Online Saturated Cost Partitioning for Classical Planning

Jendrik Seipp

August 2021

Linköping University, University of Basel

Setting

- optimal classical planning
- A^* search + admissible heuristic
- multiple abstraction heuristics
- cost partitioning

Setting

- optimal classical planning
- A^* search + admissible heuristic
- multiple abstraction heuristics
- saturated cost partitioning

Problem

different states need different cost partitionings:

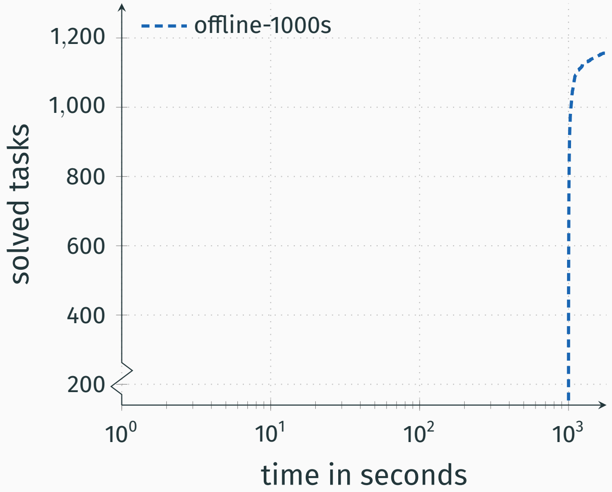
- precompute cost partitionings

→ no good stopping criterion, search starts late

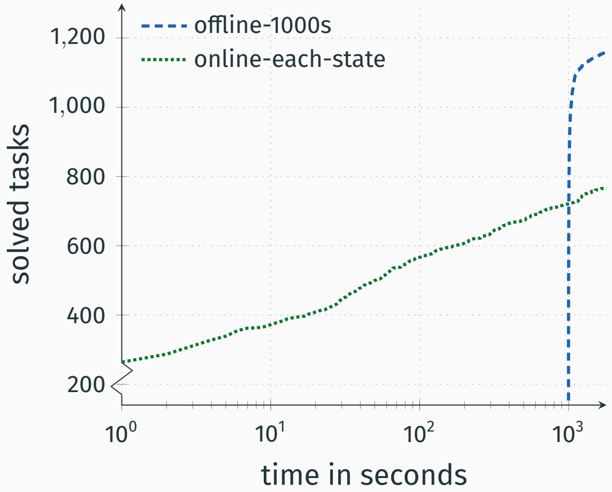
- compute cost partitioning for each state

→ too expensive

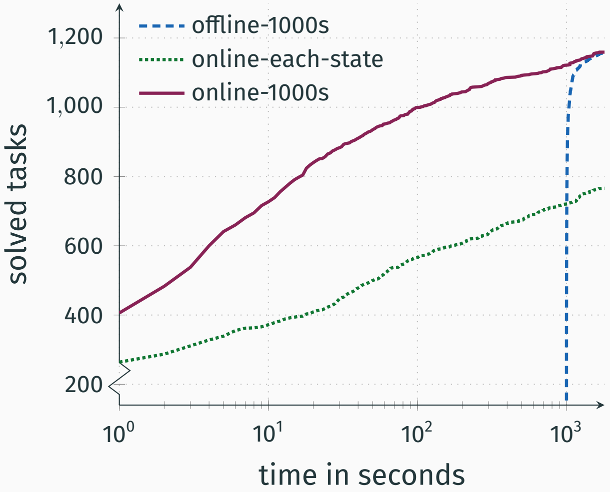
Coverage over time



Coverage over time



Coverage over time

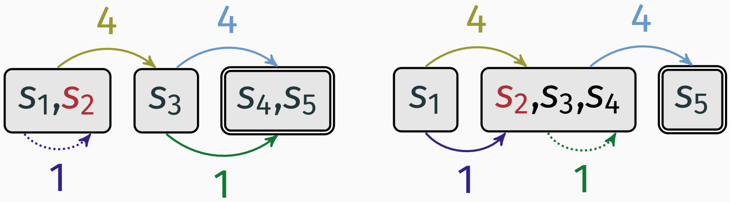


Cost partitioning

- split action costs among heuristics such that: sum of costs $\leq$ original cost

Saturated cost partitioning

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

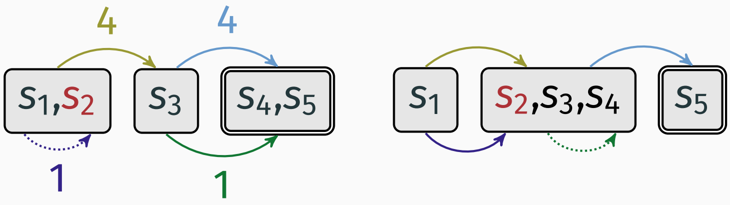


Cost partitioning

- split action costs among heuristics such that: sum of costs $\leq$ original cost

Saturated cost partitioning

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



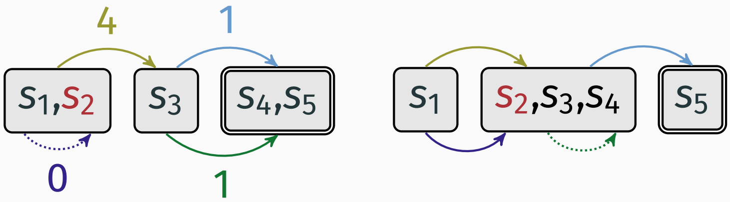
Background

Cost partitioning

- split action costs among heuristics such that: sum of costs $\leq$ original cost

Saturated cost partitioning

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



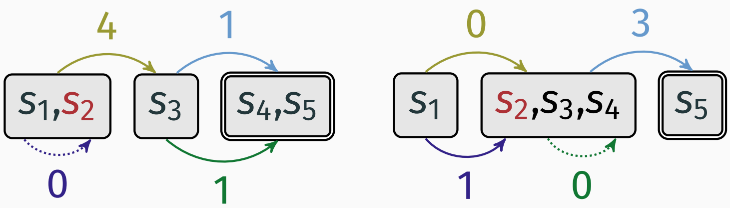
Background

Cost partitioning

- split action costs among heuristics such that: sum of costs $\leq$ original cost

Saturated cost partitioning

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

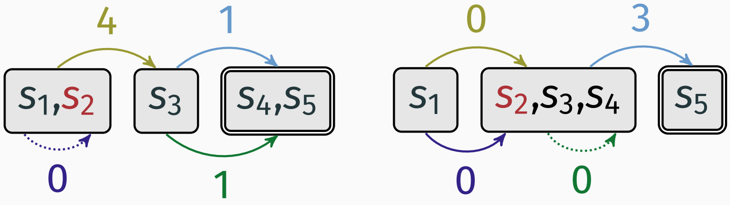


Cost partitioning

- split action costs among heuristics such that: sum of costs $\leq$ original cost

Saturated cost partitioning

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



Background

Order matters:

- $h^{\mathrm{SCP}}_{\rightarrow}(s_2) = 8$
- $h^{\mathrm{SCP}}_{\leftarrow}(s_2) = 7$

Background

Order matters:

- $h^{\text{SCP}}_{\rightarrow}(s_2) = 8$
- $h^{\text{SCP}}_{\leftarrow}(s_2) = 7$

$\rightarrow$ use multiple orders and maximize over estimates

Background

Offline diversification

- sample 1000 states
- start with empty set of orders
- until time limit is reached:
  - compute order for new sample
  - store order if a sample profits from it

Online diversification

ComputeHeuristic(s)

- if Select(s) and not time limit reached
  - compute order for s
  - store order if s profits from it
- return maximum over all stored orders for s

Offline vs. online diversification

Offline

- compute orders for samples for T seconds
- store order if one of 1000 samples profits from it

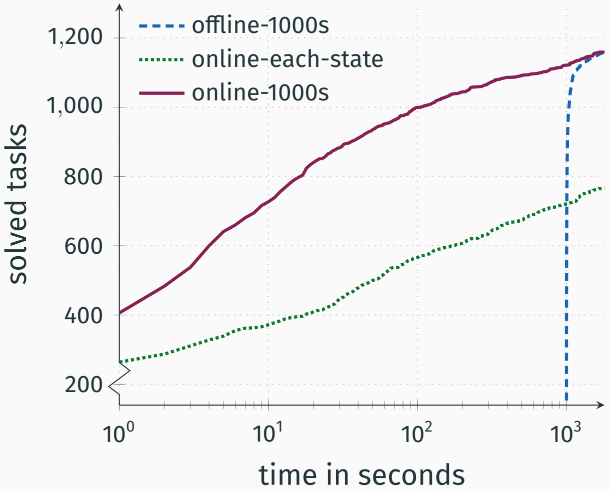
Online

- compute orders for subset of evaluated states for at most T seconds
- store order if single evaluated state profits from it

SELECT

- Bellman (Eifler and Fickert 2018)
- Novelty (Lipovetzky and Geffner 2012)
- Interval

Coverage over time



Summary

Offline diversification	Online computation	Online diversification
long precomputation	no precomputation	no precomputation
samples	states	states
fast evaluations	slow evaluations	fast evaluations
high coverage	low coverage	high coverage