

DM841  
Discrete Optimization

Part I  
Lecture 24  
**Set Variables**

Marco Chiarandini

Department of Mathematics & Computer Science  
University of Southern Denmark

# Resume and Outlook

- ▶ Modeling in CP
- ▶ Global constraints (declaration)
- ▶ Notions of local consistency
- ▶ Global constraints (operational: filtering algorithms)
- ▶ Search
- ▶ Set variables
- ▶ Symmetry breaking

1. Symmetries in CSPs
2. Group theory
3. Avoiding symmetries
  - ...by Reformulation
  - ...by static Symmetry Breaking
  - ...during Search (SBDS)
  - ...by Dominance Detection (SBDD)

## Example

$$\mathcal{P} = \langle x_i \in \{1 \dots 3\}, \forall i = 1, \dots, 3; \mathcal{C} \equiv \{x_1 = x_2 + x_3\} \rangle$$

Solutions: (2, 1, 1), (3, 1, 2), (3, 2, 1).

Because of the symmetric nature of the plus operator, swapping the values of  $x_2$  and  $x_3$  gives rise to *equivalent* solutions.

- ▶ Many constraint satisfaction problem models have symmetries (some examples in a few slides)
- ▶ Breaking symmetry reduces search by avoiding to explore equivalent states (half of the search tree in the previous case)
- ▶ Inducing a preference on a (possibly singleton) subset of each solution equivalence class

# Symmetry Example: Social Golfer Problem

## Problem statement

Given:

- ▶  $g$  groups of
- ▶  $s$  golf players,
- ▶ and  $w$  weeks.

All players play once a week and we do not want that two player play in the same group more than once.

A possible model (different from the two previously seen) considers a three-dimensional matrix  $X_{ijk}$   $i \in \{1, \dots, w\}, j \in \{1, \dots, g\}, k \in \{1, \dots, s\}$  of integer variables  $\{1, \dots, g \times s\}$  representing the player playing as  $k$ -th player during week  $i$  in group  $j$ .

# Symmetry Example: Social Golfer Problem

- ▶  $g = 5$
- ▶  $s = 3$
- ▶  $\rightsquigarrow$  players 0..14
- ▶  $w = 7$

	group 1			group 2			group 3			group 4			group 5		
week 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
week 2	0	3	6	1	4	9	2	7	12	5	10	13	8	11	14
week 3	0	4	13	1	3	11	2	6	10	5	8	12	7	9	14
week 4	0	5	14	1	10	12	2	3	8	4	7	11	6	9	13
week 5	0	7	10	1	8	13	2	4	14	3	9	12	5	6	11
week 6	0	8	9	1	5	7	2	11	13	3	10	14	4	6	12
week 7	0	11	12	1	6	14	2	5	9	3	7	13	4	8	10

# Symmetry Example: Social Golfer Problem

Symmetries in CSPs  
Group Theory  
Avoiding symmetries

Permuting position in group

	group 1			group 2			group 3			group 4			group 5		
week 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
week 2	0	3	6	1	4	9	2	7	12	5	10	13	8	11	14
week 3	0	4	13	1	3	11	2	6	10	5	8	12	7	9	14
week 4	0	5	14	1	10	12	2	3	8	4	7	11	6	9	13
week 5	0	7	10	1	8	13	2	4	14	3	9	12	5	6	11
week 6	0	8	9	1	5	7	2	11	13	3	10	14	4	6	12
week 7	0	11	12	1	6	14	2	5	9	3	7	13	4	8	10

# Symmetry Example: Social Golfer Problem

Symmetries in CSPs  
Symmetry theory  
Avoiding symmetries

Permuting position in group

	group 1			group 2			group 3			group 4			group 5		
week 1	2	1	0	3	4	5	6	7	8	9	10	11	12	13	14
week 2	6	3	0	1	4	9	2	7	12	5	10	13	8	11	14
week 3	13	4	0	1	3	11	2	6	10	5	8	12	7	9	14
week 4	14	5	0	1	10	12	2	3	8	4	7	11	6	9	13
week 5	10	7	0	1	8	13	2	4	14	3	9	12	5	6	11
week 6	9	8	0	1	5	7	2	11	13	3	10	14	4	6	12
week 7	12	11	0	1	6	14	2	5	9	3	7	13	4	8	10



# Symmetry Example: Social Golfer Problem

Symmetries in CSPs  
Symmetry theory  
Avoiding symmetries

Permuting groups

	group 1			group 2			group 3			group 4			group 5		
week 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
week 2	0	3	6	1	4	9	2	7	12	5	10	13	8	11	14
week 3	0	4	13	1	3	11	2	6	10	5	8	12	7	9	14
week 4	0	5	14	1	10	12	2	3	8	4	7	11	6	9	13
week 5	0	7	10	1	8	13	2	4	14	3	9	12	5	6	11
week 6	0	8	9	1	5	7	2	11	13	3	10	14	4	6	12
week 7	0	11	12	1	6	14	2	5	9	3	7	13	4	8	10

# Symmetry Example: Social Golfer Problem

Symmetries in CSPs  
Group Theory  
Avoiding symmetries

Permuting groups

	group 1			group 2			group 3			group 4			group 5		
week 1	0	1	2	9	10	11	6	7	8	3	4	5	12	13	14
week 2	0	3	6	5	10	13	2	7	12	1	4	9	8	11	14
week 3	0	4	13	5	8	12	2	6	10	1	3	11	7	9	14
week 4	0	5	14	4	7	11	2	3	8	1	10	12	6	9	13
week 5	0	7	10	3	9	12	2	4	14	1	8	13	5	6	11
week 6	0	8	9	3	10	14	2	11	13	1	5	7	4	6	12
week 7	0	11	12	3	7	13	2	5	9	1	6	14	4	8	10

# Symmetry Example: Social Golfer Problem

Symmetries in CSPs  
Symmetry theory  
Avoiding symmetries

Permuting weeks

	group 1			group 2			group 3			group 4			group 5		
week 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
week 2	0	3	6	1	4	9	2	7	12	5	10	13	8	11	14
week 3	0	4	13	1	3	11	2	6	10	5	8	12	7	9	14
week 4	0	5	14	1	10	12	2	3	8	4	7	11	6	9	13
week 5	0	7	10	1	8	13	2	4	14	3	9	12	5	6	11
week 6	0	8	9	1	5	7	2	11	13	3	10	14	4	6	12
week 7	0	11	12	1	6	14	2	5	9	3	7	13	4	8	10

# Symmetry Example: Social Golfer Problem

Symmetries in CSPs  
Symmetry theory  
Avoiding symmetries

Permuting weeks

	group 1			group 2			group 3			group 4			group 5		
week 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
week 2	0	7	10	1	8	13	2	4	14	3	9	12	5	6	11
week 3	0	4	13	1	3	11	2	6	10	5	8	12	7	9	14
week 4	0	5	14	1	10	12	2	3	8	4	7	11	6	9	13
week 5	0	3	6	1	4	9	2	7	12	5	10	13	8	11	14
week 6	0	8	9	1	5	7	2	11	13	3	10	14	4	6	12
week 7	0	11	12	1	6	14	2	5	9	3	7	13	4	8	10

# Symmetry Example: Social Golfer Problem

Symmetries in CSPs  
Symmetry theory  
Avoiding symmetries

Permuting players

	group 1			group 2			group 3			group 4			group 5		
week 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
week 2	0	3	6	1	4	9	2	7	12	5	10	13	8	11	14
week 3	0	4	13	1	3	11	2	6	10	5	8	12	7	9	14
week 4	0	5	14	1	10	12	2	3	8	4	7	11	6	9	13
week 5	0	7	10	1	8	13	2	4	14	3	9	12	5	6	11
week 6	0	8	9	1	5	7	2	11	13	3	10	14	4	6	12
week 7	0	11	12	1	6	14	2	5	9	3	7	13	4	8	10

# Symmetry Example: Social Golfer Problem

Symmetries in CSPs  
Symmetry theory  
Avoiding symmetries

Permuting players

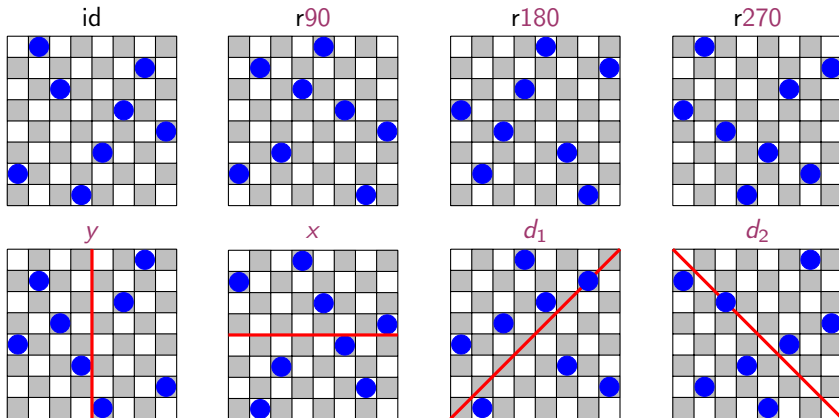
	group 1			group 2			group 3			group 4			group 5		
week 1	0	1	2	3	4	5	6	9	8	7	10	11	12	13	14
week 2	0	3	6	1	4	7	2	9	12	5	10	13	8	11	14
week 3	0	4	13	1	3	11	2	6	10	5	8	12	9	7	14
week 4	0	5	14	1	10	12	2	3	8	4	9	11	6	7	13
week 5	0	9	10	1	8	13	2	4	14	3	7	12	5	6	11
week 6	0	8	7	1	5	9	2	11	13	3	10	14	4	6	12
week 7	0	11	12	1	6	14	2	5	7	3	9	13	4	8	10

# Symmetry Example: Social Golfer Problem

Number of (equivalent) solutions:

- ▶ Permuting positions:  $3! \cdot 5 = 30$
- ▶ Permuting groups:  $5! = 120$
- ▶ Permuting weeks:  $7! = 5040$
- ▶ Permuting players:  $15! = 1,307,674,368,000$

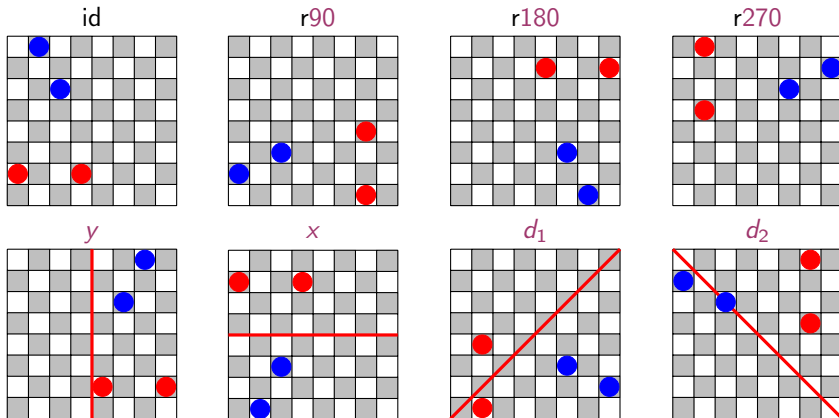
# Symmetry Example: $n$ -Queens





# Symmetry Example: $n$ -Queens

Symmetric failure



# Symmetries: general considerations

- ▶ Widespread
  - ▶ Inherent in the problem ( $n$ -Queens, chessboard)
  - ▶ Artifact of the model (Social Golfer: order of players in groups)
- ▶ Different types:
  - ▶ variable symmetry (swapping variables)
  - ▶ value symmetry (permuting values)

# Types of symmetries

- ▶ **Variable symmetry:** permuting variables is solution invariant

$$\{x_i = v_i\} \in \text{sol}(P) \iff \{x_{\sigma(i)} = v_i\} \in \text{sol}(P)$$

- ▶ **Value symmetry:** permuting values is solution invariant

$$\{x_i = v_i\} \in \text{sol}(P) \iff \{x_i = \sigma(v_i)\} \in \text{sol}(P)$$

- ▶ **Variable/value symmetry:** both variables and values permutation is solution invariant

$$\{x_i = v_i\} \in \text{sol}(P) \iff \{x_{\sigma_1(i)} = \sigma_2(v_i)\} \in \text{sol}(P)$$

1. Symmetries in CSPs
2. Group theory
3. Avoiding symmetries
  - ...by Reformulation
  - ...by static Symmetry Breaking
  - ...during Search (SBDS)
  - ...by Dominance Detection (SBDD)

# Group basics

## Group

A set  $G$  and an associated operation  $\otimes$  form a **group** if

- ▶  $G$  is closed under  $\otimes$ , i.e.,  $a, b \in G \Rightarrow a \otimes b \in G$
- ▶  $\otimes$  is associative, i.e.,  $a, b, c \in G \Rightarrow (a \otimes b) \otimes c = a \otimes (b \otimes c)$
- ▶  $G$  has an identity  $\iota_G$ , such that  $a \in G \Rightarrow a \otimes \iota_G = \iota_G \otimes a = a$
- ▶ every element has an inverse, i.e.,  
 $a \in G \Rightarrow \exists a^{-1} : a \otimes a^{-1} = a^{-1} \otimes a = \iota_G$

# Permutations

Permutation representations:

Cauchy's two-line notation:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 7 & 4 & 1 & 8 & 5 & 2 & 9 & 6 & 3 \end{pmatrix}$$

element 1 maps to 7, 7 to 9, 9 to 3,  
3 to 1.

Cycle notation:

$$(2, 4, 6, 8)(1, 3, 9, 7)(5)$$

set of cycles derived from the  
two-line notation indicating the  
mapping, ie, 2 becomes 4, 4  
becomes 6, etc.

The set of all **permutations** of a finite set  $S$  of objects together with **composition** form a group.

Group properties for permutations with composition  $\circ$  as operation. Let  $f$  and  $g$  be two permutations,  $p$  a point:

- ▶  $f \circ g$  composition
- ▶  $p^{f \circ g} = (p^f)^g$
- ▶  $id = \iota$
- ▶  $f \circ f^{-1} = id$  inverse (in Cauchy form, swap the two rows and reorder the first; in cycle notation, reverse the order of each cycle.)
- ▶  $f \circ (g \circ h) = (f \circ g) \circ h$

- ▶  $|G|$  is the order of a group, ie, number of elements in the set  $G$
- ▶ Set  $S_n$  of all permutations of  $n$  objects is called a **symmetry group** over  $n$  elements.  $|S_n| = n!$
- ▶ Any subgroup of a permutation group defines a **permutation group**
- ▶ The set of symmetries in  $n$ -queens defines a permutation group:  
 $\{id, r90, r180, r270, x, y, d_1, d_2\}$
- ▶ symmetries define a permutation of a set of points.
- ▶  $p$  a point in the solution space,  $g \in G$  a permutation,  $p^g$  the point to which  $p$  is moved under  $g$ . Eg:  $\{1, 3, 8\}^{r90} = \{1^{r90}, 3^{r90}, 8^{r90}\} = \{7, 1, 6\}$



# Generators

## Generators

A set  $S \subseteq G$  is called a **generator** of group  $G$  iff

$$\forall g \in G \quad \exists S' \subseteq S : g = \bigotimes_{s \in S'} s$$

Generators describe groups in a compact form.

For example:

- ▶ Generator of chessboard symmetries:  $\{r90, d1\}$
- ▶  $G = \langle s \rangle$
- ▶ There is always a generator of  $\log_2(|G|)$  size or smaller.

## Orbits

The **orbit** of an element with respect to a permutation group  $G$  is

$$O^G = \{p^g \mid g \in G\}$$

The orbit of a set of elements (called also **points**) is defined accordingly.

Orbits are the set of elements encountered by starting from one element and moving through different permutations.

1. Symmetries in CSPs
2. Group theory
3. **Avoiding symmetries**
  - ...by Reformulation
  - ...by static Symmetry Breaking
  - ...during Search (SBDS)
  - ...by Dominance Detection (SBDD)

# How to avoid symmetry

Never explore a state that is the symmetric of one already explored

- ▶ Model reformulation
- ▶ Addition of constraints (static symmetry breaking)
- ▶ During search (dynamic symmetry breaking)
- ▶ By dominance detection (dynamic symmetry breaking)

1. Symmetries in CSPs
2. Group theory
3. Avoiding symmetries
  - ...by Reformulation
  - ...by static Symmetry Breaking
  - ...during Search (SBDS)
  - ...by Dominance Detection (SBDD)

# Model reformulation

- ▶ Use set variables (inherently unordered)
  - ▶ In the Social Golfers example: groups can be represented as sets
  - ▶ Only within group symmetry has been removed, but not the groups/weeks/player ones
- ▶ Solve a different problem (try to redefine the problem avoiding symmetries)
- ▶ Solve the dual problem

## Solve a different problem: example

A series is a sequence of twelve tone names (pitch classes) of the chromatic scale, in which each pitch class occurs exactly once. In an all-interval series, also all eleven intervals between the twelve pitches are pairwise distinct.

### All-different series

In general words, we are asked to find a permutation of the integers  $\{0, \dots, n\}$ , such that the differences between adjacent numbers are a permutation of  $\{1, \dots, n\}$ .

0	10	1	9	2	8	3	7	4	6	5
10	9	8	7	6	5	4	3	2	1	

The problem has many symmetric solutions, e.g. reverse values, “invert” from 10, **shifting** (according to a pivot), ...

0	10	1	9	2	8	3	7	4	6	5
10	9	8	7	6	5	4	3	2	1	
3	7	4	6	5	0	10	1	9	2	8
4	3	2	1	5	10	9	8	7	6	

# Solve a different problem: example

## All-different series: new formulation

Find a permutation of the integers  $\{0, \dots, n\}$  such that:

- ▶ the permutation starts with 0,  $n$ , 1
- ▶ the differences  $|x_{i+1} - x_i|$  and  $|x_n - x_0|$  are in  $\{1, \dots, n\}$
- ▶ exactly one difference occurs twice

This extracts solutions from the original problem with a specific structure



# Solve dual problem

- ▶ Mainly for value symmetries
- ▶ Example: players in golfers
- ▶ Consider the dual problem w.r.t. each value  $v$ 
  - ▶ Introduce a set  $X_v$  such that

$$i \in X_v \iff y_i = v$$

( $y_i$  are the original variables)

- ▶ Applicable when constraints can be stated easily on the dual problem

1. Symmetries in CSPs
2. Group theory
3. Avoiding symmetries
  - ...by Reformulation
  - ...by static Symmetry Breaking
  - ...during Search (SBDS)
  - ...by Dominance Detection (SBDD)

# Symmetry breaking constraints

- ▶ Rule out symmetric solutions by adding further constraints to the original model.
- ▶ Assumption: domains are ordered

## Lex-leader constraints

Let  $\Sigma$  be the set of all variable symmetry permutations  
These symmetries are broken by imposing:

$$[x_1, \dots, x_n] \preceq_{lex} [x_{\sigma(1)}, \dots, x_{\sigma(n)}], \quad \forall \sigma \in \Sigma$$

Only the lexicographically smallest solution, called **lex-leader** is preserved

- ▶ Distinct integers,  $\sigma(1) \neq 1$ :  
 $[x_1, \dots, x_n] \preceq_{lex} [x_{\sigma(1)}, \dots, x_{\sigma(n)}] \iff x_1 < x_{\sigma(1)}$
- ▶ Disjoint integer sets,  $\sigma(1) \neq 1$ :  
 $[x_1, \dots, x_n] \preceq_{lex} [x_{\sigma(1)}, \dots, x_{\sigma(n)}] \iff \min(x_1) < \min(x_{\sigma(1)})$
- ▶ Arbitrary integers or sets: special propagators

# Lex-leader constraints: examples

- ▶  $n$ -Queens:  $\sigma(i) = n - i + 1$  (eliminate symmetry rotation on  $y$ )

$$[q_1, \dots, q_n] \preceq_{\text{lex}} [q_{\sigma(1)}, \dots, q_{\sigma(n)}] = [q_n, \dots, q_1]$$

$$\implies q_1 < q_n$$

- ▶ All-Intervals:

$$|x_2 - x_1| > |x_n - x_{n-1}|$$

# In Gecode

- ▶ Lexicographic constraints between variable arrays. (where the sizes of  $x$  and  $y$  can be different), If  $x$  and  $y$  are integer variable arrays

```
rel(home, x, IRT_LE, y);
```

- ▶  $x$  is an array of set variables and  $c$  is an array of integers

```
precede(home, x, c);
```

it is enforced that  $c_k$  precedes  $c_{k+1}$  in  $x$  for  $0 \leq k < |c| - 1$

- ▶ Using set variables to model the groups avoids introducing symmetry among the players in a group.

```
SetVarArray groups(home,g*w,IntSet::empty,0,g*s-1,s,s);  
Matrix<SetVarArray> schedule(groups,g,w);
```

- ▶ Within a week, the order of the groups is irrelevant. Static order requiring that all minimal elements of each group are ordered increasingly  $\min(\text{schedule}(g, w)) < \min(\text{schedule}(g + 1, w))$

```
for (int j=0; j<w; j++) {  
  IntVarArgs m(g);  
  for (int i=0; i<g; i++)  
    m[i] = expr(home, min(schedule(i,j)));  
  rel(home, m, IRT_LE);  
}
```

- ▶ similarly, the order of the weeks is irrelevant, hence order on group elements (remove  $\{0\}$  as from above it will be always in  $\text{schedule}(0,j)$ )

```
IntVarArgs m(w);  
for (int j=0; j<w; j++)  
  m[j] = expr(home, min(schedule(0,j)-IntSet(0,0)));  
rel(home, m, IRT_LE);
```

- ▶ the players can be permuted arbitrarily.

```
precede(home, groups, IntArgs::create(g*s, 0));
```

$c = (0, \dots, 14)$ : It enforces that for any pair of players  $c_k$  and  $c_{k+1}$ ,  $0 \leq k \leq 14$ ,  $c_{k+1}$  can appear in a group without  $c_k$  only if there is an earlier group where  $c_k$  appears without  $c_{k+1}$ . Eg, 9 appears in a group without 7 but 7 should appear earlier, hence the constraint is not satisfied.

	group 1			group 2			group 3			group 4			group 5		
week 1	0	1	2	3	4	5	6	9	8	7	10	11	12	13	14
week 2	0	3	6	1	4	7	2	9	12	5	10	13	8	11	14
week 3	0	4	13	1	3	11	2	6	10	5	8	12	9	7	14
week 4	0	5	14	1	10	12	2	3	8	4	9	11	6	7	13
week 5	0	9	10	1	8	13	2	4	14	3	7	12	5	6	11
week 6	0	8	7	1	5	9	2	11	13	3	10	14	4	6	12
week 7	0	11	12	1	6	14	2	5	7	3	9	13	4	8	10

# Value symmetries

- ▶ Same idea:

$$[x_1, \dots, x_n] \preceq_{lex} [\sigma(x_1), \dots, \sigma(x_n)], \quad \forall \sigma \in \Sigma$$

- ▶ how to implement  $\sigma(x_i)$ ?
- ▶ element constraint to implement  $\sigma(x_i)$

## Example

$$\sigma(v) = n - v$$

3	7	4	6	5	0	10	1	9	2	8
4	3	2	1	5	10	9	8	7	6	
7	3	6	4	5	10	0	9	1	8	2
4	3	2	1	5	10	9	8	7	6	

$$\sigma = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$$

$$[x_0, \dots, x_{n-1}] \preceq_{lex} [\sigma(x_0), \dots, \sigma(x_{n-1})] \iff x_0 < \sigma(x_0) \iff x_0 < x_1$$



# Pros and Cons

- ▶ Good: for each symmetry, only one solution remains
- ▶ Bad:
  - may have to add many constraints
  - remaining solution may not be the first one according to branching heuristic!
- ▶ Especially bad with dynamic variable selection (like first-fail heuristics)

1. Symmetries in CSPs
2. Group theory
3. **Avoiding symmetries**
  - ...by Reformulation
  - ...by static Symmetry Breaking
  - ...**during Search (SBDS)**
  - ...by Dominance Detection (SBDD)

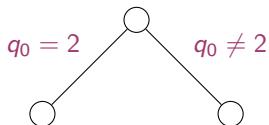
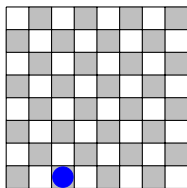
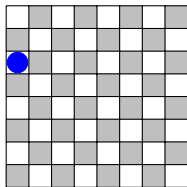
# Symmetry Breaking During Search (SBDS)

- ▶ Add constraints during backtracking to prevent the visit of symmetric search states
- ▶ Similar idea to branch-and-bound
- ▶ Pros: Works with every type of symmetry
- ▶ Cons: Can result in a huge number of constraints to be added, and all symmetries have to be specified explicitly

# SBDS Example: $n$ -Queens

Goal: Eliminate r90:

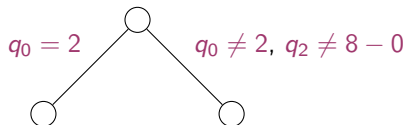
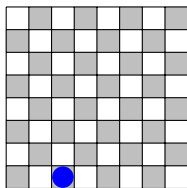
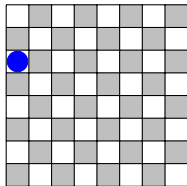
$$\{q_i = j\} \in \text{sol}(n\text{-Queens}) \iff \{q_j = n - i\} \in \text{sol}(n\text{-Queens})$$



# SBDS Example: $n$ -Queens

Goal: Eliminate r90:

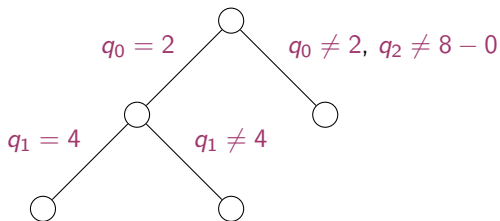
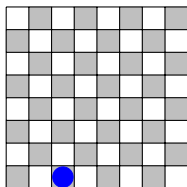
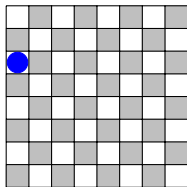
$$\{q_i = j\} \in \text{sol}(n\text{-Queens}) \iff \{q_j = n - i\} \in \text{sol}(n\text{-Queens})$$



# SBDS Example: $n$ -Queens

Goal: Eliminate r90:

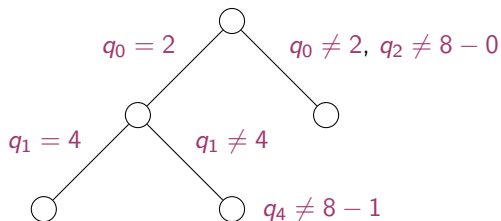
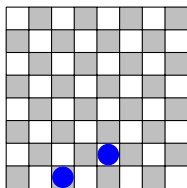
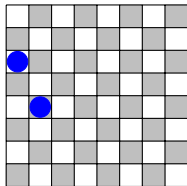
$$\{q_i = j\} \in \text{sol}(n\text{-Queens}) \iff \{q_j = n - i\} \in \text{sol}(n\text{-Queens})$$



SBDS Example:  $n$ -Queens

Goal: Eliminate r90:

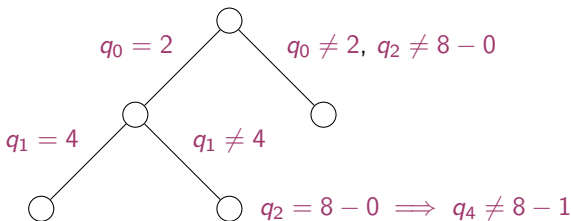
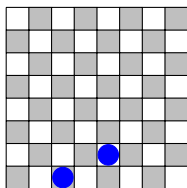
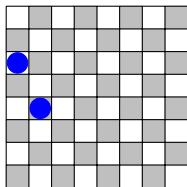
$$\{q_i = j\} \in \text{sol}(n\text{-Queens}) \iff \{q_j = n - i\} \in \text{sol}(n\text{-Queens})$$



# SBDS Example: $n$ -Queens

Goal: Eliminate  $r_{90}$ :

$$\{q_i = j\} \in \text{sol}(n\text{-Queens}) \iff \{q_j = n - i\} \in \text{sol}(n\text{-Queens})$$



Too strict: we need to post the whole path:

$$\neg(q_0 = 2 \wedge q_1 = 4) \rightsquigarrow (q_0 = 2 \implies q_1 \neq 4)^{r_{90}}$$



# SBDS in group theory perspective

$$(A \implies \text{var} \neq \text{val}) \implies (A \implies \text{var} \neq \text{val})^g$$

We do not need to add the full form. We operate dynamically:  
At the choice point  $c$  backtracking from  $\text{var} = \text{val}$  we know that  $A$  is true and  $\text{var} \neq \text{val}$  is also true, hence we add:

$$A^g \implies (\text{var} \neq \text{val})^g$$

## SBDS

For each symmetry  $g$ , and a current partial assignment  $A$  and choice  $c$ , post the constraint:

$$g(A) \implies \neg g(c)$$

Only interested in different  $g(A)$  and  $g(c)$

- compute the orbit of the current partial assignment  $A$

# Lightweight Dynamic Symmetry Breaking

In Gecode

Symmetries in CSPs  
Constraint theory  
Avoiding symmetries

Dynamic symmetry breaking: given a specification of the symmetries, avoid visiting symmetric states during the search

- ▶ break value symmetry (that is, values that are interchangeable)

```
Symmetries syms;  
syms << ValueSymmetry(IntArgs::create(n,0));  
branch(* this, x, INT_VAR_NONE(), INT_VAL_MIN(), syms);
```

- ▶ break variable symmetry (that is, certain sequences of variables are interchangeable):

```
IntVarArgs rows;  
for (int r = 0; r < m.height(); r++)  
    rows << m.row(r);  
syms << VariableSequenceSymmetry(rows, m.width());  
IntVarArgs cols;  
for (int c = 0; c < m.width(); c++)  
    cols << m.col(c);  
syms << VariableSequenceSymmetry(cols, m.height());
```

- ▶ See sec. 8.10.1 for other possibilities
- ▶ Combining LDSB with other forms of symmetry breaking — such as static ordering constraints — can cause the search to miss some sol.

1. Symmetries in CSPs
2. Group theory
3. **Avoiding symmetries**
  - ...by Reformulation
  - ...by static Symmetry Breaking
  - ...during Search (SBDS)
  - ...by Dominance Detection (SBDD)

# Symmetry Breaking by Dominance Detection (SBDD)

- ▶ Do not explore subtrees **dominated** by a previously visited node
- ▶ Multiple definitions of *dominance* are possible
- ▶ Pros: No constraints added, very configurable
- ▶ Cons: Storage of previous states, checking dominance can be expensive

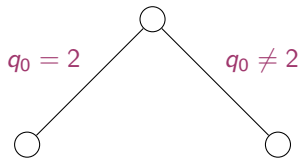
The idea is similar to *no goods*.  
It can be used for propagation.

# Ingredients

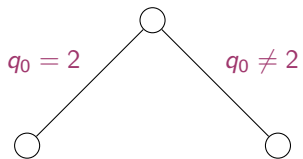
Idea: Perform a check at every node in the search tree to see if the node about to be explored is symmetrically equivalent to one already explored. If so prune this branch. Need only to store nodes at the root of fully explored subtrees.

- ▶ No-good: A node  $v$  is a no-good w.r.t. a node  $n$  if there exists an ancestor  $n_a$  of  $n$  s.t.  $v$  is the left hand child of  $n_a$  and  $v$  is not an ancestor of  $n$ .
- ▶  $\delta(v)$  set of decisions labelling the path from the root of the tree to the node  $v$
- ▶  $\Delta(v)$  set of variables whose domain is reduced to a singleton at node  $v$ .
- ▶ Dominance:  
a node  $n$  is dominated if there exists a no-good  $v$  w.r.t.  $n$  and a symmetry  $g$  s.t.  $(\delta(v))^g \subseteq \Delta(n)$
- ▶ Database  $T$  of already seen domains

# SBDD Example: $n$ -Queens

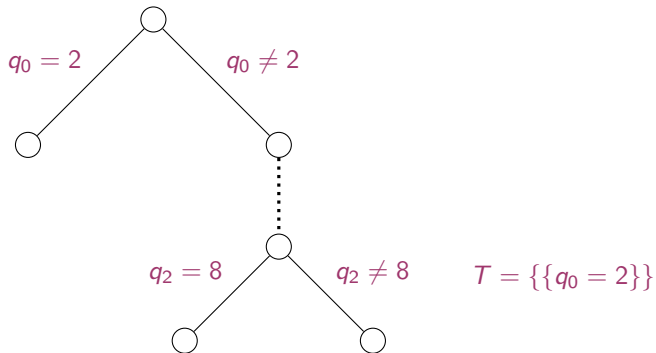


# SBDD Example: $n$ -Queens



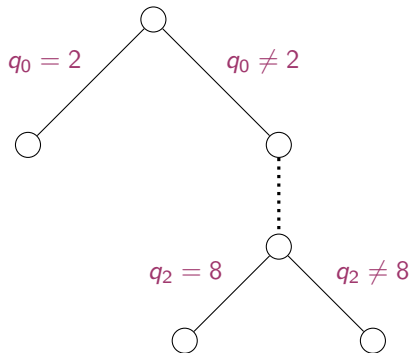
$$T = \{\{q_0 = 2\}\}$$

# SBDD Example: $n$ -Queens



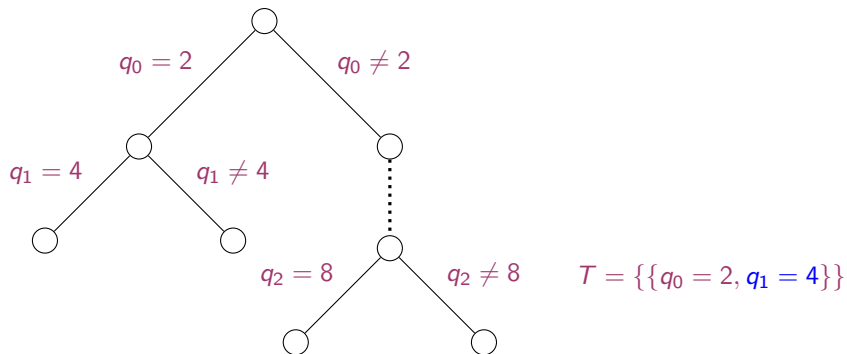


## SBDD Example: $n$ -Queens

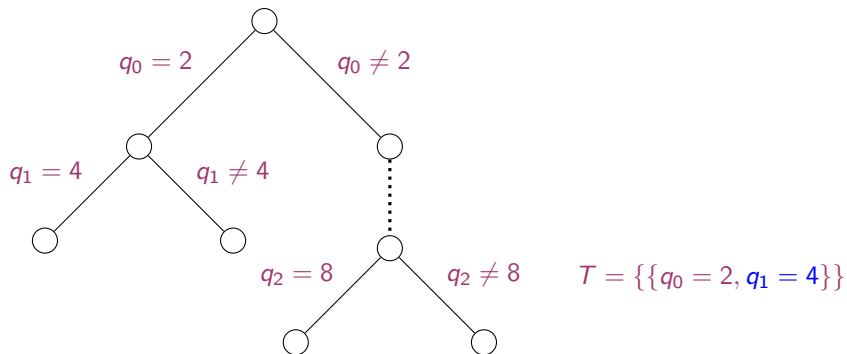


$T = \{\{q_0 = 2\}\}$   
Dominated

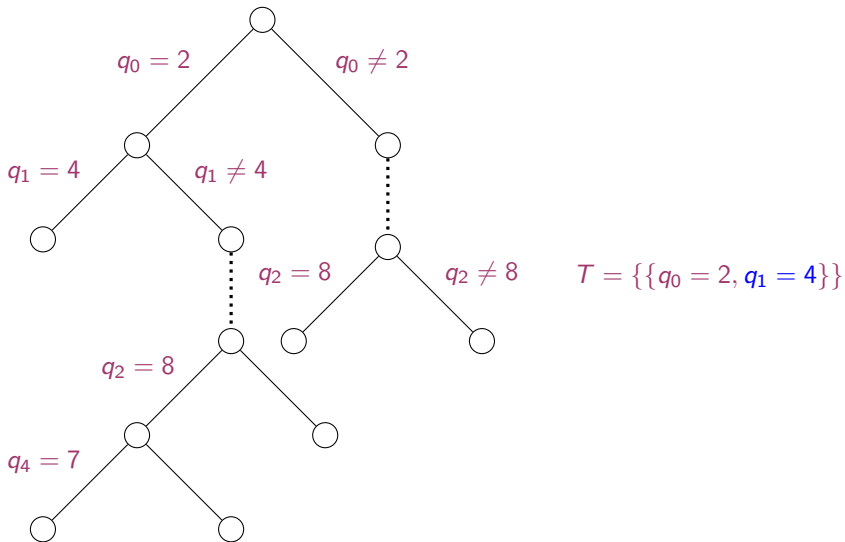
# SBDD Example: $n$ -Queens



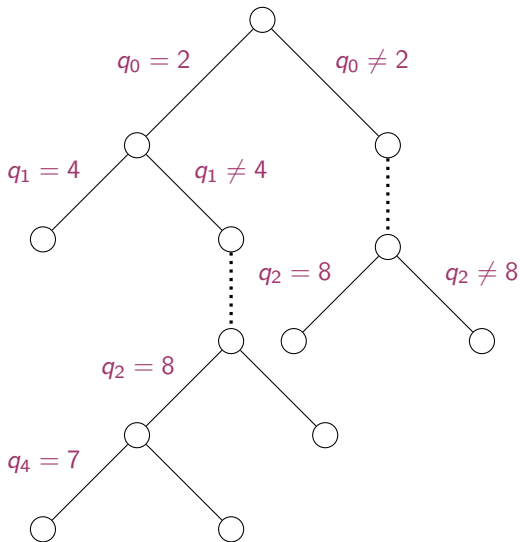
# SBDD Example: $n$ -Queens



# SBDD Example: $n$ -Queens



## SBDD Example: $n$ -Queens



$T = \{\{q_0 = 2, q_1 = 4\}\}$   
Dominated

# SBDD in the group theory perspective

## SBDD

A domain  $d$  dominates the current node  $c$  if  $c$  is in the orbit of  $d$

Detection:

function  $\Phi : \text{Dom} \times \text{Dom} \mapsto \mathbb{B}$

such that  $\Phi(\delta(v), \Delta(n)) = \text{true}$  iff  $\delta(v)$  dominates  $\Delta(n)$  under some symmetry  $\sigma$ .

Optimization: only keep domains left-adjacent to the path from the root to the current node

# Pros and Cons

- ▶ Good: No constraints added
- ▶ Good: Handles all kinds of symmetry
- ▶ Good: Very configurable (by implementing )
- ▶ Bad: Still all symmetries must be encoded
- ▶ Bad: Checking dominance at each node may be expensive

- Backofen W. (2002). **Excluding symmetries in constraint-based search.** *Constraints*, (3).
- Barnier N. and Brisset P. (2005). **Solving kirkman's schoolgirl problem in a few seconds.** *Constraints*, (10), pp. 7–21.
- Gent I.P., Petrie K.E., and Puget J.F. (2006). **Symmetry in constraint programming.** In *Handbook of Constraint Programming*, edited by F. Rossi, P. van Beek, and T. Walsh, chap. 10, pp. 329–376. Elsevier.