Optimality Certificates for Classical Planning

Esther Mugdan, Remo Christen, Salomé Eriksson

University of Basel

ICAPS 2023

July 11, 2023

Motivation

Verifying classical planning software

- Plan existence $\rightarrow$ (IN)VAL
- Unsolvability $\rightarrow$ Proof system
- Optimality $\rightarrow$?

Motivation

Verifying classical planning software

- Plan existence → (IN)VAL
- Unsolvability → Proof system
- Optimality → **Compilation to Unsolvability** and **Optimality Proof System**

Compilation to Unsolvability

1. Run task: Find optimal cost oc
2. Modify task: Require maximal cost $oc - 1$

   $\rightarrow$ task is unsolvable
3. Run modified task: Generate unsolvability certificate
4. Verify unsolvability certificate

   $\rightarrow$ oc is optimal cost

Technical Implementation

```
(: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)))
 )
```

Unsolvability Proof System

Reason about sets of dead states

Identify "dead" areas of state space (sets of dead states)

Derive new knowledge about other sets of dead states

$\rightarrow$ If all successors of S are dead, then S is dead

Optimality Proof System

Reason about sets of states with lower bound x

Definition

We say state set S_x **has a lower cost bound of** x, denoted by $gc(S_x) \geq x$, if all states in S_x need at least cost x to reach any goal state.

$\rightarrow$ If all successors of S have at least goal cost x then $gc(S) \geq x + 1$ (unit cost)

General Idea

1. Compute optimal cost oc
2. Iteratively create sets of states

   with at least cost $0, \ldots, oc$ to goal
3. If I in S_{oc}, where $gc(S_{oc}) \geq oc$

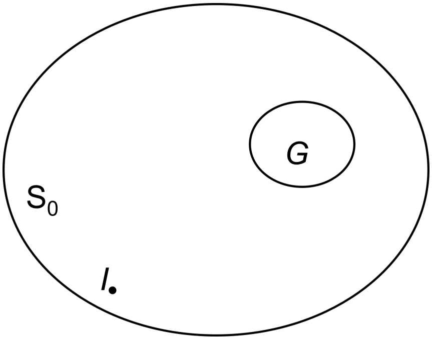
   $\rightsquigarrow$ task has optimal cost oc

General Idea

1. Compute optimal cost oc
2. Iteratively create sets of states

   with at least cost $0, \ldots, oc$ to goal
3. If I in S_{oc}, where $gc(S_{oc}) \geq oc$

   $\rightsquigarrow$ task has optimal cost oc

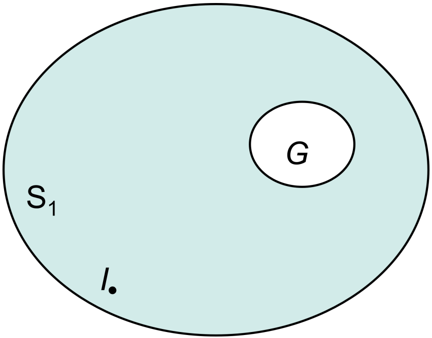


General Idea

1. Compute optimal cost oc
2. Iteratively create sets of states

   with at least cost $0, \dots, oc$ to goal
3. If I in S_{oc}, where $gc(S_{oc}) \geq oc$

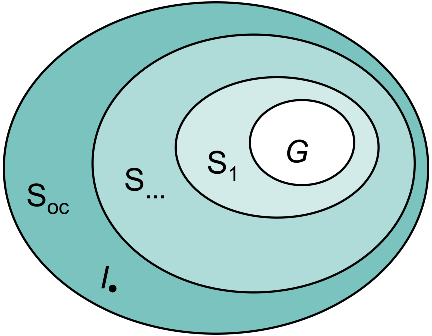
   $\rightsquigarrow$ task has optimal cost oc



General Idea

1. Compute optimal cost oc
2. Iteratively create sets of states with at least cost $0, \dots, oc$ to goal
3. If I in S_{oc}, where $gc(S_{oc}) \geq oc$

   $\rightsquigarrow$ task has optimal cost oc



Technical Implementation

Gain knowledge about goal distances and create sets $S_0, \ldots, S_{oc}$

Use g-value for **expanded** states

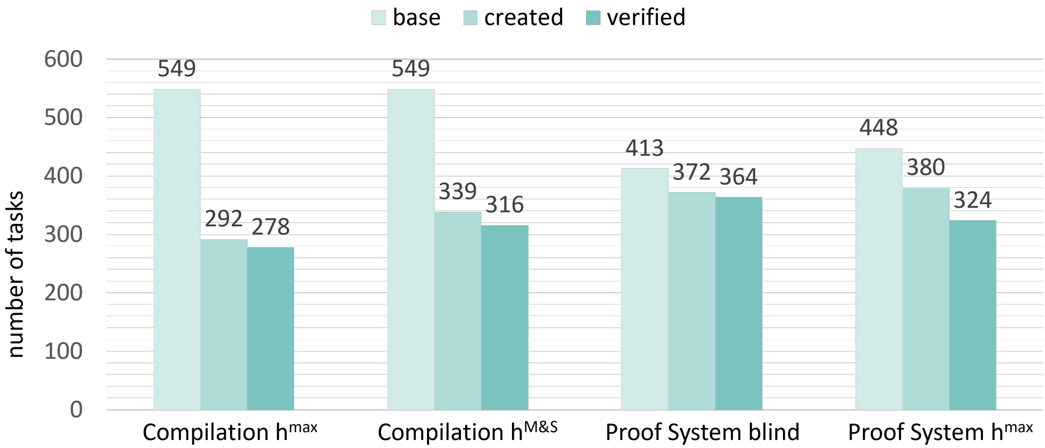
$\rightarrow$ state s with $g(s) \leq oc - x$ in set S_x

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

$\rightarrow$ state s with $h(s) \geq x$ is in set S_x

$\rightarrow$ prove separately for each heuristic (proved for h^{max})

Coverage


base
created
verified
600
500
400
300
200
100
0
number of tasks
549
292
278
549
339
316
413
372
364
448
380
324
Compilation h^{max}
Compilation $h^{M\&S}$
Proof System blind
Proof System h^{max}

Conclusion

Compilation to Unsolvability

- Generally applicable
- Compilation not verified
- Efficient generation not guaranteed

Conclusion

Compilation to Unsolvability

- Generally applicable
- Compilation not verified
- Efficient generation not guaranteed

Optimality Proof System

- Only applicable for concrete algorithms but expandable
- Efficient generation/verification depends on algorithm
- Overhead, especially for heuristic search
- Generally better results

Future work

Expand proof system to additional algorithms

Efficiently represent reasoning for different formalisms
(adapt from unsolvability proof system)

Combine knowledge of different heuristics for proof system

Thank you for
your attention!