Narrowing the Gap Between Saturated and Optimal Cost Partitioning for Classical Planning

Jendrik Seipp Thomas Keller Malte Helmert

University of Basel

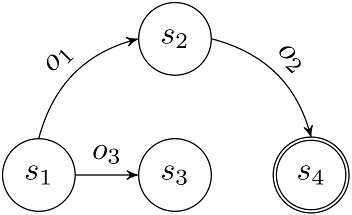
February 7, 2017

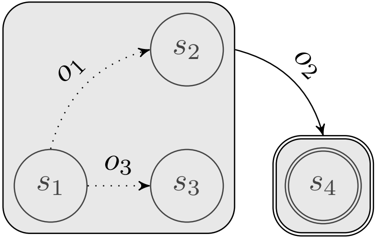
Setting

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

Setting

- optimal classical planning
- A* search + admissible heuristic
- abstraction heuristics





Problem

- single heuristic unable to capture enough information

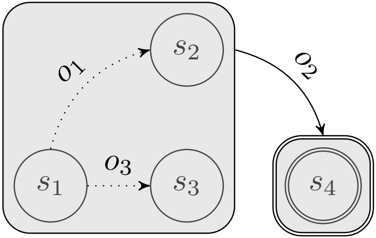
Problem

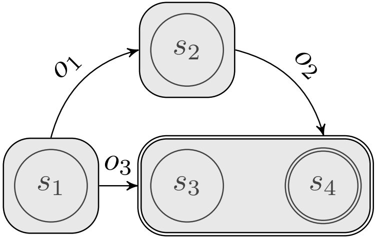
- single heuristic unable to capture enough information
  $\rightarrow$ use multiple heuristics

Problem

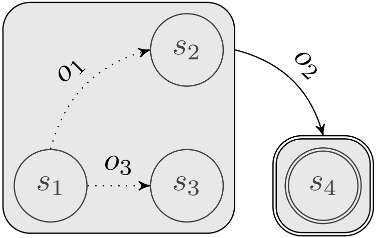
- single heuristic unable to capture enough information
  $\rightarrow$ use multiple heuristics
- how to combine multiple heuristics admissibly?

Multiple Heuristics

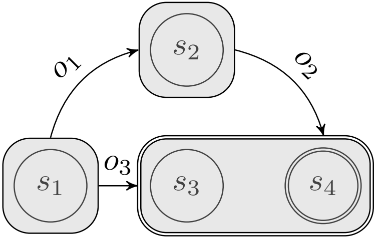




Multiple Heuristics

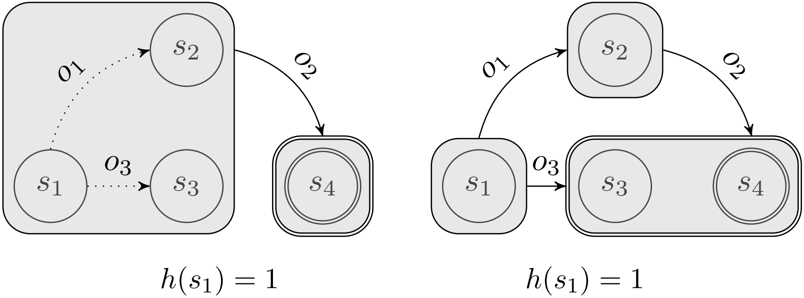


$h(s_1) = 1$



$h(s_1) = 1$

Multiple Heuristics



- maximize: $h(s_1) = 1$
  $\rightarrow$ only selects best heuristic

Multiple Heuristics: Cost Partitioning

Cost Partitioning

- split operator costs among abstractions
- total costs must not exceed original costs

→ combines heuristics

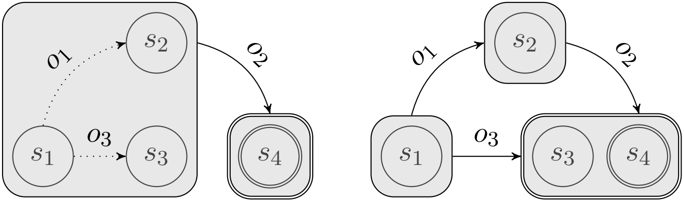
→ allows summing heuristic values admissibly

Saturated Cost Partitioning

Seipp & Helmert, 2014

Saturated Cost Partitioning Algorithm

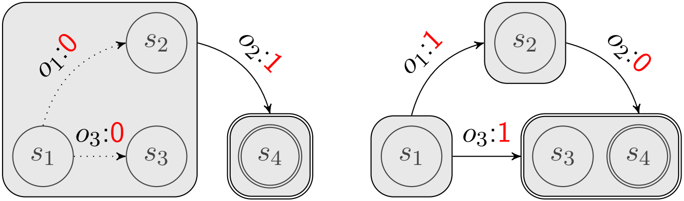
- order abstractions
- for each abstraction α:
  - use minimum costs preserving all goal distances for α
  - use remaining costs for subsequent abstractions



Saturated Cost Partitioning
Seipp & Helmert, 2014

Saturated Cost Partitioning Algorithm

- order abstractions
- for each abstraction α:
  - use minimum costs preserving all goal distances for α
  - use remaining costs for subsequent abstractions



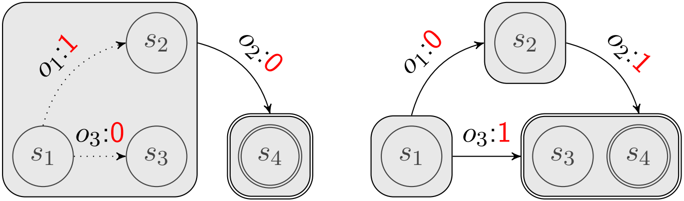
$$h^{\text{SCP}}_{\rightarrow}(s_1) = 1 + 1 = 2$$

Saturated Cost Partitioning

Seipp & Helmert, 2014

Saturated Cost Partitioning Algorithm

- order abstractions
- for each abstraction α:
  - use minimum costs preserving all goal distances for α
  - use remaining costs for subsequent abstractions



$$h^{\mathsf{SCP}}_{\rightarrow}(s_1) = 1 + 1 = 2$$

$$h^{\mathsf{SCP}}_{\leftarrow}(s_1) = 1 + 0 = 1$$

Comparison of Saturated Cost Partitioning Heuristics

Cartesian Abstractions

	2014		
Solved Tasks	$h^{\mathsf{SCP}}_{\mathsf{rand1}}$	$h^{\mathsf{SCP}}_{\mathsf{add}\uparrow}$	$h^{\mathsf{SCP}}_{\mathsf{add}\downarrow}$
Sum (1667)	759.3	789	800

Optimized Order

- $n!$ possible orders $\rightarrow$ hill climbing search in space of orders
- cover many states $\rightarrow$ samples

Optimized Order

- $n!$ possible orders $\rightarrow$ hill climbing search in space of orders
- cover many states $\rightarrow$ samples
- quality $:=$ sum of heuristic values on samples

Optimized Order

- $n!$ possible orders $\rightarrow$ hill climbing search in space of orders
- cover many states $\rightarrow$ samples
- quality $:=$ sum of heuristic values on samples

Hill climbing search

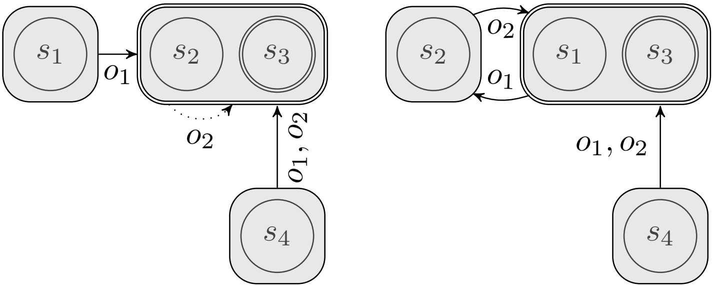
- start with random order
- until no better successor found:
  - switch positions of two heuristics
  - move to first improving successor

Comparison of Saturated Cost Partitioning Heuristics

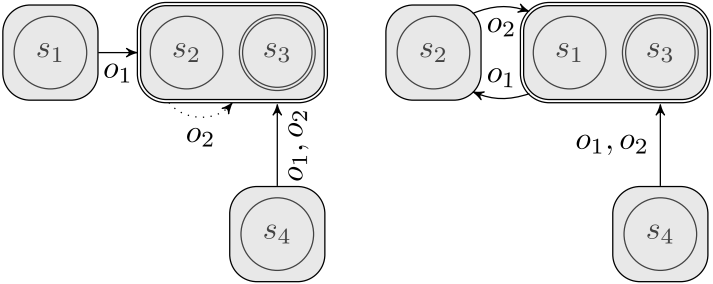
Cartesian Abstractions

	2014			
Solved Tasks	$h^{\mathsf{SCP}}_{\mathsf{rand1}}$	$h^{\mathsf{SCP}}_{\mathsf{add}\uparrow}$	$h^{\mathsf{SCP}}_{\mathsf{add}\downarrow}$	$h^{\mathsf{SCP}}_{\mathsf{HC}}$
Sum (1667)	759.3	789	800	884.9

One Order is not Enough



One Order is not Enough



$$h_{\rightarrow}^{\mathsf{SCP}}(s_1) = 1 \text{ and } h_{\leftarrow}^{\mathsf{SCP}}(s_1) = 0$$

$$h_{\rightarrow}^{\mathsf{SCP}}(s_2) = 0 \text{ and } h_{\leftarrow}^{\mathsf{SCP}}(s_2) = 1$$

Multiple Random Orders

Orders	1	2	5	10	100	200	500	1000
Coverage	759.3	797.3	866.3	907.5	942.8	**947.0**	936.3	879.5

Multiple Random Orders

Orders	1	2	5	10	100	200	500	1000
Coverage	759.3	797.3	866.3	907.5	942.8	**947.0**	936.3	879.5

- too many orders slow down heuristic evaluation
  $\rightarrow$ remove useless orders
  $\rightarrow$ find better orders

Diverse Orders

Diversification Algorithm

- sample 1000 states
- start with empty set of orders
- until time limit is reached:
  - generate a random order
  - if it improves upon current set of orders, keep it
  - otherwise, discard it

Diverse Orders

Diversification Algorithm

- sample 1000 states
- start with empty set of orders
- until time limit is reached:
  - generate a random order
  - if it improves upon current set of orders, keep it
  - otherwise, discard it

- works best with time limit of 200s

Comparison of Saturated Cost Partitioning Heuristics
Cartesian Abstractions

	2014			
Solved Tasks	$h^{\mathsf{SCP}}_{\mathsf{rand1}}$	$h^{\mathsf{SCP}}_{\mathsf{add}\uparrow}$	$h^{\mathsf{SCP}}_{\mathsf{add}\downarrow}$	$h^{\mathsf{SCP}}_{\mathsf{HC}}$
Sum (1667)	759.3	789	800	884.9

Comparison of Saturated Cost Partitioning Heuristics

Cartesian Abstractions

	2014				
Solved Tasks	$h^{\mathsf{SCP}}_{\mathsf{rand1}}$	$h^{\mathsf{SCP}}_{\mathsf{add}\uparrow}$	$h^{\mathsf{SCP}}_{\mathsf{add}\downarrow}$	$h^{\mathsf{SCP}}_{\mathsf{HC}}$	$h^{\mathsf{SCP}}_{\mathsf{rand200}}$
Sum (1667)	759.3	789	800	884.9	947.0

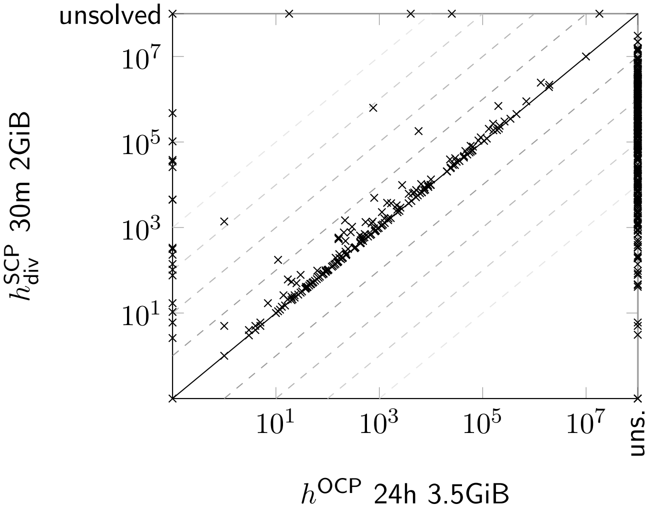
Comparison of Saturated Cost Partitioning Heuristics

Cartesian Abstractions

	2014					
Solved Tasks	$h^{\text{SCP}}_{\text{rand1}}$	$h^{\text{SCP}}_{\text{add}\uparrow}$	$h^{\text{SCP}}_{\text{add}\downarrow}$	$h^{\text{SCP}}_{\text{HC}}$	$h^{\text{SCP}}_{\text{rand200}}$	$h^{\text{SCP}}_{\text{div}}$
Sum (1667)	759.3	789	800	884.9	947.0	**966.7**

Saturated vs. Optimal Cost Partitioning

Expansions (excluding last f layer)



Comparison to Other Approaches

	$h^{\text{SCP}}_{\text{div}}$	$h^{\text{LM-cut}}$	h^{iPDB}	$h^{\text{M\&S}}$	h^{SEQ}
Coverage	**966.7**	882	814	743	734
#Domains $h^{\text{SCP}}_{\text{div}}$ better	–	20	18	25	27
#Domains $h^{\text{SCP}}_{\text{div}}$ worse	–	10	9	6	7

Follow-up Work at ICAPS 2017

- theoretical and experimental comparison of many cost partitioning algorithms
- saturated cost partitioning usually method of choice on IPC benchmarks

Conclusion

- order very important
  $\rightarrow$ hill climbing finds good order
- one order not enough
  $\rightarrow$ maximize over multiple random/diverse orders