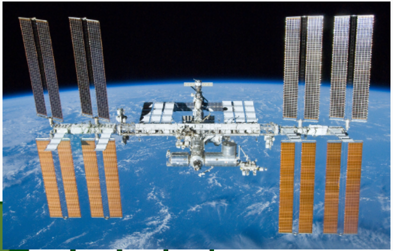
Dissecting Scorpion:
Ablation Study of an Optimal Classical Planner

Jendrik Seipp

October 23, 2024

Machine Reasoning Lab
Linköping University

AI Planning



Given a compact task model, find a sequence of actions that achieves the goal.

Optimal Classical Planning

setting:

- deterministic
- fully observable
- cost-optimal plans

main approaches:

- (explicit) A^* search [e.g., Helmert et al. 2008]
- symbolic search [e.g., Edelkamp et al. 2015]
- planning as SAT/maxSAT/QBF/CP [e.g., Rintanen 2012]

related fields:

- path finding, model checking, petri nets, ...

Scorpion: Optimal Classical Planner

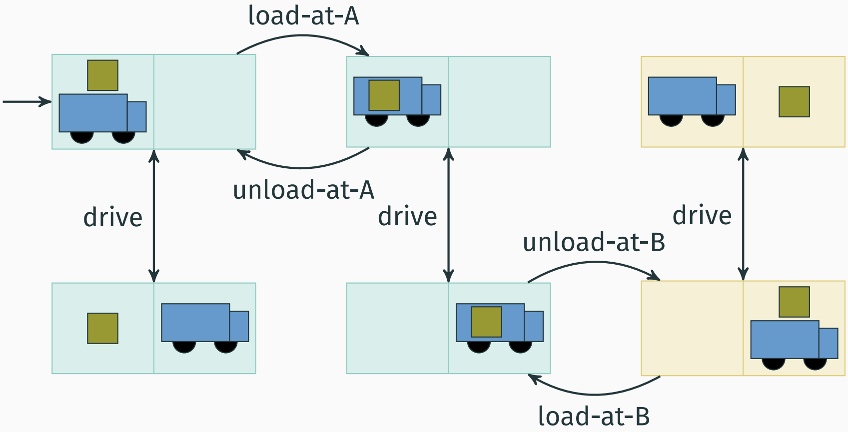
- winner of IPC 2023
- A^* search + admissible heuristic
  - abstractions: pattern databases and Cartesian abstractions
  - cost partitioning: saturated cost partitioning
- ECAI 2024 paper: literature overview and ablation study

Outline

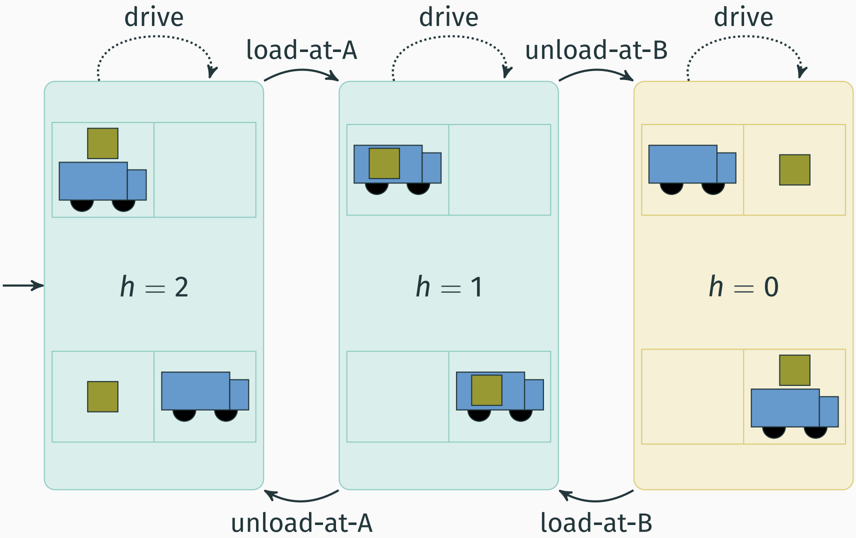
- Abstractions
- Cost Partitioning

Abstractions

Example Task



Example Abstraction

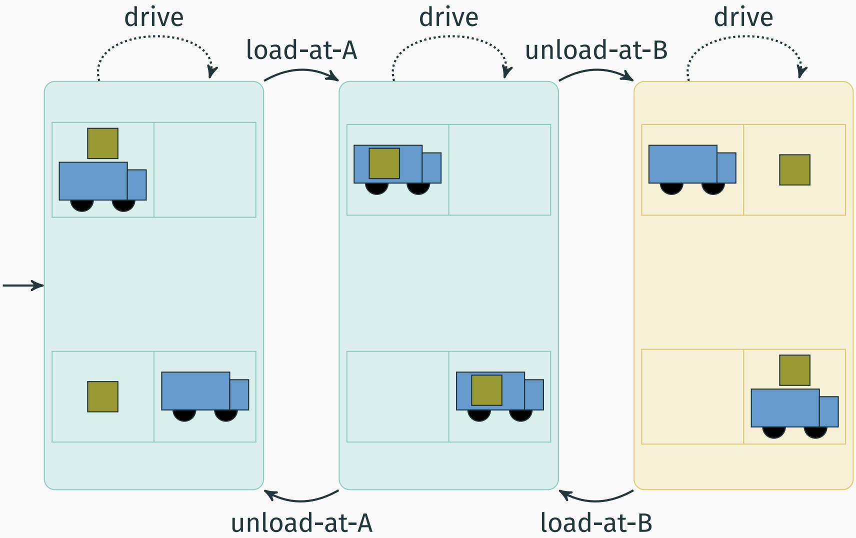


Abstraction Heuristics

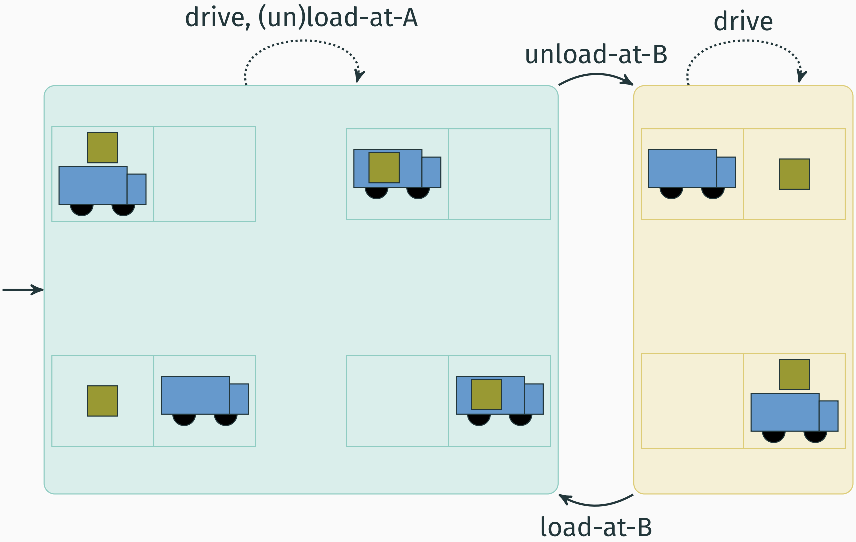
- abstractions preserve all plans $\rightarrow$ admissible
- higher accuracy $\rightarrow$ better guidance
- granularity vs. resource usage

$\rightarrow$ four main classes of abstractions

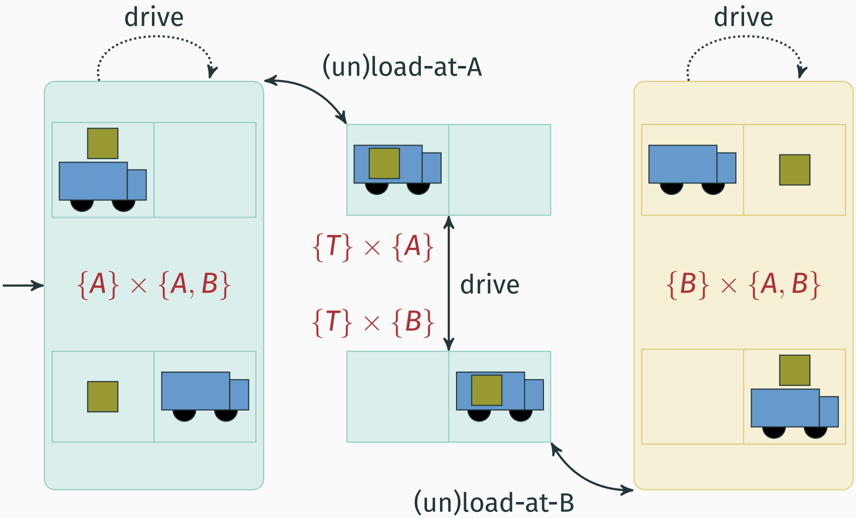
Projection (Pattern Database)



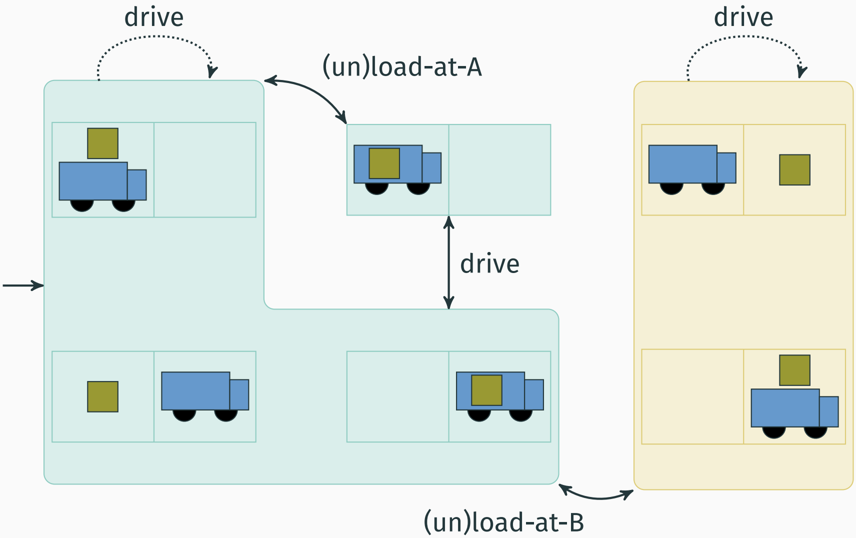
Domain Abstraction



Cartesian Abstraction



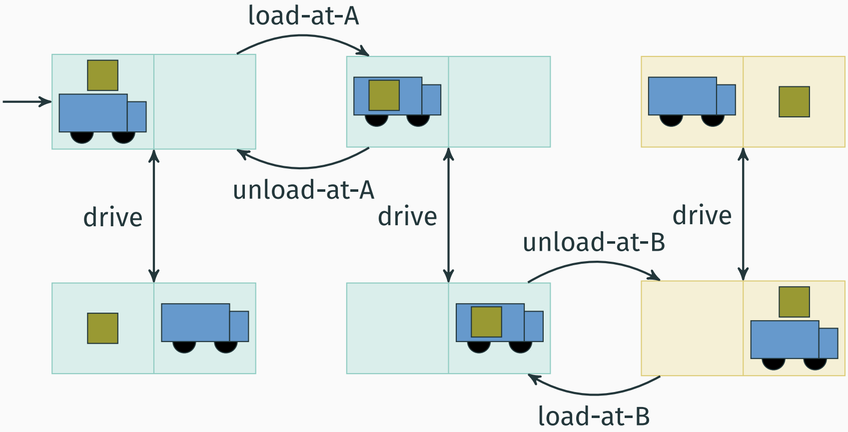
Merge-and-shrink Abstraction



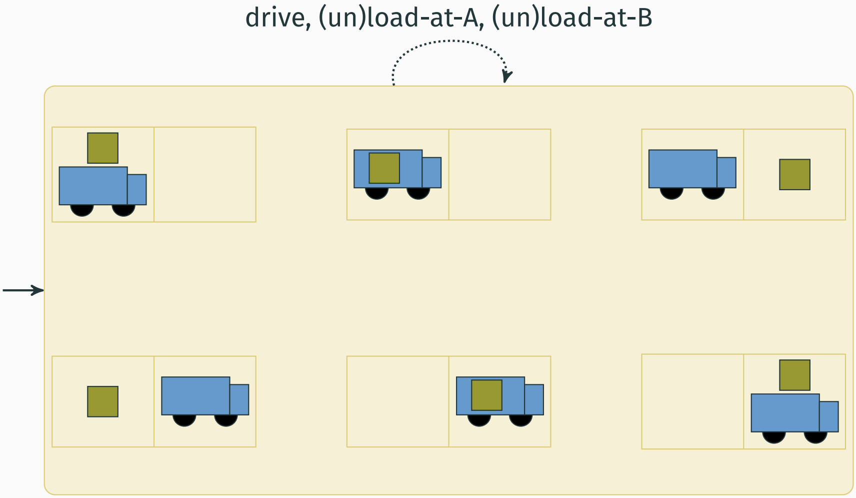
How to Generate an Abstraction?

- Projections (Pattern Databases)
  - bin packing [Edelkamp 2001]
  - genetic algorithms [Edelkamp 2006]
  - hill climbing [Haslum et al. 2007]
  - systematic [Pommerening et al. 2013]
  - CPC [Franco et al. 2017]
  - CEGAR [Rovner et al. 2019]
  - sys-SCP [Seipp 2019]
- Domain Abstractions
  - CEGAR [Kreft et al. 2023]
- Cartesian Abstractions
  - CEGAR [Pozo et al. 2024a; Seipp and Helmert 2018]
- Merge-and-shrink Abstractions
  - operations on factors [Sievers and Helmert 2021]

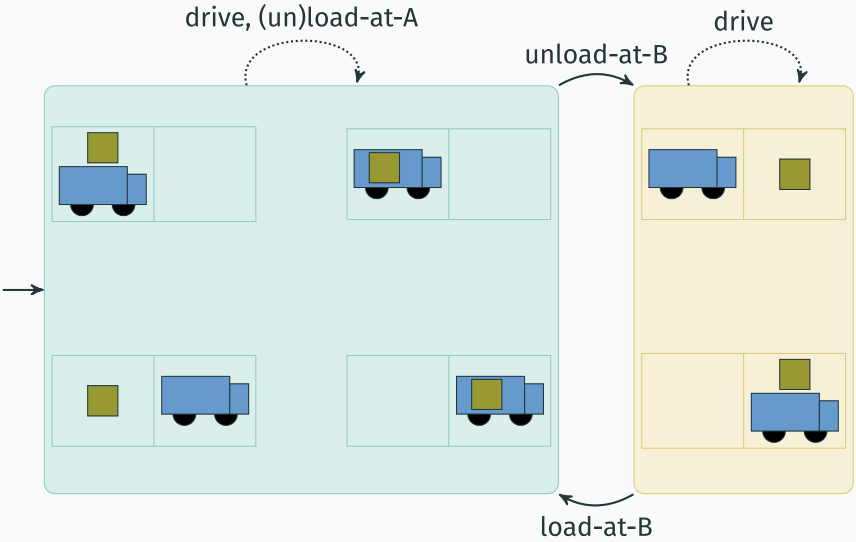
Cartesian CEGAR: Concrete Transition System



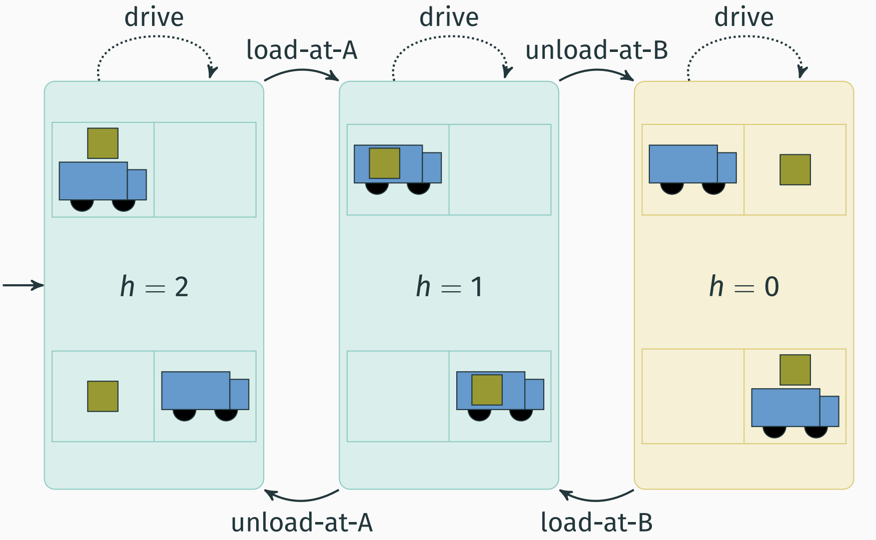
Cartesian CEGAR: Trivial Abstraction



Cartesian CEGAR: First Refinement



Cartesian CEGAR: Second Refinement



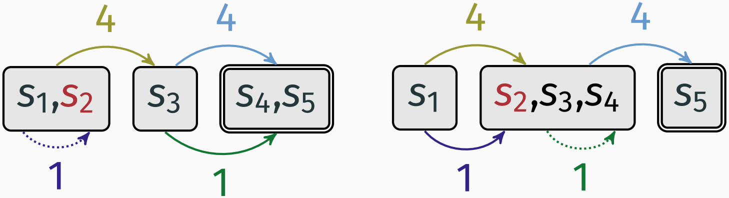
Multiple Abstractions

Generating Multiple Heuristics

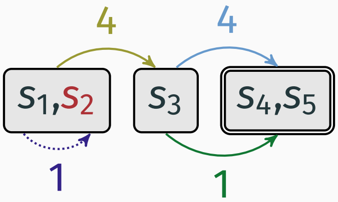
- Pattern databases: use different patterns ([P1], [P1, T], [P2, T], ...)
- Domain/Cartesian abstractions: one abstraction per goal fact (P1=B, P2=C, ...)
- Merge-and-shrink: different merge/shrink strategies

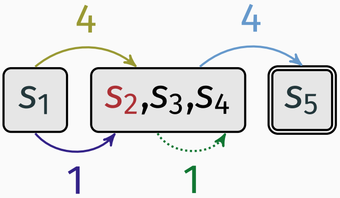
Combining Multiple Heuristics

Combining Multiple Heuristics



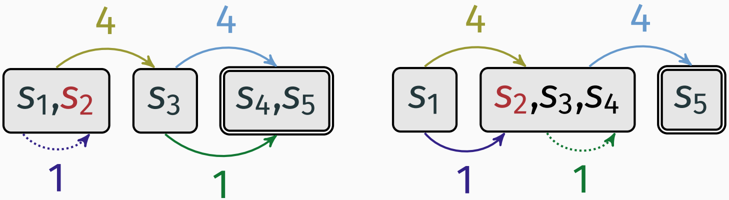
Combining Multiple Heuristics





$h_1(s_2) = 5$ $h_2(s_2) = 4$

Combining Multiple Heuristics



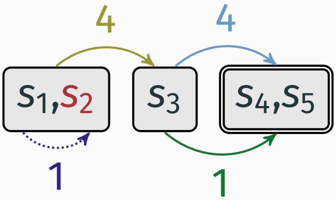
$h_1(s_2) = 5$

$h_2(s_2) = 4$

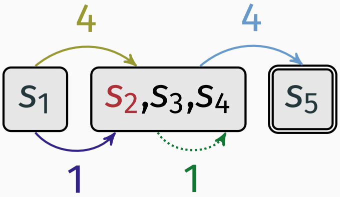
maximize over estimates:

- $h(s_2) = 5$

Combining Multiple Heuristics



$h_1(s_2) = 5$



$h_2(s_2) = 4$

maximize over estimates:

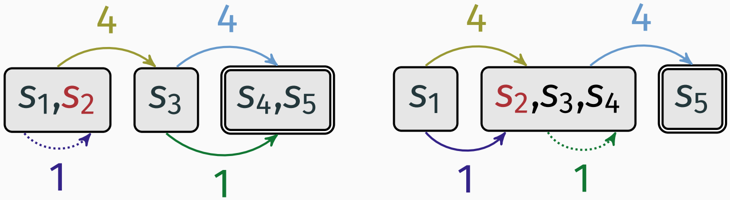
- $h(s_2) = 5$
- only selects best heuristic
- does not combine heuristics

Cost Partitioning

Multiple Heuristics: Cost Partitioning

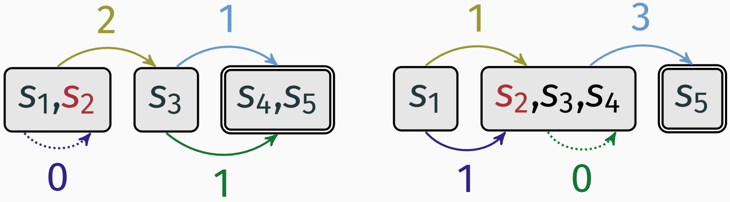
Cost Partitioning

- split operator costs among heuristics
- sum of costs must not exceed original cost



Cost Partitioning

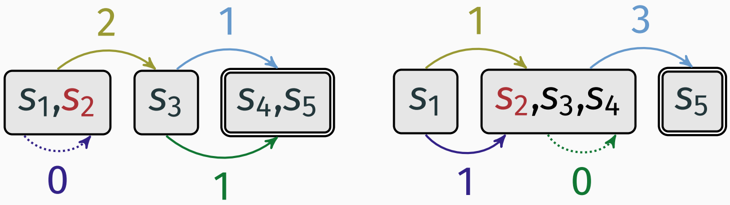
- split operator costs among heuristics
- sum of costs must not exceed original cost



Multiple Heuristics: Cost Partitioning

Cost Partitioning

- split operator costs among heuristics
- sum of costs must not exceed original cost



$$h(s_2) = 3 + 3 = 6$$

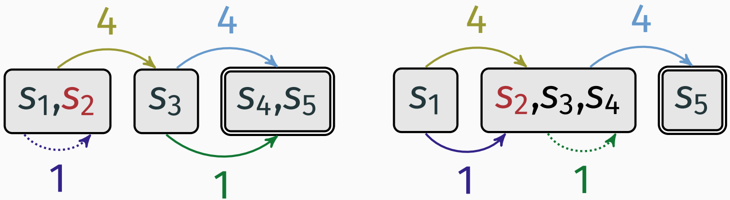
Optimal Cost Partitioning

Linear Program for state s [Katz and Domshlak 2010; Pommerening et al. 2015]

$$\text{maximize } \sum_i h_i(s) \text{ subject to}$$

$$\sum_i c_i(o) \leq cost(o) \quad \text{for all operators } o$$

$$h_i = \text{heuristic } i \text{ under cost } c_i \quad \text{for all heuristics } i$$



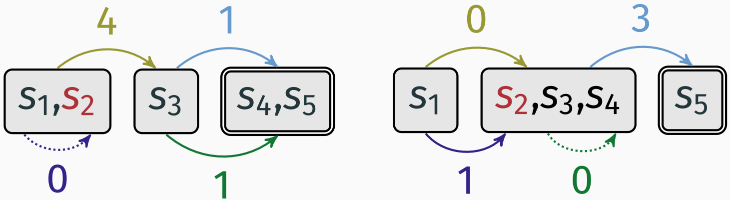
Optimal Cost Partitioning

Linear Program for state s [Katz and Domshlak 2010; Pommerening et al. 2015]

$$\text{maximize} \sum_i h_i(s) \text{ subject to}$$

$$\sum_i c_i(o) \leq cost(o) \quad \text{for all operators } o$$

$$h_i = \text{heuristic } i \text{ under cost } c_i \quad \text{for all heuristics } i$$



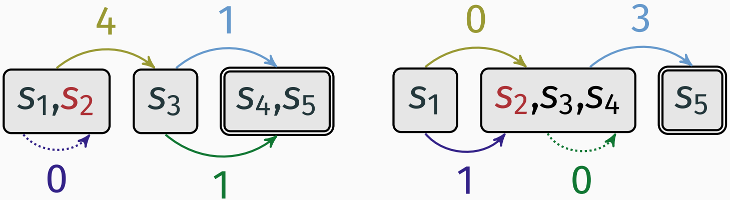
Optimal Cost Partitioning

Linear Program for state s [Katz and Domshlak 2010; Pommerening et al. 2015]

$$\text{maximize} \sum_i h_i(s) \text{ subject to}$$

$$\sum_i c_i(o) \leq cost(o) \quad \text{for all operators } o$$

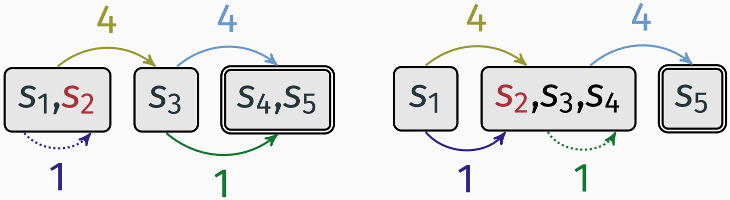
$$h_i = \text{heuristic } i \text{ under cost } c_i \quad \text{for all heuristics } i$$



$$h(s_2) = 5 + 3 = 8$$

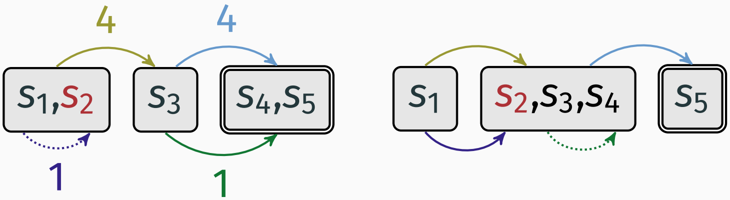
SCP Algorithm [Seipp et al. 2020a]

- order heuristics, then for each heuristic *h*:
  - use minimum costs preserving all estimates of *h*
  - use remaining costs for subsequent heuristics



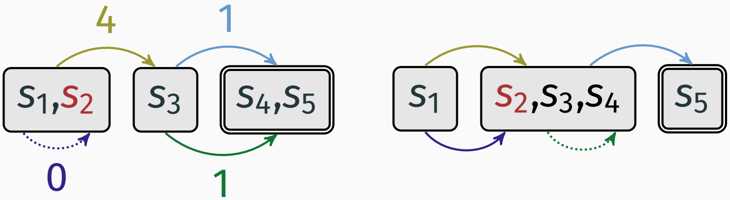
SCP Algorithm [Seipp et al. 2020a]

- order heuristics, then for each heuristic *h*:
  - use minimum costs preserving all estimates of *h*
  - use remaining costs for subsequent heuristics



SCP Algorithm [Seipp et al. 2020a]

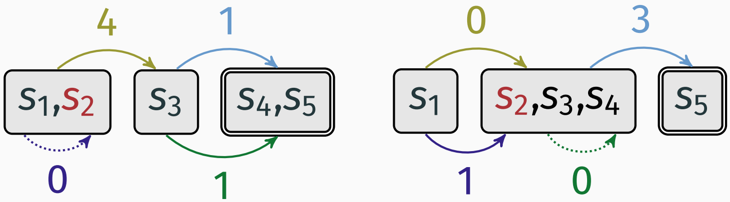
- order heuristics, then for each heuristic *h*:
  - use minimum costs preserving all estimates of *h*
  - use remaining costs for subsequent heuristics



Saturated Cost Partitioning

SCP Algorithm [Seipp et al. 2020a]

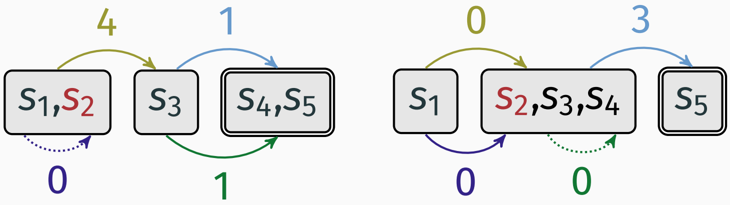
- order heuristics, then for each heuristic *h*:
  - use minimum costs preserving all estimates of *h*
  - use remaining costs for subsequent heuristics



Saturated Cost Partitioning

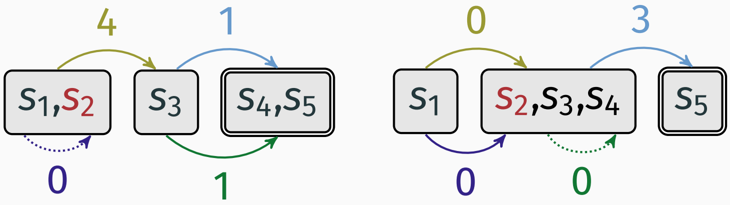
SCP Algorithm [Seipp et al. 2020a]

- order heuristics, then for each heuristic *h*:
  - use minimum costs preserving all estimates of *h*
  - use remaining costs for subsequent heuristics



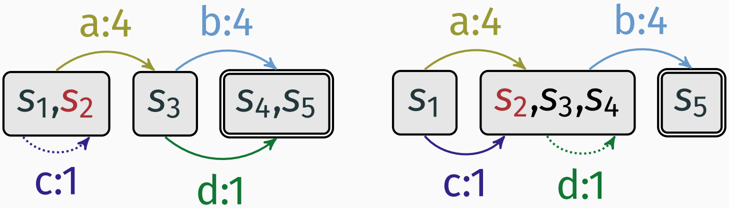
SCP Algorithm [Seipp et al. 2020a]

- order heuristics, then for each heuristic h:
  - use minimum costs preserving all estimates of h
  - use remaining costs for subsequent heuristics

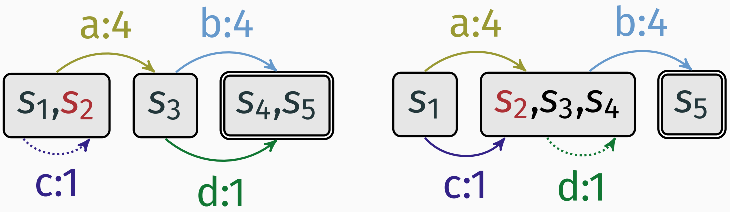


$$h(s_2) = 5 + 3 = 8$$

Post-hoc Optimization [Pommerening et al. 2013]

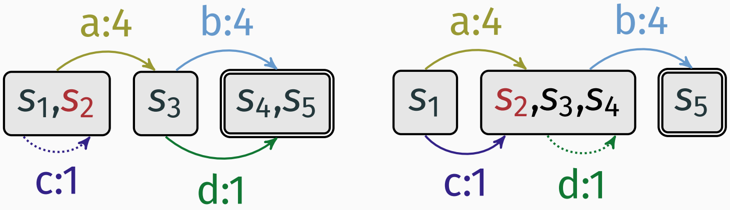


Post-hoc Optimization [Pommerening et al. 2013]



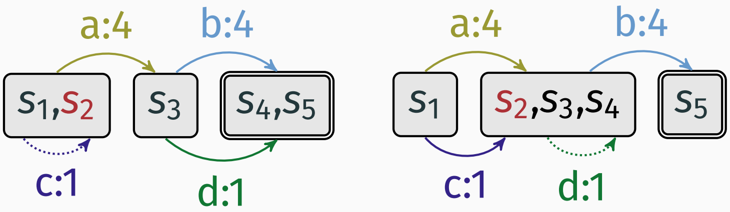
- a, b, d active $\quad h_1(s_2) = 5$

Post-hoc Optimization [Pommerening et al. 2013]



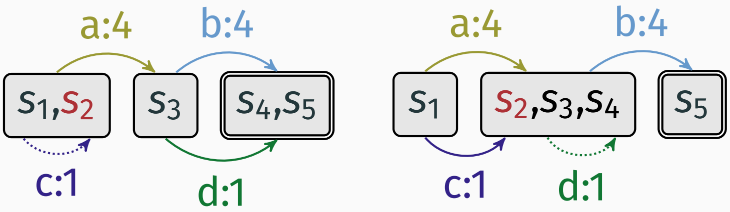
- a, b, d active $h_1(s_2) = 5 \quad \rightarrow \quad 4A + 4B + 1D \geq 5$

Post-hoc Optimization [Pommerening et al. 2013]



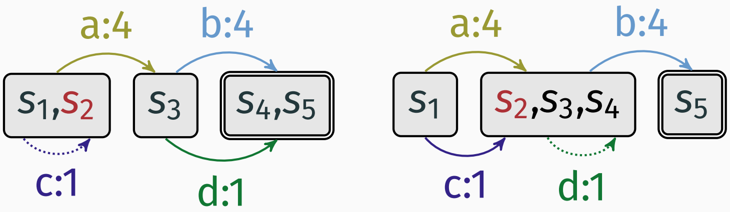
- a, b, d active $h_1(s_2) = 5 \quad \rightarrow \quad 4A + 4B + 1D \geq 5$
- a, b, c active $h_2(s_2) = 4$

Post-hoc Optimization [Pommerening et al. 2013]



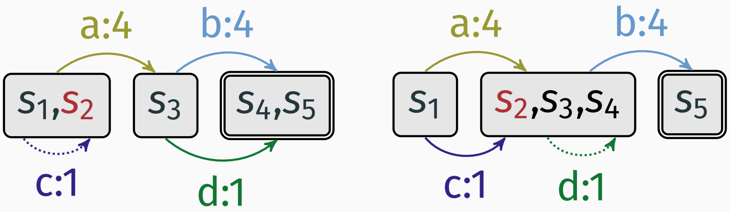
- a, b, d active $\quad h_1(s_2) = 5 \quad \rightarrow \quad 4A + 4B + 1D \geq 5$
- a, b, c active $\quad h_2(s_2) = 4 \quad \rightarrow \quad 4A + 4B + 1C \geq 4$

Post-hoc Optimization [Pommerening et al. 2013]



- a, b, d active $\quad h_1(s_2) = 5 \quad \rightarrow \quad 4A + 4B + 1D \geq 5$
- a, b, c active $\quad h_2(s_2) = 4 \quad \rightarrow \quad 4A + 4B + 1C \geq 4$
- $A \geq 0, B \geq 0, C \geq 0, D \geq 0$

Post-hoc Optimization [Pommerening et al. 2013]



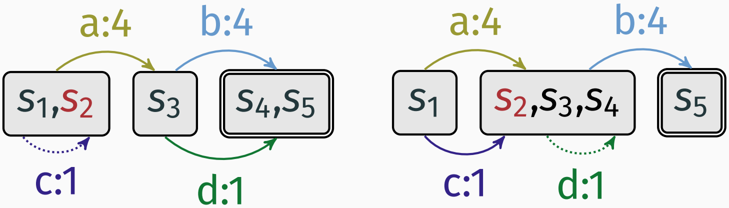
minimize $4A + 4B + 1C + 1D$ **such that**

$$4A + 4B + 1D \geq 5$$

$$4A + 4B + 1C \geq 4$$

- $A \geq 0, B \geq 0, C \geq 0, D \geq 0$

Post-hoc Optimization [Pommerening et al. 2013]



minimize $4A + 4B + 1C + 1D$ **such that**

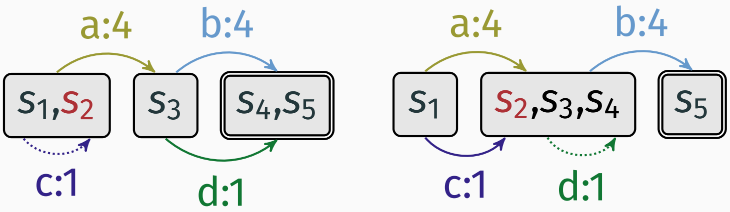
$$4A + 4B + 1D \geq 5$$

$$4A + 4B + 1C \geq 4$$

- $A \geq 0, B \geq 0, C \geq 0, D \geq 0$

$$h(s_2) = 5$$

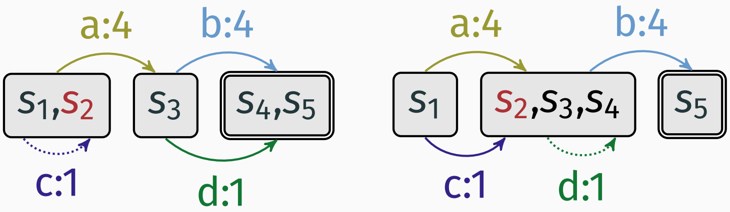
Saturated Post-hoc Optimization [Seipp et al. 2021]



minimize $4A + 4B + 1C + 1D$ **such that**

- $4A + 4B + 1D \geq 5$
- $4A + 4B + 1C \geq 4$
- $A \geq 0,\ B \geq 0,\ C \geq 0,\ D \geq 0$

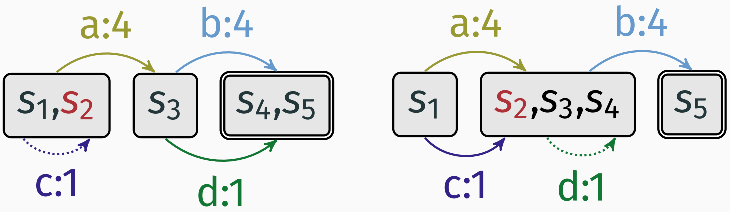
Saturated Post-hoc Optimization [Seipp et al. 2021]



minimize $4A + 4B + 1C + 1D$ **such that**

- $4A + 4B + 1D \geq 5$
- $4A + 4B + 1C \geq 4$
- $A \geq 0,\ B \geq 0,\ C \geq 0,\ D \geq 0$

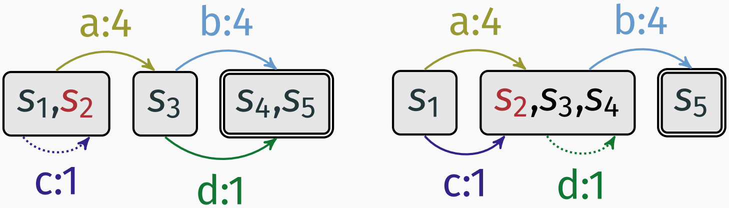
Saturated Post-hoc Optimization [Seipp et al. 2021]



minimize $4A + 4B + 1C + 1D$ **such that**

- $4A + 1B + 1D \geq 5$
- $4A + 4B + 1C \geq 4$
- $A \geq 0,\ B \geq 0,\ C \geq 0,\ D \geq 0$

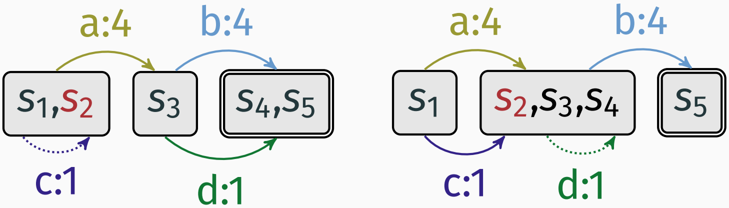
Saturated Post-hoc Optimization [Seipp et al. 2021]



minimize $4A + 4B + 1C + 1D$ **such that**

- $4A + 1B + 1D \geq 5$
- $4A + 4B + 1C \geq 4$
- $A \geq 0,\ B \geq 0,\ C \geq 0,\ D \geq 0$

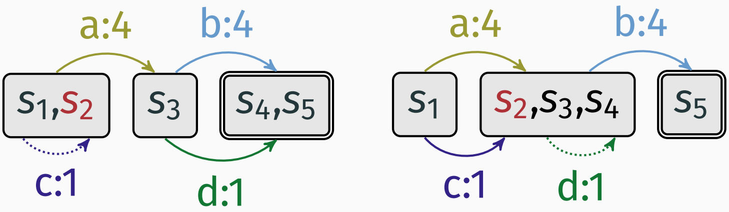
Saturated Post-hoc Optimization [Seipp et al. 2021]



minimize $4A + 4B + 1C + 1D$ **such that**

- $4A + 1B + 1D \geq 5$
- $4A + 4B + 1C \geq 4$
- $A \geq 0, B \geq 0, C \geq 0, D \geq 0$

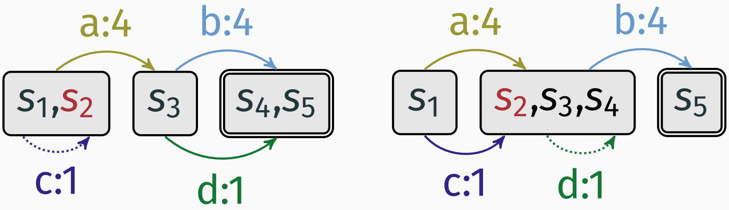
Saturated Post-hoc Optimization [Seipp et al. 2021]



minimize $4A + 4B + 1C + 1D$ **such that**

- $4A + 1B + 1D \geq 5$
- $1A + 4B + 1C \geq 4$
- $A \geq 0,\ B \geq 0,\ C \geq 0,\ D \geq 0$

Saturated Post-hoc Optimization [Seipp et al. 2021]



minimize $4A + 4B + 1C + 1D$ **such that**

- $4A + 1B + 1D \geq 5$
- $1A + 4B + 1C \geq 4$
- $A \geq 0, B \geq 0, C \geq 0, D \geq 0$

$$h(s_2) = 7.2$$

Theoretical Relationships

UCP

Uniform Cost Partitioning

distribute costs evenly among relevant heuristics

Theoretical Relationships

GZOCP

UCP

Greedy Zero-one Cost Partitioning

order heuristics and give full cost to first relevant heuristic

Theoretical Relationships

GZOCP

PhO

UCP

Post-hoc Optimization

Theoretical Relationships

GZOCP

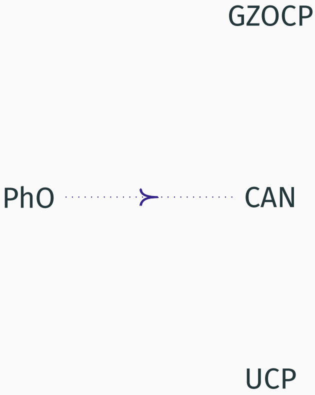
PhO CAN

UCP

Canonical Heuristic

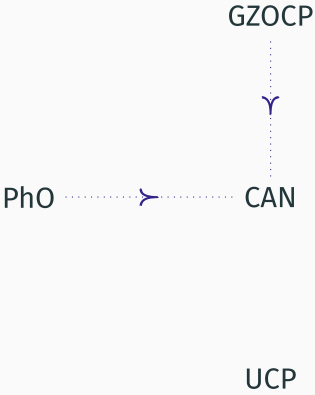
maximum over sums of independent heuristic subsets

Theoretical Relationships



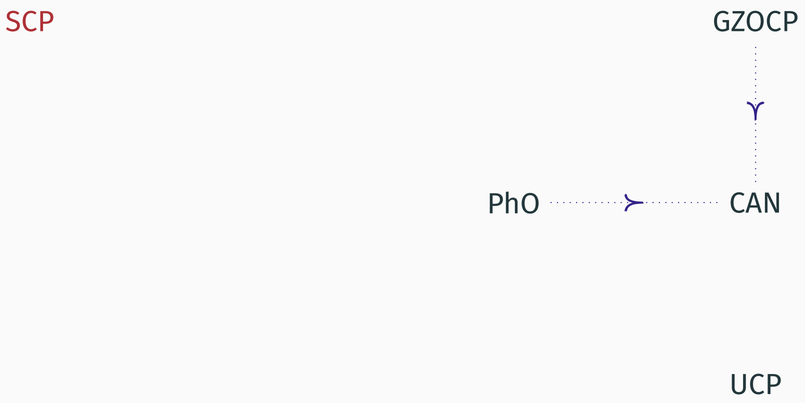
[Pommerening et al. 2013]

Theoretical Relationships

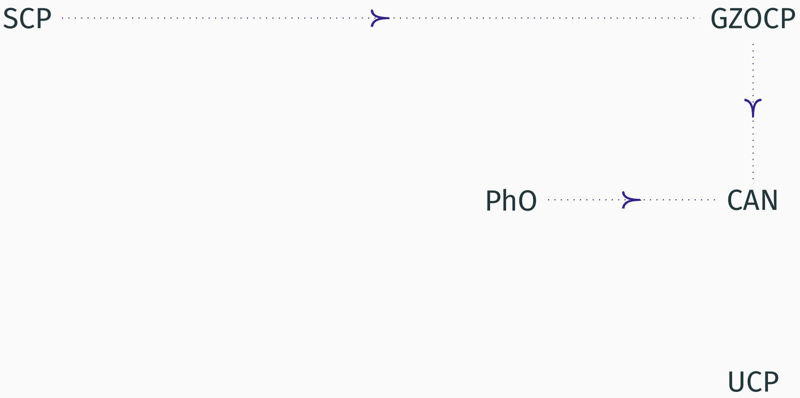


[Seipp et al. 2017]

Theoretical Relationships



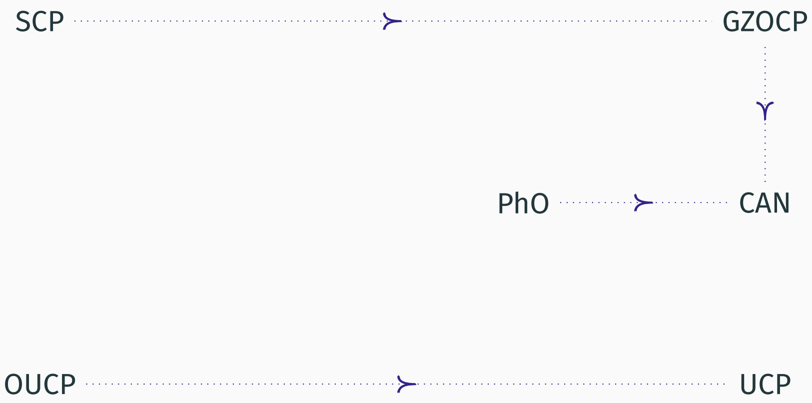
Theoretical Relationships



[Seipp et al. 2017]

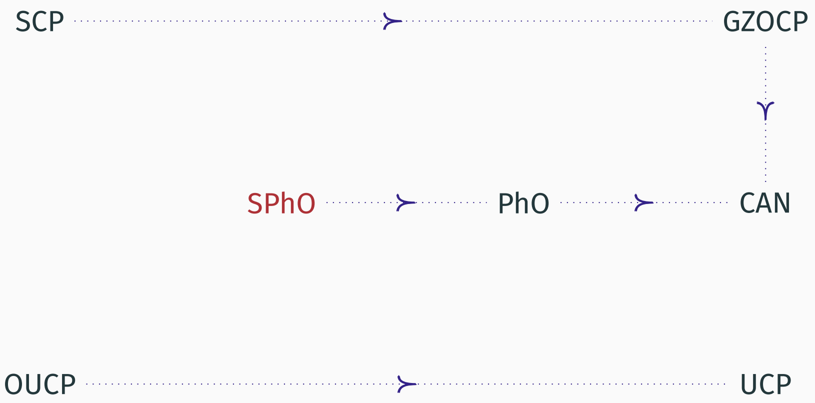
Theoretical Relationships

Theoretical Relationships



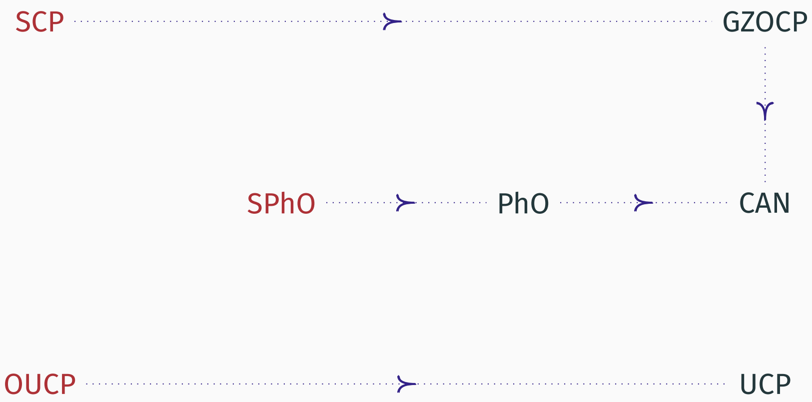
[Seipp et al. 2017]

Theoretical Relationships

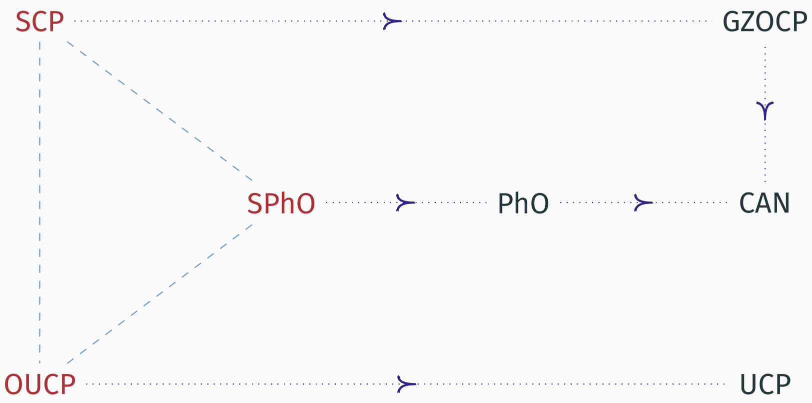


[Seipp et al. 2021]

Theoretical Relationships

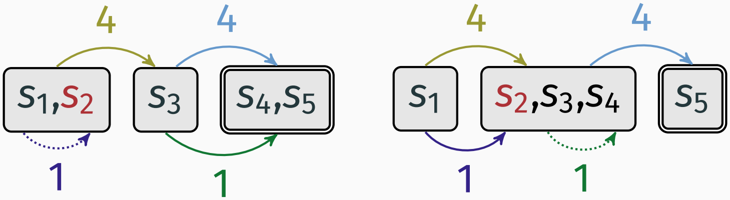


Theoretical Relationships



[Seipp et al. 2021]

SPhO vs. SCP



$h^{\mathrm{SCP}}_{\langle h_1,h_2\rangle}(s_2) = 8$ $\quad h^{\mathrm{SPhO}}(s_2) = 7.2$ $\quad h^{\mathrm{SCP}}_{\langle h_2,h_1\rangle}(s_2) = 7$

→ greedily optimize order for given state

→ use multiple orders for SCP

Latest Developments Around Scorpion

abstractions:

- incremental search for Cartesian CEGAR [Seipp et al. 2020b]
- better flaws for Cartesian CEGAR [Pozo et al. 2024a,b; Speck and Seipp 2022]
- compute Cartesian transitions on demand [Seipp 2024]
- abstraction heuristics for factored tasks [Büchner et al. 2024]

cost partitioning:

- use saturation to select patterns [Seipp 2019]
- compute orders for SCP during search [Seipp 2021]
- use Dantzig-Wolfe decomposition for OCP [Pommerening et al. 2021]
- use LP sensitivity analysis for SPhO [Höft et al. 2024, 2023]

Summary

abstractions:

- accurate heuristics
- fast to evaluate

cost partitioning:

- combine heuristics admissibly
- backbone of state-of-the-art optimal planners

Scorpion: `github.com/jendrikseipp/scorpion`
Contact: `mrlab.ai/jendrik-seipp`

References i

- Büchner, Clemens, Patrick Ferber, Jendrik Seipp, and Malte Helmert (2024). “Abstraction Heuristics for Factored Tasks”. In: *Proc. ICAPS 2024*, pp. 40–49.
- Edelkamp, Stefan (2001). “Planning with Pattern Databases”. In: *Proc. ECP 2001*, pp. 84–90.
- Edelkamp, Stefan (2006). “Automated Creation of Pattern Database Search Heuristics”. In: *Proc. MoChArt 2006*, pp. 35–50.
- Edelkamp, Stefan, Peter Kissmann, and Álvaro Torralba (2015). “BDDs Strike Back (in AI Planning)”. In: *Proc. AAAI 2015*, pp. 4320–4321.
- Franco, Santiago, Álvaro Torralba, Levi H. S. Lelis, and Mike Barley (2017). “On Creating Complementary Pattern Databases”. In: *Proc. IJCAI 2017*, pp. 4302–4309.
- Haslum, Patrik, Adi Botea, Malte Helmert, Blai Bonet, and Sven Koenig (2007). “Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal Planning”. In: *Proc. AAAI 2007*, pp. 1007–1012.
- Helmert, Malte, Patrik Haslum, and Jörg Hoffmann (2008). “Explicit-State Abstraction: A New Method for Generating Heuristic Functions”. In: *Proc. AAAI 2008*, pp. 1547–1550.
- Höft, Paul, David Speck, Florian Pommerening, and Jendrik Seipp (2024). “Versatile Cost Partitioning with Exact Sensitivity Analysis”. In: *Proc. ICAPS 2024*, pp. 276–280.

References ii

- Höft, Paul, David Speck, and Jendrik Seipp (2023). “Sensitivity Analysis for Saturated Post-hoc Optimization in Classical Planning”. In: *Proc. ECAI 2023*, pp. 1044–1051.
- Katz, Michael and Carmel Domshlak (2010). “Optimal admissible composition of abstraction heuristics”. In: *AIJ* 174.12–13, pp. 767–798.
- Kreft, Raphael, Clemens Büchner, Silvan Sievers, and Malte Helmert (2023). “Computing Domain Abstractions for Optimal Classical Planning with Counterexample-Guided Abstraction Refinement”. In: *Proc. ICAPS 2023*, pp. 221–226.
- Pommerening, Florian, Malte Helmert, Gabriele Röger, and Jendrik Seipp (2015). “From Non-Negative to General Operator Cost Partitioning”. In: *Proc. AAAI 2015*, pp. 3335–3341.
- Pommerening, Florian, Thomas Keller, Valentina Halasi, Jendrik Seipp, Silvan Sievers, and Malte Helmert (2021). “Dantzig-Wolfe Decomposition for Cost Partitioning”. In: *Proc. ICAPS 2021*, pp. 271–280.
- Pommerening, Florian, Gabriele Röger, and Malte Helmert (2013). “Getting the Most Out of Pattern Databases for Classical Planning”. In: *Proc. IJCAI 2013*, pp. 2357–2364.
- Pozo, Martín, Álvaro Torralba, and Carlos Linares López (2024a). “Gotta Catch ’Em All! Sequence Flaws in CEGAR for Classical Planning”. In: *Proc. ECAI 2024*, pp. 4287–4294.

References iii

- Pozo, Martín, Álvaro Torralba, and Carlos Linares López (2024b). "When CEGAR Meets Regression: A Love Story in Optimal Classical Planning". In: *Proc. AAAI 2024*, pp. 20238–20246.
- Rintanen, Jussi (2012). "Planning as Satisfiability: Heuristics". In: *AIJ* 193, pp. 45–86.
- Rovner, Alexander, Silvan Sievers, and Malte Helmert (2019). "Counterexample-Guided Abstraction Refinement for Pattern Selection in Optimal Classical Planning". In: *Proc. ICAPS 2019*, pp. 362–367.
- Seipp, Jendrik (2019). "Pattern Selection for Optimal Classical Planning with Saturated Cost Partitioning". In: *Proc. IJCAI 2019*, pp. 5621–5627.
- Seipp, Jendrik (2021). "Online Saturated Cost Partitioning for Classical Planning". In: *Proc. ICAPS 2021*, pp. 317–321.
- Seipp, Jendrik (2024). "Efficiently Computing Transitions in Cartesian Abstractions". In: *Proc. ICAPS 2024*, pp. 509–513.
- Seipp, Jendrik and Malte Helmert (2018). "Counterexample-Guided Cartesian Abstraction Refinement for Classical Planning". In: *JAIR* 62, pp. 535–577.
- Seipp, Jendrik, Thomas Keller, and Malte Helmert (2017). "A Comparison of Cost Partitioning Algorithms for Optimal Classical Planning". In: *Proc. ICAPS 2017*, pp. 259–268.

- Seipp, Jendrik, Thomas Keller, and Malte Helmert (2020a). “Saturated Cost Partitioning for Optimal Classical Planning”. In: *JAIR* 67, pp. 129–167.
- Seipp, Jendrik, Thomas Keller, and Malte Helmert (2021). “Saturated Post-hoc Optimization for Classical Planning”. In: *Proc. AAAI 2021*, pp. 11947–11953.
- Seipp, Jendrik, Samuel von Allmen, and Malte Helmert (2020b). “Incremental Search for Counterexample-Guided Cartesian Abstraction Refinement”. In: *Proc. ICAPS 2020*, pp. 244–248.
- Sievers, Silvan and Malte Helmert (2021). “Merge-and-Shrink: A Compositional Theory of Transformations of Factored Transition Systems”. In: *JAIR* 71, pp. 781–883.
- Speck, David and Jendrik Seipp (2022). “New Refinement Strategies for Cartesian Abstractions”. In: *Proc. ICAPS 2022*, pp. 348–352.