Unified Definition of Heuristics for Classical Planning

Jussi Rintanen[1]

Abstract. In many types of planning algorithms distance heuristics play an important role. Most of the earlier works restrict to STRIPS operators, and their application to a more general language with disjunctivity and conditional effects first requires an exponential size reduction to STRIPS operators.

I present direct formalizations of a number of distance heuristics for a general operator description language in a uniform way, avoiding the exponentiality inherent in earlier reductive approaches. The formalizations use formulae to represent the conditions under which operators have given effects. The exponentiality shows up in satisfiability tests with these formulae, but would appear to be a minor issue because of the small size of the formulae.

1 Introduction

Much of planning research has been carried out in the context of STRIPS operators, named after the famous planning system [3]. A STRIPS operator consists of three sets of facts: a precondition, an *add* list, and a *delete* list. There are many actions that cannot be expressed in terms of STRIPS operators only, for example anything with context-dependent effects, but it is well known that any classical planning problem (one initial state, deterministic actions) can be translated into an equivalent one with STRIPS operators only. However, this translation may increase the size of the planning problem exponentially, and therefore it is not desirable.

In this paper we use a general language for expressing classical planning problems, including conditional effects and disjunctive preconditions. We show that for many purposes a small toolbox of definitions based on the propositional logic is sufficient for handling a general operator definition rather elegantly. Specifically, we consider a number of *distance heuristics* developed for STRIPS planning, and generalize them to a much more expressive planning language.

Anecdotal evidence from the planning community tells that handling conditional effects and disjunctive preconditions elegantly is difficult. Indeed, no clear formalizations of the above heuristics – or many planning algorithms and other planning techniques – in the full generality of conditional effects and disjunctive preconditions have been published. Related works and planning system implementations rely on *ad hoc* techniques, for example on an exponential translation of all formulae into DNF and on eliminating conditional effects. We show how general operator definitions can be handled concisely and elegantly without an exponential blow up in size or processing time.

The outline of this paper is as follows. In Section 4.1 we generalize Bonet and Geffner's [1] max heuristic to operators with conditional effects and arbitrary disjunctive preconditions. A novelty here is that the heuristic is not defined as a cyclic recursion equation. In Section 4.2 we do the same to Bonet and Geffner's [1] additive heuristic, and in Section 4.3 to Hoffmann and Nebel's [7] *relaxed plan* heuristic.

[1] National ICT Australia, Canberra Research Laboratory, Canberra, Australia

2 Preliminaries

We give a definition of operators that corresponds to ADL/PDDL operators [4] that have been grounded which means that quantifiers *forall* and *exists* have been eliminated and all parameters have been instantiated with objects of appropriate types in all possible ways.

We use the classical propositional logic with the connectives $\vee, \wedge, \neg$ and the constant symbols *true* $\top$ and *false* $\bot$. The state variables of a planning problem are the atomic propositions of the logic. State variables $a \in A$ and their negations are *literals*. The *complement* $\overline{l}$ of a literal l is defined by $\overline{a} = \neg a$ and $\overline{\neg a} = a$ for all $a \in A$.

Definition 1. *Let A be a set of state variables. An operator over A is a pair $\langle c, e\rangle$ where c is a formula over A describing the precondition and e is an effect over A. Effects are recursively defined as follows.*

1. *$\top$ is an effect (the dummy effect).*
2. *a and $\neg a$ for state variables $a \in A$ are effects.*
3. *$e_1 \wedge \cdots \wedge e_n$ is an effect if $e_1, \ldots, e_n$ are effects over A (the special case with $n = 0$ is defined as the dummy effect $\top$.)*
4. *$c \rhd e$ is an effect if c is a formula and e is an effect over A.*

Conditional effects $c \rhd e$ mean that e is executed if c is true.

Definition 2 (Operator execution). *Let $o = \langle c, e\rangle$ be an operator over A and s a state (an assignment of truth values to A.) The operator is* executable *in s if $s \models c$ and $\{a, \neg a\} \not\subseteq [e]_s$ for all $a \in A$. The set $[e]_s$ of literals (the* active effects *of e in s) is defined as follows.*

1. *$[\top]_s = \emptyset$*
2. *$[a]_s = \{a\}$ for $a \in A$*
3. *$[\neg a]_s = \{\neg a\}$ for $a \in A$*
4. *$[e_1 \wedge \cdots \wedge e_n]_s = [e_1]_s \cup \ldots \cup [e_n]_s$*
5. *$[c \rhd e]_s = [e]_s$ if $s \models c$ and $[c \rhd e]_s = \emptyset$ otherwise.*

The successor state $app_o(s)$ of s under o is obtained from s by making the literals in $[e]_s$ true and retaining the values of state variables not occurring in $[e]_s$. For sequences $\sigma = o_1; o_2; \ldots; o_n$ of operators we define $app_\sigma(s) = app_{o_n}(\cdots app_{o_2}(app_{o_1}(s))\cdots)$.

Example 1. *Consider the operator $\langle a, e\rangle$ where $e = \neg a \wedge (\neg c \rhd \neg b)$ and a state s such that $s \models a \wedge b \wedge c$. The operator is executable because $s \models a$. Now $[e]_s = \{\neg a\}$ and $app_{\langle a,e\rangle}(s) \models \neg a \wedge b \wedge c$.* ■

A *problem instance* is a quadruple $\langle A, I, O, G\rangle$ where A is the set of state variables, I is the initial state, O is the set of operators and G is the goal formula. A sequence $\sigma = o_1; \ldots; o_n$ of operators is a *plan* for the problem instance if $app_\sigma(I) \models G$.

The formula $EPC_l(e)$ expresses the conditions under which the literal l is assigned *true* when the effect e is executed.

Definition 3. *Define the formula* $EPC_l(e)$ *recursively as follows.*

$$\begin{aligned} EPC_l(\top) &= \bot \\ EPC_l(l) &= \top \\ EPC_l(l') &= \bot \text{ when } l \neq l' \text{ (for literals } l') \\ EPC_l(e_1 \wedge \cdots \wedge e_n) &= EPC_l(e_1) \vee \cdots \vee EPC_l(e_n) \\ EPC_l(c \rhd e) &= c \wedge EPC_l(e) \end{aligned}$$

The case $EPC_l(e_1 \wedge \cdots \wedge e_n) = EPC_l(e_1) \vee \cdots \vee EPC_l(e_n)$ is defined as a disjunction because it is sufficient that at least one of the effects $e_1, \ldots, e_n$ makes l true. For example, $EPC_a(a \wedge b) \equiv \top$, $EPC_a(b) \equiv \bot$ and $EPC_a((b \rhd a) \wedge (c \rhd a)) \equiv b \vee c$.

We define a formula that indicates when an operator is executable so that a given literal becomes true.

Definition 4. *Let* A *be the set of state variables and* $o = \langle c, e \rangle$ *an operator over* A*. Define*

$$EPC_l(o) = c \wedge EPC_l(e) \wedge \bigwedge_{a \in A} \neg(EPC_a(e) \wedge EPC_{\neg a}(e)).$$

There is a connection between $EPC_l(e)$ and the definition of $[e]_s$.

Lemma 1. *Let* A *be the set of state variables,* s *a state over* A*,* l *a literal over* A*, and* o *an operator over* A *with the effect* e*. Then*

1. $l \in [e]_s$ *if and only if* $s \models EPC_l(e)$*, and*
2. $app_o(s)$ *is defined and* $l \in [e]_s$ *if and only if* $s \models EPC_l(o)$*.*

The proof of this lemma and some that follow are by structural induction and can be found in a technical report [9].

3 Reachability and Distances

The notion of reachability is important in defining whether a planning problem is solvable and in deriving techniques that speed up search for plans. Distances between states are closely related to reachability and will be defined next. Heuristics in Section 4 are approximations of distances. Define $img_o(S) = \{ app_o(s) | s \in S \}$.

Definition 5. *Let* I *be an initial state and* O *a set of operators. Define the* distance sets D_i^{fwd} *for* I, O *which consist of those states that are reachable from* I *by executing at most* i *operators.*

$$\begin{aligned} &D_0^{fwd} = \{I\} \\ &D_i^{fwd} = D_{i-1}^{fwd} \cup \{s | o \in O, s \in img_o(D_{i-1}^{fwd})\} \text{ for all } i \geq 1 \end{aligned}$$

Definition 6. *Given a state* I*, a set* O *of operators and the distance sets* $D_0^{fwd}, D_1^{fwd}, \ldots$ *for* I, O*,* the distance *of a state* s *from* I *is*

$$\delta_I^{fwd}(s) = \begin{cases} 0 \text{ if } s = I \\ i \text{ if } s \in D_i^{fwd} \setminus D_{i-1}^{fwd}. \end{cases}$$

If $s \notin D_i^{fwd}$ *for all* $i \geq 0$ *then* $\delta_I^{fwd}(s) = \infty$*. States that have a finite distance are* reachable *(from* I *with* O*).*

Distances can also be defined for formulae.

Definition 7. *The* distance $\delta_I^{fwd}(\phi)$ *of a formula* ϕ *is* $i \geq 1$ *if there is a state* s *such that* $s \models \phi$ *and* $\delta_I^{fwd}(s) = i$ *and there is no state* s' *such that* $s' \models \phi$ *and* $\delta_I^{fwd}(s') < i$*. If* $I \models \phi$ *then* $\delta_I^{fwd}(\phi) = 0$*.*

A formula ϕ has a finite distance iff $\langle A, I, O, \phi \rangle$ has a plan.

Reachability and distances are useful for implementing efficient planning systems. We mention two applications. First, if the precondition of an operator is not reachable, the operator can never be applied and can be ignored. Second, having an oracle that could tell the distances of states for free would directly yield a planning algorithm: step by step choose a sequence of operators that decrease the distance of the goal formula from the current state by one.

Computing distances is in the worst case just as difficult as planning itself (PSPACE-complete), and hence it is usually not useful to use exact distances. Instead, different kinds of *approximations* of distances and reachability can be used.

4 Distance Heuristics

Many approximations of distances are based on the following idea. Instead of considering the number of operators required to reach individual states, we compute approximate numbers of operators to reach states in which certain literals are true. So instead of distances of states, we use distances of literals. Within the computation of distances of literals, distances of formulae are estimated in terms of the distances of literals, which introduces an inaccuracy because the dependencies between literals are ignored. The heuristics therefore may underestimate the actual distance and only yield a lower bound for it.

4.1 Admissible Max Heuristic

Effective heuristics were presented by McDermott [8], Bonet et al. [2, 1] and Haslum and Geffner [6]. We first describe a generalization of the Bonet et al. *max* heuristics to our definition of operators.

We give a procedure that computes a lower bound on the number of operator applications that are needed for reaching from a state I a state in which certain literals are true. Sets D_i^{max} consist of literals that are true in all states that have a distance $\leq i$.

Definition 8. *Let* $L = A \cup \{\neg a | a \in A\}$ *be the set of literals over* A *and* I *a state. For all* $i \geq 1$ *define the* max-distance sets

$$\begin{aligned} &D_0^{max} = \{l \in L | I \models l\}, \text{ and} \\ &D_i^{max} = D_{i-1}^{max} \setminus \{l \in L | o \in O, D_{i-1}^{max} \cup \{EPC_{\bar{l}}(o)\} \text{ is satisfiable}\}. \end{aligned}$$

The computation starts from D_0^{max} which consists of all literals that are true in the initial state I. This set therefore characterizes those states that have distance 0 from the initial state.

Then we compute sets of literals characterizing states with distance 1, 2 and so on. Each set D_i^{max} is computed from the preceding set D_{i-1}^{max} by testing for each operator o whether it is executable in one of the distance $i-1$ states and whether it could make a literal l false. This is by testing whether $EPC_{\bar{l}}(o)$ is true in one of the distance $i-1$ states. If it is, l will not be included in D_i^{max}.

Since we consider only finite sets A of state variables and $|D_0^{max}| = |A|$ and $D_{i+1}^{max} \subseteq D_i^{max}$ for all $i \geq 0$, necessarily $D_i^{max} = D_j^{max}$ for some $i \leq |A|$ and all $j > i$.

Every state with distance $\leq i$ satisfies D_i^{max}.

Theorem 9. *Let* $D_i^{fwd}, i \geq 0$ *be the distance sets and* D_i^{max} *the max-distance sets for* I *and* O*. Then for all* $i \geq 0$*,* $D_i^{fwd} \subseteq \{s \in S | s \models D_i^{max}\}$ *where* S *is the set of all states.*

The sets D_i^{max} can be used for estimating the distances of formulae. The distance of a formula is the minimum of the distances of states that satisfy it.

Definition 10. *Let ϕ be a formula. Define*

$$\delta_I^{max}(\phi) = \begin{cases} 0 \text{ iff } D_0^{max} \cup \{\phi\} \text{ is satisfiable} \\ d \text{ iff } D_d^{max} \cup \{\phi\} \text{ is satisfiable and} \\ \quad D_{d-1}^{max} \cup \{\phi\} \text{ is not satisfiable, for } d \geq 1. \end{cases}$$

Lemma 2. *Let I be a state, O a set of operators, and $D_0^{max}, D_1^{max}, \ldots$ the max-distance sets for I and O. Then $app_{o_1;\ldots;o_n}(I) \models D_n^{max}$ for any operators $\{o_1, \ldots, o_n\} \subseteq O$.*

The next theorem shows that the distance estimates are a lower bound on the actual distances.

Theorem 11. *Let I be a state, O a set of operators, ϕ a formula, and $D_0^{max}, D_1^{max}, \ldots$ the max-distance sets for I and O. If $app_{o_1;\ldots;o_n}(I) \models \phi$, then $D_n^{max} \cup \{\phi\}$ is satisfiable.*

Proof. By Lemma 2 $app_{o_1;\ldots;o_n}(I) \models D_n^{max}$. By assumption $app_{o_1;\ldots;o_n}(I) \models \phi$. Hence $D_n^{max} \cup \{\phi\}$ is satisfiable. □

Corollary 12. *For a formula ϕ and a state I, if $o_1, \ldots, o_n$ is a sequence of operators and $app_{o_1;\ldots;o_n}(I) \models \phi$ then $n \geq \delta_I^{max}(\phi)$.*

The estimate $\delta_s^{max}(\phi)$ never overestimates the distance from s to ϕ and it is therefore an admissible heuristic. It may severely underestimate the distance, as discussed in Section 4.2. A family h^m of stronger admissible heuristics has been proposed by Haslum and Geffner [6]. The heuristic h^1 is the max heuristic. There is an algorithm for computing m-literal invariant clauses that generalizes the sets D^{max} to sets of m-literal clauses [9] and yields the h^m heuristic.

4.1.1 Distance Estimation in Polynomial Time

The computation of the sets D_i^{max} and the distances $\delta_I^{max}(\phi)$ is polynomial time except for the NP-complete satisfiability tests for $D \cup \{\phi\}$ where D is a set of literals. Here $\phi = EPC_l(o)$ for a literal l and an operator o. The size of ϕ is in the worst case quadratic in the size of o, and hence the formulae ϕ are typically small. The cost of the satisfiability test is exponential in the size of ϕ and linear in D, and hence the NP-completeness is almost never an issue.

However, to show that our approach is feasible even in the unrealistic case of huge operators we give a simple polynomial time approximation $\text{asat}(D, \phi)$ of the satisfiability tests with the property that if $D \cup \{\phi\}$ is satisfiable then $\text{asat}(D, \phi)$= true. The proof of Theorem 9 works when the satisfiability tests of $D \cup \{\phi\}$ are replaced by $\text{asat}(D, \phi)$. The approximation never leads to overestimating the distances, only underestimating, and for STRIPS operators the approximation never makes a difference and for general operators rarely. Define $\text{asat}(D, \phi)$ as follows.

$$\begin{aligned} \text{asat}(D, \bot) &= \text{false} \\ \text{asat}(D, \top) &= \text{true} \\ \text{asat}(D, a) &= \text{true iff } \neg a \notin D \ \ (\text{for } a \in A) \\ \text{asat}(D, \neg a) &= \text{true iff } a \notin D \ \ (\text{for } a \in A) \\ \text{asat}(D, \neg\neg\phi) &= \text{asat}(D, \phi) \\ \text{asat}(D, \phi_1 \vee \phi_2) &= \text{asat}(D, \phi_1) \text{ or } \text{asat}(D, \phi_2) \\ \text{asat}(D, \phi_1 \wedge \phi_2) &= \text{asat}(D, \phi_1) \text{ and } \text{asat}(D, \phi_2) \\ \text{asat}(D, \neg(\phi_1 \vee \phi_2)) &= \text{asat}(D, \neg\phi_1) \text{ and } \text{asat}(D, \neg\phi_2) \\ \text{asat}(D, \neg(\phi_1 \wedge \phi_2)) &= \text{asat}(D, \neg\phi_1) \text{ or } \text{asat}(D, \neg\phi_2) \end{aligned}$$

The cases for $\neg(\phi_1 \wedge \phi_2)$ and $\neg(\phi_1 \vee \phi_2)$ are respectively obtained from the cases for $\phi_1 \vee \phi_2$ and $\phi_1 \wedge \phi_2$ by the De Morgan laws.

The inaccuracy of the satisfiability tests stems from the fact that for formulae $\phi_1 \wedge \phi_2$ (respectively $\neg(\phi_1 \vee \phi_2)$) we make recursively two satisfiability tests that do not require that ϕ_1 and ϕ_2 (respectively $\neg\phi_1$ and $\neg\phi_2$) are *simultaneously* satisfiable. For example, $\text{asat}(\emptyset, a \wedge \neg a)$= true although $a \wedge \neg a$ is unsatisfiable.

Lemma 3. *Let ϕ be a formula and D a set of literals. If $D \cup \{\phi\}$ is satisfiable, then $\text{asat}(D, \phi)$ returns true.*

4.2 Inadmissible Additive Heuristic

The max heuristic is very optimistic about the distances, and in many cases it seriously underestimates them. If there are two goal literals, the maximum of the goal costs (distances) is taken to be the combined cost. This however is only accurate when the easier goal is achieved for free while achieving the more difficult one. Goals are often independent and then a more accurate estimate would be the sum of the individual costs. To fix this problem Bonet and Geffner [1] proposed a more practical variant of the max heuristic.

Our definition is based on an auxiliary definition for the cost of achieving a formula. The cost of a literal is the number of steps it takes to achieve it, in a similar approximative sense as in the *max* heuristic. The cost of a disjunction is the minimum cost of the disjuncts. The cost of a conjunction is the *sum* of the costs of the conjuncts. This is the difference to the *max* heuristic where the cost of a conjunction is the *maximum* of the costs of the conjuncts.

Definition 13. *Let I be a state and $L = A \cup \{\neg a | a \in A\}$ the set of literals. Define the sets D_i^+ for $i \geq 0$ as follows.*

$$\begin{aligned} D_0^+ &= \{l \in L | I \models l\} \\ D_i^+ &= D_{i-1}^+ \backslash \{l \in L | o \in O, cost(\, EPC_{\bar{l}}(o), i) < i\} \text{ for all } i \geq 1 \end{aligned}$$

We define $cost(\phi, i)$ as follows.

$$\begin{aligned} cost(\bot, i) &= \infty \\ cost(\top, i) &= 0 \\ cost(a, i) &= 0 \text{ if } \neg a \notin D_0^+, \text{ for } a \in A \\ cost(\neg a, i) &= 0 \text{ if } a \notin D_0^+, \text{ for } a \in A \\ cost(a, i) &= j \text{ if } \neg a \in D_{j-1}^+ \backslash D_j^+ \text{ for some } j < i \\ cost(\neg a, i) &= j \text{ if } a \in D_{j-1}^+ \backslash D_j^+ \text{ for some } j < i \\ cost(a, i) &= \infty \text{ if } \neg a \in D_j^+ \text{ for all } j < i \\ cost(\neg a, i) &= \infty \text{ if } a \in D_j^+ \text{ for all } j < i \\ cost(\phi_1 \vee \phi_2, i) &= \min(cost(\phi_1, i), cost(\phi_2, i)) \\ cost(\phi_1 \wedge \phi_2, i) &= cost(\phi_1, i) + cost(\phi_2, i) \\ cost(\neg\neg\phi, i) &= cost(\phi, i) \\ cost(\neg(\phi_1 \wedge \phi_2), i) &= \min(cost(\neg\phi_1, i), cost(\neg\phi_2, i)) \\ cost(\neg(\phi_1 \vee \phi_2), i) &= cost(\neg\phi_1, i) + cost(\neg\phi_2, i) \end{aligned}$$

The above definitions would appear to be cyclic but the definition of D_i^+ refers to $\text{cost}(\phi, j)$ for $j \leq i$ only and the definition of $\text{cost}(\phi, i)$ refers to D_j^i for $j < i$ only.

The definition of $\text{cost}(\phi, i)$ ignores dependencies between conjuncts in the same way as the definition $\text{asat}(D, \phi)$.

Definition 14. *Let ϕ be a formula. Define*

$$\delta_I^+(\phi) = cost(\phi, n)$$

where n is the smallest i such that $D_i^+ = D_{i-1}^+$.

The following theorem shows that the distance estimates of the additive heuristic are never lower than those of the max heuristic.

Theorem 15. *Let $D_i^{max}, i \geq 0$ be the sets defined in terms of the approximate tests $\text{asat}(D, \phi)$. Then $D_i^{max} \subseteq D_i^+$ for all $i \geq 0$.*

Proof. The proof is by induction on i.

Base case $i = 0$: By definition $D_0^+ = D_0^{max}$.

Inductive case $i \geq 1$: By the induction hypothesis $D_{i-1}^{max} \subseteq D_{i-1}^+$. To show $D_{i-1}^{max} \backslash \{l \in L | o \in O, \text{asat}(D_{i-1}^{max}, EPC_{\bar{l}}(o))\} \subseteq D_{i-1}^+ \backslash \{l \in L | o \in O, \text{cost}(EPC_{\bar{l}}(o), i) < i\}$ it is sufficient to show that $\text{cost}(EPC_{\bar{l}}(o), i) < i$ implies $\text{asat}(D_{i-1}^{max}, EPC_{\bar{l}}(o))$.

We show this by induction on the structure of $\phi = EPC_{\bar{l}}(o)$.

Induction hypothesis: $\text{cost}(\phi, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \phi)$=true.

Base case 1, $\phi = \bot$: $\text{cost}(\bot, i) = \infty$ and $\text{asat}(D_i^{max}, \bot)$=false.

Base case 2, $\phi = \top$: $\text{cost}(\top, i) = 0$ and $\text{asat}(D_i^{max}, \top)$=true.

Base case 3, $\phi = a$: If $\text{cost}(a, i) < i$ then $\neg a \notin D_j^+$ for some $j < i$ or $\neg a \notin D_0^+$. Hence $\neg a \notin D_{i-1}^+$. By the outer induction hypothesis $\neg a \notin D_{i-1}^{max}$ and consequently $\neg a \notin D_i^{max}$. Hence $\text{asat}(D_i^{max}, a)$=true.

Base case 4, $\phi = \neg a$: Analogous to the case $\phi = a$.

Inductive case 5, $\phi = \phi_1 \vee \phi_2$: Assume $\text{cost}(\phi_1 \vee \phi_2, i) < i$. Since $\text{cost}(\phi_1 \vee \phi_2, i) = \min(\text{cost}(\phi_1, i), \text{cost}(\phi_2, i))$, either $\text{cost}(\phi_1, i) < i$ or $\text{cost}(\phi_2, i) < i$. By the induction hypothesis $\text{cost}(\phi_1, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \phi_1)$, and $\text{cost}(\phi_2, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \phi_2)$. Hence either $\text{asat}(D_{i-1}^{max}, \phi_1)$ or $\text{asat}(D_{i-1}^{max}, \phi_2)$. Therefore by definition $\text{asat}(D_{i-1}^{max}, \phi_1 \vee \phi_2)$.

Inductive case 6, $\phi = \phi_1 \wedge \phi_2$: Assume $\text{cost}(\phi_1 \wedge \phi_2, i) < i$. Since $i \geq 1$ and $\text{cost}(\phi_1 \vee \phi_2, i) = \text{cost}(\phi_1, i) + \text{cost}(\phi_2, i)$, both $\text{cost}(\phi_1, i) < i$ and $\text{cost}(\phi_2, i) < i$. By the induction hypothesis $\text{cost}(\phi_1, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \phi_1)$, and $\text{cost}(\phi_2, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \phi_2)$. Hence both $\text{asat}(D_{i-1}^{max}, \phi_1)$ an $\text{asat}(D_{i-1}^{max}, \phi_2)$. Therefore by definition $\text{asat}(D_{i-1}^{max}, \phi_1 \wedge \phi_2)$.

Inductive case 7, $\phi = \neg\neg\phi_1$: By the induction hypothesis $\text{cost}(\phi_1, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \phi_1)$. By definition $\text{cost}(\neg\neg\phi_1, i) = \text{cost}(\phi_1, i)$ and $\text{asat}(D, \neg\neg\phi) = \text{asat}(D, \phi)$. By the induction hypothesis $\text{cost}(\neg\neg\phi_1, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \neg\neg\phi_1)$.

Inductive case 8, $\phi = \neg(\phi_1 \vee \phi_2)$: Analogously to $\phi = \phi_1 \wedge \phi_2$.
Inductive case 9, $\phi = \neg(\phi_1 \wedge \phi_2)$: Analogously to $\phi = \phi_1 \vee \phi_2$. □

That the additive heuristic gives higher estimates than the max heuristic could in many cases be viewed as an advantage because the estimates would be more accurate. However, sometimes this leads to overestimating the actual distance, and the heuristic is therefore not admissible (but see recent work by Haslum et al. [5].)

Example 2. *Consider an initial state such that $I \models \neg a \wedge \neg b \wedge \neg c$ and the operator $\langle \top, a \wedge b \wedge c \rangle$. A state satisfying $a \wedge b \wedge c$ is reached by this operator in one step but $\delta_I^+(a \wedge b \wedge c) = 3$.* ■

4.3 Inadmissible Relaxed Plan Heuristic

The max heuristic and the additive heuristic represent two extremes. The first assumes that sets of operators required for reaching the different goal literals maximally overlap in the sense that the operators for the most difficult goal literal include the operators for all the remaining ones. The second assumes that these sets are completely disjoint. In some cases the first is better and in others the second.

To estimate the distances more accurately the operators should be better taken into account, which suggests yet another approach to computing a heuristic: attempt to find a set of operators that in a very loose sense reaches the goals. This idea has been considered by Hoffman and Nebel [7]. A *relaxed plan* is computed as follows (see the algorithm in Figure 1.) We first choose a set of goal literals the truth of which is sufficient for the truth of the goal formula G. These literals must be reachable in the sense of the sets D_i^{max} from Section 4.1. Then we identify the goal literals l with the highest distance (in the D_i^{max} sense) and a set of operators making them true. These operators form the last step of the relaxed plan. A new goal formula represents the conditions under which these operators can make the literals true. Then a new set of goal literals is produced by a form of regression from the new goal formula. The computation is repeated until we have a set of goal literals that are true in the initial state. Along the way we have constructed a relaxed plan.

The function $\text{goals}(D, \phi)$ recursively finds a set M of literals such that $M \models \phi$ and each literal in M is consistent with D. Note that M itself is not necessarily consistent, for example for $D = \emptyset$ and $\phi = a \wedge \neg a$ we get $M = \{a, \neg a\}$. If a set M is found $\text{goals}(D, \phi) = \{M\}$ and otherwise $\text{goals}(D, \phi) = \emptyset$.

Definition 16. *Let D be a set of literals.*

$$
\begin{aligned}
goals(D, \bot) &= \emptyset \\
goals(D, \top) &= \{\emptyset\} \\
goals(D, a) &= \{\{a\}\} \text{ if } \neg a \notin D \\
goals(D, a) &= \emptyset \text{ if } \neg a \in D \\
goals(D, \neg a) &= \{\{\neg a\}\} \text{ if } a \notin D \\
goals(D, \neg a) &= \emptyset \text{ if } a \in D \\
goals(D, \neg\neg\phi) &= goals(D, \phi) \\
goals(D, \phi_1 \vee \phi_2) &= \begin{cases} goals(D, \phi_1) \text{ if } goals(D, \phi_1) \neq \emptyset \\ goals(D, \phi_2) \text{ otherwise} \end{cases} \\
goals(D, \phi_1 \wedge \phi_2) &= \begin{cases} \{L_1 \cup L_2\} & \text{if } goals(D, \phi_1) = \{L_1\} \\ & \text{and } goals(D, \phi_2) = \{L_2\} \\ \emptyset & \text{otherwise} \end{cases} \\
goals(D, \neg(\phi_1 \wedge \phi_2)) &= \begin{cases} goals(D, \neg\phi_1) \text{ if } goals(D, \neg\phi_1) \neq \emptyset \\ goals(D, \neg\phi_2) \text{ otherwise} \end{cases} \\
goals(D, \neg(\phi_1 \vee \phi_2)) &= \begin{cases} \{L_1 \cup L_2\} & \text{if } goals(D, \neg\phi_1) = \{L_1\} \\ & \text{and } goals(D, \neg\phi_2) = \{L_2\} \\ \emptyset & \text{otherwise} \end{cases}
\end{aligned}
$$

If both ϕ_1 and ϕ_2 yield goal literals for $\phi_1 \vee \phi_2$ the set for ϕ_1 is chosen. A practically better choice is the smaller of the two sets.

Lemma 4. *Let D be a set of literals and ϕ a formula.*

1. *$goals(D, \phi) \neq \emptyset$ if and only if $asat(D, \phi) = true$.*
2. *If $goals(D, \phi) = \{M\}$ then $\{\bar{l} | l \in M\} \cap D = \emptyset$ and $asat(D, \bigwedge_{l \in M} l) = true$.*

Proof. 1. This is an easy induction proof on the structure of ϕ based on the definitions of $\text{asat}(D, \phi)$ and $\text{goals}(D, \phi)$.
2. This is because $\bar{l} \notin D$ for all $l \in M$. This can be shown by a simple induction proof. □

Lemma 5. *Let D and $D' \subseteq D$ be sets of literals. If $goals(D, \phi) = \emptyset$ and $goals(D', \phi) = \{M\}$, then there is $l \in M$ such that $\bar{l} \in D \backslash D'$.*

Definition 17. *Define $\delta_I^{rlx}(\phi) = relaxedplan(A, I, O, \phi)$.*

The additive and the relaxed plan heuristic are incomparable and both give estimates at least as high as the max heuristic.

Theorem 18. *Let ϕ be a formula and $\delta_I^{max}(\phi)$ the max-distance defined in terms of $asat(D, \phi)$. Then $\delta_I^{rlx}(\phi) \geq \delta_I^{max}(\phi)$.*

Proof. We have to show that for any formula G the procedure call *relaxedplan*(A,I,O,G) returns a number $\geq \delta_I^{max}(G)$.

The procedure returns ∞ if and only if $\text{asat}(D_i^{max}, G) = \text{false}$ for all $i \geq 0$. In this case by definition $\delta_I^{max}(G) = \infty$. Otherwise $t = \delta_I^{max}(G)$. Now $t = 0$ if and only if $\text{asat}(D_0^{max}, G) = \text{true}$. In this case the procedure returns 0 without iterating the loop (line 9.)

1: **procedure** relaxedplan(A,I,O,G);
2: $L := A \cup \{\neg a | a \in A\}$; *(* Set of all literals *)*
3: compute sets D_i^{max} as in Definition 8;
4: **if** asat(D_i^{max}, G) = false for all $i \geq 0$ **then return** ∞;
5: $t := \delta_I^{max}(G)$;
6: $L_{t+1}^G := \emptyset$; *(* Goal literals initially *)*
7: $N_{t+1} := \emptyset$;
8: $G_t := G$; *(* Initial goal formula *)*
9: **for** $i := t$ **downto** 1 **do**
10: $L_i^G := (L_{i+1}^G \backslash N_{i+1}) \cup \{l \in M | M \in \text{goals}(D_i^{max}, G_i)\}$;
11: $N_i := \{l \in L_i^G | \bar{l} \in D_{i-1}^{max}\}$; *(* Goals becoming true at i *)*
12: T_i := a minimal subset of O *(* Operators making N_i true *)*
13: so that $N_i \subseteq \{l \in L | o \in T_i, \text{asat}(D_{i-1}^{max}, EPC_l(o))\}$;
14: $G_{i-1} := \bigwedge_{l \in N_i} \bigvee\{EPC_l(o) | o \in T_i\}$; *(* New goal *)*
15: **end do**
16: **return** $|T_1| + |T_2| + \cdots + |T_t|$;

Figure 1. Algorithm for finding a relaxed plan

We show that if $t \geq 1$ then $T_i \neq \emptyset$ for every $i \in \{1, \ldots, t\}$, entailing $|T_1| + \cdots + |T_t| \geq t = \delta_I^{max}(G)$. This is by induction from t to 1. We use the following auxiliary result. If asat(D_{i-1}^{max}, G_i) = false and asat(D_i^{max}, G_i) = true and $\bar{l} \notin D_i^{max}$ for all $l \in L_i^G$ then T_i is well-defined and $T_i \neq \emptyset$. The proof is as follows.

> By Lemma 4 goals$(D_{i-1}^{max}, G_i) = \emptyset$ and goals$(D_i^{max}, G_i) = \{M\}$ for some M. By Lemma 5 there is $l \in M$ such that $\bar{l} \in D_{i-1}^{max}$ and hence $N_i \neq \emptyset$. By definition $\bar{l} \in D_{i-1}^{max}$ for all $l \in N_i$. By $N_i \subseteq L_i^G$ and the assumption about L_i^G $\bar{l} \notin D_i^{max}$ for all $l \in N_i$. Hence $\bar{l} \in D_{i-1}^{max} \backslash D_i^{max}$ for all $l \in N_i$. Hence by definition of D_i^{max} for every $l \in N_i$ there is $o \in O$ such that asat$(D_{i-1}^{max}, EPC_l(o))$. Hence there is $T_i \subseteq O$ so that $N_i \subseteq \{l \in L | o \in T_i, \text{asat}(D_{i-1}^{max}, EPC_l(o))\}$ and the value of T_i is defined. As $N_i \neq \emptyset$ also $T_i \neq \emptyset$.

In the induction proof we establish the assumptions of the auxiliary result and then invoke the auxiliary result itself.

Induction hypothesis: For all $j \in \{i, \ldots, t\}$

1. $\bar{l} \notin D_j^{max}$ for all $l \in L_j^G$,
2. asat(D_j^{max}, G_j) = true and asat(D_{j-1}^{max}, G_j) = false, and
3. $T_j \neq \emptyset$.

Base case $i = t$:

1. $\bar{l} \notin D_t^{max}$ for all $l \in L_t^G$ by (2) of Lemma 4 because $L_t^G = \{l \in \text{goals}(D_t^{max}, G_t)\}$.
2. As $t = \delta_I^{max}(G_t)$ by definition asat(D_{t-1}^{max}, G_t) = false and asat(D_t^{max}, G_t) = true.
3. By the auxiliary result from the preceding case.

Inductive case $i < t$:

1. We have $\bar{l} \notin D_i^{max}$ for all $l \in L_i^G$ because $L_i^G = (L_{i+1}^G \backslash N_{i+1}) \cup \{l \in \text{goals}(D_i^{max}, G_i)\}$ and by the induction hypothesis $\bar{l} \notin D_{i+1}^{max}$ for all $l \in L_{i+1}^G$ and by (2) of Lemma 4 $\bar{l} \notin D_i^{max}$ for all $l \in M$ for $M \in \text{goals}(D_i^{max}, G_i)$.
2. By definition $G_i = \bigwedge_{l \in N_{i+1}} \bigvee\{EPC_l(o) | o \in T_{i+1}\}$. By definition of T_{i+1} for every $l \in N_{i+1}$ there is $o \in T_{i+1}$ such that asat$(D_i^{max}, EPC_l(o))$ = true. By definition of asat$(D_i^{max}, \phi_1 \vee \phi_2)$ and asat$(D_i^{max}, \phi_1 \wedge \phi_2)$ also asat(D_i^{max}, G_i) = true.
   Then we show that asat(D_{i-1}^{max}, G_i) = false. By definition of D_i^{max}, asat$(D_{i-1}^{max}, EPC_{\bar{l}}(o))$ = false for all $l \in D_i^{max}$ and $o \in O$. Hence asat$(D_{i-1}^{max}, EPC_l(o))$ = false for all $l \in N_{i+1}$ and $o \in O$ because $\bar{l} \in D_i^{max}$. Hence asat$(D_{i-1}^{max}, EPC_l(o))$ = false for all $l \in N_{i+1}$ and $o \in T_{i+1}$ because $T_{i+1} \subseteq O$. By definition $G_i = \bigwedge_{l \in N_{i+1}} \bigvee\{EPC_l(o) | o \in T_{i+1}\}$. Hence by definition of asat(D, ϕ) also asat(D_{i-1}^{max}, G_i) = false.
3. By the auxiliary result from the preceding case. □

5 Conclusions

We have shown how a number of distance heuristics can be elegantly generalized to an expressive planning language with conditional effects and arbitrary formulae. In comparison to the reductive approach for handling general operators, our approach does not suffer from the problem of exponential size of operator sets. The only exponentiality is in the satisfiability tests, but as the structure of formulae $EPC_l(o)$ is usually very simple even in cases that are difficult to the reductive approach, the worst-case exponentiality is not a serious issue, and can be completely avoided by approximate satisfiability tests.

We have also clarified relations between different heuristics. Hoffmann and Nebel [7] do not elaborate on the close connection between their Graphplan-based relaxed plan heuristic and Bonet and Geffner's max heuristic. Our formalization of the former makes the connection very clear: a relaxed plan is a set of operator occurrences that in a certain sense covers the changes between sets D_i^{max} and D_{i+1}^{max} that are relevant for reaching the goals. Planning graphs are not needed for defining the relaxed plan heuristic.

Acknowledgements

This research was supported by National ICT Australia (NICTA). in connection with the DPOLP project. NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian National Research Council.

REFERENCES

[1] Blai Bonet and Héctor Geffner, 'Planning as heuristic search', *Artificial Intelligence*, **129**(1-2), 5–33, (2001).
[2] Blai Bonet, Gábor Loerincs, and Héctor Geffner, 'A robust and fast action selection mechanism for planning', in *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97) and 9th Innovative Applications of Artificial Intelligence Conference (IAAI-97)*, pp. 714–719, Menlo Park, California, (July 1997). AAAI Press.
[3] Richard E. Fikes and Nils J. Nilsson, 'STRIPS: a new approach to the application of theorem proving to problem solving', *Artificial Intelligence*, **2**(2-3), 189–208, (1971).
[4] Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins, 'PDDL - the Planning Domain Definition Language, version 1.2', Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University, (October 1998).
[5] Patrik Haslum, Blai Bonet, and Héctor Geffner, 'New admissible heuristics for domain-independent planning', in *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-2005)*, pp. 1163–1168, (2005).
[6] Patrik Haslum and Héctor Geffner, 'Admissible heuristics for optimal planning', in *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, eds., Steve Chien, Subbarao Kambhampati, and Craig A. Knoblock, pp. 140–149. AAAI Press, (2000).
[7] J. Hoffmann and B. Nebel, 'The FF planning system: Fast plan generation through heuristic search', *Journal of Artificial Intelligence Research*, **14**, 253–302, (2001).
[8] Drew V. McDermott, 'Using regression-match graphs to control search in planning', *Artificial Intelligence*, **109**(1–2), 111–159, (1999).
[9] Jussi Rintanen, 'State-space traversal techniques for planning', Report 220, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, (2005).