Pattern Selection for Optimal Classical Planning with Saturated Cost Partitioning

Jendrik Seipp

July 11, 2019

University of Basel, Switzerland

Setting

- optimal classical planning
- A^* search + admissible heuristic
- pattern databases

How to select patterns?

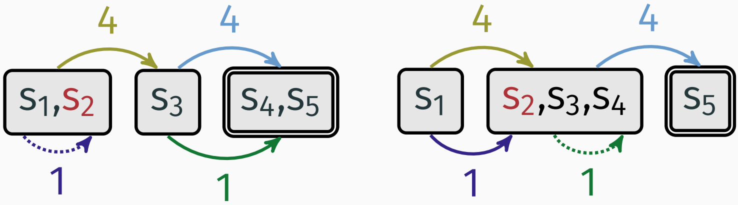
- bin packing (Edelkamp 2001)
- genetic algorithms (Edelkamp 2006)
- hill climbing (Haslum et al. 2007)
- CPC (Franco et al. 2017)
- CEGAR (Rovner et al. 2019)
- systematic naive (Felner et al. 2004)
- systematic (Pommerening et al. 2013)

How to combine multiple PDB heuristics?

- maximize
- cost partitioning
- saturated cost partitioning

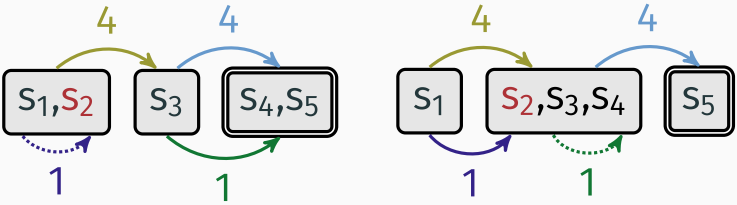
Saturated cost partitioning algorithm

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



Saturated cost partitioning algorithm

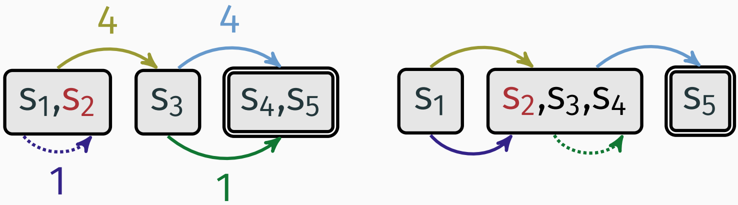
- order heuristics, then for each heuristic h:
  - use minimum costs preserving all estimates of h
  - use remaining costs for subsequent heuristics



$\max(h_1(s_2), h_2(s_2)) = \max(5, 4) = 5$

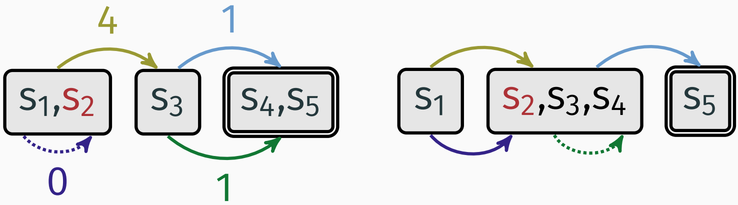
Saturated cost partitioning algorithm

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



Saturated cost partitioning algorithm

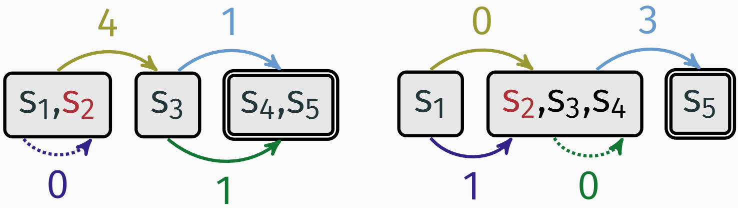
- order heuristics, then for each heuristic h:
  - use minimum costs preserving all estimates of h
  - use remaining costs for subsequent heuristics



Saturated cost partitioning

Saturated cost partitioning algorithm

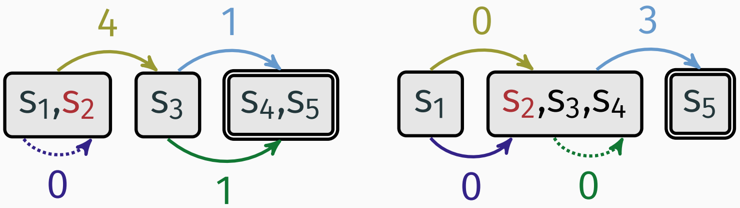
- order heuristics, then for each heuristic h:
  - use minimum costs preserving all estimates of h
  - use remaining costs for subsequent heuristics



Saturated cost partitioning

Saturated cost partitioning algorithm

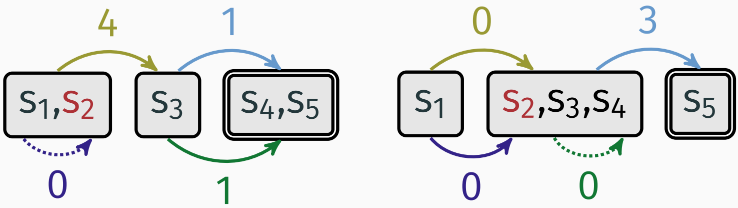
- order heuristics, then for each heuristic h:
  - use minimum costs preserving all estimates of h
  - use remaining costs for subsequent heuristics



Saturated cost partitioning

Saturated cost partitioning algorithm

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



$$h^{\mathrm{SCP}}_{\langle h_1,h_2\rangle}(s_2) = 5 + 3 = 8$$

Diverse orders for saturated cost partitioning

Diversification algorithm

- sample 1000 states $\hat{S}$
- start with empty set of orders
- for 200 seconds:
    - sample a new state s
    - find a greedy order for s
    - if a sample in $\hat{S}$ profits from it, keep it
    - otherwise, discard it

Idea

- select patterns
- compute diverse saturated cost partitionings over PDBs

Idea

- select patterns with saturated cost partitioning
- compute diverse saturated cost partitionings over PDBs

A new pattern selection algorithm

function $\textsc{Sys-SCP}(\Pi)$
    $C \leftarrow \emptyset$
    repeat for at most T_x seconds
        $\sigma \leftarrow \langle\rangle$
        for $P \in \textsc{Order}(\textsc{Sys})$ **and** at most T_y seconds **do**
            if $P \notin C$ **and** $\textsc{PatternUseful}(\sigma, P)$ **then**
                $\sigma \leftarrow \sigma \oplus P$
                $C \leftarrow C \cup \{P\}$
    until $\sigma = \langle\rangle$
    return C

function $\textsc{PatternUseful}(\sigma, P)$
    return $\exists s \in S(\mathcal{T}) : h^{SCP}_{\sigma}(cost, s) < h^{SCP}_{\sigma \oplus P}(cost, s) < \infty$

Computing PatternUseful on projections

Theorem

$$\begin{aligned}\exists s \in S(\mathcal{T}) : h^{SCP}_{\sigma}(cost, s) < h^{SCP}_{\sigma \oplus P}(cost, s) < \infty \\ \Leftrightarrow \exists s' \in S(\mathcal{T}_P) : 0 < h^{*}_{\mathcal{T}_P}(rem, s') < \infty\end{aligned}$$

Using the theorem

- keep track of the remaining cost function
- select a PDB if it has positive finite goal distances

Pattern orders

order by increasing pattern size, break ties by:

- random
- states in projection
- active operators
- Fast Downward variable order:
    - up: [7, 5], [8, 2], [8, 5]
    - down: [8, 5], [8, 2], [7, 5]

Pattern orders

order by increasing pattern size, break ties by:

- random
- states in projection
- active operators
- Fast Downward variable order:
  - up: [7, 5], [8, 2], [8, 5]
  - down: [8, 5], [8, 2], [7, 5]

Algorithm details

- store dead ends to prune states during search
- reuse Sys-SCP pattern sequences for diversification

Systematic patterns with limits

LIM: 2M states per PDB, 20M states in collection, 100 seconds

Max pattern size	1	2	3	4	5
SYS-NAIVE	840	937	914	752	571
SYS-NAIVE-LIM	840	968	1004	912	878
SYS	840	986	1057	922	731
SYS-LIM	840	985	1088	1050	1035

SYS-SCP vs. other pattern selection algorithms

	HC	SYS-3-LIM	CPC	CEGAR	SYS-SCP
Coverage	966	1088	1055	1098	1168
#domains SYS-SCP better	28	23	21	21	–
#domains SYS-SCP worse	3	2	3	3	–

Future work

- test patterns on samples

Summary

- new pattern selection algorithm based on saturated cost partitioning
- outperforms all previous pattern selection algorithms

Pattern orders

	fd-up	states-up	random	ops-down	states-down	ops-up	fd-down	Coverage
fd-up	–	5	6	5	4	3	3	1140.0
states-up	6	–	6	8	5	2	2	1153.0
random	10	10	–	8	7	6	3	1148.2
ops-down	7	8	9	–	4	7	3	1141.0
states-down	9	8	9	7	–	4	2	1152.0
ops-up	11	12	12	11	11	–	6	1166.0
fd-down	12	10	12	10	9	6	–	1168.0