Chapter 10: Customizing Search (Progressive Party Problem)

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

ECLiPSe ELearning

Overview

Licence

This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.
What we want to introduce

- Problem decomposition
  - Decide which problem to solve
  - Not always required to solve complete problem in one go
- Modelling with bin packing
- Customized search routines can bring dramatic improvements
- Understanding what is happening important to find improvements
Progressive Party

The problem is to timetable a party at a yacht club. Certain boats are to be designated hosts, and the crews of the remaining boats in turn visit the host boats for several successive half-hour periods. The crew of a host boat remains on board to act as hosts while the crew of a guest boat together visits several hosts. Every boat can only host a limited number of guests at a time (its capacity) and crew sizes are different. The party lasts for 6 time periods. A guest boat cannot not revisit a host and guest crews cannot meet more than once. The problem facing the rally organizer is that of minimizing the number of host boats.

Data

<table>
<thead>
<tr>
<th>Boat</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>6</td>
<td>8</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Crew</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Boat</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>Capacity</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Crew</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Boat</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>Capacity</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Crew</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
High Level Problem Decomposition

- Phase 1: Select minimal set of host boats
  - Manually
- Phase 2: Create plan to assign guest boats to hosts in multiple periods
  - Done as a constraint program

Idea

- Decompose problem into multiple, simpler sub problems
- Solve each sub problem in turn
- Provides solution of complete problem
- Challenge: How to decompose so that good solutions are obtained?
- How to show optimality of solution?
### Selecting Host boats

- Some additional side constraints
  - Some boats must be hosts
  - Some boats may not be hosts
- Reason on total or spare capacity
- No solution with 12 boats (with side constraints)

### Solution to Phase 1

- Select boats 1 to 12 and 14 as hosts
- Many possible problem variants by selecting other host boats
Phase 2 Sub-problem

- Host boats and their capacity given
- Ignore host teams, only consider free capacity
- Assign guest teams to host boats

Model

- Assign guest boats to hosts for each time period
- Matrix (size $NrGuests \times NrPeriods$) of domain variables $x_{ij}$
- Variables range over possible hosts $1..NrHosts$
Constraints

- Each guest boat visits a host boat at most once
- Two guest boats meet at most once
- All guest boats assigned to a host in a time period fit within spare capacity of host boat

Each guest visits a host at most once

- The variables for a guest and different time periods must be pairwise different
- \textit{alldifferent} constraint on rows of matrix
- alldifferent(\{x_{ij}|1 \leq j \leq NrPeriods\})
Two guests meet at most once

- The variables for two guests can have the same value for at most one time period
- Constraints on each pair of rows in matrix
  \[ x_{i_1j} = x_{i_2j}, \ i_1 \neq i_2 \Rightarrow x_{i_1k} \neq x_{i_2k} \ 1 \leq k \leq \text{NrPeriods}, \ k \neq j \]

All guests assigned to a host in a time period fit within spare capacity of host boat

- Capacity constraint expressed as bin packing for each time period
- Each host boat is a bin with capacity from 0 to its unused capacity
- Each guest is an item to be assigned to a bin
- Size of item given by crew size of guest boat
Bin Packing Constraint

- Global constraint
  
  \[
  \text{bin\_packing}(\text{Assignment}, \text{Sizes}, \text{Capacity})
  \]

- Items of different sizes are assigned to bins
- Assignment of item modelled with domain variable (first argument)
- Size of items fixed: integer values (second argument)
- Each bin may have a different capacity
- Capacity of each bin given as a domain variable (third argument)

Main Program

\[
\text{top}:-
\]

\[
\text{top}(10, 6).
\]

\[
\text{top}(\text{Problem}, \text{Size}):-
\]

\[
\text{problem}(\text{Problem}, \text{Hosts}, \text{Guests}),
\]

\[
\text{model}(\text{Hosts}, \text{Guests}, \text{Size}, \text{Matrix}),
\]

\[
\text{writeln}(\text{Matrix}).
\]
problem(10, 
   [10,10,9,8,8,8,8,8,7,6,6,4],
   [7,6,5,5,5,4,4,4,4,4,4,4,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2]).

model(Hosts, Guests, NrPeriods, Matrix):-
   length(Hosts, NrHosts),
   length(Guests, NrGuests),
   dim(Matrix, [NrGuests, NrPeriods]),
   Matrix[1..NrGuests,1..NrPeriods] :: 1..NrHosts,
   ...
Setting up **alldifferent** constraints

... 
(for(I,1,NrGuests),
  param(Matrix,NrPeriods) do 
    ic:alldifferent(Matrix[I,1..NrPeriods]) 
  ),
...

Setting up **bin_packing** constraints

... 
(for(J,1,NrPeriods),
  param(Matrix,NrGuests,Guests,Hosts) do 
    make_bins(Hosts,Bins),
    bin_packing(Matrix[1..NrGuests,J],
                Guests,Bins)
  ),
...

Each pair of guests meet at most once

... 
(for(I,1,NrGuests-1), 
  param(Matrix,NrGuests,NrPeriods) do 
    (for(I1,I+1,NrGuests), 
      param(Matrix,NrPeriods,I) do 
        card_leq(Matrix[I,1..NrPeriods], 
                  Matrix[I1,1..NrPeriods],1) 
    ), 
), 
...

Call search

... 
extract_array(col,Matrix,List), 
assign(List).
Make Bin variables

```prolog
make_bins(HostCapacity, Bins):-
    (foreach(Cap, HostCapacity),
     foreach(B, Bins) do
         B :: 0..Cap
    ).
```

Each pair of guests meet atmost once

```prolog
card_leq(Vector1, Vector2, Card):-
    collection_to_list(Vector1, List1),
    collection_to_list(Vector2, List2),
    (foreach(X, List1),
     foreach(Y, List2),
     fromto(0, A, A+B, Term) do
         #=(X, Y, B)
    ),
    eval(Term) #=> Card.
```
assign(List):-
    search(List,0,input_order,indomain,
            complete,[]).
Observations

- Not too many wrong choices
- But very deep backtracking required to discover failure
- Most effort wasted in “dead” parts of search tree
assign(List):-
    search(List, 0, \texttt{first\_fail}, \texttt{indomain},
    complete, []).
Observations

- Assignment not done in row or column mode
- Tree consists of straight parts without backtracking
- ... and nearly fully explored parts
Idea

- Assign variables by time period
- Within one time period, use first_fail selection
- Solves bin packing packing for each period completely
- Clearer impact of disequality constraints
- Serial composition of search procedures

Layered Search

assign(Matrix, NrPeriods, NrGuests) :-
    (for(J, 1, NrPeriods),
     param(Matrix, NrGuests) do
        search(Matrix[1..NrGuests, J], 0, first_fail, indomain, complete, []))
Layered Search

Layered Solution (Zoomed)
Observations

- Deep backtracking for last time period
- No backtracking to earlier time periods required
- Small amount of backtracking at other time periods

Idea

- Use credit based search
- But not for complete search tree
- Lose too much useful work
- Backtrack independently for each time period
- Hope to correct wrong choices without deep backtracking
Reminder: Credit Based Search

- Explore top of tree completely, based on credit
- Start with fixed amount of credit
- Each node consumes one credit unit
- Split remaining credit amongst children
- When credit runs out, start bounded backtrack search
- Each branch can use only $K$ backtracks
- If this limit is exceeded, jump to unexplored top of tree

Layered with Credit

```prolog
assign(Matrix, NrPeriods, NrGuests):-
    (for(J, 1, NrPeriods),
     param(Matrix, NrGuests) do
        NSq is NrGuests * NrGuests,
        search(Matrix[1..NrGuests, J], 0,
                first_fail, indomain,
                credit(NSq, 10), []))
).```

Helmut Simonis  Customizing Search  41
Layered with Credit Search

Layered with Credit Search (Zoomed)
Observations

- Improved search
- Need more sample problems to really understand impact

Idea

- Randomize value selection
- Remove bias picking bins in same order
- Allows to add restart
- When spending too much time without finding solution
- Restart search from beginning
- Randomization will explore other initial assignments
- Do not get caught in “dead” part of search tree
assign(Matrix,NrPeriods,NrGuests):-
  repeat,
    (for(J,1,NrPeriods),
      param(Matrix,NrGuests) do
        NSq is NrGuests*NrGuests,
        once(search(Matrix[1..NrGuests,J],0,
                    first_fail,\textit{indomain_random},
                    credit(NSq,10),[]))
    ),
  !.
Observations

- Avoids deep backtracking in last time periods
- Perhaps by mixing values more evenly
- Impose fewer disequality constraints for last periods
- Easier to find solution
- Should allow to find solutions with more time periods

### Changing time periods

<table>
<thead>
<tr>
<th>Problem Size</th>
<th>Naive</th>
<th>FF</th>
<th>Layered</th>
<th>Credit</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 5</td>
<td>0.812</td>
<td>1.453</td>
<td>1.515</td>
<td>0.828</td>
<td>1.922</td>
</tr>
<tr>
<td>10 6</td>
<td>14.813</td>
<td>2.047</td>
<td>2.093</td>
<td>1.219</td>
<td>2.469</td>
</tr>
<tr>
<td>10 7</td>
<td>79.109</td>
<td>3.688</td>
<td>50.250</td>
<td>3.234</td>
<td>3.672</td>
</tr>
<tr>
<td>10 8</td>
<td>-</td>
<td>-</td>
<td>141.609</td>
<td>55.156</td>
<td>6.328</td>
</tr>
<tr>
<td>10 9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10.281</td>
</tr>
</tbody>
</table>
Observations

- Randomized method is strongest for this problem
- Not always fastest for smaller problem sizes
- Restart required for size 9 problems
- Same model, very different results due to search
- Very similar results for other problem instances

Further Improvement

- Idea: There is no real effect of including later time periods in constraint model
- Only current time period matters
- Decomposition: Set up model for one period at a time
Fine Grained Decomposition

<table>
<thead>
<tr>
<th>Old</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bin packing</td>
<td>Bin packing</td>
</tr>
<tr>
<td>Alldifferent</td>
<td>Domain restrictions</td>
</tr>
<tr>
<td>Meet at most once</td>
<td>Disequalities between guest boats</td>
</tr>
</tbody>
</table>

Guest boats = Nodes
Host boats = Colors
Disequality constraints = Edges in graph
Visualization (Time period 2)

Visualization (Time period 3)
Visualization (Time period 4)

Visualization (Time period 5)
Solving the Graph Coloring Problem

- Use disequality constraints
  - Weak propagation
- Extract \texttt{alldifferent} constraints
  - Edge clique cover problem
  - Choice of consistency method
- Use \texttt{somedifferent} global constraint
  - Heavy
  - Interaction with \texttt{bin packing} constraint
## Comparison to Comet

<table>
<thead>
<tr>
<th>Nr</th>
<th>Size</th>
<th>Solved</th>
<th>ECLiPSe 6.0</th>
<th>Comet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>100</td>
<td>0.187</td>
<td>0.343</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>100</td>
<td>0.218</td>
<td>0.515</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>100</td>
<td>0.250</td>
<td>2.469</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>100</td>
<td>0.266</td>
<td>9.906</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>100</td>
<td>0.375</td>
<td>136.828</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>100</td>
<td>0.218</td>
<td>2.375</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>100</td>
<td>0.266</td>
<td>3.453</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>100</td>
<td>0.297</td>
<td>15.421</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>100</td>
<td>0.469</td>
<td>107.187</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>100</td>
<td>0.219</td>
<td>3.266</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>100</td>
<td>0.250</td>
<td>3.734</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>100</td>
<td>0.296</td>
<td>21.360</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>100</td>
<td>1.078</td>
<td>173.266</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>100</td>
<td>0.219</td>
<td>9.222</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>100</td>
<td>0.360</td>
<td>25.297</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>100</td>
<td>0.438</td>
<td>53.547</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>63</td>
<td>3.062</td>
<td>494.109</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>100</td>
<td>0.203</td>
<td>7.922</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>100</td>
<td>0.266</td>
<td>28.000</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>100</td>
<td>0.219</td>
<td>15.219</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>100</td>
<td>0.407</td>
<td>64.312</td>
</tr>
</tbody>
</table>

---

**More Information**

- **Beldiceanu, N., Bourreau, E., Chan, P., Rivreau, D.:** Partial search strategy in CHIP. In: 2nd International Conference on Metaheuristics MIC97, Sophia Antipolis, France (1997)
- **Van Hentenryck, P., Michel, L.:** Constraint-Based Local Search. MIT Press, Boston (2005)
More Information

- **Shaw, P.**
  A constraint for bin packing.

- **Richter, Y., Freund, A., Naveh, Y.**
  Generalizing alldifferent: The somedifferent constraint.

- **Philippe Galinier, Alain Hertz, Sandrine Paroz and Gilles Pesant:**

- **Simonis, H.**
  Progress on the progressive party problem (Extended Abstract).
  *In: CP-AI-OR 2009, Pittsburgh, PA (2009)*