University of Basel

Optimality Certificates for Classical Planning

Esther Mugdan <esther.mugdan@unibas.ch>

Department of Mathematics and Computer Science, University of Basel

Artificial Intelligence Research Group

June 17, 2022

Motivation

Verify classical planning software (certificate)

So far only for plans in general and unsolvability

Motivation

Verify classical planning software (certificate)

So far only for plans in general and unsolvability

What about the optimality of a plan?

Motivation

Verify classical planning software (certificate)

So far only for plans in general and unsolvability

What about the optimality of a plan?

- Reduction to Unsolvability

Motivation

Verify classical planning software (certificate)

So far only for plans in general and unsolvability

What about the optimality of a plan?

- Reduction to Unsolvability
- Certificates for Optimality

Planning Task - Definition

$\Pi = \langle V, A, I, G \rangle$

V	finite set of state variables
A	finite set of actions
I	initial state
G	goal of the task

Planning Task - Goal

Find plan $\pi = \langle a_0, ..., a_n \rangle$ which leads from the initial state to a goal state

Optimal plan: plan with minimal cost

Unit cost: all actions have cost 1

PDDL - Domain

```
(define (domain LIGHTS)

 (:predicates (on ?x) (off ?x))

 (:action switch-on
   :parameters (?x)
   :precondition (off ?x)
   :effect (and (on ?x) (not(off ?x)))))
```

PDDL - Task

```
(define (problem LIGHTS-1)
 (:domain LIGHTS)

 (:objects A B)

 (:init (off A) (off B))

 (:goal (and (on A) (on B))))
```

General Idea

1. Solve task to find optimal cost x
2. Modify task: require cost $x - 1$

   $\rightsquigarrow$ task is unsolvable
3. Run modified task and generate unsolvability certificate
4. Verify unsolvability certificate

   $\rightsquigarrow$ x is optimal cost

Modification in PDDL - Domain

```
(define (domain LIGHTS)

 (:predicates (on ?x) (off ?x) (cost ?c) (next ?c ?n))

 (:action switch-on
   :parameters (?x ?c ?n)
   :precondition (and (off ?x)
                      (cost ?c) (next ?c ?n))
   :effect (and (on ?x) (not(off ?x))
                (cost ?n) (not(cost ?c))))
```

Modification in PDDL - Task

```
(define (problem LIGHTS-1)
 (:domain LIGHTS)

 (:objects A B 0 1)

 (:init (off A) (off B) (cost 0) (next 0 1))

 (:goal (and (on A) (on B))))
```

Setup

- Initial run to determine cost

  A^* with h^{LM-cut}

- Modified run and certificate

  A^* with h^{max}

  A^* with $h^{M\&S}$

  A^* with h^{LM-cut}

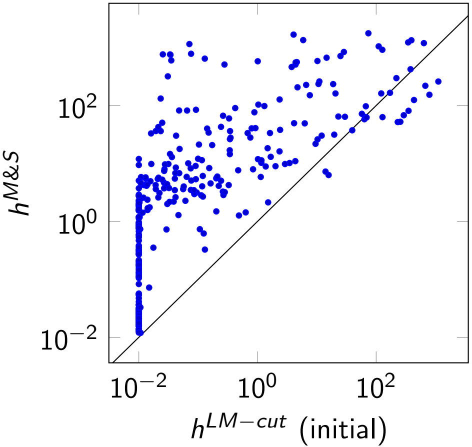
Total Runs

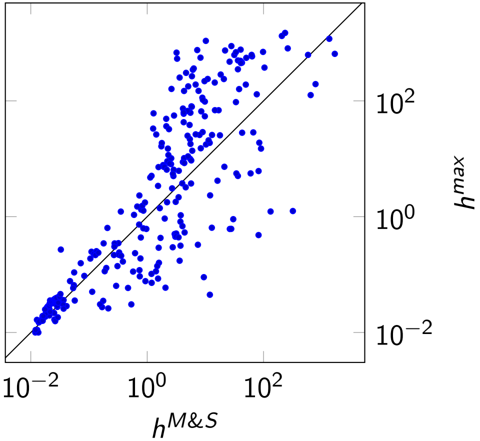
	h^{max}	$h^{M\&S}$
certificate created	292	338
search out of time	230	34
search out of memory	5	155
translate out of memory	22	22
total	549	549

278/292 certificates verified for h^{max}

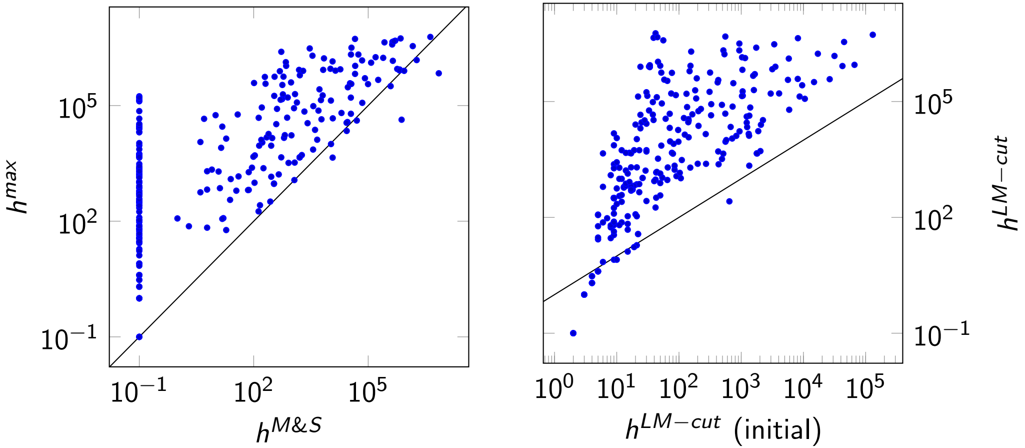
315/338 certificates verified for $h^{M\&S}$

Time Comparison



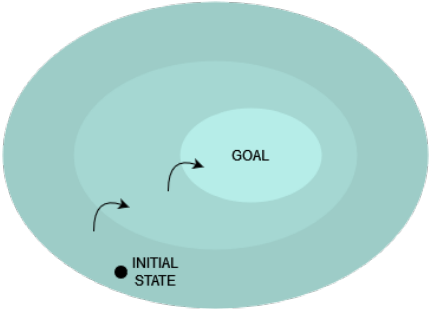


Expansion Comparison



General Idea

1. Compute optimal cost x
2. Iteratively create sets of states with at least cost $0, \ldots, x$ to goal (x-state sets)
3. If I in x-state set $\leadsto$ task has minimal cost x



Derivation Rules

D1 S_{All} is a 0-state set.

...

D6 The cost of the generated plan is x.

D7 If S_x is an x-state set, $S[A] \subseteq S_x$ and $S \cap S_G \subseteq \emptyset$,
then S is an (x+1)-state set.

D8 If S_x is an x-state set, $\{I\} \subseteq S_x$ and x is the cost of the generated plan,
then the optimal solution has cost x.

Basic Statements

B1 $L \subseteq L'$

B2 $X \subseteq X' \cup X''$

B3 $L \cap S_G \subseteq L'$

B4 $X[A] \subseteq X \cup L$

Blind Search

Use g-value of states to create sets $S_0, \dots S_x$

$\rightsquigarrow$ state with g-value $x - 1$ (or less) in set S_1

Expansion in order of g-value

$\rightsquigarrow$ goal states only in S_0

All states with g-value $< x$ are expanded

$\rightsquigarrow$ all successors of state sets are known

Blind Search - Proof Sketch

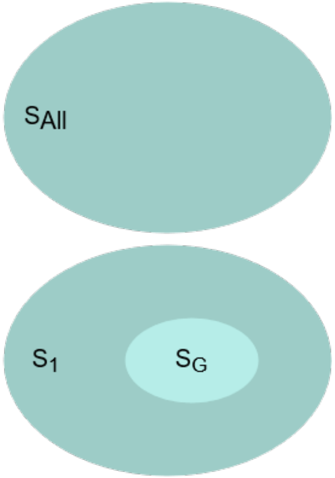
(1) $D1 \rightarrow S_{All}$ is a 0-state set

(2) $B4 \rightarrow S_1[A] \subseteq S_{All}$

(3) $B3 \rightarrow S_1 \cap S_G \subseteq \emptyset$

(4) $D7, \{(1), (2), (3)\} \rightarrow S_1$ is a 1-state set

...



Blind Search - Proof Sketch

...

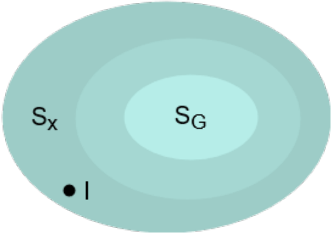
(n-3) $D7, \{(\text{n-6}), (\text{n-5}), (\text{n-4})\} \rightarrow S_x$ is a x-state set

(n-2) $B1 \rightarrow \{I\} \subseteq S_x$

(n-1) $D6 \rightarrow$ The cost of the generated plan is x.

(n) $D8, \{(\text{n-3}), (\text{n-2}), (\text{n-1})\}$

$\rightarrow$ The optimal solution has cost x.



A^* Search

Use g-value for **expanded** states to create sets $S_0, \ldots S_x$

$\rightsquigarrow$ state with g-value $x - 1$ (or less) in set S_1

Use h-value for **non-expanded** states

$\rightsquigarrow$ state with h-value h is h-state

(prove separately for each heuristic - proved for h^{max})

A^* Search

Only expanded states in $S_1, \ldots, S_x$ (according to g-value)

$\rightsquigarrow$ goal states only in S_0

All expanded states in sets $S_1, \ldots, S_x$,

All non-expanded states are h-states

$\rightsquigarrow$ all successors in union of expanded and non-expanded states

A^* Search - Proof Sketch

(0) Proof that every non-expanded state is an h-state

(1-4) As for blind search

(5) $D5, \{(0), (4)\} \rightarrow S_1 \cup \bigcup_{h(s)\geq 1}\{s\}$ is a 1-state set.

(6) $B4 \rightarrow S_2[A] \subseteq S_1 \cup \bigcup_{h(s)\geq 1}\{s\}$

(7) $B3 \rightarrow S_2 \cap S_G \subseteq \emptyset$

(8) $D7, \{(5), (6), (7)\} \rightarrow S_2$ is a 2-state set

...

Results

- **Reduction to Unsolvability**
  - Modified task
  - Good results
  - Prone to error
- **Proof System for Optimality**
  - Original task
  - Independent verification

Future work

Extend search algorithm by creation of certificate

Stand-alone verifier for certificate

(find suitable state set representation)

Consider non-unit cost tasks

Questions?