Automatic Configuration of Sequential Planning Portfolios

Jendrik Seipp[1] Silvan Sievers[1] Malte Helmert[1]
Frank Hutter[2]

[1]University of Basel

[2]University of Freiburg

January 29, 2014

Why is this interesting?

- You have:
  algorithm with many parameters
  training instances
- You want to:
  solve new similar instances

Why is this interesting?

- You have:
  algorithm with many parameters
  training instances
- You want to:
  solve new similar instances
- You get:
  sequential portfolio of complementary parameter configurations

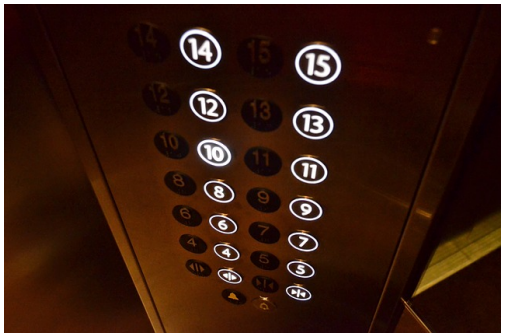
Why is this interesting?

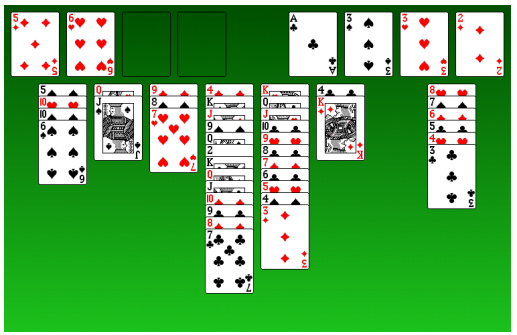
- You have:
  algorithm with many parameters
  training instances
- You want to:
  solve new similar instances
- You get:
  sequential portfolio of complementary parameter configurations

- Only planning here
- Literature pointers in the paper

Background

AI planning



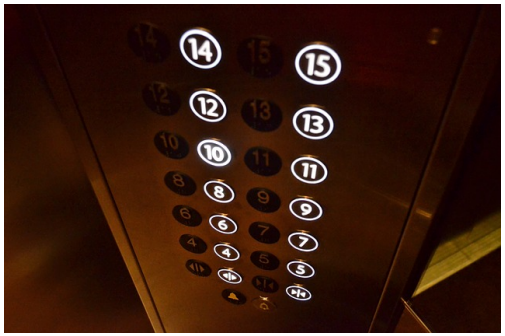


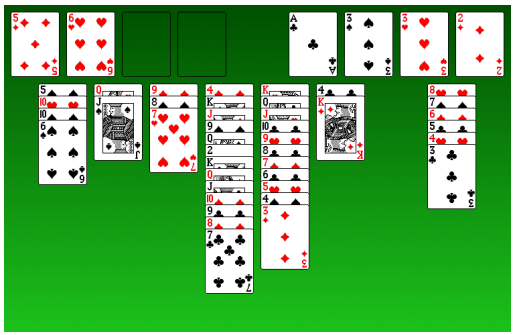
Algorithm configuration

- Takes:
  parameterized algorithm
  training instances
- Returns:
  good parameter configuration for these instances

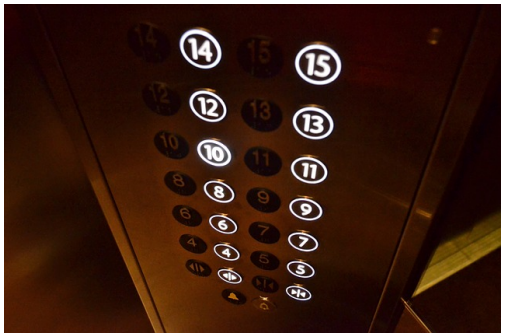
Tools: ParamILS, GGA, irace, SMAC

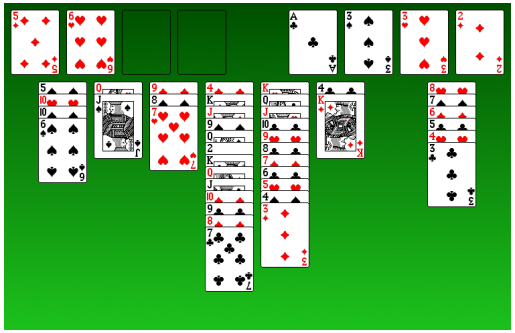
How to solve new planning tasks?





How to solve new planning tasks?

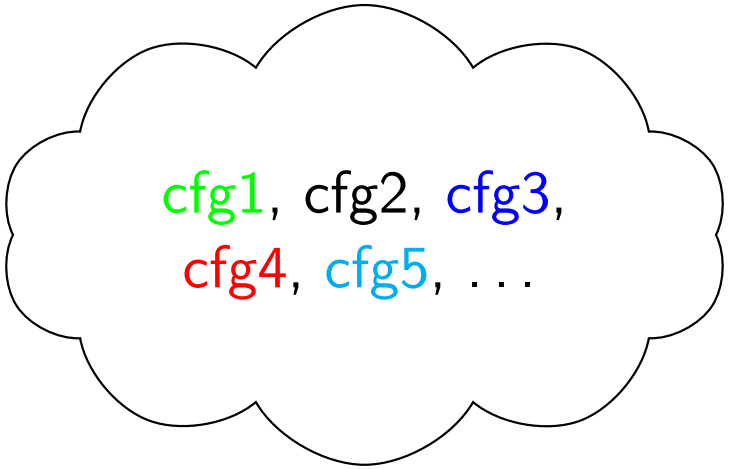




cfg1, cfg2, cfg3,
cfg4, cfg5, . . .

Sequential portfolios

Sequential portfolios



Choose configurations manually

Example: Fast Downward Stone Soup

- Manually select set of "good" configurations
- Calculate time slices in second step
- One first and one second place in IPC 2011

Drawbacks:

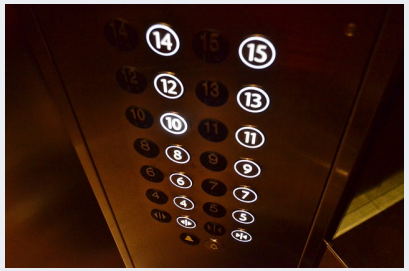
- Experts need to choose configurations
- Configurations complementary?

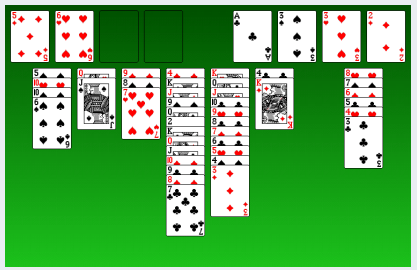
Use algorithm configuration to find configurations

Example: domain-wise

- Find configuration for each domain separately
- Assign time slices in second step


cfg4


cfg1


cfg6

Drawbacks:

- How many domains are enough?
- Configurations complementary?

Cedalion

Algorithm

Cedalion

- Use algorithm configuration to find complementary configurations
- Include time in the configuration space
- Iteratively add configuration that solves the most additional instances per time

Cedalion by example

$$\text{maximize } \frac{|\text{newly solved instances in time } t|}{t}$$

Rem. time:

Rem. instances:

Config. space:

cfg1:

cfg2:

cfg3:

Portfolio:

Cedalion by example

$$\text{maximize } \frac{|\text{newly solved instances in time } t|}{t}$$

Rem. time: 30
Rem. instances: 10
Config. space: [1,30]×{cfg1,cfg2,cfg3}

cfg1:
cfg2:
cfg3:

Portfolio:

Cedalion by example

$$\text{maximize } \frac{|\text{newly solved instances in time } t|}{t}$$

Rem. time: 30
Rem. instances: 10
Config. space: [1,30]×{cfg1,cfg2,cfg3}

cfg1: 8 in 4s → 2
cfg2: 3 in 6s → 0.5
cfg3: 5 in 1s → **5**

Portfolio:

Cedalion by example

$$\text{maximize } \frac{|\text{newly solved instances in time } t|}{t}$$

Rem. time: 30
Rem. instances: 10
Config. space: [1,30]×{cfg1,cfg2,cfg3}

cfg1: 8 in 4s → 2
cfg2: 3 in 6s → 0.5
cfg3: 5 in 1s → **5**

Portfolio: cfg3

Cedalion by example

$$\text{maximize } \frac{|\text{newly solved instances in time } t|}{t}$$

Rem. time:	30	29
Rem. instances:	10	5
Config. space:	[1,30]×{cfg1,cfg2,cfg3}	[1,29]×{cfg1,cfg2,cfg3}

cfg1: 8 in 4s → 2
cfg2: 3 in 6s → 0.5
cfg3: 5 in 1s → **5**

Portfolio: cfg3

Cedalion by example

$$\text{maximize } \frac{|\text{newly solved instances in time } t|}{t}$$

Rem. time:	30	29
Rem. instances:	10	5
Config. space:	[1,30]×{cfg1,cfg2,cfg3}	[1,29]×{cfg1,cfg2,cfg3}
cfg1:	8 in 4s → 2	3 in 4s → **0.75**
cfg2:	3 in 6s → 0.5	2 in 6s → 0.33
cfg3:	5 in 1s → **5**	1 in 20s → 0.05
Portfolio:	cfg3	

Cedalion by example

$$\text{maximize } \frac{|\text{newly solved instances in time } t|}{t}$$

Rem. time:	30	29
Rem. instances:	10	5
Config. space:	[1,30]×{cfg1,cfg2,cfg3}	[1,29]×{cfg1,cfg2,cfg3}
cfg1:	8 in 4s → 2	3 in 4s → **0.75**
cfg2:	3 in 6s → 0.5	2 in 6s → 0.33
cfg3:	5 in 1s → **5**	1 in 20s → 0.05
Portfolio:	cfg3	cfg3 cfg1

Cedalion by example

$$\text{maximize } \frac{|\text{newly solved instances in time } t|}{t}$$

Rem. time:	30	29	25
Rem. instances:	10	5	2
Config. space:	[1,30]×{cfg1,cfg2,cfg3}	[1,29]×{cfg1,cfg2,cfg3}	...
cfg1:	8 in 4s → 2	3 in 4s → **0.75**	...
cfg2:	3 in 6s → 0.5	2 in 6s → 0.33	...
cfg3:	5 in 1s → **5**	1 in 20s → 0.05	...
Portfolio:	cfg3	cfg3 cfg1	...

Cedalion's properties

Drawbacks:

- Only works for instances from seen domains
- Long learning time

Cedalion's properties

Drawbacks:

- Only works for instances from seen domains
- Long learning time

Advantages:

- Needs no planning expertise
- Selects configurations and time slices together
- Operates on all instances at once
- Returns complementary configurations

Evaluation

Results

- Configuration space: Fast Downward
  45 parameters, 3×10^{13} configurations
- Benchmarks: IPC 2011 instances
- 10h/30h per iteration

Comparison to most closely related methods

Setting
satisficing
optimal
agile
learning

Results

- Configuration space: Fast Downward
  45 parameters, 3×10^{13} configurations
- Benchmarks: IPC 2011 instances
- 10h/30h per iteration

Comparison to most closely related methods

Setting	**Iterations**
satisficing	48
optimal	15
agile	10
learning	2–14 (8.77)

Results

- Configuration space: Fast Downward
  45 parameters, 3×10^{13} configurations
- Benchmarks: IPC 2011 instances
- 10h/30h per iteration

Comparison to most closely related methods

Setting	Iterations	Performance
satisficing	48	slightly better than domain-wise
optimal	15	slightly worse than FD Stone Soup
agile	10	better than LAMA-2011
learning	2–14 (8.77)	better than FD-Autotune

IPC 2014 learning track

Learn on training instances $\rightarrow$ evaluate on unseen instances from same domain

IPC 2014 learning track

Learn on training instances $\rightarrow$ evaluate on unseen instances from same domain

Overall best quality

1. MIPlan
2. Fast Downward Cedalion
3. Fast Downward SMAC

IPC 2014 learning track

Learn on training instances $\rightarrow$ evaluate on unseen instances from same domain

Overall best quality

1. MIPlan
2. Fast Downward Cedalion
3. Fast Downward SMAC

Best learner

1. Fast Downward Cedalion
2. Eroller
3. Fast Downward SMAC

Conclusion

Summary

- Make time slices part of the configuration space
- Iteratively add configuration solving the most additional instances per time
- Competitive empirical performance

Image Credits

- Truck: https://www.flickr.com/photos/25328551@N08/ (CC BY 2.0)
- Elevator: Public Domain (CC0 1.0)
- Freecell: GNOME Project (GNU General Public License)