Living on the Edge: Safe Search with Unsafe Heuristics

Erez Karpas and **Carmel Domshlak**
Faculty of Industrial Engineering & Management,
Technion, Israel

Abstract

Considering heuristic search for satisficing and cost-optimal planning, we introduce the notion of *global safeness* of a heuristic, a property weaker than the standard safeness, which we claim is better for heuristic search. We describe one concrete approach for creating globally safe and optimality preserving heuristics by exploiting information in the history of the search via a new form of inference related to existential landmarks. We evaluate our approach on some state-of-the-art heuristic search tools for cost-optimal planning, and discuss the outcomes of this evaluation.

Introduction

These days, state-space heuristic search is the most prominent approach to classical planning, though the way it is exploited in planning differ in some crucial respects from the way it is approached in the search community. In most heuristic search literature, the assumption is that the search problem and the heuristic are both given as a "black box"—the structure of the search space and the way the heuristic works are hidden from the search algorithm. This assumption suffices for analyzing various properties of the search algorithms, but limits the palette of generic enhancements that can possibly be applied to heuristic search. In contrast, in domain independent planning, the structure of the search space is made explicit by means of a description of the problem in terms of state variables, action preconditions and effects specified in terms of these state variables, and so on. This allows for exploiting more information than would be available in "classical" heuristic search. This is not a new observation — in fact, helpful actions and preferred operators (Hoffmann and Nebel 2001; Helmert 2006) are just one example of using more information than would be available for a "classical" heuristic-search algorithm.

Although preferred operators still only look at one search state, recent work has shown that it's possible use more information than is available in a single state. One example is the *LM-A** search algorithm (Karpas and Domshlak 2009), which uses information from multiple paths in the search space to enhance landmark-based heuristics. Another example of a "non-classical" heuristic is selective-max, an online learning approach which uses information from previously evaluated states, in the form of examples for a classifier (Domshlak, Karpas, and Markovitch 2010).

In this paper, we continue this line of investigation by taking a closer look at the *safeness* property of heuristics. A heuristic function h is called *safe* if for all states $s \in S$, if h declares s to be a dead-end (that is, $h(s) = \infty$), then s really is a dead-end. We claim that this definition is too restrictive, and instead formulate the more desired property of *global-safeness*. We also demonstrate how one can derive a path-dependent globally-safe heuristic, and draw connections to a new type of information which can be inferred, which we call existential landmark.

Preliminaries

We consider planning tasks formulated in STRIPS (Fikes and Nilsson 1971). A planning task $\Pi = \langle P, A, s_0, G \rangle$, where P is a set of propositions, A is a set of actions, each of which is a triple $a = \langle \mathsf{pre}(a), \mathsf{add}(a), \mathsf{del}(a) \rangle$, $s_0 \subseteq P$ is the inital state, and $G \subseteq P$ is the goal. For ease of notation and without loss of generality, in what follows we assume that there is a single goal proposition ($G = \{p_g\}$), which can only be achieved by one action, END.

An action a is applicable in a state s if $\mathsf{pre}(a) \subseteq s$. Applying action a in state s results in the new state $s' = (s \setminus \mathsf{del}(a)) \cup \mathsf{add}(a)$. A sequnce of actions $\langle a_0, a_1, \ldots, a_n \rangle$ is applicable in state s_0 if a_0 is applicable in s_0 and results in state s_1, a_1 is applicable in s_1 and results in s_2, and so on. An action sequence $\langle a_0, a_1, \ldots, a_n \rangle$ is a valid plan for Π if it is applicable in s_0, and $a_n = END$. An optimal plan is (one of) the shortest such valid plans. If π_1 and π_2 are two action sequences, by $\pi_1 \cdot \pi_2$ we denote the concatenation of π_1 and π_2.

Globally Safe Heuristics

A heuristic h is *safe*, if for every state s from which there is a path to the goal, $h(s) < \infty$. In other words, h does not declare any false dead ends. Most of the heuristics used in domain-independent planning (including those based on delete-relaxation, abstractions, and critical paths) are safe.

While safeness can be a useful property for pruning search sub-spaces, it is relevant only in problems exhibiting dead-ends in the first place. In general, we claim that safeness is too strong a requirement. Consider the hypothetical ideal

case, where we have a heuristic assigning a value of h^* (or, actually, any other finite value) to the states along some optimal plan, and ∞ to all other states. Using any reasonable search algorithm with this heuristic would lead straight to finding an optimal solution, without getting sidetracked at all. However, this heuristic is clearly not safe, since it declares many false dead-ends. This idealized scenario suggests examining substantial relaxations of the safeness property, defined below.

Definition 1 (G-safe & GO-safe heuristics) *A heuristic h is called* globally safe (G-safe, for short) *if, for any planning task Π, if Π is solvable, then there exists a valid plan ρ for Π such that for any state s along ρ, $h(s) < \infty$. In particular, if there exists such an* optimal *plan ρ, then h is called* globally safe with respect to optimality (or GO-safe, for short).

The following proposition follows directly from the definition.

Proposition 1 *Safeness implies GO-safeness implies G-safeness.*

The next propositions capture the main attractiveness of employing G-safe and GO-safe heuristics.

Proposition 2 *Any complete systematic search algorithm, which detects dead-ends using a G-safe heuristic h, will find a solution for any solvable planning task Π.*

Proof sketch: h is G-safe, and thus there is a plan ρ such that states along ρ are not declared by h to be dead-ends. Let s be the shallowest state along ρ which has not been expanded yet. Since we are using a complete search algorithm, search will not terminate before s is expanded, unless another solution is found before that. Thus, either the solution ρ will be found, or another solution will be found before ρ is fully expanded. ∎

Proposition 3 *Let h be a GO-safe heuristic, admissible for all states which it does not declare as dead-ends. Then, for any solvable planning task Π, A^* search using h will find an optimal solution for Π.*

Proof sketch: Consider the transition system induced by h, which is the transition system of Π, with edges outgoing from states that are declared by h to be dead-ends removed. This is the transition system that is searched when using h. Because h is GO-safe, every optimal path to the goal in the induced transition system corresponds to an optimal solution of Π. And since h is admissible in the induced transition system, A^* will find an optimal path in that system. ∎

While the notion of globally safe heuristics is appealing in theory, we still need to show that a concrete non-trivial instance of this notion actually exists. Indeed, at the moment of writing this paper, we could not provide the reader with an interesting example of *state-dependent* G-safe heuristics. However, the situation is different with the more general family of *path-dependent heuristics*, and thus we proceed with formalizing our relaxed versions of safeness with respect to such heuristics.

Definition 2 (G-safe path-dependent heuristic) *A path-dependent heuristic h is called* G-safe*, if, for any planning task Π, if Π is solvable, then there exists a valid plan ρ for Π, such that for every prefix ρ' of ρ, $h(\rho) < \infty$. That is, when h evaluates any prefix of ρ, it is not declared as a dead-end. In particular, if there exists such an* optimal *plan ρ, then h is called* GO-safe.

In fact, since any state-dependent heuristic can be seen as a path-dependent heuristic which only looks at the last state reached by the path, Definition 1 can be seen as a special case of Definition 2. In the rest of the paper, we describe a procedure for creating GO-safe path-dependent heuristics, and share our findings on the empirical effectiveness of adopting GO-safeness.

Unjustified Actions and Global Safeness

Our G-safe path-dependent heuristics are based upon the notion *unjustified actions*. Informally, an action along a plan is unjustified if removing it from that plan does not invalidate the latter. In order to formally define unjustified actions, we use the notion of plan's *causal links*.

Let $\pi = \langle a_0, a_1, \ldots a_n \rangle$ be an action sequence applicable in state s_0. The triple $\langle a_i, p, a_j \rangle$ forms a *causal link* if $i < j$, $p \in \mathsf{add}(a_i)$, $p \in \mathsf{pre}(a_j)$, $p \notin s_i$ (s_i is the state before applying action a_i), and for $i < k < j$, $p \notin \mathsf{del}(a_k) \cup \mathsf{add}(a_k)$. In other words, p is a precondition of a_j, and is achieved by a_i, and is not deleted or added by some other action until a_j occurs. In such a causal link, a_i is called its *supporter*, and a_j is called its *consumer*.

Definition 3 (Unjustified Action) *Given a plan $\rho = \langle a_0, a_1, \ldots a_n \rangle$, the action instance $a_i \neq END$ is* unjustified *if there is no causal link in ρ such that a_i is the supporter in that causal link.*

An immediate from the definition, yet important for what comes next, property is that *optimal plans never contain unjustified actions*. This brings us to define "hopeless paths" in the (forward) search.

Definition 4 (Hopeless paths) *For any planning task Π, path π from the initial state s_0 to a state s is* hopeless *if there is no path π' from s to the goal such that $\pi \cdot \pi'$ contains no unjustified actions.*

In other words, if π is hopeless, then any plan that has π as a prefix will contain unjustified occurrences of actions. Using this definition, we can finally formulate the connection between unjustified actions and global safeness.

Theorem 1 (Global path-dependent safeness) *Let h be a safe path-dependent heuristic. Then the path-dependent heuristic*

$$h'(\pi) := \begin{cases} \infty & \text{if } \pi \text{ is hopeless} \\ h(\pi) & \text{otherwise} \end{cases}$$

is a GO-safe path-dependent heuristic.

Proof sketch: Again, the claim trivially holds for unsolvable tasks. Let Π be a solvable planning task, and let $\rho = \langle a_0, a_1, \dots a_n \rangle$ be an optimal plan for Π. By the virtue of being optimal, π does not contain any unjustified actions. Let $\pi = \langle a_0, a_1, \dots a_i \rangle$ be a prefix of ρ, leading to state s. Then $\pi' = \langle a_{i+1}, a_{i+2}, \dots a_n \rangle$ is a path from s to the goal, such that $\pi \cdot \pi'$ contains no unjustified actions. Therefore π is not hopeless, and there thus is at least one optimal plan which is never declared as a dead-end by h'. ∎

A word of caution is in place here. Note that according to Theorem 1 it might be safe to declare a path π as a dead-end, but *not* the state s to which π leads. This happens because of the path-dependent nature of unjustified actions. To illustrate that, consider the following simple planning task $\Pi = \langle P, A, s_0, G \rangle$, where $P = \{p_1, p_2, p_g\}$, $s_0 = \{\}$, $G = \{p_g\}$, and the following actions:

- $a_1 = \langle \emptyset, \{p_1\}, \emptyset \rangle$
- $a_2 = \langle \{p_1\}, \{p_2\}, \emptyset \rangle$
- $a_{12} = \langle \emptyset, \{p_1, p_2\}, \emptyset \rangle$
- $END = \langle \{p_1, p_2\}, \{p_g\}, \emptyset \rangle$

Following path $\pi = \langle a_1, a_{12} \rangle$ leads to state $s = \{p_1, p_2\}$. Action a_1 along π can not be justified, since a_{12} achieves p_1, the proposition that a_1 achieves, and it is not a consumer of a_1. Therefore, π is hopeless (since there is no path π' from s to the goal that can justify a_1), and it is indeed globally safe to declare path π as a dead-end. However, it is *not* safe to declare state s as a dead-end, since Π is solvable, and it is easy to see that s lies on any solution path. The following theorem addresses this issue.

Theorem 2 (Global safeness along optimal paths)
If h be a safe heuristic, then the heuristic

$$h'(s) := \begin{cases} \infty & \text{if some optimal path from } s_0 \text{ to } s \text{ is hopeless} \\ h(s) & \text{otherwise} \end{cases}$$

is a GO-safe heuristic.

Proof sketch: Let Π be a (solvable) planning task, and s be a state of Π. If there is no optimal plan for Π which goes through state s, then by definition, it is GO-safe to declare s as a dead end. Assume now that there is an optimal plan $\rho = \rho_1 \cdot \rho_2$ for Π such that applying ρ_1 in s_0 leads to s. Let π be an optimal path from s_0 to s. Assume for contradiction that π is hopeless, and there is no path π' from s to the goal such that $\pi \cdot \pi'$ contains no unjustified actions. Then, specifically, $\pi \cdot \rho_2$ contains some unjustified action. However, $\pi \cdot \rho_2$ is an optimal plan (since π is an optimal path to s, ρ_2 is an optimal path from s to the goal, and s is along an optimal plan), and that contradicts the fact that optimal plans never contain unjustified actions. Hence, if an optimal path π from s_0 to state s is hopeless, then s is not on an optimal path from s_0 to the goal. ∎

The problem with applying Theorem 2 is that we do not always know when the current path to some state s is indeed optimal. However, by modifying the A^* search algorithm to reevaluate the heuristic *every time* a state s is reopened (because a shorter path to s has been found), we can prune states according to Theorem 2, and still guarantee that the search will return an optimal solution. For the sake of brevity, we omit a formal proof of this, but refer the reader to the proof that A^* with an admissible heuristic returns an optimal solution, and specifically to the following lemma (Pearl 1984, p. 78, Lemma 2): *Let n' be the shallowest open node on an optimal path π from s_0 to some arbitrary node n''. Then $g(n') = g^\star(n')$.*

Exploiting Unjustified Actions in Search

While Theorem 1 brings us closer to globally safe heuristics, we still need a way of identifying whether a given path from the initial state is hopeless. In what follows, we close the gap by presenting two such methods.

Existential Landmarks

Suppose state s was reached via path $\pi = \langle a_0, a_1, \dots, a_n \rangle$. Using standard causal link analysis, we can identify the causal links present in π. We make one slight enhancment to the standard analysis by not allowing an action a to justify its inverse action a', where inverse actions are identified according to the criteria in Hoffmann (2002). Denote by U the set of actions in π which are not supporters in any causal link in π. The pseudo code for extracting U is depicted in Figure 1.

For π to have a continuation π' such that $\pi \cdot \pi'$ has no useless occurrences of actions (and thus not be hopeless), every action in U must be the supporter of some action in π'. Using the same causal link analysis, we can also identify which propositions can possibly appear in such causal links for each action. Denote by $pp(a)$ the set of propositions which $a \in U$ is potentially a supporter of. Note that in some cases, it is possible to detect dead-ends, just by noticing that $pp(a) = \emptyset$, which means that all of the effects of a have either been reachieved by some other action or deleted, and so there is no way to justify a.

For $\pi \cdot \pi'$ to contain no useless actions, for all $a \in U$, a must support some proposition in $pp(a)$. Let $ia(a) = \{a' \mid \mathsf{pre}(a') \cap pp(a) \neq \emptyset\}$ denote the set of all actions which have a precondition in $pp(a)$. Then any continuation π' of π, such that $\pi \cdot \pi'$ contains no useless occurrences of actions, must use an action from $ia(a)$ for each $a \in U$. Although this seems similar to the notion of disjunctive action landmark, the difference is that $ia(a)$ does not constrain *all* plans, but rather only *optimal plans having π as a prefix*. Since $ia(a)$ means that there *exists* some plan which achieves it, we call it an existential disjunctive action landmark.

Once we have these existential disjunctive action landmarks, it is possible to reason about them in combination with "regular" landmarks. For example, it is possible to find a cost partitioning between these existential disjunctive action landmarks, and any other set of landmarks, such as those used by h_{LA} (Karpas and Domshlak 2009). The "achievers" of such an existential disjunctive action landmark are simply the actions that compose the landmark.

causal-link-analysis(π)
$support := \emptyset$
// $support$ holds pairs of propositions
// and the actions that supported them
for $a_i \in \pi$ (in order)
  update(a_i)

update(a)
for $p \in (\mathsf{pre}(a) \cap supported\_props)$
  // We have used the supported proposition,
  // it is no longer unjustified
  $a' := supporter(p)$
  // Make sure an action doesn't justify its inverse
  if a' is not the inverse of a
    // remove a' and all its supported propositions
    $support := support \setminus \{\langle p', a' \rangle | \exists p' : \langle p', a' \rangle \in support\}$
for $p \in (\mathsf{add}(a) \cup \mathsf{del}(a))$
  // We need to justify the effects of a later
  if $p \in supported\_props$
    // This effect was already achived by another action
    // we need to make sure this is not its last effect
    $a' := supporter(p)$
    $support := support \setminus \{\langle p, a' \rangle\}$
    if $supported\_by(a') = \emptyset$ **then**
      return dead-end
  $support := support \cup \{\langle p, a \rangle\}$

Macros used:
$supported\_props \equiv \{p | \exists a : \langle p, a \rangle \in support\}$
$supported\_by(a) \equiv \{p | \langle p, a \rangle \in support\}$
$supporter(prop) \equiv a$ s.t. $\langle p, a \rangle \in support$

Figure 1: Procedure for the causal link analysis.

A Compilation-Based Approach

A different approach for identifying hopeless paths is based on compiling into the planning task Π (part of) the constraints imposed on it by unjustified actions. Let state s be a state reached via path π. We will define a new planning task $\Pi'_{s,\pi} = \langle P', A', s', G' \rangle$ as follows:

- $P' = P \cup \{justified(a) \mid a \in A\}$
- $A' = \{\langle \mathsf{pre}(a) \cup \{justified(a)\},$
  $\mathsf{add}(a) \cup \{justified(a') \mid a' \in A, \mathsf{add}(a') \cap \mathsf{pre}(a) \neq \emptyset\},$
  $\mathsf{del}(a) \cup \{justified(a)\} \rangle \mid a \in A\}$
- $s' = s \cup \{justified(a) \mid a$ is not unjustified in π
  or a does not occur in $\pi\}$
- $G' = G \cup \{justified(a) \mid a \in A\}$

In words, we add a proposition $justified(a)$ for each action a, with the meaning that $justified(a)$ holds when action a has been justified (or does not need to be justified). Applying action a deletes $justified(a)$, thus forcing some later action to use one of a's effects. Applying action a', which has one of a's effects as a precondition, adds $justified(a)$, since a' used an effect of a.

Note that this compilation is weaker than Definition 3, since we do not require that no action deletes or adds the supported proposition along the way, and we do not account for which action achieved which proposition. However, we can still show the following.

Theorem 3 (Soundness of Π') *Let Π be a solvable planning task, and let state s be reached via path π. If π is not hopeless, then $\Pi'_{s,\pi}$ is solvable.*

Proof sketch: Path π is not hopeless, therefore there exists some path π' such that $\pi \cdot \pi'$ contains no unjustified actions. We will show that π' is also a solution for $\Pi'_{s,\pi}$. Since $\pi \cdot \pi'$ contains no unjustified actions, for every unjustified action a in π, there is an action a' in π' that uses an effect of a. Therefore π' will achieve all the $justified(a)$ propositions which were false in s'. Furthermore, no action in π' is unjustified, and so for every action a along π', there is another action in π' which achieves $justified(a)$. Since π' achieves all of the goals of Π, and all of the $justified$ goals of $\Pi'_{s,\pi}$, π' is a solution for $\Pi'_{s,\pi}$. ∎

It is also possible to come up with a different compilation, which has a proposition $achieved(a, p)$ for each pair of action a and proposition p (from the original task), and denotes that p was achieved by action a. Such a compilation must use conditional effects, but has the potential to be more informative than the simple compilation described here, since it does not ignore the information about which action achieved which proposition.

Empirical Evaluation

We have presented the concept of unjustified actions, and suggested two principled ways in which they can be explited in heuristic-search planning. Since the compilation-based approach requires introducing many new state variables (one for each grounded action), which severly limits the applicabilty of this approach in practice, we did not perform an empirical evaluation of it in this paper.

We have implemented the existential landmark approach on top of the Fast Downward planning system (Helmert 2006), and combined the existential disjunctive action landmarks with the optimal cost partitioning of landmarks (Karpas and Domshlak 2009). In our evaluation we used three baseline heuristic: $h_{\text{LM-CUT}}$, h_{LA}, and h_{GC}.

- h_{GC} is the most basic admissible landmark-based heuristic: it only accounts for the goal landmarks, and combines their information via the optimal action cost partitioning. In other words, h_{GC} is simply an admissible goal-count heuristic. We add the existential action landmarks discovered using our approach to the cost partitioning.
- h_{LA} (Karpas and Domshlak 2009) is state-of-the-art admissible landmarks heuristic, with efficient optimal cost partitioning (Keyder, Richter, and Helmert 2010). The landmarks used here are those discovered using the RHW method (Richter and Westphal 2010). Here as well, we add the existential action landmarks discovered using our approach to the cost-partitioning.

Domain	h_{GC}	h_{GC}+	h_{LA}	h_{LA}+	$h_{\text{LM-CUT}}$	$h_{\text{LM-CUT}}$+
BLOCKS	17	17	21	21	28	28
DEPOT	3	**4**	7	7	7	7
DRIVERLOG	7	**9**	12	**13**	13	13
LOGISTICS00	10	10	20	20	20	20
TRUCKS-STRIPS	3	3	**6**	5	**10**	9
ZENOTRAVEL	8	8	9	9	13	13
TOTAL	48	51	75	75	91	90

Table 1: Total number of solved tasks for each method. Results where there was a change between a baseline and its enhancement are in bold.

- $h_{\text{LM-CUT}}$ (Helmert and Domshlak 2009) is a state-of-the-art admissible landmarks heuristic, but unfortunately, it does not support adding the existential landmark information to its cost partitioning. Therefore, the only enhancement we used with $h_{\text{LM-CUT}}$ was path pruning: upon reaching a state s via path π, we look at the causal link analysis of π, and if there is an action in π such that $pp(a) = \emptyset$ (and therefore a can not be justified), then we prune path π.

We compare the baseline heuristics to the same heuristics, enhanced by our existential landmarks (as explained above). To ensure admissibility, in the runs enhanced with unjustified action information, we modified A^* to re-evaluate the heuristic once a shorter path to a known state has been found. All of the experiments were conducted with a 3GB memory limit and a 30 minute time limit, on a single core of a 2.33GHz Intel Q8200 CPU.

Table 1 lists the number of tasks solved using each baseline configuration, and the number of tasks solved when using unjustified actions. Overall, there is not much change in the number of solved tasks, and the overhead involved in keeping track of unjustified actions even leads to some losses. However, looking into the results in more details reveals a more colorful picture.

Tables 2 lists the average ratios of expanded states, evaluations and total solution time, when using unjustified actions, relative to to the baseline of the same method. For each heuristic, this is averaged on the tasks solved by both the baseline and the version enhanced with unjustified actions.

Looking at Table 2a, we can see that the number of expanded states is reduced drastically. We remark that only in one ZENOTRAVEL task, using unjustified actions led to more expanded states than with the baseline, when using $h_{\text{LM-CUT}}$. Especially remarkable is the finding that in LOGISTICS00, we can reduce the number of expanded states by more than half, simply by pruning hopeless paths, without modifying the $h_{\text{LM-CUT}}$ heuristic at all. On the other hand, using unjustified actions with $h_{\text{LM-CUT}}$ in BLOCKS did not make any difference. This is because the *hand-empty* predicate is achieved by any action which puts a block down, and is a precondition of any action which picks a block up, thus any action justifies the next action.

From Table 2b, we can see that even considering that the same state might be evaluated more than once with unjustified actions, the number of evaluations is still reduced overall. However, there is more than one task where the number of evaluations is increased over the baseline. Even taking the overhead of keeping track of unjustified actions into account, Table 2c still shows us that overall, using unjustified actions with $h_{\text{LM-CUT}}$ and h_{GC} speeds up search.

Domain	h_{GC} ratio	h_{LA} ratio	$h_{\text{LM-CUT}}$ ratio
BLOCKS	0.93	0.99	1.00
DEPOT	0.56	0.84	0.98
DRIVERLOG	0.58	0.68	0.82
LOGISTICS00	0.57	0.97	0.43
TRUCKS-STRIPS	0.5	0.57	0.9
ZENOTRAVEL	0.53	0.83	0.92
AVG.	0.69	0.87	0.82
NORMALIZED AVG.	0.61	0.81	0.84

(a) Expanded States

Domain	h_{GC} ratio	h_{LA} ratio	$h_{\text{LM-CUT}}$ ratio
BLOCKS	0.93	1.00	1.00
DEPOT	0.64	0.92	0.99
DRIVERLOG	0.64	0.76	0.86
LOGISTICS00	0.61	0.99	0.52
TRUCKS-STRIPS	0.64	0.73	0.90
ZENOTRAVEL	0.58	0.89	0.91
AVG.	0.73	0.92	0.85
NORMALIZED AVG.	0.67	0.88	0.86

(b) Evaluations

Domain	h_{GC} ratio	h_{LA} ratio	$h_{\text{LM-CUT}}$ ratio
BLOCKS	1.12	1.11	1.06
DEPOT	1.03	1.22	1.02
DRIVERLOG	0.84	0.96	0.98
LOGISTICS00	0.86	1.30	0.64
TRUCKS-STRIPS	1.03	1.29	1.01
ZENOTRAVEL	0.81	1.16	0.93
AVG.	0.96	1.17	0.93
NORMALIZED AVG.	0.95	1.17	0.94

(c) Total Time

Table 2: Average ratios of expanded states / evaluations / total time with unjustified actions relative to the baseline. For each heuristic, the average is over tasks solved by both the baseline and the version enhanced with unjustified actions. AVG. is the average over all tasks, and NORMALIZED AVG. is the average over domain averages.

Related

While the use of information from other states presented here might seem somewhat similar to the heuristic value propagation of pathmax (Mero 1984) or BPMX (Felner et al. 2005), this is not the case. Our approach uses knowledge about the explicit structure of the problem, in the form of causal links, and is therefore not applicable within the "classic" heuristic search, black-box view of the problem. On the other hand, pathmax and BPMX only need information about heuristic values and successor relations, and so they are applicable (and have been applied) without explicit structure of the problem.

Another field where more information than present in a single search state is used to make decisions is learning for planning. Typically, a learning/planning system attempts to learn some domain-specific knowledge offline from training examples, in order to create a more efficient planner on unseen problem instances from that domain. However, there is very little work on *online* learning for planning, which would also constitute a form of non-classical heuristic search.

A similar notion to unjustified actions, termed useless actions, was introduced by Wehrle, Kupferschmid, and Podelski (2008). The authors there defined a useless action to be an action which is not the start of an optimal path. However, it is PSPACE-hard to determine if an action is useless, and so Wehrle et al. use an approximation of useless actions, and can no longer guarantee optimal solutions. Furthermore, the definition of useless action suffers from the same problem as the definition of a safe heuristic — any action which is the start of an optimal path is not useless. It would be very interesting to identify a condition which is sufficient to ensure that a given action is the start of an optimal path, and then just prune the rest of the search space.

Conclusion

We have defined a novel notion of global safeness, and have demonstrated how unjustified actions can be used to derive globally safe heuristics. We also performed an empirical evaluation, demonstrating that exploiting global safeness via unjustified actions can lead to significant improvements to the performance of sequentially-optimal planning.

Note that using unjustified actions in satisficing search is even easier since there is no need to worry about maintaining admissibility, and we intend to explore these possibilities in our future work. Finding an efficient way to apply the compilation-based approach presented here is yet another interesting direction to explore.

References

Domshlak, C.; Karpas, E.; and Markovitch, S. 2010. To max or not to max: Online learning for speeding up optimal planning. In *AAAI*.

Felner, A.; Zahavi, U.; Schaeffer, J.; and Holte, R. C. 2005. Dual lookups in pattern databases. In *IJCAI*, 103–108.

Fikes, R. E., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *AIJ* 2:189–208.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *ICAPS*. In press.

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26:191–246.

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

Hoffmann, J. 2002. Local search topology in planning benchmarks: A theoretical analysis. In *AIPS*. 379-387.

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In *IJCAI*.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and complete landmarks for and/or graphs. In *ECAI*, 335–340.

Mero, L. 1984. A heuristic search algorithm with modifiable estimate. *AIJ* 23:13–27.

Pearl, J. 1984. *Heuristics: intelligent search strategies for computer problem solving*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Richter, S., and Westphal, M. 2010. The lama planner: Guiding cost-based anytime planning with landmarks. *JAIR* 39:127–177.

Wehrle, M.; Kupferschmid, S.; and Podelski, A. 2008. Useless actions are useful. In *ICAPS*, 388–395. AAAI Press.