Chapter 8: Symmetry Breaking (Balanced Incomplete Block Designs)

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

ECLiPSe ELEarning
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.
Outline

1. Problem
2. Program
3. Symmetry Breaking
What we want to introduce

- BIBD - Balanced Incomplete Block Designs
- Using lex constraints to remove symmetries
- Only one of many ways to deal with symmetry in problems
- Finding all solutions to a problem
- Using timeout to limit search
Outline

1 Problem

2 Program

3 Symmetry Breaking
Problem Definition

BIBD (Balanced Incomplete Block Design)

A BIBD is defined as an arrangement of $v$ distinct objects into $b$ blocks such that each block contains exactly $k$ distinct objects, each object occurs in exactly $r$ different blocks, and every two distinct objects occur together in exactly $\lambda$ blocks. A BIBD is therefore specified by its parameters $(v, b, r, k, \lambda)$. 

Problem

Program

Symmetry Breaking

Helmut Simonis

Symmetry Breaking
Motivation: Test Planning

Consider a new release of some software with $v$ new features. You want to regression test the software against combinations of the new features. Testing each subset of features is too expensive, so you want to run $b$ tests, each using $k$ features. Each feature should be used $r$ times in the tests. Each pair of features should be tested together exactly $\lambda$ times. How do you arrange the tests?
Another way of defining a BIBD is in terms of its incidence matrix, which is a binary matrix with \( v \) rows, \( b \) columns, \( r \) ones per row, \( k \) ones per column, and scalar product \( \lambda \) between any pair of distinct rows.
Another way of defining a BIBD is in terms of its incidence matrix, which is a binary matrix with $v$ rows, $b$ columns, $r$ ones per row, $k$ ones per column, and scalar product $\lambda$ between any pair of distinct rows.

A (6,10,5,3,2) BIBD
Outline

1 Problem
2 Program
3 Symmetry Breaking
A binary $v \times b$ matrix. Entry $V_{ij}$ states if item $i$ is in block $j$.

- Sum constraints over rows, each sum equal $r$
- Sum constraints over columns, each sum equal $k$
- Scalar product between any pair of rows, the product value is $\lambda$. 

Model for $(v, b, r, k, \lambda)$ BIBD
:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    bibd(6,10,5,3,2,Matrix),writeln(Matrix).

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),
    extract_array(row,Matrix,List),
    search(L,0,input_order,indomain,
        complete,[[]]).
:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    bibd(6,10,5,3,2,Matrix), writeln(Matrix).

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix), Set up model
    extract_array(row,Matrix,List),
    search(L,0,input_order,indomain, complete,[]).
:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
bibd(6,10,5,3,2,Matrix), writeln(Matrix).

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),
    extract_array(row, Matrix, List),
    search(List, 0, input_order, indomain, complete, []).
:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    bibd(6,10,5,3,2,Matrix),writeln(Matrix).

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),
    extract_array(row,Matrix,List),
    search(L,0,input_order,indomain,
        complete,[]).⇒ Search
model(V,B,R,K,L,Matrix,Method):-
    dim(Matrix, [V,B]),
    Matrix[1..V,1..B] :: 0..1,
    (for(I,1,V), param(Matrix,B,R) do 
        sumlist(Matrix[I,1..B],R)
    ),
    (for(J,1,B), param(Matrix,V,K) do 
        sumlist(Matrix[1..V,J],K)
    ),
    (for(I,1,V-1), param(Matrix,V,B,L) do 
        (for(I1,I+1,V), param(Matrix,I,B,L) do 
            scalar_product(Matrix[I,1..B], 
                Matrix[I1,1..B],L)
        )
    ).
model(V,B,R,K,L,Matrix,Method):-
  dim(Matrix,[V,B]), % Define Binary Matrix
  Matrix[1..V,1..B] :: 0..1,
  (for(I,1,V), param(Matrix,B,R) do
    sumlist(Matrix[I,1..B],R)
  ),
  (for(J,1,B), param(Matrix,V,K) do
    sumlist(Matrix[1..V,J],K)
  ),
  (for(I,1,V-1), param(Matrix,V,B,L) do
    (for(I1,I+1,V), param(Matrix,I,B,L) do
      scalar_product(Matrix[I,1..B],
      Matrix[I1,1..B],L)
    )
  ).
model(V,B,R,K,L,Matrix,Method):-
dim(Matrix,[V,B]),
Matrix[1..V,1..B] :: 0..1,
(for(I,1,V), param(Matrix,B,R) do
  sumlist(Matrix[I,1..B],R)
),\[ Row \text{ Sum } = R\]
(for(J,1,B), param(Matrix,V,K) do
  sumlist(Matrix[1..V,J],K)
),
(for(I,1,V-1), param(Matrix,V,B,L) do
  (for(I1,I+1,V), param(Matrix,I,B,L) do
   scalar_product(Matrix[I,1..B],
               Matrix[I1,1..B],L)
  )
).

Helmut Simonis
model(V,B,R,K,L,Matrix,Method):-
   dim(Matrix,[V,B]),
   Matrix[1..V,1..B] :: 0..1,
   (for(I,1,V), param(Matrix,B,R) do
      sumlist(Matrix[I,1..B],R)
   ),
   (for(J,1,B), param(Matrix,V,K) do
      sumlist(Matrix[1..V,J],K)
   ),\ Column Sum = K
   (for(I,1,V-1), param(Matrix,V,B,L) do
      (for(I1,I+1,V), param(Matrix,I,B,L) do
         scalar_product(Matrix[I,1..B],
                        Matrix[I1,1..B],L)
      )
   )).
model(V,B,R,K,L,Matrix,Method):-
    dim(Matrix,[V,B]),
    Matrix[1..V,1..B] :: 0..1,
    (for(I,1,V), param(Matrix,B,R) do
        sumlist(Matrix[I,1..B],R)
    ),
    (for(J,1,B), param(Matrix,V,K) do
        sumlist(Matrix[1..V,J],K)
    ),
    (for(I,1,V-1), param(Matrix,V,B,L) do
        (for(I1,I+1,V), param(Matrix,I,B,L) do
            scalar_product(Matrix[I,1..B],
                            Matrix[I1,1..B],L)
        )
    ).

⇒ Scalar product between all rows.
scalar_product(XVector, YVector, V):-
collection_to_list(XVector, XList),
collection_to_list(YVector, YList),
(foreach(X, XList),
    foreach(Y, YList),
    fromto(0, A, A1, Term) do
        A1 = A + X * Y
),
eval(Term) #= V.
scalar_product(XVector, YVector, V):-
collection_to_list(XVector, XList),
collection_to_list(YVector, YList),
(foreach(X, XList),
 foreach(Y, YList),
 fromto(0, A, A1, Term) do
   A1 = A + X * Y
 ),
eval(Term) #= V.
scalar_product(XVector, YVector, V):-
collection_to_list(XVector, XList),
collection_to_list(YVector, YList),
(foreach(X, XList), \(\Rightarrow\) Iterate over lists
  foreach(Y, YList), \(\Rightarrow\) ...in parallel
  fromto(0, A, A1, Term) do
    A1 = A + X*Y
  ),
eval(Term) #= V.
scalar_product(XVector, YVector, V):-
collection_to_list(XVector, XList),
collection_to_list(YVector, YList),
(foreach(X, XList),
  foreach(Y, YList),
  fromto(0, A, A1, Term) do Build term
    A1 = A + X * Y
  ),
eval(Term) #= V.
scalar_product(XVector, YVector, V):-
collection_to_list(XVector, XList),
collection_to_list(YVector, YList),
(foreach(X, XList),
  foreach(Y, YList),
  fromto(0, A, A1, Term) do
    A1 = A + X * Y) ↓ Construct term
),
eval(Term) #= V.
scalar_product(XVector, YVector, V):-
  collection_to_list(XVector, XList),
  collection_to_list(YVector, YList),
  (foreach(X, XList),
    foreach(Y, YList),
    fromto(0, A, A1, Term) do
      A1 = A + X * Y
  ),
  eval(Term) #= V.⇒ State Constraint
Search Routine

- Static variable order
- First fail does not work for binary variables
- Enumerate variables by row
- Use utility predicate `extract_array/3`
- Assign with `indomain`, try value 0, then value 1
- Use simple `search` call
Basic Model - First Solution

1
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution

The image shows a grid with some cells highlighted. The grid has rows and columns labeled from 0 to 11. Several cells are marked with different colors, indicating a solution to a problem under the context of Symmetry Breaking.
Basic Model - First Solution
Basic Model - First Solution

![Diagram of a grid with selected cells highlighted]
Basic Model - First Solution
## Basic Model - First Solution

- **Problem:**
- **Program:**
- **Symmetry Breaking**

### Basic Model - First Solution Diagram

![Diagram](image-url)
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution

Problem
Program
Symmetry Breaking
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Problem
Program
Symmetry Breaking

Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution

Diagram showing a tree structure with nodes and branches, and a grid with cells shaded in various colors. The grid has a selected cell marked with a red square.
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution

[Diagram of a tree structure with a canvas on the right side, showing a grid with selected cells highlighted in green.]

- Back to Start
- Skip Animation
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution

Problem
Program
Symmetry Breaking

Helmut Simonis  Symmetry Breaking  144
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Basic Model - First Solution
Finding all solutions - Hack!

:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    bibd(6,10,5,3,2,Matrix), writeln(Matrix), fail.

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),
    extract_array(row,Matrix,List),
    search(L,0,input_order,indomain, complete, []).
Finding all solutions - Hack!

```prolog
:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    bibd(6,10,5,3,2,Matrix),writeln(Matrix),
    fail.  \ Force Backtracking

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),
    extract_array(row,Matrix,List),
    search(L,0,input_order,indomain,complete,[]).
```

Helmut Simonis
Symmetry Breaking
:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    findall(Matrix,bibd(6,10,5,3,2,Matrix),Sols),
    writeln(Sols).

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),
    extract_array(row,Matrix,List),
    search(L,0,input_order,indomain,
        complete,[]).
Finding all solutions - Proper

:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    findall(Matrix,bibd(6,10,5,3,2,Matrix),Sols),
    writeln(Sols).

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),
    extract_array(row,Matrix,List),
    search(L,0,input_order,indomain,
        complete,[]).
**findall** predicate

- **findall(Template, Goal, Collection)**
- **Finds all solutions to** \( \text{Goal} \) **and collects them into a list** \( \text{Collection} \)
- **Template** is used to extract arguments from Goal to store as solution
- **Backtracks through all choices in** Goal
- **Solutions are returned in order in which they are found**
Problem

- Program now only stops when it has found all solutions
- This takes too long!
- How can we limit the amount of time to wait?
- Use of the `timeout` library
Finding all solutions - Proper

:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).
:-lib(timeout).

top:-
    findall(Matrix, timeout(bibd(6,10,5,3,2,Matrix), 10, fail), Sols),
    writeln(Sols).
Finding all solutions - Proper

:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).
:-lib(timeout).

\[ \text{Load library} \]

\[
\text{top:- findall}(\text{Matrix}, \text{timeout}(\text{bibd}(6,10,5,3,2,\text{Matrix}), 10, \text{fail}), \text{Sols}), \text{writeln}(\text{Sols}).
\]
Finding all solutions - Proper

:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).
:-lib(timeout).

top:-
    findall(Matrix, timeout(bibd(6,10,5,3,2,Matrix), 10, \(\Rightarrow\) seconds fail), Sols),
    writeln(Sols).

Helmut Simonis
Symmetry Breaking
timeout library

- `timeout(Goal, Limit, TimeoutGoal)`
- Runs Goal for Limit seconds
- If Limit is reached, Goal is stopped and TimeoutGoal is run instead
- If Limit is not reached, it has no impact
- Must load :-lib(timeout).
Finding all Solutions - Search Tree 200 Nodes
Observation

- Surprise! There are many solutions
Search Tree 300 Nodes
Search Tree 400 Nodes
Search Tree 500 Nodes
Search Tree 1000 Nodes
Search Tree 2000 Nodes
Problem

- There are too many solutions to collect in a reasonable time
- Most of these solutions are very similar
- If you take one solution and
  - exchange two rows
  - and/or exchange two columns
- ... you have another solution
- Can we avoid exploring them all?
Outline

1. Problem
2. Program
3. Symmetry Breaking
   - Experiment with alternative value order
Symmetry Breaking Techniques

- Remove all symmetries
  - Reduce the search tree as much as possible
  - May be hard to describe all symmetries
  - May be expensive to remove symmetric parts of tree

- Remove some symmetries
  - Search is not reduced as much
  - May be easier to find some symmetries to remove
  - Cost can be low
Symmetry Breaking Techniques

- Remove all symmetries
  - Reduce the search tree as much as possible
  - May be hard to describe all symmetries
  - May be expensive to remove symmetric parts of tree

- Remove some symmetries
  - Search is not reduced as much
  - May be easier to find some symmetries to remove
  - Cost can be low
Symmetry Breaking Techniques

- Symmetry removal by forcing partial, initial assignment
  - Easy to understand
  - Rather weak, does not affect search

- Symmetry removal by stating constraints
  - Removing all symmetries may require exponential number of constraints
  - Can conflict with search strategies

- Symmetry removal by controlling search
  - At each node, decide if it needs to be explored
  - Can be expensive to check
Symmetry Breaking Techniques

- Symmetry removal by forcing partial, initial assignment
  - Easy to understand
  - Rather weak, does not affect search

- Symmetry removal by stating constraints
  - Removing all symmetries may require exponential number of constraints
  - Can conflict with search strategies

- Symmetry removal by controlling search
  - At each node, decide if it needs to be explored
  - Can be expensive to check
Solution used here: Double Lex

- Partial symmetry removal by adding lexicographical ordering constraints
- Our problem has full row and column symmetries
- Any permutation of rows and/or columns leads to another solution
- Idea: Order rows lexicographically
- Rows must be different from each other, strict order on rows
- Columns might be identical, non-strict order on columns
  - This can be improved in some cases
- Constraints only between adjacent rows(columns)
Added Constraints

\[
\text{dim}(\text{Matrix}, [V,B]),
\]
\[
(\text{for}(I, 1, V-1),
\quad \text{param}(\text{Matrix}, B) \text{ do }
\quad I1 \text{ is } I+1,
\quad \text{lex}_{\text{less}}(\text{Matrix}[I1,1..B],\text{Matrix}[I,1..B])
\),
\]
\[
(\text{for}(J, 1, B-1),
\quad \text{param}(\text{Matrix}, V) \text{ do }
\quad J1 \text{ is } J+1,
\quad \text{lex}_{\text{leq}}(\text{Matrix}[1..V,J1],\text{Matrix}[1..V,J])
\),
\]
Added Constraints

\[ \text{dim}(\text{Matrix}, [V, B]), \]
\( (\text{for}(I, 1, V-1), \)
\( \text{param}(\text{Matrix}, B) \text{ do} \)
\( \quad I_1 \text{ is } I+1, \)
\( \quad \text{lex}_\text{less}(\text{Matrix}[I_1, 1..B], \text{Matrix}[I, 1..B]) \)
\), \( \Rightarrow \) \text{Row lex constraints} \)
\( (\text{for}(J, 1, B-1), \)
\( \quad \text{param}(\text{Matrix}, V) \text{ do} \)
\( \quad J_1 \text{ is } J+1, \)
\( \quad \text{lex}_\text{leq}(\text{Matrix}[1..V, J_1], \text{Matrix}[1..V, J]) \)
\),
Added Constraints

dim(Matrix, [V,B]),
(for(I,1,V-1),
  param(Matrix,B) do
    I1 is I+1,
    lex_less(Matrix[I1,1..B], Matrix[I,1..B])
),
(for(J,1,B-1),
  param(Matrix,V) do
    J1 is J+1,
    lex_leq(Matrix[1..V,J1], Matrix[1..V,J])
)
⇒ Column lex constraints
Using Two Global Constraints

- **lex_leq(List1, List2)**
  - List1 is lexicographical smaller than or equal to List2
  - Achieves domain consistency

- **lex_less(List1, List2)**
  - List1 is lexicographical smaller than List2
  - Achieves domain consistency
Example propagation \texttt{lex\_less}

\begin{verbatim}
lex_less(
[ 2, X2 \in \{1, 3, 4\}, X3 \in \{1, 2, 3\}, X4 \in \{1, 2\}, X5 \in \{3, 4\},
[ Y1 \in \{0, 1, 2\}, 1, Y3 \in \{0, 1, 2, 3\}, Y4 \in \{0, 1\}, Y5 \in \{0, 1\}]\end{verbatim}
Example propagation \texttt{lex\_less}

\begin{verbatim}
lex_less(
    [ 2, X2 ∈ \{1, 3, 4\}, X3 ∈ \{1, 2, 3\}, X4 ∈ \{1, 2\}, X5 ∈ \{3, 4\},
    [ Y1 ∈ \{0, 1, 2\}, 1, Y3 ∈ \{0, 1, 2, 3\}, Y4 ∈ \{0, 1\}, Y5 ∈ \{0, 1\}])
\end{verbatim}
Example propagation \textit{lex\_less}

\begin{verbatim}
lex_less(
[ 2, X2 \in \{1, 3, 4\}, X3 \in \{1, 2, 3\}, X4 \in \{1, 2\}, X5 \in \{3, 4\}],
[ 2, 1, Y3 \in \{0, 1, 2, 3\}, Y4 \in \{0, 1\}, Y5 \in \{0, 1\}]\)
\end{verbatim}
Example propagation `lex_less`

```
lex_less(
[ 2,
 X2 ∈ {1, 3, 4}, X3 ∈ {1, 2, 3}, X4 ∈ {1, 2}, X5 ∈ {3, 4}],
[ 2, Y3 ∈ {0, 1, 2, 3}, Y4 ∈ {0, 1}, Y5 ∈ {0, 1}]
)
```
Example propagation \texttt{lex\_less}

\begin{verbatim}
lex_less(
    [ 2, 1, X3 ∈ \{1, 2, 3\}, X4 ∈ \{1, 2\}, X5 ∈ \{3, 4\}],
    [ 2, 1, Y3 ∈ \{0, 1, 2, 3\}, Y4 ∈ \{0, 1\}, Y5 ∈ \{0, 1\}]
)
\end{verbatim}
Example propagation \textit{lex} \textit{less}

\begin{verbatim}
lex_less(
[ 2, 1, X3 \in \{1, 2, 3\}, X4 \in \{1, 2\}, X5 \in \{3, 4\}],
[ 2, 1, Y3 \in \{0, 1, 2, 3\}, Y4 \in \{0, 1\}, Y5 \in \{0, 1\}])
\end{verbatim}
Example propagation \texttt{lex\_less}

\begin{verbatim}
lex_less(
[ 2, 1, X3 ∈ \{1, 2, 3\}, X4 ∈ \{1, 2\}, X5 ∈ \{3, 4\}],
[ 2, 1, Y3 ∈ \{1, 2, 3\}, Y4 ∈ \{0, 1\}, Y5 ∈ \{0, 1\}])
\end{verbatim}
Example propagation \texttt{lex\_less}

\begin{verbatim}
lex_less(
[  2, 1, \text{X3} \in \{1, 2, 3\}, X4 \in \{1, 2\}, X5 \in \{3, 4\}],
[  2, 1, \text{Y3} \in \{1, 2, 3\}, Y4 \in \{0, 1\}, Y5 \in \{0, 1\}]\n)
\end{verbatim}
Example propagation \textit{lex\_less}

\[
\text{lex\_less(}
\begin{bmatrix}
2, & 1,
\end{bmatrix},
\begin{bmatrix}
X3 \in \{1, 2\},
\end{bmatrix},
\begin{bmatrix}
X4 \in \{1, 2\},
\end{bmatrix},
\begin{bmatrix}
X5 \in \{3, 4\}\right],
\begin{bmatrix}
2, & 1,
\end{bmatrix},
\begin{bmatrix}
Y3 \in \{2, 3\},
\end{bmatrix},
\begin{bmatrix}
Y4 \in \{0, 1\},
\end{bmatrix},
\begin{bmatrix}
Y5 \in \{0, 1\}\right])
\]
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex

Experiment with alternative value order
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex

Experiment with alternative value order

Helmut Simonis  Symmetry Breaking  233
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Complete Search Tree with Double Lex
Observation

- Enormous reduction in search space
- We are solving a different problem!
- Not just good for finding all solutions, also for first solution!
- Value choice not optimal for finding first solution
- There is a lot of very shallow backtracking, can we avoid that?
Effort for First Solution

Basic Model

With double Lex
Alternative Value Order

:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    bibd(6,10,5,3,2,Matrix), writeln(Matrix).

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),
    extract_array(row,Matrix,List),
    search(L,0,input_order,
        indomain_max,
        complete,[]).
Alternative Value Order

:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    bibd(6,10,5,3,2,Matrix), writeln(Matrix).

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),
    extract_array(row, Matrix, List),
    search(L, 0, input_order, indomain_max, \( \Rightarrow \) Start with 1 complete, []).
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First

Experiment with alternative value order
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First

Experiment with alternative value order
Assigning Value 1 First

Experiment with alternative value order
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>21</td>
<td>21</td>
<td>21</td>
<td>21</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>23</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>23</td>
</tr>
</tbody>
</table>
Assigning Value 1 First

Experiment with alternative value order
Assigning Value 1 First
Assigning Value 1 First
Experiment with alternative value order

Assigning Value 1 First

Back to Start

Skip Animation
Assigning Value 1 First
Assigning Value 1 First

Experiment with alternative value order
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First

Problem
Program
Symmetry Breaking

Experiment with alternative value order

Back to Start  Skip Animation

Helmut Simonis  Symmetry Breaking
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First

Experiment with alternative value order
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First

Experiment with alternative value order
Assigning Value 1 First
Assigning Value 1 First
Assigning Value 1 First
Observation

- First solution is found more quickly
- Size of tree for all solutions unchanged
- Value order does not really affect search space when exploring all choices!
Observation

- First solution is found more quickly
- Size of tree for all solutions unchanged
- **Value order does not really affect search space when exploring all choices!**
Effort for All Solutions

Assign 0, then 1

Assign 1, then 0
Conclusions

- Symmetry breaking can have huge impact on model
- Mainly works for pure problems
- Partial symmetry breaking with additional constraints
- Double lex for row/column symmetries
- Only one variant of many symmetry breaking techniques
4. Why assign by row?

5. Exercises
Row- or Column-wise Assignment?

- We did assign matrix by row, why?
- What happens if we assign variables by column?
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column

Why assign by row?
Exercises
Variable Selection by Column

Why assign by row?
Exercises
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Why assign by row?

Exercises

Variable Selection by Column
Variable Selection by Column
Why assign by row?
Exercises

Variable Selection by Column

[Diagram of a grid with variable selection by column highlighted]
Variable Selection by Column
Why assign by row?

Exercises

Variable Selection by Column
### Variable Selection by Column

The image shows a grid with numbers from 1 to 26 along the left side, presumably representing the columns, and rows labeled with a different set of numbers. The grid is partially filled with numbers, indicating a selection process that could be related to the concept of variable selection by column. The title suggests a discussion on why assigning by row might be preferred over assigning by column. This might be relevant in the context of data analysis or constraint satisfaction problems where strategic placement of values can affect efficiency and outcome.
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column

Why assign by row?
Exercises
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column

Why assign by row?

Exercises
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column

Why assign by row?

Exercises
Why assign by row?

Exercises

Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column

Why assign by row?

Exercises
Variable Selection by Column
Why assign by row?

Exercises

Variable Selection by Column
Variable Selection by Column
Variable Selection by Column

Why assign by row?
Exercises
Why assign by row?

Exercises

Variable Selection by Column

[Diagram of a variable selection process with a tree-like structure and a grid representation showing colored squares.]

Back to Start  Skip Animation

Helmut Simonis  Symmetry Breaking
Variable Selection by Column
Why assign by row?
Exercises

Variable Selection by Column

[Diagram of a tree structure and a grid with highlighted cells]
Variable Selection by Column
Variable Selection by Column
Why assign by row?
Exercises

Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column

Why assign by row?

Exercises
Variable Selection by Column

Why assign by row?

Exercises
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Why assign by row?

Exercises

Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column

Why assign by row?

Exercises
Why assign by row?

Exercises

Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Why assign by row?
Exercises

Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column

Why assign by row?
Exercises
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Why assign by row?

Exercises

Variable Selection by Column
Variable Selection by Column

Why assign by row?
Exercises
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Why assign by row?

Exercises

Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Why assign by row?

Exercises

Variable Selection by Column

Back to Start  Skip Animation

Helmut Simonis  Symmetry Breaking 453
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column
Variable Selection by Column

Why assign by row?
Exercises
Variable Selection by Column
Variable Selection by Column
Observation

- Good, but not as good as row order
- Value choice (0/1) or (1/0) unimportant even for first solution
- Changing the variable selection does affect size of search space, even for all solutions
Why assign by row?

Exercises

Effort for All Solutions

By Row

By Column
Possible Explanations

- There are fewer rows than columns
- Strict lex constraints on rows, but not on columns
  - More impact of first row
- Needs better understanding
Does this scale?

<table>
<thead>
<tr>
<th>v</th>
<th>b</th>
<th>r</th>
<th>k</th>
<th>λ</th>
<th>asym</th>
<th>lex²</th>
<th>STAB</th>
<th>lex² + SBNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>24</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>36</td>
<td>5,987</td>
<td>344</td>
<td>311</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>46</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>21</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>26</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>12,800</td>
<td>21</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>35</td>
<td>15</td>
<td>3</td>
<td>5</td>
<td>109</td>
<td>33,304</td>
<td>542</td>
<td>282</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>5</td>
<td>118</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>21</td>
<td>21</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>30</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>864</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>9</td>
<td>5</td>
<td>4</td>
<td>21</td>
<td>8,031</td>
<td>302</td>
<td>139</td>
</tr>
<tr>
<td>7</td>
<td>42</td>
<td>18</td>
<td>3</td>
<td>6</td>
<td>418</td>
<td>250,878</td>
<td>2,334</td>
<td>1,247</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
<td>7</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>49</td>
<td>21</td>
<td>3</td>
<td>7</td>
<td>1,508</td>
<td>1,460,332</td>
<td>8,821</td>
<td>4,353</td>
</tr>
<tr>
<td>8</td>
<td>28</td>
<td>14</td>
<td>4</td>
<td>6</td>
<td>2,310</td>
<td>2,058,523</td>
<td>17,890</td>
<td>11,424</td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td>9</td>
<td>9</td>
<td>4</td>
<td>6</td>
<td>6,520</td>
<td>71</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>30</td>
<td>9</td>
<td>3</td>
<td>2</td>
<td>960</td>
<td>724,662</td>
<td>24,563</td>
<td>15,169</td>
</tr>
<tr>
<td>31</td>
<td>31</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>864</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>56</td>
<td>24</td>
<td>3</td>
<td>8</td>
<td>5,413</td>
<td>6,941,124</td>
<td>32,038</td>
<td>14,428</td>
</tr>
<tr>
<td>9</td>
<td>36</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>22,521</td>
<td>14,843,772</td>
<td>315,531</td>
<td>85,605</td>
</tr>
<tr>
<td>7</td>
<td>63</td>
<td>27</td>
<td>3</td>
<td>9</td>
<td>?</td>
<td>28,079,394</td>
<td>105,955</td>
<td>43,259</td>
</tr>
<tr>
<td>15</td>
<td>35</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>80</td>
<td>32,127,296</td>
<td>6,782</td>
<td>35,183</td>
</tr>
<tr>
<td>21</td>
<td>28</td>
<td>8</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>26</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>2461</td>
<td>3,664,243</td>
<td>83,337</td>
<td>31,323</td>
</tr>
<tr>
<td>11</td>
<td>22</td>
<td>10</td>
<td>5</td>
<td>4</td>
<td>4393</td>
<td>6,143,408</td>
<td>106,522</td>
<td>32,908</td>
</tr>
<tr>
<td>12</td>
<td>22</td>
<td>11</td>
<td>6</td>
<td>5</td>
<td>?</td>
<td>?</td>
<td>228,146</td>
<td>76,572</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td>?</td>
<td>?</td>
<td>17,016</td>
<td>1,355</td>
</tr>
<tr>
<td>16</td>
<td>24</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>?</td>
<td>?</td>
<td>769,482</td>
<td>76,860</td>
</tr>
</tbody>
</table>
Scalability

- $\text{lex}^2$ good, but not good enough
- Still leaves too many symmetries to explore
- Better techniques in the literature
  - STAB, group theory based, Puget 2003.
  - SBNO, local search based domination check, Prestwich, 2008.
Do we need binary variables?

- The 0/1 model does very little propagation
- Consider a model with finite domain variables
- Each of $b$ blocks consists of $k$ variables ranging over $v$ values
- The values in a block must be alldifferent (ordered)
- Each value can occur $r$ times
- Scalar product more difficult
- Even better expressed with finite set variables
Why assign by row?

Exercises

More Information

Symmetry in constraint programming.

Jean-Francois Puget.
Symmetry breaking using stabilizers.
Outline

4. Why assign by row?

5. Exercises
Exercises

1