Symmetry and Optimization

François Margot
Tepper School of Business
Carnegie Mellon University

September 26, 2014

ISMP 2015: July 12-17, 2015

ISMP2015
PITTSBURGH
22nd INTERNATIONAL SYMPOSIUM
on MATHEMATICAL PROGRAMMING
JULY 12–17

Organized by the Mathematical Optimization Society

Carnegie Mellon University and University of Pittsburgh

`www.ismp2015.org`

Abstract submission is open

Registration: Opens December 2014

Symmetry and Optimization

$$
\begin{array}{ll}
\min & f(x) \\
\text{s.t.} & g(x) \leq 0 \\
& x_i \in \mathbb{Z} \quad \text{for} \quad i \in I \\
& x \in \mathbb{R}^n
\end{array}
$$

Symmetry and Optimization

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & x_i \in \mathbb{Z} \ \text{ for } \ i \in I \\ & x \in \mathbb{R}^n \end{array}$$

π : permutation of $\{1, \ldots, n\}$

$\pi(x) = \pi(x_1, \ldots, x_n) = (x_{\pi(1)}, \ldots, x_{\pi(n)})$

Symmetry and Optimization

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \\ & x_i \in \mathbb{Z} \ \text{ for } \ i \in I \\ & x \in \mathbb{R}^n \end{aligned}$$

π : permutation of $\{1, \ldots, n\}$
$\pi(x) = \pi(x_1, \ldots, x_n) = (x_{\pi(1)}, \ldots, x_{\pi(n)})$

π is a symmetry of the problem if

- x feasible $\Leftrightarrow \pi(x)$ feasible
- $f(x) = f(\pi(x))$

Special Case: Integer Linear Programming

Integer Linear Program (ILP):

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b \qquad\qquad\qquad A : m \times n \\ & x \in \{0, \ldots, k\}^n \end{aligned}$$

Special Case: Integer Linear Programming

Integer Linear Program (ILP):

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b \qquad\qquad A : m \times n \\ & x \in \{0, \dots, k\}^n \end{aligned}$$

π is a symmetry of the ILP if

- x feasible $\Leftrightarrow$ $\pi(x)$ feasible
- $c^T x = c^T \pi(x)$

Symmetry Group of the ILP

Example:

$$
\begin{array}{lrrrrcl}
\min & -x_1 & -x_2 & -x_3 & -x_4 & & \\
\text{s.t.} & x_1 & +x_2 & & +2x_4 & \leq & 2 \\
 & & x_2 & +x_3 & +2x_4 & \leq & 2 \\
 & x_1 & & +x_3 & +2x_4 & \leq & 2 \\
\end{array}
$$

$$x \in \{0,1\}^4$$

Symmetry Group of the ILP

Example:

$$\begin{array}{lrrrrcl} \min & -x_1 & -x_2 & -x_3 & -x_4 & & \\ \text{s.t.} & x_1 & +x_2 & & +2x_4 & \leq & 2 \\ & & x_2 & +x_3 & +2x_4 & \leq & 2 \\ & x_1 & & +x_3 & +2x_4 & \leq & 2 \\ & & & x \in \{0,1\}^4 & & & \end{array}$$

Feasible solutions:

$$(x_1, x_2, x_3, x_4) \in \{(0,0,0,0), (1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1),$$
$$(1,1,0,0),\ (1,0,1,0),\ (0,1,1,0),\ (1,1,1,0)\}$$

Symmetry Group of the ILP

Example:

$$\begin{array}{lrrrrcl}
\min & -x_1 & -x_2 & -x_3 & -x_4 & & \\
\text{s.t.} & x_1 & +x_2 & & +2x_4 & \leq & 2 \\
 & & x_2 & +x_3 & +2x_4 & \leq & 2 \\
 & x_1 & & +x_3 & +2x_4 & \leq & 2 \\
\end{array}$$

$$x \in \{0,1\}^4$$

Feasible solutions:

$$(x_1, x_2, x_3, x_4) \in \{(0,0,0,0), (1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1), (1,1,0,0), (1,0,1,0), (0,1,1,0), (1,1,1,0)\}$$

G : set of all symmetries of the ILP

$G = \{[1,2,3,4], [1,3,2,4], [2,1,3,4], [3,2,1,4], [2,3,1,4], [3,1,2,4]\}$

Symmetry Group

G with composition of permutation is a group:

- composition of permutations in G is a permutation in G
- G has a neutral element (identity permutation I)
- $g \in G \Rightarrow$ there exists $h \in G$ such that $g \cdot h = h \cdot g = I$

Symmetry Group

G with composition of permutation is a group:

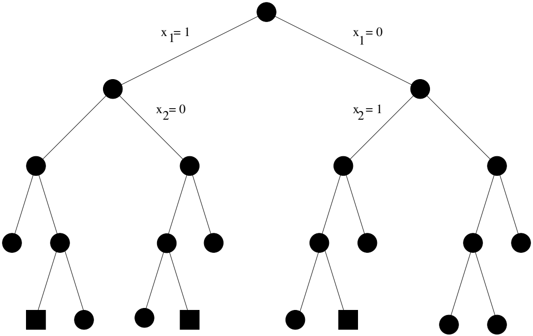
- composition of permutations in G is a permutation in G
- G has a neutral element (identity permutation I)
- $g \in G \Rightarrow$ there exists $h \in G$ such that $g \cdot h = h \cdot g = I$

Examples of symmetric ILPs:

- Coloring problems
- Network Design (with symmetric network)
- Flexible Manufacturing Operations (with identical machines)
- Design of statistical experiments
- Benchmark problems for QAP, Steiner Tree Problems

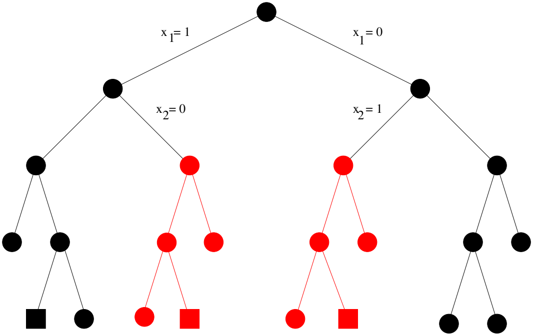
Branch-and-Bound for ILP

Branch-and-Bound tree for a symmetric problem: x_1, x_2 symmetric



Branch-and-Bound for ILP

Branch-and-Bound tree for a symmetric problem: x_1, x_2 symmetric



Problem:

$|G|$ large $\Rightarrow$ Most solution techniques become inefficient

Problem Characteristics

Problem	n	$\hat{z}$	LP	Group order	Comm. Solver
$OA_7(7,2,4,7)$	128	–	112	645,120	45m
$CA_7(7,2,4,7)$	128	113	112	645,120	2.5h
$PA_7(7,2,4,7)$	128	-108	-112	645,120	$>$ 4h
$OA_2(6,3,3,2)$	729	–	54	33,592,320	$>$ 4h
$cov1054$	252	51	50	3,628,800	$>$ 4h
$cov1174$	330	17	15.71	39,916,800	$>$ 4h
$cod93$	512	-40	-51.20	185,794,560	$>$ 4h
$cod105$	1024	-12	-18.29	371,5891,200	$>$ 4h
$STS81$	81	61	27	1,965,150,720	$>$ 4h

- "small" number of variables

Problem Characteristics

Problem	n	$\hat{z}$	LP	Group order	Comm. Solver
$OA_7(7,2,4,7)$	128	–	112	645,120	45m
$CA_7(7,2,4,7)$	128	113	112	645,120	2.5h
$PA_7(7,2,4,7)$	128	-108	-112	645,120	$>$ 4h
$OA_2(6,3,3,2)$	729	–	54	33,592,320	$>$ 4h
$cov1054$	252	51	50	3,628,800	$>$ 4h
$cov1174$	330	17	15.71	39,916,800	$>$ 4h
$cod93$	512	-40	-51.20	185,794,560	$>$ 4h
$cod105$	1024	-12	-18.29	371,5891,200	$>$ 4h
$STS81$	81	61	27	1,965,150,720	$>$ 4h

- "small" number of variables
- "small" integrality gap

Problem Characteristics

Problem	n	$\hat{z}$	LP	Group order	Comm. Solver
$OA_7(7,2,4,7)$	128	–	112	645,120	45m
$CA_7(7,2,4,7)$	128	113	112	645,120	2.5h
$PA_7(7,2,4,7)$	128	-108	-112	645,120	$>$ 4h
$OA_2(6,3,3,2)$	729	–	54	33,592,320	$>$ 4h
$cov1054$	252	51	50	3,628,800	$>$ 4h
$cov1174$	330	17	15.71	39,916,800	$>$ 4h
$cod93$	512	-40	-51.20	185,794,560	$>$ 4h
$cod105$	1024	-12	-18.29	371,5891,200	$>$ 4h
$STS81$	81	61	27	1,965,150,720	$>$ 4h

- "small" number of variables
- "small" integrality gap
- large group

Detecting Symmetric ILPs

Automatic detection is hard:

- Symmetry is a property of the feasible set (empty $\Rightarrow S^n$)
- May consider symmetry of the LP formulation
  (drawback: easy to destroy)

Detecting Symmetric ILPs

Automatic detection is hard:

- Symmetry is a property of the feasible set (empty $\Rightarrow S^n$)
- May consider symmetry of the LP formulation
  (drawback: easy to destroy)

Generating G:

- Known from model
- Compute from formulation : Graph automorphism problem
- Usually, work with a subgroup

Constructing G*: Graph Automorphism*

Example:

$$\begin{array}{ll} \min & 1^T x \\ \text{s.t.} & Ax \geq 1 \\ & x \in \{0,1\}^n \end{array} \qquad A: 0,1 \text{ matrix}$$

Constructing G: Graph Automorphism

Example:

$$\begin{array}{ll} \min & 1^T x \\ \text{s.t.} & Ax \geq 1 \\ & x \in \{0,1\}^n \end{array} \qquad A : 0,1 \text{ matrix}$$

$$\left(\begin{array}{c|c} 0 & A^T \\ \hline A & 0 \end{array}\right) : \text{ Adjacency matrix of a bipartite graph}$$

Constructing G: Graph Automorphism

Example:

$$\begin{array}{ll} \min & 1^T x \\ \text{s.t.} & Ax \geq 1 \\ & x \in \{0,1\}^n \end{array} \qquad\qquad A : 0, 1 \text{ matrix}$$

$$\left(\begin{array}{c|c} 0 & A^T \\ \hline A & 0 \end{array}\right) : \text{ Adjacency matrix of a bipartite graph}$$

- Can handle arbitrary matrix using color classes for identical entries

Constructing G: Graph Automorphism

Example:

$$\begin{array}{ll} \min & 1^T x \\ \text{s.t.} & Ax \geq 1 \\ & x \in \{0,1\}^n \end{array} \qquad\qquad A : 0, 1 \text{ matrix}$$

$$\left(\begin{array}{c|c} 0 & A^T \\ \hline A & 0 \end{array}\right)$$: Adjacency matrix of a bipartite graph

- Can handle arbitrary matrix using color classes for identical entries
- Efficient (in practice) software for graph automorphism: Nauty

[McKay]

Constructing G (cont.)

Extension to Nonlinear Programs [Liberti, 2010]

Library	# Instances	$\#\|G\| > 1$	% of library
MIPLIB3	62	22	35.4%
MIPLIB2003 $\setminus$ MIPLIB3	20	7	35.0%
GLOBALLIB	390	58	14.8%
MINLPLIB	197	32	16.2%
Total	669	112	16.7%

General Setting and Problems

Settings:

- Arbitrary symmetry group
- General integer variables

General Setting and Problems

Settings:

- Arbitrary symmetry group
- General integer variables

Problems:

- Finding optimal solution
- Optimality proof of known solution
- Finding all nonisomorphic optimal solutions

Approaches

"Exact" Algorithms:

"Approximate" Algorithms:

Approaches

"Exact" Algorithms:

- I: Perturbation
- II: Reformulations
- III: Symmetry breaking inequalities
- IV: Symmetry breaking during search
- V: Pruning the enumeration tree

"Approximate" Algorithms:

Approaches

"Exact" Algorithms:

- I: Perturbation
- II: Reformulations
- III: Symmetry breaking inequalities
- IV: Symmetry breaking during search
- V: Pruning the enumeration tree

"Approximate" Algorithms:

- VI: Orbital branching
- VII: Dominance relations

Approaches

"Exact" Algorithms:

- I: Perturbation
- II: Reformulations
- III: Symmetry breaking inequalities
- IV: Symmetry breaking during search
- V: Pruning the enumeration tree

"Approximate" Algorithms:

- VI: Orbital branching
- VII: Dominance relations

Can be used to enumerate all nonisomorphic solution

Approaches

"Exact" Algorithms:

- I: Perturbation
- II: Reformulations
- III: Symmetry breaking inequalities
- IV: Symmetry breaking during search
- V: Pruning the enumeration tree

"Approximate" Algorithms:

- VI: Orbital branching
- VII: Dominance relations

Use local symmetry information

Approach I: Perturbation

Modify the objective function:

- Replace by lexicographic minimization ($c_i = 2^i$, $i = 1, \ldots n$)
  - Works only for some problems
  - Numerical issues

Approach I: Perturbation

Modify the objective function:

- Replace by lexicographic minimization ($c_i = 2^i$, $i = 1, \ldots n$)
  - Works only for some problems
  - Numerical issues

- Add small perturbation to destroy symmetry
  - Counterproductive when trying to prove infeasibility
  - Once optimal solution found, all problems are of this type

Approach I: Perturbation

Modify the objective function:

- Replace by lexicographic minimization ($c_i = 2^i$, $i = 1, \ldots n$)
  - Works only for some problems
  - Numerical issues

- Add small perturbation to destroy symmetry
  - Counterproductive when trying to prove infeasibility
  - Once optimal solution found, all problems are of this type

- Perturbation using hierarchical functions related to symmetry-breaking constraints (only for G product of symmetric groups)
  - Good results for specific aplications.

[Ghoniem, Sherali 2011]

Approach II: Reformulations for $G = \mathcal{S}^n$

- Column generation
- Dantzig-Wolfe Decomposition

Approach II: Reformulations for $G = \mathcal{S}^n$

- Column generation
- Dantzig-Wolfe Decomposition

Example: Minimize # of identical machines to perform m tasks

- Column: subset of tasks that can be assigned to a single machine
- Dantzig-Wolfe Decomposition: Minimize # subsets to cover all tasks

[Barnhart, Johnson, Nemhauser, 1998]

Approach II: Reformulations for $G = \mathcal{S}^n$

- Column generation
- Dantzig-Wolfe Decomposition

Example: Minimize # of identical machines to perform m tasks

- Column: subset of tasks that can be assigned to a single machine
- Dantzig-Wolfe Decomposition: Minimize # subsets to cover all tasks

[Barnhart, Johnson, Nemhauser, 1998]

Edge Coloring: [Nemhauser, Park, 1991]
Vertex Coloring: [Mehrotra, Trick, 1995]

Approach II: Reformulations by Lift-and-Relax

[Östergård, Weakley, 2000]

[Östergård, Blass, 2001]

[Östergård, Wassermann, 2002]

[Östergård, M., 2003]

[Linderoth, M., Thain, 2007]

- Add integer variables y to the ILP: $y = Zx$
- Project (relax) some (or all) of the x variables $\rightarrow ILP(x', y)$
- Enumerate all nonisomorphic solutions $(\bar{x}', \bar{y})$ to $ILP(x', y)$
- Solve original ILP for each $(\bar{x}', \bar{y})$, adding constraints $\bar{x}' = x'$ and $\bar{y} = Zx$

Approach II: Reformulations by orbit shrinking

[Fischetti, Liberti, 2013]

- G': subgroup of G
- Partition variables into orbits in G': $\{O_1, \ldots, O_t\}$
- For $i = 1, \ldots, t$ add an integer variable $y_t = \sum_{j \in O_t} x_j$
- remove integrality restriction on x_j for all j
- For $x_j \in O_t$, replace x_j by $\frac{y_t}{|O_t|}$

Approach II: Reformulations by orbit shrinking

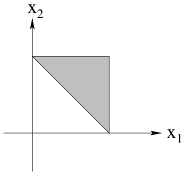
[Fischetti, Liberti, 2013]

- G': subgroup of G
- Partition variables into orbits in G': $\{O_1, \ldots, O_t\}$
- For $i = 1, \ldots, t$ add an integer variable $y_t = \sum_{j \in O_t} x_j$
- remove integrality restriction on x_j for all j
- For $x_j \in O_t$, replace x_j by $\frac{y_t}{|O_t|}$
- Solves a "smaller" and "easier" ILP
- Significantly improves the LP relaxation lower bound

Approach III: Adding inequalities

Idea:

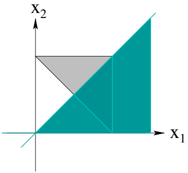
- Add inequalities remove some of the symmetry, keeping at least one optimal solution



Approach III: Adding inequalities

Idea:

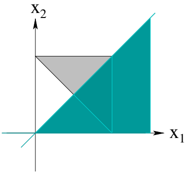
- Add inequalities remove some of the symmetry, keeping at least one optimal solution
- Usually: Intersect original formulation with a cone pointed at the origin



Approach III: Adding inequalities

Idea:

- Add inequalities remove some of the symmetry, keeping at least one optimal solution
- Usually: Intersect original formulation with a cone pointed at the origin



Drawbacks:

- Isomorphic solutions may remain feasible
- May create highly fractional LP relaxations

III Adding Inequalities (cont.)

Typical constraints: Let $O = \{x_1, x_2, \ldots, x_t\}$ be one orbit in G

- If G restricted to O is the symmetric group $\mathcal{S}^t$:

$$x_1 \geq x_2 \geq \ldots \geq x_t$$

- Otherwise

$$x_1 \geq x_2, \quad x_1 \geq x_3, \quad \ldots x_1 \geq x_t$$

Applications:

- Selection of orbit [Liberti 2010]
- Using group operation to use several orbits [Liberti, Ostrowski 2013]

Finding Symmetry Breaking Inequalities

Fundamental Region: closed set F in $\mathbb{R}^n$ such that:

- $g(\mathrm{int(F)}) \cap \mathrm{int(F)} = \emptyset, \quad \forall \mathrm{g} \in \mathrm{G}, \mathrm{g} \neq \mathrm{I}$
- $\bigcup_{g \in G} g(F) = \mathbb{R}^n$

Finding Symmetry Breaking Inequalities

Fundamental Region: closed set F in $\mathbb{R}^n$ such that:

- $g(\mathrm{int(F)}) \cap \mathrm{int(F)} = \emptyset, \quad \forall \mathrm{g} \in \mathrm{G}, \mathrm{g} \neq \mathrm{I}$
- $\bigcup_{g \in G} g(F) = \mathbb{R}^n$

Theorem:

- G symmetry group for polytope P
- F fundamental region

Then:

$$\min\{c^T x \mid x \in P\} = \min\{c^T x \mid x \in P \cap F\}$$

Finding Symmetry Breaking Inequalities (cont.)

Proposition: [Grove, Benson, 1985]

- G group of permutations of $\{1, \ldots, n\}$.
- z such that $g(z) \neq z$ for all $g \in G, g \neq I$.

Then

$$F = \{x \in \mathbb{R}^n \mid (g(z) - z) \cdot x \leq 0, \ \forall g \in G, \ g \neq I\}$$

is a fundamental region for G.

Finding Symmetry Breaking Inequalities (cont.)

Proposition: [Grove, Benson, 1985]

- G group of permutations of $\{1, \ldots, n\}$.
- z such that $g(z) \neq z$ for all $g \in G, g \neq I$.

Then

$$F = \{x \in \mathbb{R}^n \mid (g(z) - z) \cdot x \leq 0, \ \forall g \in G, \ g \neq I\}$$

is a fundamental region for G.

Example: $G = \mathcal{S}^4$

$$z = (0, 1, 2, 3)$$

$$g(z) = (1, 0, 2, 3)$$

$$\Rightarrow \qquad (x_2 + 2x_3 + 3x_4) - (x_1 + 2x_3 + 3x_4) \leq 0 \qquad \Rightarrow$$

$$x_1 \geq x_2$$

III Adding Inequalities (cont.): Orbitopes

For packing or partitioning problems of the form:

$$
\begin{array}{lll}
Ax & \leq & b \\
\sum_{j=1}^{n} x_{ij} & \{=; \leq\} & 1 \qquad \forall i = 1, \ldots, m \\
x_{ij} & \geq & 0
\end{array}
$$

III Adding Inequalities (cont.): Orbitopes

For packing or partitioning problems of the form:

$$
\begin{array}{lcll}
Ax & \leq & b & \\
\sum_{j=1}^{n} x_{ij} & \{=; \leq\} & 1 & \forall i = 1, \ldots, m \\
x_{ij} & \geq & 0 &
\end{array}
$$

Collect all variables in a 2-dimensional matrix:

$$
X = \begin{array}{|c|c|c|c|c|}
\hline
x_{1,1} & x_{1,2} & x_{1,3} & \ldots & x_{1,n} \\
\hline
x_{2,1} & x_{2,2} & x_{2,3} & \ldots & x_{2,n} \\
\hline
\ldots & \ldots & \ldots & \ldots & \ldots \\
\hline
x_{m,1} & x_{m,2} & x_{m,3} & \ldots & x_{m,n} \\
\hline
\end{array}
$$

Orbitopes

If any permutation of the columns of X is a symmetry of the problem:

Orbitopes

If any permutation of the columns of X is a symmetry of the problem:

- a family of symmetry breaking inequalities is known (*shifted column inequalities*)
- polynomial time separation algorithm
- Describes the convex hull of non isomorphic solutions of

$$\begin{aligned} \sum_{j=1}^{n} x_{ij} \quad &\{=; \leq\} \quad 1 \qquad \forall i = 1, \ldots, m \\ x_{ij} \quad &\geq \quad 0 \end{aligned}$$

[Kaibel, Pfetsch, 2006]

[Kaibel, Peinhardt, Pfetsch, 2007]

Isomorphism-free backtracking enumeration

[Butler, Ivanov, Kreher, Lam, Leon, McKay, Read, Stinson]

Example: Solving an ILP with 0, 1 variables:
a : node of the enumeration tree

$$F_1^a = \{i \mid x_i \text{ fixed to 1 at } a\}$$

$$F_0^a = \{i \mid x_i \text{ fixed to 0 at } a\}$$

Isomorphism-free backtracking enumeration

[Butler, Ivanov, Kreher, Lam, Leon, McKay, Read, Stinson]

Example: Solving an ILP with 0, 1 variables:
a : node of the enumeration tree

$$F_1^a = \{i \mid x_i \text{ fixed to 1 at } a\}$$

$$F_0^a = \{i \mid x_i \text{ fixed to 0 at } a\}$$

Problems at a and b are isomorphic if

$$\exists\ g \in G \text{ with}$$

$$g(F_1^a) = F_1^b \qquad \text{and} \qquad g(F_0^a) = F_0^b$$

$\Rightarrow$ May prune one of a or b

[Bazaraa, Kirca, 1983]

Approach IV: Symmetry Breaking During Search (SBDS)

Constraint Programming Approach:

- Add constraints for each created node of the tree to forbid isomorphic ones
- May require huge number of constraints
- Need to keep track of some of the visited nodes

Approach IV: Symmetry Breaking During Search (SBDS)

Constraint Programming Approach:

- Add constraints for each created node of the tree to forbid isomorphic ones
- May require huge number of constraints
- Need to keep track of some of the visited nodes

- Symmetry Breaking During Search:

[Gent, Smith, 2002]

[Gent, Kelsey, et al. 2005]

Approach IV: Symmetry Breaking During Search (SBDS)

Constraint Programming Approach:

- Add constraints for each created node of the tree to forbid isomorphic ones
- May require huge number of constraints
- Need to keep track of some of the visited nodes

- Symmetry Breaking During Search:

[Gent, Smith, 2002]

[Gent, Kelsey, et al. 2005]

- GAP_SBDS: Group representation of the symmetries:

[Gent, Harvey, Kelsey 2002]

Approach IV: Symmetry Breaking During Search (SBDS)

Constraint Programming Approach:

- Add constraints for each created node of the tree to forbid isomorphic ones
- May require huge number of constraints
- Need to keep track of some of the visited nodes

- Symmetry Breaking During Search:

  [Gent, Smith, 2002]

  [Gent, Kelsey, et al. 2005]

- GAP_SBDS: Group representation of the symmetries:

  [Gent, Harvey, Kelsey 2002]

- SBDS-CP-LP hybrid [Petrie, Smith, 2004]

Approach V: Pruning

Assumptions:

- Branch by partitioning the domain of a variable into $k \geq 2$ subdomains
- Complete ordering of the sons (topological in drawing of tree)

Approach V: Pruning

Assumptions:

- Branch by partitioning the domain of a variable into $k \geq 2$ subdomains
- Complete ordering of the sons (topological in drawing of tree)

Sought: Minimum interference with usual operations

- Freedom to restrict variable domains
- Freedom to choose the branching variable
- Freedom to choose the partitioning
- Pruning uses only information available at a single node of tree

Approach V: Pruning

Assumptions:

- Branch by partitioning the domain of a variable into $k \geq 2$ subdomains
- Complete ordering of the sons (topological in drawing of tree)

Sought: Minimum interference with usual operations

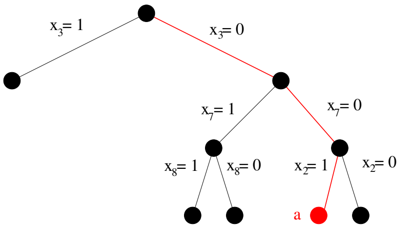
- Freedom to restrict variable domains
- Freedom to choose the branching variable
- Freedom to choose the partitioning
- Pruning uses only information available at a single node of tree

Achievable if:

- Algorithm for restrictions is not be based on symmetry considerations

V.I: Left-of-path Mapping

- a: node of the tree
- $F_0^a = \{i \mid x_i \text{ fixed to 0 at } a\}$ $F_1^a = \{i \mid x_i \text{ fixed to 1 at } a\}$
- Compare fixing at a with left sons issued from ancestors of a

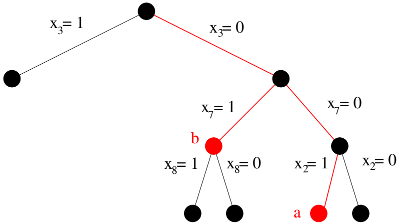


$F_1^a = \{2\}$ $F_0^a = \{3, 7\}$

V.I: Left-of-path Mapping

- a: node of the tree
- $F_0^a = \{i \mid x_i \text{ fixed to 0 at } a\}$ $\quad F_1^a = \{i \mid x_i \text{ fixed to 1 at } a\}$

- Compare fixing at a with left sons issued from ancestors of a
- Prune a if there exists $g \in G$ mapping a subset of F_1^a to F_1^b and a subset of F_0^a to F_0^b

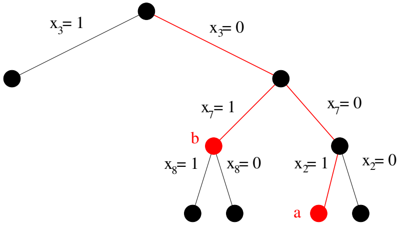


$F_1^b = \{7\}$ $\; F_0^b = \{3\}$

$F_1^a = \{2\}$ $\; F_0^a = \{3, 7\}$

V.I: Left-of-path Mapping

- a: node of the tree
- $F_0^a = \{i \mid x_i \text{ fixed to 0 at } a\}$ $\quad F_1^a = \{i \mid x_i \text{ fixed to 1 at } a\}$
- Compare fixing at a with left sons issued from ancestors of a
- Prune a if there exists $g \in G$ mapping a subset of F_1^a to F_1^b and a subset of F_0^a to F_0^b



$F_1^b = \{7\}$ $F_0^b = \{3\}$

$F_1^a = \{2\}$ $F_0^a = \{3, 7\}$

$\exists g \in G$ with $g(3) = 3,\ g(2) = 7 \Rightarrow$ Prune node a

V.I: Left-of-path Mapping (cont.)

- Backtrack Searching with Symmetry (BSS)

[Brown, Finkelstein, Purdom 1988, 1995]

- Symmetry Breaking By Dominance Detection (SBDD)

[Fahle, Shamberger, Sellman 2001]

V.I: Left-of-path Mapping (cont.)

- Backtrack Searching with Symmetry (BSS)

  [Brown, Finkelstein, Purdom 1988, 1995]

  Paper Description:
  - General integer variables
  - Branch by creating one son for each possible value of branching variable

- Symmetry Breaking By Dominance Detection (SBDD)

  [Fahle, Shamberger, Sellman 2001]

  Paper Description:
  - General integer variables
  - Branch by arbitrary partition of domain of branching variable

V.I: Left-of-path Mapping (cont.)

- Backtrack Searching with Symmetry (BSS)

  [Brown, Finkelstein, Purdom 1988, 1995]

  Paper Description:
  - General integer variables
  - Branch by creating one son for each possible value of branching variable

  Implementation:
  - Generic code working for any group
  - Few numerical results

- Symmetry Breaking By Dominance Detection (SBDD)

  [Fahle, Shamberger, Sellman 2001]

  Paper Description:
  - General integer variables
  - Branch by arbitrary partition of domain of branching variable

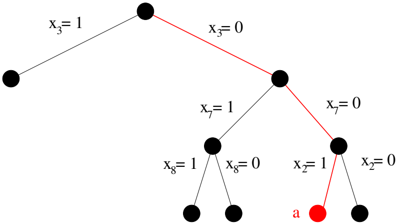
  Implementation:
  - Ad hoc code for several applications

V.II: Lexicomin Support Pruning

[Butler, Ivanov, Kreher, Lam, Leon, McKay, Read, Stinson]

- a: node of the tree
- $F_1^a = \{i \mid x_i \text{ fixed to 1 at } a\}$

- Compare fixing at a with left sons issued from ancestors of a



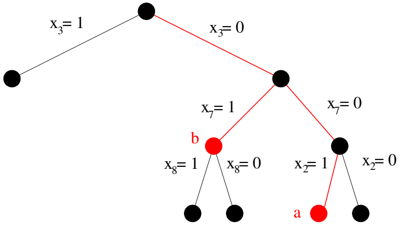
$F_1^a = \{2\}$

V.II: Lexicomin Support Pruning

[Butler, Ivanov, Kreher, Lam, Leon, McKay, Read, Stinson]

- a: node of the tree
- $F_1^a = \{i \mid x_i \text{ fixed to 1 at } a\}$

- Compare fixing at a with left sons issued from ancestors of a
- Prune a if there exists $g \in G$ mapping a subset of F_1^a to F_1^b



$$F_1^b = \{7\}$$

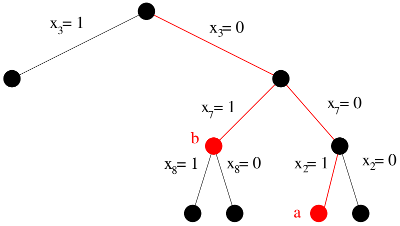
$$F_1^a = \{2\}$$

V.II: Lexicomin Support Pruning

[Butler, Ivanov, Kreher, Lam, Leon, McKay, Read, Stinson]

- a: node of the tree
- $F_1^a = \{i \mid x_i \text{ fixed to 1 at } a\}$

- Compare fixing at a with left sons issued from ancestors of a
- Prune a if there exists $g \in G$ mapping a subset of F_1^a to F_1^b



$F_1^b = \{7\}$

$F_1^a = \{2\}$

$\exists g \in G$ with $g(2) = 7 \Rightarrow$ Prune node a

V.II: Lexicomin Support Pruning (cont.)

- Isomorphism Pruning (IP)

[M 2002, 2003, 2003b, 2007]

V.II: Lexicomin Support Pruning (cont.)

- Isomorphism Pruning (IP)

[M 2002, 2003, 2003b, 2007]

Paper Description:

- General integer variables
- Branch by creating one son for each possible value of branching variable
- Rigid branching scheme

V.II: Lexicomin Support Pruning (cont.)

- Isomorphism Pruning (IP)

[M 2002, 2003, 2003b, 2007]

Paper Description:

- General integer variables
- Branch by creating one son for each possible value of branching variable
- Rigid branching scheme

V.II: Lexicomin Support Pruning (cont.)

- Isomorphism Pruning (IP)

[M 2002, 2003, 2003b, 2007]

Paper Description:

  - General integer variables
  - Branch by creating one son for each possible value of branching variable
  - Rigid branching scheme
    Can be relaxed [Ostrowski 2007]

V.II: Lexicomin Support Pruning (cont.)

- Isomorphism Pruning (IP)

[M 2002, 2003, 2003b, 2007]

Paper Description:

- General integer variables
- Branch by creating one son for each possible value of branching variable
- Rigid branching scheme
  Can be relaxed [Ostrowski 2007]

Implementation:

- Generic code working for any group
- Applications:
  - covering designs [M 2003]
  - orthogonal arrays [Bulutoglu, M 2007]
  - edge coloring [M 2007]
  - codes [Linderoth, Thain, M 2007]

Left-of-path Mapping vs. Lexicomin Support

Bare bone comparison:

- Left-of-path mapping pruning $\subseteq$ Lexicomin Support pruning

Left-of-path Mapping vs. Lexicomin Support

Bare bone comparison:

- Left-of-path mapping pruning $\subseteq$ Lexicomin Support pruning
- Algorithms based on group representation are backtracking algorithms with depth $|F_1^a \cup F_0^a|$ and $|F_1^a|$ respectively

Left-of-path Mapping vs. Lexicomin Support

Bare bone comparison:

- Left-of-path mapping pruning $\subseteq$ Lexicomin Support pruning
- Algorithms based on group representation are backtracking algorithms with depth $|F_1^a \cup F_0^a|$ and $|F_1^a|$ respectively
- Lexicomin Support clear winner

Left-of-path Mapping vs. Lexicomin Support

Bare bone comparison:

- Left-of-path mapping pruning $\subseteq$ Lexicomin Support pruning
- Algorithms based on group representation are backtracking algorithms with depth $|F_1^a \cup F_0^a|$ and $|F_1^a|$ respectively
- Lexicomin Support clear winner

However:

Can set variables to 0 during backtracking. If there exists:

- $F \subseteq F_1^a$
- b a left-ancestor of a
- $g \in G$ with $g(F) = F_1^b$

then set to 0 all vars in $g^{-1}(F_0^b)$

Left-of-path Mapping vs. Lexicomin Support

Bare bone comparison:

- Left-of-path mapping pruning $\subseteq$ Lexicomin Support pruning
- Algorithms based on group representation are backtracking algorithms with depth $|F_1^a \cup F_0^a|$ and $|F_1^a|$ respectively
- Lexicomin Support clear winner

However:

Can set variables to 0 during backtracking. If there exists:

- $F \subseteq F_1^a$
- b a left-ancestor of a
- $g \in G$ with $g(F) = F_1^b$

then set to 0 all vars in $g^{-1}(F_0^b)$ (0-setting)

Left-of-path Mapping vs. Lexicomin Support

If full 0-setting is done in Left-of-path mapping then

- Left-of-path mapping pruning = Lexicomin Support pruning

Left-of-path Mapping vs. Lexicomin Support

If full 0-setting is done in Left-of-path mapping then

- Left-of-path mapping pruning = Lexicomin Support pruning
- More variables set to 0 by Left-of-path mapping than with Lexicomin Support

Left-of-path Mapping vs. Lexicomin Support

If full 0-setting is done in Left-of-path mapping then

- Left-of-path mapping pruning = Lexicomin Support pruning
- More variables set to 0 by Left-of-path mapping than with Lexicomin Support
- Much slower Left-of-path mapping checking than Lexicomin Support checking for deep trees

Left-of-path Mapping vs. Lexicomin Support

If full 0-setting is done in Left-of-path mapping then

- Left-of-path mapping pruning = Lexicomin Support pruning
- More variables set to 0 by Left-of-path mapping than with Lexicomin Support
- Much slower Left-of-path mapping checking than Lexicomin Support checking for deep trees

BSS implementation of [Brown, Finkelstein, Purdom 1988, 1995]

- Fast comput. of generators of stabilizer of $(F_1^a \cup j, F_0^a)$ in G
- Does not always do full 0-setting

Left-of-path Mapping vs. Lexicomin Support

If full 0-setting is done in Left-of-path mapping then

- Left-of-path mapping pruning = Lexicomin Support pruning
- More variables set to 0 by Left-of-path mapping than with Lexicomin Support
- Much slower Left-of-path mapping checking than Lexicomin Support checking for deep trees

BSS implementation of [Brown, Finkelstein, Purdom 1988, 1995]

- Fast comput. of generators of stabilizer of $(F_1^a \cup j, F_0^a)$ in G
- Does not always do full 0-setting

IP implementation of [M 2007]

- Fast computation of one orbit of stabilizer of $F_1^a \cup j$ in G
- Does not always do full 0-setting

Approach VI: Orbital Branching

Recompute symmetry group at each node; use orbit information for branching

Approach VI: Orbital Branching

Recompute symmetry group at each node; use orbit information for branching

Orbital Branching:

[Ostrowski, Linderoth, Rossi, Smriglio 2007]

Approach VI: Orbital Branching

Recompute symmetry group at each node; use orbit information for branching

Orbital Branching:

- Recompute symmetry group for free variables

[Ostrowski, Linderoth, Rossi, Smriglio 2007]

Approach VI: Orbital Branching

Recompute symmetry group at each node; use orbit information for branching

Orbital Branching:

- Recompute symmetry group for free variables
- Compute partition of free variables into orbits

[Ostrowski, Linderoth, Rossi, Smriglio 2007]

Approach VI: Orbital Branching

Recompute symmetry group at each node; use orbit information for branching

Orbital Branching:

- Recompute symmetry group for free variables
- Compute partition of free variables into orbits
- Select one orbit $\mathcal{O}$ and one $x_i \in \mathcal{O}$;

[Ostrowski, Linderoth, Rossi, Smriglio 2007]

Approach VI: Orbital Branching

Recompute symmetry group at each node; use orbit information for branching

Orbital Branching:

- Recompute symmetry group for free variables
- Compute partition of free variables into orbits
- Select one orbit $\mathcal{O}$ and one $x_i \in \mathcal{O}$;
- Branch:
  either all vars in $\mathcal{O}$ fixed to 0 or $x_i = 1$

[Ostrowski, Linderoth, Rossi, Smriglio 2007]

Approach VI: Constraint Orbital Branching

[Ostrowski, Linderoth, Rossi, Smriglio, 2008, 2009, 2011]

$$a^T x \geq b : \text{valid constraint of the LP, } \mathrm{a} \in \mathbb{Z}^{\mathrm{n}}, \mathrm{b} \in \mathbb{Z}$$

Approach VI: Constraint Orbital Branching

[Ostrowski, Linderoth, Rossi, Smriglio, 2008, 2009, 2011]

$$a^T x \geq b : \text{valid constraint of the LP, } a \in \mathbb{Z}^n, b \in \mathbb{Z}$$

Then

$$\pi(a)^T x \geq b : \text{ also valid constraint of the LP for all } \pi \in G$$

Approach VI: Constraint Orbital Branching

[Ostrowski, Linderoth, Rossi, Smriglio, 2008, 2009, 2011]

$$a^T x \geq b : \text{valid constraint of the LP, } a \in \mathbb{Z}^n, b \in \mathbb{Z}$$

Then

$$\pi(a)^T x \geq b : \text{ also valid constraint of the LP for all } \pi \in G$$

Disjunction:

$$a^T x \leq b \qquad or \qquad \pi(a)^T x \geq b + 1 \text{ for all } \pi \in G$$

Approach VI: Constraint Orbital Branching

[Ostrowski, Linderoth, Rossi, Smriglio, 2008, 2009, 2011]

$$a^T x \geq b : \text{valid constraint of the LP, } a \in \mathbb{Z}^n, b \in \mathbb{Z}$$

Then

$$\pi(a)^T x \geq b : \text{ also valid constraint of the LP for all } \pi \in G$$

Disjunction:

$$a^T x \leq b \qquad \textit{or} \qquad \pi(a)^T x \geq b + 1 \text{ for all } \pi \in G$$

- Particularly useful for instances built from smaller instances
- Can combine knowledge of all nonisomorphic solutions of smaller instances
- Gives first proof of optimality for `STS135` and `STS243`

Approach VII: Dominance Relations

- Use MIP to detect assignment of variables that are dominated
- Limited efficiency for highly symmetric problems

[Fischetti, Toth, 1988]

[Fischetti, Salvagnin, 2007]

ISOP-1.2

Code implementing:

- Group representation: Schreier-Sims table, set stabilizer computation
- Branch-and-Bound with isomorphism pruning
  - Based on code BCP of COIN-OR
  - Use CPLEX as LP solver
  - Input: problem description (sparse LP), group description (Schreier-Sims table), upper bound UB
  - output: Either one optimal solution with value $< UB$ or list of all nonisomorphic solutions with value $< UB$
  - more than 50 options for branching selection, bound propagation, etc.
  - More than 100 instances of highly symmetric problems (BIBD, COD, OA, STS)

ISOP-1.2 (cont.)

Available from:
`http://wpweb2.tepper.cmu.edu/fmargot/source_code.html`

Automatic generation of 11 codes using various options.

ISOP-1.2 (cont.)

Available from:
`http://wpweb2.tepper.cmu.edu/fmargot/source_code.html`

Automatic generation of 11 codes using various options.

Important options active in precompiled codes:

- `NO_SOL_X0_0`: All variables are in a single orbit of G
- `ALL_SOL`: Output all nonisomoprhic solutions

ISOP-1.2 (cont.)

Available from:
http://wpweb2.tepper.cmu.edu/fmargot/source_code.html

Automatic generation of 11 codes using various options.

Important options active in precompiled codes:

- NO_SOL_X0_0: All variables are in a single orbit of G
- ALL_SOL: Output all nonisomoprhic solutions

Main Options:

- FREE_BRANCHING: Selected branching variable freely
- FREE_STAB_GRP: Compute a Schreier-Sims representation of the stabilizer

ISOP-1.2 (cont.)

If `FREE_BRANCHING`

- If `FREE_BRANCH_GRP`
  - `FB_SCORE_KEEP_SYM`: `ks` keep largest group order
  - `FB_SCORE_KEEP_SYM_NONZ`: keep largest group order when not fixing to 0
  - `FB_SCORE_BREAK_SYM`: `bs` keep smallest group order
  - `FB_SCORE_BREAK_SYM_NONZ`: keep smallest group order when not fixing to 0
  - `FB_SCORE_MAX_PROD`: `mp` keep max product of current orbit and largest orbit in sons

ISOP-1.2 (cont.)

- Otherwise (not FREE_BRANCH_GRP)
  - orig: order vars by given indexing
  - orig+fix: order vars by given indexing, do strong fixing
  - FB_SCORE_LARGEST_ORB: lo order vars according to orbit sizes
  - FB_SCORE_LARGEST_LP_ORB: lplo order vars according to largest sum of LP values in orbit
  - FB_SCORE_STRONG: str5 use strong branching/fixing at depth multiple of a constant

CPU Time

code	avg.	std dev	min	max	TL
orig	26.55	63.54	0.00	360.10	0
orig+fix	122.91	635.05	0.00	5,757.60	0
str5	26.56	67.94	0.00	459.20	0
ks	328.06	1,360.54	0.00	10,004.60	0
bs	924.33	3,733.70	0.00	27,500.00	7
lo	275.87	1,391.94	0.00	12,859.10	2
lplo	409.36	2,522.26	0.00	23,938.40	6
mp	611.11	2,384.66	0.00	18,880.20	5

- 94 “easy” instances
- enumerate all nonisomorphic solutions
- TL: not finished in 10h

CPU Time

code	avg.	std dev	min	max	TL
orig	26.55	63.54	0.00	360.10	0
orig+fix	122.91	635.05	0.00	5,757.60	0
str5	26.56	67.94	0.00	459.20	0
ks	328.06	1,360.54	0.00	10,004.60	0
bs	924.33	3,733.70	0.00	27,500.00	7
lo	275.87	1,391.94	0.00	12,859.10	2
lplo	409.36	2,522.26	0.00	23,938.40	6
mp	611.11	2,384.66	0.00	18,880.20	5

- 94 “easy” instances
- enumerate all nonisomorphic solutions
- TL: not finished in 10h

Nodes

code	avg.	std dev	min	max	TL
orig	27,759.84	123,184.01	11.00	944,517.00	0
orig+fix	6,277.61	21,386.44	11.00	160,363.00	0
str5	28,838.34	126,534.00	11.00	972,642.00	0
ks	29,332.57	127,451.56	11.00	1,150,059.00	0
bs	34,816.05	122,522.98	11.00	727,348.00	7
lo	99,046.55	396,735.30	11.00	294,3748.00	2
lplo	52,473.76	189,442.26	13.00	1,101,518.00	6
mp	28,123.15	131,512.86	13.00	1,180,263.00	5

- 94 “easy” instances
- enumerate all nonisomorphic solutions
- TL: not finished in 10h

Nodes

code	avg.	std dev	min	max	TL
`orig`	27,759.84	123,184.01	11.00	944,517.00	0
`orig+fix`	6,277.61	21,386.44	11.00	160,363.00	0
`str5`	28,838.34	126,534.00	11.00	972,642.00	0
`ks`	29,332.57	127,451.56	11.00	1,150,059.00	0
`bs`	34,816.05	122,522.98	11.00	727,348.00	7
`lo`	99,046.55	396,735.30	11.00	294,3748.00	2
`lplo`	52,473.76	189,442.26	13.00	1,101,518.00	6
`mp`	28,123.15	131,512.86	13.00	1,180,263.00	5

- 94 “easy” instances
- enumerate all nonisomorphic solutions
- TL: not finished in 10h

Group Operations

[Butler, Cannon, Lam, Kreher, Stinson, Leon]

$G_0 = G$
$G_1 = \{g \in G_0 \mid g(1) = 1\} \quad U_1 = \mathrm{orb}(1, G_0)$
$G_2 = \{g \in G_1 \mid g(2) = 2\} \quad U_2 = \mathrm{orb}(2, G_1)$
$\ldots$
$G_n = \{g \in G_{n-1} \mid g(n) = n\} \; U_n = \mathrm{orb}(n, G_{n-1})$

Group Operations

[Butler, Cannon, Lam, Kreher, Stinson, Leon]

$G_0 = G$
$G_1 = \{g \in G_0 \mid g(1) = 1\} \quad U_1 = \mathrm{orb}(1, G_0)$
$G_2 = \{g \in G_1 \mid g(2) = 2\} \quad U_2 = \mathrm{orb}(2, G_1)$
...
$G_n = \{g \in G_{n-1} \mid g(n) = n\} \; U_n = \mathrm{orb}(n, G_{n-1})$

Schreier-Sims Table:
$T : n \times n$ table of permutations

$$T_{ij} \neq \emptyset \Leftrightarrow \exists\, g \in G_{i-1} \text{ with } g(i) = j$$

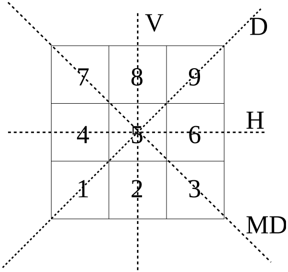
Schreier-Sims Table Example

$G_0 = \{I, R_{90}, R_{180}, R_{270}, H, V, D, MD\}$

$G_1 = \{I, D\} \qquad U_1 = \{1, 3, 7, 9\}$

$G_2 = \{I\} \qquad U_2 = \{2, 4\}$

$G_i = \{I\} \qquad U_i = \{i\}$ for all $i \geq 3$



	1	2	3	4	5	6	7	8	9
1	I		V				R_{270}		R_{180}
2		I		D					
3			I						
4				I					
5					I				
6						I			
7							I		
8								I	
9									I

Schreier-Sims Table Properties

- First row is $\text{orb}(1, G)$

Schreier-Sims Table Properties

- First row is $\text{orb}(1, G)$
- $g \in G \Leftrightarrow g = g_1 \cdot g_2 \cdot \ldots \cdot g_n$ with $g_i \in$ row i (unique, strong generators)

Schreier-Sims Table Properties

- First row is $\text{orb}(1, G)$
- $g \in G \Leftrightarrow g = g_1 \cdot g_2 \cdot \ldots \cdot g_n$ with $g_i \in$ row i (unique, strong generators)
- $|G| = |U_1| \cdot |U_2| \cdot \ldots \cdot |U_n|$

Schreier-Sims Table Properties

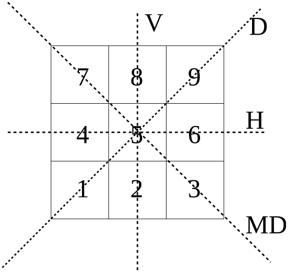
- First row is $\mathrm{orb}(1, G)$
- $g \in G \Leftrightarrow g = g_1 \cdot g_2 \cdot \ldots \cdot g_n$ with $g_i \in$ row i
  (unique, strong generators)
- $|G| = |U_1| \cdot |U_2| \cdot \ldots \cdot |U_n|$
- Construction Algorithm from generators ($O(n^4)$ per generator)

Schreier-Sims Table Properties

- First row is orb$(1, G)$
- $g \in G \Leftrightarrow g = g_1 \cdot g_2 \cdot \ldots \cdot g_n$ with $g_i \in$ row i
  (unique, strong generators)
- $|G| = |U_1| \cdot |U_2| \cdot \ldots \cdot |U_n|$
- Construction Algorithm from generators ($O(n^4)$ per generator)
- β : permutation, $G_0 = G$, $G_i = \{g \in G_{i-1} \mid g(\beta[i]) = \beta[i]\}$
  Algorithms for changing the base exists ($O(n^6)$)

Schreier-Sims Table with Base

$$\beta = [1, 5, 7, 2, 9, 8, 3, 6, 4]$$



	1	2	3	4	5	6	7	8	9
1	I		V				R_{270}		R_{180}
2		I							
3			I						
4				I					
5					I				
6						I			
7			D				I		
8								I	
9									I

Re-usable code: Binary variables only

```
clean_orbit_in_xstab( int g_deg,
mygroup g,
int *cv_orb,
int *list_orb,
int *card_list_orb,
int base_ind)
```

- Compute orbit of g→base[base_ind] in stabilizer of g→base[0..base_ind-1] in g
- returns 1 if no var in the orbit is set to 0 and base_ind can be fixed to 1
- returns 0 otherwise

Re-usable code: General integer variables

```
clean_korbit_in_xstab( int g_deg,
mygroup g,
int *cv_orb,
int *list_orb,
int *card_list_orb,
int base_ind,
int *max_val,       /* max val still allowed for each var */
int **max_val_date,  /* entry [w, j] # variables set to > 0
                         when value w was excluded for x_j*/
int *glob_stop,
int k_upbnd,        /* Upper bound for integer variables */
int **part_mat_orb)     /* if not NULL, store all orbits */
```

- Compute orbit of g→base[base_ind] in stabilizer of g→base[0..base_ind-1] in g
- returns 1 if no var in the orbit is set to 0 and base_ind can be fixed to max_val[base_ind]
- returns 0 otherwise

Computation of orb$(f, \mathrm{stab}(F_1^a, G))$

$$\beta = [\underbrace{., \ ., \ .,}_{F_1^a} \ \underbrace{f, \ ., \ .,}_{\text{free}} \ \underbrace{., \ ., \ .}_{F_0^a}]$$

Computation of orb$(f, \mathrm{stab}(F_1^a, G))$

$$\beta = [\underbrace{.,\ .,\ .,}_{F_1^a}\ \underbrace{f,\ .,\ .,}_{\text{free}}\ \underbrace{.,\ .,\ .}_{F_0^a}]$$

$$g \in \mathrm{stab}(F_1^a, G) \Rightarrow g = g_{\beta[1]} \cdot g_{\beta[2]} \cdot \ldots \cdot g_{\beta[|F_1^a|]} \cdot g_f \cdot h$$

with

- $g(\beta[i])$ in row $\beta[i]$ of T for $i = 1, \ldots, |F_1^a|$
- $g(f)$ in row f of T
- $h(f) = f$
- $h(\beta[i]) = \beta[i]$ for $i = 1, \ldots, |F_1^a|$

Computation of orb$(f, \mathrm{stab}(F_1^a, G))$

$$\beta = [\underbrace{., \ ., \ .,}_{F_1^a} \ \underbrace{f, \ ., \ .,}_{\text{free}} \ \underbrace{., \ ., \ .}_{F_0^a}]$$

$$g \in \mathrm{stab}(F_1^a, G) \Rightarrow g = g_{\beta[1]} \cdot g_{\beta[2]} \cdot \ldots \cdot g_{\beta[|F_1^a|]} \cdot g_f \cdot h$$

with

- $g(\beta[i])$ in row $\beta[i]$ of T for $i = 1, \ldots, |F_1^a|$
- $g(f)$ in row f of T
- $h(f) = f$
- $h(\beta[i]) = \beta[i]$ for $i = 1, \ldots, |F_1^a|$

Use Backtracking to explore all

$$p = g_{\beta[1]} \cdot g_{\beta[2]} \cdot \ldots \cdot g_{\beta[|F_1^a|]} \cdot g_f$$

If $p(F_1^a) = F_1^a$, add $p(f)$ to the orbit of f

Complexity: $O(n \cdot |F_1^a|!)$

Computation of Stabilizer

Theorem [Luks], [Hoffman]

Computing stab(S, G) is as hard as deciding if two graphs are isomorphic

Algorithm: Backtracking similar to previous one.

Complexity: $O(n \cdot |F_1^a|!)$