Accuracy of Admissible Heuristic Functions in Selected Planning Domains

Malte Helmert and **Robert Mattmüller**
Albert-Ludwigs-Universität Freiburg
{helmert,mattmuel}@informatik.uni-freiburg.de

Abstract

The efficiency of optimal planning algorithms based on heuristic search crucially depends on the accuracy of the heuristic function used to guide the search. Often, we are interested in domain-independent heuristics for planning. In assessing the limitations of domain-independent heuristic planning, it appears interesting to analyse the (in)accuracy of common domain-independent planning heuristics in the IPC benchmark domains. For a selection of these domains, we analytically investigate the accuracy of the h^+ heuristic, the h^k family of heuristics, and certain (additive) pattern database heuristics, compared to the optimal heuristic h^*. Whereas h^+ and additive pattern database heuristics usually return cost estimates proportional to the true cost, non-additive h^k and non-additive pattern-database heuristics can yield results underestimating the true cost by arbitrarily large factors.

Introduction

Heuristic search with A^* and similar algorithms remains the most popular method for optimal sequential planning, with significant effort spent on perfecting old heuristic estimators (Haslum *et al.* 2007) or deriving new ones (Helmert, Haslum, & Hoffmann 2007). While methods not based on state-space search have achieved remarkable success in addressing related problems, such as optimal parallel planning (Kautz & Selman 1996; 1999), the state of the art in optimal sequential planning is still defined by heuristic search methods almost exclusively, with symbolic state-space exploration (Edelkamp & Helmert 2001) being the only non-classical approach that can outperform heuristic search in some benchmark domains.

Considering the important role of admissible heuristics for optimal sequential planning, the question arises how to evaluate the quality of a given heuristic. A popular method is to run a planning algorithm against a number of benchmark tasks and count the number of node expansions performed by a search algorithm. The fewer nodes an algorithm expands, the better.

While experiments of this kind are certainly useful, there are some questions they cannot address. In particular, their results can almost exclusively be interpreted with *relative*, i. e., *comparative* statements: "Heuristic h expands fewer nodes than heuristic h' for benchmark suite X". Unless experiments show polynomial scaling behaviour on a family of benchmark tasks of growing size, which they very rarely do, the data usually does not lend itself to *absolute* statements of the type "Heuristic h is well-suited for solving problems from benchmark suite X."

In this contribution, we address this issue by providing absolute quality results for certain popular planning heuristics on some popular benchmark domains taken from the first four International Planning Competitions (McDermott 2000; Bacchus 2001; Long & Fox 2003; Hoffmann & Edelkamp 2005), in the form of comparisons to the optimal heuristic function h^*.

Planning Domains

We consider the planning domains GRIPPER, LOGISTICS, BLOCKSWORLD, MICONIC-STRIPS, MICONIC-SIMPLE-ADL, SCHEDULE and SATELLITE. Familiarity with the domains is assumed. For an overview, see e. g. Helmert's Ph. D. thesis (Helmert 2006).

Heuristics

We compare the accuracy of the heuristics h^+, h^k, and (additive) pattern database heuristics relative to the optimal heuristic.

A heuristic h maps states s to estimates of the true goal cost. Whenever the planning task to which s belongs, including the available operators and the goal description, is clear from the context, we will simply write $h(s)$ without explicitly mentioning operators or goal.

The h^* Heuristic The h^* heuristic assigns to each state s of a planning task $\mathcal{T}$ the length of a shortest plan leading from s to a goal state of $\mathcal{T}$, i. e., h^* is the *optimal heuristic*.

In the language of approximation complexity (Ausiello *et al.* 1999), computing the optimal heuristic for the initial state of a planning task is the evaluation problem counterpart of the PLANLEN decision problem. It is PSPACE-equivalent in general (Bylander 1994), but easier for a fixed domain. For the domains we consider, the problem is mostly NP-equivalent, with the exception of GRIPPER and SCHEDULE, where it is polynomial (Helmert 2006).

The h^+ Heuristic The h^+ heuristic (McDermott 1996; Bonet & Geffner 2001; Hoffmann 2005) assigns to each state s of a planning task $\mathcal{T}$ the length of a shortest plan leading from s to a goal state in the relaxed problem $\mathcal{T}^+$.

Evaluating h^+ is NP-equivalent in general (Bylander 1994), but easier for many of the domains we consider. We believe that the problem is NP-equivalent for MICONIC-SIMPLE-ADL and SATELLITE, and polynomial for all others.[1]

The h^k Family of Heuristics The h^k, $k = 1, 2, \ldots$, family of heuristics (Haslum & Geffner 2000) is based on the relaxation where the cost of reaching a state with n satisfied atoms is approximated by the highest cost of reaching a subset with at most k satisfied atoms. It can be computed in polynomial time, with the degree of the polynomial depending on k.

Recently, the h^k family has been extended by introducing *additive h^k heuristics* (Haslum, Bonet, & Geffner 2005). The results we present for the h^k family do not apply to their additive counterparts; their study is left as future work.

Pattern Database Heuristics Pattern database heuristics (Culberson & Schaeffer 1998) are computed by projecting the state space onto a subset of the state variables and computing the shortest plan in the resulting abstract state space. As these abstractions are solution preserving, abstract solutions are never longer than corresponding concrete ones, making the heuristic admissible. In the case of pattern database heuristics, unlike in the other cases, we will argue about multi-valued state variable representations instead of binary ones.

Given a task $\mathcal{T}$, possible pattern database heuristics for $\mathcal{T}$ are parametrised by the set of variables retained in the abstract problem (the *pattern*). Another parameter more closely related to the parameter k of the h^k family is the maximum number of state variables to be retained. Keeping that number K small is important as the number of states in the abstract problem is exponential in K, and so is the time to compute and the memory to store a pattern database for K variables. Pattern database heuristics can be computed in polynomial time in the task size iff $K \in O(\log \|\mathcal{T}\|)$. We will in the following adhere to this restriction.

Additive Pattern Database Heuristics With a pattern database consisting of tables for more than one pattern it is possible to get a more accurate cost estimate by maximizing over the corresponding entries in the tables. If *summing up* these entries is still guaranteed to yield a value not greater than the true cost, i. e., if the heuristic obtained by summation is admissible, the set of sets of variables defining the patterns and the corresponding databases are called *additive* (Haslum *et al.* 2007). Since this admissibility criterion cannot be checked efficiently, it is commonly replaced by a simple syntactic test that is sufficient but not necessary for additivity: Two patterns are considered additive iff no operator modifies variables from both patterns. (Note that this implies that additive patterns are disjoint, unless there are variables which can never be modified.)

An additive pattern database heuristic assigns to each state such a sum of additive pattern values. As in the case of regular pattern databases, we must restrict the size of *individual* patterns by $\Theta(\log \|\mathcal{T}\|)$ for polynomial-time computability.

Note that it is common to not just consider *one* sum of individual pattern heuristics, but maximise over several such sums. For example, Haslum et al. (2007) define the *canonical heuristic function* for a set of patterns as a maximum over all additive sums of pattern heuristics. In this work, we limit ourselves to additive pattern database heuristics without maximization over several sums, leaving this extension for future work.

Asymptotic Performance Ratio

By definition, the h^* heuristic is exact, i. e., it returns the length of a shortest plan. For the other heuristics, we give domain-dependent worst-case bounds on the accuracy compared to h^* and show that these bounds are tight in the sense that there are families of increasing-size tasks for which the accuracy tends towards the respective bound.

Formally, given a planning domain $\mathcal{D}$ and heuristic h, we want to find the asymptotic performance ratio of h in $\mathcal{D}$. More precisely, we want to find a value $c \in \mathbb{R}$ such that (a) for all states s in all tasks $\mathcal{T}$ of domain $\mathcal{D}$, $h(s) \geq ch^*(s) + o(h^*(s))$, and (b) there is a family of tasks $(\mathcal{T}_n)_{n\in\mathbb{N}}$ in $\mathcal{D}$ and solvable states $(s_n)_{n\in\mathbb{N}}$ such that s_n belongs to $\mathcal{T}_n$, $h^*(s_n) \to \infty$ for $n \to \infty$ and $h(s_n) \leq ch^*(s_n) + o(h^*(s_n))$.

In other words, h is *never* worse than ch^* (plus a sublinear term), and it can become as bad as ch^* (plus a sublinear term) for arbitrarily large inputs. These conditions can only be satisfied for a single value c; moreover, there must be a value $c \in [0, 1]$ which satisfies them for a given heuristic and domain. We denote this constant by $\alpha(h, \mathcal{D})$. Moreover, by $\alpha(h, s)$ we denote the ratio $h(s)/h^*(s)$ (we only consider non-goal states; otherwise $\alpha(h, s)$ is undefined). We simply write $\alpha(s)$ when the heuristic is clear from the context.

Results

Performance Ratio of h^+

First, we investigate the accuracy of the h^+ heuristic compared to h^*. As a general observation, $h^+(s)$, if not infinite, is always bounded by a linear term in the number of propositional state variables of the task. Therefore, for all domains $\mathcal{D}$ where there exists a family of tasks $(\mathcal{T}_n)_{n\in\mathbb{N}}$ of growing size where optimal solution lengths grow super-linearly with task size (formally, $h^*(s_n) = \omega(\|\mathcal{T}_n\|)$ for states s_n of $\mathcal{T}_n$), we must have $\alpha(h^+, \mathcal{D}) = 0$. However, none of the benchmark domains we study has this property – they can all be solved by linear-sized plans.

GRIPPER Let s_n be a state with $n > 0$ balls still at their start location. We assume that the robot is at the balls' start location, that both grippers are empty and that n is

[1] Formal proofs may appear in an extended version of this paper. In the meantime, they are available on request from the authors.

even. Otherwise, the h^* and h^+ values can increase by a small constant. The optimal plan consists of n *pickup* and *drop* actions each, $n/2$ forth moves (holding two balls) and $n/2 - 1$ back moves (with empty grippers). Thus, $h^*(s_n) = n + n + n/2 + n/2 - 1 = 3n - 1$. In the relaxed problem, the robot can be at both locations simultaneously after having performed its first *move* action. Together with the $2n$ *pickup* and *drop* actions still necessary, this gives an optimal relaxed plan length of $h^+(s_n) = 2n + 1$. Thus, $\alpha(s_n) = (2n+1)/(3n-1) > 2/3$. On the other hand, for the family $(s_n)_{n\in\mathbb{N}}$ of tasks described above, $\lim_{n\to\infty} \alpha(s_n) = 2/3$. Thus, $\alpha(h^+, \text{GRIPPER}) = 2/3$.

LOGISTICS Consider a LOGISTICS state s_n with n cities, no airports, and for each city c_i, two locations ℓ_{i1} and ℓ_{i2}, one truck t_i located at ℓ_{i1}, and one package with origin ℓ_{i2} and destination ℓ_{i1}. An optimal plan consists of $4n$ actions, namely two *drive*, one *pick-up* and one *drop* action in each city. An optimal relaxed plan only needs three actions per city since the second *drive* action can be omitted because the atom $at(t_i, \ell_{i1})$ holding initially is never deleted. Thus, we get $\alpha(h^+, \text{LOGISTICS}) \leq 3/4$.

We believe that also $\alpha(h^+, \text{LOGISTICS}) \geq 3/4$, and thus the bound is strict, but do not have a proof for this yet. (We do have a proof that works under the restriction that no truck is initially located at an airport, though.) A simple lower bound for $\alpha(h^+, \text{LOGISTICS})$ is $1/2$, because an optimal plan contains as many pickup and drop actions as an optimal relaxed plan, and never contains more movements than pickup or drop actions.

BLOCKSWORLD Let us first consider the family $(s_n)_{n\in\mathbb{N}}$ of states where s_n consists of one stack $B_1, \ldots, B_{n+1}$ (from top to bottom) and another singleton stack B_{n+2} and the goal is to produce the stack $B_1, \ldots, B_n, B_{n+2}$, with the position of B_{n+1} not being important. The optimal plan for s_n consists of $n - 1$ pairs of *unstack* and *put-down* actions to put the blocks $B_1, \ldots, B_{n-1}$ on the table, one *unstack* and *stack* action to move B_n from B_{n+1} to B_{n+2}, and $n - 1$ pairs of *pick-up* and *stack* actions to rebuild the stack on top of block B_n. Thus, $h^*(s_n) = 4n - 2$. In order to solve the relaxed problem, we have to *unstack* blocks $B_1, \ldots, B_n$ in order to add the $holding(B_n)$ proposition. We can unstack all blocks without emptying the hand as the proposition $handempty$ is true initially and is never falsified. To satisfy the goal, we then only need to stack B_n on B_{n+2}. The rest of the stack does not need to be rebuilt, as all propositions $on(B_i, B_{i+1})$ for $i < n$ still hold. This means that $\lim_{n\to\infty} \alpha(s_n) = \lim_{n\to\infty}(n + 1)/(4n - 2) = 1/4$.

For the other direction of the proof, we will show that $h^+(s) \geq 1/4 h^*(s) + d$ for a constant $d \in \mathbb{N}$ in all non-goal BLOCKSWORLD states. Consider a block B. If B is not touched in an optimal plan, it is not touched in an optimal relaxed plan either. If it is touched in an optimal plan, there are at most four actions moving B around, as it is always sufficient to pick-up/unstack and put-down/stack B once or twice. At least one of those up to four actions is also needed in an optimal relaxed plan. Therefore, $h^+(s) \geq 1/4 h^*(s)$, and thus $\alpha(h^+, \text{BLOCKSWORLD}) = 1/4$.

MICONIC-STRIPS Let $(s_n)_{n\in\mathbb{N}}$ be the following family of MICONIC-STRIPS states: There are $2n+1$ locations, with location 0 the elevator start location. There are $2n$ passengers, exactly one waiting at each location $i = 1, \ldots, 2n$. The destination of the passenger at location i is $i + 1$ if i is odd and $i-1$ if i is even. The optimal solution contains precisely seven action per passenger pair. Therefore, $h^*(s_n) = 7n$. In the relaxed task, a shortest plan is essentially the same as in the original task, the only exception being the fact that we can save one move action per location pair (the second move action back to the location visited twice). Therefore, $h^+(s_n) = 6n$ and $\lim_{n\to\infty} \alpha(s_n) = 6/7$.

To see that $h^+(s) \geq 6/7 h^*(s) + d$ for all MICONIC-STRIPS states s and a suitable constant d, note that the only actions that can potentially be saved in a relaxed plan are *move* actions when a location has to be visited twice. There is never a need to visit a location ℓ more than twice. A location may only need to be visited twice if it is part of a cycle in the graph defined by the "passenger edges". The shortest such cycle consists of just two locations. This is exactly the case in the task family described above. In that case, one in seven actions can be saved in a relaxed plan. If some passenger origin or destination locations are shared, the ratio of actions that can be skipped in a relaxed plan only decreases. The same holds if the length of the "passenger cycles" increases. Therefore, at least 6 out of 7 actions from an optimal plan are also needed in an optimal relaxed plan. Consequently, $\alpha(h^+, \text{MICONIC-STRIPS}) = 6/7$.

MICONIC-SIMPLE-ADL Assume that there are n floors and each floor is origin or destination of at least one passenger. Let K^* be the size of a minimum feedback vertex set for the dependency graph given by the passengers' origin-destination arcs. Then, in an optimal plan, there are $n + K^*$ or $n + K^* - 1$ *move* (up or down) and $n + K^*$ *stop* actions. In a corresponding relaxed plan, the $n + K^*$ *stop* actions are still necessary, but $n - 1$ *move* actions are sufficient as the *lift-at*(f) propositions once added are never removed. This gives a performance ratio bounded by $(2n + K^* - 1)/(2n + 2K^*) = 1 - (K^* + 1)/(2n + 2K^*) \geq 1 - (n + 1)/(2n + 2n) \to 3/4$. This bound is reached if the dependency graph is a complete graph, which can be achieved for tasks of any size introducing passengers from each floor to each other floor. We thus get $\alpha(h^+, \text{MICONIC-SIMPLE-ADL}) = 3/4$.

SCHEDULE The objective in the SCHEDULE domain is to change certain attributes of (some of) the parts to work on, which can include their *shape*, *surface condition* and *colour*. Parts also have a *temperature*, but there are no goals defined on that. We consider states $(s_n)_{n\in\mathbb{N}}$ defined as follows: All machines and parts are currently available, and there are n parts which are currently polished, coloured blue, and have no particular shape. The goal is to have all these parts polished, blue, and in cylindrical shape. Notice that in the SCHEDULE domain there are two types of actions: transformation actions and time-step actions, but in a relaxed plan the latter are never necessary. In particular, a relaxed plan for s_n simply applies the *do-lathe* operator to each part, so $h^+(s_n) = n$. An actual optimal plan must

however apply a sequence of three operators for each part, including the *do-lathe* operator, the *do-polish* operator, and either *do-spray-paint* or *do-immersion-paint*. The latter are necessary because the *do-lathe* operator, which is necessary for obtaining the cylindrical shape, removes the polish and colour from the part. (There are certain ordering constraints between these operators, but we do not discuss them in detail.) Moreover, $n + 1$ time-step actions are necessary: n time-step actions are needed until all parts have been lathed and the last lathed part is available again, and then one more time-step action is needed to polish and paint the last part (a time-step action is not needed at the end of the plan). Thus, $\lim_{n\to\infty} \alpha(s_n) = \lim_{n\to\infty} n/(4n+1) = 1/4$.

We now show that the h^+ heuristic is never worse than $1/4 h^* + d$. For this, consider a state s where n parts need to be processed (i. e., have a defined goal that is not already satisfied). Clearly, $h^+(s) \geq n$. To obtain an upper bound on $h^*(s)$, we describe a particular plan for s: First, classify the parts into different groups according to their current and goal attributes. The set $\mathcal{C}$ of possible classes is constant, i. e., fixed for all SCHEDULE tasks. For each class $C \in \mathcal{C}$, we can find a sequence of at most 3 transformation actions that creates the goal attributes for the parts in C. (This is always possible.) If there are m_C objects in the class, all parts in C can then be transformed into their goal states by using at most $3m_C$ transformation actions and at most $m_C + 2$ time-step actions, after which all machines are available again. Thus, altogether we need no more than $\sum_{C\in\mathcal{C}}(4m_C + 2) = 4n + 2|\mathcal{C}| = 4n + d$ actions, where $d := 2|\mathcal{C}|$. For our analysis, the additive constant does not matter, so we get $\alpha(h^+, \text{SCHEDULE}) = 1/4$.

SATELLITE The only propositions appearing in delete lists are *pointing*, *power-avail*, *calibrated* and *power-on* propositions. Consider the family of states $(s_n)_{n\in\mathbb{N}}$ where s_n is described by one satellite with n cameras, each only supporting one mode which is different for each camera, all cameras having the same calibration target (different from the current pointing direction) and the same image target (different from the calibration target and the current pointing direction) in their respective modes. In an optimal plan, six actions per image are necessary, namely powering the ith camera on, turning towards the calibration target, calibrating, turning towards the image target, taking the image, and switching the ith camera off. (The last step can be omitted for the last image.) In an optimal relaxed plan, only two turn actions and no switch-off actions are necessary. Otherwise, it is identical to the non-relaxed plan. The *power-on*, *calibrate*, and *take-image* actions are still necessary. Therefore, in the limit, optimal relaxed plans are merely half as long as non-relaxed optimal plans, and $\lim_{n\to\infty} \alpha(s_n) = 1/2$.

For the other direction, i. e. $h^+(s) \geq 1/2 h^*(s)$ for all states s, we first observe that there exists an optimal relaxed plan that only uses a camera after it has been powered on and calibrated and before the next camera has been powered on and calibrated. (For example, first move all *take-image* actions to the end of the plan, then group them by the used camera, then move the power-on and calibrate actions before each corresponding group of take-image actions.) This optimal relaxed plan can then be transformed into a plan for the original problem by adding an appropriate *power-off* action before each *power-on* action in order to ensure power availability, and adding *turn-to* actions before each *calibrate* and *take-image* action. Thus, optimal plans are at most twice as long as optimal relaxed plans.

We conclude that $\alpha(h^+, \text{SATELLITE}) = 1/2$.

Synopsis In all domains we considered, the h^+ heuristic yields results which are within a constant factor from the optimal heuristic values. The concrete factors vary from domain to domain.

Performance Ratio of h^k

For the performance ratio of the h^k family of heuristics, we get the same result for all domains, namely $\alpha(h^k, \mathcal{D}) = 0$. The common reason for this is that we can find families of states $(s_n)_{n\in\mathbb{N}}$ such that $h^*(s_n)$ is unbounded as n approaches infinity, whereas there exists a function f such that $h^k(s_n) \leq f(k)$ for all $k \in \mathbb{N}$. For any $k \in \mathbb{N}$, $f(k)$ is fixed and $h^*(s_n)$ can get arbitrarily large, so the performance ratio of the h^k heuristic tends towards zero.

Note that $h^k(s_n)$ can always be bounded from above by the cost of reaching a k-elementary subgoal set from s_n, maximised over all such subgoals. Thus, it suffices to show that for the states we consider, every k-elementary subgoal set can be reached in a number of steps that only depends on k, but not on n. (We are interested in behaviour in the limit, so we can always safely assume $n \geq k$.)

The property that $h^*(s_n)$ is unbounded for growing n will be trivial for the states we present, so we will not mention it explicitly.

GRIPPER From any GRIPPER state, it is possible to move k balls to the goal room with at most $4k + 1$ actions (drop a ball to free a gripper if necessary, then move, pick up, move, drop for each ball).

LOGISTICS Each subgoal set of size k can be reached in $12k$ steps, 12 for each goal (in the worst case of transportation between non-airport locations in different cities: move truck, pick up, move truck, drop; move airplane, pick up, move airplane, drop; move truck, pick up, move truck, drop).

BLOCKSWORLD Consider a family of states s_n with n blocks on the table to be stacked into a single tower (Haslum, Bonet, & Geffner 2005). Then each k-elementary subgoal can be reached in $2k$ steps.

MICONIC-STRIPS and MICONIC-SIMPLE-ADL Consider states with n passengers to be served with n different origin floors, different from the current elevator location. All subgoals of size k, i. e., transporting k passengers to their goals, can be reached in $4k$ steps (for each passenger: move to origin, stop/pick up, move to destination, stop/drop).

SCHEDULE Consider states with n parts to be processed. Any subset of k parts can be processed in $6k$ steps (each part needs up to three transformations as argued earlier, and each can be followed by a time-step action to guarantee availability of the machines).

SATELLITE Consider states s_n with n remaining image goals. Any subset of k goals can be satisfied in $6k$ steps (for each objective: power off an instrument if necessary, power on an instrument, turn to calibration target, calibrate, turn to objective, take image).

Synopsis In all the domains, there are trivial families of increasing-size tasks with h^* values roughly proportional to n and h^k values roughly proportional to k. As h^* increases indefinitely for growing n, whereas h^k never returns a value greater than a domain-dependent constant which is a function of k, the performance ratio of h^k tends to zero in all domains.

Performance Ratio of Pattern Database Heuristics

The accuracy of the h^{PDB} family of heuristics depends on the choice of the pattern. We assume that the pattern database has a size limit polynomial in the input size n, say $O(n^k)$. Then a pattern may contain no more than $O(\log n^k) = O(k \log n)$ state variables in order to respect the limit. (Since k is an arbitrarily chosen constant, we can simplify this expression to $O(\log n)$.) This implies that $h^{\mathrm{PDB}}(s)$ cannot be greater than the cost of reaching the most expensive subgoal set of cardinality $O(\log n)$.

As the proofs in the previous section showed, in all considered domains there exist families of states $(s_n)_{n\in\mathbb{N}}$ for which h^* grows linearly with n but m-elementary subgoal costs only grow linearly with m. Since $O(\log n)/\Omega(n) \to 0$ for $n \to \infty$, we obtain the same results for h^{PDB} as for h^k: for all domains, $\alpha(h^{\mathrm{PDB}}, \mathcal{D}) = 0$. Note, however, that while for h^k, the heuristic estimates were always bounded by constants for a fixed value of k, for pattern database heuristics with suitably chosen patterns they are bounded by values that grow logarithmically with n. In that sense, pattern database heuristics are more powerful than h^k.

Performance Ratio of Additive Pattern Database Heuristics

As in the previous section, we need to bound the size of *individual* patterns by $O(\log n)$ for tasks of size n. However, for additive pattern database heuristics, multiple such patterns are considered, and their values summed. We remark that since patterns in these domains need to be disjoint to be additive, there cannot be more than n pairwise additive patterns for a task with n state variables, so there is no need to impose a bound on the number of patterns to ensure polynomial time and space requirements.

GRIPPER We can restrict attention to states s_n with n balls in the initial room and no balls currently carried. Balls currently carried can be ignored because they only contribute a constant to the overall cost (as there are only two grippers).

Using an additive pattern database with singleton patterns for each ball location variable, $h^{\mathrm{PDB}}_{\mathrm{add}}(s_n) = 2n$, whereas the optimal cost is $3n - 1$ if n is even and $3n$ if n is odd. In either case, the performance ratio tends towards $2/3$, so we get $\alpha(h^{\mathrm{PDB}}_{\mathrm{add}}, \textsc{Gripper}) = 2/3$.

LOGISTICS Again, by using singleton patterns for each goal variable, we obtain a heuristic which accurately captures all pick-up and drop actions needed to reach the goal. An optimal solution to a LOGISTICS task never requires more movements than pick-up and drop actions combined, so the accuracy is at least $1/2$.

To prove a lower bound, consider the family $(s_n)_{n\in\mathbb{N}}$ of tasks defined as follows. There is a single truck in a city with $2n + 1$ locations. There are n packages, all to be moved within this city. All initial and goal locations of the packages are different from each other and from the initial truck location. The optimal cost for this task is clearly $h^*(s_n) = 4n$. The multi-valued encoding of the task consists of $n + 1$ state variables; one for the truck, and one for each package. Forming non-singleton patterns that do not include the truck variable is not useful, since this does not improve performance over the sum of singleton patterns of the involved package variables. Thus, an optimal additive pattern database heuristic for this task includes one pattern including the truck variable and as many package variables as possible (say, K), and singleton patterns for all remaining packages. This results in a heuristic estimate of $h^{\mathrm{PDB}}_{\mathrm{add}}(s_n) = 4K + 2(n - K) = 2n + 2K$. As argued above, $K \in O(\log n)$, so $\lim_{n\to\infty} \alpha(s_n) = (2n + O(\log n))/4n = 1/2$. We thus get $\alpha(h^{\mathrm{PDB}}_{\mathrm{add}}, \textsc{Logistics}) = 1/2$.

BLOCKSWORLD In the BLOCKSWORLD domain, consider the state s_n consisting of a stack $B_1, \ldots, B_n, B_{n+1}$ to be transformed into the stack $B_1, \ldots, B_{n+1}, B_n$. In other words, the two bottommost blocks must be swapped. The true cost is $4n$.

There are only two variables whose current values differ between s_n and the goal, and hence all but two patterns in the additive pattern database heuristic must assign heuristic values of 0 (since, relative to these patterns, the goal has already been reached). Thus, there are at most two patterns contributing to $h^{\mathrm{PDB}}_{\mathrm{add}}$, and again their magnitude is bounded by $O(\log n)$ since they can only incorporate $O(\log n)$ many variables and BLOCKSWORLD problems (or subproblems) of size k can be solved with $O(k)$ steps.

Therefore, the additive pattern database heuristics for BLOCKSWORLD can get arbitrarily inaccurate with a performance ratio of $\alpha(h^{\mathrm{PDB}}_{\mathrm{add}}, \textsc{Blocksworld}) = 0$.

MICONIC-STRIPS This domain is a special case of LOGISTICS, so from the result for that domain, we can conclude $\alpha(h^{\mathrm{PDB}}_{\mathrm{add}}, \textsc{Miconic-Strips}) \geq 1/2$. On the other hand, the family of tasks used for showing the LOGISTICS upper bound are also MICONIC-STRIPS tasks, so $\alpha(h^{\mathrm{PDB}}_{\mathrm{add}}, \textsc{Miconic-Strips}) = 1/2$.

MICONIC-SIMPLE-ADL For the ADL domain variant, we cannot provide a useful lower or upper bound yet.

SCHEDULE Consider the pattern collection where there is one pattern for each part, containing all attribute variables describing that part (i. e., its colour, surface condition, shape, temperature, and has-hole variables). These patterns are all additive and only comprise a constant number of variables each. With this collection, an additive pattern database heuristic captures the costs of all necessary transformation

Domain	h^+	h^k	h^{PDB}	$h^{\mathrm{PDB}}_{\mathrm{add}}$
GRIPPER	$2/3$	0	0	$2/3$
LOGISTICS	$1/2$–$3/4$	0	0	$1/2$
BLOCKSWORLD	$1/4$	0	0	0
MICONIC-STRIPS	$6/7$	0	0	$1/2$
MICONIC-SIMPLE-ADL	$3/4$	0	0	open
SCHEDULE	$1/4$	0	0	$1/2$
SATELLITE	$1/2$	0	0	$1/6$–$1/2$

Figure 1: Performance ratios of the h^+, h^k, and (additive) pattern database heuristics in selected planning domains.

actions correctly, but ignores all time-step actions. In general, there are never more time-step than other actions (because two time-step actions in sequence can be collapsed) and there never is a time-step action at the end of an optimal plan, so $h^{\mathrm{PDB}}_{\mathrm{add}}(s) \geq 1/2 h^*(s)$ for all states s.

On the other hand, there exist state families $(s_n)_{n\in\mathbb{N}}$ for which this bound is tight. In particular, consider states where n parts need to be polished, the polisher and the parts are available, and there are no further goals. An optimal solution requires $2n-1$ actions, whereas $h^{\mathrm{PDB}}_{\mathrm{add}}(s) = n$ for the pattern collection we described. By using a different pattern collection, the result cannot be improved significantly; similar to the previous proofs, we could only capture the interaction between the polish and time-step actions for logarithmically many parts without exceeding the pattern size bound. Consequently, $\alpha(h^{\mathrm{PDB}}_{\mathrm{add}}, \mathrm{SCHEDULE}) = 1/2$.

SATELLITE As argued in the section on h^k, the optimal solution length from any SATELLITE state with n goals is at most $6n$. On the other hand, an additive pattern database heuristic which only includes singleton patterns for the goal variables achieves a heuristic estimate of n, so $\alpha(h^{\mathrm{PDB}}_{\mathrm{add}}, \mathrm{SCHEDULE}) \geq 1/6$.

Now consider state s_n with one satellite with a single instrument supporting a single mode, initially turned off. There are n image targets (all of different pointing directions). The satellite is originally pointed to another direction, which is the calibration target of its instrument.

Clearly $h^*(s_n) = 2n+2$: an optimal plan begins with *power-on* and *calibrate* actions, followed by n pairs of *turn-to* and *take-image* actions. An additive pattern database heuristic can only capture a logarithmic number of turn-to actions (from the pattern including the variable encoding the pointing direction), so it is bounded by $n + O(\log n)$, and hence $\alpha(h^{\mathrm{PDB}}_{\mathrm{add}}, \mathrm{SCHEDULE}) \leq 1/2$. (We cannot presently close the gap between the upper and lower bound.)

Synopsis Except for the case of the BLOCKSWORLD domain, where a "reasonable" additive pattern database heuristic performs arbitrarily badly, we could not find families of tasks for which a "reasonable" choice of patterns would perform worse than a constant factor times the optimal heuristic value.

Conclusion

As far as we are aware, we have presented the first detailed analysis of domain-specific accuracy results for a number of popular planning heuristics (summarised in Fig. 1). One maybe unsurprising, but still interesting observation is that the h^k heuristic family and plain pattern database heuristics become arbitrarily inaccurate as task size increases.

With the exception of the SCHEDULE domain, we generally got the best results for the h^+ heuristic. However, in interpreting this result, one should not forget that this heuristic is NP-hard to compute in general, which makes additive pattern database heuristics attractive.

For these, we remark that the problem of optimizing the pattern collections – a task we performed by hand in our analysis – is of critical importance for obtaining good performance. On the other hand, some of our results already hold when using singleton patterns exclusively, and in other cases (such as SCHEDULE), the result would not be much worse with singleton patterns. In particular, current techniques for the automatic selection of pattern collections (Haslum *et al.* 2007) easily achieve the best-possible performance ratios from Fig. 1 (in the case of SATELLITE, the lower bound on this ratio), with the possible exception of the SCHEDULE domain, where at least a ratio of ¼ is achieved.

Finally, we again point out some open issues. In addition to the incomplete entries in Fig. 1, one important omission is that we did not discuss additive h^k heuristics and that our discussion of additive pattern databases was restricted to the case of a single sum. Given that the latter heuristics in particular have been very successful recently (Haslum *et al.* 2007), this is an interesting and important avenue for future work.

References

Ausiello, G.; Crescenzi, P.; Gambosi, G.; Kann, V.; Marchetti-Spaccamela, A.; and Protasi, M. 1999. *Complexity and Approximation*. Springer-Verlag.

Bacchus, F. 2001. The AIPS'00 planning competition. *AI Magazine* 22(3):47–56.

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

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *AIJ* 69(1–2):165–204.

Culberson, J. C., and Schaeffer, J. 1998. Pattern databases. *Computational Intelligence* 14(3):318–334.

Edelkamp, S., and Helmert, M. 2001. The model checking integrated planning system (MIPS). *AI Magazine* 22(3):67–71.

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

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *Proc. AAAI 2007*, 1007–1012.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New admissible heuristics for domain-independent planning. In *Proc. AAAI 2005*, 1163–1168.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In *Proc. ICAPS 2007*, 176–183.

Helmert, M. 2006. *Solving Planning Tasks in Theory and Practice*. Ph.D. Dissertation, Albert-Ludwigs-Universität Freiburg.

Hoffmann, J., and Edelkamp, S. 2005. The deterministic part of IPC-4: An overview. *JAIR* 24:519–579.

Hoffmann, J. 2005. Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *JAIR* 24:685–758.

Kautz, H., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proc. AAAI 1996*, 1194–1201.

Kautz, H., and Selman, B. 1999. Unifying SAT-based and graph-based planning. In *Proc. IJCAI 1999*, 318–325.

Long, D., and Fox, M. 2003. The 3rd International Planning Competition: Results and analysis. *JAIR* 20:1–59.

McDermott, D. 1996. A heuristic estimator for means-ends analysis in planning. In *Proc. AIPS 1996*, 142–149.

McDermott, D. 2000. The 1998 AI Planning Systems competition. *AI Magazine* 21(2):35–55.