Automatic Configuration of Sequential Planning Portfolios

Jendrik Seipp\textsuperscript{1} \hspace{1em} Silvan Sievers\textsuperscript{1} \hspace{1em} Malte Helmert\textsuperscript{1}
Frank Hutter\textsuperscript{2}

\textsuperscript{1}University of Basel
\textsuperscript{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

  - cfg4
  - cfg1
  - cfg3
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

  - `cfg4`
  - `cfg1`
  - `cfg3`

- Only **planning** here

- Literature pointers in the paper
<table>
<thead>
<tr>
<th>Background</th>
<th>Sequential portfolios</th>
<th>Cedalion</th>
<th>Evaluation</th>
<th>Conclusion</th>
</tr>
</thead>
</table>

**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

Automatic Configuration of Sequential Planning Portfolios

**J. Seipp, S. Sievers, M. Helmert, F. Hutter**
Sequential portfolios

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

? →

cfg4  cfg1  cfg3
Choose configurations manually

<table>
<thead>
<tr>
<th>Example: Fast Downward Stone Soup</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Manually</strong> select set of “good” configurations</td>
</tr>
<tr>
<td>• Calculate time slices in second step</td>
</tr>
<tr>
<td>• One first and one second place in IPC 2011</td>
</tr>
</tbody>
</table>

**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

**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

\[
\text{maximize } \left| \text{newly solved instances in time } t \right| \div t
\]

Rem. time:
Rem. instances:
Config. space:

\[\text{cfg1:} \]
\[\text{cfg2:} \]
\[\text{cfg3:} \]

Portfolio:
Cedalion by example

maximize \[
\left| \frac{\text{newly solved instances in time } t}{t} \right|
\]

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: \([1,30] \times \{\text{cfg1}, \text{cfg2}, \text{cfg3}\}\)

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

Portfolio:
## Cedalion by example

The objective is to maximize the number of newly solved instances in time $t$.

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

| Rem. time: | 30 |
| Rem. instances: | 10 |
| Config. space: | $[1,30] \times \{\text{cfg1, cfg2, cfg3}\}$ |

### Results

- **cfg1**: 8 in 4s $\rightarrow$ 2
- **cfg2**: 3 in 6s $\rightarrow$ 0.5
- **cfg3**: 5 in 1s $\rightarrow$ 5

**Portfolio:** cfg3
Cedalion by example

maximize \[ \frac{\text{\# newly solved instances in time } t}{t} \]

| Rem. time: | 30 | 29 |
| Rem. instances: | 10 | 5 |
| Config. space: | \([1,30] \times \{\text{cfg1, cfg2, cfg3}\}\) | \([1,29] \times \{\text{cfg1, cfg2, cfg3}\}\) |

| cfg1: | 8 in 4s \(\rightarrow\) 2 |
| cfg2: | 3 in 6s \(\rightarrow\) 0.5 |
| cfg3: | 5 in 1s \(\rightarrow\) 5 |

Portfolio: \(\text{cfg3}\)
### Cedalion by example

Maximize the number of newly solved instances in time $t$.

| Rem. time:  | 30 | 29 |
| Rem. instances: | 10 | 5 |
| Config. space: | $[1,30] \times \{\text{cfg1, cfg2, cfg3}\}$ | $[1,29] \times \{\text{cfg1, cfg2, cfg3}\}$ |

|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

Maximize \[ \left| \text{newly solved instances in time } t \right| \div t \]

| Rem. time: | 30 | 29 |
| Rem. instances: | 10 | 5 |
| Config. space: | \([1,30] \times \{ \text{cfg1, cfg2, cfg3} \} \) | \([1,29] \times \{ \text{cfg1, cfg2, cfg3} \} \) |

**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] \times \{\text{cfg1, cfg2, cfg3}\} | [1,29] \times \{\text{cfg1, cfg2, cfg3}\} | \ldots |
| \text{cfg1:} | 8 in 4s \rightarrow 2 | 3 in 4s \rightarrow 0.75 | \ldots |
| \text{cfg2:} | 3 in 6s \rightarrow 0.5 | 2 in 6s \rightarrow 0.33 | \ldots |
| \text{cfg3:} | 5 in 1s \rightarrow 5 | 1 in 20s \rightarrow 0.05 | \ldots |
| Portfolio: | \text{cfg3} | \text{cfg3, cfg1} | \ldots |
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, $3 \times 10^{13}$ configurations
- **Benchmarks:** IPC 2011 instances
- 10h/30h per iteration

**Comparison to most closely related methods**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>satisficing</td>
<td>slightly better than domain-wise optimal</td>
</tr>
<tr>
<td>optimal</td>
<td>slightly worse than FD Stone Soup</td>
</tr>
<tr>
<td>agile</td>
<td>better than LAMA-2011</td>
</tr>
<tr>
<td>learning</td>
<td>better than FD-Autotune</td>
</tr>
</tbody>
</table>
Results

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

**Comparison to most closely related methods**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>satisficing</td>
<td>48</td>
</tr>
<tr>
<td>optimal</td>
<td>15</td>
</tr>
<tr>
<td>agile</td>
<td>10</td>
</tr>
<tr>
<td>learning</td>
<td>2–14 (8.77)</td>
</tr>
</tbody>
</table>
Results

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

### Comparison to most closely related methods

<table>
<thead>
<tr>
<th>Setting</th>
<th>Iterations</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>satisficing</td>
<td>48</td>
<td>slightly better than domain-wise</td>
</tr>
<tr>
<td>optimal</td>
<td>15</td>
<td>slightly worse than FD Stone Soup</td>
</tr>
<tr>
<td>agile</td>
<td>10</td>
<td>better than LAMA-2011</td>
</tr>
<tr>
<td>learning</td>
<td>2–14 (8.77)</td>
<td>better than FD-Autotune</td>
</tr>
</tbody>
</table>
IPC 2014 learning track

Learn on training instances → evaluate on unseen instances from same domain
IPC 2014 learning track

Learn on training instances → 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)