Combining Heuristics for Configuration Problems using ASP

Martin Gebser,
Aalto University, HIIT, Finland and Potsdam University, Germany

Anna Ryabokon,
Alpen-Adria-Universität Klagenfurt, Austria

Gottfried Schenner,
Siemens AG Österreich, Austria

funded by FFG, COIN and AoF (grants 840242 and 251170)
Outline

• Motivation
• Combined Configuration Problem (CCP)
• Solving framework for the CCP
• Heuristic approaches
• Benchmarks
• Evaluation
• Summary
Research projects

RECONCILE (Reconciling Legacy Instances with changed Ontologies): 01.06.2010 – 31.05.2013

HINT (Heuristic Intelligence): 01.06.2013 – 31.05.2016

AINF and Cognitive Psychology Unit
Reconcile results

Use cases: Partner Units Problem [Aschinger et al. 2011], House problem (Rack configuration) [Friedrich et al. 2011], Reviewer Assignment Problem [Ryabokon et al. 2012]

http://isbi.aau.at/reconcile/benchmarks

Approaches:

• Answer set programming and SAT
• Constraint programming
• Object-oriented programming
• Integer programming

Configuration problem is too hard! (without heuristics)

[Teppan et al. 2012], [Ryabokon et al. 2013]
HINT goals

“Development of new methods for the efficient generation of heuristics” [http://isbi.aau.at/hint/]

- Identify promising domain-specific heuristics and express them within general purpose framework
- Combine different heuristics for different problems
- Create new heuristics out of existing ones

create + adapt + evolve heuristics

= Heuristic INTelligence
Configuration problem

Customer requirements

Solution

Components

Configuration requirements
CCP instance

- A directed acyclic graph (edges, vertices)
- Type and size of a vertex
- Sets of vertices denoting paths in the graph
- Set of areas and their possible border elements
- Maximal number of selected border elements
- Number of available colors
- Number of bins and their capacity

Benchmarks

<table>
<thead>
<tr>
<th>Instance</th>
<th>Vertices</th>
<th>Colors</th>
<th>Bins</th>
<th>MaxBinCapacity</th>
<th>MaxBorder</th>
</tr>
</thead>
<tbody>
<tr>
<td>tg_001004</td>
<td>1004</td>
<td>58</td>
<td>4</td>
<td>20</td>
<td>2</td>
</tr>
</tbody>
</table>
Combined Configuration Problem

Given a CCP instance, solve the following problems separately or in combinations:

P1 Coloring
P2 Bin-Packing
P3 Disjoint Paths
P4 Matching
P5 Connectedness

[Mayer et al. 2009],
[Friedrich et al. 2011]

[Aschinger et al. 2011a],
[Aschinger et al. 2011b]
Coloring (P1)
Bin-Packing (P2)

Bin capacity = 5

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>2</td>
</tr>
<tr>
<td>s</td>
<td>3</td>
</tr>
<tr>
<td>p</td>
<td>4</td>
</tr>
</tbody>
</table>
Disjoint Paths (P3)

path1

path2

path1

path2
Matching requirements (P4)

Each area can have at most 2 border elements.

The selected border elements of an area must have the same color.
Matching solution (P4)

Each area has at most 2 border elements

The selected border elements of an area have the same color
Connectedness (P5)
Solving framework
CCP greedy algorithms

**Algorithm 1:** Matching (P4)

For each border element select an area with the minimum number of already matched elements

**Algorithm 2:** Coloring_Bin-Packing_Connectedness (P1, P2, P5)

Select a subset of connected vertices, color them with a selected color and place them to bins. Change the color and repeat until all vertices are processed.
Heuristics in ASP


- Specified using atoms `heuristic(a,m,v,p)`
- Shortcuts are used, e.g. `heuristic(a,true,v)`
- Configuration example:

```
size(a,1). size(b,2). size(c,3).
1 {selected(V,S): size(V,S)} 1.
heuristic(selected(V,S), true, S) :- size(V,S).
```

```
size(a,1). size(b,2). size(c,3).
1<= {selected(a,1), selected(b,2), selected(c,3)} <=1.
heuristic(selected(a,1),true,1).
heuristic(selected(b,2),true,2).
heuristic(selected(c,3),true,3).
```

```
selected(c,3) ... 
```
Greedy vs. ASP

Greedy

😊 An implementation of a subproblem of the CCP can be done easy and is usually efficient
😊 Designing a mixed greedy method for the problem is difficult

ASP

😊 The addition of requirements in ASP is just a matter of adding some rules to an encoding
😊 Generation of heuristics is “expensive”

**Combine** two “worlds” effectively!
Greedy & ASP architecture

- ASP encoding
- Problem instance
- Greedy search
- Partial solution
- Solution
- ASP solver
- ASP heuristic
- Heuristic generator
- ASP solver
- ASP heuristic
Benchmarks

Set 1:
Bin-Packing instances converted to the CCP instances
http://www.wiwi.uni-jena.de/Entscheidung/binpp/index.htm

Set 2 and Set 3:
• Moderate and hard CCP instances derived from Siemens configurations
• Available from http://isbi.aau.at/hint/problems
• Submitted to the ASP competition 2015
http://aspcomp2015.dibris.unige.it/
Evaluation

• **Experiment 1:**
  - Instances Set 1
  - P2 (Bin-Packing) must be solved
  - Plain ASP encoding vs. ASP encodings extended with the BPP heuristics (FF(D), BF(D) and NF(D))
  - FF(D) is on average 2.5 times faster, less bins are utilized

• **Experiment 2 and Experiment 3:**
  - Instances Set 2 and instances Set 3 resp.
  - P1 - P5 (all subproblems) must be solved
  - Plain ASP encoding vs. Greedy & ASP approach
  - Combined approach outperforms Plain ASP encoding, the quality of solutions is the same

* Gringo 4.4.0, Clasp 3.0.5; Intel i7-3930K CPU (3.20GHz), 64 GB RAM, timeout 900 sec
Summary

• Heuristic greedy algorithms can find a solution faster, but the design of such algorithms is complicated
• ASP allows for combination of requirements in an easier way, but has performance issues
• Combining different solving methods is possible and seems to be promising!
• ~50% more instances can be solved
• up to 18 times faster on average
• Next steps…
Thank you! Questions?

The images are taken from: http://psychstrike.com/ and http://www.dreamstime.com/