Pseudo-Boolean Proof Logging for Optimal Planning

Simon Dold Malte Helmert Jakob Nordström
Gabriele Röger Tanja Schindler

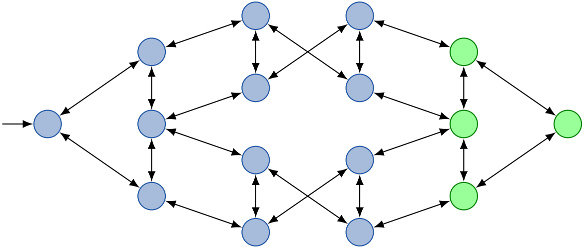
University of Basel
University of Copenhagen and Lund University

ICAPS 2025

Introduction

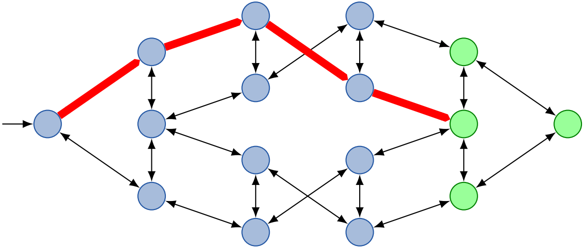
Optimal Planning

Optimal Planning
Given: a planning task Π
Output: "$\langle a_1, \ldots, a_n \rangle$ is a plan for Π with minimal cost", or "no plan for Π exists".



Optimal Planning

Optimal Planning
Given: a planning task Π
Output: "$\langle a_1, \ldots, a_n \rangle$ is a plan for Π with minimal cost",
or "no plan for Π exists".

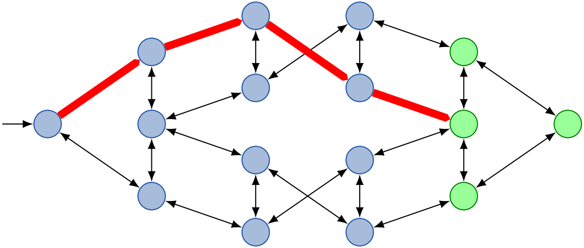


Optimal Planning

Optimal Planning

Given: a planning task Π

Output: "$\langle a_1, \ldots, a_n \rangle$ is a plan for Π with minimal cost ",
or "no plan for Π exists".



Optimal Planning Algorithms

Promises:

- the produced action sequence is a plan for the input task
- no cheaper plan exists for the input task
- any task reported as unsolvable actually is

Optimal Planning Algorithms

Promises:

- the produced action sequence is a plan for the input task
- no cheaper plan exists for the input task
- any task reported as unsolvable actually is

Trust:

- should we blindly trust the result?
- what do other communities do?

SAT algorithms

Example:
Is the there a variable assignement that satisfies
$(\bar{x} \vee y) \wedge (x \vee y) \wedge (\bar{y})$?
Promises:

- the produced variable assignment satisfies the input formula
- any formula reported as unsatisfiable actually is

Trust:

- should we blindly trust the result?
  - No. Use certifying algorithms.

Certifying Algorithms

Used in SAT competitions

- A proof is written during the SAT-solver run to certify that the formula is UNSAT
- This proof is evaluated by a formally verified checker
  $\rightsquigarrow$ Trust in the solution

Certifying Planning Algorithms

Proofs for planner outputs

- "$\langle a_0, \ldots, a_n \rangle$ is a plan for Π"
  $\rightsquigarrow$ The plan is the proof. Use validator (e.g., VAL, INVAL)
- "no plan for Π exists"
  $\rightsquigarrow$ unsolvability certificate[1]
- "$\langle a_0, \ldots, a_n \rangle$ is a plan for Π with minimal cost"
  $\rightsquigarrow$ lower-bound certificate[2] (and validate plan)

[1] Salomé Eriksson. *Certifying Planning Systems: Witnesses for Unsolvability* (Ph.D. Thesis 2019)

[2] Esther Mugdan, Remo Christen and Salomé Eriksson. *Optimality Certificates for Classical Planning* (ICAPS 2023)

Certifying Planning Algorithms

Proofs for planner outputs

- "$\langle a_0, \ldots, a_n \rangle$ is a plan for Π"
  $\rightsquigarrow$ The plan is the proof. Use validator (e.g., VAL, INVAL)
- "no plan for Π exists"
  $\rightsquigarrow$ unsolvability certificate[1]
- "$\langle a_0, \ldots, a_n \rangle$ is a plan for Π with minimal cost"
  $\rightsquigarrow$ lower-bound certificate[2] (and validate plan)

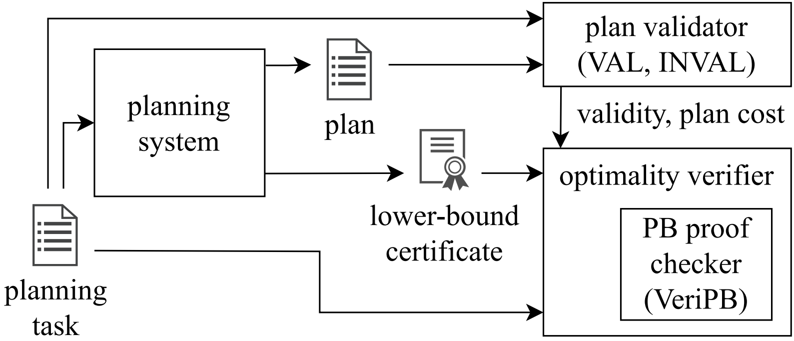
We use a different language/ proof system than previous approaches.

- Use an established checker $\rightsquigarrow$ VeriPB
- Be more fine-grained with the arguments

[1] Salomé Eriksson. *Certifying Planning Systems: Witnesses for Unsolvability* (Ph.D. Thesis 2019)

[2] Esther Mugdan, Remo Christen and Salomé Eriksson. *Optimality Certificates for Classical Planning* (ICAPS 2023)

Certified Optimal Planning



Lower-Bound Certificates

Lower-Bound Certificates

There is no plan with cost lower than B iff there is a property φ over state-cost pairs that

1. holds for the initial state with cost 0
2. is inductive under action applications (a.k.a. an invariant)
3. and does not hold for a goal state with cost lower than B

Lower-bound certificate: $\varphi +$ proofs for (1)–(3)

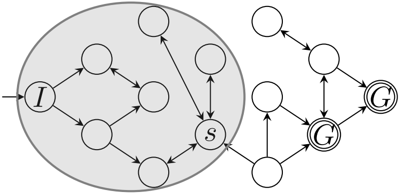
Lower-Bound Certificates

There is no plan with cost lower than B iff there is a property φ over state-cost pairs that

1. holds for the initial state with cost 0
2. is inductive under action applications (a.k.a. an invariant)
3. and does not hold for a goal state with cost lower than B

Lower-bound certificate: φ + proofs for (1)–(3)

$\rightsquigarrow$ corresponds to separating set for unsolvability



Certifying Optimality based on Pseudo-Boolean Constraints

- during search:
  - log representation of invariant φ
  - log proofs for the three properties (initial state, goal, inductivity)
- verification after search:
  - encode planning semantics
  - invariant and proof
  - use VeriPB to check proof

Everything in the language of VeriPB $\rightsquigarrow$ 0/1 integer linear programming constraints a.k.a. pseudo-Boolean constraints.

Pseudo-Boolean Encoding of Planning Semantics - Part I

Given: STRIPS planning task $\Pi = \langle V, I, G, A \rangle$
Encoding: (similar to SAT encoding with horizon 1)

- Boolean state variables V:
  Boolean variables V, cost variables $V_c = \{c_0, \ldots, \lceil \log_2 B \rceil\}$, and copies V', V'_c
- initial state $I \subseteq V$:

$$r_I \Leftrightarrow \bigwedge_{v \in I} v \wedge \bigwedge_{v \in V \setminus I} \bar{v}$$

- goal $G \subseteq V$:

$$r_G \Leftrightarrow \bigwedge_{v \in G} v$$

Pseudo-Boolean Encoding of Planning Semantics - Part II

- actions $a \in A$ with preconditions $\mathit{pre}(a) \subseteq V$, add effects $\mathit{add}(a) \subseteq V$, delete effects $\mathit{del}(a) \subseteq V$, cost $\mathit{cost}(a) \in \mathbb{N}_0$:

$$r_a \Rightarrow \bigwedge_{v \in \mathit{pre}(a)} v \wedge \bigwedge_{v \in \mathit{add}(a)} v' \wedge \bigwedge_{v \in \mathit{del}(a)} \overline{v'} \wedge \bigwedge_{v \in V \setminus \mathit{evars}(a)} eq_{v,v'} \wedge \Delta c^{=\mathit{cost}(a)}$$

where (here the pseudo-Boolean encoding is very useful)

$$\Delta c^{=k} \Leftrightarrow \sum_{i=0}^{\lceil \log_2 B \rceil} 2^i c'_i - \sum_{i=0}^{\lceil \log_2 B \rceil} 2^i c_i = k$$

- transition relation:

$$r_T \Leftrightarrow \bigvee_{a \in A} r_a$$

Pseudo-Boolean Lower Bound Certificates

Lower-bound certificate for Π with bound B:

- PB circuit representing invariant φ based on variables V, V_c:

$$
\begin{aligned}
r_0 &:\Leftrightarrow C(V, V_c) \\
&\dots \\
r_n &:\Leftrightarrow C(V, V_c, r_0, \dots, r_{n-1}) \\
r_\varphi &:\Leftrightarrow C(V, V_c, r_0, \dots, r_{n-1}, r_n)
\end{aligned}
$$

- VeriPB proof for initial state lemma $(r_I \wedge \mathit{cost}_{=0}) \Rightarrow r_\varphi$
- VeriPB proof for goal lemma $(r_G \wedge r_\varphi) \Rightarrow \mathit{cost}_{\geq B}$
- VeriPB proof for inductivity lemma $(r_\varphi \wedge r_T) \Rightarrow r'_\varphi$

Note: VeriPB proof contains two synchronized copies (unprimed+primed) of the circuit reifications (and some proof parts)

Proof Logging Heuristic Search

Proof Logging A*

- each expanded state occurs explicitly in the invariant
  - log reification constraint representing state-cost pairs:

$$r_{s,g} \Leftrightarrow \bigwedge_{v \in s} v \wedge \bigwedge_{v \in V \setminus s} \bar{v} \wedge \mathit{cost}_{\geq g}$$

- for states not expanded, heuristic must prove that a plan containing this state would be too expensive, i.e., $g(s) + h(s) \geq B$
  - provide heuristic certificate with invariant r_s^h

Proof Logging A*

- each expanded state occurs explicitly in the invariant
  - log reification constraint representing state-cost pairs:

$$r_{s,g} \Leftrightarrow \bigwedge_{v \in s} v \wedge \bigwedge_{v \in V \setminus s} \bar{v} \wedge \mathit{cost}_{\geq g}$$

- for states not expanded, heuristic must prove that a plan containing this state would be too expensive, i.e., $g(s) + h(s) \geq B$
  - provide heuristic certificate with invariant r_s^h
- final invariant:

$$r_\varphi \Leftrightarrow \bigvee_{\langle s,g \rangle \in \mathit{Closed}} r_{s,g} \vee \bigvee_{\langle s,g,h \rangle \in \mathit{Open}} r_s^h$$

In the paper we showed how to generate a lower-bound certificate with only a constant-factor overhead.

Heuristic Certificates

heuristic certificate for state s: Basically lower-bound certificate starting from s for bound $h(s)$!

- PB circuit defining r_s^h

$$r_0 \Leftrightarrow C(V, V_c) \quad \dots \quad r_n \Leftrightarrow C(V, V_c, r_0, \dots, r_{n-1})$$

$$r_s^h \Leftrightarrow C(V, V_c, r_0, \dots, r_n)$$

- VeriPB proof for state lemma $(r_s \wedge \mathit{cost}_{\geq B-h(s)}) \Rightarrow r_s^h$
- VeriPB proof for goal lemma $(r_G \wedge r_s^h) \Rightarrow \mathit{cost}_{\geq B}$
- VeriPB proof for inductivity lemma $(r_s^h \wedge r_T) \Rightarrow r_s^{h\prime}$

In the paper we showed how to efficiently generate a heuristic certificate for h^{max} and PDBs.

Take-Away

Take-Away

- We provide definition and encoding for optimality certificates in planning.
- Certificate specifies a seperating set and shows that it
  - contains the initial state,
  - contains no goal state with cost below the plan cost,
  - no action application leads out of it.
- This can be done efficiently for A* with h^{max} or PDBs.