# Local Search Topology in Planning Benchmarks: A Theoretical Analysis

**Jörg Hoffmann**
Institute for Computer Science
Albert Ludwigs University
Georges-Köhler-Allee, Geb. 52
79110 Freiburg, Germany
hoffmann@informatik.uni-freiburg.de

## Abstract

Many state-of-the-art heuristic planners derive their heuristic function by relaxing the planning task at hand, where the relaxation is to assume that all delete lists are empty. The success of such planners on many of the current benchmarks suggests that in those task's state spaces relaxed goal distances yield a heuristic function of high quality. Recent work has revealed empirical evidence confirming this intuition, stating several hypotheses about the local search topology of the current benchmarks, concerning the non-existence of dead ends and of local minima, as well as a limited maximal distance to exits on benches.

Investigating a large range of planning domains, we prove that the above hypotheses do in fact hold true for the majority of the current benchmarks. This explains the recent success of heuristic planners. Specifically, it follows that FF's search algorithm, using an idealized heuristic function, is polynomial in (at least) eight commonly used benchmark domains. Our proof methods shed light on what the structural reasons are behind the topological phenomena, giving hints on how these phenomena might be automatically recognizable.

## Introduction

In the last three years, planning systems based on the idea of heuristic search have been very successful. At the AIPS-1998 planning systems competition, HSP1 compared well with the other systems (McDermott 2000), and at the AIPS-2000 competition, out of five prize-winning fully automatic planners, FF and HSP2 were based on heuristic search, while another two, Mips and STAN, were hybrids that incorporated, amongst other things, heuristic search (Bacchus 2001).

Interestingly, four of these five planners use the same base approach for deriving their heuristic functions: they relax the planning task description by ignoring all delete lists, and estimate, to each search state, the length of an optimal relaxed solution to that state. This idea has first been proposed by Bonet et al. (1997). The length of an optimal relaxed solution would yield an admissible heuristic. However, as was proven by Bylander (1994), computing the optimal relaxed solution length is still NP-hard. Therefore, Bonet et al. introduced a technique for approximating optimal relaxed solution length, which they use in both versions of HSP (Bonet & Geffner 2001). The heuristic engines in FF (Hoffmann & Nebel 2001) and Mips (Edelkamp & Helmert 2001) use different approximation techniques.

Three of the above planners, HSP1, FF, and Mips, use their heuristic estimates in variations of local search algorithms, where the search space to a task is the state space, i.e., the space of all states that are reachable from the initial state. Now, the behavior of local search depends crucially on the topology of the search space (as has been studied in the SAT community, for example by Frank et al. (1997)). Thus, the success of these heuristic planners on many planning tasks suggests that in those task's state spaces relaxed goal distances yield a heuristic function of high quality. Recent work has revealed empirical evidence confirming this intuition. By computing the optimal relaxed solution length to reachable states in small planning tasks from the competition domains, and measuring parameters of the resulting local search topology, the following was found (Hoffmann 2001b). In the majority of the domains, the investigated instances did not contain any dead ends (states from which the goal is unreachable), and neither did they contain any local minima (regions of the state space where all neighbours look worse). Sometimes all instances in a domain had the same constant maximal exit distance (roughly, the maximal distance to a state with better evaluation). It was hypothesized that these observations on the *small* example instances carry over directly to *all* instances in the respective domains (Hoffmann 2001b).

In the presented investigation, we consider 20 often used planning benchmark domains, including all competition examples. We prove that the majority of these domains do in fact have the aforementioned topological properties, confirming all of the above hypotheses except one. The results give a strong argument for interpreting the recent success of heuristic planners as utilizing the topology of the benchmarks, as was suggested by the previous empirical work (Hoffmann 2001b). Specifically, it follows that FF's search algorithm is a polynomial solving mechanism in eight of the domains, under the idealizing assumption that its heuristic identifies the optimal relaxed distances. What's more, our proof methods shed light on which structural properties—at the level of the planning task's definition—are responsible for the topological phenomena. As most of the structural properties we identify are of a syntactical nature, this knowledge gives hints as to how the topological phenomena might be automatically recognizable.

The full details of the investigation form a long article that is available as a technical report (Hoffmann 2001a). Here, we summarize the definitions, and identify the key ideas in the form of proof sketches. The paper is organized as follows. The next section gives the background. We then include a section presenting the key lemmata underlying our proofs; the three sections after that prove the topological phenomena concerning dead ends, local minima, and maximal exit distance, respectively. Afterwards, one section gives the overall picture that our results determine, before we finish the paper by concluding and pointing to future research directions.

## Background

Background is necessary on the planning framework, the investigated domains, local search topology, and the previous empirical work.

### Planning Framework

To enable theoretical proofs to properties of planning *domains* rather than single tasks, we have developed a formal framework for STRIPS and ADL domains, formalizing in a straightforward manner the way how domains are usually handled in the community. The only other work we know of that uses a formal notion of planning domains is recent work by Malte Helmert (2001). There, the semantics of different transportation domains are formalized in order to prove their computational complexity. In difference to that, our definitions are strictly syntax-oriented—after all, the heuristic function we consider is extracted directly from the syntax of the planning task. We only summarize the rather lengthy definitions here, and refer the reader to our technical report (Hoffmann 2001a) for details.

A planning domain is defined in terms of a set of *predicate symbols*, a set of *operators*, and a set of *instances*. All logical constructs in the domain are based on the set of predicate symbols. The operators are ($k$-ary, where $k$ is the number of operator parameters) functions from the set of all objects into the set of all STRIPS or ADL actions. A *STRIPS action* $a$ is the usual triple $(pre(a), add(a), del(a))$ of fact sets; an *ADL action* consists of a first order logical formula without free variables as precondition, and a set of effects of the form $(con, add, del)$ where $con$ can again be a formula without free variables. An instance of a domain is defined in terms of a set of objects, an *initial state*, and a *goal condition*. The initial state is a set of facts, and the goal condition can be an arbitrary formula without free variables. An instance together with the respective operators constitutes a propositional planning task $(A, I, G)$ where the action set $A$ is the result of applying the operators to the objects, and the initial state $I$ and goal condition $G$ are those of the instance. We identify instances with the respective propositional tasks. The *result* $Result(s, a)$ of applying a STRIPS or ADL action $a$ to a state $s$ is defined in the usual manner: the result is defined only if the precondition is true in $s$; in that case, the action's add effects are made true and the delete effects are made false—for an ADL action, only those effects are applied that have their condition fulfilled in $s$. A *plan, or solution, for a task* $(A, I, G)$ is a sequence of actions $P \in A^*$ that, when successively applied to $I$, yields a goal state, i.e., a state that fulfills $G$. $P$ is *optimal* if there is no shorter plan for $(A, I, G)$.

We investigate topological properties that arise when using the optimal relaxed solution length as a heuristic function. We name that function $h^+$. It is formally defined as follows. For any state $s$ that is reachable in a propositional planning task $(A, I, G)$, the *relaxed task to* $s$ is $(A^+, s, G)$: the task defined by the initial state $s$, the original goal condition, and the original action set except that all delete lists are empty. Then, $h^+(s)$ is the length of an optimal plan for $(A^+, s, G)$ or $h^+(s) = \infty$ if there is no such plan. We will frequently make use of the following abbreviations: if a sequence of actions $P^+$ is a plan for $(A^+, s, G)$, then we also say that $P^+$ is a *relaxed plan for* $(A, s, G)$, or short a *relaxed plan for* $s$. [1] To give an example for a relaxed plan, consider the *Gripper* domain, as it was used in the AIPS-1998 competition. A real solution picks up two balls in room A, moves to room B, drops the balls, moves back, and does the same again until all balls have been transported. A relaxed plan simply picks up all balls with the same hand—the *free* predicate for the hand is not deleted—moves to room B, and drops all balls. While this might seem very simplistic, we will see, in fact prove, that it yields a high-quality heuristic function in a lot of benchmark domains.

### Investigated Domains

We investigate the properties of 20 different STRIPS and ADL benchmark domains, including all 13 domains that have been used in the AIPS-1998 and AIPS-2000 planning system competitions. The competition domains are *Assembly*, *Blocksworld-arm*, *Freecell*, *Grid*, *Gripper*, *Logistics*, *Miconic-ADL*, *Miconic-SIMPLE*, *Miconic-STRIPS*, *Movie*, *Mprime*, *Mystery*, and *Schedule*. We assume that the reader is familiar with these domains, and do not describe them here. Descriptions can be looked up in the articles on the competitions (McDermott 2000; Bacchus 2001), and details are in our technical report (Hoffmann 2001a). Apart from the 13 competition domains, we investigate 7 more benchmark domains often used in the literature. These domains can be briefly described as follows.

1. *Blocksworld-no-arm*: unlike in the competition version, this encoding does not use an explicit robot arm; instead, blocks are moved around directly by operators moving them from one block to another block, or from the table to a block, or from a block to the table.
2. *Briefcaseworld*: transportation domain using conditional effects; objects can be put into or taken out of the (single)

[1] Ignoring the delete lists simplifies a task only if all formulae are negation free. In STRIPS, this is the case by definition. In general, for a fixed domain, any task can be polynomially transformed to have that property: compute the negation normal form to all formulae (negations only in front of atoms), then introduce for each negated atom $\neg B$ a new atom not-$B$ and make sure it is true in a state iff $B$ is false (Gazen & Knoblock 1997).

briefcase, and a move operator, which can be applied between any two locations, moves all objects along that are currently inside.

3. *Ferry*: also a transportation domain, with operators that board a car onto the (single) ferry, or debark a car from it; the ferry can only transport one car at a time, and a sail operator can be applied to move the ferry between any two locations.
4. *Fridge*: for a number of fridges, the broken compressors must be replaced. This involves un-fastening a number of screws that hold the compressors, removing the old compressors and attaching the new ones, and fastening all screws again.
5. *Hanoi*: encoding of the classical Towers of Hanoi problem, using a move$(x, y, z)$ operator to move a disc $x$ from a disc $y$ to a disc $z$ (the pegs are encoded as discs that can not be moved).
6. *Simple-Tsp*: a trivial version of the TSP problem, where the goal is that all locations have been visited, and a move operator can be applied between any two locations (all action costs being equal, as usual in STRIPS).
7. *Tyreworld*: a number of flat tyres must be replaced, which involves inflating the spare tyres, loosening the nuts, and in turn jacking up the hubs, undoing the nuts, removing the flat tyre and putting on the spare one, doing up the nuts, and jacking down the hub again; finally, all nuts must be tightened, and the tools must be put away into the boot.

For formally defining a domain, one must amongst other things decide what exactly the instances are. For almost none of the investigated domains is there such a definition in the literature. The obvious approach we have taken is to abstract from the known example suits. Full details for all domains are given in our technical report (Hoffmann 2001a).

## Local Search Topology

We now define a number of topological phenomena that are relevant for local search. The definitions are summarizations of what was given in the previous empirical work (Hoffmann 2001b), slightly simplified to improve readability; details are in the technical report (Hoffmann 2001a). A propositional planning task is associated with its *state space* $(S, E)$, which is a graph structure where $S$ are all states that are reachable from the initial state, and $E$ is the set of all pairs $(s, s') \in S \times S$ of states where there is an action that leads to $s'$ when executed in $s$. The *goal distance* $gd(s)$ for a state $s \in S$ is the length of a shortest path in $(S, E)$ from $s$ to a goal state, or $gd(s) = \infty$ if there is no such path. In the latter case, $s$ is a *dead end*.

When there are single-directed state transitions, there can be dead ends. A dead end $s$ is *recognized* if $h^+(s) = \infty$, and *unrecognized* otherwise. To explain that terminology, note that $h^+(s) = \infty \Rightarrow gd(s) = \infty$: if a task can not be solved even when ignoring the delete lists, then the task is unsolvable. With respect to dead ends, any state space falls into one of the following four classes: the state space is

1. *undirected*, if $\forall (s, s') \in E : (s', s) \in E$,
2. *harmless*, if $\exists (s, s') \in E : (s', s) \notin E$, and $\forall s \in S : gd(s) < \infty$,
3. *recognized*, if $\exists s \in S : gd(s) = \infty$, and $\forall s \in S : gd(s) = \infty \Rightarrow h^+(s) = \infty$,
4. *unrecognized*, if $\exists s \in S : gd(s) = \infty \wedge h^+(s) < \infty$.

In the first class, there can be no dead ends because everything can be undone; in the second class, some things can not be undone, but those single-directed state transitions do not do any harm; in the third class, there are dead end states but all of them are recognized by the heuristic function. The only critical case for local search is class four, where a local search algorithm can run into an unrecognized dead end, and be trapped.

A different way of getting trapped is when local search ends up in a region of the state space where all neighbors look worse from the point of view of the heuristic function, i.e., when search encounters a local minimum: with all neighbors looking worse, it is not clear in which direction search should proceed. The formal definition of local minima, and of benches below, follows the definitions of Frank et al. (1997) for SAT problems; in difference to the *undirected* search spaces Frank et al. consider, we need to take care of single-directed state transitions. The adapted definitions are the following. A *flat path* is a path in $(S, E)$ on which the heuristic value does not change. A *plateau of level* $l$ is a set of states that have the same heuristic value $l$, and that form a strongly connected component in $(S, E)$. An *exit of a plateau* is a state $s$ that can be reached from the plateau on a flat path, and that has a better evaluated neighbor, i.e., $(s, s') \in E$ with $h^+(s') < h^+(s)$. A *local minimum* is a plateau of level $0 < l < \infty$ that has no exits. Note that we allow exits to not lie on the plateau itself, namely when the flat path leaves the plateau (which can happen if there is a single-directed state transition to a state with the same $h^+$ value); the main characteristic of local minima is that, starting from them, one must temporarily increase the heuristic value in order to improve it.

Finally, local search can get lost on large flat regions of the state space, usually referred to as benches (Frank, Cheeseman, & Stutz 1997). These are regions from which the heuristic value can be improved without temporarily increasing it, i.e., plateaus with exits. The hard bit for local search is to find the exits. The difficulty of doing this can be assessed by a variety of parameters like the size of the bench, or the exit percentage; in the previous empirical work (Hoffmann 2001b), the so-called maximal exit distance was measured. This parameter is especially relevant for FF's search algorithm, as will be explained in the next subsection. The definition is as follows. The *exit distance* of any state is the length of a shortest flat path connecting the state to an exit, or $\infty$ if there is no such path. The *maximal exit distance* in a state space is the maximum over the exit distances of all states $s$ with $h^+(s) < \infty$, i.e., we ignore recognized dead ends—these can be skipped by search anyway. Note that states on local minima have infinite exit distance.

## Previous Work: The Hypotheses

As described above, previous work has been done on empirically investigating topological properties in the competition domains, with respect to $h^+$ (Hoffmann 2001b); the approach being to compute $h^+$ for the states in example state spaces and measure parameters of the resulting local search topology. Because computing $h^+$ is NP-hard, the investigation was restricted to small instances. Amongst other things, the measured parameters were the dead-end class of the instances, the number of states on local minima, and the maximal exit distance. The observations are summarized in the table shown in Figure 1.

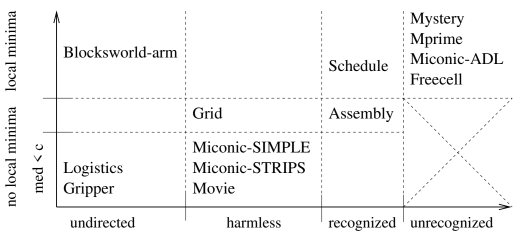


Figure 1: Overview of the empirical observations in small instances of the competition domains.

The x-axis in Figure 1 shows the different dead-end classes by increasing difficulty (for the domains in a dead-end class $i$, all investigated instances belong to a class $j \leq i$, and at least one instance belongs to class $i$). The y-axis combines local minima percentage with maximal exit distance: in the uppermost part of the table, the domains are shown were local minima were found; below that, domains are shown were there were no local minima at all in the small instances (unrecognized dead ends imply the existence of local minima, so that part of the table is crossed out); in the lower most part of the table domains are shown were all instances had the same constant maximal exit distance (remember that this parameter is infinity in the presence of local minima).

The table in Figure 1 was named the *planning domain taxonomy*, and it was hypothesized that these observations hold true for *all* instances in the respective domains: that all instances of a domain shown in dead-end class $i$ are in a class $j \leq i$; that no instance of a domain shown in the lower parts of the table contains any local minima; that for a domain shown in the lowest part there is a constant $c$ such that the maximal exit distance in all instances is at most $c$. The practical relevance of this is the following. The domains on the left bottom side of the taxonomy are "simple" for heuristic planners like HSP and FF because there the $h^+$ function, which those planners approximate, is a heuristic function of high quality. The majority of the competition domains seem to lie on the simple side. So, if these observations carry over to all instances in the respective domains, then the taxonomy can serve as an explanation for the good behavior of the aforementioned planners.

For domains in the lower most part of the taxonomy, FF's search algorithm is in fact polynomial in the sense that (assuming the heuristic function is given) it looks at polynomially many states before reaching the goal (Hoffmann 2001b). FF's search algorithm is the following.

$s := I$
**while** $h^+(s) \neq 0$ **do**
  do breadth first search for $s'$, $h^+(s') < h^+(s)$
  $s := s'$
**endwhile**

Without local minima each iteration of this algorithm crosses a bench so breadth first search finds a better state at maximal depth $c + 1$; the branching factor is limited by the number of actions in the task; each iteration improves the heuristic value by at least one, so after at most $h^+(I)$ iterations a goal state is reached.

In the subsequent investigation, we verify how much truth there is in the hypotheses issued from observations on small examples. Going beyond that, we look at a larger number of domains, and determine exactly which part of the taxonomy they belong to. Obviously, we need only prove positive results (like the non-existence of local minima) for those domains were the empirical investigation did not reveal a negative example. As it turns out, all of the hypotheses are true except those about the *Assembly* domain. Like in the previous investigation, our main aim is to explain the good performance of local search planners, so we focus on *solvable* instances only—on unsolvable instances, local search is lost anyway.

# A Theoretical Core

The following is the core of our theory, i.e., the key lemmata underlying our proofs in the single domains. This simplifies the subsequent proofs, and gives an insight into what the main structural reasons are behind the topological phenomena that we identify. The lemmata formulate sufficient criteria implying that (the state space of) a planning task has certain topological properties. Proofs for domains will proceed by applying the lemmata to arbitrary instances. For the sake of simplicity, we give our definitions only for STRIPS tasks. The proofs for ADL tasks are along the same lines of argumentation.

## Dead Ends

We first identify sufficient criteria for a planning task containing no dead ends. Our starting point is a reformulated version of results published by Koehler and Hoffmann (2000). We need the notion of *inconsistency*, which is defined as follows. Two facts are inconsistent if there is no reachable state that contains both of them. Two sets of facts $F$ and $F'$ are inconsistent if each fact in $F$ is inconsistent with at least one fact in $F'$.

If to each action there is an inverse action that undoes the action's effects, then the corresponding state space is undirected.

**Definition 1** *Given a planning task* $(A, I, G)$. *An action* $a \in A$ *is* invertible*, if*

1. *there is an action* $\overline{a} \in A$ *such that*
   (a) $pre(\overline{a}) \subseteq (pre(a) \cup add(a)) \setminus del(a)$,
   (b) $add(\overline{a}) = del(a)$, *and*

*(c)* $del(\overline{a}) = add(a)$,

2. $add(a)$ *is inconsistent with* $pre(a)$*, and*
3. $del(a) \subseteq pre(a)$.

**Lemma 1** *Given a planning task* $(A, I, G)$*. If all actions* $a \in A$ *are invertible, then the state space to the task is undirected.*

**Proof Sketch:** For any state $s$ and applicable action $a$, $\overline{a}$ is applicable in $Result(s, a)$ due to part 1(a) of Definition 1. Parts 2 and 3 of that definition make sure that $a$'s effects do in fact appear, and parts 1(b) and (c) make sure that $\overline{a}$ undoes exactly those effects. ■

The next is a new criterion that is weaker and only implies the non-existence of dead ends. For an action not to lead into such a dead end, it is already sufficient if the inverse action re-achieves *at least* what has been deleted, and does not delete any facts that have been true before.

**Definition 2** *Given a planning task* $(A, I, G)$*. An action* $a \in A$ *is* at least invertible*, if there is an action* $\overline{a} \in A$ *such that*

1. $pre(\overline{a}) \subseteq (pre(a) \cup add(a)) \setminus del(a)$,
2. $add(\overline{a}) \supseteq del(a)$*, and*
3. $del(\overline{a})$ *is inconsistent with* $pre(a)$.

Note that the previous Definition 1 is strictly stronger than Definition 2: if $del(\overline{a}) = add(a)$, and $add(a)$ is inconsistent with $pre(a)$, then, of course, $del(\overline{a})$ is inconsistent with $pre(a)$.

Another reason for an action not leading into a dead end is this. If the action must be applied at most once (because its add effects will remain true), and it deletes nothing but its own preconditions, then that action needs not be inverted.

**Definition 3** *Given a planning task* $(A, I, G)$*. An action* $a \in A$ *has* static add effects*, if*

$$add(a) \cap \bigcup_{a' \in A} del(a') = \emptyset$$

*An action has* irrelevant delete effects*, if*

$$del(a) \cap (G \cup \bigcup_{a \neq a' \in A} pre(a')) = \emptyset$$

If all actions in a task are either at least invertible or have static add- and irrelevant delete-effects, then the state space is at most harmless.

**Lemma 2** *Given a solvable planning task* $(A, I, G)$*. If it holds for all actions* $a \in A$ *that either*

1. *a is at least invertible, or*
2. *a has static add effects and irrelevant delete effects,*

*then there are no dead ends in the state space to the task, i.e.,* $gd(s) < \infty$ *for all* $s \in S$.

**Proof Sketch:** For any state $s = Result(I, P)$ a plan can be constructed by inverting $P$ (applying the respective inverse actions in the inverse order), and executing an arbitrary plan for $(A, I, G)$ thereafter. In the first process, actions that are not (at least) invertible can be skipped: for those the second prerequisite holds true, so once they are applied their add effects remain true and their delete effects are no longer needed. In the second process, all actions are safely applicable except those not invertible ones that have been skipped in the first process. But for the same reasons as outlined above those actions need not be applied anyway. ■

The two properties handled so far, undirected and harmless state spaces, are properties of the planning tasks themselves, independent of the $h^+$ function. Different from that, the third dead end class, recognized dead ends, *does* depend on the heuristic function. We did, however, not find a general sufficient criterion for this case. Anyway, only two of our domains, *Schedule* and *Assembly*, belong to that class according to the hypotheses, and as it will turn out for *Assembly* the hypothesis is wrong.

## Local Minima

We now identify a sufficient criterion for the non-existence of local minima under evaluation with $h^+$. As will be shown, the criterion can be directly applied to (the instances of) 6 of our 20 domains, and can be applied with slight modifications to 4 more domains. The criterion is based on actions that fulfill a weak notion of invertibility, and that are respected by the relaxation in the sense defined below.

When using a relaxed action to invert an action's effects, it is already enough if the inverse action re-achieves all facts that have been deleted—as the delete effects of the inverse action will be ignored anyway, there needs be no constraint about their form.

**Definition 4** *Given a planning task* $(A, I, G)$*. An action* $a \in A$ *is* at least relaxed invertible*, if there is an action* $\overline{a} \in A$ *such that*

1. $pre(\overline{a}) \subseteq (pre(a) \cup add(a)) \setminus del(a)$,
2. *and* $add(\overline{a}) \supseteq del(a)$.

Note that the previous Definitions 1 and 2 are strictly stronger than Definition 4.

The following property is the key behind the non-existence of local minima in most of our domains: actions that are good for the real task are also good for the relaxed task.

**Definition 5** *Given a solvable planning task* $(A, I, G)$*. An action* $a \in A$ is respected by the relaxation *if, for any reachable state s such that a starts an optimal plan for* $(A, s, G)$*, there is an optimal relaxed plan for* $(A, s, G)$ *that also starts with a.*

As a simple example for an action that is respected by the relaxation, consider picking up a ball in *Gripper*: if that action starts an optimal plan, then the ball must be transported; any relaxed plan needs to transport the ball, and there is no other way of accomplishing this (except using the other hand, which does not yield a shorter solution). A similar argument applies to loading or unloading a truck in *Logistics*: the only way of transporting a package within its initial or destination city is by using the local truck, so *all* plans, real or relaxed, must use the respective action.

**Lemma 3** *Given a solvable planning task $(A, I, G)$, such that the state space does not contain unrecognized dead ends. If each action $a \in A$ either*

1. *is respected by the relaxation and at least relaxed invertible, or*
2. *has irrelevant delete effects,*

*then there are no local minima in the state space under evaluation with $h^+$.*

**Proof Sketch:** States $s$ where $gd(s) = \infty$ have $h^+(s) = \infty$ by prerequisite and are therefore not on local minima. We show below that, from states $s$ with $0 < gd(s) < \infty$, $h^+$ decreases monotonically on any optimal path to the goal. This finishes the argument: the goal state has a better $h^+$ value than $s$, and the path to it does not increase.

Say an action $a$ starts an optimal plan in a state $s$. We identify, for the successor state $Result(s, a)$, a relaxed plan that has at most the same length as an optimal relaxed plan for $s$. If the first case of the prerequisite holds, then $a$ starts an optimal relaxed plan $P^+$ for $s$. A relaxed plan for $Result(s, a)$ can be constructed by replacing $a$ in $P^+$ with the inverse action $\overline{a}$: the inverse action is applicable in $Result(s, a)$, and it re-achieves all facts that $a$ has deleted. In the second case of the prerequisite, if $a$ has irrelevant delete effects, we distinguish two cases. Let $P^+$ be an optimal relaxed plan for $s$. First case, $a$ is contained in $P^+$: then it can be removed from $P^+$ to form a relaxed plan for $Result(s, a)$, as $a$ does not delete anything that is needed by other actions. Second case, $a$ is not contained in $P^+$: then $P^+$ is still a relaxed plan for $Result(s, a)$ due to the same reason. ■

Planning tasks where there are unrecognized dead ends do contain local minima anyway (Hoffmann 2001b), so we must postulate that this is not the case; like when the task at hand is undirected or harmless.

## Maximal Exit Distance

Concerning the maximal exit distance, we remark only the following simple property which underlies our proof technique.

**Proposition 1** *Given a planning task $(A, I, G)$, a reachable state $s$, and an action $a$ that starts an optimal relaxed plan $P^+$ for $(A, s, G)$. If removing $a$ from $P^+$ yields a relaxed plan for $Result(s, a)$, then $h^+(Result(s, a)) < h^+(s)$, i.e., $s$ is an exit state under $h^+$.*

The prerequisite holds, for example, if the action $a$ is respected by the relaxation and has irrelevant delete effects.

In the following sections, we will focus on dead ends, local minima, and maximal exit distance in turn. For each of these three phenomena, we summarize our proofs for all domains in a single proof sketch. This improves readability, and makes it easier to see the common ideas behind the proofs.

## Dead Ends

We first prove to which dead-end class our domains belong. Remember that we need only consider those domains where the empirical work did not reveal a negative example (like the unrecognized dead ends in *Freecell*, *Miconic-ADL*, *Mprime*, and *Mystery*). Most of the proofs are simple applications of the lemmata presented in the previous section.

**Theorem 1** *The state space to any solvable planning task belonging to the*

1. *Blocksworld-arm, Blocksworld-no-arm, Briefcaseworld, Ferry, Fridge, Gripper, Hanoi, or Logistics domains is undirected,*
2. *Grid, Miconic-SIMPLE, Miconic-STRIPS, Movie, Simple-Tsp, or Tyreworld domains is harmless,*
3. *Schedule domain is recognized under evaluation with $h^+$.*

**Proof Sketch:** All actions in *Blocksworld-arm*, *Blocksworld-no-arm*, *Ferry*, *Gripper*, *Hanoi*, and *Logistics* instances are invertible in the sense of Definition 1, so we can apply Lemma 1 and are finished. In the *Briefcaseworld* and *Fridge* domains, while not strictly obeying the syntax of Definition 1, there is still always an action leading back to the state one started from.

In the *Movie*, *Simple-Tsp*, and *Tyreworld* domains, all actions are either at least invertible in the sense of Definition 2 or have irrelevant delete effects and static add effects in the sense of Definition 3, so Lemma 2 can be applied. In the *Grid*, *Miconic-SIMPLE*, and *Miconic-STRIPS* domains, while not strictly adhering to these definitions, similar arguments prove the non-existence of dead ends: in *Grid*, to all actions there is an inverse action, except opening a lock; the latter action excludes only other actions opening the same lock (similar to irrelevant deletes), and each lock needs to be opened at most once, as locks can not be closed (static add effects). In the *Miconic* domains, moving the lift can be inverted; letting passengers in- or out of the lift can not be inverted (as the passengers will only get in or out at their respective origin or destination floors), but those actions need to be applied at most once (similar to static add effects) and they do not interfere with anything else (similar to irrelevant deletes).

In *Schedule*, any state $s$ with $gd(s) < \infty$ can be solved by applying a certain sequence of working steps to each part in turn. If that sequence can not be applied for some part $p$—which must be the case in any dead end state—then it follows that this part is hot in $s$. No operator adds the fact that a part is cold. But from the dead end state $s$ at least one needed working step requires $p$ being cold as a precondition. It follows that there can be no relaxed solution to $s$ either, as the relaxation does not improve on the add effects. ■

In *Assembly*, one *can* construct an unrecognized dead end state, falsifying the hypothesis that all dead ends are recognized there. We have proven that the construction of an unrecognized dead end involves complex interactions between the ordering constraints that can be present in *Assembly*. These complex interactions are not likely to appear when ordering constraints are sparse like in the AIPS-1998 benchmark suit, and the interactions are particularly unlikely to appear in small instances as were used in the previous investigation. We refer the interested reader to the technical report (Hoffmann 2001a) for details. As the existence of unrecognized dead ends implies the existence of local minima, in consequence the hypothesis that there are no local minima in *Assembly* is also falsified.

## Local Minima

Like before, there is no need to prove anything where the empirical work already revealed a negative example. Most of our positive results concerning local minima are proven by application, or along the lines of, Lemma 3. A few results make use of rather individual properties of the respective domains.

**Theorem 2** *The state space of any solvable planning task belonging to the Blocksworld-no-arm, Briefcaseworld, Ferry, Fridge, Grid, Gripper, Hanoi, Logistics, Miconic-SIMPLE, Miconic-STRIPS, Movie, Simple-Tsp, or Tyreworld domains does not contain any local minima under evaluation with* $h^+$.

**Proof Sketch:** With Theorem 1, none of those domains contains unrecognized dead ends. As follows from the theorem's proof sketch, all actions in the *Ferry*, *Gripper*, *Logistics*, *Movie*, *Simple-Tsp*, and *Tyreworld* domains are either at least relaxed invertible, or have irrelevant delete effects. With Lemma 3 it suffices to show that all actions are respected by the relaxation. In *Movie*, if a snack is bought in an optimal plan then the snack must also be bought in any relaxed plan, likewise for rewinding the movie or resetting the counter; in *Simple-Tsp*, the optimal plan visits a location that is not yet visited, and any relaxed plan must also visit that location; in *Tyreworld*, if some working step has not yet been done then the relaxed plan must also do it; the *Ferry*, *Gripper*, and *Logistics* domains are all variations of the transportation theme, with actions that load objects onto vehicles, actions that move the vehicles, and actions that unload objects. As an example proof, consider *Logistics*: if an optimal plan loads or unloads some package then that package must still be transported and the relaxed plan has no better option of doing so; if an optimal plan moves a vehicle then that vehicle must either deliver or collect some package, and again the relaxed plan has no better choice.

In the *Fridge*, *Miconic-SIMPLE*, and *Miconic-STRIPS* domains, the actions do not adhere strictly to invertibility according to Definitions 3 and 4; but we have seen that they have similar semantics, i.e., they can either be inverted, or delete only facts that are no longer needed once they are applied. Furthermore, all actions in these domains are respected by the relaxation: in *Fridge*, similar to *Tyreworld*, missing working steps must also be done in the relaxed plan; in *Miconic-SIMPLE* and *Miconic-STRIPS* similar arguments like above for the transportation domains apply.

In *Briefcaseworld*, all actions can be inverted. Actions that move the briefcase or put in objects are respected by the relaxation due to the transportation arguments. Taking out objects is not respected because the objects already have their *at*-relation added (as a conditional effect) by moving the briefcase. However, taking out an object does not delete important facts if that object is already at its goal location. Thus, in a state $s$ where an optimal plan starts with a take out action, an optimal relaxed plan for $s$ can also be used for the successor state. It follows that $h^+$ does not increase on optimal solution paths.

For the remaining three domains, the proofs are more sophisticated. In all cases it can be proven that there is a path to the goal on which $h^+$ does not increase. In *Blocksworld-no-arm*, if an optimal starting action $a$ stacks a block into its goal position, then $a$ also starts an optimal relaxed plan. If there is no such action $a$ in a state $s$, then one optimal plan starts by putting some block $b$—that must be moved—from some block $c$ onto the table, yielding the state $s'$. Any relaxed plan $P^+$ for $s$ also moves $b$. To obtain a relaxed plan for $s'$, that moving action can be replaced in $P^+$ by moving $b$ from the table instead of from $c$. So in all states there is an optimal starting action leading to a state with equal or less $h^+$ value.

In *Grid*, a rather complex procedure can be applied to identify a flat path to a state with better $h^+$ value. In a state $s$, let $P^+$ be an optimal relaxed plan for $s$, and $a$ the first unlock action in $P^+$ or a putdown if there is no such unlock action (the last action in $P^+$ is a putdown without loss of generality, as the only goals are to have some keys at certain locations). Identifying a flat path to a state $s'$ where $a$ can be applied suffices with Proposition 1: unlocking deletes only facts that are irrelevant once the lock is open, and the deletes of putting down a key are irrelevant if there are no more locks that must be opened. The selected action $a$ uses some key $k$ at a position $x$. $P^+$ must contain a sequence of actions moving to $x$. Moving along the path defined by those actions does not increase $h^+$: those actions are contained in an optimal relaxed plan, and they can be inverted. If $k$ is already held in $s$, then we can now apply $a$. If the hand is empty in $s$, or some other key is held, then one can use $P^+$ to identify, in a similar fashion, a flat path to a state where one *does* hold the appropriate key $k$.

In *Hanoi*, it can be proven that the optimal relaxed solution length for any state is equal to the number of discs that are not yet in their goal position. As no optimal plan moves a disc away from its goal position, $h^+$ does thus not increase on optimal solution paths. ■

## Maximal Exit Distance

We finally present our results concerning the maximal exit distance. The proof technique is to walk along optimal solution paths until an action is reached whose delete effects are no longer needed once it is applied.

**Theorem 3** *To any of the Ferry, Gripper, Logistics, Miconic-SIMPLE, Miconic-STRIPS, Movie, Simple-Tsp, or Tyreworld domains, there is a constant c such that, for all solvable tasks belonging to that domain, the maximal exit distance in the task's state space is at most c under evaluation with* $h^+$.

**Proof Sketch:** We have seen that in all these domains the actions are respected by the relaxation, and can either be inverted or have irrelevant deletes. If $s$ is a state and $a$ an action starting an optimal plan for $s$, then $a$ starts an optimal relaxed plan $P^+$ for $s$, and a relaxed plan for $Result(s, a)$ can be constructed by either: replacing $a$ in $P^+$ by the respective inverse action, which re-achieves $a$'s delete effects; or by removing $a$ entirely, which can be done if the delete effects of $a$ are not needed by $P^+$. In the latter case, $h^+(Result(s, a)) < h^+(s)$ follows (as is stated by Proposition 1). So it suffices to derive a constant number $c$ of steps on any optimal solution path such that after $c$ steps there is an optimal starting action for which the second case holds.

In *Movie*, all actions have no, and therefore irrelevant, delete effects, with the single exception of rewinding the movie (which deletes the counter being at zero). Obviously, no optimal plan rewinds the movie twice in a row. Thus, $c = 1$ is the desired upper limit.

In *Simple-Tsp*, $c = 0$ suffices. With the terminology two paragraphs above, say we are at location $l$ in $s$. Any optimal plan starts by visiting a yet unvisited location $l'$. A relaxed plan for $Result(s, a)$ can be constructed by removing $a$ from $P^+$, and replacing all moves from $l$ to some $l''$ with moves from $l'$ to $l''$.

In the transportation domains *Ferry*, *Gripper*, *Logistics*, *Miconic-SIMPLE*, and *Miconic-STRIPS*, the argument is the following. If the optimal plan starts with an unload- or load-type of action, then that action can be removed from $P^+$ to form a relaxed plan for $Result(s, a)$: for unloads, once an object is where you wanted it to be, you don't need to have it in the vehicle anymore; similarly for loads, once the object is inside the appropriate vehicle, you do not need it anymore at its origin location (in *Gripper*, "loading" a ball also deletes the respective hand being free; however, the hand is made free again anyway by the relaxed plan, when it puts the ball into its goal location; a similar argument applies in *Ferry*). Concerning moves, in all these domains all locations are immediately accessible from all other locations (for the appropriate type of vehicle), so no optimal plan moves a vehicle twice in a row, which gives us $c = 1$ as the constant upper limit.

In *Tyreworld*, the lowest constant upper limit is $c = 6$. If the optimal plan carries out some working step $a$ that needs to be undone later on (like jacking up the hub with the flat wheel on), then the relaxed plan for $Result(s, a)$ must include the inverse action to $a$ (like jacking down the hub). If the optimal plan carries out some final working step that does *not* need to be undone (like putting away a tool no longer needed, or jacking down the hub), then that action can be removed from $P^+$. As it turns out, $c = 6$ is the maximal number of non-final working steps that any optimal plan does in a row. ■

For the *Blocksworld-no-arm*, *Briefcaseworld*, *Fridge*, *Grid*, and *Hanoi* domains, Theorem 2 proves that there are no local minima. Thus, those domains stand a chance of having a constant upper limit to the maximal exit distance. However, in all of these domains one can easily construct instances where the maximal exit distance takes on arbitrarily high (finite) values. In *Grid*, for example, consider the instances where the robot is located on a $n \times 1$ grid (a line) without locked locations, the robot starts at the leftmost location, and shall transport a key from the rightmost location to the left end. The initial value of $h^+$ is $n + 2$ (walk over to the key, pick it up, and put it down—the *at* relation is not deleted), and the value does not get better until the robot has actually picked up the key. In *Hanoi*, the maximal exit distance grows in fact exponentially with the number of discs. For details the reader is referred to our technical report (Hoffmann 2001a).

## The Taxonomy

Our results together with the negative examples found in the previous empirical investigation (Hoffmann 2001b) prove the picture specified in Figure 2.

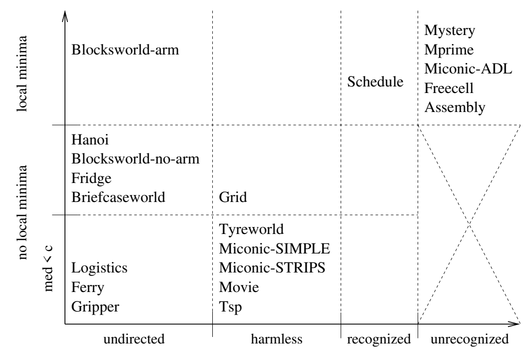


Figure 2: The extended and revised planning domain taxonomy, overviewing our results.

All of the competition domains belong to exactly those parts of the taxonomy, shown in Figure 2, where the empirical observations suggested to put them; except the *Assembly* domain. Obviously, most of the domains are to be found near the left bottom side of the taxonomy.

As discussed in the previous work, from the point of view of heuristic planners like HSP or FF which derive their heuristic function by approximating $h^+$, the left bottom side of the taxonomy is intuitively the "simple" corner: there are no or only recognized dead ends, and $h^+$ is a heuristic of high quality. In contrast, the top right corner of the taxonomy contains the "demanding" domains: there can be dead ends that are not recognized by $h^+$, and local minima. Consistently with this, the *Freecell* and *Miconic-ADL* domains constituted much more of a problem to the heuristic planners in the AIPS-2000 competition than, for example, the *Logistics* domain did. More intriguingly, we have seen that FF's search algorithm is polynomial when the heuristic has the quality corresponding to the lower most part of the taxonomy, i.e., a limited maximal exit distance. For 8 of our 20 domains, $h^+$ does in fact have this quality.

## Conclusion and Outlook

Looking at a large collection of commonly used benchmark domains, we have proven that in the majority of these domains the $h^+$ heuristic function has the qualities hypothesized by previous work (Hoffmann 2001b). As many current state-of-the-art heuristic planners work by approximating that same $h^+$ function, this suggests to interpret those planner's success as utilizing the quality of $h^+$.

In the process of proving our results, we have also determined the reasons behind the quality of $h^+$, from a more structural point of view: the reasons are mainly that in many domains all actions are (at least) invertible or need not be inverted (like when their adds are static and their deletes are irrelevant); and that the actions are respected by the relaxation, i.e., often there is simply no other way to achieve the goal than by applying them. The knowledge about these implications opens up the possibility of recognizing the relevant structural properties automatically, and thereby predicting the performance of planners like FF. This can be useful for designing hybrid systems; in particular, it might be possible to identify sub-parts of a task (some more on that below) that can efficiently be solved by heuristic planners. In fact, most of the definitions given in our theoretical core are purely syntactical. Lemma 3 gives a new sufficient criterion for recognizing planning tasks where there are no dead ends in the state space (a problem which Hoffmann and Nebel (2001) have proven to be PSPACE-hard). The only non-syntactical prerequisite of Lemma 3 is inconsistency, for which there are several good approximation techniques in the literature (Fox & Long 1998; Gerevini & Schubert 2000; Rintanen 2000). The challenge is how to determine that an action is respected by the relaxation, and thereby identify situations where $h^+$ does not yield local minima.

The main piece of work left to do is to corroborate the argumentation that planners like FF and HSP are in essence utilizing the quality of $h^+$—that is, it must be verified to which extent those planner's approximative heuristic functions really have the same quality as $h^+$. On the collection of small examples looked at in the previous empirical investigation, FF's heuristic function is similar to $h^+$ (Hoffmann 2001b). To verify this observation in general, one can look at fragments of the state spaces of larger planning tasks, and compute statistics about the distribution of local minima etc.

Another interesting future direction is to try and prove properties of $h^+$ for domain *classes* rather than for single domains in turn. Recent work by Malte Helmert (2001) has defined a hierarchy of transportation domains in the context of investigating their complexity. Judging from our results, it seems to be the case that, with Helmert's terminology, any transportation domain where there is unlimited fuel does not contain local minima under $h^+$. The main difficulty in such an investigation is the definition of the domain class: in difference to Helmert who focuses on the *semantics* of the domains, for our purposes we need a strictly *syntactical* definition: after all, $h^+$ depends directly on the syntax of the operators.

Let us finally address one of the most pressing questions opened up by this work: is this a *good* or a *bad* result for AI planning? Does it mean that we have identified widely spread structural properties that can make planning feasible, or does it mean that our benchmarks are superficial? Probably, both. The author's personal opinion is this. As for the benchmarks, it is certainly no new perception that they are more simplistic than realistic. On the other hand, it will not serve the goals of the field to construct more complex examples with the sole intention of outwitting heuristic planners. The best would be to use real-world applications, or at least as close as possible models thereof. As for the structural properties we have identified, it might be feasible to exploit these even if they occur only in sub-parts of a task, and it might well turn out that (sub-parts of) real-world tasks (like transportation issues with sufficient fuel available) exhibit that structure quite naturally.

## References


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(1–2):5–33.

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

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

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

Fox, M., and Long, D. 1998. The automatic inference of state invariants in tim. *JAIR* 9:367–421.

Frank, J.; Cheeseman, P.; and Stutz, J. 1997. When gravity fails: Local search topology. *JAIR* 7:249–281.

Gazen, B. C., and Knoblock, C. 1997. Combining the expressiveness of UCPOP with the efficiency of Graphplan. In *Proc. ECP-97*, 221–233. Toulouse, France: Springer-Verlag.

Gerevini, A., and Schubert, L. 2000. Inferring state constraints in DISCOPLAN: Some new results. In *Proc. AAAI-00*, 761–767. Austin, TX: MIT Press.

Helmert, M. 2001. On the complexity of planning in transportation domains. In *Proc. ECP-01*, 349–360. Toledo, Spain: Springer-Verlag.

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

Hoffmann, J. 2001a. Local search topology in planning benchmarks: A theoretical analysis. Technical Report 165, Albert-Ludwigs-Universität, Institut für Informatik, Freiburg, Germany. ftp://ftp.informatik.uni-freiburg.de/documents/reports/report165/report00165.ps.gz

Hoffmann, J. 2001b. Local search topology in planning benchmarks: An empirical analysis. In *Proc. IJCAI-01*, 453–458. Seattle, Washington, USA: Morgan Kaufmann.

Koehler, J., and Hoffmann, J. 2000. On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm. *JAIR* 12:338–386.

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

Rintanen, J. 2000. An iterative algorithm for synthesizing invariants. In *Proc. AAAI-00*, 806–811. Austin, TX: MIT Press.