A Unified View of Cost-Based Heuristics

Raquel Fuentetaja and **Daniel Borrajo** and **Carlos Linares López**
Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30. 28911 Leganés (Madrid). Spain
rfuentet@inf.uc3m.es, dborrajo@ia.uc3m.es, clinares@inf.uc3m.es

Abstract

Many cost-based planning heuristics are based on partial or complete delete relaxation. Definitions of these heuristics are of different nature, which makes it difficult to establish relations and formal comparisons among them. In this paper, we propose a unified definition with enough generality to cover most of existing heuristics. Important relations among heuristics can be derived from our study, as, for example, that some heuristics are equivalent to others under some conditions. Also, a side effect is that the generalized definition provides a framework to derive new heuristics for cost-based planning.

Introduction

Currently, finding high quality plans (not necessarily optimal) is both a requirement for most real world domains, and an open research direction. The development of planners that are able to explicitly reason about the cost of the solutions according to a specific metric is a key question. In fact, in the last International Planning Competition (IPC-2008) there was a track for cost-based planners.[1]

Several numerical heuristics have been defined for classical and cost-based planning. Some of them have been procedurally defined (expressing *how* to compute the heuristic) and some of them have been defined declaratively (expressing *what* should be computed). Though both types of definitions are correct, it can be somehow confusing not to have all heuristics defined in the same way, since it is much more difficult to recognize the similarities and differences. We tend to agree with Geffner (Geffner 2007) in that defining heuristics declaratively, using the mathematical language, seems to be more adequate given that such formal definitions are usually very clear and concise. In this paper we present a generalized declarative and mathematical definition for cost-based heuristics, as well as for classical heuristics, given that the cost-based planning model generalizes the classical one. The generalization covers also heuristics defined in their original papers from a procedural point of view, using Relaxed Planning Graphs (RPGs). We pursue the following objectives: (1) to unify the definition of several existing heuristics; (2) to clarify the relation between them; and (3) to provide a framework for deriving new heuristics.

[1] http://ipc.informatik.uni-freiburg.de/

Generalized Definition of Heuristics

We consider a cost-based planning problem as a tuple $(\mathcal{P}, \mathcal{A}, \mathcal{I}, \mathcal{G}, \mathcal{C})$, where $\mathcal{P}$ is a set of propositions, $\mathcal{A}$ is a set of grounded actions, $\mathcal{I} \subseteq \mathcal{P}$ and $\mathcal{G} \subseteq \mathcal{P}$ are the initial state and the set of goals respectively, and $\mathcal{C}$ is the set of action costs. $\mathcal{C}$ contains a real value, $cost(a)$, for each action a in $\mathcal{A}$, that represents its cost. Action costs are assumed to be state independent and non-negative. The cost of a plan α is the sum of its action costs:

$$cost(\alpha) = \sum_{a \in \alpha} cost(a)$$

The cost of a plan can be also expressed as a function of all actions in the domain as:

$$cost(\alpha) = \sum_{a \in \mathcal{A}} N(a) \times cost(a)$$

where the factor $N(a) \in \mathbb{N} \cup \{0\}$ indicates how many times the action a appears in the plan α. The factors $N(a)$ for the optimal plan from a given state s are unknown a priori. Heuristics are based on obtaining an estimate $N'(a)$ for these factors. For most of the heuristics based on *delete-relaxation* including those based on RPGs, $N'(a) \in \{0, 1\}$, given that to solve the relaxed task each action has to be applied at most once, since its effects can not be deleted. An exception is the *additive* heuristic, where $N'(a)$ can be higher than 1. The idea of the generalized definition is to propagate (from action preconditions to effects) information general enough to compute the $N'(a)$ factors for the existing heuristics. The objective is to establish a common framework in order to discover commonalities and differences among them.

The generalized definition uses *multisets*. Multisets are sets in which the same member can appear several times. Elements of multisets are usually represented as pairs in the form $(e, m(e))$, where e is a member of the multiset, and $m(e)$ is an integer value representing its multiplicity. If a multiset M has a member e, we will say that e belongs to the multiset ($e \in M$). We define the *additive-union*, $\uplus$, of two multisets, M_1 and M_2, as a new multiset containing all elements of M_1 and M_2 whose members are different, together with a new pair for each member of both multisets,

whose multiplicity is the sum of multiplicities:

$$\begin{aligned} M_1 \uplus M_2 = & \{(e, m(e)) \mid (e, m(e)) \in M_1, e \notin M_2\} \cup \\ & \{(e, m(e)) \mid (e, m(e)) \in M_2, e \notin M_1\} \cup \\ & \{(e, m_1(e) + m_2(e)) \mid (e, m_1(e)) \in M_1, \\ & \qquad (e, m_2(e)) \in M_2\} \end{aligned} \tag{1}$$

Similarly, the *max-union*, $\underset{\max}{\cup}$, of two multisets is defined using the maximum of multiplicities instead of the sum:

$$\begin{aligned} M_1 \underset{\max}{\cup} M_2 = & \{(e, m(e)) \mid (e, m(e)) \in M_1, e \notin M_2\} \cup \\ & \{(e, m(e)) \mid (e, m(e)) \in M_2, e \notin M_1\} \cup \\ & \{(e, \max(m_1(e), m_2(e))) \mid (e, m_1(e)) \in M_1, \\ & \qquad (e, m_2(e)) \in M_2\} \end{aligned} \tag{2}$$

Factors $N'(a)$ are estimates of the number of times each action a has to be used to generate propositions estimated to be required. Thus, the relevant information to compute these factors should include how many times each action is used to achieve each (estimated) required proposition. For this reason, in our generalized definition we will use multisets whose members are action-proposition pairs (a, p) (i.e. the elements of the multisets have the form $((a, p), m((a, p)))$), where $a \in \mathcal{A}$ is an action and $p \in \mathcal{P}$ is a proposition. With these pairs we will denote that action a has been used to achieve proposition p as many times as indicated by the multiplicity. Let $\mathcal{A} \times \mathcal{P}$ be the set of action-proposition pairs in a planning domain. We will denote the multisets with members in the domain $\mathcal{A} \times \mathcal{P}$ by $M_{\mathcal{A}\times\mathcal{P}}$. We will use two types of functions that operate with these multisets, *aggregation* (Agg) and *cost* ($Cost$) functions:

- $Agg : M_{\mathcal{A}\times\mathcal{P}} \times \ldots \times M_{\mathcal{A}\times\mathcal{P}} \rightarrow M_{\mathcal{A}\times\mathcal{P}}$: this function determines how to aggregate a finite number of multisets of action-proposition pairs. The result will be another multiset of action-proposition pairs. For example, the function Agg will be used to group the information each action receives from its preconditions. Classical aggregation functions are the *maximum* and the *sum*, defined for the *max* and the *additive* heuristics respectively. For these cases the aggregation is defined over scalars instead of over multisets.
- $Cost : M_{\mathcal{A}\times\mathcal{P}} \rightarrow \mathbb{R}$: this function determines how to compute a cost value from a multiset of action-proposition pairs. The result will be a real value.

The generalized declarative definition, based on propagating multisets of action-proposition pairs, is composed of the following four equations (Equations (3), (4), (5), and (6)). The letter π represents a multiset of action-proposition pairs.

The multiset with information enough to compute the $N'(a)$ factors to estimate the cost of achieving a set of propositions P from a given state s using an aggregation function Agg is:

$$\pi(P; s; Agg) = \begin{cases} \emptyset & \text{if } P \subseteq s \\ \pi(a_P; P; s; Agg) & \text{if } P \not\subseteq s, |P| \leq m \\ Agg(\{\pi(P_m; s; Agg\}) & \text{if } P \not\subseteq s, |P| > m \end{cases} \tag{3}$$

where P_m represents any subset of P of size m: $P_m \subseteq P$ and $|P_m| = m$. This definition follows the idea of the h^m heuristics (Haslum & Geffner 2000) in that the multiset for a set of propositions P with size higher than m, is computed by aggregating the multisets of all the subsets of P of size m. For sets of propositions with size lower than m, the result is the multiset obtained by forcing the use of the action a_P, $\pi(a_P; P; s; Agg)$, defined below. a_P is an action that generates at least a proposition in P without deleting any other proposition in that set. For the rest of heuristics we have analyzed, $m = 1$.

When we force the use of an action a, the result will contain a pair (a, p) with multiplicity 1 for each proposition p in P the action adds, together with the action-proposition pairs for obtaining the preconditions of a and the other propositions in P the action a does not add, i.e. to obtain the propositions in the set obtained by regression of the atoms in P through a, $Reg(P, a) = P \setminus add(a) \cup pre(a)$. When P only has one atom, $Reg(P, a) = pre(a)$. As some of these pairs can refer to the same action and proposition they should be combined by the *additive-union*. Thus:

$$\pi(a; P; s; Agg) = \left\{ \biguplus_{p \in add(a) \cap P} \{((a, p), 1)\} \right\} \biguplus \pi(Reg(P, a); s; Agg) \tag{4}$$

The action a_P in Equation (3) is one of the actions in the domain that generates a proposition in P without deleting any other. Let $A(P)$ be this set of actions ($A(P) = \{a \in \mathcal{A} \mid add(a) \cap P \neq \emptyset \wedge del(a) \cap P = \emptyset\}$). Then, the election of a_P is usually done by choosing the action whose multiset minimizes the cost function $Cost_{a_P}$:

$$a_P \in \underset{a \in A(P)}{\arg\min}\ Cost_{a_P}\left(\pi(a; P; s; Agg_{a_P})\right) \tag{5}$$

where Agg_{a_P} is the aggregation function used. When P only has a proposition p, the action a_P is usually called *best supporter* of p.

Finally, the heuristic estimate of obtaining the goals $\mathcal{G}$ from a state s, is computed using the cost function $Cost_h$ over $\pi(\mathcal{G}; s; Agg_h)$:

$$h(\mathcal{G}, s) = Cost_h(\pi(\mathcal{G}; s; Agg_h)) \tag{6}$$

The function $Cost_h$ indicates how the information in $\pi(\mathcal{G}; s; Agg_h))$ determines the factors $N'(a)$ used by the heuristic. Note that the aggregation function for selecting a_P, Agg_{a_P}, can be different from the aggregation function Agg_h, used to compute $h(\mathcal{G}, s)$.

To instantiate the generalized definition and obtain a particular heuristic, we need to specify m (size of subsets) and four functional parameters (the aggregation and cost functions for selecting a_P and the aggregation and cost function to compute $h(\mathcal{G}, s)$): $(m, Agg_{a_P}, Cost_{a_P}, Agg_h, Cost_h)$.

Aggregation and Cost Functions

Several aggregation and cost functions can be defined. In this section we include the aggregation and cost functions that should be defined to cover most of the existent heuristics. In the next section we will show what functions use each heuristic (instantiation of this generalized framework).

Aggregation functions

- Function *Agg-union+*: this function aggregates a finite number of multisets by applying the *additive-union* of multisets (defined in equation (1)) among them:

$$\textit{Agg-union+}(\pi_1, \pi_2, \ldots, \pi_n) = \biguplus \pi_i \quad (7)$$

- Function *Agg-union-max*: this function aggregates a finite number of multisets by applying the *max-union* of multisets (defined in equation (2)) among them:

$$\textit{Agg-union-max}(\pi_1, \pi_2, \ldots, \pi_n) = \bigcup_{\max} \pi_i \quad (8)$$

- Function *Agg-max*: this function aggregates a finite number multisets $\pi_1, \pi_2, \ldots, \pi_n$ by selecting the multiset that maximizes a particular cost function, $Cost_{\textit{Agg-max}}$:

$$\textit{Agg-max}(\pi_1, \pi_2, \ldots, \pi_n) = \pi_i \quad (9)$$

where $\pi_i \in \underset{\pi_j, j \in [1,\ldots,n]}{\arg\max} \; Cost_{\textit{Agg-max}}(\pi_j)$

When the function to aggregate is *Agg-max* it will be necessary to specify also the cost function $Cost_{\textit{Agg-max}}$.

Cost functions For defining cost functions we will use a new function, *Compact* : $M_{\mathcal{A}\times\mathcal{P}} \rightarrow M_{\mathcal{A}}$, that compacts a multiset of action-proposition pairs into a multiset of actions. Given a multiset of action-proposition pairs π, the multiset *Compact*(π) will contain an element for every different action in the pairs (a, p) of π. The multiplicity of the action a in the compacted multiset will be the sum of multiplicities of all the pairs containing a in the source multiset.

$$\textit{Compact}(\pi) = \{(a, m(a)) \quad | \quad ((a,p), m((a,p))) \in \pi, \; m(a) = \sum_p m((a,p))\}$$

- Function *Cost-single*: this function computes the cost of a multiset of action-proposition pairs, π, as the sum of costs of each different action in the compacted multiset *Compact*(π):

$$\textit{Cost-single}(\pi) = \sum_{(a,m(a)) \in \textit{Compact}(\pi)} cost(a) \quad (10)$$

- Function *Cost-multiple*: this function computes the cost of a multiset of action-proposition pairs, π, as the sum of the cost of each action in the compacted multiset multiplied by its multiplicity.

$$\textit{Cost-multiple}(\pi) = \sum_{(a,m(a)) \in \textit{Compact}(\pi)} m(a) \times cost(a) \quad (11)$$

- Function *Cost-single-eff*: this function computes the cost of a multiset of action-proposition pairs as the sum of the cost of each action a in a pair (a, p) divided by the number of positive effects of that action (*#add*(a)):

$$\textit{Cost-single-eff}(\pi) = \sum_{((a,p),m((a,p))) \in \pi} \frac{cost(a)}{\textit{\#add}(a)} \quad (12)$$

Instantiations

Under several conditions we will explain below, the following heuristics are instantiations of the generalized framework:[2]

- The *additive* and *max* heuristics (McDermott 1996; Bonet, Loerincs, & Geffner 1997), and their cost-based versions in (Haslum & Geffner 2001).
- The h^m family of heuristics (Haslum & Geffner 2000).
- The heuristic used by the FF planner (Hoffmann & Nebel 2001a) for STRIPS planning and its cost-based version implemented in METRIC-FF (Hoffmann 2003).
- The heuristics defined for the SAPA planner (Do & Kambhampati 2003) for cost-based planning in its *max* and *additive* versions.
- The level-based heuristic in (Fuentetaja, Borrajo, & Linares 2006).[3]
- The heuristic resulting of extracting a relaxed plan from the planning graph expanded by the SIMPLANNER (Sapena & Onaindía 2004).
- The *set-additive* heuristic, first defined in (Keyder & Geffner 2007b). This heuristic propagates sets of actions instead of multisets of actions. The set of actions for a given proposition is the union of two sets: a set containing the *best supporter* of the proposition, and the set of actions to generate the preconditions of that *best supporter* from the evaluated state. The *best supporter* of a proposition is the action (over all actions that generate the proposition) that minimizes the *Cost-single* function applied over the set containing the action and the supports of the preconditions.
- The h_a heuristic, defined in (Keyder & Geffner 2007a). The h_a heuristic is similar to the *set-additive* heuristic. The only difference is that the *best supporter* of each proposition is computed following the equations of the *additive* heuristic, instead of by minimizing the cost of a set of actions.
- The h_{pmax} heuristic, defined in (Mirkis & Domshlak 2007). This heuristic propagates vectors of costs with an element per proposition. The cost vector for an action contains for each proposition the maximum value for that proposition in the cost vectors of its preconditions. The cost vector of a proposition is the cost vector of the support action for which the sum of the elements of its cost vector is minimal. This heuristic considers the cost of each action divided by the number of its positive effects.

The particular instantiations are shown in Table 1. For STRIPS planning all cost functions are computed using unitary action costs. Cost functions whose name is followed by the word *unit* assume always unitary action costs (even when they are not unitary). Regarding the heuristic of the FF planner, we consider here a basic version. As it was explained

[2] Due to lack of space we can not describe each heuristic deeply. We refer the reader to the corresponding reference(s).

[3] The level-based heuristic is referred as v_3 in this paper.

Heuristic	m	Agg_{a_P}	Agg_h	$Cost_{a_P}$	$Cost_h$
additive	1	*Agg-union+*	*Agg-union+*	*Cost-multiple*	*Cost-multiple*
max h^m	1	*Agg-max* with *Cost-single*	*Agg-max* with *Cost-single*	*Cost-single*	*Cost-single*
h^m	> 1	*Agg-max* with *Cost-single*	*Agg-max* with *Cost-single*	*Cost-single*	*Cost-single*
FF-nse	1	*Agg-max* with *Cost-single*	*Agg-union+*	*Cost-single-unit*	*Cost-single*
level-based-max basic-sapa-max	1	*Agg-max* with *Cost-single*	*Agg-union+*	*Cost-single*	*Cost-single*
level-based-add basic-sapa-add h_a	1	*Agg-union+*	*Agg-union+*	*Cost-multiple*	*Cost-single*
set-additive	1	*Agg-union+*	*Agg-union+*	*Cost-single*	*Cost-single*
h_{pmax}	1	*Agg-union-max*	*Agg-union-max*	*Cost-single-eff*	*Cost-single-eff*

Table 1: Some instantiations of the parameters of the generalized formulation.

in (Hoffmann & Nebel 2001b; 2001a), the main difference between the FF heuristic and the *additive* heuristic is that FF captures some positive interactions between (sub)goals while the *additive* heuristic assumes (sub)goals to be independent. Some of these positive interactions are captured by the fact of computing a *relaxed plan*, that is represented as a list without duplicates (i.e. as a set). Thus, when the same action is used to achieve several (sub)goals it is only taken into account once. However, FF also captures another type of positive interactions. The relaxed plan extraction algorithm selects for each (sub)goal g the action that achieves the (sub)goal as early as possible in the relaxed planning graph (the *best supporter*). This happens always except in the case a previously selected action (for another (sub)goal g') achieves also g, and g' occurs for the first time in the relaxed planning graph at the same level as g, or at a level immediately after. In such a case, the previously selected action is also selected for g (i.e. the action selected for g is not its *best supporter*, but the *best supporter* of g'). We call this type of positive interactions *side effects*. To generate a (sub)goal with the *best supporter* selected to achieve another (sub)goal can be considered a side effect of that selection. FF does not detect all possible side effects due to the restriction over the levels of (sub)goals. Furthermore, the method can be considered somehow arbitrary, given that different actions will be selected depending on the order in which (sub)goals are selected, that in part depends on the order in which objects are defined in the problem file. Other works generalizing heuristics discussed in the related work section also used this heuristic without side effects. In our framework, to consider side-effects, equation (5) should be modified. A cost-based version of the FF heuristic was implemented in METRIC-FF. For the cost-based version, the relaxed plan in the same, but the final heuristic value is computed adding action costs. We call this heuristic, without considering side effects *FF-nse*.

The SAPA planner can deal with action costs and also with temporal planning problems. The planning model we define is not temporal. Therefore, we omit for this heuristic all parts of the SAPA algorithm dealing with temporal issues. We skip also other techniques described in (Do & Kambhampati 2003) to improve the heuristic, like the inclusion of static mutexes. The reason is that these techniques can also be applied for other heuristics as well, but actually they have not been applied yet, and we pursue a general framework. We refer to the basic cost propagation process to compute the heuristics in its *max* and *additive* versions described in (Bryce & Kambhampati 2007). We call these heuristics *basic-sapa-max(-add)*. The main difference between the algorithm for the SAPA heuristics and the algorithm of METRIC-FF is that the former propagates cost values in the planning graph. The idea is to propagate the best cost to achieve each proposition at each level. In the planning graph, each proposition has an associated value indicating its cost, and each action is also associated a value indicating the cost to support (cost to achieve its preconditions) and execute the action. Costs of propositions and actions are not correlated with the level in the planning graph they appear. The planning graph expansion does not always finish when all the goals are included in a layer, because costs can decrease when more layers are generated. One can choose to finish the planning graph at the first proposition level containing the goals, or to build n additional levels doing an *n-lookahead*. We consider here the case of ∞-*lookahead*. According to (Do & Kambhampati 2003), this case should produce the most accurate heuristic.

The *level-based* heuristic builds a relaxed planning graph in increasing levels of costs. The total cost of applying each action (and also each level cost) is computed performing a *max* propagation process over action preconditions. Another difference with the *basic-sapa-max* heuristic is that in the *level-based* heuristic the process of building the relaxed planning graph finishes at the first level the goals are present. From the procedural point of view this heuristic differs from the *basic-sapa-max* heuristic. However, from the declarative point of view both heuristics compute the same value if ties for selecting a unique action when the equations (5) and (9) generate a set of actions are broken in the same way.

Both heuristics appear in the same row in Table 1 as they are the same instantiation. The *level-based* heuristic doing *max* propagation is referred to as *level-based-max*.

Finally, SIMPLANNER does not extract a heuristic value from the relaxed planning graph it expands. Instead, it uses the relaxed planning graph for other purposes. However, the expansion of the planning graph is also performed in increasing levels of costs and can be used to derive heuristic estimates. In fact, it is equivalent to a *level-based* expansion, but with additive propagation (instead of *max* propagation). For this reason, we denote the heuristic derived from this type of expansion as *level-based-add*. We found that the heuristics *basic-sapa-max*, *level-based-add* and h_a can be generated using the same instantiation. As in the previous case, to obtain the same heuristic estimate, ties for selecting a unique action when the equation (5) generates a set of actions should be broken in the same way.

To the best of our knowledge, the only heuristics that implement a partial tie breaking policy for equation (5) are the heuristics of FF and METRIC-FF (other heuristics break ties arbitrarily).[4] This policy, called the *difficulty* heuristic in (Hoffmann & Nebel 2001a), selects among all the *best supporters*, the action(s) whose preconditions are easier to achieve. Different versions of the difficulty heuristic can be derived from the initial idea. For example, action costs can be taken also into account to consider that easier preconditions are less costly preconditions. In our generalized framework we did not formalize the difficulty heuristic. On one hand, because only two heuristics use it. On the other hand, because there are no reported experiments isolating the difficulty heuristic and showing how significant it is to use it. Anyway, introducing the idea of the difficulty heuristic in the generalized definition is straightforward.

Relationships between Heuristics

Interesting conclusions can be extracted from Table 1 about the relations among heuristics:

- Heuristics in the same row are equivalent (i.e. compute the same value) when ties in Equation (5) and in Equation (9) (for heuristics using *Agg-max*) are broken in the same way.
- The difference between the *additive* heuristic, the *set-additive* heuristic and the heuristics in the row of *level-based-add* lies on the instantiation of the cost functions. In fact, the only difference between the *set-additive* and the heuristics *level-based-add/basic-sapa-add/*h_a is that, in the former, the cost function to select a_P is *Cost-single*, while in the latter is *Cost-multiple*.
- The heuristics *level-based-add/basic-sapa-add/*h_a and the *additive* heuristic always choose the same action a_P to generate each needed proposition (i.e. the *best supporter* of each proposition always coincides), given that they use the same aggregation and cost functions to select that action. Furthermore, since the functions to compute Agg_h also coincide, the multiset for the goals will always be the same for both heuristics. The difference lies only on how the final heuristic value is extracted from that multiset (i.e. on the $Cost_h$ function). As $\forall \pi$ *Cost-single(*π*)* $\leq$ *Cost-multiple(*π*)*, the *level-based-add* heuristic always provides lower or equal values than the *additive* heuristic.
- The heuristic *max* and the heuristics *level-based-max/basic-sapa-max* always choose the same supporter action a_P for each needed proposition, since they also use the same aggregation and cost functions to select that action. When applied over the same multisets, the result of *Agg-union+* contains always the same or more actions that the result of *Agg-max*. Since both heuristics use also the same $Cost_h$ function, the *level-based-max* heuristic always provide higher or equal values than the *max* heuristic. The same happens for the heuristics *max* for unitary costs and *FF-nse*.
- The only difference between the heuristic *FF-nse* and the heuristics *level-based-max/basic-sapa-max* is the cost function used to select the supporter action (a_P) of each needed proposition. In the former, the cost function is *Cost-single-unit*, while in the latter is *Cost-single*.
- All heuristics whose Agg_h function is a union (additive or max), i.e. all but the *max* and h^m ones, as have been defined here, generate a multiset whose actions that can be ordered to be a *relaxed plan*, since the union (additive or max) maintains all required actions. For this, the order should observe the causality constraints. Once we have a relaxed plan, it is easy to derive from it the *helpful actions* (Hoffmann & Nebel 2001a) to give priority to most promising successors. Helpful actions are all actions that achieve propositions generated by actions in the relaxed plan that are applicable in the evaluated state and required by other actions in the relaxed plan.

Example

We show now an example of the different instantiations for $m = 1$. Suppose a relaxed problem with the structure in Figure 1. Propositions connected with each action at the left hand side are preconditions. Propositions connected on the right are positive effects. The value under each action is its cost. The initial state, $\mathcal{I}$, is the state to be evaluated, in which we only have the proposition p. Goals are t and u.

$\pi(\{p\}; \mathcal{I}; Agg) = \emptyset$ for any of the defined aggregation functions, Agg, since p belongs to $\mathcal{I}$. The multisets for actions a_1 (for $\{q\}$ and $\{r\}$) and a_2 (for $\{s\}$) are the same for the three aggregation functions defined, given that both actions only have the precondition p. Thus:

$$\pi(a_1; \{q\}; \mathcal{I}; Agg) = \{((a_1, q), 1)\}$$
$$\pi(a_1; \{r\}; \mathcal{I}; Agg) = \{((a_1, r), 1)\}$$
$$\pi(a_2; \{s\}; \mathcal{I}; Agg) = \{((a_2, s), 1)\}$$

The multisets for q and r are given by the action a_1, since a_1 is the only action that generates these propositions. For the same reason, the multiset for the proposition s is given by the action a_2. Thus:

$$\pi(\{q\}; \mathcal{I}; Agg) = \{((a_1, q), 1)\}$$

[4] The *max*, *additive* and h^m heuristics always provide the same value independently of the way ties are broken.

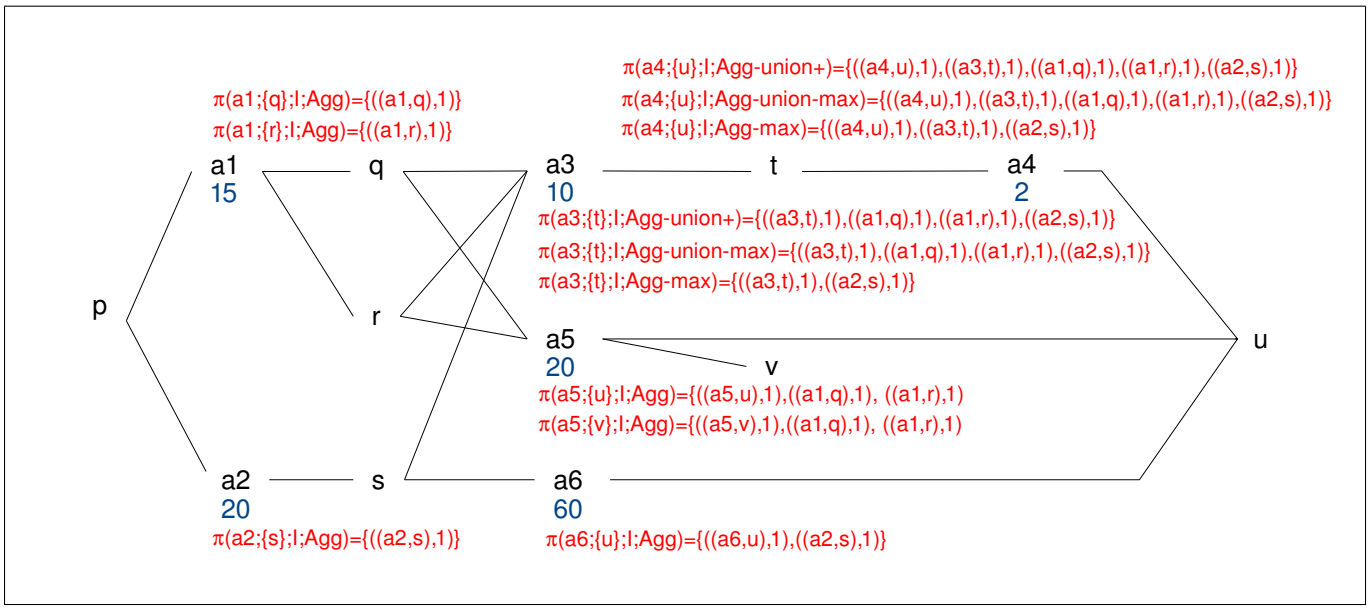

Figure 1: Example of relaxed domain to show the differences among the defined heuristics.

$$\pi(\{r\};\mathcal{I};Agg) = \{((a_1,r),1)\}$$
$$\pi(\{s\};\mathcal{I};Agg) = \{((a_2,s),1)\}$$

Next, the multisets for actions a_5 (for $\{u\}$) and a_6 (for $\{u\}$) are also the same for *Agg-union+*, *Agg-union-max* and *Agg-max*. a_6 only has a precondition, s. a_5 has two preconditions that are given by the same action, a_1. Thus:

$$\pi(a_5;\{u\};\mathcal{I};Agg) = \{((a_5,u),1),((a_1,q),1),((a_1,r),1)\}$$
$$\pi(a_6;\{u\};\mathcal{I};Agg) = \{((a_6,u),1),((a_2,s),1)\}$$

The multiset for the action a_3 (for $\{t\}$) depends on the aggregation function applied. Using *Agg-union+* and *Agg-union-max*, the multisets of the preconditions (q, r and s) are combined using the *additive-union* and the *max-union* respectively, and the result coincides, so:

$$\pi(a_3;\{t\};\mathcal{I};\textit{Agg-union+/Agg-union-max}) = \{((a_3,t),1),((a_1,q),1),((a_1,r),1),((a_2,s)1)\}$$

However, using *Agg-max* (with $Cost_{Agg\text{-}max}$ = *Cost-single*), we have that multisets of preconditions are aggregated taking the multiset which maximizes the function *Cost-single*. In this case, as a_2 is more expensive than a_1, the multiset that maximizes that function is $\{((a_2,s),1)\}$, so

$$\pi(a_3;\{t\};\mathcal{I};\textit{Agg-max}) = \{((a_3,t),1),((a_2,s),1)\}$$

Since a_3 is the only action that generates t, the multiset for t is the multiset for a_3. For a_4 we have the π multisets showed in Figure 1(right hand side, top).

Now, we have that the proposition u can be achieved using a_4, a_5 or a_6. Apart from the function Agg_h, the multiset associated with u depends also on the cost and aggregation functions for selecting the *best supporter* of each proposition: $Cost_{a_P}$ and Agg_{a_P}. The value of the $Cost_{a_P}$ function in each case is shown in Table 2 for the three cases of the function Agg_{a_p} (in this table we use *Agg-union* to refer to both *additive* and *max* union, since for both cases the multiset is the same). The value marked in bold is the minimum. The selected action is the action in the same row as this minimum. For example, the heuristics using $Cost_{a_p}$ = *Cost-single* and Agg_{a_p} = *Agg-union+*, select the action a_5 to achieve u, because the minimum in that case is 35, in the second row.

Table 3 shows for each analyzed heuristic: a_u, the *best supporter* of u (a_u corresponds to a_P in Equation 5); the multisets π for each goal, u and t; the multiset for the set of goals; and the final heuristic value (for simplicity reasons multisets have been compacted for all heuristics but for h_{pmax}, that is the only one whose cost function uses directly the action-proposition pairs multiset). For example, the heuristic *level-based-max* uses the function Agg_{a_P}= *Agg-max* with *Cost-single* to select the *best supporter* of u (a_u). So, the selected action (see Table 2) is a_4. Then, to compute $\pi(\{u\},\mathcal{I},Agg_h)$ and $\pi(\{t\},\mathcal{I},Agg_h)$, it uses Agg_h= *Agg-union+*. Thus, the multiset selected for u is the multiset of a_4 with *Agg-union+* (see Figure 1). For t, the multiset is the multiset of a_3 with *Agg-union+*. Then, the multiset for both goals, $\pi(\mathcal{G},\mathcal{I},Agg_h)$ is computed applying Agg_h = *Agg-union+*. Finally, the heuristic value is computed using $Cost_h$ = *Cost-single*. For cases where there is a tie, the table shows all possibilities. In this example, the more accurate heuristics are the ones in the row of *level-based-max* since they provide the optimal cost (47) to achieve the goals.

Multiset	*Cost-single*	*Cost-single-unit*	*Cost-multiple*	*Cost-multiple-unit*	*Cost-single-eff*
$\pi(a_4; \mathcal{I}; Agg\text{-}union/Agg\text{-}max)$	47/**32**	4/3	62/**32**	5/3	47/
$\pi(a_5; \mathcal{I}; Agg\text{-}union/Agg\text{-}max)$	**35**/35	**2/2**	**50**/50	3/3	**25/**
$\pi(a_6; \mathcal{I}; Agg\text{-}union/Agg\text{-}max)$	80/80	**2/2**	80/80	**2/2**	80/

Table 2: Cost values for the multisets of the actions generating u using *Agg-union* (*additive* and *max*) and *Agg-max*.

Heuristic	a_u	$\pi(\{u\}, \mathcal{I}, Agg_h)$	$\pi(\{t\}, \mathcal{I}, Agg_h)$	$\pi(\mathcal{G}, \mathcal{I}, Agg_h)$	$h(\mathcal{G}, \mathcal{I})$
additive	a_5	$\{(a_5,1),(a_1,2)\}$	$\{(a_3,1),(a_1,2),(a_2,1)\}$	$\{(a_5,1),(a_1,4),(a_3,1),(a_2,1)\}$	110
additive (unit. costs)	a_6	$\{(a_6,1),(a_2,1)\}$	$\{(a_3,1),(a_1,2),(a_2,1)\}$	$\{(a_6,1),(a_2,2),(a_3,1),(a_1,2)\}$	6
max h^1	a_4	$\{(a_4,1),(a_3,1),(a_2,1)\}$	$\{(a_3,1),(a_2,1)\}$	$\{(a_4,1),(a_3,1),(a_2,1)\}$	32
max	a_5	$\{(a_5,1),(a_1,2)\}$	$\{(a_3,1),(a_2,1)\}$	$\{(a_5,1),(a_1,2)\}$	2
h^1	a_5	$\{(a_5,1),(a_1,2)\}$	$\{(a_3,1),(a_2,1)\}$	$\{(a_3,1),(a_2,1)\}$	2
(unit. costs)	a_6	$\{(a_6,1),(a_2,1)\}$	$\{(a_3,1),(a_2,1)\}$	$\{(a_6,1),(a_2,1)\}$	2
	a_6	$\{(a_6,1),(a_2,1)\}$	$\{(a_3,1),(a_2,1)\}$	$\{(a_3,1),(a_2,1)\}$	2
FF-nse	a_5	$\{(a_5,1),(a_1,2)\}$	$\{(a_3,1),(a_1,2),(a_2,1)\}$	$\{(a_5,1),(a_1,4),(a_3,1),(a_2,1)\}$	65
	a_6	$\{(a_6,1),(a_2,1)\}$	$\{(a_3,1),(a_1,2),(a_2,1)\}$	$\{(a_6,1),(a_2,2),(a_3,1),(a_1,2)\}$	105
FF-nse	a_5	$\{(a_5,1),(a_1,2)\}$	$\{(a_3,1),(a_1,2),(a_2,1)\}$	$\{(a_5,1),(a_1,4),(a_3,1),(a_2,1)\}$	4
(unit. costs)	a_6	$\{(a_6,1),(a_2,1)\}$	$\{(a_3,1),(a_1,2),(a_2,1)\}$	$\{(a_6,1),(a_2,2),(a_3,1),(a_1,2)\}$	4
level-based-max basic-sapa-max	a_4	$\{(a_4,1),(a_3,1),(a_1,2),(a_2,1))\}$	$\{(a_3,1),(a_1,2),(a_2,1)\}$	$\{(a_4,1),(a_3,2),(a_1,4),(a_2,2)\}$	47
level-based-add basic-sapa-add	a_5	$\{(a_5,1),(a_1,2)\}$	$\{(a_3,1),(a_1,2),(a_2,1)\}$	$\{(a_5,1),(a_1,4),(a_3,1),(a_2,1)\}$	65
h_a set-additive	a_5	$\{(a_5,1),(a_1,2)\}$	$\{(a_3,1),(a_1,2),(a_2,1)\}$	$\{(a_5,1),(a_1,4),(a_3,1),(a_2,1)\}$	65
h_{pmax}	a_5	$\{((a_5,u),1),((a_1,q),1)),$ $((a_1,r),1)\}$	$\{((a_3,t),1),((a_1,q),1)),$ $((a_1,r),1),((a_2,s),1)\}$	$\{((a_5,u),1),((a_1,q),1),$ $((a_1,r),1),((a_3,t),1),$ $((a_2,s),1)\}$	55

Table 3: Heuristic values for different instantiations.

Related Work

Several previous efforts have been done in order to generalize planning heuristics. Rintanen (Rintanen 2006) presents a unified definition of heuristics for classical planning. The author generalizes the *max*, *additive* and FF heuristics (ignoring also side effects) to operators with conditional effects and arbitrary disjunctive preconditions. He uses propositional logic. This work differs from ours in several aspects. We use mathematical recursive equations, following the line of the original definitions of the *set-additive*, *max* and *additive* heuristics. On the other hand, we include also cost-based heuristics. An interesting conclusion of this work is the relation between the FF heuristic and the *max* heuristic, that we also mention in this paper. Another interesting conclusion is that planning graphs are not needed for defining the FF heuristic. This is also the case for all heuristics we relate in this paper. We have shown that all of them can be defined using a generalized definition without using planning graphs.

Mirkis and Domshlak (Mirkis & Domshlak 2007) define a computational generalized framework for cost-propagation over planning graphs. Planning graphs are treated as *waodags* (weighted and-or dags). The general cost propagation process is defined over *waodags* elements (nodes and edges). As in our work, the definition uses a mathematical model. The generalization they propose is more general than ours, given that one can propagate elements in any domain (scalars, vectors, sets, etc.). While this framework can be very useful to generate new heuristics, it is less useful to detect relations among the defined ones, because it is too general. In the extreme, the same heuristic can be computed propagating elements in very different domains. We think it is important to have common functions applied over elements in the same domain. Thus, in this work we unify the domain of the elements to be propagated to multisets of action-proposition pairs. Then, we generalize the functions to propagate such multisets. As action-proposition pairs multisets contain enough information to derive most of the existing heuristics, we have a unified view of them. Using this framework, heuristics can be compared by analyzing the particular instantiations of general functions that operate with multisets of action-proposition pairs.

Geffner and Helmert (Geffner 2007; Helmert & Geffner 2008) unify the *additive* and the causal graph heuristics (Helmert 2004; 2006). The result is the *context-enhanced additive* heuristic. This is the *additive* heuristic translated to the *multi-valued planning task* language (MPT) and endowed with context information. The context information is obtained by evaluating the cost of preconditions in different states. It would be interesting to study whether such or similar contexts could be included in our generalized framework.

Conclusions

We have introduced a general declarative definition of cost-based heuristics based on complete or partial delete relaxation, that propagates action-proposition pairs through actions (from preconditions to effects). Most of existing heuristics are instantiations of this framework, including those that typically have been defined using planning graphs. Heuristics differ on the aggregation and costs functions they use. Instantiating the generalized definition we have discovered important relations among heuristics. Under the same tie breaking policy, several heuristics are equivalent, and others are closely related.

The presented generalization allows one to compare heuristics from a theoretical point of view. From a computational point of view, propagating action-proposition pairs is not the most efficient way of computing heuristics. There are heuristics (as the *max* or *additive* ones) that only need to propagate scalars. Finally, from a experimental point of view, it can be important to fix the tie-breaking policy for all heuristics to be compared. This is the only way to avoid the results to be influenced by arbitrary behaviours.

The framework presented can also be used to generate new heuristics, since aggregation and costs function can be combined in different ways. Also, new functions could be defined. We also believe that deeply studying the properties of the aggregation and costs functions could be an interesting future work to derive more relations among heuristics values. For generating new heuristics, it would be interesting to modify common techniques, for example the best supporters could be selected using other criteria than minimizing a cost function.

Acknowledgements

This work has been partially supported by the Spanish MICINN project TIN2008-06701-C03-03 and the UC3M-CAM project CCG08-UC3M/TIC-4141.

References

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

Bryce, D., and Kambhampati, S. 2007. How to skin a planning graph for fun and profit: A tutorial on planning graph based reachability heuristics. *AI Magazine* 28 No. 1:47–83.

Do, M. B., and Kambhampati, S. 2003. Sapa: A scalable multi-objective heuristic metric temporal planner. *JAIR* 20:155–194.

Fuentetaja, R.; Borrajo, D.; and Linares, C. 2006. Improving relaxed planning graph heuristics for metric optimization. In *Workshop on Heuristic Search, Memory Based Heuristics and its Applications. AAAI*. AAAI Press.

Geffner, H. 2007. The causal graph heuristic is the additive heuristic plus context. In *Proc. of the Workshop on Heuristics for Domain-Independent Planning. ICAPS*.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *Artificial Intelligence Planning Systems*, 140–149.

Haslum, P., and Geffner, H. 2001. Heuristic planning with time and resources. In *Proc. of the 6th ECP*.

Helmert, M., and Geffner, H. 2008. Unifying the causal graph and additive heuristics. In *Proc. of the 18th ICAPS*.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proc. of the 14th ICAPS*, 161–170.

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

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

Hoffmann, J., and Nebel, B. 2001b. What makes the difference between HSP and FF? In *Proc. of the Workshop on Empirical Methods in AI. IJCAI*.

Hoffmann, J. 2003. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *JAIR* 20:291–341.

Keyder, E., and Geffner, H. 2007a. Heuristics for planning with action costs. In *Proc. of the 12th CAEPIA*. Springer.

Keyder, E., and Geffner, H. 2007b. Set-additive and TSP heuristics for planning with action costs and soft goals. In *Workshop on Heuristics for Domain-Independent Planning. ICAPS*.

McDermott, D. 1996. A heuristic estimator for means-ends analysis in planning. In *Proc. of the 3rd AIPS*, 142–149. AAAI Press.

Mirkis, V., and Domshlak, C. 2007. Cost-sharing approximations for h+. In *Proc. of the 17th ICAPS*, 240–247.

Rintanen, J. 2006. Unified definition of heuristics for classical planning. In *Proc. of the 17th ECAI*, 600–604. IOS Press.

Sapena, O., and Onaindía, E. 2004. Handling numeric criteria in relaxed planning graphs. In *Advances in Artificial Intelligence. IBERAMIA, LNAI 3315*, 114–123. Springer Verlag.