apart from those where each of the three heuristics can solve each benchmark instance in less than one second (thus excluding trivial domains like MOVIE and GRIPPER).

We measure the informedness of a heuristic for a given planning instance by the number of node expansions needed to solve it. Table 2 shows comparative results for the causal graph and context-enhanced additive heuristic. To be able to provide a comparison, we only consider instances solved by both heuristics. We show both qualitative and quantitative results. Qualitatively, we show the *percentage* of tasks in each domain where h^{cea} required fewer expansions. Larger values than 50% indicate that h^{cea} was more informed for *most* tasks of the domain. Quantitatively, we computed the ratio between the number of node expansions of h^{CG} and h^{cea} for each task (where values greater than 1 indicate that h^{cea} required fewer expansions). For each domain, we report

Domain	Percent	Median	Average
GRID	100%	2.714	2.244
DEPOT	93%	2.389	2.781
LOGISTICS	100%	2.236	3.117
PIPESWORLD-TANKAGE	64%	1.901	2.255
MPRIME	91%	1.691	4.720
ZENOTRAVEL	92%	1.437	1.723
AIRPORT	74%	1.353	1.924
MYSTERY	78%	1.320	2.319
PIPESWORLD-NOTANKAGE	65%	1.242	1.572
ASSEMBLY	78%	1.145	1.356
DRIVERLOG	69%	1.031	1.758
OPTICALTELEGRAPHS	100%	1.023	1.023
PHILOSOPHERS	76%	1.019	1.362
OPENSTACKS	59%	1.003	1.070
PSR-SMALL	55%	1.000	1.007
BLOCKSWORLD	49%	1.000	0.871
STORAGE	47%	1.000	0.922
ROVERS	45%	1.000	0.992
TPP	44%	1.000	0.978
PSR-MIDDLE	44%	1.000	0.955
PATHWAYS	30%	0.999	0.948
MICONIC-FULLADL	31%	0.992	0.996
PSR-LARGE	42%	0.988	0.890
SATELLITE	17%	0.962	0.920
SCHEDULE	26%	0.931	0.650
FREECELL	25%	0.770	0.605
TRUCKS	20%	0.727	0.837

Table 3: Comparison of node expansions of h^{cea} and h^{add}. Columns are as in Table 2. Above the first line, h^{cea} is more informed than h^{add}; below the second line, h^{cea} is less informed than h^{add}. In between, the median number of node expansions is identical.

the *median* of these values, indicating the *typical* advantage of h^{cea} over h^{CG}, and the *geometric mean*,[4] indicating the *average* advantage of h^{cea} over h^{CG}.

The table clearly shows that h^{cea} provides better heuristic estimates than h^{CG}. The context-enhanced additive heuristic provides better estimates in 21 out of 27 domains; in 15 of these cases, the median improvement is a factor of 1.7 or better. In four cases, the median improvement is enormous, by a factor of 10 or better. Conversely, h^{CG} only provides better estimates in 6 out of 27 domains, and its advantage in the median case only exceeds a factor of 1.1 in a single case (PHILOSOPHERS), where it is 1.634.

Table 3 shows a similar comparison between the additive and context-enhanced additive heuristics. While the advantage for h^{cea} is less pronounced in this case, it is still quite apparent. The context-enhanced additive heuristic is more informed in the median case for 14 domains, and less informed for 7 domains. Only in two cases is the advantage of h^{add} larger than 1.1, and it is never larger than 1.376. On the other hand, h^{cea} shows an improvement by a factor of more than 1.1 for ten domains, and in three cases it exceeds a factor of 2.

---

[4]For ratios, the geometric mean, rather than the arithmetic mean, is the aggregation method of choice. To see this, note that the "average" of 5.0 and 0.2 should be 1.0, not 2.6.

Discussion

The context-enhanced heuristic h^{cea} is more general than the causal graph heuristic as it applies to problems with cyclic causal graphs. It is also more general than the additive heuristic, because it can take into account context information ignored by h^{add}. We have seen that this increase in generality translates to better heuristic guidance in many standard planning domains.

The heuristic can be generalized further, however, while remaining tractable, by relaxing the two assumptions that led us to it from the more general form in Eq. 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 $o : x', x_1, \ldots, x_n \to x$ 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 $\text{var}(x_i)$ in s', thus effectively throwing away all the information in s' that does not pertain to the variable associated with x_i. Both of these assumptions can be relaxed, leading to potentially more informed heuristics, still expressible in the style of Eq. 5 and computable in polynomial time. We now discuss some possibilities.

Generalizations

The formulation that follows from Assumptions 1 and 2 above presumes that in every rule $z \to x$ of the problem, a) there is a condition in the body of the rule that must be achieved first (we 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":

$$have\_key(D), at = L \to unlocked(D) \qquad (11)$$

and say that the key is at a location L' different from 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 does not follow from the formulation above or from the causal graph heuristic. We can actually cast this rule into the format assumed by the formulation as

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

where for Boolean variables v, v abbreviates $v = true$ and $\neg v$ abbreviates $v = false$. However, as indicated in the proof sketch for Theorem 2, the context mechanism that the causal graph heuristic adds to the normal additive heuristic has no effect on rules $z \to v = d$ with *Boolean variables* v in the head. For a rule such as Eq. 11, it makes sense to treat the atom $have\_key(D)$ as the *pivot condition*, even though it involves a variable that is different from 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*, we should evaluate the second condition in the context that results from achieving the first, the third condition in the context that results from achieving 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, and Nau 1994) from "classical" planning. A variation of the context-enhanced additive heuristic 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, which maps the estimates $h(x_i|s')$ for contexts s' into the estimates $h(x_i|s[x_i'])$ where x_i' is the value of $\mathrm{var}(x_i)$ in s', throws away information in the context s' that does not pertain directly to the multi-valued 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 $\Pi_{v,d,d'}$. Instead, one may want to keep a bounded number of such variables v' in all subproblems. Such an extension would cause polynomial growth in complexity, and can be accommodated simply by changing the assumption $h(x_i|s') = h(x_i|s[x_i'])$ implicit in Eq. 6, where x_i' is the value of $\mathrm{var}(x_i)$ 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, however, that all these generalizations inherit certain commitments from the additive and causal graph heuristics, in particular, the additivity of the former, and the greedy rule selection of the latter.

Summary

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 defining the additive heuristic, which 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 and Geffner 2000), and more recently the set-additive heuristic (Keyder and Geffner 2008) and cost-sharing heuristics (Mirkis and Domshlak 2007).

The resulting declarative formulation of the causal graph heuristic, called the *context-enhanced additive heuristic*, does not require acyclicity or causal graphs and leads to a natural generalization of both the causal graph and additive heuristics. We consider this observation an interesting contribution in its own right, as it establishes a strong connection between two previously unrelated research strands in heuristic search planning. In addition to its theoretical interest, the context-enhanced additive heuristic also proves to be very strong in practice.

Finally, the ideas presented in this paper admit some interesting generalizations, all of which have to do with *the use of ordering information among operator preconditions 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, which are also worth exploring.

Acknowledgments

References

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

Bertsekas, D. P. 2000. *Linear Network Optimization: Algorithms and Codes*. The MIT Press.

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

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

Cormen, T. H.; Leiserson, C. E.; and Rivest, R. L. 1990. *Introduction to Algorithms*. The MIT Press.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *Proc. AAAI-94*, 1123–1128.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *Proc. AIPS 2000*, 140–149.

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

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26:191–246.

Helmert, M. 2008. *Understanding Planning Tasks – Domain Complexity and Heuristic Decomposition*, volume 4929 of *LNAI*. Springer-Verlag.

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

Keyder, E., and Geffner, H. 2008. Heuristics for planning with action costs revisited. In *Proc. ECAI 2008*.

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

Mirkis, V., and Domshlak, C. 2007. Cost-sharing approximations for h^+. In *Proc. ICAPS 2007*, 240–247.