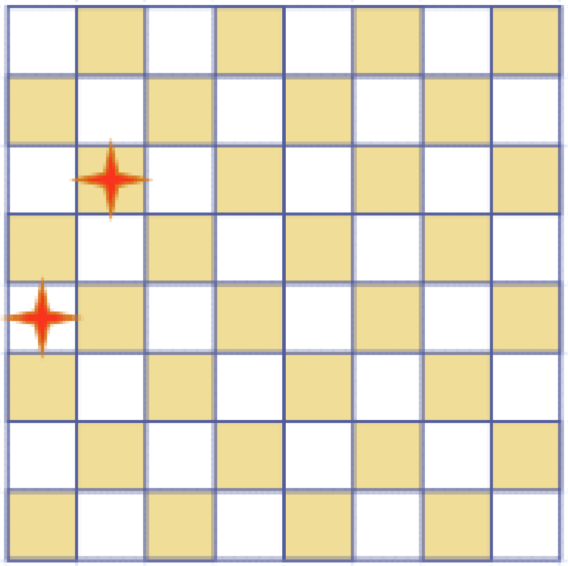
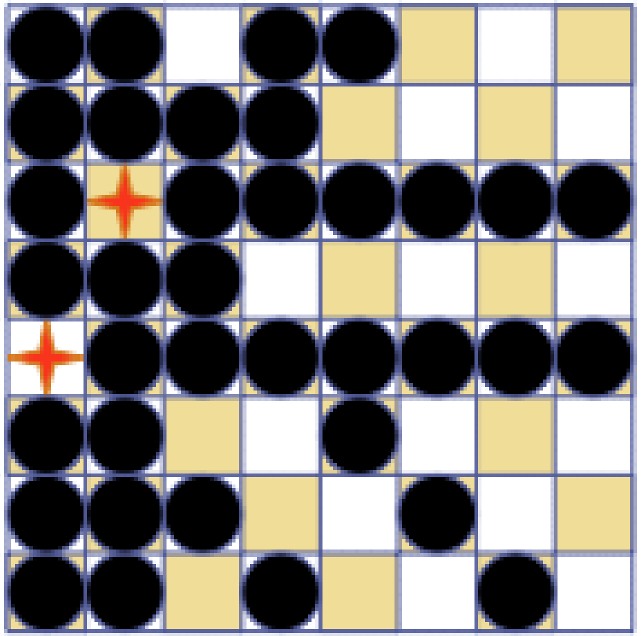
Constraint Satisfaction Problems

Intro Example: 8-Queens



Generate-and-test: 8^8 combinations

Intro Example: 8-Queens



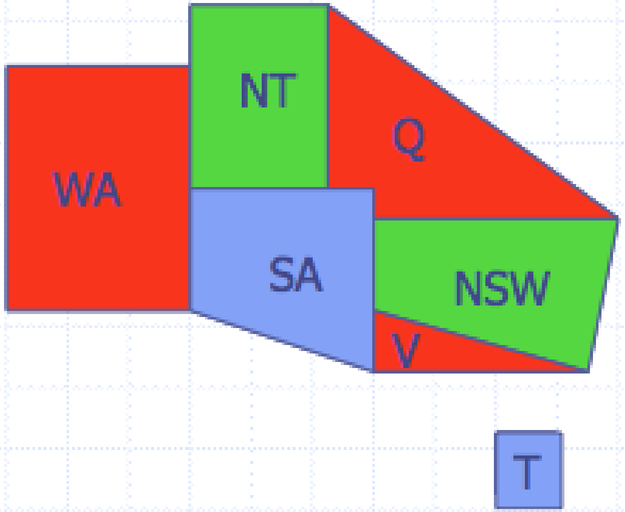
Constraint Satisfaction Problem

- Set of variables $\{X_1, X_2, \ldots, X_n\}$
- Each variable X_i has a domain D_i of possible values
  - Usually D_i is discrete and finite
- Set of constraints $\{C_1, C_2, \ldots, C_p\}$
  - Each constraint C_k involves a subset of variables and specifies the allowable combinations of values of these variables
- Assign a value to every variable such that all constraints are satisfied

Example: 8-Queens Problem

- 8 variables X_i, i = 1 to 8
- Domain for each variable {1,2,…,8}
- Constraints are of the forms:
  - $X_i = k \rightarrow X_j \neq k$ for all j = 1 to 8, j≠i
  - $X_i = k_i, X_j = k_j \rightarrow |i-j| \neq | k_i - k_j|$
    - for all j = 1 to 8, j≠i

Example: Map Coloring

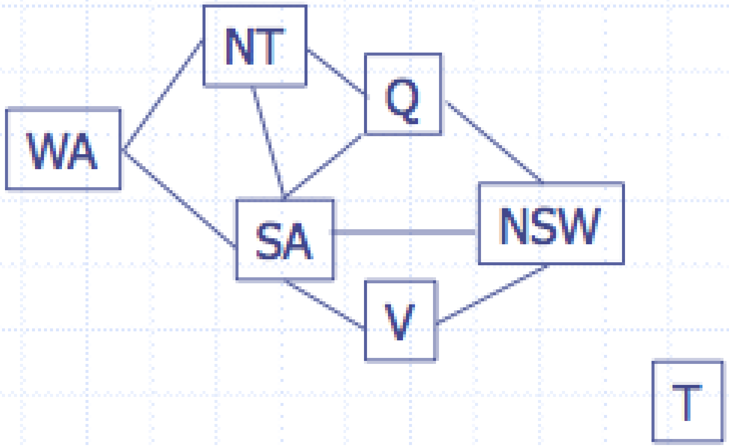


- 7 variables {WA,NT,SA,Q,NSW,V,T}
- Each variable has the same domain {red, green, blue}
- No two adjacent variables have the same value:

WA≠NT, WA≠SA, NT≠SA, NT≠Q, SA≠Q, SA≠NSW, SA≠V,Q≠NSW, NSW≠V

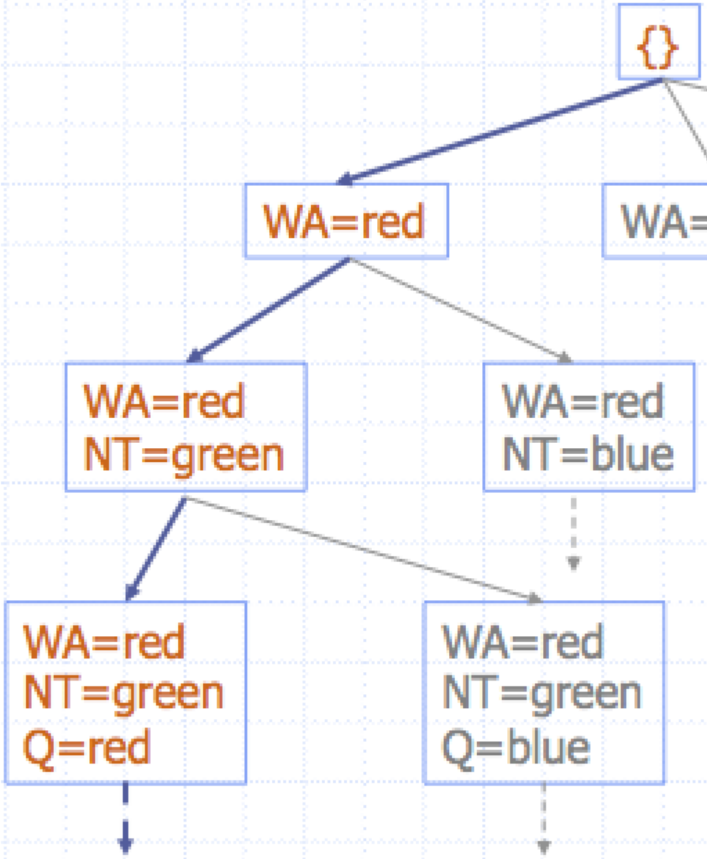
Constraint Graph

Binary constraints



Two variables are adjacent or neighbors if they are connected by an edge or an arc

Map Coloring



{}

WA=red

WA=green

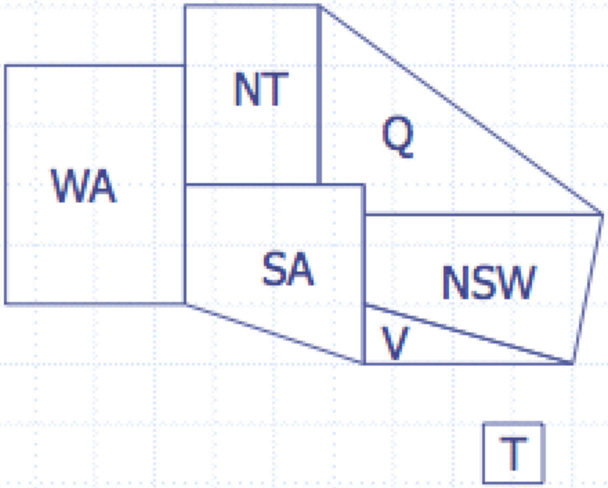
WA=blue

WA=red
NT=green

WA=red
NT=blue

WA=red
NT=green
Q=red

WA=red
NT=green
Q=blue



Backtracking Algorithm

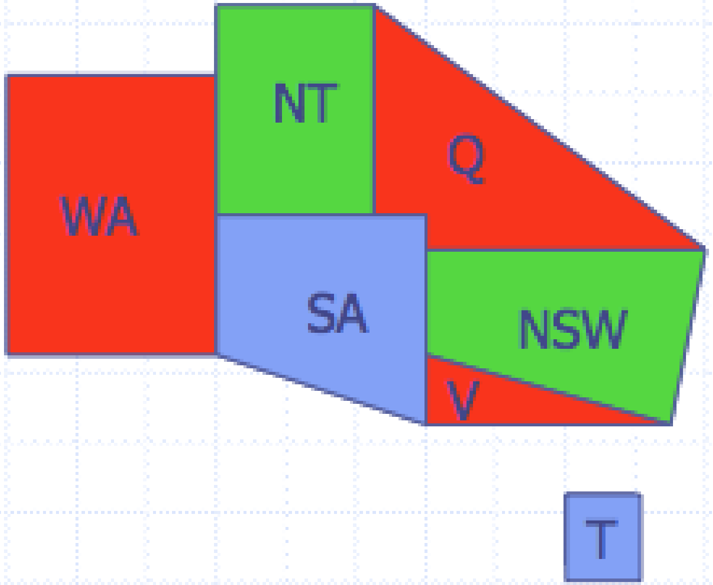
```
CSP-BACKTRACKING(PartialAssignment a)
    If a is complete then return a
    X ← select an unassigned variable
    D ← select an ordering for the domain of X
    For each value v in D do
        If v is consistent with a then
            Add (X= v) to a
            result ← CSP-BACKTRACKING(a)
            If result ≠ failure then return result
    Return failure

CSP-BACKTRACKING({})
```

Questions

1. Which variable X should be assigned a value next?
2. In which order should its domain D be sorted?
3. In which order should constraints be verified?

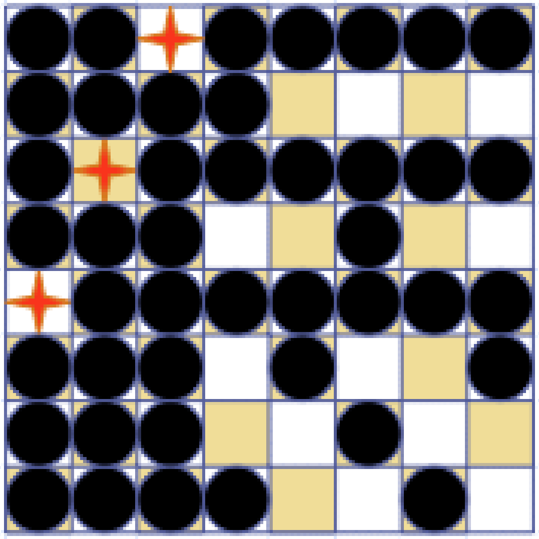
Choice of Variable

- Map coloring



Choice of Variable

- 8-queen



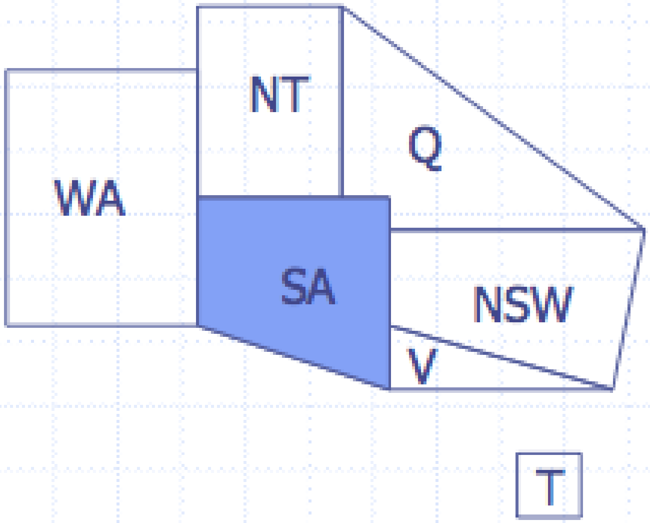
Choice of Variable

Most-constrained-variable heuristic:

Select a variable with the fewest remaining values

= Fail First Principle

Choice of Variable

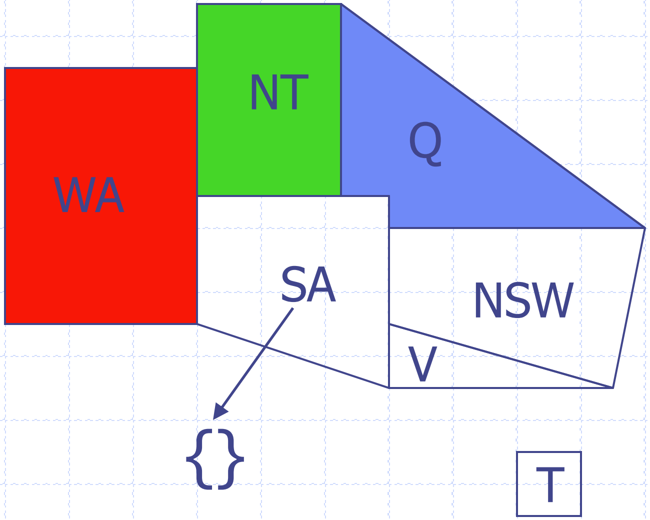


Most-constraining-variable heuristic:

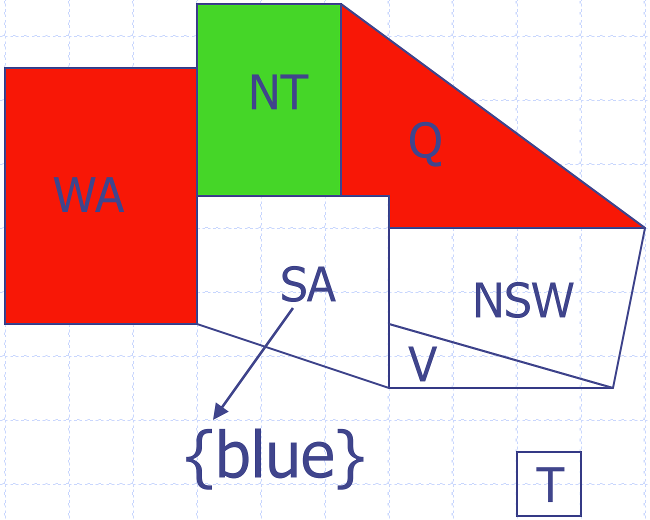
Select the variable that is involved in the largest number of constraints on other unassigned variables

= Fail First Principle again

Choice of Value



Choice of Value



Least-constraining-value heuristic:

Prefer the value that leaves the largest subset of legal values for other unassigned variables

Choice of Constraint to Test

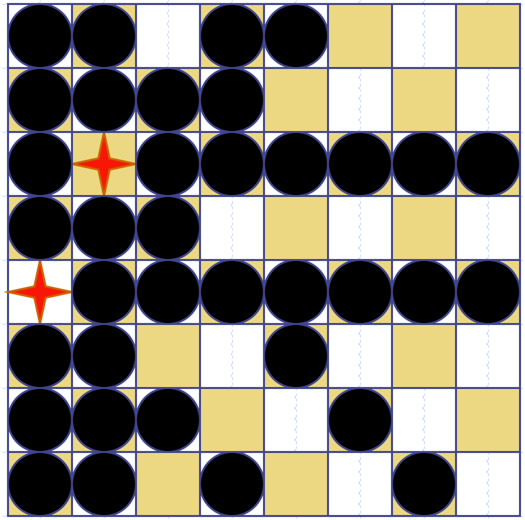
Most-constraining-Constraint:

Prefer testing constraints that are more difficult to satisfy

= Fail First Principle

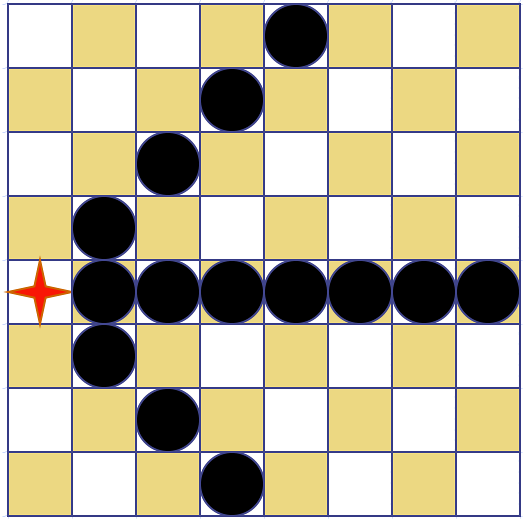
Constraint Propagation …

… is the process of determining how the possible values of one variable affect the possible values of other variables

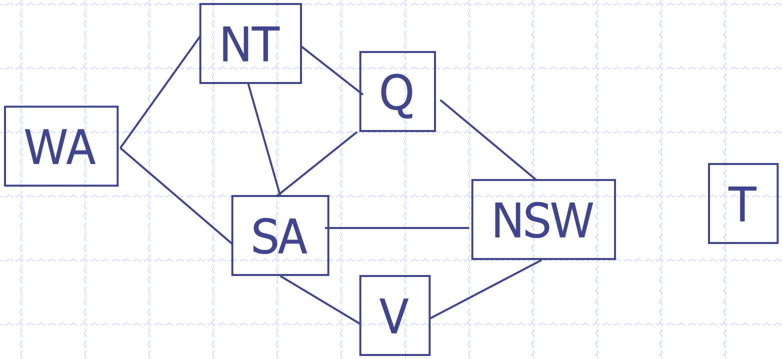


Forward Checking

After a variable X is assigned a value v, look at each unassigned variable Y that is connected to X by a constraint and deletes from Y’s domain any value that is inconsistent with v

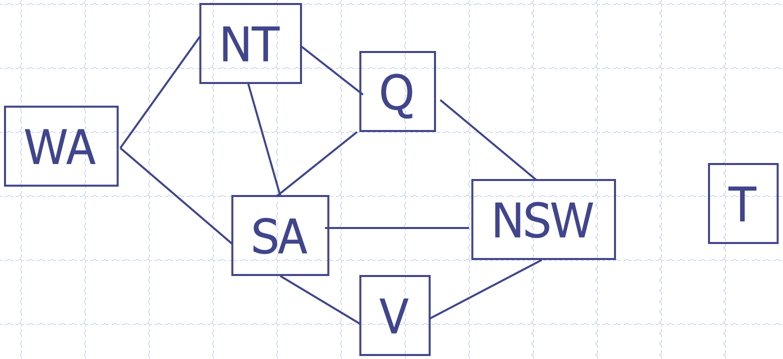


Map Coloring



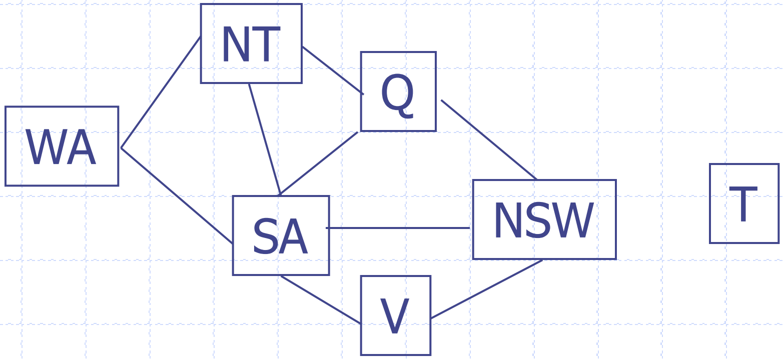
WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB

Map Coloring



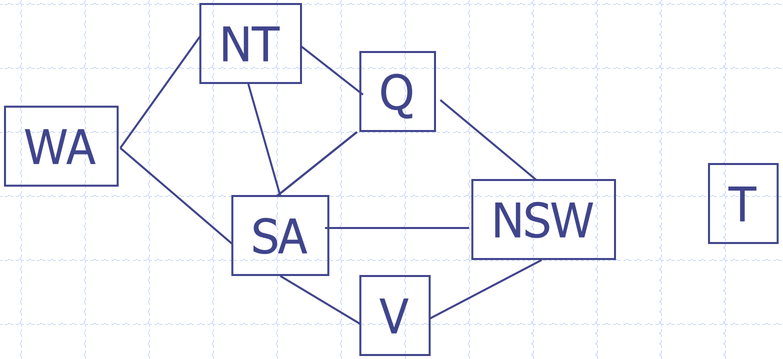
WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	RGB	RGB	RGB	GB	RGB

Map Coloring



WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	RGB	RGB	RGB	GB	RGB
R	B	G	RB	RGB	B	RGB

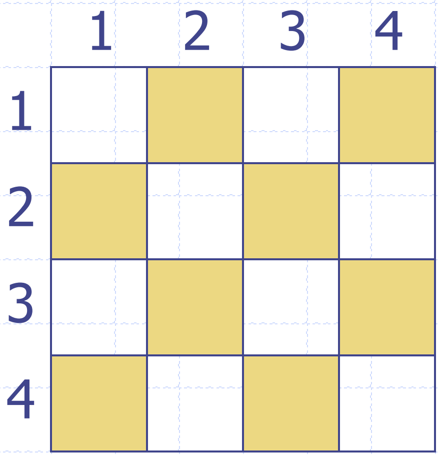
Map Coloring

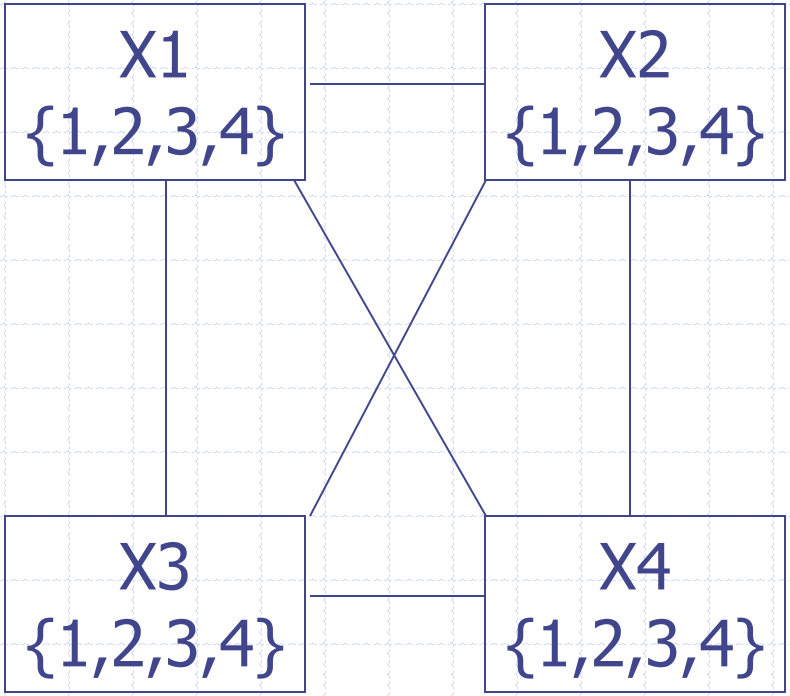


Impossible assignments that forward checking do not detect

WA	NT	Q				
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	RGB	RGB	RGB	GB	RGB
R	B	G	RB	RGB	B	RGB
R	B	G	R	B		RGB

Example: 4-Queens Problem





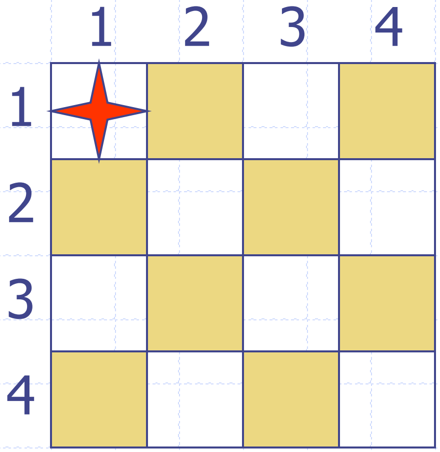
X1
{1,2,3,4}

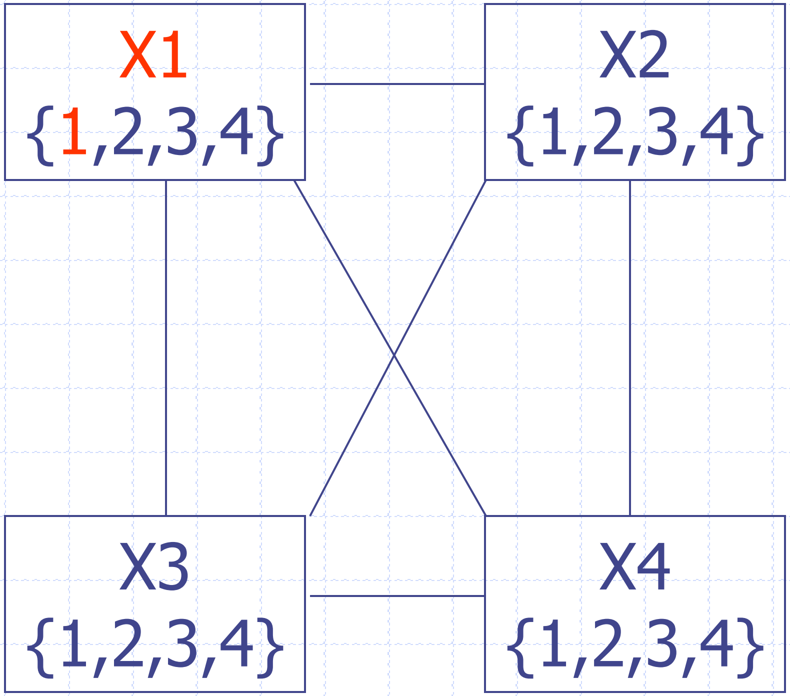
X2
{1,2,3,4}

X3
{1,2,3,4}

X4
{1,2,3,4}

Example: 4-Queens Problem





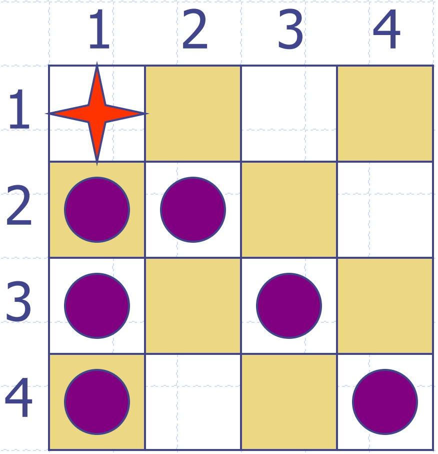
X1
{1,2,3,4}

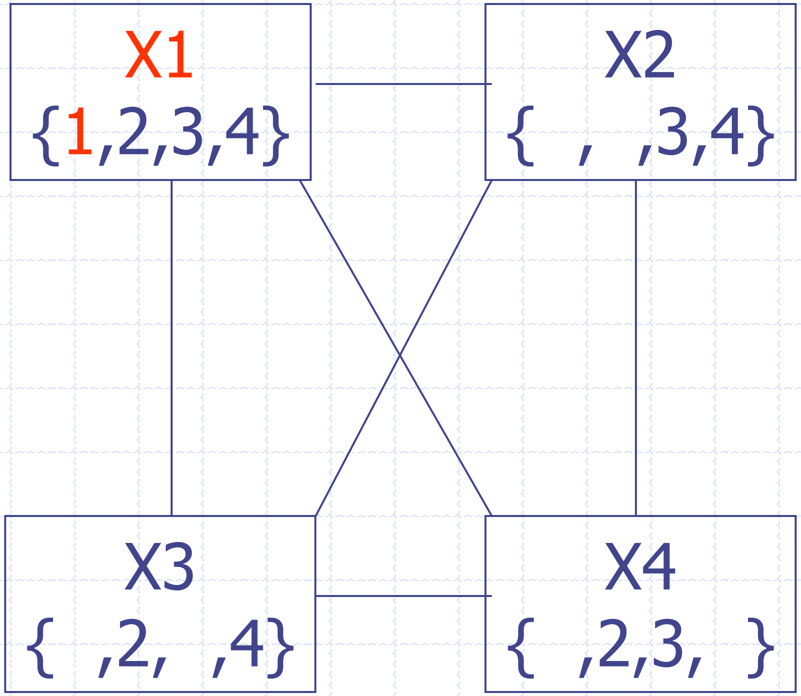
X2
{1,2,3,4}

X3
{1,2,3,4}

X4
{1,2,3,4}

Example: 4-Queens Problem





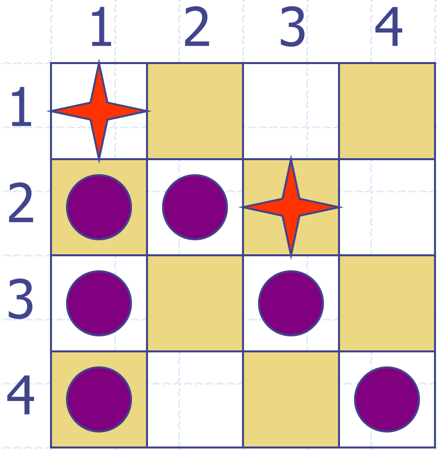
X1 {1,2,3,4}

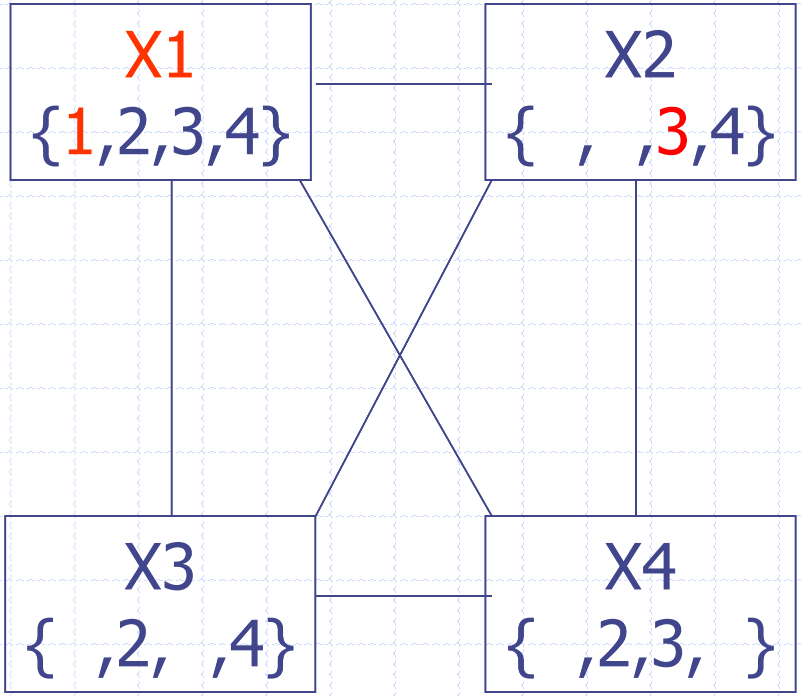
X2 { , ,3,4}

X3 { ,2, ,4}

X4 { ,2,3, }

Example: 4-Queens Problem





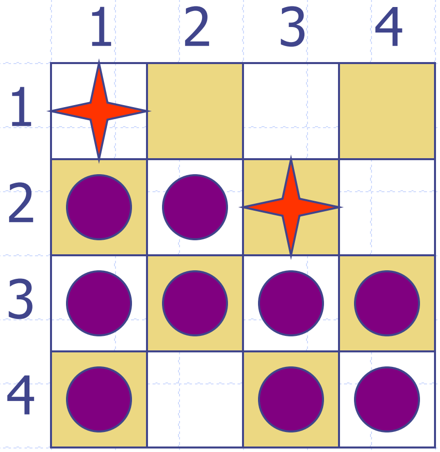
X1
{1,2,3,4}

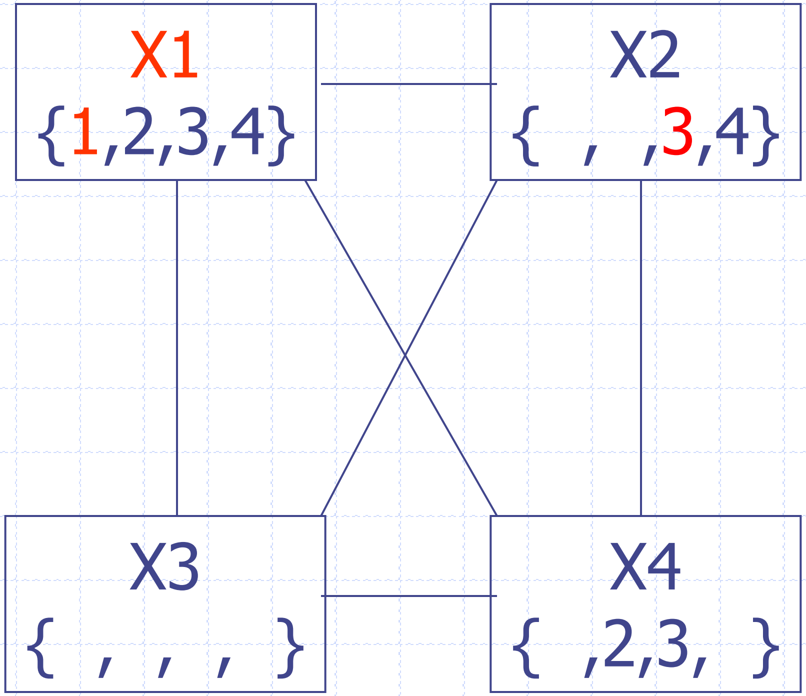
X2
{ , ,3,4}

X3
{ ,2, ,4}

X4
{ ,2,3, }

Example: 4-Queens Problem





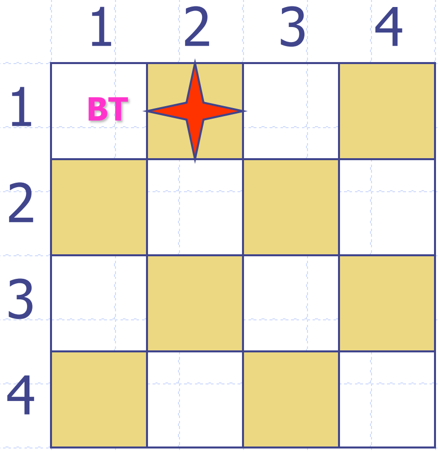
X1
{1,2,3,4}

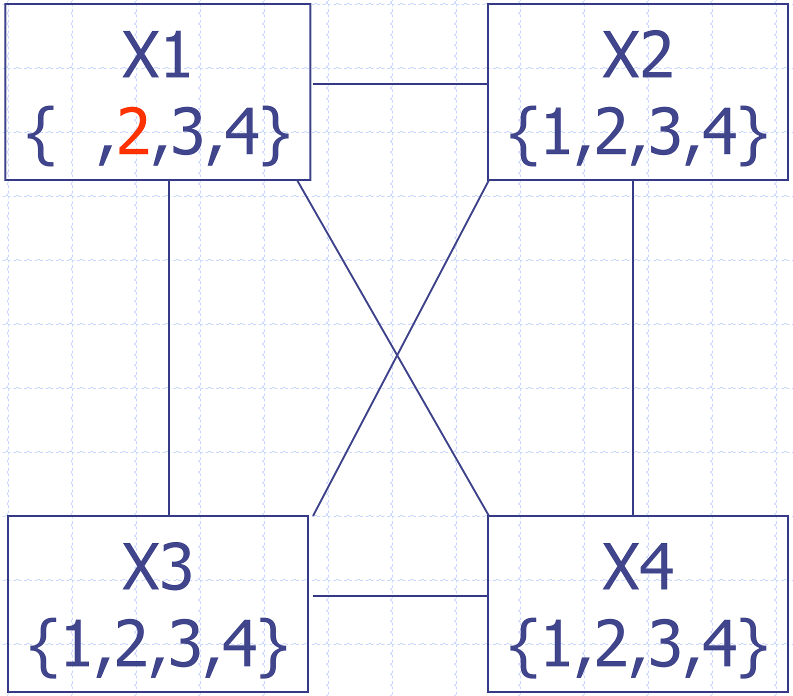
X2
{ , ,3,4}

X3
{ , , , }

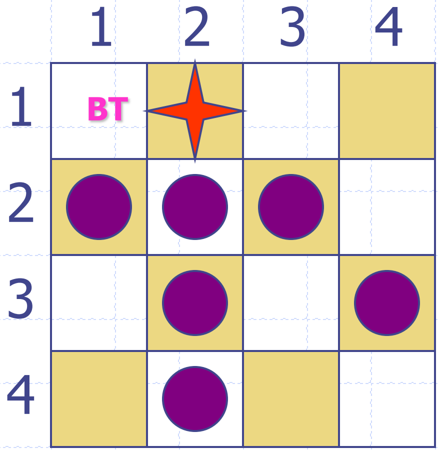
X4
{ ,2,3, }

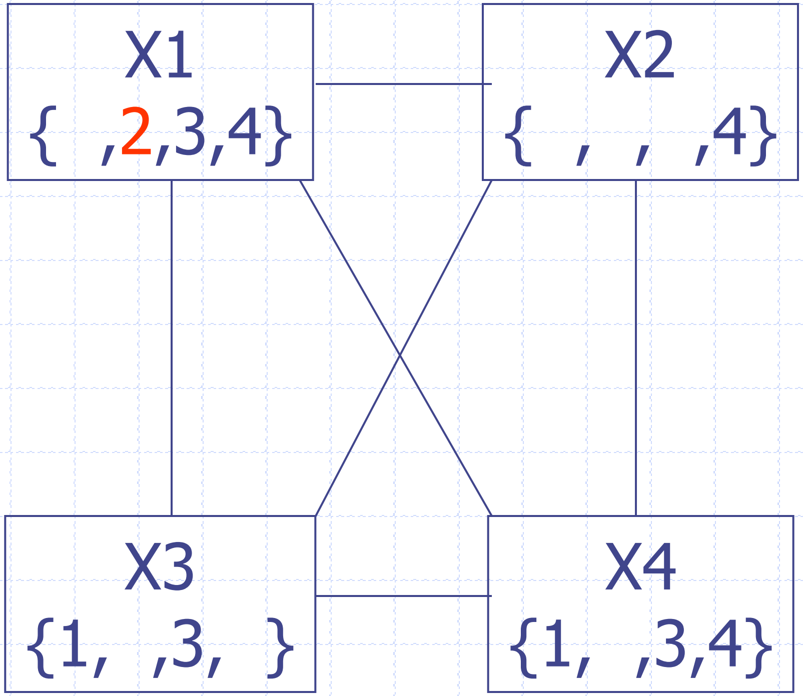
Example: 4-Queens Problem





Example: 4-Queens Problem





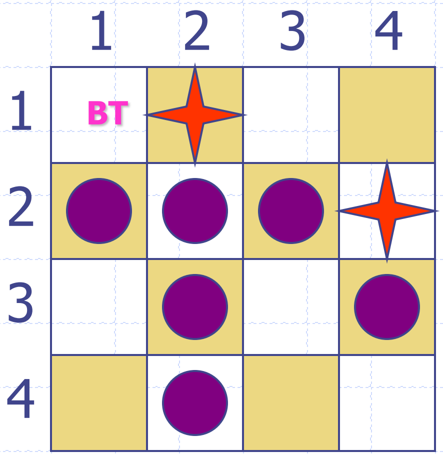
X1
{ ,2,3,4}

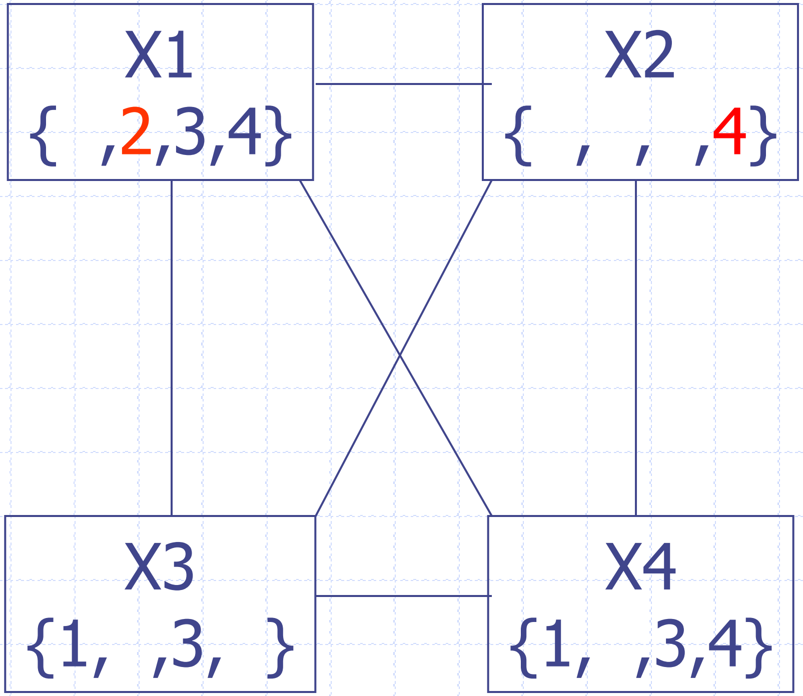
X2
{ , , ,4}

X3
{1, ,3, }

X4
{1, ,3,4}

Example: 4-Queens Problem





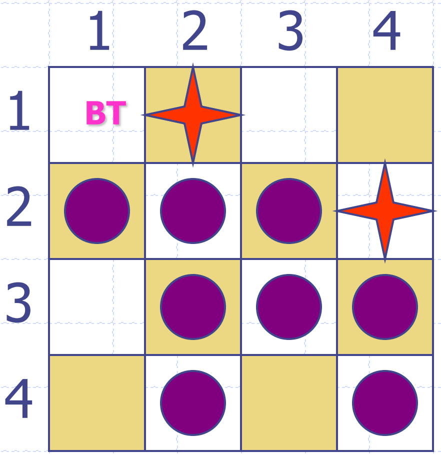
X1
{ ,2,3,4}

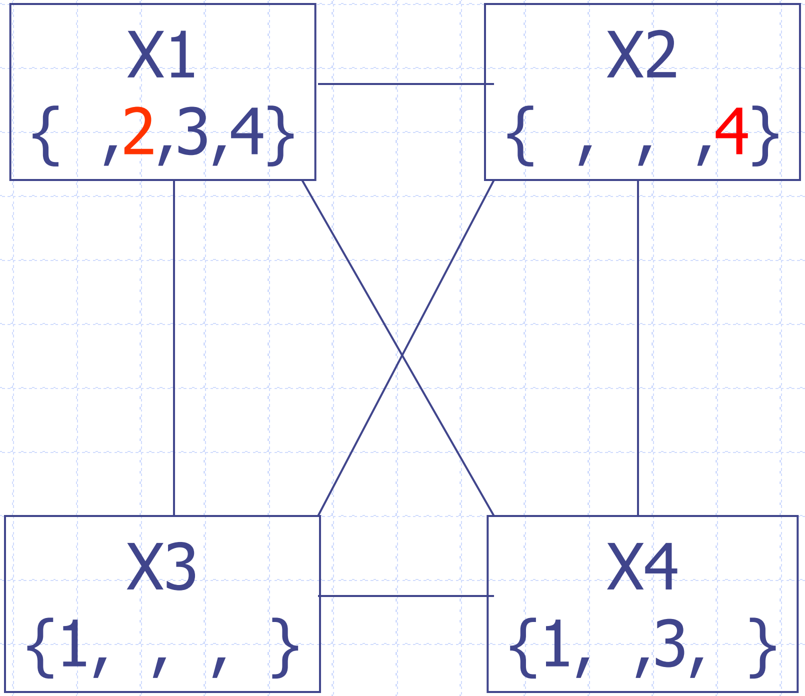
X2
{ , , ,4}

X3
{1, ,3, }

X4
{1, ,3,4}

Example: 4-Queens Problem





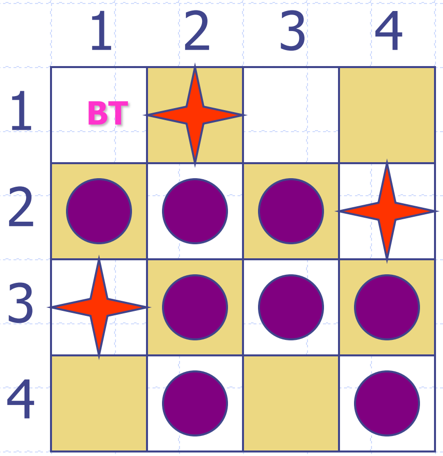
X1
{ ,2,3,4}

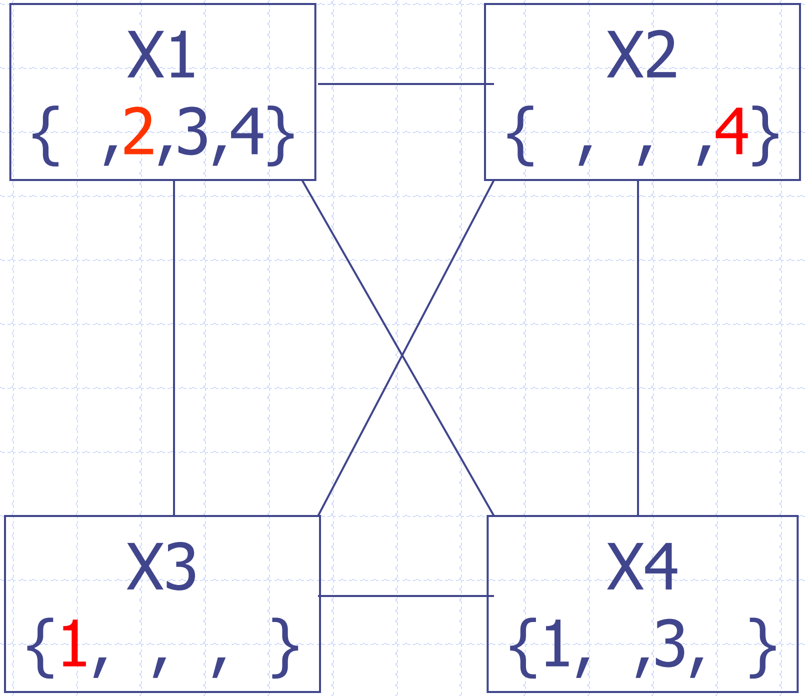
X2
{ , , ,4}

X3
{1, , , }

X4
{1, ,3, }

Example: 4-Queens Problem





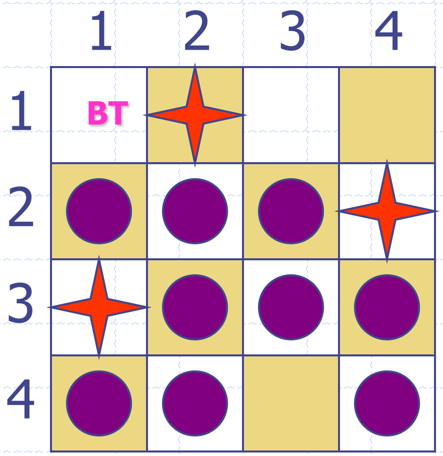
X1
{ ,2,3,4}

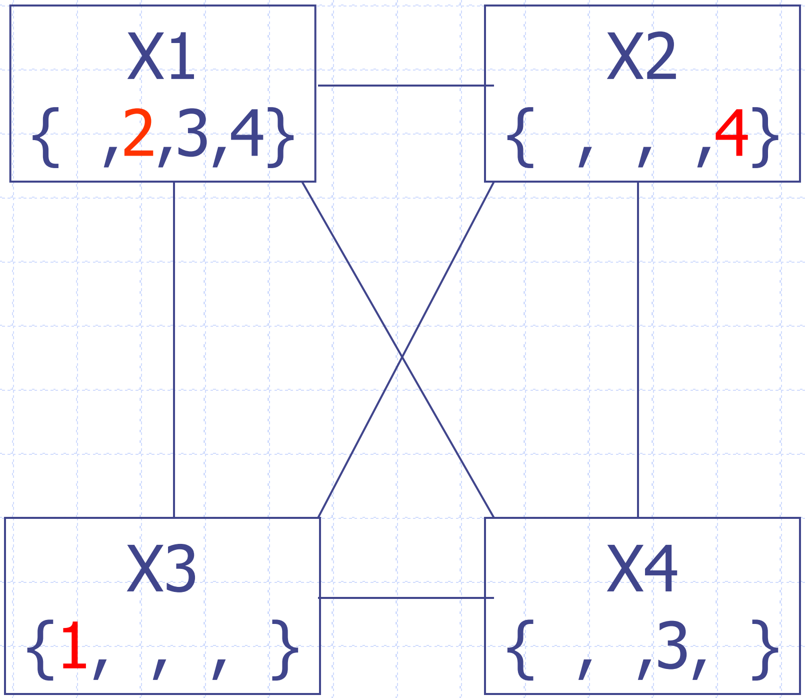
X2
{ , , ,4}

X3
{1, , , }

X4
{1, ,3, }

Example: 4-Queens Problem





X1
{ ,2,3,4}

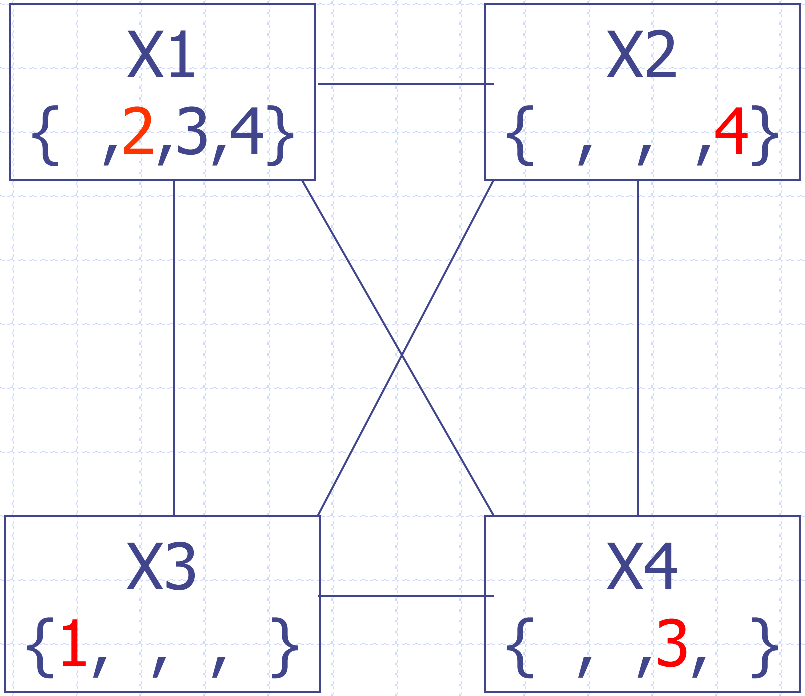
X2
{ , , ,4}

X3
{1, , , }

X4
{ , ,3, }

Example: 4-Queens Problem

	1	2	3	4
1	BT	✦		
2	●	●	●	✦
3	✦	●	●	●
4	●	●	✦	●


X1
{ ,2,3,4}
X2
{ , , ,4}
X3
{1, , , }
X4
{ , ,3, }