Useless Actions are Useful

Martin Wehrle and **Sebastian Kupferschmid** and **Andreas Podelski**
University of Freiburg
{mwehrle, kupfersc, podelski}@informatik.uni-freiburg.de

Abstract

Planning as heuristic search is a powerful approach to solving domain independent planning problems. In recent years, various successful heuristics and planners like FF, LPG, FAST DOWNWARD or SGPLAN have been proposed in this context. However, as heuristics only *estimate* the distance to goal states, a general problem of heuristic search is the existence of plateaus in the search space topology which can cause the search process to degenerate. Additional techniques like helpful actions or preferred operators that evaluate the "usefulness" of actions are often successful strategies to support the search in such situations.

In this paper, we introduce a general method to evaluate the usefulness of actions. We propose a technique to enhance heuristic search by identifying "useless" actions that are not needed to find optimal plans. In contrast to helpful actions or preferred operators that are specific to the FF and Causal Graph heuristic, respectively, our method can be combined with arbitrary heuristics. We show that this technique often yields significant performance improvements.

Introduction

Planning as heuristic search is a well-established technique to solving domain independent planning problems and has found its way into state-of-the-art planners such as FF (Hoffmann and Nebel 2001), LPG (Gerevini, Saetti, and Serina 2003), FAST DOWNWARD (Helmert 2006) or SGPLAN (Chen, Wah, and Hsu 2006). However, a general problem of heuristic search are plateaus in the search space topology. In such situations, it is not clear which state is best to be expanded next, and hence, the overall search process can be severely worsened. One possibility to tackle this problem is to additionally evaluate *actions* (not just states) which provides additional information. In this context, Hoffmann and Nebel (2001) proposed the *helpful actions* approach. Helpful actions are specified for h^{FF} and extracted during the computation of the heuristic values. Helmert proposed *preferred operators* which is an adaptation of helpful actions to the Causal Graph heuristic (Helmert 2006). Both concepts have shown its potential by providing additional information to escape from plateaus during search. However, these techniques are specialized to the corresponding heuristic.

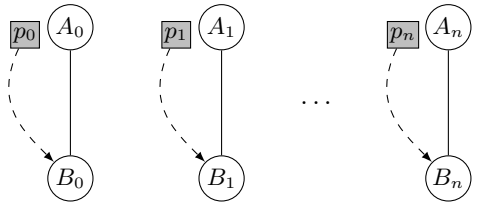

Figure 1: A simple transportation problem

In this paper, we propose a general technique to evaluate actions during heuristic search. In contrast to helpful actions or preferred operators, we identify *useless* actions that are not needed to find optimal plans starting with the current state. Our method is not bound to a special heuristic, but can be applied to arbitrary heuristics. We will first characterize an exact notion of uselessness of actions, which we will then approximate with the given heuristic.

Our experiments show a significant improvement on a number of commonly known heuristics and benchmarks from the International Planning Competitions. In the context of the FF heuristic and a generalization of the Causal Graph and additive heuristics (Helmert and Geffner 2008), we are competitive with helpful actions and preferred operators. Moreover, the concepts of uselessness and helpfulness can be successfully combined, yielding a significantly better classification of useful and useless actions than before. Finally, we show that our technique also improves planning with h^{add} and h^{max} (Bonet and Geffner 2001).

We will next give an example that greatly oversimplifies the issues at hand but gives an intuition about useless actions and their potential usefulness.

Example. Consider the simple transportation task shown in Fig. 1. It consists of the locations $A_0, \ldots, A_n$ and $B_0, \ldots, B_n$, where the task is to transport a package p_i from A_i to B_i with truck t_i for all $i \in \{0, \ldots, n\}$. A truck t_i can drive from A_i to B_i, and vice versa. Initially, all packages and trucks are in the A locations, and the goal state is to have all packages in the B locations. For simplicity, assume that all packages are already loaded on the corresponding truck, and the only actions that have to be performed to reach the goal state is to drive all trucks from A_i to B_i.

Suppose that we use the h^{max} heuristic to find a solution to that problem. It turns out that, for this problem, h^{max} is a rather uninformed heuristic, because it cannot distinguish states that are nearer to the goal state from others. If there is at least one package that has not been delivered yet, then the heuristic value for that state is 1. We may characterize the state space topology induced by h^{max} as follows. There is one single plateau, i. e., for all of the 2^{n+1} reachable states but for the goal state the heuristic value is 1. This means that the guidance based on this heuristic is very poor here. In fact, for every heuristic there are problems where it runs into similar situations.

For the simple transportation problem, it is trivial to see that driving from a B location to an A location does not make sense, i. e., it is a useless action. Without steps corresponding to such actions, the search stops after $n + 1$ steps and returns a shortest possible plan. Otherwise, the search degenerates to uninformed search.

Figure 2 provides a foretaste of the usefulness of useless actions. It shows the search space of the example problem that is explored by a best-first search with h^{max}, on top without taking useless actions into account and below when useless actions are less preferred. As argued above, avoiding such actions has a significant impact on the size of the explored search space.

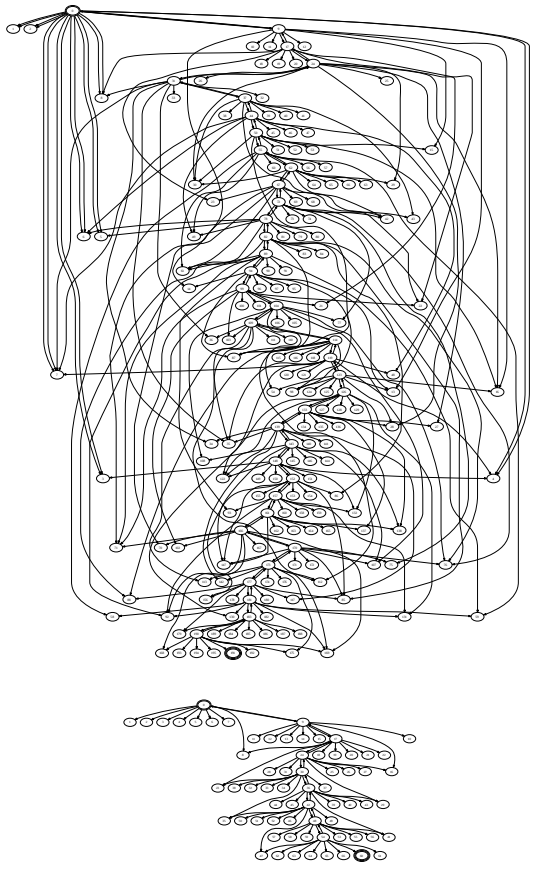

Figure 2: The effect of our method when applied to the simple transportation problem with $n = 8$

The paper is organized as follows. In the next section, we give the basic notation. Then we present the notion of useless actions and evaluate its performance. This is followed by a discussion on related work. Finally, we conclude the paper and give a short outlook on future work.

Notation

We consider planning in a setting where the states of the world are represented in terms of a set P of Boolean state variables that take the value *true* or *false*. Each *state* is a valuation on P, i. e., an assignment $s : P \to \{true, false\}$. A literal is a formula of the form a or $\neg a$ for all $a \in P$. The set of literals is denoted by $\mathcal{L} = P \cup \{\neg a \mid a \in P\}$. We use *operators* for expressing how the state of the world can be changed. An *operator* is a tuple $o = \langle p, e\rangle$ where $p \subset \mathcal{L}$ is a set of literals, the *precondition*, and $e \subset \mathcal{L}$ is a set of literals, the *effect*. The operator is *applicable* in a state s if $s \models p$ (where we identify a set of literals with their conjunction). In this case, we define $\text{app}_o(s) = s'$ as the unique state that is obtained from s by setting the effect literals of o to *true* and retaining the truth values of the state variables not occurring in the effect. For sequences of operators $\sigma = o_1, \ldots, o_n$ we define $\text{app}_\sigma(s)$ as $\text{app}_{o_n}(\ldots \text{app}_{o_2}(\text{app}_{o_1}(s)) \ldots)$.

Let $\Pi = \langle P, I, O, G\rangle$ be a *planning instance*, consisting of a set P of state variables, a state I on P (the initial state), a set O of operators on P, and a formula G on P (the goal formula). A (sequential) plan for Π is a sequence $\sigma = o_1, \ldots, o_n$ of operators from O such that $\text{app}_\sigma(I) \models G$, i. e., applying the operators in the given order starting in the initial state is defined (the precondition of every operator is true when the operator is applied) and produces a state that satisfies the goal formula.

The Notion of Useless Actions

In this section we will first define the theoretical concept of useless actions and then its practical counterpart, the relatively useless actions. As actions are represented as operators, we will use the terms *useless action* and *useless operator* synonymously.

Useless Actions

Let us first give the definition of useless actions which precisely captures the intuition. An action is useless if it is not needed to reach a goal state in an optimal plan. This is formally stated in the next definition.

Definition 1 (Useless Action). Let $\Pi = \langle P, I, O, G\rangle$ be a planning instance, s a state, $o \in O$. The operator o is *useless in* s if and only if no optimal plan from s starts with o.

Obviously, this definition is not practical since recognizing useless actions is as hard as planning itself. Hence, we will investigate other ways to express this property. We will arrive at a characterization that lends itself to the definition of a practical version of useless actions, a definition that we will give in the next section.

We use $d(s)$ to denote the length of a shortest sequential plan from a state s to a goal state. When we want to stress

that d also is a function on the planning instance Π, we will write $d(\Pi,\ s)$.

By Definition 1, an operator o is useless in a state s if and only if the real goal distance d does not decrease by one, i. e. an action is useless iff $d(s) \leq d(s')$, where $s' = \text{app}_o(s)$. To see this, recall that $d(s) \leq d(s') + 1$ for every action. If an optimal plan starts from s with o, then $d(s) = d(s') + 1$. Otherwise we have $d(s) < d(s') + 1$, and since the distance values are all integers, this is equivalent to $d(s) \leq d(s')$.

We will use the inequality $d(s) \leq d(s')$ in connection with the idea of *removing* operators from Π.

Definition 2 (Reduced planning instance). For a given planning instance $\Pi = \langle P,\ I,\ O,\ G\rangle$ and $o \in O$, the *reduced planning instance* Π_o is defined as $\langle P,\ I,\ O\backslash\{o\},\ G\rangle$.

We will characterize the notion of useless actions with the help of reduced planning instances in the following proposition.

Proposition 1. *Let $\Pi = \langle P,\ I,\ O,\ G\rangle$ be a planning instance, s a state, $o \in O$ applicable in s, $s' = app_o(s)$. If $d(\Pi_o,\ s) \leq d(\Pi,\ s')$, then o is useless in s.*

Proof. If $d(\Pi_o,\ s) \leq d(\Pi,\ s')$, then $d(\Pi,\ s) \leq d(\Pi,\ s')$ because $d(\Pi,\ s) \leq d(\Pi_o,\ s)$ for all s and o. With the argumentation above (i. e., $d(s) \leq d(s')$ if and only if no optimal plan from s starts with o), it follows that o is useless in s. □

This characterization can be read as saying that an action o is useless in s if a goal state is still reachable from s with the same optimal plan when o is *removed* from the planning instance.

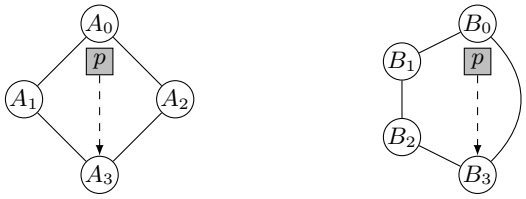

Figure 3: Example transportation problems

The above characterization of useless actions has its own intricacies. Observe that, for an action o to be useless in a state s, it is *not* enough to require $d(\Pi_o,\ s) \leq d(\Pi,\ s)$. See the left example in Fig. 3. In this transportation problem, the task is to transport an object p from A_0 to A_3 via A_1 *or* A_2. The operators `drive-A0-A1` and `drive-A0-A2` would both (wrongly) be classified as useless in the initial state.

Furthermore, it does *not* suffice to require $d(\Pi_o,\ s) = d(\Pi,\ s')$. See the right example in Fig. 3. The operator `drive-B0-B1` would not be classified as useless in the initial state although it is, i. e., it is not part of an optimal plan from B_0 to B_3.

The Practical Counterpart

Obviously, the detection of useless actions is PSPACE-hard and therefore too expensive in practice. Proposition 1 motivates a direct approximation, namely by simply replacing the goal distance $d(s)$ with a heuristic function $h(s)$ that estimates the distance from a state s to a nearest goal state. When we want to stress that h also is a function on the planning instance Π, we will write $h(\Pi,\ s)$.

Definition 3 (Relatively Useless Action). For a planning instance $\Pi = \langle P,\ I,\ O,\ G\rangle$, $o \in O$, a state s and a heuristic function $h(\Pi,\ s)$, the operator o is *relatively useless (for h)* in s iff $h(\Pi_o,\ s) \leq h(\Pi,\ s')$. We denote a state as relatively useless if it is reached by applying an operator that is relatively useless.

Note that also for heuristics h, the requirement $h(\Pi_o,\ s) \leq h(\Pi,\ s')$ is usually stronger than $h(\Pi,\ s) \leq h(\Pi,\ s')$, e. g. for abstraction heuristics (Helmert, Haslum, and Hoffmann 2007). This is because whenever $h(\Pi_o,\ s) \leq h(\Pi,\ s')$, then $h(\Pi,\ s) \leq h(\Pi,\ s')$, since $h(\Pi,\ s) \leq h(\Pi_o, s)$ for all states s and operators o. It is not difficult to construct an example to see that the other direction does not hold.

For abstraction heuristics, actions that lead to states with lower heuristic values are *never* relatively useless. More precisely, for a state s, operator o and $s' = \text{app}_o(s)$, if $h(s') < h(s)$, then o is not relatively useless in s. This is simply because if o is relatively useless in s, i. e. $h(\Pi_o,\ s) \leq h(\Pi,\ s')$, it follows that $h(\Pi,\ s) \leq h(\Pi,\ s')$.

Obviously, the quality of this approximation strongly depends on h's precision. A very uninformed heuristic that e. g. constantly returns zero recognizes every action as relatively useless. However, the more informative the heuristic function is, the more precise is the approximation. Intuitively, taking a relatively useless operator o does not seem to guide the search towards a goal state as the *stricter* distance estimate in Π_o does not increase.

Evaluation

We have implemented our method into FAST DOWNWARD and evaluated it on a number of commonly known heuristics and benchmarks from the recent International Planning Competitions (IPC).

Integration into FAST DOWNWARD

FAST DOWNWARD is a recent state-of-the-art planner, in particular winning the IPC 2004 in the track for classical planning. FAST DOWNWARD uses the SAS$^+$ representation of planning instances where state variables are not Boolean, but have an integer domain (Bäckström and Nebel 1995). In a best-first search setting, FAST DOWNWARD maintains two open queues, one for states that result from applying preferred operators, the other one for all states. During search, states are taken alternately from these queues.

We will next describe how we integrated the technique of useless actions into FAST DOWNWARD. We maintain three different open queues *helpfulQueue*, *unknownQueue* and *uselessQueue*, containing states as indicated by their names (where states that are classified as helpful *and* relatively useless are maintained in *uselessQueue*). More precisely, for each state s that is expanded with operator o, we check if o is relatively useless in s for the heuristic that is

also used for the search. If this is the case, the corresponding successor state $s' = \text{app}_o(s)$ is pushed into the *uselessQueue*. The modified planning algorithm is based on how the different open queues are selected. The straight forward algorithm would be to take states from *uselessQueue* only if the other two queues are empty. However, as the approximation of useless may fail (depending on the quality of h), we also take states from *uselessQueue* with a small probability $\vartheta = 0.1$. We will come back to this point in the discussion section. The modified algorithm for selecting the open queues is shown in Fig. 4. Line 11 ensures that an empty queue is never returned if there exists a non empty one. The function selectNonEmptyQueue() returns a non empty last recently used queue if there exists one.

```
1  function selectOpenQueue():
2      choose p ∈ [0.0, 1.0] uniformly distributed
3      if p < ϑ and uselessQueue not empty:
4          return uselessQueue
5      if preferred and helpfulQueue not empty:
6          preferred = false
7          return helpfulQueue
8      elif unknownQueue not empty:
9          preferred = true
10         return unknownQueue
11     return selectNonEmptyQueue()
```

Figure 4: Selection of open queues

Additionally, to reduce misclassifications of relatively useless actions, we implemented our concept in a stronger way than described in the last section. When computing Π_o, we additionally remove all operators from Π that require a variable to have a value that is set by o.

Experimental Results

We evaluated our method for the h^{FF}, h^{cea}, h^{add} and h^{max} heuristics. The h^{cea} heuristic is a generalization of the h^{add} and Causal Graph heuristics (Helmert and Geffner 2008).

In all configurations, the open queues are selected according to the algorithm in Fig. 4. The experimental results in Table 1, 2, 3 and 4 were obtained on a 2.66 GHz Intel Xeon computer with memory up to 3 GB and a Linux operating system. For each configuration, we report the plan length, the number of expanded states and the search time in seconds. We set a time bound of 10 minutes per run and averaged the data (i. e. plan length, states, time) over 10 runs. A dash indicates that all runs failed (out of time or out of memory). If some, but not all runs failed, we indicate the number n of runs that failed with a superscript n. We report detailed results for the domains PATHWAYS, TPP, DEPOT, ASSEMBLY and PIPESWORLD-NOTANKAGE. These domains are hard for FAST DOWNWARD when considering the total number of instances that could not be solved without our technique. For each domain, we give the results for the ranges of instances up to the largest instance that could be solved *without* and *with* our technique. If these ranges are disjoint, they are separated by a horizontal line. Missing instances in these ranges could not be solved by any configuration within our time and memory bounds.

Tables 1, 2 and 3 show the results for the h^{FF}, h^{cea} and h^{add} heuristics. With our approach, a significant performance gain is obtained in all settings. When h^{FF} or h^{cea} is applied, we are competitive with helpful actions or preferred operators. In most cases, we even explored significantly less states. Moreover, the notions of helpfulness and uselessness can be efficiently combined, yielding a significantly better classification of useful and useless actions than before. Exemplary, this shows up very well in the PATHWAYS domain. Without considering useless actions, the largest instance that could be solved with h^{FF} without (with) helpful actions is PATHWAYS 10 (18). For h^{cea}, the largest solved instance without (with) preferred operators is PATHWAYS 10 (16). The combination of helpful and useless actions yields a planner that could even solve the largest instance.

Moreover, we did experiments for the h^{max} heuristic. Although the improvements are less significant than with h^{add}, we still got considerable performance gains. We can summarize the results as follows: for instances that could be solved by h^{max}, the number of expanded states decreases in 32 cases when our concept is applied, whereas in 9 instances more states are expanded. Furthermore, we could solve more instances than before.

To get a more detailed overview of the performance of each configuration, we report the total number of unsolved instances in Table 4, where an instance is counted as unsolved if all runs failed. For each heuristic h, we compare *normal* heuristic search, (i. e. h with no search enhancements), h and *helpful* actions (or preferred operators), h and *useless* actions and h when *both*, useless and helpful actions (or preferred operators), are applied.

Discussion

Overall, the concept of useless actions has shown its potential in an impressive way. In particular, it is interesting that the combination with helpful actions or preferred operators often yields such significant performance gains. Although the ideas of useful (helpful or preferred) and useless actions are similar, the experiments suggest that different aspects are covered by these notions. Therefore, we are able to identify "useful" actions, i. e. actions that are classified as useful by the heuristic, that are actually not useful in practice.

In our experiments, we give preference to operators that are not relatively useless. However, with small probability ϑ, we also expand relatively useless states. This is because the heuristic may be not informative enough which can lead to wrong classifications. On the one hand, with the alternative strategy of taking relatively useless states *only* if the other queues are empty ($\vartheta = 0$), we often expand slightly less states in the reported instances. This is rational and shows that our approximation of useless actions usually is good. On the other hand, however, wrong classifications as relatively useless may be serious because important states may wrongly be deferred. Hence, in some instances the number of explored states increases significantly. Therefore, it is adequate to expand also states with a small probability that are classified as useless.

	h^{FF}			h^{FF} + helpful			h^{FF} + useless			h^{FF} + helpful + useless		
inst.	length	states	time	length	states	time	length	states	time	length	states	time
PATHWAYS												
13	–	–	–	107	35810	2.4	98	6518	0.9	99	3002	0.4
14	–	–	–	113	31525	3.5	113	11127	1.7	110	11853	2.1
15	–	–	–	105	130300	262.9	102	5620	1.0	102	3459	0.6
16	–	–	–	147	105314	12.6	150	8008	1.7	145	9763	2.1
17	–	–	–	164	46275	5.4	165	30165	7.0	156	86351	19.8
18	–	–	–	167	108323	9.3	179	37274	5.7	176	12103	1.8
26	–	–	–	–	–	–	233	49300	14.5	227[1]	89623[1]	28.0[1]
27	–	–	–	–	–	–	272	106127	69.2	259	272868	334.1
28	–	–	–	–	–	–	204	21747	7.2	217	29402	9.7
29	–	–	–	–	–	–	266	72542	26.2	263	28408	7.4
30	–	–	–	–	–	–	299	43242	12.0	279	54163	14.8
TPP												
17	–	–	–	131	271323	84.8	111	3353	2.3	110	2699	1.8
18	108	1294309	528.2	108	10027	4.8	106	2004	2.3	103	2002	2.4
19	–	–	–	177	388578	221.0	151	3612	4.6	141	2769	3.5
20	–	–	–	–	–	–	188	8587	12.6	162	2991	5.1
21	–	–	–	210	128360	358.2	179	6404	40.3	171	4275	28.0
22	–	–	–	144	87707	356.6	136	5744	49.2	115	1265	12.2
26	–	–	–	–	–	–	203[4]	10364[4]	499.8[4]	186	4623	263.7
27	–	–	–	–	–	–	230	8130	500.2	222	4034	251.0
28	–	–	–	–	–	–	–	–	–	217	5784	389.6
29	–	–	–	–	–	–	–	–	–	262[2]	6588[2]	564.3[2]
30	–	–	–	–	–	–	–	–	–	252[2]	4993[2]	518.2[2]
DEPOT												
13	26	131	0.1	28	85	0.1	26	132	0.1	27	113	0.1
14	–	–	–	48	1130989	459.5	43	1526	1.3	42	2530	1.9
15	–	–	–	124	211715	259.1	115	53804	119.0	109	7932	17.7
16	28	643	0.2	28	67344	16.6	28	373	0.3	28	557	0.3
17	23	239	0.3	25	157	0.3	25	844	1.1	24	620	0.9
18	–	–	–	–	–	–	62[3]	27669[3]	124.2[3]	56	623	4.3
19	–	–	–	49	24005	10.7	49	5844	5.0	50	5281	4.5
20	–	–	–	–	–	–	–	–	–	119	31084	223.0
21	33	381	3.0	33	112	1.5	34	760	11.0	35	387	6.8
22	–	–	–	–	–	–	–	–	–	111	4351	482.3
ASSEMBLY												
21	75	624	0.1	75	132	0.0	75	686	0.1	78	655	0.1
22	78	625	0.1	78	160	0.0	78	692	0.1	80	257	0.0
23	–	–	–	107	332	0.1	107	1266	0.2	103	1446	0.3
24	–	–	–	103	390	0.1	101	1375	0.2	107	352	0.1
25	–	–	–	108	1469	0.2	112	1893	0.3	102	1157	0.2
26	98	507589	60.7	112	778	0.1	98	2285	0.4	99	1124	0.2
27	–	–	–	–	–	–	143	15199	2.8	123	3837	0.7
28	–	–	–	102	731	0.1	96	1223	0.2	96	1174	0.2
29	96	749	0.1	96	194	0.0	97	864	0.2	96	940	0.2
30	112	6422	0.7	123	879	0.1	112	5703	1.0	111	1275	0.2
PIPESWORLD-NOTANKAGE												
35	–	–	–	28	3162	1.7	29	22622	17.8	29	326	0.6
36	–	–	–	84	105270	53.2	–	–	–	83	20456	20.5
37	–	–	–	38	35865	23.7	49	131850	152.8	60	156041	188.1
38	–	–	–	53	58454	38.0	60	4562	5.1	38	1543	2.3
39	–	–	–	33	104	0.6	38	2688	6.6	31	264	1.3
40	–	–	–	53	196310	270.0	63	3254	8.8	49	4904	13.9
41	14	21338	33.1	12	141	0.8	12	150	1.3	12	50	0.9
45	–	–	–	–	–	–	–	–	–	97[9]	3763[9]	64.9[9]
49	–	–	–	46	569	8.5	50	2111	54.3	53	1907	39.7
50	–	–	–	63	7730	110.4	70	3832	110.6	62[3]	4879[3]	152.9[3]

Table 1: Results for the FF heuristic

	h^{cea}			h^{cea} + preferred			h^{cea} + useless			h^{cea} + preferred + useless		
inst.	length	states	time	length	states	time	length	states	time	length	states	time
PATHWAYS												
11	–	–	–	84	311891	20.7	82	4817	0.7	82[2]	2143[2]	0.3[2]
12	–	–	–	106	173239	16.0	–	–	–	116[6]	3637[6]	0.6[6]
13	–	–	–	105	27731	2.4	–	–	–	106	1685	0.3
14	–	–	–	–	–	–	116	6625	1.2	119[1]	4793[1]	1.0[1]
15	–	–	–	–	–	–	112	4291	0.8	112	9917	6.0
16	–	–	–	148	154849	27.5	–	–	–	–	–	–
26	–	–	–	–	–	–	–	–	–	231[7]	44675[7]	18.3[7]
27	–	–	–	–	–	–	257	137407	79.7	253	78272	76.2
28	–	–	–	–	–	–	228	25468	9.8	223[2]	26904[2]	12.2[2]
29	–	–	–	–	–	–	281	327166	321.0	288[4]	112094[4]	48.6[4]
30	–	–	–	–	–	–	–	–	–	293[2]	56674[2]	26.1[2]
TPP												
12	72	3126	0.4	84	916	0.2	74	4306	1.1	74	3738	1.0
13	60	12992	2.6	61	1155	0.3	68	646	0.3	66	2650	1.1
14	97	15416	3.1	100	4654	1.1	92	11860	4.9	89	8560	3.6
15	110	10676	2.2	120	4705	1.1	104	10197	4.7	107	10131	4.5
16	126	37546	12.4	135	66008	31.7	121	6855	7.3	140	28210	30.6
18	99	140394	360.4	96	3854	7.1	98	33962	122.1	98	26065	90.5
21	–	–	–	192	87043	567.1	173[1]	41160[1]	331.8[1]	185[6]	47493[6]	360.0[6]
22	–	–	–	136	21697	224.1	122	9175	149.5	128	7417	109.6
23	–	–	–	–	–	–	168[9]	48270[9]	545.1[9]	189[7]	24610[7]	341.8[7]
24	–	–	–	–	–	–	156	14438	259.8	152	22794	379.3
DEPOT												
11	87	56860	99.2	77	24531	44.1	77	1187	4.8	68	1888	7.6
12	–	–	–	–	–	–	–	–	–	111	9441	169.9
13	25	135	0.2	25	76	0.1	25	147	0.3	26	83	0.3
14	–	–	–	–	–	–	42	1297	5.9	44	1132	5.4
15	–	–	–	–	–	–	94	4379	93.2	116	3011	67.5
16	31	999	0.9	30	175	0.2	31	370	0.8	26	67	0.2
17	–	–	–	26	25455	67.5	23	1287	7.4	29	169	1.5
18	–	–	–	–	–	–	–	–	–	57[1]	514[1]	22.2[1]
19	–	–	–	–	–	–	53	3862	17.3	48	978	5.4
20	–	–	–	–	–	–	89	4326	175.9	102[7]	5763[7]	246.1[7]
21	34	526	11.2	34	179	4.5	34	582	29.6	34	286	16.8
ASSEMBLY												
21	–	–	–	93	404	0.1	79	361818	251.7	88	703	0.2
22	–	–	–	122	61853	15.2	–	–	–	78	9958	3.8
23	–	–	–	–	–	–	115	45765	20.1	105	1430	0.5
24	–	–	–	–	–	–	–	–	–	105	1442	0.6
25	–	–	–	–	–	–	–	–	–	106	969	0.5
26	–	–	–	–	–	–	–	–	–	105	1274	0.4
27	–	–	–	–	–	–	–	–	–	119	2098	1.1
28	–	–	–	–	–	–	–	–	–	113	1313	0.5
29	–	–	–	–	–	–	–	–	–	114	1346	0.6
30	–	–	–	176	64206	21.6	138	407651	272.7	112	1636	0.7
PIPESWORLD-NOTANKAGE												
34	61	6082	94.8	61	11219	75.9	78	3714	142.9	73	1201	48.8
35	–	–	–	–	–	–	32	436	24.6	48	1965	57.3
36	–	–	–	–	–	–	–	–	–	62	12346	451.4
37	–	–	–	40	15640	533.9	50	2372	156.0	46[9]	6008[9]	569.8[9]
38	–	–	–	36	1880	88.3	–	–	–	52[2]	2313[2]	190.2[2]
39	–	–	–	49	10260	489.6	–	–	–	34	224	21.3
40	–	–	–	–	–	–	46	2297	320.4	43	5877	382.9
41	14	11489	189.2	12	40	1.6	12	116	7.8	12	102	7.5
49	–	–	–	38	1004	228.9	–	–	–	56	661	308.7
50	–	–	–	53	834	180.5	–	–	–	–	–	–

Table 2: Results for the h^{cea} heuristic

	h^{add}			h^{add} + useless		
inst.	length	states	time	length	states	time
PATHWAYS						
3	11	13	0.0	11	13	0.0
4	14	16	0.0	14	16	0.0
5	30	777	0.0	31	231	0.0
7	–	–	–	82	1361	0.1
10	82	790913	116.0	82	2230	0.2
15	–	–	–	112	3938	0.8
17	–	–	–	163[1]	9835[1]	2.9[1]
23	–	–	–	189	6713	2.0
27	–	–	–	247[6]	37430[6]	28.3[6]
28	–	–	–	226	20204	12.4
TPP						
10	98	9199	0.7	90	574	0.1
11	121	9931	1.1	115	1222	0.2
12	129	20566	2.4	110	1104	0.2
13	68	34679	6.0	67	483	0.2
14	104	56733	11.8	99	1413	0.6
15	124	37706	6.9	117	1803	0.7
21	–	–	–	214	5538	72.8
22	–	–	–	152	3170	62.5
23	–	–	–	212	9829	203.5
24	–	–	–	204	5532	127.8
25	–	–	–	244	9848	346.3
DEPOT						
11	100	90113	62.0	78	1803	1.9
12	–	–	–	121[1]	91787[1]	353.6[1]
13	26	139	0.1	26	163	0.2
14	–	–	–	44	1504	2.4
15	–	–	–	100	4710	24.8
16	–	–	–	28	4703	3.6
17	–	–	–	23	1465	2.7
19	–	–	–	54	5432	9.1
20	–	–	–	109	9257	151.6
21	33	530	10.0	33	642	17.7
ASSEMBLY						
9	55	105514	12.0	55	44395	6.1
10	60	922059	221.2	60	124743	17.3
11	59	146251	22.0	59	30178	4.7
12	71	333499	45.3	71	5441	0.8
13	71	227975	27.3	71	23754	3.2
18	–	–	–	94	138411	57.2
19	–	–	–	90	530361	135.9
21	–	–	–	79	435846	209.2
23	–	–	–	115	50696	14.6
30	–	–	–	138	639642	293.6
PIPESWORLD-NOTANKAGE						
31	–	–	–	37	773	0.8
32	–	–	–	47	5355	4.7
33	–	–	–	64	3313	3.1
34	–	–	–	66[6]	45506[6]	39.4[6]
35	–	–	–	46	5593	9.8
37	–	–	–	46	3082	9.3
38	–	–	–	65	8818	28.0
39	–	–	–	38	3152	18.8
40	–	–	–	47	11003	58.4
41	14	21349	117.0	12	132	2.0
50	–	–	–	83[1]	8541[1]	365.5[1]

Table 3: Results for the additive heuristic

	normal	+ helpful	+ useless	+ both
h^{FF}				
PATHWAYS (30)	23	14	1	0
TPP (30)	13	9	3	0
DEPOT (22)	8	5	3	0
ASSEMBLY (30)	15	1	0	0
PIPESWORLD (50)	22	9	10	7
h^{cea}				
PATHWAYS (30)	24	18	15	5
TPP (30)	13	11	9	9
DEPOT (22)	11	9	4	1
ASSEMBLY (30)	20	12	8	0
PIPESWORLD (50)	25	16	13	8
h^{add}				
PATHWAYS (30)	24	n/a	16	n/a
TPP (30)	15	n/a	5	n/a
DEPOT (22)	11	n/a	3	n/a
ASSEMBLY (30)	20	n/a	8	n/a
PIPESWORLD (50)	29	n/a	13	n/a
h^{max}				
PATHWAYS (30)	25	n/a	24	n/a
TPP (30)	21	n/a	17	n/a
DEPOT (22)	17	n/a	12	n/a
ASSEMBLY (30)	20	n/a	8	n/a
PIPESWORLD (50)	33	n/a	26	n/a

Table 4: Number of unsolved instances by domain. First column shows total number of instances in parentheses.

Let us have a closer look at the class of heuristics for which our method is best suited. Although it can be combined with *arbitrary* heuristics, useless actions work best with heuristics that are computed on-the-fly because of the low time overhead. Contrarily, heuristics based on pattern databases (Culberson and Schaeffer 1998) compute a lookup table for the heuristic values in a preprocessing step. It is not obvious how useless actions can be combined *efficiently* with such heuristics, because for every reduced planning instance, a separate pattern database had to be computed, which usually causes a too large time overhead. However, we have shown that for heuristics that are computed on-the-fly, the time overhead is not an issue, and the overall performance is often improved. How to combine our method efficiently for heuristics based on pattern databases is an interesting topic for future work.

Related Work

Planning as heuristic search was pioneered by Bonet and Geffner (2001). They proposed the h^{add} and the h^{max} heuristic as well as the first popular planner based on heuristic search HSP. Meanwhile, the evaluation of actions (not just states) has also been studied. Related work in this area can roughly be divided into two categories: methods that evaluate actions *during* the search and methods that do it *prior* to the search.

Hoffmann and Nebel (2001) proposed *helpful actions*. They are used to select a set of promising successors to a search state. An action is considered as helpful if it is applicable and adds at least one goal at the first time step during the computation of the FF heuristic. Vidal (2004) extended

the idea of helpful actions. Instead of preferring states that result from applying helpful actions, he proposed to extract a prefix of actions from the relaxed solution that are applicable in sequence to the current search state. Helmert's *preferred operators* (2006) are similar to helpful actions in the context of the Causal Graph heuristic. All these approaches identify useful actions during the computation of the heuristic. Contrarily, we identify useless actions based on reduced planning instances. Therefore, our method can be combined with arbitrary heuristics.

Nebel et al. (1997) removed *irrelevant facts and operators* from a planning task in a preprocessing step. They removed operators and variables that are probably not needed in order to find a solution. They are identified by solving a relaxed problem. If certain variables and operators are not needed they are removed from the original planning task. However, in contrast to our approach, this method is not solution-preserving. In our work, relatively useless operators are identified on-the-fly, where it depends on the current state if an operator is relatively useless.

In the area of domain-specific planning, Bacchus and Ady (2001) proposed to avoid certain action sequences that do not make sense. For example, a transporter that has to deliver an object to some place should not load and unload it without moving. However, such obviously unnecessary action sequences are identified manually.

Conclusion

We have introduced a general technique to enhance heuristic search by identifying relatively useless actions. In contrast to earlier work like helpful actions or preferred operators, our method can be combined with arbitrary heuristics. We have shown that it often yields significant performance improvements on a number of commonly known heuristics and planning benchmarks from the International Planning Competitions.

In the future, we plan to extend the concept of useless and relatively useless actions to *paths*. Where we considered single actions for uselessness in this work, we will investigate how *sequences* of actions can be classified as useless. Another extension of our concept is to consider more than two degrees of uselessness. We expect that algorithms exploiting that knowledge further improve planning as heuristic search.

Acknowledgments

Many thanks to Malte Helmert for his assistance with FAST DOWNWARD, for providing us an early implementation of the h^{cea} heuristic and for fruitful discussions about useless actions.

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

Bacchus, F., and Ady, M. 2001. Planning with resources and concurrency: A forward chaining approach. In Nebel, B., ed., *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, volume 1, 417–424. Morgan Kaufmann.

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS^+ planning. *Computational Intelligence* 11:625–656.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

Chen, Y.; Wah, B. W.; and Hsu, C.-W. 2006. Temporal planning using subgoal partitioning and resolution in SGPlan. *Journal of Artificial Intelligence Research* 26:323–369.

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

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research* 20:239–290.

Helmert, M., and Geffner, H. 2008. Unifying the causal graph and additive heuristics. In Rintanen, J.; Nebel, B.; Beck, C.; and Hansen, E., eds., *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS 2008)*. AAAI Press.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In Boddy, M.; Fox, M.; and Thiébaux, S., eds., *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS 2007)*, 176–183. AAAI Press.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

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

Nebel, B.; Dimopoulos, Y.; and Koehler, J. 1997. Ignoring irrelevant facts and operators in plan generation. In Steel, S., and Alami, R., eds., *Proceedings of the 4th European Conference on Planning (ECP 1997)*, volume 1348 of *LNCS*, 338–350. Springer-Verlag.

Vidal, V. 2004. A lookahead strategy for heuristic search planning. In Zilberstein, S.; Koehler, J.; and Koenig, S., eds., *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS 2004)*, 150–159. AAAI Press.