Living on the Edge:
Safe Search with Unsafe Heuristics

Erez Karpas Carmel Domshlak

Faculty of Industrial Engineering and Management,
Technion — Israel Institute of Technology

June 6, 2011

Outline

Safe Heuristics

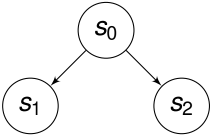
- A heuristic *h* is safe if it never declares a false dead end

$$\forall s : h(s) = \infty \implies h^*(s) = \infty$$

- Looks like a good property

Safeness - Not Such a Good Idea

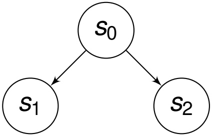
- Consider this example:



- We can prove there is a path from s_1 to the goal
- Is it safe to set $h(s_2) = \infty$?
- Should it be?

Safeness - Not Such a Good Idea

- Consider this example:



- We can prove there is a path from s_1 to the goal
- Is it safe to set $h(s_2) = \infty$?
- Should it be?

Global Safeness

- To address this, we suggest the following definitions

Globally Safe (G-Safe) Heuristic

Let Π be a solvable planning task. A heuristic h is globally-safe, if there exists a valid plan ρ for Π, such that for any state s along ρ, $h(s) < \infty$.

- In other words, when h evaluates any state along ρ, it is not declared as a dead-end.
- If ρ is optimal, h is called Globally Optimally Safe (GO-Safe)

G-Safe Heuristic

- Great — where can I get one of those?
- I don't know. But I can tell how how to find path-dependent GO-Safe heuristic

G-Safe Heuristic

- Great — where can I get one of those?
- I don't know. But I can tell how how to find path-dependent GO-Safe heuristic

G-Safe Heuristic

- Great — where can I get one of those?
- I don't know. But I can tell how how to find path-dependent GO-Safe heuristic

Globally Safe (G-Safe) Path Dependent Heuristic

Let Π be a solvable planning task. A path dependent heuristic h is globally-safe, if there exists a valid plan ρ for Π, such that for any prefix ρ' of ρ, $h(\rho') < \infty$.

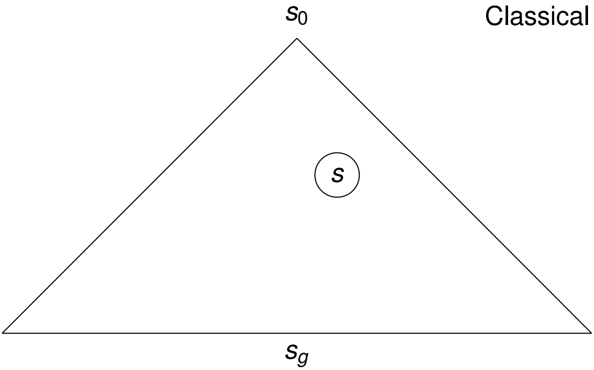
- Path dependent GO-Safeness is defined accordingly
- Since any state dependent heuristic is path dependent, this is the more general definition

Outline

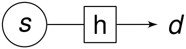
Heuristic Search — Different Perspectives

- The Classical Approach
  - Search space is given by initial state and black box successor generator
  - Heuristic function is a black box
- In Planning
  - State and Successor generator are structured and known
  - Heuristic functions are not black boxes
- This has been exploited by preferred operators, symmetry detection, . . .
- But we can do more

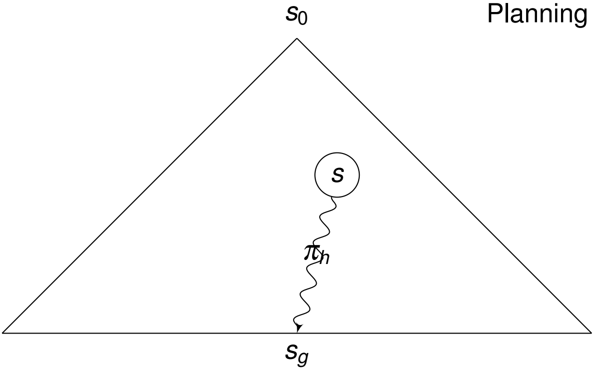
Different Perspectives — Illustrated



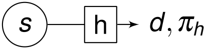
Classical Heuristic Search



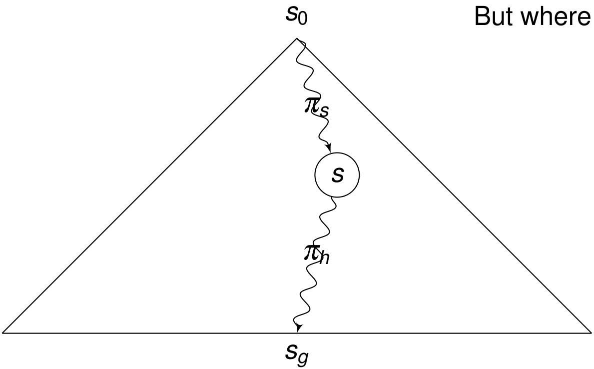
Different Perspectives — Illustrated



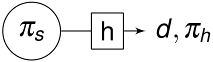
Planning (Helpful Actions)



Different Perspectives — Illustrated



But where did s come from?



Outline

A Path Dependent Information Source - Unjustified Actions

- Informally, an action a along a plan ρ is unjustified if removing a from ρ does not invalidate ρ.
- For a formal definition, we need to define causal links

Causl Link

The triple $\langle a_i, p, a_j \rangle$ forms a causal link in action sequence $\langle a_0, a_1, \ldots a_n \rangle$ if $i < j$, $p \in \text{add}(a_i)$, $p \in \text{pre}(a_j)$, $p \notin s_i$, and for $i < k < j$, $p \notin \text{del}(a_k) \cup \text{add}(a_k)$.

- In other words, p is achieved by a_i and is not deleted or added by some other action until a_j occurs, and is a precondition of a_j.
- a_i is called the supporter in this causal link, and a_j is the consumer.

Unjustified Actions

Unjustified Action

An action occurrence $a_i \neq END$ in plan $\rho = \langle a_0, a_1, \ldots a_n \rangle$ is unjustified if there is no causal link in ρ, such that a_i is the supporter in that causal link.

Easy to see:

1. Any unjusitified action occurrence can be removed from a valid plan, and the plan is still valid
2. Any optimal plan does not contain any unjustified actions

Hopeless Paths

Hopeless Path

Path π from s_0 to 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, any continuation of π will always contain unjustified actions
- Hopeless paths are the connection between path dependent GO-Safe heuristics and unjustified actions

Outline

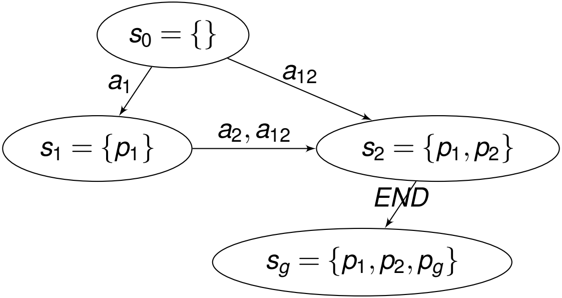
Path Dependent GO-Safe Heuristic

- Let h be any safe, path dependent heuristic for solvable planning task Π

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

  is a GO-safe path-dependent heuristic.
- This refers to the path π, not to the last state in that path

Caution is Needed — Example



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

Path $\langle a_1, a_{12} \rangle$ is hopeless

But it's not safe to prune s_2

Caution is Needed — But Not Always

- Let h be any safe heuristic for solvable planning task Π

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

is a GO-safe heuristic.

Searching with Unjustified Actions

- We know that if the path to s is optimal, $h'(s)$ is GO-Safe.
- With A^*, we don't know when the path to s is optimal.
- However, if we find a cheaper path to s, s will be reopened.
- So using A^*, but re-evaluating $h'(s)$ whenever s is reopened, will ensure optimality.

Outline

Identifying Hopeless Paths

- Everything discussed before is purely theoretical, unless we can identify if a path π is hopeless
- We propose two approaches:
  - Compilation
  - Existential Landmarks
- Both approaches are based on causal link analysis

Causal Link Analysis

- Suppose we reached state s via path π
- We can easily identify which action achieved which proposition, and which propositions were used by actions
- So we can identify which actions have already been justified, and which were not
- We make one enhancement to standard causal link analysis, by not allowing an action to justify its inverse action

Causal Link Analysis — Unjustified Actions

- Denote by U the set of actions which were not justified yet
- For each action $a \in U$, denote by $pp(a)$ the set of propositions which a is a supporter of
- a is justified when some action a' has one of $pp(a)$ as a precondition
- Therefore, for π to be non-hopeless, there must be an action a' in the continuation of π, which uses one of $pp(a)$ for every $a \in U$
- Note: if $pp(a) = \emptyset$ then π must be hopeless, since a can not be justified

Compilation Based Approach

- One way of checking if π is non-hopeless is by compilation
- For each action a we add a proposition $justified(a)$, which holds when action a has been justified (or does not need to be justified).
- Applying action a deletes $justified(a)$
- Applying action a', which has one of a's effects as a precondition, adds $justified(a)$

- This compilation is sound (although a bit weak)
- It is also not practical — it adds too many propositions, and must be updated at each state

Existential Landmarks

- Remember that for every $a \in U$ we want to have some action which uses one of $pp(a)$
- Then $pp(a)$ can be seen as a landmark, which is achieved by any action a' which has one of $pp(a)$ as a precondition
- Note that this is not a standard landmark — it is only a landmark for the continutation of π
- Therefore we call this an existential landmark

Outline

Experiments

- We implemented the existential landmarks approach
- We applied it to 3 heuristics:
  - h_{GC} — an admissible goal count heuristic
  - h_{LA} — the admissible landmarks heuristic
  - $h_{\text{LM-CUT}}$
- We added the information from the existential landmarks to the first two, using optimal cost partitioning
- We did not add the existential landmark information to $h_{\text{LM-CUT}}$, we only used pruning of hopeless paths (if $pp(a) = \emptyset$)

Coverage

Domain	h_{GC}	h_{GC} + uj	h_{LA}	h_{LA} + uj	$h_{\text{LM-CUT}}$	$h_{\text{LM-CUT}}$ + uj
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

Expansions

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

Evaluations

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

Total Time

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

Conclusion

- We have shown how to create a path dependent globally optimally safe heuristic
- In the process, we have also shown a type of existential landmark
- Applying this seems to lead to improvements in practice

Thank You

- Thank You