Better Orders for Saturated Cost Partitioning in Optimal Classical Planning

Jendrik Seipp

University of Basel

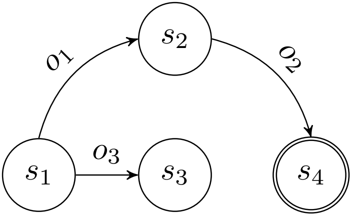
June 17, 2017

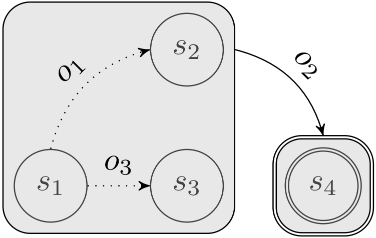
Setting

- 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

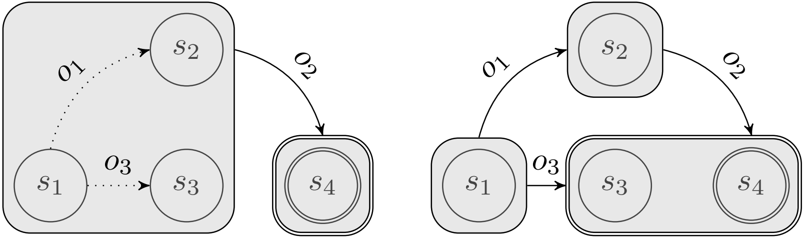
Problem

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

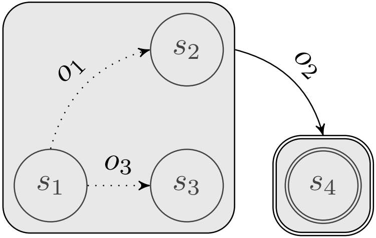
Problem

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

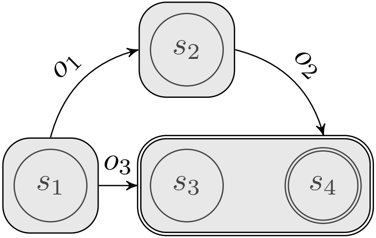
Multiple Heuristics



Multiple Heuristics

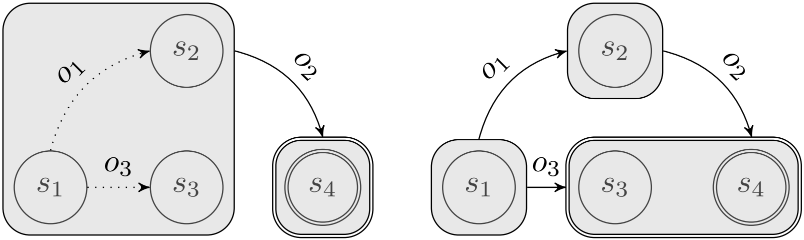


$h_1(s_1) = 1$



$h_2(s_1) = 1$

Multiple Heuristics



$h_1(s_1) = 1$ $h_2(s_1) = 1$

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

Multiple Heuristics: Cost Partitioning

Cost Partitioning

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

$\rightarrow$ combines heuristics

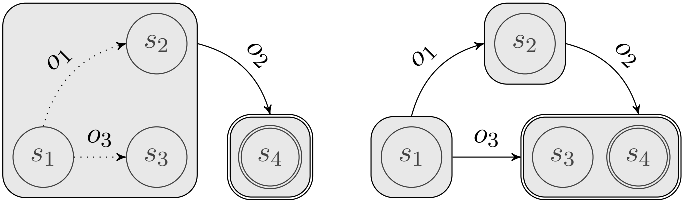
$\rightarrow$ allows summing heuristic values admissibly

Saturated Cost Partitioning

Seipp & Helmert (ICAPS 2014)

Saturated Cost Partitioning Algorithm

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

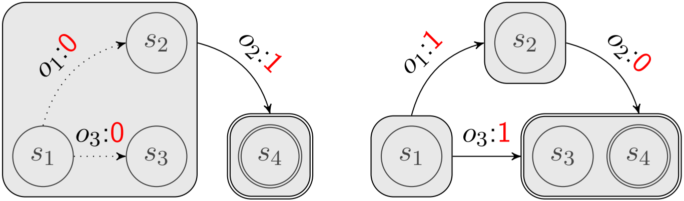


Saturated Cost Partitioning

Seipp & Helmert (ICAPS 2014)

Saturated Cost Partitioning Algorithm

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

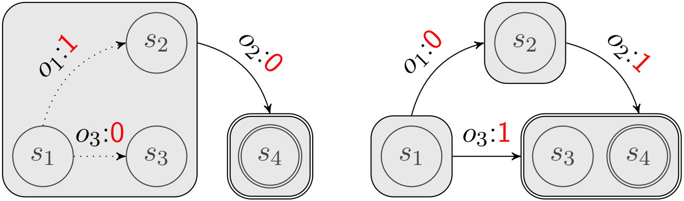


$$h^{\mathsf{SCP}}_{\langle h_1, h_2 \rangle}(s_1) = 1 + 1 = 2$$

Saturated Cost Partitioning
Seipp & Helmert (ICAPS 2014)

Saturated Cost Partitioning Algorithm

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



$$h^{\mathsf{SCP}}_{\langle h_1,h_2\rangle}(s_1) = 1+1 = 2$$

$$h^{\mathsf{SCP}}_{\langle h_2,h_1\rangle}(s_1) = 1+0 = 1$$

Orders for Saturated Cost Partitioning

Seipp, Keller & Helmert (AAAI 2017)

- orders can be arbitrarily bad
  $\rightarrow$ hill climbing in space of orders
- a single order might only be good for a single state
  $\rightarrow$ use multiple orders
- using too many orders slows down evaluation
  $\rightarrow$ use diverse orders

Combining Heterogeneous Abstraction Heuristics

	ICAPS 2017		
	$h^{\mathsf{SCP}}_{\mathsf{HC}}$	$h^{\mathsf{SCP}}_{\mathsf{Sys}}$	$h^{\mathsf{SCP}}_{\mathsf{Cart}}$
Tasks (1667)	805	852	965

Combining Heterogeneous Abstraction Heuristics

	ICAPS 2017			
	$h^{\mathsf{SCP}}_{\mathsf{HC}}$	$h^{\mathsf{SCP}}_{\mathsf{Sys}}$	$h^{\mathsf{SCP}}_{\mathsf{Cart}}$	$h^{\mathsf{SCP}}_{\mathsf{HC,Sys,Cart}}$
Tasks (1667)	805	852	965	1006

Combining Heterogeneous Abstraction Heuristics

	ICAPS 2017				
	$h^{\mathsf{SCP}}_{\mathsf{HC}}$	$h^{\mathsf{SCP}}_{\mathsf{Sys}}$	$h^{\mathsf{SCP}}_{\mathsf{Cart}}$	$h^{\mathsf{SCP}}_{\mathsf{HC,Sys,Cart}}$	$\max(h^{\mathsf{SCP}}_{\mathsf{HC}}, h^{\mathsf{SCP}}_{\mathsf{Sys}}, h^{\mathsf{SCP}}_{\mathsf{Cart}})$
Tasks (1667)	805	852	965	1006	**1032**

Combining Heterogeneous Abstraction Heuristics

	ICAPS 2017				
	$h^{\text{SCP}}_{\text{HC}}$	$h^{\text{SCP}}_{\text{Sys}}$	$h^{\text{SCP}}_{\text{Cart}}$	$h^{\text{SCP}}_{\text{HC,Sys,Cart}}$	$\max(h^{\text{SCP}}_{\text{HC}}, h^{\text{SCP}}_{\text{Sys}}, h^{\text{SCP}}_{\text{Cart}})$
Tasks (1667)	805	852	965	1006	**1032**

$$\max(\langle h_1 \rangle, \langle h'_1, h'_2 \rangle) \leq$$
$$\max(\langle h_1, h'_1, h'_2 \rangle, \langle h'_1, h'_2, h_1 \rangle)$$

Drawbacks of Diversification

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

Drawbacks of Diversification

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

- considers only random orders
  $\rightarrow$ too many orders to stumble over good ones

Drawbacks of Hill Climbing

Hill climbing search

- sample 1000 states
- start with random order
- until no better successor for samples found:
  - swap positions of two heuristics
  - move to first improving successor

Drawbacks of Hill Climbing

Hill climbing search

- sample 1000 states
- start with random order
- until no better successor for samples found:
  - swap positions of two heuristics
  - move to first improving successor

- tries to find order for set of states
  $\rightarrow$ there may not be a single order for multiple states
- starts with random order
  $\rightarrow$ decent initial solution important for local optimization

Improvements

- optimize for single state
- start with greedy order
- diversify optimized greedy orders

Greedy Order

Objectives:

- increase heuristic value quickly
- preserve costs for subsequent heuristics

Greedy Order

Objectives:

- increase heuristic value quickly
- preserve costs for subsequent heuristics

Value-per-cost ratio

$$\textit{ratio}(h, s) = \frac{h(s)}{\textit{saturated costs for } h}$$

Greedy Order

Objectives:

- increase heuristic value quickly
- preserve costs for subsequent heuristics

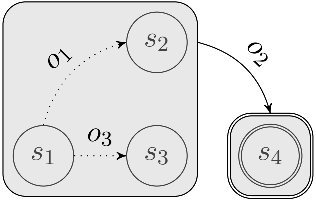
Value-per-cost ratio

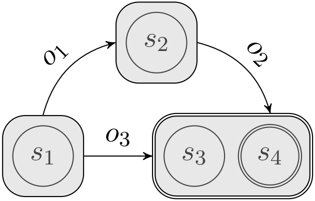
$$\mathit{ratio}(h, s) = \frac{h(s)}{\textit{saturated costs for } h}$$

Greedy algorithm

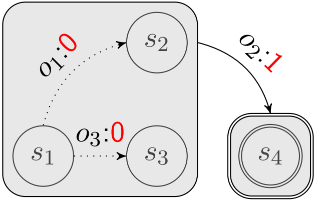
- start with empty order
- until all heuristics are ordered
  - append heuristic h with highest value-per-cost ratio
  - subtract saturated costs for h from overall costs

Greedy Order: Example

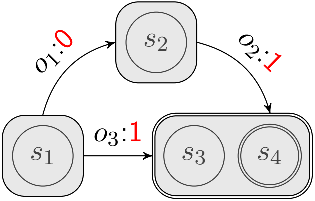




Greedy Order: Example

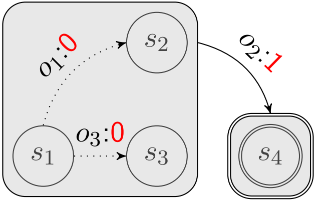


$h_1(s_1) = 1$
saturated costs: 1
ratio: 1

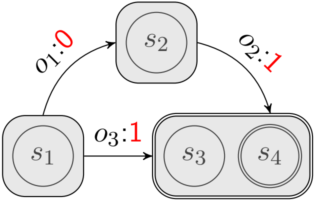


$h_2(s_1) = 1$
saturated costs: 2
ratio = 0.5

Greedy Order: Example



$h_1(s_1) = 1$
saturated costs: 1
ratio: 1



$h_2(s_1) = 1$
saturated costs: 2
ratio = 0.5

$\rightarrow \langle h_1, h_2 \rangle$

Single Order for Initial State

	random	greedy	random-opt	greedy-opt
random	–	105	0	20
greedy	**989**	–	268	0
random-opt	**1086**	**463**	–	111
greedy-opt	**1108**	**548**	**402**	–

Diverse Orders

New diversification algorithm

Until time limit is reached:

- sample state s
- find greedy order for s
- optimize order with hill climbing
- keep order if diverse

Diverse Orders

New diversification algorithm

Until time limit is reached:

- sample state s
- find greedy order for s
- optimize order with hill climbing
- keep order if diverse

	max	random	random-opt	greedy-opt
Tasks (1667)	1032	1006	1009	**1048**

Comparison to State of the Art

Using h^2 mutexes:

- SymBA_2^* outperforms $h^{\text{SCP}}_{\text{greedy-opt}}$ in 11 domains
- $h^{\text{SCP}}_{\text{greedy-opt}}$ outperforms SymBA_2^* in 22 domains
- SymBA_2^* solves 1008 tasks
- $h^{\text{SCP}}_{\text{greedy-opt}}$ solves 1084 tasks

Related ICAPS Talk and Poster

Seipp, Keller & Helmert (ICAPS 2017)

A Comparison of Cost Partitioning Algorithms for Optimal Classical Planning

- theoretical and empirical comparison of cost partitioning algorithms
- saturated cost partitioning usually method of choice for hill climbing PDBs, systematic PDBs, Cartesian abstractions and landmark heuristics

Presented on Wednesday in the first ICAPS session "Optimal Planning"

Conclusion

- new greedy algorithm for finding orders
- pair with optimization and diversification
- combine heterogeneous abstraction heuristics