DM826 – Spring 2014

Modeling and Solving Constrained Optimization Problems

Lecture 13

Symmetries

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

[*Slides by Marco Kuhlmann, Guido Tack and Luca Di Gaspero*]

Resume

- Modelling in IP and CP
- Global constraints
- Local consistency notions
- Filtering algorithms for global constraints
  Scheduling
- Search
- Set variables
- Symmetries

Outline

1. Symmetries in CSPs

2. Group theory

3. Avoiding symmetries
    ...by Reformulation
    ...by static Symmetry Breaking
    ...during Search
    ...by Dominance Detection (SBDD)

Symmetries

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 raise 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 p golf players, and w weeks. All players plays once a week, and we do not want that two player play in the same group more than once.

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

Symmetry Example: Social Golfer Problem

	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
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

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
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

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

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

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
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
Permuting players

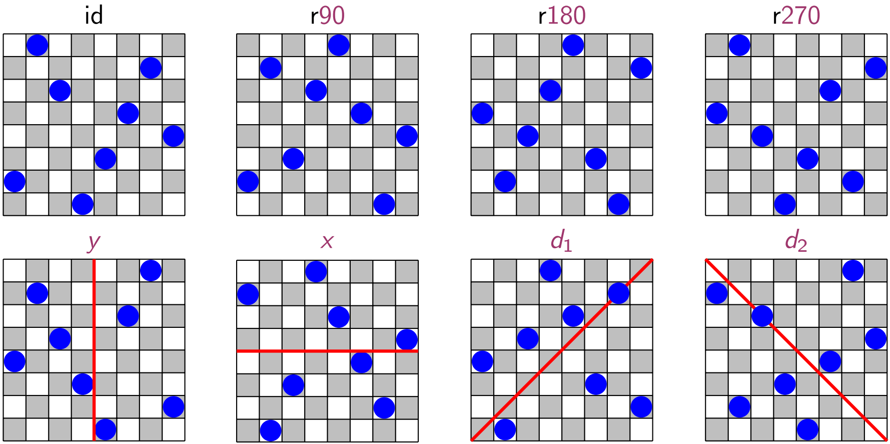
	group 1			group 2			group 3			group 4			group 5		
week 1	0	1	2	3	4	5	6	9	8	7	10	11	12	13	14
week 2	0	3	6	1	4	7	2	9	12	5	10	13	8	11	14
week 3	0	4	13	1	3	11	2	6	10	5	8	12	9	7	14
week 4	0	5	14	1	10	12	2	3	8	4	9	11	6	7	13
week 5	0	9	10	1	8	13	2	4	14	3	7	12	5	6	11
week 6	0	8	7	1	5	9	2	11	13	3	10	14	4	6	12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

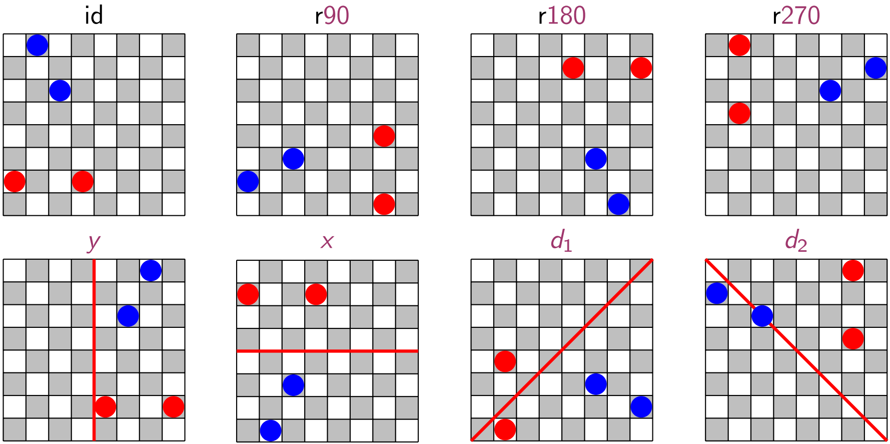
- Permuting positions: $3! = 6$
- Permuting groups: $5! = 120$
- Permuting weeks: $7! = 5040$
- Permuting players: $15! = 1,307,674,368,000$

Symmetry Example: *n*-Queens



Symmetry Example: *n*-Queens

Symmetric failure



Symmetries: general considerations

- Widespread
  - Inherent in the problem (n-Queens, chessboard)
  - Artefact 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

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

- Value symmetry: permuting values is solution invariant

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

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

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

Outline

1. Symmetries in CSPs

2. Group theory

3. Avoiding symmetries
   ...by Reformulation
   ...by static Symmetry Breaking
   ...during Search
   ...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 ι_G, such that $a \in G \Rightarrow a \otimes \iota_G = \iota_G \otimes a = a$
- every element has an inverse, i.e., $a \in G \Rightarrow \exists a^{-1} a \otimes a^{-1} = a^{-1} \otimes a = \iota_G$

The set of permutations forms a group, together with concatenation.

Generators

Generators

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

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

Generators describe groups in a compact form.
For example:

- symmetries of a square $\{r90, d1\}$
- permutations of $\{1, \ldots, n\}$: $\{(123 \ldots n), (12)\}$

Orbits

Orbits

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

$$O_G(g) = \{\sigma(g) | \sigma \in G\}$$

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.

Outline

1. Symmetries in CSPs

2. Group theory

3. Avoiding symmetries
   ...by Reformulation
   ...by static Symmetry Breaking
   ...during Search
   ...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)

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 no 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 required to find a permutation of the integers $\{0, \ldots, n\}$, such that the differences between adjacent numbers are a permutation of $\{1, \ldots, n\}$.

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 $\{0, \ldots, n\}$ such that:

- the permutation starts with 0, n, 1
- the differences $|x_{i+1} - x_i|$ and $|x_n - x_0|$ are in $\{1, \ldots, n\}$
- 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

Symmetry breaking constraints

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

Lex-leader constraints

Let Σ be the set of all variable symmetry permutations
These symmetry are broken by imposing:

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

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

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

Lex-leader constraints: examples

- n-Queens: $\sigma(i) = n - i + 1$

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

$$\Rightarrow q_1 < q_n$$

- 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 $0 \leq k < |c| - 1$

Social Golfers

In Gecode

- 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 $\min(groups(g, w)) < \min(group(g + 1, w))$

```
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 (remove $\{0\}$ or no effect)

```
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−1, 0)); \\ different from manual
```

$c = (0, \ldots, 14)$: It enforces that for any pair of players c_k and c_{k+1}, $0 \leq k \leq 14$ that c_{k+1} can only appear in a group without c_{k+1} if there is an earlier group where c_k appears without c_{k+1}. Eg, 8 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:

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

- 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

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

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

Pros and Cons

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

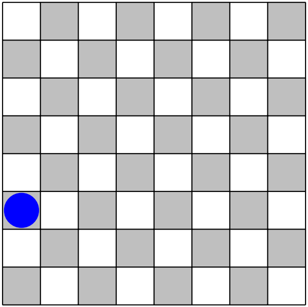
Symmetry Breaking During Search (SBDS)

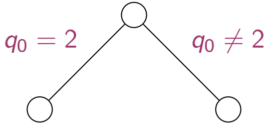
- Add constraints during backtracking to prevent the visit of symmetric search states
- Similar idea to branch-and-bound
- Pros: Works with every type of symmetry
- Cons: Can result in a huge nubmer of constraints to be added, and all symmetries have to be specified explicitly

SBDS Example: n-Queens

Goal: Eliminate r90:

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

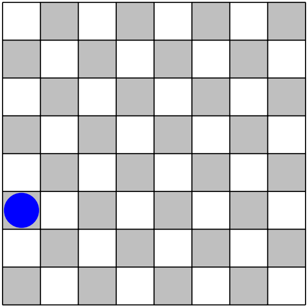


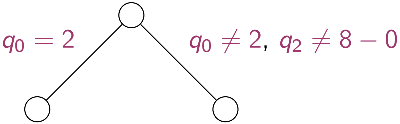


SBDS Example: n-Queens

Goal: Eliminate r90:

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

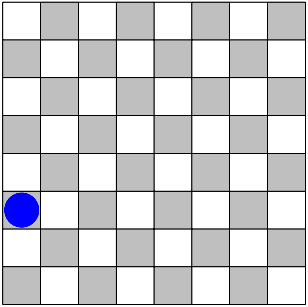


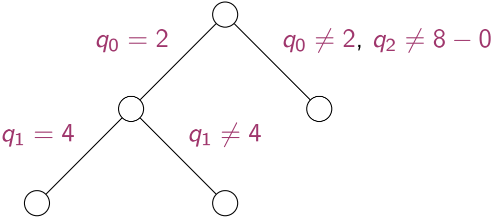


SBDS Example: n-Queens

Goal: Eliminate r90:

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

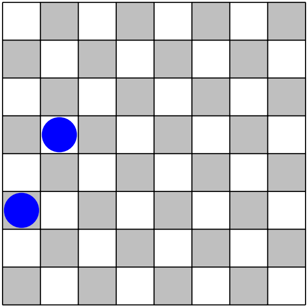


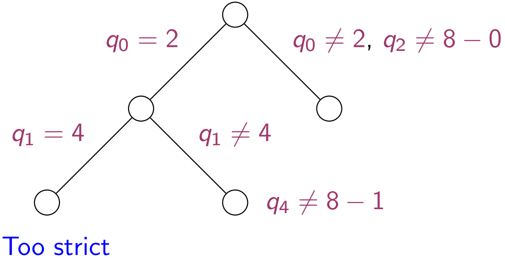


SBDS Example: n-Queens

Goal: Eliminate r90:

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

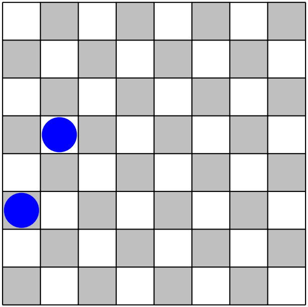


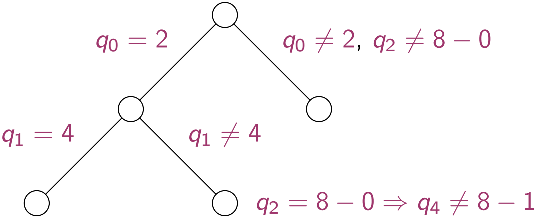


SBDS Example: n-Queens

Goal: Eliminate r90:

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





SBDS in group theory perspective

SBDS

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

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

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

- compute the orbit of the current partial assignment A

Symmetry Breaking by Dominance Detection (SBDD)

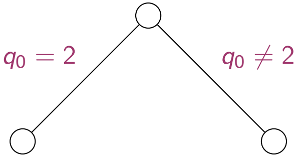
- Do not explore subtrees dominated by a previously visited node
- Multiple definitions of *dominance* are possible
- Pros: No constraints added, very configurable
- Cons: Storage of previous states, checking dominance can be expensive

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

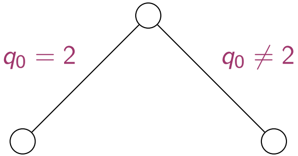
Ingredients

- No-good: A node v is a no-good w.r.t. a node n if there exists an acenstor n_a of n s.t. v is the l;eft hand child of n_a and v is not an ancestor of n.
- Dominance:
  a node n is dominated if there exists a no-good v w.r.t. n and a symmetry g s.t. $(\delta(v))^g \subseteq \mathcal{DE}(n)$
  ($\delta(v)$ set of decisions labelling the path from the root of the tree to the node v)
- Database T of already seen domains

SBDD Example: n-Queens

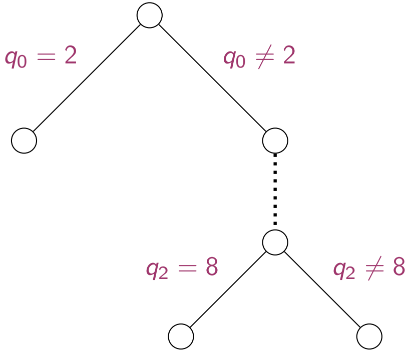


SBDD Example: n-Queens



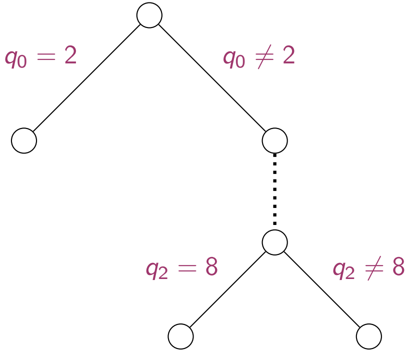
$T = \{\{q_0 = 2\}\}$

SBDD Example: n-Queens



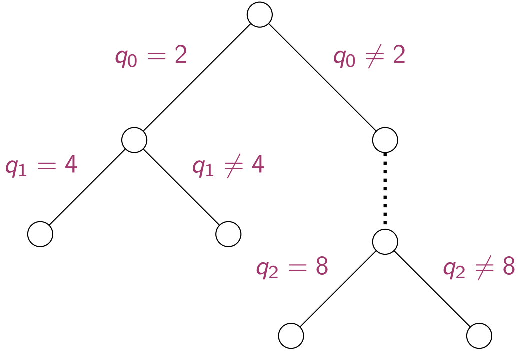
$T = \{\{q_0 = 2\}\}$

SBDD Example: n-Queens

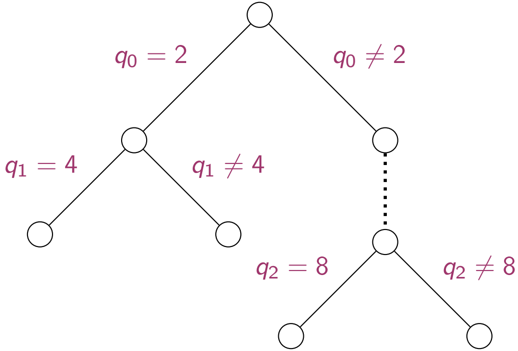


$T = \{\{q_0 = 2\}\}$
Dominated

SBDD Example: n-Queens

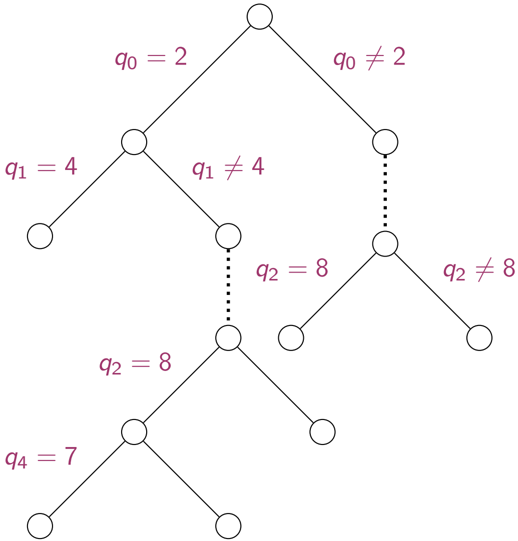


SBDD Example: n-Queens

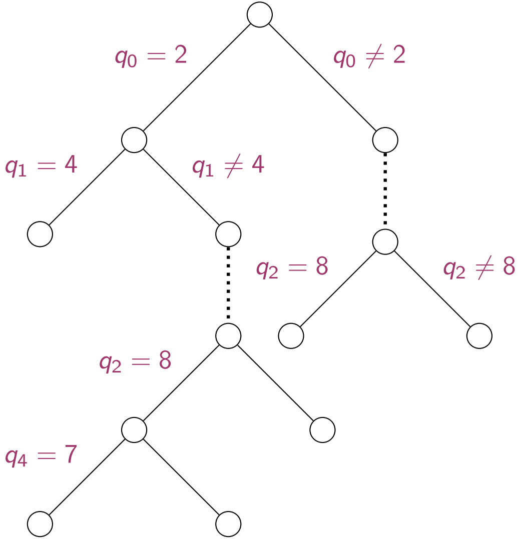


$T = \{\{q_0 = 2, q_1 = 4\}\}$

SBDD Example: n-Queens



SBDD Example: n-Queens



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 $\Phi : \mathrm{Dom} \times \mathrm{Dom} \mapsto \mathbb{B}$
such that $\Phi(\delta(v), \mathcal{DE}(n)) = \mathit{true}$ iff $\delta(v)$ dominates $\mathcal{DE}(n)$ under some symmetry σ.

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: V ery configurable (by implementing)
- Bad: Still all symmetries must be encoded
- Bad: Checking dominance at each node may be expensive

References

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.