Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (2008)

# How Good is Almost Perfect?

**Malte Helmert** and **Gabriele Röger**
Albert-Ludwigs-Universität Freiburg, Germany
{helmert,roeger}@informatik.uni-freiburg.de

## Abstract

Heuristic search using algorithms such as A* and IDA* is the prevalent method for obtaining optimal sequential solutions for classical planning tasks. Theoretical analyses of these classical search algorithms, such as the well-known results of Pohl, Gaschnig and Pearl, suggest that such heuristic search algorithms can obtain better than exponential scaling behaviour, provided that the heuristics are accurate enough.

Here, we show that for a number of common planning benchmark domains, including ones that admit optimal solution in polynomial time, general search algorithms such as A* must *necessarily* explore an exponential number of search nodes even under the optimistic assumption of *almost perfect* heuristic estimators, whose heuristic error is bounded by a small additive constant.

Our results shed some light on the comparatively bad performance of optimal heuristic search approaches in "simple" planning domains such as GRIPPER. They suggest that in many applications, further improvements in run-time require changes to other parts of the search algorithm than the heuristic estimator.

## Introduction

Optimal sequential planning is harder than satisficing planning. While there is no difference in theoretical complexity in the general case (Bylander 1994), many of the classical planning domains are provably easy to solve sub-optimally, but hard to solve optimally (Helmert 2003).

Moreover, strikingly different scaling behaviour of satisficing and optimal planners has been observed in practice (Hoffmann and Edelkamp 2005). In fact, this disparity even extends to planning domains which are known to be *easy* to solve optimally in theory. If we apply two state-of-the-art optimal planning algorithms (Haslum et al. 2007; Helmert, Haslum, and Hoffmann 2007) to the GRIPPER domain, neither of them can optimally solve more than 8 of the standard suite of 20 benchmarks within reasonable run-time and memory limits, whereas the whole suite is solved in a few milliseconds by satisficing planners like FF (Hoffmann and Nebel 2001). Moreover, those 8 tasks are quickly solved by breadth-first search, showing no significant advantage of sophisticated heuristic methods over brute force.



Why is this the case? One possible explanation is that the heuristic estimators of these planning systems may be grossly misleading for GRIPPER tasks. However, we do not believe that this is the case – the GRIPPER domain in particular has resisted many attempts by optimal heuristic planners, hinting at a different, more fundamental problem. In this contribution, we argue the following claim:

> For many, maybe *all* of the standard benchmark domains in planning, standard heuristic search algorithms such as A* quickly become prohibitively expensive even if *almost perfect heuristics* are used.

We suggest that, beyond a certain point, trying to improve a heuristic search algorithm by refining its heuristic estimates is basically fruitless. Instead, one should look for orthogonal performance enhancements such as symmetry elimination (Fox and Long 1999) or domain simplification (Haslum 2007) to improve the scaling behaviour of optimal planners – unless one can reach the extremely ambitious goal of deriving *perfect*, and not merely *almost perfect* heuristics.

## Almost Perfect Heuristics

The performance of heuristic search is commonly measured by the number of performed node expansions. Of course, this measure depends on the search algorithm used; for example, A* (Hart, Nilsson, and Raphael 1968) will usually explore fewer states than IDA* (Korf 1985) in the same search space, and never more (assuming that successors are ordered in the same way).

Here, we consider lower bounds for node expansions of the A* algorithm with full duplicate elimination. Results for this algorithm immediately apply to other search algorithms that rely exclusively on node expansions and admissible heuristic estimates to guide search, such as IDA*, A* with partial expansion (Yoshizumi, Miura, and Ishida 2000), breadth-first heuristic search (Zhou and Hansen 2006), and many more. However, they do *not* apply to algorithms that use additional information for state pruning, such as symmetry reduction, and neither to algorithms that use fundamentally different techniques to find optimal plans, such as symbolic breadth-first search (Edelkamp and Helmert 2001) or SAT planning (Kautz and Selman 1999).

How many nodes does A* expand for a planning task $\mathcal{T}$,

given an admissible heuristic $h$? Clearly, this depends on the properties of $\mathcal{T}$ and $h$. It also depends on some rather accidental features of the search algorithm implementation, in particular on the order in which nodes with identical $f$ values are explored. Because we are interested in lower bounds, one conservative assumption is to estimate the solution effort by the number of states $s$ with the property $f(s) := g(s) + h(s) < h^*(\mathcal{T})$, where $g(s)$ is the cost (or distance – we assume a unit cost model) for reaching $s$ from the initial state, $h(s)$ is the heuristic estimate for $s$, and $h^*(\mathcal{T})$ is the optimal solution cost for $\mathcal{T}$. All states with this property *must* be considered by A$^*$ in order to guarantee that no solutions of length below $h^*(\mathcal{T})$ exist. In practice, a heuristic search algorithm will also expand *some* states with $f(s) = h^*(\mathcal{T})$, but we ignore those in our estimates.

Our aim is to demonstrate fundamental limits to the scaling possibilities of optimal heuristic search algorithms when applied to planning tasks. For this purpose, we show that A$^*$ search effort already grows extremely fast for a family of very powerful heuristic functions. More precisely, we consider heuristics parameterized by a natural number $c \in \mathbb{N}_1$ where the heuristic estimate of state $s$ is defined as $\max(h^*(s) - c, 0)$, with $h^*(s)$ denoting the length of an optimal solution from state $s$ as usual. In the following, we use the notation "$h^* - c$" to refer to this heuristic (not reflecting the maximization operation in the notation). In other words, $(h^* - c)(s) := \max(h^*(s) - c, 0)$.

We call heuristics like $h^* - c$, which only differ from the perfect heuristic $h^*$ by an additive constant, *almost perfect*. Almost perfect heuristics are unlikely to be attainable in practice in most planning domains. Indeed, for any of the planning benchmark domains from IPC 1–4 that are NP-hard to optimize, if there exists a polynomial-time computable almost perfect heuristic, then APX = PTAS and hence P = NP (Helmert, Mattmüller, and Röger 2006).

Throughout our analysis, we use the notation $N^c(\mathcal{T})$ to denote the A$^*$ node expansion lower bound for task $\mathcal{T}$ when using heuristic $h^* - c$. In other words, $N^c(\mathcal{T})$ is the number of states $s$ with $g(s) + (h^* - c)(s) < h^*(\mathcal{T})$. This can be equivalently expressed as the number of states with $f^*(s) := g(s) + h^*(s) < h^*(\mathcal{T}) + c$ and $g(s) < h^*(\mathcal{T})$. The objective of our analysis is to provide results for $N^c(\mathcal{T})$ for planning tasks $\mathcal{T}$ drawn from the standard IPC benchmark domains, focusing on the "easiest" domains, namely those that fall within the approximation class APX (Helmert, Mattmüller, and Röger 2006). These domains are particularly relevant because they would intuitively appear most likely to be within reach of optimal planning techniques.

## Related Work

There is quite a bit of literature on the computational costs of A$^*$ and related algorithms. A widely cited result by Pohl (1977) considers precisely the kind of heuristics we study in this paper, i. e., those with constant absolute error. He proves that the A$^*$ algorithm requires a linear number of node expansions in this case. However, the analysis relies on certain critical assumptions which are commonly violated in planning tasks. First, it assumes that the branching factor of the search space is constant across inputs. If the branching factor is polynomially related to input size, as is often the case in planning, the number of node expansions is still polynomial in the input size for a fixed error $c$, but of an order that grows with $c$. More importantly, the analysis requires that there is only a single goal state and that the search space contains no transpositions. The latter assumption, critical to Pohl's tractability result, is violated in all common benchmark tasks in planning. For example, in the following section we show that the GRIPPER domain requires an exponential number of node expansions even with very low heuristic inaccuracies due to the large number of transpositions. Pohl also considers the case of heuristics with a constant *relative* error, i. e., $h(s) \approx c \cdot h^*(s)$ for some constant $c < 1$. However, as our negative results already apply to the much more optimistic case of constant absolute error, we do not discuss this analysis.

Gaschnig (1977) extends Pohl's analysis to logarithmic absolute error (i. e., $h^*(s) - h(s) = O(\log(h^*(s)))$), showing that A$^*$ also requires a polynomial number of expansions under this less restrictive assumption. However, his analysis requires the same practically rare search space properties (no transpositions, a single goal state).

Both Pohl's and Gaschnig's work is concerned with worst-case results. Pearl (1984) extends their analyses by showing that essentially the same complexity bounds apply to the *average case*.

More recently, Dinh et al. (2007) consider heuristics with constant relative error in a setting with multiple goal states. While this is a significant step towards more realistic search spaces, absence of transpositions is still assumed.

In addition to these analyses of A$^*$, the literature contains a number of results on the complexity of the IDA$^*$ algorithm. The article by Korf et al. (2001) is particularly relevant in the planning context, because it presents an analysis which has recently been used to guide heuristic selection in the very effective optimal sequential planner of Haslum et al. (2007). Korf et al. show that IDA$^*$ expands at most $\sum_{i=0}^{k} N_i P(k-i)$ nodes to prove that no solution of length at most $k$ exists (or to find such a solution, if it exists). Here, $N_i$ is the number of nodes at depth $i$ in the search tree, and $P(m)$ is the probability that a randomly drawn node "deep" in the search tree has a heuristic value of $m$.

However, the formula only applies to IDA$^*$, and only in the limit of large $k$. We are interested in lower bounds on complexity for *any* general heuristic search algorithm, including ones that perform complete duplicate elimination. In that case, the number of states $N_i$ at depth $i$ eventually becomes zero, so that it is not clear what results that apply "in the limit of large $k$" signify. Moreover, with complete duplicate elimination, there is no reasonable way of defining the equilibrium distribution $P$. Therefore, we do not discuss this complexity result further.

## Theoretical Results

There are several ways of obtaining $N^c(\mathcal{T})$ estimates. One way is to *measure* them empirically for particular values of $c$ and particular tasks $\mathcal{T}$ by using an algorithm that conducts

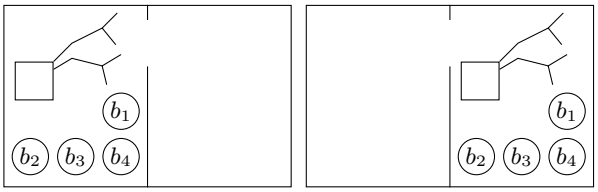


Figure 1: Initial state and goal of GRIPPER task $\mathcal{T}_4$.

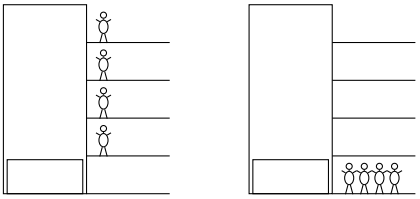

Figure 2: Initial state and goal of MICONIC task $\mathcal{T}_4$.

A$^*$ searches with the $h^* - c$ heuristic. One advantage of this method is that it is fully general – it can be directly applied to arbitrary planning tasks, and any value $c \in \mathbb{N}_1$. We present some results obtained by this method in the following section.

However, the empirical approach has some drawbacks. Firstly, it is computationally expensive and thus limited to comparatively small planning tasks (in particular, a subset of those which we can solve optimally by heuristic search). Secondly, its results are fairly opaque. In addition to knowing lower bounds for a certain set of planning tasks, we would also like to know *why* they arise, and how to extrapolate them to instances of larger size. For these purposes, theoretical results are preferable.

In this section, we present such theoretical results for three planning domains: GRIPPER, MICONIC-SIMPLE-ADL, and BLOCKSWORLD. We assume familiarity with these domains and point to the literature (Helmert 2008) for definitions and details. We choose these particular domains because we consider them fairly representative of the IPC domains in class APX. Similar theorems can be shown for most other APX domains such as MICONIC-STRIPS, LOGISTICS, ZENOTRAVEL, DEPOTS, and SCHEDULE.

Our theorems take the form of *worst case* results: In each of the domains, we show that there exist tasks of scaling size for which the number of states expanded by $h^* - c$ grows exponentially. One problem with worst case considerations is that they might only apply in some fairly unusual "corner cases" that are unlikely to arise in practice. We partially avoid this problem by discussing families of tasks of different types. For example, we can observe exponential scaling of $N^i(\mathcal{T})$ both for families of BLOCKSWORLD tasks where initial and goal configurations consist of a single tower and for families of tasks consisting of a large number of small towers. Still, *average-case* results are clearly also of interest, and are left as a topic for future work.

## GRIPPER

In the GRIPPER domain, a robot with two arms needs to transport a number of balls from one room to another (Fig. 1). GRIPPER tasks are completely characterized by the number of balls, so there is no difference between worst-case and average-case results in this domain. We denote with $\mathcal{T}_n$ the task with $n$ balls. The total number of reachable states of $\mathcal{T}_n$ is $S_n := 2 \cdot (2^n + 2n2^{n-1} + n(n-1)2^{n-2})$. (The initial factor of 2 represents the two possible robot locations; the three terms of the sum correspond to the cases of 0, 1 and 2 carried balls, respectively, and represent the contents of the robot arms and the locations of the balls not being carried. Note that the two robot arms can be distinguished.) We now state our main result for GRIPPER.

**Theorem 1** *Let $n \in \mathbb{N}_0$ with $n \geq 3$. If $n$ is even, then $N^1(\mathcal{T}_n) = N^2(\mathcal{T}_n) = \frac{1}{2}S_n - 3$ and $N^c(\mathcal{T}_n) = S_n - 2n - 2$ for all $c \geq 3$. If $n$ is odd, then $N^1(\mathcal{T}_n) = N^2(\mathcal{T}_n) = S_n - 3$ and $N^c(\mathcal{T}_n) = S_n - 2$ for all $c \geq 3$.*

**Proof sketch:** In the case where $n$ is even, we call a state $s$ *even* if the number of balls in each room is even, and *odd* otherwise. (Balls carried by the robot are counted towards the room in which the robot is located.)

The basic proof idea is that if $n$ is even, then all even states apart from the two states where all balls are in the same room and the robot is in the other room are part of some optimal plan. Moreover, all odd states are parts of plans of length $h^*(\mathcal{T}_n) + 2$. (However, $2n$ odd states are still ruled out from $N^c(\mathcal{T}_n)$ because their $g$ value exceeds $h^*(\mathcal{T}_n)$.)

In the case where $n$ is odd, all states except for the two states where all balls are in the same room and the robot is in the other room are part of some optimal plan. ■

The theorem shows that there is little hope of achieving significant pruning for GRIPPER by using heuristic estimators, even in the easier case where the number of balls is even. With a heuristic error of 1, about half of all reachable states need to be considered already, because they all lie on optimal paths to the goal. Moreover, once the heuristic error is greater than 2, the A$^*$ algorithm has essentially no pruning power, and offers no advantage over breadth-first search with duplicate elimination.

## MICONIC-SIMPLE-ADL

In the MICONIC domain family, there is an elevator moving between the floors of a building. There are passengers waiting at some of the floors; the goal is to transport each passenger to their destination floor. In the MICONIC-SIMPLE-ADL domain variant, there are *movement* actions, which move the elevator from one floor to any other floor in a single step, and *stop* actions, which drop off all boarded passengers whose destination is the current floor and cause all passengers waiting to be served at that floor to enter.

We consider the following family of MICONIC-SIMPLE-ADL tasks $\mathcal{T}_n$: There are $(n + 1)$ floors and $n$ passengers. The elevator is initially located at the bottom floor, which is also the destination of all passengers. There is one passenger waiting to be served at each floor except the bottom one (Fig. 2).

An optimal plan for $\mathcal{T}_n$ clearly needs $h^*(\mathcal{T}_n) = 2(n+1)$ steps: Move to each floor where a passenger is waiting and stop there. Once all passengers are boarded, move to the destination floor and stop again to drop off all passengers.

Altogether there are $S_n := 3^n(n+1)$ states in the search space (each passenger can be waiting, boarded, or served; the elevator can be at $(n+1)$ different locations). Some of these states are never expanded by A$^*$ with any heuristic (not even by $h = 0$) before the final search layer, because they cannot be reached within less than $h^*(\mathcal{T}_n)$ steps. However, the number of such states is fairly small, and *all other reachable states must be expanded by A$^*$ if the heuristic error is at least* 4.

**Theorem 2** *For all $c \geq 4$, $N^c(\mathcal{T}_n) = S_n - (2^n - 1)(n+1)$.*

**Proof sketch:** As noted above, $h^*(\mathcal{T}_n) = 2(n+1)$, which means that nodes with $g(s) \geq 2(n+1)$ cannot possibly be expanded before the final $f$ layer of the A$^*$ search (which our definition of $N^c$ optimistically excludes from consideration). These nodes can be exactly characterized: A state $s$ requires at least $2(n+1)$ actions to reach iff no passenger is waiting and at least one passenger is served in $s$. (Under these conditions, the elevator must have moved to and stopped at each of the $(n+1)$ locations at least once, which requires $2(n+1)$ actions.) There are exactly $(2^n-1)(n+1)$ states of this kind, where the first term characterizes the $2^n - 1$ possible nonempty sets of served passengers, and the second term characterizes the possible elevator locations.

Now let $s$ be an arbitrary state which is *not* of this form and where the elevator is at floor $l$. Consider the following plan: First collect all passengers which are served in $s$ and drop them off at the bottom floor (if there are any such passengers), then collect all passengers boarded in $s$, then move to $l$ if not already there. (At this point, we are in state $s$.) Finally, pick up the remaining passengers and drop them off at the bottom floor. The plan contains no loops, visits $s$ after less than $2(n+1)$ steps, and has length at most $h^*(\mathcal{T}_n) + 3$. Together, these facts imply that $s$ will be counted towards $N^c(\mathcal{T}_n)$ for $c \geq 4$. ∎

We remark that the fraction of states *not* expanded by $h^* - 4$ according to our optimistic assumptions is almost exactly $(\frac{2}{3})^n$, which quickly tends towards zero as $n$ increases. Thus, for larger values of $n$, heuristics with an error of 4 have no significant advantage over blind search.

We also remark that a similar construction is possible with tasks where all origin and destination floors are disjoint.

### BLOCKSWORLD

In the BLOCKSWORLD domain blocks stacked into towers must be rearranged by a robotic arm which can pick up the top block of a tower and place it on another tower or on the table.

One easy way of defining a family of BLOCKSWORLD tasks for which non-perfect heuristics must perform exponential amounts of search is to create a linear number of "independent" subproblems. (This is a general idea that works for a large number of benchmark domains, including many of those not discussed here.) One example is an initial state with $n$ stacks of two blocks each, each of which needs to be reversed in the goal. Since the independent subplans can be interleaved arbitrarily, there are exponentially many states that are part of optimal plans, which implies that $N^1(\mathcal{T}_n)$ grows exponentially.

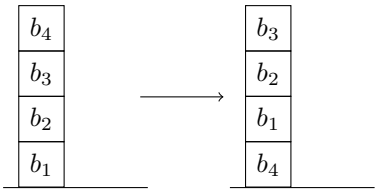


Figure 3: Initial and goal state of BLOCKSWORLD task $\mathcal{T}_4$.

| $n$ | $N^1(\mathcal{T}_n)$ | $n$ | $N^1(\mathcal{T}_n)$ |
|---|---|---|---|
| 2 | 4 | 9 | 3748 |
| 3 | 8 | 10 | 17045 |
| 4 | 15 | 11 | 84626 |
| 5 | 32 | 12 | 453698 |
| 6 | 82 | 13 | 2605383 |
| 7 | 253 | 14 | 15924744 |
| 8 | 914 | 15 | 103071652 |

Figure 4: Lower bound of the number of expanded states in a BLOCKSWORLD task $\mathcal{T}_n$ with heuristic error $c = 1$.

However, we can prove a similar result even in the case where all blocks are stacked into a single tower in the initial state and goal. Consider the class of BLOCKSWORLD instances $\mathcal{T}_n$ ($n \geq 2$) where one block shall be moved from the top of a tower with $n$ blocks to the base of the tower (Fig. 3).

For these tasks, a heuristic planner needs to expand an exponential number of states – even if the heuristic has only an error of $c = 1$. The number $N^1(\mathcal{T}_n)$ of states that need to be expanded depends on the Bell numbers $B_k$ (defined as the number of partitions of a set with $k$ elements), which grow exponentially in $k$. More precisely, we get the following formula (technical proof omitted):

$$N^1(\mathcal{T}_n) = 4 \cdot \sum_{k=0}^{n-3} B_k + 3B_{n-2} + 1$$

To get an impression of the rate of growth of $N^1(\mathcal{T}_n)$, values for $n = 2, \ldots, 15$ are depicted in Fig. 4.

## Empirical Results

We have seen that the $h^* - c$ family of heuristics has surprisingly bad theoretical properties in a number of common planning domains. This result immediately prompts the question: Can we observe this behaviour in practice?

To answer this question, we have devised an algorithm to compute $N^c(\mathcal{T})$ values and applied it to planning tasks from the IPC benchmark suite. One obvious problem in computing these values is that they are defined in terms of the perfect heuristic estimate of a state $h^*(s)$, which usually cannot be determined efficiently. (Otherwise, there would be no need to write this paper.)

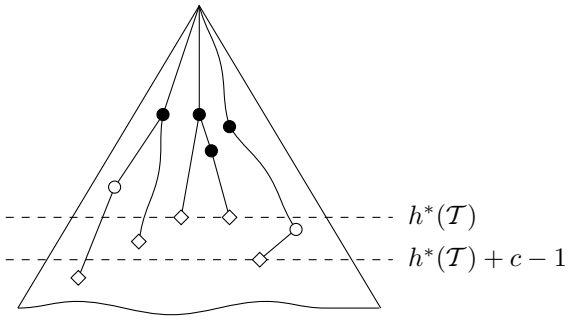


Figure 5: Schematic view of the search space. Goal nodes are depicted as diamonds, others as circles. Solid nodes belong to $N^c(\mathcal{T})$; hollow ones do not.

One possible way of computing $N^c(\mathcal{T})$ is to completely explore the state space of $\mathcal{T}$, then search backwards from the goal states to determine the $h^*(s)$ values. However, generating *all* states is not actually necessary. Recall that we are interested in $N^c(\mathcal{T})$, which is the number of states $s$ with $g(s) + (h^* - c)(s) < h^*(\mathcal{T})$. Obviously, all these states are reachable within less than $h^*(\mathcal{T})$ steps from the initial state. Furthermore, they must have a descendant which is a goal state and has a depth of at most $h^*(\mathcal{T}) + c - 1$ (Fig. 5). Thus, for determining $h^*(s)$ for the relevant nodes $s$, it is sufficient to know all goal states until that depth. For this purpose, we employ a consistent and admissible heuristic to expand all nodes up to and including the $f$ layer $h^*(\mathcal{T}) + c - 1$.

In summary, we use the following algorithm:

1. Perform an A$^*$ search with an arbitrary consistent, admissible heuristic until a goal state is found at depth $h^*(\mathcal{T})$.
2. Continue the search until all states in layer $h^*(\mathcal{T}) + c - 1$ have been expanded.
3. Determine the optimal goal distance $h^*(s)$ within the explored part of the search space for all expanded nodes $s$ by searching backwards from the detected goal states.
4. Count the nodes with $g(s) + (h^* - c)(s) < h^*(\mathcal{T})$.

The outcome of the experiment is shown in Fig. 6. For all IPC tasks from the domains considered, we attempted to measure $N^c(\mathcal{T})$ for $c \in \{1, 2, 3, 4, 5\}$ using the technique outlined above. Instances within a domain are ordered by scaling size; in cases where the IPC suite contains several tasks of the same size, we only report data for the one that required the largest search effort. Results are shown up to the largest instance size for which we could reliably complete the computation for $c = 1$ within a 3 GB memory limit. (Blank entries for $c > 1$ correspond to cases where the memory limit was exceeded for these larger values.)

For the GRIPPER domain and both MICONIC variants, the theoretical worst-case results are clearly confirmed by the outcome of the experiments. Even for $c = 1$, run-time of the algorithm scales exponentially with instance size. The data for BLOCKSWORLD and LOGISTICS is less conclusive, but there appears to be a similar trend (for BLOCKSWORLD, this is most pronounced for $c = 5$).

| task | $h^*(\mathcal{T})$ | $N^1(\mathcal{T})$ | $N^2(\mathcal{T})$ | $N^3(\mathcal{T})$ | $N^4(\mathcal{T})$ | $N^5(\mathcal{T})$ |
|---|---|---|---|---|---|---|
| BLOCKSWORLD | | | | | | |
| 04-1 | 10 | 10 | 10 | 16 | 16 | 29 |
| 05-2 | 16 | 28 | 28 | 72 | 72 | 162 |
| 06-2 | 20 | 27 | 27 | 144 | 144 | 476 |
| 07-1 | 22 | 106 | 106 | 606 | 606 | 2244 |
| 08-1 | 20 | 66 | 66 | 503 | 503 | 2440 |
| 09-0 | 30 | 411 | 411 | 3961 | 3961 | 21135 |
| GRIPPER | | | | | | |
| 01 | 11 | 125 | 125 | 246 | 246 | 246 |
| 02 | 17 | 925 | 925 | 1842 | 1842 | 1842 |
| 03 | 23 | 5885 | 5885 | 11758 | 11758 | 11758 |
| 04 | 29 | 34301 | 34301 | 68586 | 68586 | 68586 |
| 05 | 35 | 188413 | 188413 | 376806 | 376806 | 376806 |
| 06 | 41 | 991229 | 991229 | 1982434 | 1982434 | 1982434 |
| 07 | 47 | 5046269 | 5046269 | 10092510 | 10092510 | 10092510 |
| LOGISTICS (IPC 2) | | | | | | |
| 4-0 | 20 | 159 | 408 | 1126 | 1780 | 2936 |
| 5-0 | 27 | 459 | 2391 | 5693 | 14370 | 21124 |
| 6-0 | 25 | 411 | 2160 | 5712 | 14485 | 23967 |
| 7-1 | 44 | 17617 | 111756 | 427944 | 1173096 | |
| 8-1 | 44 | 4843 | 27396 | 157645 | 558869 | |
| 9-0 | 36 | 2778 | 15878 | 61507 | 183826 | 460737 |
| 10-0 | 45 | 10847 | | | | |
| 11-0 | 48 | 10495 | | | | |
| MICONIC-SIMPLE-ADL | | | | | | |
| 1-0 | 4 | 4 | 4 | 4 | 4 | 4 |
| 2-1 | 6 | 6 | 22 | 26 | 26 | 26 |
| 3-1 | 10 | 58 | 102 | 102 | 102 | 102 |
| 4-2 | 14 | 148 | 280 | 470 | 560 | 560 |
| 5-1 | 15 | 209 | 759 | 1136 | 1326 | 1399 |
| 6-4 | 18 | 397 | 948 | 1936 | 2844 | 3436 |
| 7-4 | 23 | 3236 | 7654 | 11961 | 15780 | 16968 |
| 8-3 | 24 | 1292 | 5870 | 15188 | 25914 | 34315 |
| 9-3 | 28 | 20891 | 39348 | 39348 | 39348 | 39348 |
| 10-3 | 28 | 6476 | 16180 | 65477 | 129400 | 224495 |
| 11-3 | 32 | 58268 | 130658 | 258977 | 399850 | 497030 |
| 12-4 | 34 | 83694 | 181416 | 541517 | 970632 | 1640974 |
| 13-2 | 40 | 461691 | 947674 | 2203931 | 3443154 | 4546823 |
| MICONIC-STRIPS | | | | | | |
| 1-0 | 4 | 4 | 4 | 4 | 4 | 4 |
| 2-1 | 7 | 18 | 29 | 34 | 37 | 37 |
| 3-1 | 11 | 70 | 138 | 195 | 241 | 251 |
| 4-4 | 15 | 166 | 507 | 814 | 1182 | 1348 |
| 5-4 | 18 | 341 | 1305 | 2708 | 4472 | 5933 |
| 6-4 | 21 | 509 | 2690 | 7086 | 13657 | 21177 |
| 7-4 | 25 | 3668 | 13918 | 32836 | 61852 | 95548 |
| 8-3 | 28 | 4532 | 35529 | 97529 | 205009 | 349491 |
| 9-3 | 32 | 25265 | 114840 | 321202 | 700640 | 1239599 |
| 10-3 | 34 | 8150 | 97043 | 423641 | 1151402 | 2505892 |

Figure 6: Empirical results for IPC benchmark tasks.

Extrapolating from the expansion counts in the figure, it appears unlikely that a standard heuristic search approach can be used to solve GRIPPER tasks of size beyond 10–12, or reliably solve MICONIC tasks of size beyond 16–18.

## Discussion

We have presented an analysis of heuristic search algorithms for optimal sequential planning in some standard planning domains under the assumption of *almost perfect heuristics*. Theoretical and empirical results show an exponential cost increase as tasks grow larger. In many cases, such as the GRIPPER domain and a family of MICONIC tasks, there is no significant difference in node expansions between A$^*$ with an almost perfect heuristic and breadth-first search.

We argue that this is not just a theoretical problem. There is a barrier to the scaling capabilities of A$^*$-family algorithms, and current optimal heuristic planners are pushing against it. To break the barrier, other ideas are needed.

One possible source of such ideas is the literature on *domain-dependent (optimal) search*. For example, Junghanns and Schaeffer (2001) observe that their Sokoban solver *Rolling Stone* only solves 5 out of 90 problem in-

stances from the standard benchmark suite when using only the basic heuristic search algorithm with transposition tables. When coupled with other, domain-specific search enhancements, the total number of problem instances solved increases to 57 out of 90. Many of the techniques they present easily generalize to domain-independent planning.

However, several of the search enhancements they consider would not improve our analysis, which already makes a number of optimistic assumptions. For example, their use of *move ordering* only helps in the last search layer, which we optimistically ignored in our analysis. Their use of *deadlock tables* to detect and prune states with infinite heuristic values cannot improve the performance of our almost perfect heuristics, which by definition detect all infinite-heuristic states reliably (otherwise the heuristic error of these states would exceed the given constant). Other search enhancements, such as *overestimation* and *relevance cuts* either lose optimality or even completeness of the search algorithm.

A few techniques remain that may reduce the numbers in our analysis. These are mostly *forward pruning* techniques, which limit the set of allowed interleavings of independent parts of the solution (e. g., *tunnel macros*). However, these techniques are the ones that are most Sokoban-specific, and finding a widely useful generalization appears challenging. Some of these ideas are closely related to the concept of *partial-order reductions* in model checking, which are powerful for tree searches, but difficult to integrate into algorithms that prune duplicates such as A$^*$.

One type of partial-order reduction that has been studied before in the context of domain-independent planning and in our opinion deserves further attention is symmetry reduction (Fox and Long 1999). For example, by detecting and exploiting the equivalence of the balls of a GRIPPER task, we can easily solve arbitrarily large tasks in this domain in low-order polynomial time. However, clean, general ideas for exploiting symmetries in planning tasks are still few.

Another approach that we deem to be promising that has not been thoroughly explored yet is *problem simplification* (Haslum 2007). The main difficulty in optimal sequential planning does not lie in finding the optimal plan; it lies in proving that no shorter plan exists. Cutting down the number of possibilities for "wasting time" by performing irrelevant actions may be a key idea for this. This is another research area that is wide open for future work.

## Acknowledgments

This work was supported by the German Research Council (DFG) by DFG grant NE 623/10-1 and as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS). See http://www.avacs.org/ for more information.

## References

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

Dinh, H.; Russell, A.; and Su, Y. 2007. On the value of good advice: The complexity of A$^*$ search with accurate heuristics. In *Proc. AAAI 2007*, 1140–1145.

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

Fox, M., and Long, D. 1999. The detection and exploitation of symmetry in planning problems. In *Proc. IJCAI-99*, 956–961.

Gaschnig, J. 1977. Exactly how good are heuristics? Toward a realistic predictive theory of best-first search. In *Proc. IJCAI-77*, 434–441.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

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. 2007. Reducing accidental complexity in planning problems. In *Proc. IJCAI 2007*, 1898–1903.

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

Helmert, M.; Mattmüller, R.; and Röger, G. 2006. Approximation properties of planning benchmarks. In *Proc. ECAI 2006*, 585–589.

Helmert, M. 2003. Complexity results for standard benchmark domains in planning. *AIJ* 143(2):219–262.

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

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

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

Junghanns, A., and Schaeffer, J. 2001. Sokoban: Enhancing general single-agent search methods using domain knowledge. *AIJ* 129(1–2):219–251.

Kautz, H., and Selman, B. 1999. Unifying SAT-based and graph-based planning. In *Proc. IJCAI-99*, 318–325.

Korf, R. E.; Reid, M.; and Edelkamp, S. 2001. Time complexity of iterative-deepening A$^*$. *AIJ* 129:199–218.

Korf, R. E. 1985. Depth-first iterative-deepending: An optimal admissible tree search. *AIJ* 27(1):97–109.

Pearl, J. 1984. *Heuristics: Intelligent Strategies for Computer Problem Solving*. Addison-Wesley.

Pohl, I. 1977. Practical and theoretical considerations in heuristic search algorithms. In Elcock, E. W., and Michie, D., eds., *Machine Intelligence 8*. Ellis Horwood. 55–72.

Yoshizumi, T.; Miura, T.; and Ishida, T. 2000. A$^*$ with partial expansion for large branching factor problems. In *Proc. AAAI 2000*, 923–929.

Zhou, R., and Hansen, E. A. 2006. Breadth-first heuristic search. *AIJ* 170(4–5):385–408.