Planning with h^+ in Theory and Practice

Christoph Betz and Malte Helmert

Albert-Ludwigs-Universität Freiburg, Institut für Informatik,
Georges-Köhler-Allee 52, 79110 Freiburg, Germany
{betzc,helmert}@informatik.uni-freiburg.de

Abstract. Many heuristic estimators for classical planning are based on the so-called *delete relaxation*, which ignores negative effects of planning operators. Ideally, such heuristics would compute the actual goal distance in the delete relaxation, i. e., the cost of an *optimal relaxed plan*, denoted by h^+. However, current delete relaxation heuristics only provide (often inadmissible) estimates to h^+ because computing the correct value is an NP-hard problem.
In this work, we consider the approach of planning with the actual h^+ heuristic from a theoretical and computational perspective. In particular, we provide *domain-dependent complexity results* that classify some standard benchmark domains into ones where h^+ can be computed efficiently and ones where computing h^+ is NP-hard. Moreover, we study *domain-dependent implementations* of h^+ which show that the h^+ heuristic provides very informative heuristic estimates compared to other state-of-the-art heuristics.

1 Introduction

Many algorithms for classical planning employ the approach of heuristic search based on *delete relaxation heuristics*. Given a state s of a planning task, such planners approximate the distance to the goal by solving a simplified planning task with initial state s where all undesirable effects of planning operators are ignored. For some planning formalisms, such as the STRIPS fragment of PDDL [1], such undesirable effects of planning operators can easily be identified syntactically. For more general planning formalisms like the ADL fragment of PDDL, linear-time compilation methods exist that can transform input tasks into the desired form [2].

Delete relaxation heuristics have been used quite successfully in the past. For example, they are a key component of the winners of the sequential satisficing tracks of the International Planning Competitions in 2000 (FF), 2004 (Fast Downward), 2006 (SGPlan) and 2008 (LAMA).

Once we commit to the idea of using heuristics based on delete relaxations, ideally we would like to use the cost of an *optimal relaxed plan* from a state s, denoted by $h^+(s)$, as the heuristic estimate for s [3]. However, computing $h^+(s)$ is an NP-equivalent problem [4], and good admissible approximations to h^+ are also provably hard to compute, even in a restricted propositional STRIPS formalism where each operator has only one precondition and only one effect (as shown later in this paper.) Therefore, common delete relaxation heuristics use approximations to h^+ which can differ from h^+ by an arbitrarily large multiplicative factor. Many such approaches are described in the planning literature:

- The max heuristic $h^{\max}$ [5] computes the makespan of an optimal *parallel plan* for the relaxed task. The $h^{\max}$ value of the state is always a lower bound to the h^+ value, i. e., provides an admissible estimate to h^+.
- The additive heuristic h^{add} [5] computes the cost of a relaxed plan under the pessimistic assumption that there are no positive interactions between goal conditions and between operator preconditions, i. e., all conditions have to be achieved completely independently. The h^{add} value of a state is always an upper bound to the h^+ value and is in general not admissible.
- The FF heuristic h^{FF} [3] computes an actual relaxed plan for the delete relaxation, using a greedy algorithm based on backchaining in so-called relaxed planning graphs. The heuristic value is then the cost of that plan. The FF heuristic is defined procedurally, and $h^{\text{FF}}(s)$ is generally ambiguous because the precise heuristic values depend on tie-breaking behaviour in the backchaining step of the algorithm. Similar to h^{add}, the FF heuristic is generally not admissible and provides an upper bound to h^+.
- The *cost-sharing heuristic* h^{cs} and *pairwise max heuristic* h^{pmax} are other delete relaxation heuristics based on relaxed planning graphs, but using different propagation rules from h^{FF} [6]. The pairwise max heuristic is inadmissible; the cost-sharing heuristic is admissible but typically less informed than $h^{\max}$.
- The *FF/additive heuristic* $h^{\text{FF/a}}$ and *set-additive heuristic* h^{sa} [7] are variants of the FF heuristic which use different methods for computing the relaxed plans that define the heuristic value. The set-additive heuristic in particular can be considered more accurate than h^{FF} because it keeps track of positive interactions between operator preconditions in a more precise way than h^{FF}. However, theoretically, the heuristics are incomparable (that is, either can be larger than the other). Neither $h^{\text{FF/a}}$ nor h^{sa} is admissible.
- The recently introduced *local Steiner tree* heuristic h^{lst} [8] is another method for computing more accurate relaxed plans than h^{FF} in order to get closer approximations to h^+. The local Steiner tree heuristic first computes a relaxed plan using the $h^{\text{FF/a}}$ method, then reduces the size of this plan by exploiting local improvement properties of Steiner trees. Like the heuristics it builds on, it is inadmissible.
- The *LAMA heuristic* h^{LAMA} [9] counts the number of *landmarks* (from a set of possible landmarks that is precomputed prior to search) for which it can prove that they need to be achieved on the way to the goal. While it is not introduced as a kind of delete relaxation heuristic in the original paper, it can be considered such because the set of landmarks it considers are guaranteed to be landmarks of the delete relaxation, so that in particular the h^{LAMA} value for the initial state does not change when the task is replaced by its delete relaxation. The LAMA heuristic is not admissible, although it is admissible in the special case where each operator achieves at most one landmark. A family of admissible landmark heuristics built on top of LAMA has recently been introduced by Karpas and Domshlak [10].
- Finally, *additive* $h^{\max}$ *heuristics* are a family of admissible approximations to h^+ based on the *action cost partitioning* paradigm introduced by Haslum et al. [11] and later generalized by Katz and Domshlak [12]. Examples of additive $h^{\max}$ heuristics include the original algorithm of Haslum et al. [11] and the additive-disjunctive heuristic graphs of Coles et al. [13].

This large number of relaxation-based planning heuristics is clear evidence that delete relaxations are a very important approach to heuristic planning. Still, quite little is known about their theoretical properties, and in particular about their *limitations*. The motivation of most of the research efforts mentioned above is to find more and more precise estimates to the h^+ heuristic. However, it is not clear how good the estimates provided by h^+ *itself* actually are. The "holy grail" of delete relaxation would be an efficient heuristic estimator which provides perfect h^+ values. But would this actually be a good heuristic, compared to approaches not based on delete relaxation, such as the context-enhanced additive heuristic [14] or abstraction heuristics [15]?

Hoffmann [16] provides a partial answer to this question by showing that certain *satisficing* (suboptimal) h^+-based planners have guaranteed polynomial runtime on many classical planning benchmarks. Additionally, Helmert and Mattmüller [17] provide a theoretical analysis that shows that h^+ generally outperforms pattern database heuristics *in the limit of large problems* on typical planning domains. In this paper, we complement these results by evaluating the quality of h^+ as an admissible heuristic for *optimal planning* on *practical benchmarks*, i. e., tasks of a size for which we can actually hope to compute a solution (unlike the in-the-limit results of Helmert and Mattmüller).

Since computing h^+ is generally NP-hard and no empirically fast algorithms are known, we perform our study by designing and evaluating *domain-dependent* algorithms for h^+ in a number of classical benchmark domains. One obvious question when designing such domain-dependent h^+ implementations is whether we can come up with sub-exponential algorithms by exploiting that we only have to deal with, e. g., LOGISTICS or BLOCKSWORLD tasks. For the domains we study in this paper, we answer this question by either describing a polynomial-time algorithm or proving that computing h^+ value remains NP-hard even when restricted to tasks of the given domain. These theoretical results, discussed in Sect. 3, form the first contribution of this paper.

In addition to this theoretical study, we provide empirical results obtained by using our domain-specific h^+ implementations as a heuristic in an A*-based planner. These results, discussed in Sect. 4, form the second contribution of this paper. Of course, runtime results obtained through domain-dependent implementations cannot be directly compared to domain-independent planners (e. g., using abstraction heuristics) in order to judge which of the approaches is generally more useful. However, they can tell us what the *theoretical limits* of relaxation-based approaches to optimal planning are, so that we can give an answer whether it is actually worth working on increasingly more sophisticated methods to compute more and more accurate approximations to h^+. To anticipate our experimental results, it appears that the answer to this question is affirmative: delete relaxations compare very favourably with the state of the art, and it definitely appears to be worth looking at their application to optimal planning more closely.

2 Background

For the theoretical results of this paper, we use the propositional STRIPS formalism [4]. (Some of the planning tasks we consider go slightly beyond STRIPS by requiring *conditional effects*, but we omit the formal details for these because they are not relevant to the abbreviated proofs we can present within the limited space of this paper.)

Definition 1. *A* planning task *is a 4-tuple* $\Pi = \langle V, O, I, G \rangle$*, where*

- V *is a finite set of* propositional state variables *(also called* propositions *or* facts*),*
- O *is a finite set of* operators*, each with associated* preconditions *pre*$(o) \subseteq V$*,* add effects *add*$(o) \subseteq V$ *and* delete effects *del*$(o) \subseteq V$*,*
- $I \subseteq V$ *is the* initial state*, and*
- $G \subseteq V$ *is the set of* goals*.*

A *state* is a subset of facts, $s \subseteq V$, representing the propositions which are currently true. *Applying* an operator o in s results in state $(s \setminus \mathit{del}(o)) \cup \mathit{add}(o)$, which we denote as $s[o]$. The notation is only defined if o is *applicable* in s, i. e., if $\mathit{pre}(o) \subseteq s$. Applying a sequence $o_1, \ldots, o_n$ of operators to a state is defined inductively as $s[\epsilon] := s$ and $s[o_1, \ldots, o_{n+1}] := (s[o_1, \ldots, o_n])[o_{n+1}]$. A plan for a state s (*s-plan*, or *plan* when s is clear from context) is an operator sequence π such that $s[\pi]$ is defined and satisfies all goals (i. e., $G \subseteq s[\pi]$). The objective of *optimal planning* is to find an I-plan of minimal length (called an *optimal I-plan*) or prove that no plan exists.

Heuristic functions or *heuristics* are a key ingredient of heuristic search planners. A heuristic is a function $h : 2^V \to \mathbb{N}_0 \cup \{\infty\}$ with the intuition that $h(s)$ estimates the length of an s-plan. The *perfect heuristic* h^* maps each state to the length of an optimal s-plan (infinite if no s-plan exists). A heuristic h is *admissible* if $h(s) \leq h^*(s)$ for all states s. All common heuristic search algorithms for optimal planning require admissible heuristics. If $h(s) \geq h'(s)$ for all states s, we say that h *dominates* h'.

Relaxation heuristics estimate the distance to the goal by considering a *relaxed task* Π^+ derived from the actual planning task Π by ignoring all delete effects of operators, i. e., replacing each operator o by a new operator o^+ with the same preconditions and add effects as o and $\mathit{del}(o^+) = \emptyset$. The h^+ *heuristic* [3] uses the length of an optimal s-plan in Π^+ as the heuristic estimate $h^+(s)$ for a state s of the original task Π.

3 Theory: Complexity of Computing h^+

Computing h^+ estimates for states of a planning task is an NP-equivalent problem [4]. It is due to this computational complexity that h^+ has not previously been used in an actual planning system designed to solve planning tasks of interesting size. However, far from being optimal, all approximations to h^+ discussed in the introduction can actually be arbitrarily far off from the correct h^+ values, i. e., $h(s)/h^+(s)$ can be arbitrarily large for the inadmissible heuristics h discussed there, and $h^+(s)/h(s)$ can be arbitrarily large for the admissible ones. We now prove that there is a theoretical reason for this.

Theorem 2. *If* P $\neq$ NP*, then there exists no constant $c > 0$ and no polynomial-time algorithm for computing an admissible heuristic function h such that for all states s, $h(s) \geq c \cdot h^+(s)$. This is true even when only allowing planning tasks where each operator has only a single precondition and only a single add effect.*

Proof sketch: We present an approximation-preserving reduction (see the textbook by Ausiello et al. [18]) from MINIMUM SET COVER to planning for delete relaxations. Since MINIMUM SET COVER has no constant-factor approximations unless P = NP [18, problem SP4], the claim follows. Given a MINIMUM SET COVER instance with set

S and subsets $C_1, \ldots, C_m \subseteq S$, the reduction generates operators $o_i^1, o_i^2, \ldots, o_i^N$ for each subset C_i such that all these operators need to be applied (in sequence) in order to achieve a fact a_i that marks that C_i has been selected. From a_i, facts corresponding to the elements of C_i can then be directly achieved using operators which have precondition a_i and add one element of C_i at a time. To satisfy the goal, which consists of all facts corresponding to elements of S, we must select enough subsets C_i to cover S completely. By choosing N appropriately (e. g., $N = |S|$), we can ensure that the overall plan length is dominated by the number of subsets chosen, and hence short relaxed plans correspond to small set covers. ∎

Theorem 2 shows that we cannot hope to find a polynomial algorithm that is guaranteed to find good approximations to h^+. However, since theoretical results of this kind tend to rely on somewhat pathological problem instances, this does not mean that computing or approximating h^+ is necessarily difficult for *practically interesting planning tasks*. Hence, to get a handle on the complexity of computing h^+ in more typical cases, we now investigate the behaviour of h^+ in *specific* planning domains used as benchmarks in the planning community, specifically the domains considered by Helmert and Mattmüller in their theoretical study of admissible planning heuristics [17].

It turns out that, at least for these domains, the situation is not quite as bleak. In all cases, we can compute constant-factor approximations to h^+ in polynomial time, and in some cases we even have polynomial algorithms for the perfect h^+ value, despite the fact that in most of these domains (all except GRIPPER and SCHEDULE [19]), computing the actual goal distance h^* is NP-hard.

For space reasons, we refer to the literature for formal definitions of these common planning benchmarks [19] and only provide very brief proof sketches. An extensive discussion of these results, including full proofs, can be found in Betz's thesis [20].

Theorem 3. *There exists a polynomial-time algorithm for computing $h^+(s)$ for arbitrary reachable states s of* BLOCKSWORLD *tasks.*
Proof sketch: The subgoal ordering issues that make optimal BLOCKSWORLD planning hard in general [21] do not exist in the delete relaxation where simple greedy criteria are sufficient to decide which blocks to pick up and, after all pick-ups have been performed, where to drop them. See Betz's thesis for details [20, Corollary 6.1]. ∎

Theorem 4. *There exists a polynomial-time algorithm for computing $h^+(s)$ for arbitrary reachable states s of* GRIPPER *tasks.*
Proof sketch: Due to symmetries in GRIPPER tasks, a closed formula for h^+ can be given. This formula can be evaluated in linear time [20, Theorem 5.1]. ∎

Theorem 5. *Computing $h^+(s)$ for arbitrary reachable states s of* LOGISTICS *tasks is NP-hard, but polynomial-time constant-factor approximations exist.*
Proof sketch: Hardness is proved by a reduction from SET COVER. There is one truck corresponding to each candidate subset, which is loaded with one package for each element of that subset. The instance is then constructed in such a way that a subset of trucks need to visit a special location, called the *Ω-location*, and the overall quality of a relaxed plan is determined by how many trucks visit the Ω-location. In optimal relaxed plans this subset corresponds to an optimal set cover [20, Theorem 8.3]. For the constant-factor approximation result, we refer to Betz's thesis [20, Theorem 8.5]. ∎

We remark that polynomial h^+-algorithms for LOGISTICS exist if we only consider valid *initial states*, where vehicles are required to be empty [20, Theorem 8.2], and also when there is only one truck per city and only one airplane [20, Theorem 8.1].

Theorem 6. *There exists a polynomial-time algorithm for computing $h^+(s)$ for arbitrary reachable states s of* MICONIC-STRIPS *tasks.*
Proof sketch: This follows directly from the previous remark due to the similarity of MICONIC-STRIPS to LOGISTICS with only one truck [20, Theorem 3.1]. ■

Theorem 7. *Computing $h^+(s)$ for arbitrary reachable states s of* MICONIC-SIMPLEADL *tasks is NP-hard, but polynomial-time constant-factor approximations exist.*
Proof sketch: In MICONIC-SIMPLEADL, computing h^+ is closely related to computing h^*, and the known results for h^* [22] carry over [20, Theorem 3.2]. ■

Theorem 8. *Computing $h^+(s)$ for arbitrary reachable states s of* SATELLITE *tasks is NP-hard, but polynomial-time constant-factor approximations exist.*
Proof sketch: The proof [20, Theorem 7.1] is again based on a reduction from SET COVER and uses similar ideas to the proof that establishes NP-hardness for h^* [22]. ■

Theorem 9. *There exists a polynomial-time algorithm for computing $h^+(s)$ for arbitrary reachable states s of* SCHEDULE *tasks.*
Proof sketch: A simple algorithm that achieves the goals one object ("part") at a time is sufficient [20, Theorem 4.1]. ■

4 Practice: Using h^+ Inside an Optimal Planner

As noted in the introduction, delete relaxation heuristics are state of the art for satisficing planning. For optimal planning, however, the literature suggests that the admissible representatives of the family, $h^{\max}$ and h^{cs}, lag behind other approaches such as abstraction. For example, merge-and-shrink abstractions ($h^{\text{m\&s}}$ in the following) clearly outperform $h^{\max}$ [15], and h^{cs} is empirically even worse than $h^{\max}$ [6]. Does this indicate that delete relaxation heuristics are generally not useful for optimal planning, or is this a specific weakness of $h^{\max}$ and h^{cs}? To answer that question, we have added domain-specific implementations of the h^+ heuristic to a state-of-the-art A*-based optimal planner [15] and empirically compared it to $h^{\max}$, to see how far current admissible relaxation heuristics are from what is possible, and to $h^{\text{m\&s}}$, to see if relaxation heuristics may be competitive with the state of the art in optimal planning.

Experiments were conducted under the usual planning competition settings. Table 1 shows the results. Note that while our h^+ *implementations* are domain-dependent, the *estimates* themselves are fully domain-independent, and hence comparisons of heuristic quality (e. g., number of A* state expansions) are meaningful. We compare on all domains considered in the previous section except for those not supported by the underlying planner, MICONIC-SIMPLEADL and SCHEDULE. Note that this includes the LOGISTICS and SATELLITE domains where computing h^+ is NP-hard; in these cases, each state evaluation in our implementation can require exponential time. Table 1 indicates that the time per state expansion is indeed very high for SATELLITE, but h^+

Table 1. Experimental comparison of h^+, $h^{\text{m\&s}}$ and $h^{\max}$. Parameters for $h^{\text{m\&s}}$ are hand-tuned per domain. We report heuristic values for the initial state (h), number of expanded states (Exp.), and runtime in seconds (Time) for the largest tasks solved in each domain. Dashes indicate running out of time (30 minutes) or memory (2 GB). Best results for each task are highlighted in bold.

		h^+			$h^{\text{m\&s}}$			$h^{\max}$		
Inst.	h^*	h	Exp.	Time	h	Exp.	Time	h	Exp.	Time
BLOCKSWORLD ($h^{\text{m\&s}}$: one abstraction of size 50000)										
#9-0	30	**16**	**13215**	**0.65**	**16**	971774	191.12	9	3840589	85.00
#9-1	28	**16**	**360**	**0.02**	**16**	60311	69.25	10	1200345	32.06
#9-2	26	**17**	**594**	**0.04**	16	54583	90.12	9	1211463	32.15
#10-0	34	**18**	**241489**	**15.42**	**18**	19143580	367.82	9	—	—
#10-1	32	**19**	**29144**	**1.82**	16	12886413	316.28	8	—	—
#10-2	34	**19**	**83007**	**5.68**	18	—	—	10	—	—
#11-0	32	19	**63891**	**4.35**	**21**	7291064	199.01	8	—	—
#11-1	30	**21**	**59340**	**4.64**	17	—	—	4	—	—
#11-2	34	**19**	**53642**	**3.39**	**19**	—	—	9	—	—
#12-0	34	**22**	**58124**	**4.54**	21	—	—	10	—	—
#12-1	34	**22**	**6284**	**0.48**	21	—	—	11	—	—
#13-1	44	**25**	**9990123**	**1078.59**	23	—	—	12	—	—
#14-0	38	**25**	**100499**	**10.64**	19	—	—	10	—	—
#14-1	36	**27**	**160352**	**19.99**	20	—	—	6	—	—
#15-0	40	**28**	**3540691**	**420.91**	18	—	—	7	—	—
GRIPPER ($h^{\text{m\&s}}$: one abstraction of size 5000)										
#1	11	9	82	**0.00**	**11**	**12**	**0.00**	2	208	**0.00**
#2	17	13	1249	**0.00**	**15**	**975**	0.10	2	1760	0.01
#3	23	**17**	**10304**	**0.06**	11	11506	0.34	2	11616	0.08
#4	29	**21**	**65687**	**0.44**	13	68380	1.04	2	68468	0.56
#5	35	**25**	**371726**	**2.86**	14	376510	3.59	2	376496	3.51
#6	41	**29**	**1974285**	17.79	16	1982018	**16.19**	2	1982016	21.57
#7	47	**33**	**10080252**	97.60	18	10091970	**79.83**	2	10091968	119.64
LOGISTICS-1998 ($h^{\text{m\&s}}$: one abstraction of size 200000)										
#1	26	24	**8623**	**2.91**	**25**	375885	67.64	6	—	—
#5	22	**22**	**30**	**0.03**	20	527498	99.94	4	—	—
#17	42	**42**	**67**	**0.62**	—	—	—	6	—	—
#31	13	12	68	**0.02**	**13**	**14**	7.22	4	32282	0.57
#32	20	18	116	**0.02**	**20**	**21**	1.97	6	81156	1.00
#33	27	25	88629	**21.80**	**27**	**28992**	81.01	4	—	—
#35	30	**29**	**1682**	**2.65**	22	—	—	5	—	—

		h^+			$h^{\text{m\&s}}$			$h^{\max}$		
Inst.	h^*	h	Exp.	Time	h	Exp.	Time	h	Exp.	Time
LOGISTICS-2000 ($h^{\text{m\&s}}$: one abstraction of size 200000)										
#8-0	31	29	3269	**0.32**	**31**	**32**	26.75	6	—	—
#8-1	44	41	43665	**3.87**	**44**	**45**	28.17	6	—	—
#9-0	36	33	14090	**1.62**	**36**	**37**	37.58	6	—	—
#9-1	30	29	707	**0.10**	**30**	**31**	37.49	6	—	—
#10-0	45	41	**193846**	**26.39**	**45**	196342	79.12	6	—	—
#10-1	42	39	**165006**	**24.88**	**42**	518215	86.22	6	—	—
#11-0	48	45	156585	**28.81**	**48**	**12822**	87.99	6	—	—
#11-1	60	55	5649882	775.53	**59**	**2608870**	**187.85**	6	—	—
#12-0	42	39	**116555**	**22.25**	**42**	272878	117.98	6	—	—
#12-1	68	63	—	—	**68**	**828540**	**137.67**	6	—	—
SATELLITE ($h^{\text{m\&s}}$: three abstractions of size 10000)										
#1	9	8	**10**	**0.00**	**9**	**10**	**0.00**	3	59	**0.00**
#2	13	12	**14**	0.02	**13**	**14**	0.09	3	940	**0.00**
#3	11	10	21	**0.07**	**11**	**12**	4.01	3	6822	0.11
#4	17	**17**	**26**	**0.15**	**17**	237	7.57	3	180815	3.37
#5	15	**14**	**34**	**5.02**	12	38598	44.66	3	—	—
#6	20	**18**	**526**	**16.23**	16	375938	48.73	3	10751017	371.43
#7	21	**20**	**812**	**250.37**	15	—	—	3	—	—
#9	27	**26**	**264**	**1350.72**	17	—	—	3	—	—
#10	29	**29**	**40**	**401.41**	16	—	—	3	—	—
MICONIC-STRIPS ($h^{\text{m\&s}}$: one abstraction of size 200000)										
#28-4	92	**92**	**123**	**0.05**	54	—	—	3	—	—
#29-0	94	**94**	**126**	**0.06**	54	—	—	3	—	—
#29-1	91	**91**	**184**	**0.08**	53	—	—	3	—	—
#29-2	95	**95**	**155**	**0.06**	55	—	—	3	—	—
#29-3	97	**96**	**178**	**0.07**	56	—	—	3	—	—
#29-4	99	**99**	**141**	**0.05**	55	—	—	3	—	—
#30-0	95	**95**	**138**	**0.06**	55	—	—	3	—	—
#30-1	98	**98**	**150**	**0.06**	55	—	—	3	—	—
#30-2	97	**96**	**130**	**0.04**	55	—	—	3	—	—
#30-4	99	**99**	**124**	**0.05**	53	—	—	3	—	—

still scales much further than the other approaches due to the accuracy of the heuristic. Aggregating results over all domains, h^+ convincingly outperforms the other heuristics considered, including the state-of-the-art $h^{\text{m\&s}}$. This suggests that the comparatively bad results obtained with earlier delete relaxation heuristics are mostly due to their inability to accurately approximate h^+ rather than a general weakness of delete relaxations.

5 Conclusion

Starting from the observation that many current planning heuristics are based on delete relaxations, we have taken a deeper look at the optimal delete relaxation heuristic, h^+, which all these heuristics strive to approximate. Theoretically, we have seen that h^+ is in general not just hard to compute (as proved already by Bylander), but also hard to approximate. However, these worst-case results do not carry over to most planning domains, for which we could show much better theoretical results – including polynomial-time algorithms for h^+ in four of the seven benchmark results considered.

Experimentally, we have shown that h^+ is very informative across a range of planning domains, improving on the state of the art in domain-independent optimal planning. Hence, it appears worth investigating practically efficient general implementations of h^+, or alternatively better admissible approximations, more closely. In our opinion, despite the multitude of existing approaches, there is still considerable scope for research on delete relaxation heuristics, in particular admissible ones. Our results presented here can serve as a useful methodological basis for such future work by allowing, for the first time, direct comparisons of practical relaxation heuristics to h^+.

Acknowledgments

This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems". See `http://www.avacs.org/` for more information.

References

1. Fox, M., Long, D.: PDDL2.1: An extension to PDDL for expressing temporal planning domains. JAIR **20** (2003) 61–124
2. Gazen, B.C., Knoblock, C.A.: Combining the expressivity of UCPOP with the efficiency of Graphplan. In: Proc. ECP-97. (1997) 221–233
3. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. JAIR **14** (2001) 253–302
4. Bylander, T.: The computational complexity of propositional STRIPS planning. AIJ **69**(1–2) (1994) 165–204
5. Bonet, B., Geffner, H.: Planning as heuristic search. AIJ **129**(1) (2001) 5–33
6. Mirkis, V., Domshlak, C.: Cost-sharing approximations for h^+. In: Proc. ICAPS 2007. (2007) 240–247
7. Keyder, E., Geffner, H.: Heuristics for planning with action costs revisited. In: Proc. ECAI 2008. (2008) 588–592
8. Keyder, E., Geffner, H.: Trees of shortest paths vs. Steiner trees: Understanding and improving delete relaxation heuristics. In: Proc. IJCAI 2009. (2009) To appear.
9. Richter, S., Helmert, M., Westphal, M.: Landmarks revisited. In: Proc. AAAI 2008. (2008) 975–982
10. Karpas, E., Domshlak, C.: Cost-optimal planning with landmarks. In: Proc. IJCAI 2009. (2009) To appear.
11. Haslum, P., Bonet, B., Geffner, H.: New admissible heuristics for domain-independent planning. In: Proc. AAAI 2005. (2005) 1163–1168
12. Katz, M., Domshlak, C.: Optimal additive composition of abstraction-based admissible heuristics. In: Proc. ICAPS 2008. (2008) 174–181
13. Coles, A., Fox, M., Long, D., Smith, A.: Additive-disjunctive heuristics for optimal planning. In: Proc. ICAPS 2008. (2008) 44–51
14. Helmert, M., Geffner, H.: Unifying the causal graph and additive heuristics. In: Proc. ICAPS 2008. (2008) 140–147
15. Helmert, M., Haslum, P., Hoffmann, J.: Flexible abstraction heuristics for optimal sequential planning. In: Proc. ICAPS 2007. (2007) 176–183
16. Hoffmann, J.: Where 'ignoring delete lists' works: Local search topology in planning benchmarks. JAIR **24** (2005) 685–758
17. Helmert, M., Mattmüller, R.: Accuracy of admissible heuristic functions in selected planning domains. In: Proc. AAAI 2008. (2008) 938–943
18. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation. Springer-Verlag (1999)
19. Helmert, M.: Understanding Planning Tasks – Domain Complexity and Heuristic Decomposition. Volume 4929 of LNAI. Springer-Verlag (2008)
20. Betz, C.: Komplexität und Berechnung der h^+-Heuristik. Diplomarbeit, Albert-Ludwigs-Universität Freiburg (2009)
21. Gupta, N., Nau, D.S.: On the complexity of blocks-world planning. AIJ **56**(2–3) (1992) 223–254
22. Helmert, M., Mattmüller, R., Röger, G.: Approximation properties of planning benchmarks. In: Proc. ECAI 2006. (2006) 585–589