Topic 9: Modelling for CBLS
(Version of 25th July 2017)

Jean-Noël Monette and Pierre Flener

ASTRA Research Group
on Combinatorial Optimisation
Uppsala University
Sweden

Course 1DL449:
Modelling for Combinatorial Optimisation
Outline

1. CBLS

2. Modelling for CBLS in MiniZinc

3. Case Studies
   Warehouse Location
   Graph Colouring

4. Bibliography
Outline

1. CBLS
2. Modelling for CBLS in MiniZinc
3. Case Studies
   Warehouse Location
   Graph Colouring
4. Bibliography
Local Search

Revisit the slides on local search (LS) and constraint-based local search (CBLS) of Topic 7: Solving Technologies.
Local Search

Revisit the slides on local search (LS) and constraint-based local search (CBLS) of Topic 7: Solving Technologies.

Historically:

- No general-purpose LS solvers.
- No separation between model and search.
Local Search

Revisit the slides on local search (LS) and constraint-based local search (CBLS) of Topic 7: Solving Technologies.

Historically:
- No general-purpose LS solvers.
- No separation between model and search.

Recently:
- There are general-purpose CBLS solvers, supporting some degree of separation between model and search. **Ex**: Comet, EasyLocal++, LocalSolver, OscaR/CBLS.
- But there is still not much good-modelling practice.
Local Search from a MiniZinc Model

Our `fzn-oscar-cbls` backend for OscaR/CBLS:

- It is one of the few LS backends; it does tabu search.
- Currently, the following predicates have a specific neighbourhood: `all_different`, `circuit`, `subcircuit`, `inverse`, `global_cardinality`, `global_cardinality_closed`, `int_lin_eq`, `bool_lin_eq`, `global_cardinality_low_up`, and `global_cardinality_low_up_closed`. Note that just because a constraint-specific neighbourhood exists does not mean that it will be used.
- Currently, it ignores all the constraints flagged using `symmetry_breaking_constraint` and keeps those flagged using `implied_constraint` (see slide 8).
- Currently, there is no way to suggest a search heuristic by annotations, but we are working on it.
Outline

1. CBLS

2. Modelling for CBLS in MiniZinc

3. Case Studies
   Warehouse Location
   Graph Colouring

4. Bibliography
Make explicit as much problem structure as possible:

- Use higher-level variables whenever possible. **Examples**: Use a single integer variable with a domain of $k$ values instead of an array of $k$ Boolean variables. Use a single set variable of cardinality $k$ instead of an array of $k$ integer variables.

- Define redundant variables and the objective function by constraints expressing total functions, as those are candidate one-way constraints.

- Use predicates that have specific neighbourhoods.

- Avoid disjunction whenever possible.
There is some evidence that symmetry-breaking constraints hinder local search.

One reason might be that symmetry-breaking constraints forbid some solutions, but do not change the search space and hence do not prevent the search from moving in their direction.

It might be beneficial rather to increase the number of symmetries in a model, but this may be hard to do.
There is some evidence that symmetry-breaking constraints hinder local search.

One reason might be that symmetry-breaking constraints forbid some solutions, but do not change the search space and hence do not prevent the search from moving in their direction.

It might be beneficial rather to increase the number of symmetries in a model, but this may be hard to do.

The effect of implied constraints on local search is not really known. You need to make experiments.
Outline

1. CBLS

2. Modelling for CBLS in MiniZinc

3. Case Studies
   Warehouse Location
   Graph Colouring

4. Bibliography
Outline

1. CBLS

2. Modelling for CBLS in MiniZinc

3. Case Studies
   Warehouse Location
   Graph Colouring

4. Bibliography
Example (The Warehouse Location Problem, WLP)

A company considers opening warehouses at some candidate locations in order to supply its existing shops:

- Each candidate warehouse has the same maintenance cost.
- Each candidate warehouse has a supply capacity, which is the maximum number of shops it can supply.
- The supply cost to a shop depends on the warehouse.

Determine which warehouses to open, and which of them should supply the various shops, so that:

1. Each shop must be supplied by exactly one actually opened warehouse.
2. Each actually opened warehouse supplies at most a number of shops equal to its capacity.
3. The sum of the actually incurred maintenance costs and supply costs is minimised.
WLP: Sample Instance Data

\[
\begin{align*}
\text{Shops} &= \{\text{Shop}_1, \text{Shop}_2, \ldots, \text{Shop}_{10}\} \\
\text{Whs} &= \{\text{Berlin}, \text{London}, \text{Ankara}, \text{Paris}, \text{Rome}\} \\
\text{maintCost} &= 30
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>Berlin</th>
<th>London</th>
<th>Ankara</th>
<th>Paris</th>
<th>Rome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SuppliesCost</th>
<th>Berlin</th>
<th>London</th>
<th>Ankara</th>
<th>Paris</th>
<th>Rome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shop_1</td>
<td>20</td>
<td>24</td>
<td>11</td>
<td>25</td>
<td>30</td>
</tr>
<tr>
<td>Shop_2</td>
<td>28</td>
<td>27</td>
<td>82</td>
<td>83</td>
<td>74</td>
</tr>
<tr>
<td>Shop_3</td>
<td>74</td>
<td>97</td>
<td>71</td>
<td>96</td>
<td>70</td>
</tr>
<tr>
<td>Shop_4</td>
<td>2</td>
<td>55</td>
<td>73</td>
<td>69</td>
<td>61</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Shop_{10}</td>
<td>47</td>
<td>65</td>
<td>55</td>
<td>71</td>
<td>95</td>
</tr>
</tbody>
</table>

There is an optimal solution where \(\text{Shop}_4\) is supplied by Rome, not by Berlin.
WLP Model 1: Variables (reminder)

Automatic enforcement of the total-function constraint (1):

\[
\text{Supplier} = \begin{array}{cccc}
\text{Shop}_1 & \text{Shop}_2 & \cdots & \text{Shop}_{10} \\
\in \text{Whs} & \in \text{Whs} & \cdots & \in \text{Whs}
\end{array}
\]

\text{Supplier}[s] \text{ denotes the supplier warehouse for shop } s.

Redundant decision variables:

\[
\text{Open} = \begin{array}{ccccc}
\text{Berlin} & \text{London} & \text{Ankara} & \text{Paris} & \text{Rome} \\
\in 0..1 & \in 0..1 & \in 0..1 & \in 0..1 & \in 0..1
\end{array}
\]

\text{Open}[w]=1 \text{ if and only if (iff) warehouse } w \text{ is opened.}
WLP Model 1: Objective & Cons. (reminder)

Objective (3):

\[
\text{minimize} \quad \text{maintCost} \times \sum (\text{Open}) \\
+ \sum (s \text{ in Shops})(\text{SupplyCost}[s,\text{Supplier}[s]])
\]

One-way channelling constraint:

\[
\text{forall}(s \text{ in Shops})(\text{Open} [\text{Supplier}[s]] = 1)
\]

Capacity constraint (2):

\[
\text{global_cardinality_low_up_closed} \\
(\text{Supplier}, \text{Whs}, [0 \mid i \text{ in Whs}], \text{Capacity})
\]
WLP Model 1: Objective & Cons. (reminder)

Objective (3): a total function

\[
\text{minimize} \\
\quad \text{maintCost} \times \sum (\text{Open}) \\
\quad + \\
\quad \sum (s \text{ in Shops})(\text{SupplyCost}[s,\text{Supplier}[s]])
\]

One-way channelling constraint: not a total function

\[
\forall (s \text{ in Shops})(\text{Open}[	ext{Supplier}[s]] = 1)
\]

Capacity constraint (2): a specific neighbourhood exists

\[
\text{global_cardinality_low_up_closed} \\
(Supplier, \text{Whs}, [0 \mid i \text{ in Whs}], \text{Capacity})
\]
Replace the one-way channelling by the two-way one:

\[
\text{forall}(w \text{ in } \text{Whs}) \quad (\text{Open}[w] = \text{bool2int}(\exists s \text{ in } \text{Shops})(\text{Supplier}[s]=w))
\]

The \text{Open}[w] variables are now defined by a total function.
Replace the one-way channelling by the two-way one:

\[
\text{forall}(w \in \text{Whs}) \quad (\text{Open}[w] = \text{bool2int}(\exists s \in \text{Shops})(\text{Supplier}[s]=w))
\]

The \text{Open}[w] variables are now defined by a total function.

Search may now be performed only on the \text{Supplier}[s] decision variables, using the specific neighbourhood for \text{global_cardinality_low_up_closed}. 

---

Finding a known-to-be optimal solution to the instance of slide 12, using \text{fzn-oscar-cbls}, averaged over 5 runs:

<table>
<thead>
<tr>
<th>Model</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1 with one-way channelling</td>
<td>3.88</td>
</tr>
<tr>
<td>Model 1 with two-way channelling</td>
<td>0.97</td>
</tr>
</tbody>
</table>

---

M4CO
Replace the one-way channelling by the two-way one:

\[
\text{forall}(w \text{ in Whs})
(\text{Open}[w] = \text{bool2int}(\exists(s \text{ in Shops})(\text{Supplier}[s]=w)))
\]

The \( \text{Open}[w] \) variables are now defined by a total function.

Search may now be performed only on the \( \text{Supplier}[s] \) decision variables, using the specific neighbourhood for \texttt{global_cardinality_low_up_closed}.

Finding a known-to-be optimal solution to the instance of slide 12, using \texttt{fzn-oscar-cbls}, averaged over 5 runs:

<table>
<thead>
<tr>
<th>Model</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1 with one-way channelling</td>
<td>3.88</td>
</tr>
<tr>
<td>Model 1 with two-way channelling</td>
<td>0.97</td>
</tr>
</tbody>
</table>
Outline

1. CBLS
2. Modelling for CBLS in MiniZinc
3. Case Studies
   - Warehouse Location
   - Graph Colouring
4. Bibliography
The Graph Colouring Problem

Given a graph \((V, E)\), colour each vertex so that adjacent vertices have different colours, using at most \(n\) colours.

Model 1, MIP-style

Variable \(C[v, k] \in 0..1\) is 1 iff vertex \(v\) has colour \(k\).

1. One colour per vertex:

\[
\forall v \in V \left( \sum_{k \in 1..n} C[v, k] = 1 \right)
\]

\[
\forall (u, v) \in E, k \in 1..n \left( C[u, k] + C[v, k] \leq 1 \right)
\]

Depending on the used moves, such as changing the value of one variable, the constraint (1) might be violated.
The Graph Colouring Problem

Given a graph \((V, E)\), colour each vertex so that adjacent vertices have different colours, using at most \(n\) colours.

Model 1, MIP-style

Variable \(C[v, k]\) in 0..1 is 1 iff vertex \(v\) has colour \(k\).

1. One colour per vertex:

\[
\text{forall}(v \text{ in } V)(\text{sum}(C[v, \ldots]) = 1)
\]

2. Different colours on edge ends:
The Graph Colouring Problem

Given a graph \((V,E)\), colour each vertex so that adjacent vertices have different colours, using at most \(n\) colours.

Model 1, MIP-style

Variable 

\[
C[v,k] \text{ in } 0..1 \text{ is } 1 \text{ iff vertex } v \text{ has colour } k.
\]

1. One colour per vertex:

\[
\text{forall } (v \text{ in } V) (\text{sum}(C[v,..]) = 1)
\]

2. Different colours on edge ends:

\[
\text{forall } ((u,v) \text{ in } E, k \text{ in } 1..n) \quad (C[u,k] + C[v,k] \leq 1)
\]

Depending on the used moves, such as changing the value of one variable, the constraint (1) might be violated.
Given a graph \((V, E)\), colour each vertex so that adjacent vertices have different colours, using at most \(n\) colours.

Model 2, MiniZinc/CP/LCG-style

Variable \(C[v]\) in \(1..n\) is \(k\) iff vertex \(v\) has colour \(k\).

1. One colour per vertex:
Given a graph \((V, E)\), colour each vertex so that adjacent vertices have different colours, using at most \(n\) colours.

**Model 2, MiniZinc/CP/LCG-style**

Variable \(C[v]\) in 1..\(n\) is \(k\) iff vertex \(v\) has colour \(k\).

1. One colour per vertex: enforced by choice of variables.
2. Different colours on edge ends:
Given a graph \((V,E)\), colour each vertex so that adjacent vertices have different colours, using at most \(n\) colours.

### Model 2, MiniZinc/CP/LCG-style

Variable \(C[v]\) in 1..\(n\) is \(k\) iff vertex \(v\) has colour \(k\).

1. One colour per vertex: enforced by choice of variables.
2. Different colours on edge ends:
   
   \[
   \text{forall}((u,v) \in E)(C[u] \neq C[v])
   \]

The search only needs to satisfy the constraint (2).
The Graph Colouring *Optimisation* Problem

Given a graph \((V, E)\), colour each vertex so that adjacent vertices have different colours, using a minimum of colours.
The Graph Colouring Optimisation Problem

Given a graph \((V, E)\), colour each vertex so that adjacent vertices have different colours, using a minimum of colours.

Model 2.1

Variable \(C[v]\) in \(1..\text{card}(V)\) is the colour of vertex \(v\).

Minimise variable \(N\), the number of possibly used colours:

1. One colour per vertex: enforced by choice of variables.
2. Different colours on edge ends:
   \[
   \forall ((u,v) \in E) (C[u] \neq C[v])
   \]
3. Objective var:
The Graph Colouring *Optimisation* Problem

Given a graph \((V, E)\), colour each vertex so that adjacent vertices have different colours, using a minimum of colours.

**Model 2.1**

Variable \(C[v]\) in \(1 \ldots \text{card}(V)\) is the colour of vertex \(v\). Minimise variable \(N\), the number of *possibly used* colours:

1. One colour per vertex: enforced by choice of variables.
2. Different colours on edge ends:
   
   \[
   \text{forall}((u,v) \text{ in } E)(C[u] \neq C[v])
   \]

3. Objective var: \(N = \text{max}(C)\), a total-function constraint

The variable \(N\) in \(1 \ldots \text{card}(V)\) represents the highest used colour: in a minimal solution, the \(N\) first colours are used and \(N\) is the number of *actually used* colours.
Despite $N$ being total-function defined, Model 2.1 poorly drives the search as there is no way to distinguish between a candidate solution with only one variable equal to $N$ and a candidate solution with several variables equal to $N$:

**Example** ($N = \max(C)$)

Assume currently $C[1..8] = [1, 2, 3, 4, 2, 3, 1, 4]$, with $N=4$ if constraint (3) becomes a one-way constraint:

1. The move $C[8] := 3$ to $[1, 2, 3, 4, 2, 3, 1, 3]$ keeps $N=4$ and one move, namely $C[4] := 1$, suffices to reach the assumed minimal solution $[1, 2, 3, 1, 2, 3, 1, 3]$ with $N=3$. 
Despite $N$ being total-function defined, Model 2.1 poorly drives the search as there is no way to distinguish between a candidate solution with only one variable equal to $N$ and a candidate solution with several variables equal to $N$:

**Example ($N = \max(C)$)**

Assume currently $C[1..8] = [1, 2, 3, 4, 2, 3, 1, 4]$, with $N=4$ if constraint (3) becomes a one-way constraint:

1. **The move** $C[8] := 3$ to $[1, 2, 3, 4, 2, 3, 1, 3]$ keeps $N=4$ and one move, namely $C[4] := 1$, suffices to reach the assumed minimal solution $[1, 2, 3, 1, 2, 3, 1, 3]$ with $N=3$.

Despite $N$ being total-function defined, Model 2.1 poorly drives the search as there is no way to distinguish between a candidate solution with only one variable equal to $N$ and a candidate solution with several variables equal to $N$:

**Example ($N = \max(C)$)**

Assume currently $C[1..8] = [1, 2, 3, 4, 2, 3, 1, 4]$, with $N=4$ if constraint (3) becomes a one-way constraint:

1. The move $C[8] := 3$ to $[1, 2, 3, 4, 2, 3, 1, 3]$ keeps $N=4$ and one move, namely $C[4] := 1$, suffices to reach the assumed minimal solution $[1, 2, 3, 1, 2, 3, 1, 3]$ with $N=3$.


These two moves are not distinguished by the search, although the first move is better than the second one.
A better model bounds the number of colours by $N$:

**Model 2.2**

Model 2.1, except:

3. Objective variable:

```plaintext
forall(v in V)(C[v] <= N)
```

Now the search is better driven, as the violation of the non-total-function constraint (3) depends on the number of variables with a value larger than $N$:

This is a counter-example to a rule of thumb on slide 7: the objective variable is not functionally defined anymore.

This slide reflects the current version of `fzn-oscar-cbls`; in future versions or in other (CB)LS backends, Model 2.1 might be better handled than Model 2.2.
A better model bounds the number of colours by $N$:

**Model 2.2**

Model 2.1, except:

3. **Objective variable**: $\forall v \in V \ (C[v] \leq N)$

Now the search is better driven, as the violation of the non-total-function constraint (3) depends on the number of variables with a value larger than $N$: see the next slide.
A better model bounds the number of colours by $N$:

**Model 2.2**

Model 2.1, except:

1. Objective variable: $\forall (v \in V) (C[v] \leq N)$

Now the search is better driven, as the violation of the non-total-function constraint (3) depends on the number of variables with a value larger than $N$: see the next slide.

This is a counter-example to a rule of thumb on slide 7: the objective variable is not functionally defined anymore.
A better model bounds the number of colours by $N$:

**Model 2.2**

Model 2.1, except:

1. **Objective variable**: $\forall (v \in V) (C[v] \leq N)$

Now the search is better driven, as the violation of the non-total-function constraint (3) depends on the number of variables with a value larger than $N$: see the next slide.

This is a counter-example to a rule of thumb on slide 7: the objective variable is not functionally defined anymore.

This slide reflects the current version of `fzn-oscar-cbls`; in future versions or in other (CB)LS backends, Model 2.1 might be better handled than Model 2.2.
Assume currently $C[1..8] = [1, 2, 3, 4, 2, 3, 1, 4]$, but $N=3$, as $N$ is not fixed by the constraint (3), hence two conjuncts of (3) are violated, namely $C[4] <= N$ and $C[8] <= N$, with the violation $(4 - 3) + (4 - 3) = 2$ of (3):
Example (\texttt{forall(v in V)(C[v] <= N)})

Assume currently \( C[1..8] = [1, 2, 3, 4, 2, 3, 1, 4] \), but \( N=3 \), as \( N \) is not fixed by the constraint (3), hence two conjuncts of (3) are violated, namely \( C[4] \leq N \) and \( C[8] \leq N \), with the violation \((4 - 3) + (4 - 3) = 2\) of (3):

1. The move \( C[8] := 3 \) to \( [1, 2, 3, 4, 2, 3, 1, 3] \) keeps \( N=3 \) but only one conjunct of (3) is now violated, namely \( C[4] \leq N \), with the violation \( 4 - 3 = 1 \) of (3).
Example \( \text{forall}(v \in V)(C[v] \leq N) \)

Assume currently \( C[1..8] = [1, 2, 3, 4, 2, 3, 1, 4] \), but \( N=3 \), as \( N \) is not fixed by the constraint (3), hence two conjuncts of (3) are violated, namely \( C[4] \leq N \) and \( C[8] \leq N \), with the violation \( (4 - 3) + (4 - 3) = 2 \) of (3):

1. **The move** \( C[8] := 3 \) to \( [1, 2, 3, 4, 2, 3, 1, 3] \) keeps \( N=3 \) but only **one** conjunct of (3) is now violated, namely \( C[4] \leq N \), with the violation \( 4 - 3 = 1 \) of (3).

2. **The move** \( C[6] := 4 \) to \( [1, 2, 3, 4, 2, 4, 1, 4] \) keeps \( N=3 \) but **three** conjuncts of (3) are now violated, namely \( C[4] \leq N \) and \( C[6] \leq N \) and \( C[8] \leq N \), with the violation \( (4 - 3) + (4 - 3) + (4 - 3) = 3 \) of (3).

In order to reach the assumed minimal solution \( [1, 2, 3, 1, 2, 3, 1, 3] \) with \( N=3 \), probing the first move reveals a decrease by 1 of the violation of (3), while probing the second move reveals an increase by 1 of that violation, hence the first move is now preferred over the second one.
Example \( \text{(forall}\ v \in V \ (C[v] \leq N)) \)

Assume currently \( C[1..8] = [1, 2, 3, 4, 2, 3, 1, 4] \), but \( N=3 \), as \( N \) is not fixed by the constraint (3), hence two conjuncts of (3) are violated, namely \( C[4] \leq N \) and \( C[8] \leq N \), with the violation \((4 - 3) + (4 - 3) = 2\) of (3):

1. The move \( C[8] := 3 \) to \([1, 2, 3, 4, 2, 3, 1, 3]\) keeps \( N=3 \) but only one conjunct of (3) is now violated, namely \( C[4] \leq N \), with the violation \( 4 - 3 = 1 \) of (3).

2. The move \( C[6] := 4 \) to \([1, 2, 3, 4, 2, 4, 1, 4]\) keeps \( N=3 \) but three conjuncts of (3) are now violated, namely \( C[4] \leq N \) and \( C[6] \leq N \) and \( C[8] \leq N \), with the violation \((4 - 3) + (4 - 3) + (4 - 3) = 3\) of (3).

In order to reach the assumed minimal solution \([1, 2, 3, 1, 2, 3, 1, 3]\) with \( N=3 \), probing the first move reveals a decrease by 1 of the violation of (3), while probing the second move reveals an increase by 1 of that violation, hence the first move is now preferred over the second one.
We implicitly broke some value symmetries in the previous models: if there is a solution with $N$ colours, then it does not matter whether those are the values $1 \ldots N$ or not.

**Model 2.3**

Model 2.1, except:

3. Objective var:
We implicitly broke some value symmetries in the previous models: if there is a solution with \( N \) colours, then it does not matter whether those are the values 1..\( N \) or not.

### Model 2.3

Model 2.1, except:

1. **Objective var:** \( N = nvalue(C) \), a total-fct constraint

The local search may again be poorly driven, in the same way as with Model 2.1: see the next slide.
Example ($N = nvalue(C)$)

Assume currently $C[1..8] = [1, 2, 3, 4, 2, 3, 1, 4]$, with $N=4$ if constraint (3) becomes a one-way constraint:

1. The move $C[8] := 3$ to $[1, 2, 3, 4, 2, 3, 1, 3]$ keeps $N=4$ and one move, namely $C[4] := 1$, suffices to reach the assumed minimal solution $[1, 2, 3, 1, 2, 3, 1, 3]$ with $N=3$.

2. The move $C[6] := 4$ to $[1, 2, 3, 4, 2, 4, 1, 4]$ keeps $N=4$ but two moves, namely $C[3] := 4$ and $C[4] := 1$, are required to reach the value-symmetric variant $[1, 2, 4, 1, 2, 4, 1, 4]$ of the assumed minimal solution $[1, 2, 3, 1, 2, 3, 1, 3]$ with $N=3$.

These two moves are not distinguished by the search, although the first move is better than the second one.
Outline

1. CBLS

2. Modelling for CBLS in MiniZinc

3. Case Studies
   Warehouse Location
   Graph Colouring

4. Bibliography
Bibliography

Björdal, Gustav; Monette, Jean-Noël; Flener, Pierre; and Pearson, Justin.  
A constraint-based local search backend for MiniZinc.  

Hoos, Holger H. and Stützle, Thomas.  
*Stochastic Local Search: Foundations & Applications*.  

Van Hentenryck, Pascal and Michel, Laurent.  
*Constraint-Based Local Search*.  

Prestwich, Steven.  
*Supersymmetric modeling for local search*.  
SymCon 2002.