What is it about?

- Theory
  - The core of solving constraint problems using Constraint Programming (CP), with emphasis on:
    - Modeling
    - Solving:
      - Local consistency and propagation;
      - Backtracking search + heuristics.
  - Brief overview of advanced topics.
- Practice
  - Programming examples.
  - Project work.

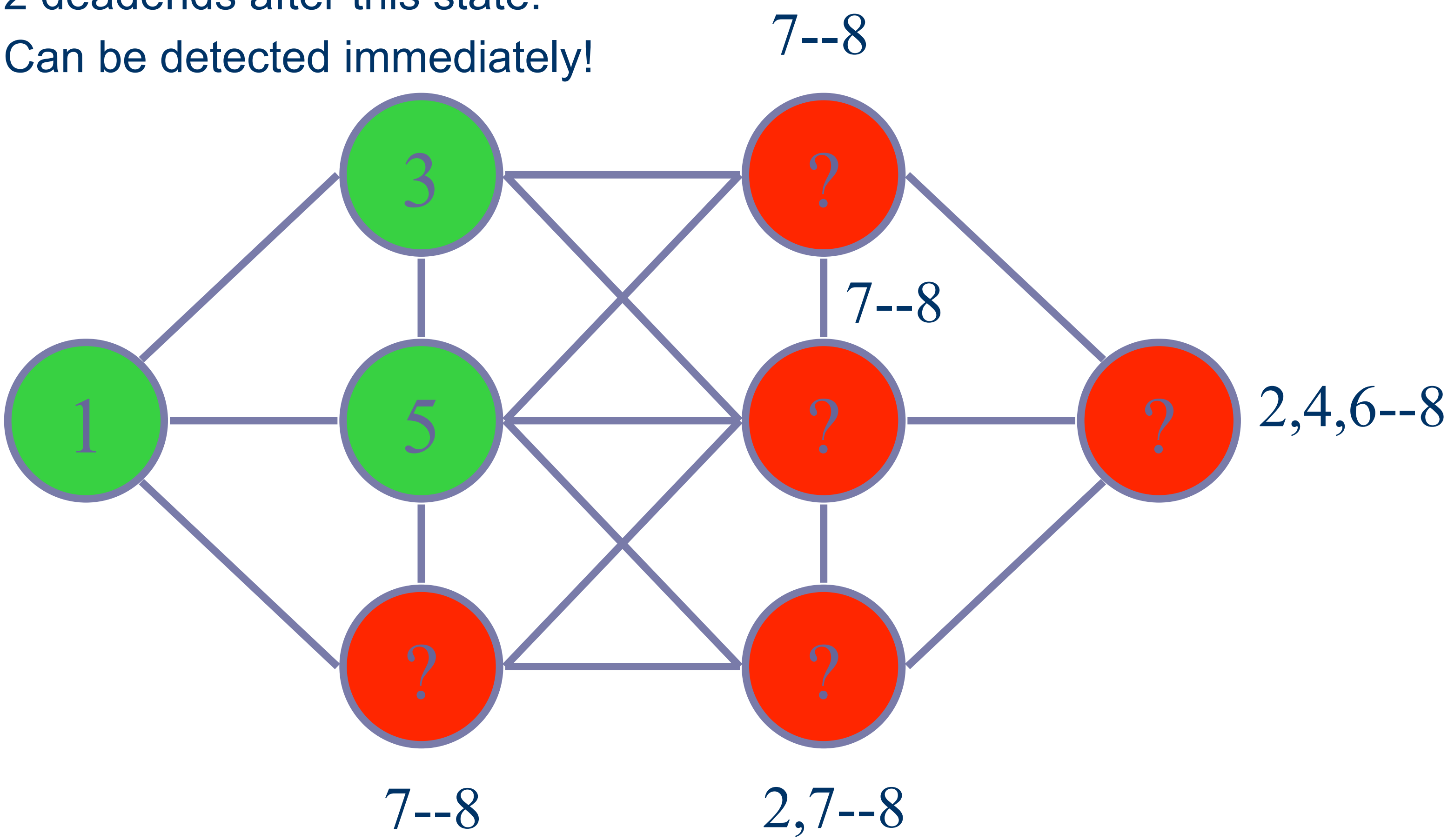
Efficiency

- Bad choice of nodes, bad assignment of values

  => Good heuristic choice is very important!
- Good heuristics are always possible?
  - No!
- What can we do then?
  - Good consistency and propagation!
- Didn't we do that already?
  - Not as strong as we could!

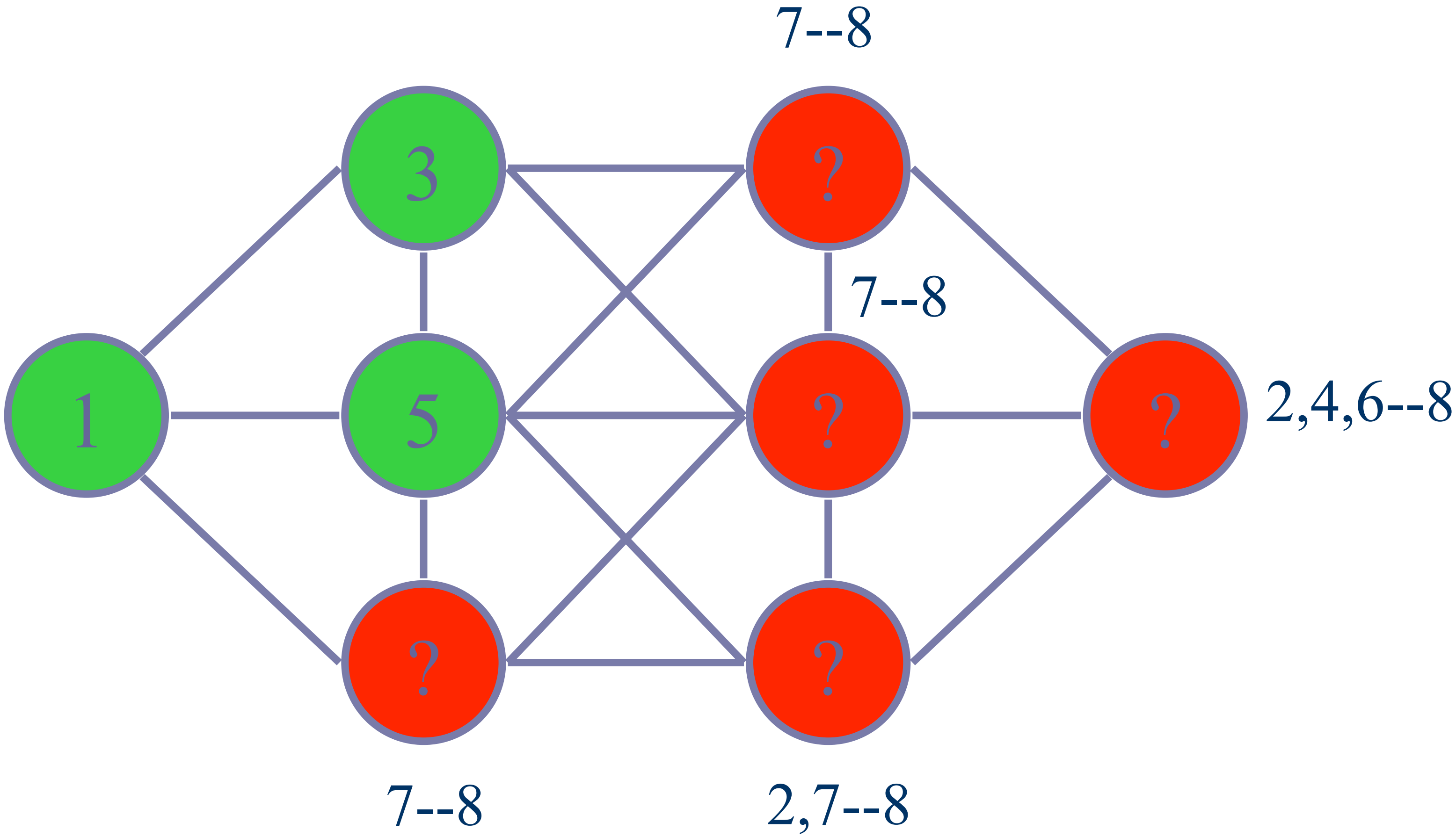
A State During Search

- 2 deadends after this state.
- Can be detected immediately!



A State During Search

- Check the relations between the future nodes.



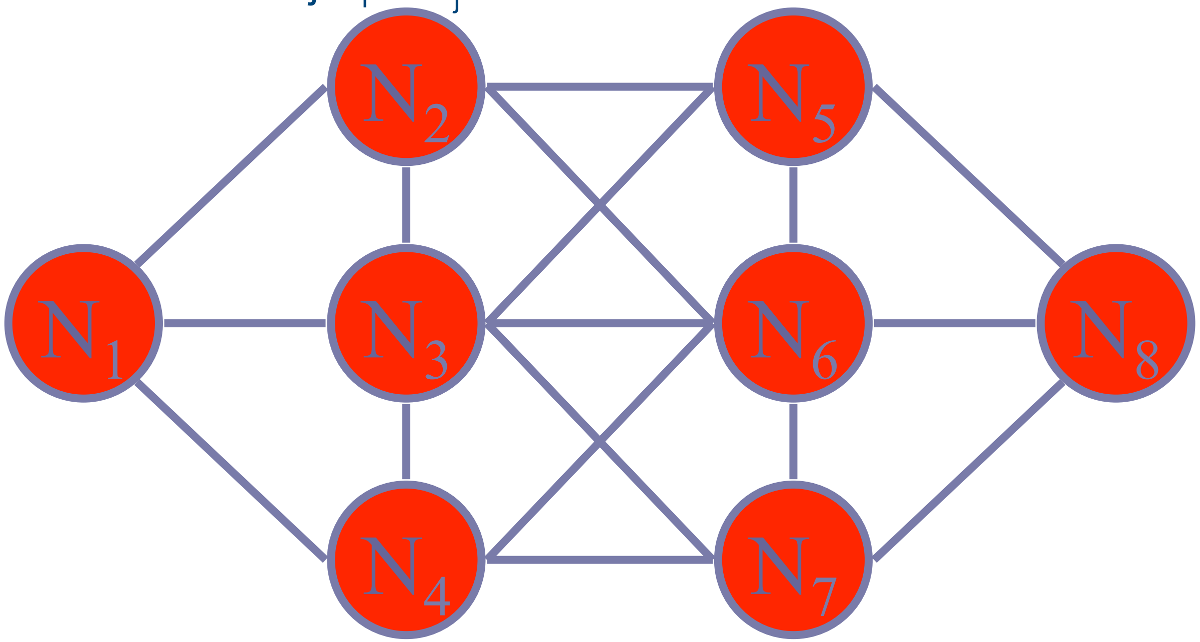
Efficiency

- Bad choice of nodes, bad assignment of values

  => Good heuristic choice is very important!
- Good heuristics are always possible?
  - No!
- What can we do then?
  - Good consistency and propagation!
- Didn't we do that already?
  - Not as strong as we could!
- Is that all?
  - Nope!
  - Good modeling can result in even stronger consistency & propagation

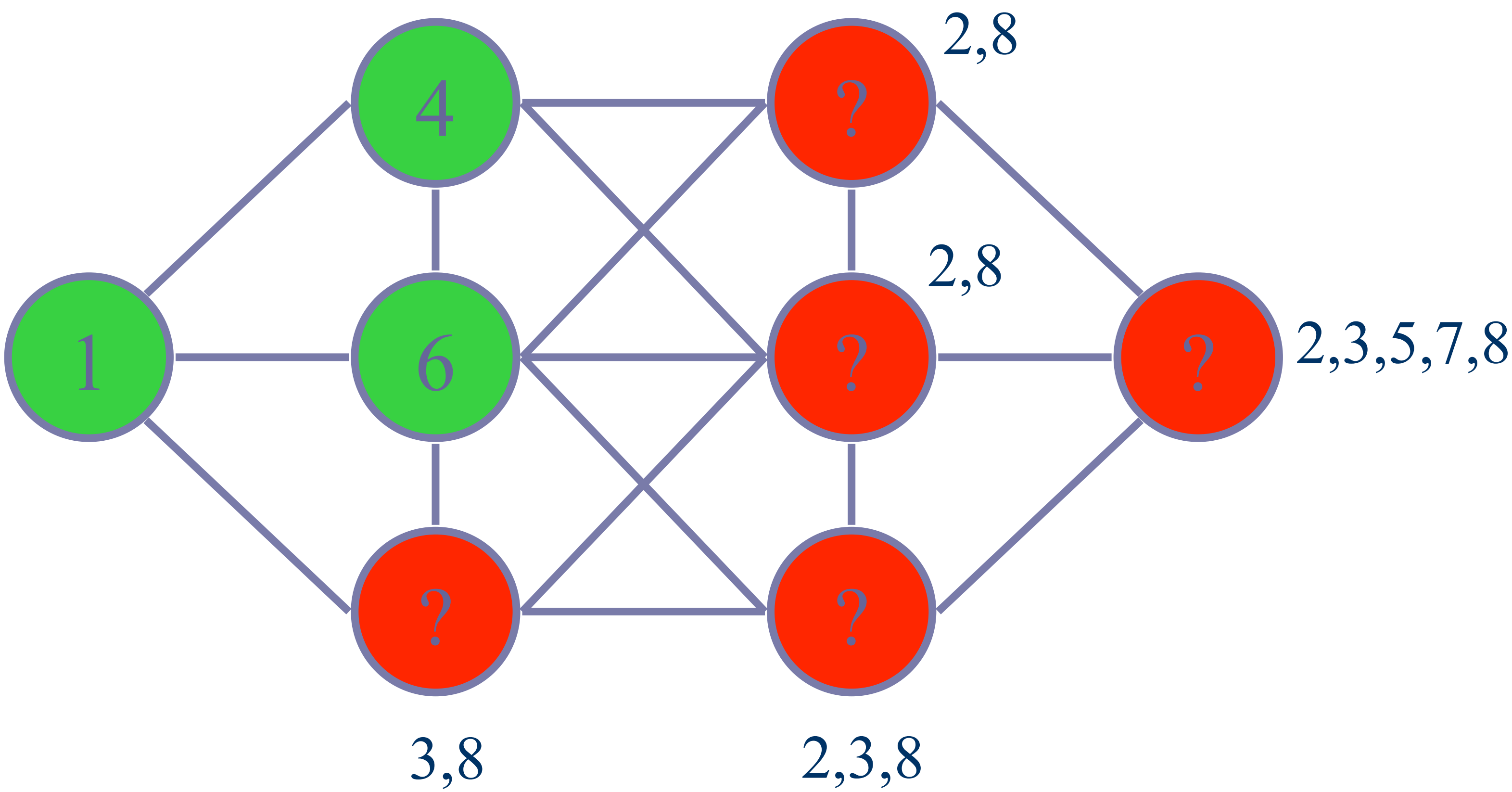
Modeling

- Unknowns: $N_1..N_8$ represent the nodes
- Values that $N_1..N_8$ can take: {1,2,3,4,5,6,7,8}
- Relations: for all i < j s.t. Ni and Nj are adjacent. $|N_i - N_j| > 1$
  for all i < j $N_i \neq N_j$



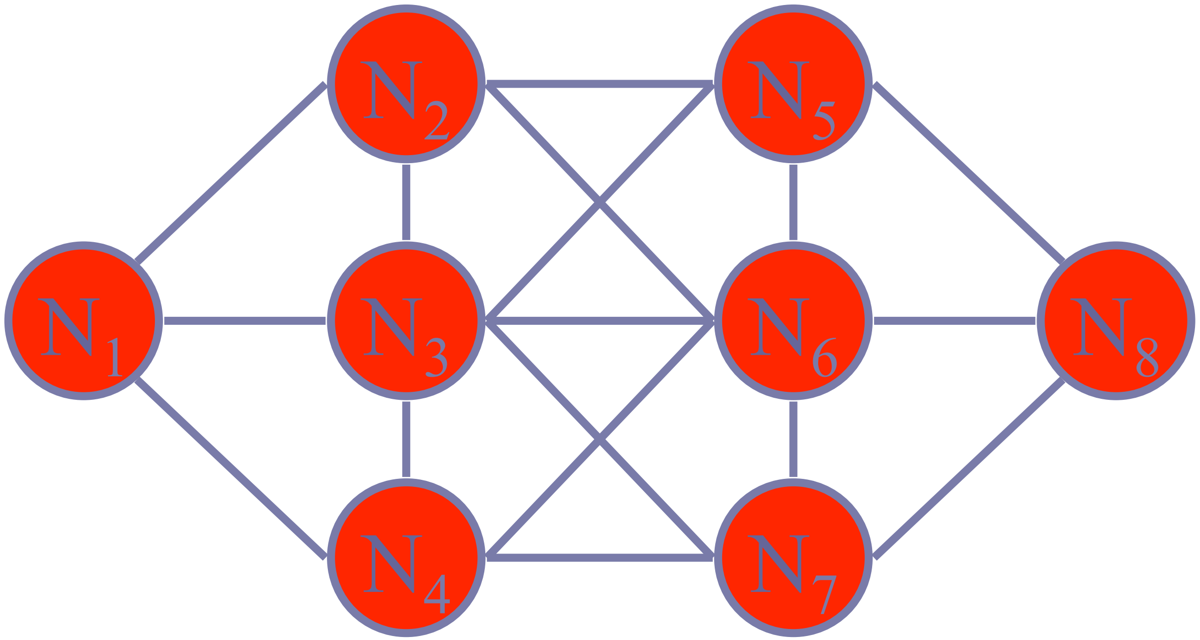
Another State

- Cannot detect the inconsistency of N_3= 6.
  - Future nodes are fine wrt the relations.



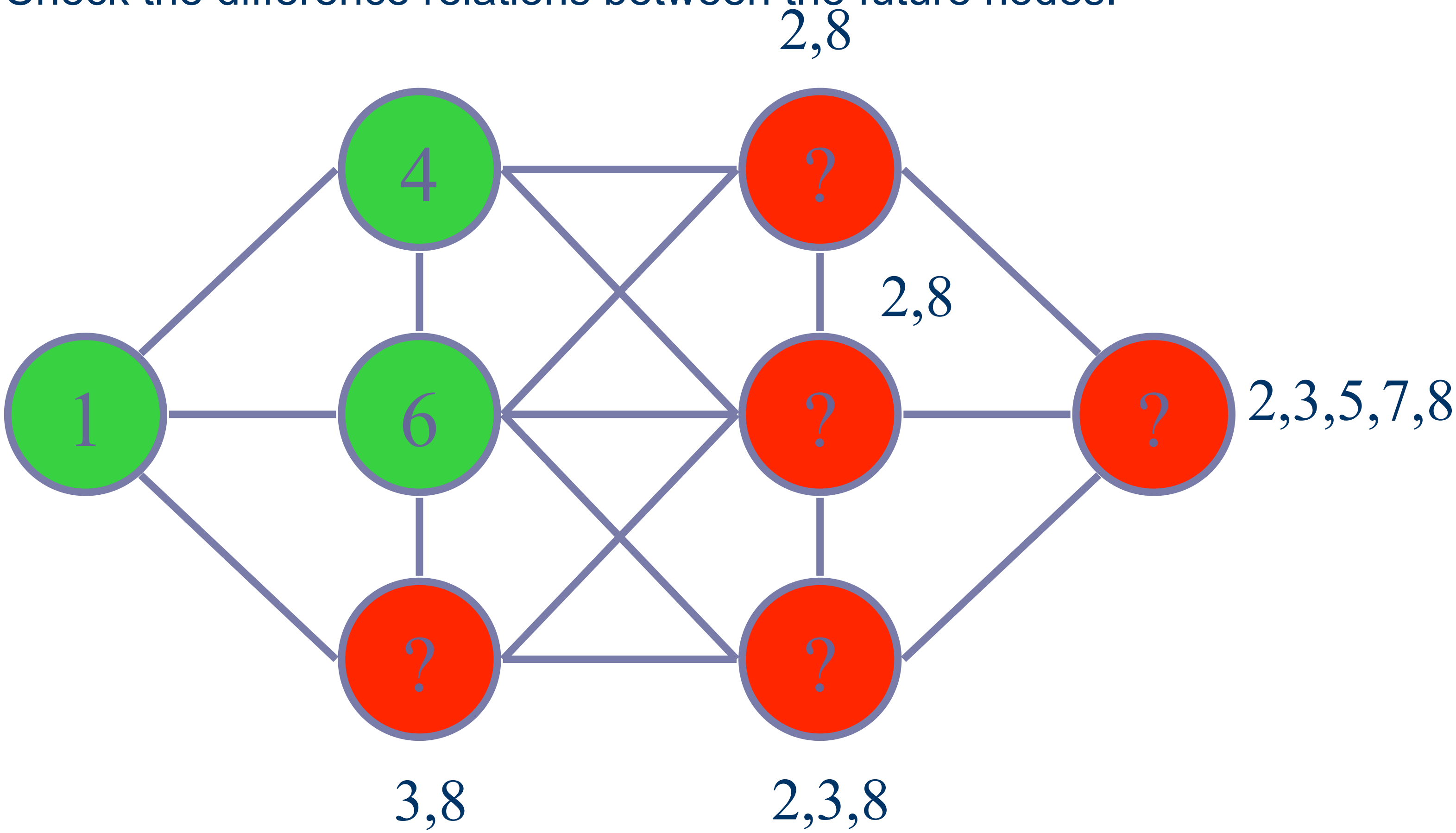
Stronger Model

- Relations:
  - for all $i < j$ s.t. Ni and Nj are adjacent $|N_i - N_j| > 1$
  - for all $i < j$ $N_i \neq N_j$ alldifferent($[N_1, N_2, N_3, N_4, N_5, N_6, N_7,]$)



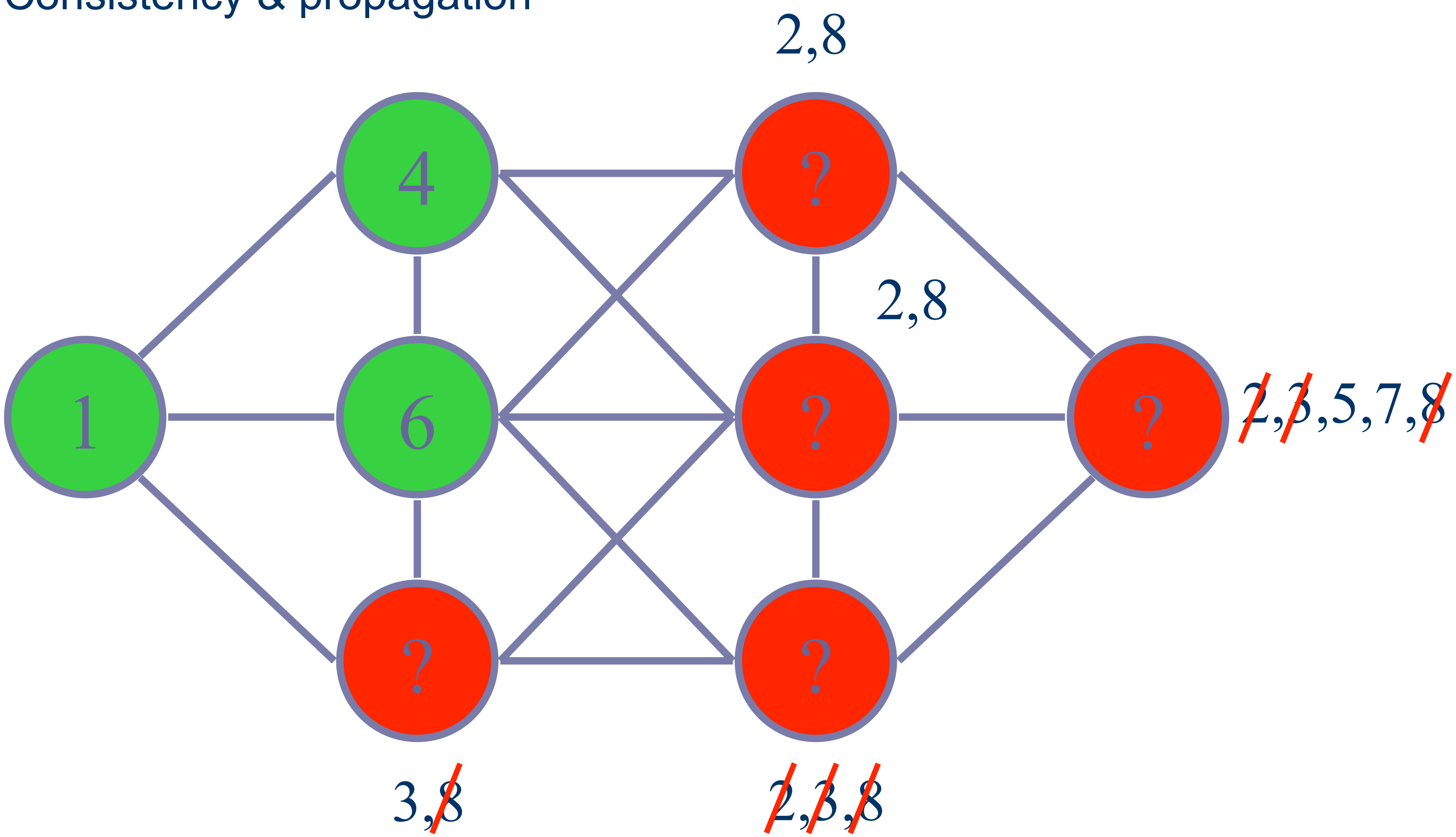
Another State

- Check the difference relations between the future nodes.



Another State

- Consistency & propagation



PART I: Modeling

Modeling

- Representation of a decision problem as a CSP.
- Formalize:
  - unknowns of the decision → variables
  - possible values for unknowns → domains
  - relations between the unknowns → constraints

CSPs: More formally

- A CSP is a triple **<X,D,C>** where:
  - **X** is a set of decision variables **$\{X_1,\ldots,X_n\}$**;
  - **D** is a set of domains **$\{D_1,\ldots,D_n\}$** for **X**:
    - D_i is a set of possible values for X_i;
    - usually assume finite domain.
  - **C** is a set of constraints **$\{C_1,\ldots,C_m\}$**:
    - C_i is a relation over $X_j,\ldots,X_k$, giving the set of combination of allowed values;
    - $C_i \subseteq D(X_j) \times \ldots \times D(X_k)$

- A solution to a CSP is an assignment of values to the variables which satisfies all the constraints simultaneously.

Constraint Optimization Problems

- CSP enhanced with an optimization criterion, e.g.:
  - minimum cost;
  - shortest distance;
  - fastest route;
  - maximum profit.
- Formalization of the optimization criterion as an objective function.

CSPs: A simple example

- Variables

  $X = \{X_1, X_2, X_3\}$

- Domains

  $D(X_1) = \{1,2\}$, $D(X_2) = \{0,1,2,3\}$, $D(X_3) = \{2,3\}$

- Constraints

  $X_1 > X_2$ and $X_1 + X_2 = X_3$ and $X_1 \neq X_2 \neq X_3 \neq X_1$

  ↓ alldifferent($[X_1, X_2, X_3]$)

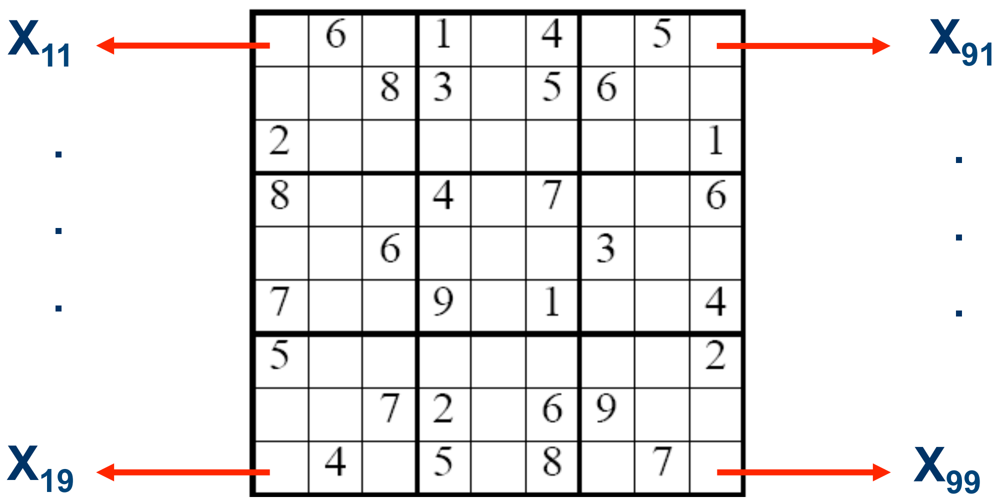
- Solution

  $X_1 = 2$, $X_2 = 1$, $X_3 = 3$

Sudoku

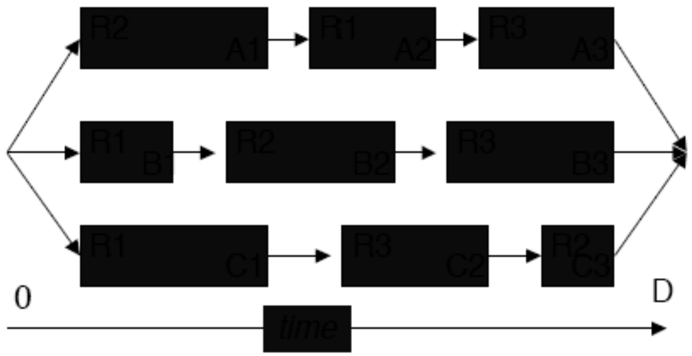
	6		1		4		5	
		8	3		5	6		
2								1
8			4		7			6
		6				3		
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	

Sudoku



- A simple CSP
  - 9x9 variables (X_{ij}) for each cell with domains {1,...., 9}
  - Not-equals constraints on the rows, columns, and 3x3 boxes. E.g.,
    alldifferent([X_{11}, X_{21}, X_{31}, …, X_{91}])
    alldifferent([X_{11}, X_{12}, X_{13}, …, X_{19}])
    alldifferent([X_{11}, X_{21}, X_{31}, X_{12}, X_{22}, X_{32}, X_{13}, X_{23}, X_{33}])

Task Scheduling



- Schedule tasks on a machine, in time D, by obeying the temporal and precedence constraints:
  - each task i has a specific fixed processing time p_i;
  - each task can be started after its release date r_i, and must be completed before its deadline d_i;
  - tasks cannot overlap in time;
  - precedence relations must be respected.

Task Scheduling

- Variables and Domains
  - $Start_i$, representing the starting time of each task i, taking a value from {1,2,…, D}.
  - This immediately ensures that each task starts at exactly one time point.
- Constraints
  - Respect release date and deadline:
    - for all i $r_i \leq Start_i \leq d_i - p_i$
  - Tasks cannot overlap in time:
    - for all i < j $(Start_i + p_i < Start_j)$ OR $(Start_j + p_j < Start_i)$
  - Precedences:
    - $Start_i < Start_j$ for every pair of task i with precedence over task i

Variables and Domains

- Variable domains include the classical:
  - binary, integer, continuous.
- In addition, variables may take a value from *any* finite set.
  - e.g., X in {a,b,c,d,e}.
- There exist special “structured” variable types.
  - Set variables (take a set of elements as value).
  - Activities or interval variables (for scheduling applications).

Properties of Constraints

- Declarative (invariant) relations among objects.
  - X > Y
- The order of imposition does not matter.
  - X + Y <= Z, X + Y >=Z
- Non-directional.
  - A constraint between X and Y can be used to infer information on Y given information on X and vice versa.
- Rarely independent.
  - Shared variables as communication mechanism.
- Incremental operational behavior.
  - When new information available, the computation does not necessarily start from scratch.

Constraints – Examples

- Algebraic expressions.
  - $X_1 > X_2$
  - $X_1 + X_2 = X_3$
  - $X^3(Y^2 - Z) \geq 25 + X^2 \cdot \max(X, Y, Z)$
- Extensional constraints ('table' constraints).
  - (X,Y,Z) in {(a, a, a), (b, b, b), (c, c, c)}
- Variables as subscripts ('element' constraints).
  - Y = cost[X] (here Y and X are variables, 'cost' is an array of parameters)

Constraints – Examples

- Reasoning with meta-constraints.
  - $\sum_i (X_i > t_i) \leq 5$
- Logical relations in which (meta-)constraints can be mixed.
  - $((X < Y) \text{ OR } (Y < Z)) \Rightarrow (C = \min(X,Y))$
- Global constraints.
  - alldifferent($[X_1, X_2, X_3]$) instead of $X_1 \neq X_2$, $X_1 \neq X_3$, $X_2 \neq X_3$

Modeling is Critical!

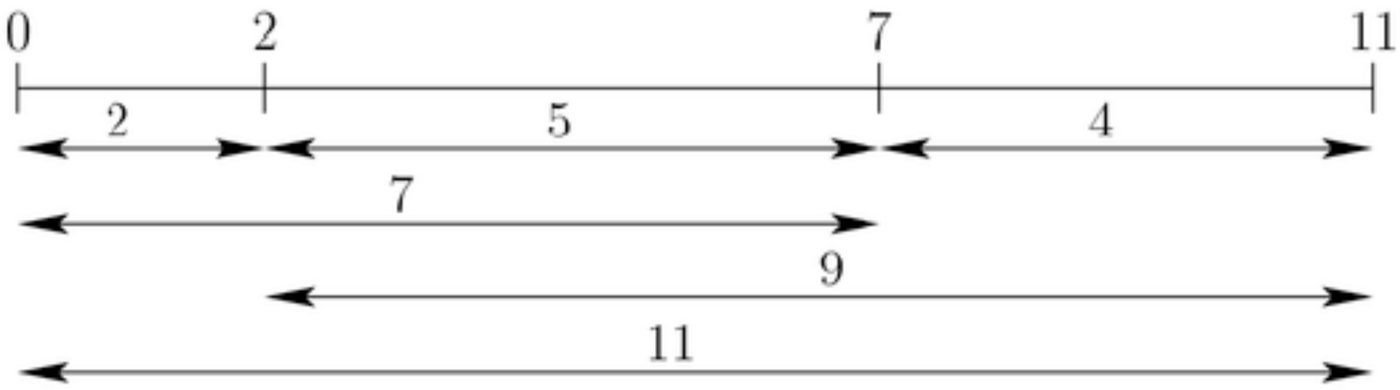
- Choice of variables defines search space.
  - $D(X_1) \times D(X_2) \times \ldots \times D(X_n)$
- Choice of constraints defines:
  - how domains search space can be reduced;
  - how search can be guided.
- Need to go beyond the declarative specification!

Modeling is Critical

- Given the human understanding of a problem, we need to answer questions like:
  - which variables shall I choose?
  - which constraints shall I enforce?
  - can I exploit any global constraints?
  - so I need any auxiliary variables?
  - are some constraints redundant, therefore can be avoided?
  - are there any implied constraints?
  - can symmetry be eliminated?
  - are there any dual viewpoints?
  - among alternative models, which one shall I prefer?

Golomb Ruler

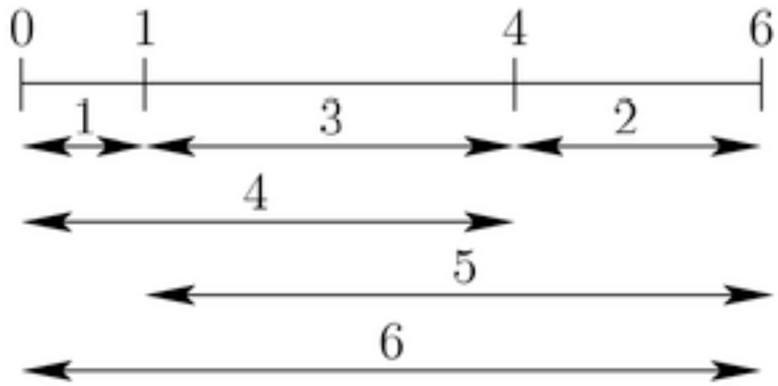
- Place m marks on a ruler such that:
  - distance between each pair of marks is different;
  - the length of the ruler is minimum.
- Applications in radio astronomy and information theory.
- Difficult to solve! Largest known ruler is of order 27.



A non optimal Golomb ruler of order 4.

Golomb Ruler

- Place m marks on a ruler such that:
  - distance between each pair of marks is different;
  - the length of the ruler is minimum.
- Applications in radio astronomy and information theory.
- Difficult to solve! Largest known ruler is of order 27.



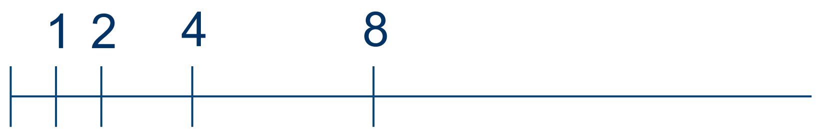
An optimal Golomb ruler of order 4.

Naive Model

- Variables and Domains
  - $[X_1, X_2, .., X_m]$
  - X_i, representing the position of the i^{th} mark, taking a value from $\{0,1,\ldots,2^{(m-1)}\}$

Naive Model

- Variables and Domains
  - $[X_1, X_2, .., X_m]$
  - X_i, representing the position of the i^{th} mark, taking a value from $\{0,1,\ldots,2^{(m-1)}\}$



Naive Model

- Variables and Domains
  - $[X_1, X_2, .., X_m]$
  - X_i, representing the position of the i^{th} mark, taking a value from $\{0,1,\ldots,2^{(m-1)}\}$
- Constraints
  - for all $i_1<j_1$, $i_2<j_2$, $i_1 \neq i_2$ or $j_1 \neq j_2$ $|X_{i1}-X_{j1}| \neq |X_{i2}-X_{j2}|$
- Objective: minimize $(\max([X_1, X_2, .., X_m]))$

Naive Model

- Variables and Domains
  - $[X_1, X_2, .., X_m]$
  - X_i, representing the position of the i^{th} mark, taking a value from $\{0,1,\ldots,2^{(m-1)}\}$
- Constraints
  - for all $i_1<j_1$, $i_2<j_2$, $i_1 \neq i_2$ or $j_1 \neq j_2$ $|X_{i1}-X_{j1}| \neq |X_{i2}-X_{j2}|$
- Objective: minimize $(\max([X_1, X_2, .., X_m]))$
- Problematic model.
  - $O(n^4)$ quaternary constraints.
  - Loose reduction in domains.

Better Model

- Auxiliary Variables
  - Variables introduced into a model, because either:
    - it is difficult/impossible to express some constraints on the main variables;
    - or some constraints on the main variables do not lead to significant domain reductions.
  - for all $i<j$ D_{ij}, representing the distance between i^{th} and the j^{th} marks.
- Constraints
  - for all $i<j$ $D_{ij} = |X_i - X_j|$
  - alldifferent([D_{12}, D_{13}, …, $D_{(m-1)m}$])

Better Model

- Constraints
  - for all i<j $D_{ij} = |X_i - X_j|$
  - alldifferent([D_{12}, D_{13}, …, $D_{(m-1)m}$])
  - alldifferent([X_1, X_2, …, X_m])
- Improvements:
  - $O(n^2)$ ternary constraints.
  - Implied constraint.
    - Propagation is in general incomplete: inconsistent values are left in the domains.
    - Constraint implied by the constraints defining the problem which cannot be deduced by the incomplete solver.
    - No change in set of solutions, great reductions in search space!
  - Tighter constraints and denser constraint graph.

Improved Model

- Symmetry
  - Positions can be permuted.
    - $X_1 = 0, X_2 = 1, X_3 = 4, X_4 = 6$
    - $X_1 = 0, X_2 = 1, X_3 = 6, X_4 = 4$
    - $X_1 = 0, X_2 = 4, X_3 = 1, X_4 = 6$
    - $X_1 = 0, X_2 = 4, X_3 = 6, X_4 = 1$
    - $X_1 = 0, X_2 = 6, X_3 = 1, X_4 = 4$
    - $X_1 = 0, X_2 = 6, X_3 = 4, X_4 = 1$
    - …
    - n! permutations

Improved Model

- Symmetry
  - Positions can be permuted.
  - A ruler can be reversed.

  - $X_1 = 0, X_2 = 1, X_3 = 4, X_2 = 6$
  - $X_1 = 0, X_2 = 2, X_3 = 5, X_2 = 6$

Improved Model

- Symmetry
  - Positions can be permuted.
  - A ruler can be reversed.

Many symmetrically equivalent search states!

Improved Model

- Symmetry
  - Positions can be permuted.
  - A ruler can be reversed.

  Many symmetrically equivalent search states!
  - Symmetry breaking constraints.
    - Not implied by the constraints defining the problem.
    - Reduce the set of solutions and search space!
- Symmetry Breaking Constraints
  - $X_1 < X_2 < \ldots < X_m$
  - $X_1 = 0$
  - $D_{12} < D_{(m-1)m}$
- New objective
  - minimize (X_m)

Benefits of Remodeling

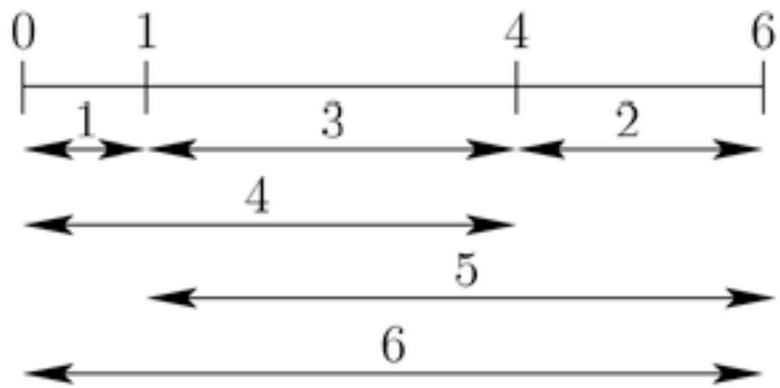
- The m marks must all be different in 1..d.
  - Search space = number of m-permutations: d! / (d-m)!
- But they can also be totally ordered.
  - Search space = number of m-combinations: d! /m!(d-m)!
  - m! times smaller!
- Instead of 0 being any mark on the ruler we can constrain it to be at position 1.
  - Reduction of d.
- Then we just have to minimise the position of the last mark.

Improved Model

- Symmetry Breaking Constraints Enable Constraint Simplication
  - $X_1 < X_2 < \ldots < X_m$
    - alldifferent on X_i becomes redundant.
    - for all i<j $D_{ij} = |X_i - X_j|$ → for all i<j $D_{ij} = X_j - X_i$

Improved Model

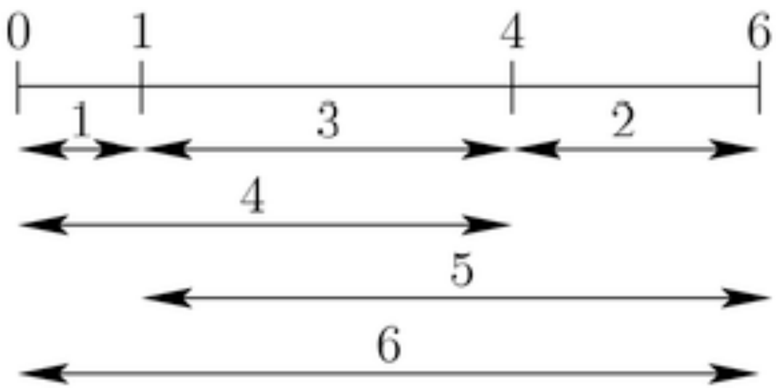
- Symmetry Breaking Constraints Enable Additional Implied Constraints
  - for all i<j<k $D_{ij} < D_{ik}$ and $D_{jk} < D_{ik}$
    $D_{ik} = D_{ij} + D_{jk}$



An optimal Golomb ruler of order 4.

Improved Model

- Deducing information from Golomb Rulers of smaller order
  - If you consider any k consecutive marks of a Golomb Ruler of order n > k, they form a Golomb Ruler of order k.



An optimal Golomb ruler of order 4.

Improved Model

- Deducing information from Golomb Rulers of smaller order
  - If you consider any k consecutive marks of a Golomb Ruler of order $n > k$, they form a Golomb Ruler of order k.
  - Therefore they must span over a distance at least as long as the optimal size of Rulers of order k.
  - for all $i<j$ $D_{ij} \geq Ruler(j-i+1)$
  - Can you find others?

Modeling is Critical!

- Given the human understanding of a problem, we need to answer questions like:
  - which variables shall I choose?
  - which constraints shall I enforce?
  - can I exploit any global constraints?
  - so I need any auxiliary variables?
  - are some constraints redundant, therefore can be avoided?
  - are there any implied constraints?
  - can symmetry be eliminated?
  - are there any dual viewpoints?
  - among alternative models, which one shall I prefer?

Dual Viewpoint

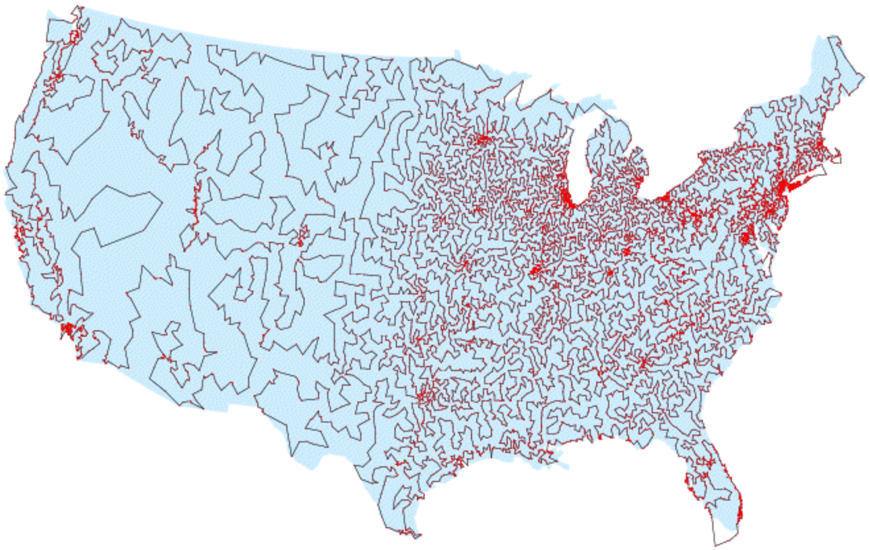
- Viewing a problem P from different perspectives may result in different models of P.
- Each models yields the same set of solutions.
- Each dual model exhibits in general a different representation of P.
  - Different variables.
  - Different domains.
  - Different constraints.
    - Different size of the search space!

Permutation Problems

- A CSP is a permutation problem if:
  - it has the same number of values as variables;
  - all variables have the same domain;
  - each variable must be assigned a different value → alldifferent constraint.
- Any solution assigns a permutation of the values to the variables.
- Other constraints determine which permutations are solutions.

Permutation Problems

- Many examples in:
  - scheduling;
  - assignment;
  - routing;
  - timetabling.



TSP problem = find permutation of cities which makes a tour of minimum length

Dual Viewpoints in Permutations

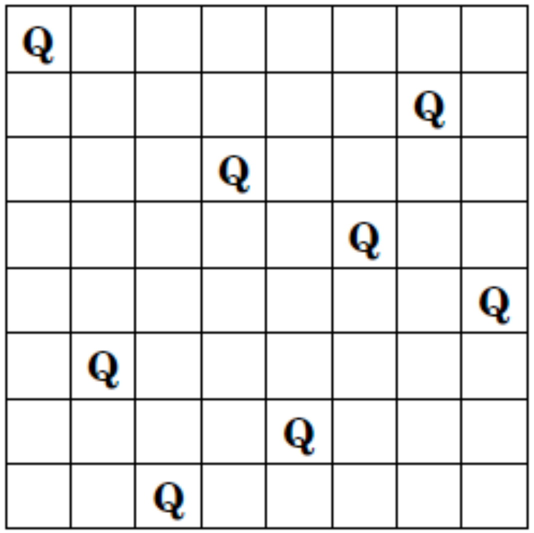
- Each variable is assigned exactly one value.
- Each possible value is assigned to exactly one variable.
- The dual CSP:
  - Also a permutation problem.
  - Interchanges variables and values.
- E.g., if there are n people to do n tasks:
  - Tasks → people
  - People → tasks

Dual Viewpoints in Permutations

- Consider a permutation problem with n variables and values.
- One model:
  - Variables $[X_1, \ldots, X_n]$ for n different positions.
  - Domains {1,...,n} for n different values in the positions.
  - alldifferent($[X_1, X_2, \ldots, X_n]$).
  - Other problem constraints.
- Dual model:
  - Variables $[Y_1, \ldots, Y_n]$ for n different values in the positions.
  - Domains {1,...,n} for n different positions.
  - alldifferent($[Y_1, Y_2, \ldots, Y_n]$).
  - Other problem constraints.
- The two models may yield the same or totally different CSPs.

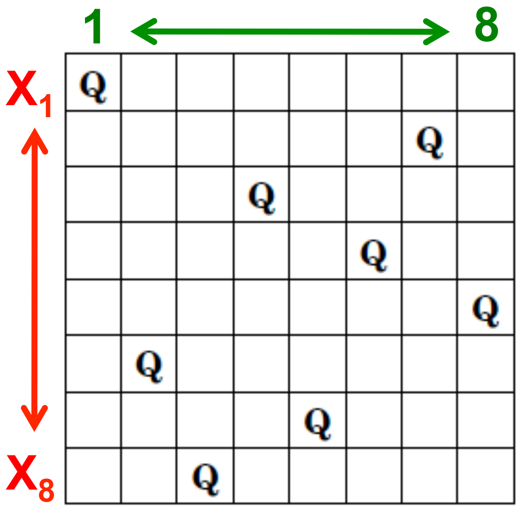
N-Queens

- Place n queens in an nxn board so that no two queens can attack each other.



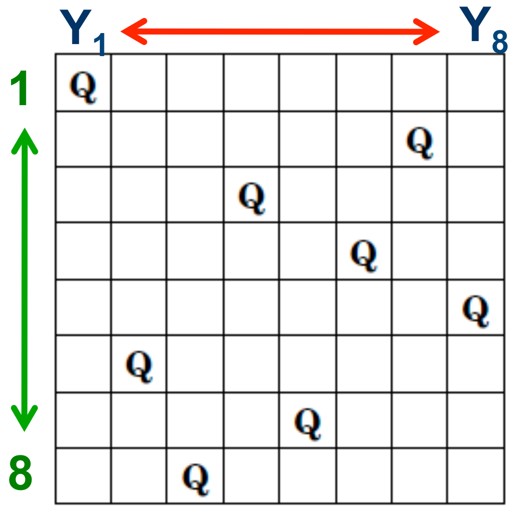
8 queens on an 8×8 chessboard, no queen attacking any other

N-Queens



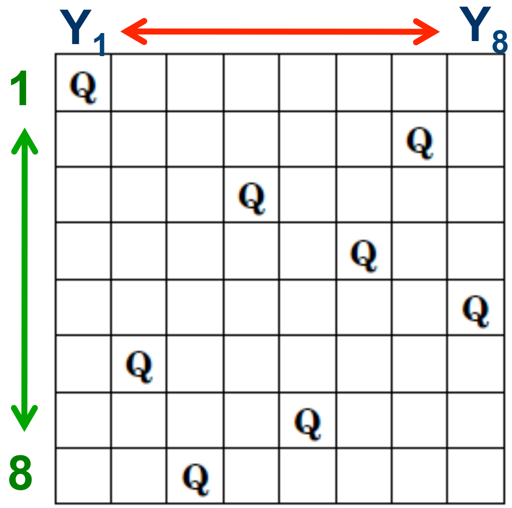
- One model
  - A variable for each row : $[X_1, X_2, \ldots, X_n]$ → no row attack
  - Domain values represent the columns: {1,…,n}
    - $X_i = j$ means that the queen in row i is in column j
  - Constraints:
    - alldifferent($[X_1, X_2, \ldots, X_n]$) → no column attack
    - for all $i<j$ $|X_i - X_j| \neq |i-j|$ → no diagonal attack

N-Queens



- Dual model
  - A variable for each column : $[Y_1, Y_2, \ldots, Y_n]$ → no column attack
  - Domain values represent the rows: {1,...,n}
    - $Y_i = j$ means that the queen in column i is in row j
  - Constraints:
    - alldifferent($[Y_1, Y_2, \ldots, Y_n]$) → no row attack
    - for all i<j $|Y_i - Y_j| \neq |i-j|$ → no diagonal attack

N-Queens



Both viewpoints yield the same CSP!

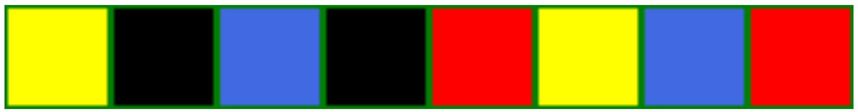
- Dual model
  - A variable for each column : $[Y_1, Y_2, ..., Y_n]$ → no column attack
  - Domain values represent the rows: {1,...,n}
    - $Y_i = j$ means that the queen in column i is in row j
  - Constraints:
    - alldifferent($[Y_1, Y_2, \ldots, Y_n]$) → no row attack
    - for all $i<j$ $|Y_i-Y_j| \neq |i-j|$ → no diagonal attack

Langford’s Problem: L(k,n)

- Find a sequence of length n*k of numbers [1,n] s.t. for all i in [1,n]:
  - i appears k times in the sequence;
  - there are i other numbers between the k successive occurrences of i.

L(2,4)

4 1 3 1 2 4 3 2



- Due to the mathematician Dudley Langford.
  - Watched his son build a tower which solved L(2,3).

First Basic Model

- Variables and Domains
  - Find a position for each number.
  - How many numbers?
  - How many positions?

First Basic Model

- Variables and Domains
  - Find a position for each number.
  - How many numbers?
  - How many positions?
    - n*k numbers, n*k positions

First Basic Model

- Variables and Domains
  - Find a position for each number.
  - $[N_{11,} N_{21,} \ldots, N_{n1}, \ldots, N_{1k,} N_{2k,} \ldots, N_{nk}]$
  - N_{ij} for the j^{th} occurrence of i in [1,n]:
    - N_{11} → 1^{sth} occurrence of 1
    - N_{21} → 1^{sth} occurrence of 2
    - …
    - N_{12} → 2^{nd} occurrence of 1
    - N_{22} → 2^{nd} occurrence of 2
    - …

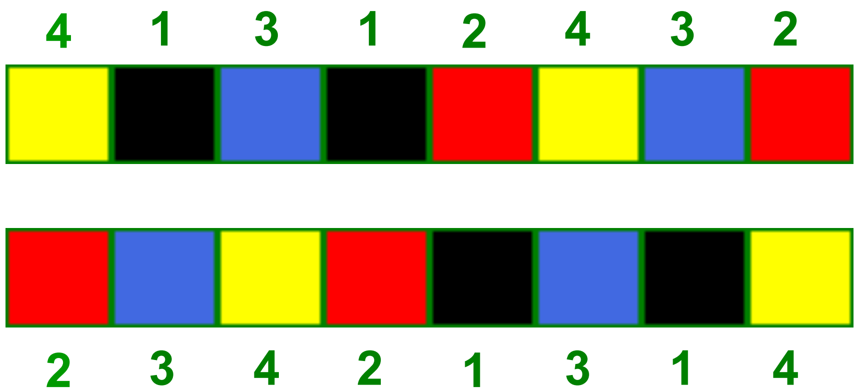
    ⇨ i appears k times in the sequence
  - Domain values {1,...,n*k} represent the positions in the sequence:
    - N_{ij}= p means the j^{th} occurrence of i is in position p.

First Basic Model

- Constraints
  - alldifferent([N_{11}, N_{21}, …, N_{n1}, …, N_{1k}, N_{2k}, …, N_{nk}])
  - for all i,j, $N_{i(j+1)} = (i+1)+N_{ij}$

First Basic Model

- Symmetry
  - Each sequence can be inverted.



- Symmetry Breaking Constraints
  - L(3,9):

    $N_{92} < 14$ (2nd occurrence of 9 is in the 1sth half)

    OR $N_{92}=14$ and $N_{82}<14$ (2nd occurrence of 8 is in the 1sth half)

Dual Model

- Variables and Domains
  - Find a number for each position.
  - $[P_{1,} P_{2,} \ldots, P_{n*k}]$
  - Domain values?

Dual Model

- Variables and Domains
  - Find a number for each position.
  - $[P_{1}, P_{2}, \ldots, P_{n*k}]$
  - Domain values?
    - If use the numbers to place, we cannot use an alldifferent constraint.
    - Each number occurs not once but k times.

Dual Model

- Variables and Domains
  - Find a number for each position.
  - $[P_1, P_2, \ldots, P_{n*k}]$
  - Domain values
    - Solution 1: use {1,...,n*k} with the value i+(j-1)*n standing for the j^{th} occurrence of i.
- Constraints
  - alldifferent($[P_1, P_2, \ldots, P_{n*k}]$)
  - distance constraints?

Dual Model

- Variables and Domains
  - Find a number for each position.
  - $[P_{1,} P_{2,} \ldots, P_{n*k}]$
  - Domain values
    - Solution 2: use {1,...,n}
- Constraints
  - Each number must occur exactly k times!
    - Global cardinality constraint

      $gcc([X_1, X_2, \ldots, X_n], [v_1, \ldots, v_m], [O_1, \ldots, O_m])$ iff

      forall $j \in \{1,\ldots, m\}$ $O_j = |\{X_i \mid X_i = v_j, 1 \leq i \leq n \}|$
    - $gcc([P_1, P_2, \ldots, P_{n*k}], [1, \ldots, n], [k, \ldots, k])$

Dual Model

- Variables and Domains
  - Find a number for each place.
  - $[P_{1,} P_{2,} \ldots, P_{n*k}]$
  - Domain values
    - Solution 2: use {1,..., n}
- Constraints
  - Each number must occur exactly k times!
    - $gcc([P_1, P_2, \ldots, P_{n*k}], [1, ..., n], [k, \ldots, k])$
  - Distance constraints?

Which Model?

- First Model
  - alldifferent constraint.
  - Distance constraints.
    - Easily expressible.
    - Good propagation.
- Dual model
  - More natural to find numbers for positions.
    - Searching on P variables might be beneficial.
  - alldifferent/gcc constraint.
  - Distance constraints?

Combined Models

- Often different views allow different expression of the constraints and different implied constraints:
  - can be hard to decide which is better!
- We can then use multiple models and combine them via channelling constraints to keep consistency between the variables.
- Benefits:
  - Enhanced constraint propagation.
  - Facilitation of the expression of constraints.
  - More options for search variables.

Combined Permutation Models

- Model 1
  - Variables [X_1, …, X_n] for n different positions.
  - Domains {1,...,n} for n different values in the positions.
  - alldifferent([X_1, X_2, …, X_n]).
  - Other problem constraints PC_1.
- Model 2
  - Variables [Y_1, …, Y_n] for n different values in the positions.
  - Domains {1,...,n} for n different positions.
  - alldifferent([Y_1, Y_2, …, Y_n]).
  - Other problem constraints PC_2.
- Combined Model
  - Model 1 + Model 2
  - Channelling constraints

Combined Permutation Models

- Model 1
  - Variables [X_1, …, X_n] for n different positions.
  - Domains {1,...,n} for n different values in the positions.
  - alldifferent([X_1, X_2, …, X_n]).
  - Other problem constraints PC_1.
- Model 2
  - Variables [Y_1, …, Y_n] for n different values in the positions.
  - Domains {1,...,n} for n different positions.
  - alldifferent([Y_1, Y_2, …, Y_n]).
  - Other problem constraints PC_2.
- Combined Model
  - Model 1 + Model 2
  - Channelling constraints
    - for all i,j $X_i = j \leftrightarrow Y_j = i$

Combined Permutation Models

- Channelling Constraints
  - N-Queens
    - for all i,j $X_i = j \leftrightarrow Y_j = i$

Combined Permutation Models

- Channelling Constraints
  - N-Queens
    - for all i,j $X_i = j \leftrightarrow Y_j = i$
  - Langford
    - for all i,j,p $N_{ij} = p \leftrightarrow P_p = i+(j-1)*n$
    - for all i,j,p $N_{ij} = p \rightarrow P_p = i$

Combined Permutation Models

- Benefits of Channelling
  - N-Queens
    - Improved propagation.
    - Search on both sets of variables.
  - Langford
    - Improved propagation.
    - Search on both sets of variables.
    - Facilitates the expression of messy constraints.

Combined Permutation Models

- Combined Model 1
  - Model 1 + Model 2
  - Channelling constraints
    - for all i,j $X_i = j \leftrightarrow Y_j = i$
- Combined Model 2
  - Model 1 + [Y_1, Y_2, …, Y_n], or
  - [X_1, X_2, …, X_n] + Model 2
  - Channelling constraints
    - for all i,j $X_i = j \leftrightarrow Y_j = i$

Combined Permutation Models

- Observation
  - alldifferent is redundant.
- Combined Model 3
  - $[X_1, X_2, \ldots, X_n]$ + PC_1 + $[Y_1, Y_2, \ldots, Y_n]$ + PC_2
  - Channelling constraints
    - for all i,j $X_i = j \leftrightarrow Y_j = i$
- Combined Model 4
  - $[X_1, X_2, \ldots, X_n]$ + $[Y_1, Y_2, \ldots, Y_n]$ + PC on suitable variables
  - Channelling constraints
    - for all i,j $X_i = j \leftrightarrow Y_j = i$

Combined Permutation Models

- Best N-Queens Problem Model
  - Variables
    - $[X_1, X_2, \ldots, X_n], [Y_1, Y_2, \ldots, Y_n]$
  - Constraints
    - for all i,j $X_i = j \leftrightarrow Y_j = i$ $\longrightarrow$ Channelling constraints
    - for all i<j $|X_i - X_j| \neq |i-j|$ $\longrightarrow$ PC_1
  - Search variables
    - $[X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_n]$

Combined Permutation Models

- Best Langford's Problem Model
  - Variables
    - $[N_{11}, N_{21}, \ldots, N_{n1}, \ldots, N_{1k}, N_{2k}, \ldots, N_{nk}], [P_1, P_2, \ldots, P_{n*k}]$
  - Constraints
    - for all i,j,p $N_{ij} = p \leftrightarrow P_p = i+(j-1)*n$ $\longrightarrow$ Channelling constraints
    - for all i,j, $N_{i(j+1)} = (i+1)+N_{ij}$ $\longrightarrow$ PC_1
  - Search variables
    - $[P_1, P_2, \ldots, P_{n*k}]$
    - $[N_{11}, N_{21}, \ldots, N_{n1}, \ldots, N_{1k}, N_{2k}, \ldots, N_{nk}, P_1, P_2, \ldots, P_{n*k}]$

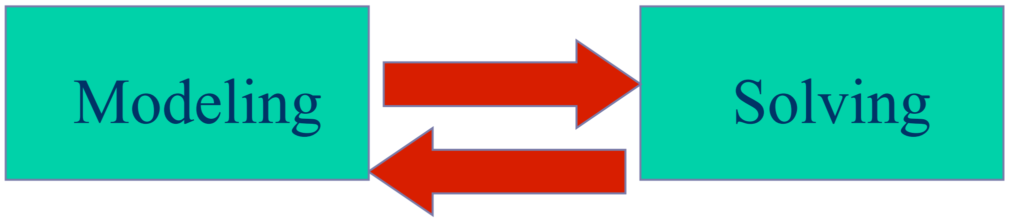
Combined Models in General

- Best Golomb Ruler Problem Model
  - Variables:
    - $[X_1, X_2, .., X_m], [D_{12}, D_{13}, \ldots, D_{(m-1)m}]$
  - Constraints:
    - for all $i<j$ $D_{ij} = X_j - X_i$ → Channelling constraints
    - $alldifferent([D_{12}, D_{13}, \ldots, D_{(m-1)m}])$
    - $X_1 < X_2 < \ldots < X_m$
    - $X_1 = 0$
    - for all $i<j<k$ $D_{ij} < D_{ik}$ and $D_{jk} < D_{ik}$
      $D_{ik} = D_{ij} + D_{jk}$
    - for all $i<j$ $D_{ij} \geq Ruler(j-i+1)$

    } Problem/implied/symmetry breaking constraints on suitable variables
  - Search Variables:
    - $[X_1, X_2, .., X_m]$

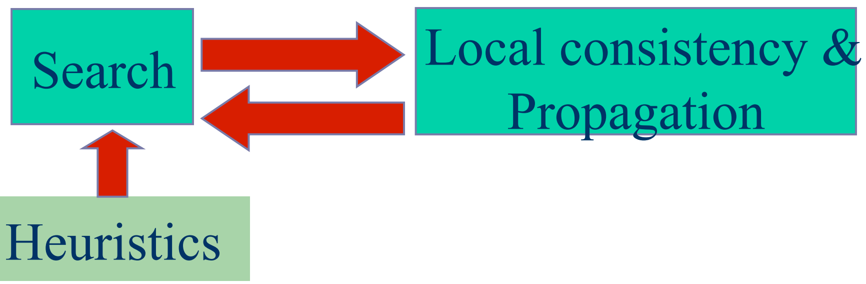
CP Machinery

- Modeling and solving are strongly interconnected.



Solving

- The user lets the CP technology solve the CSP:
  - choose a search algorithm:
    - usually backtracking search performing a depth-first traversal of a search tree.
  - integrate local consistency and propagation.
  - choose heuristics for branching on the search tree:
    - which variable to branch on?
    - which value to branch on?

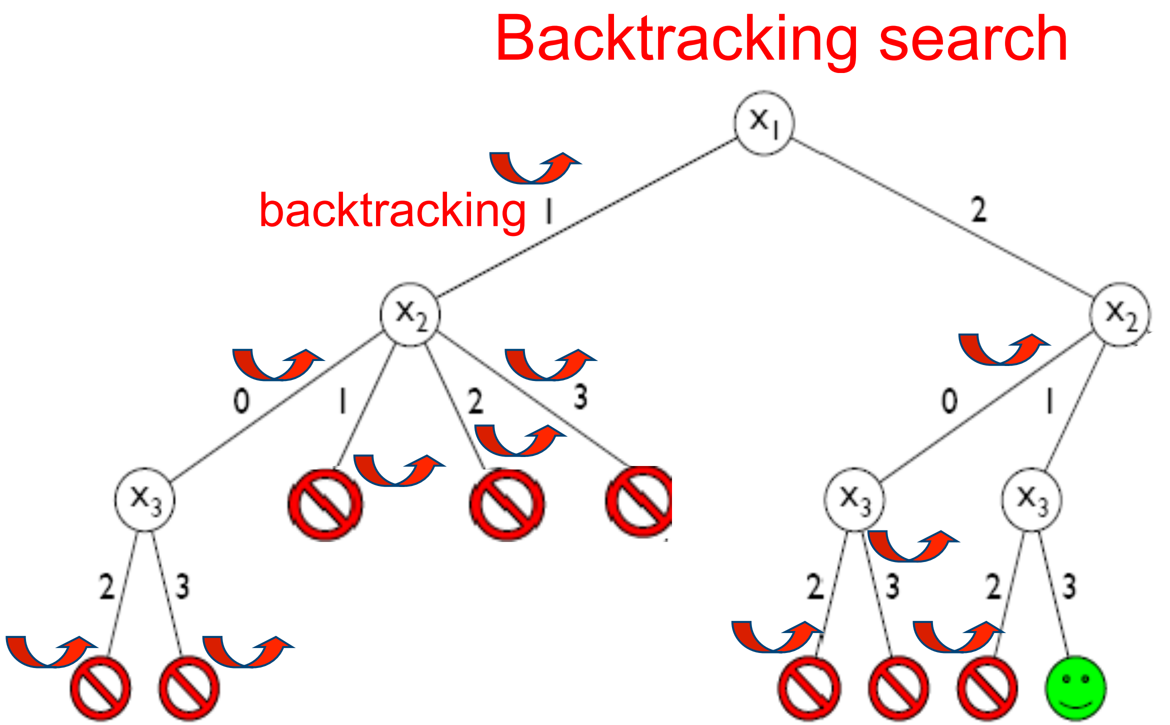


Backtracking Search

- A possible efficient and simple method.
- Variables are instantiated sequentially.
- Whenever all the variables of a constraint is instantiated, the validity of the constraint is checked.
- If a (partial) instantiation violates a constraint, backtracking is performed to the most recently instantiated variable that still has alternative values.
- Backtracking eliminates a subspace from the cartesian product of all variable domains.
- Essentially performs a depth-first search.

Backtracking Search

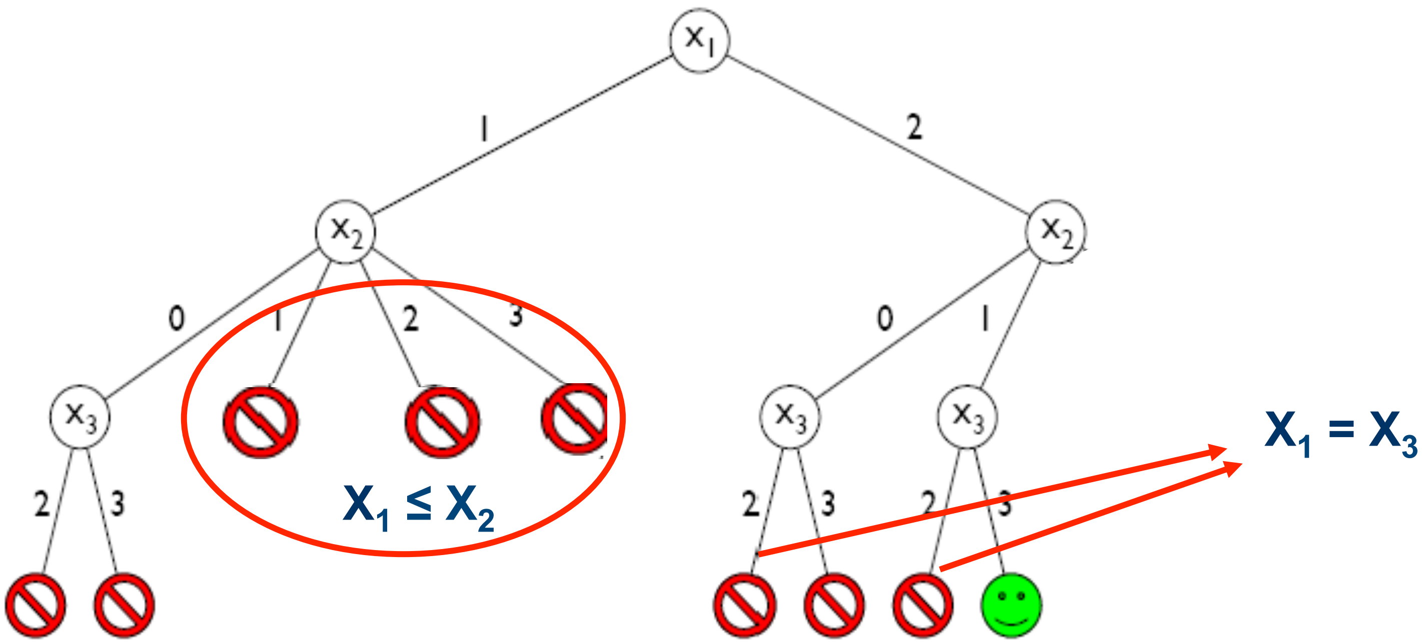
- $X_1 \in \{1,2\}$ $X_2 \in \{0,1,2,3\}$ $X_3 \in \{2,3\}$
- $X_1 > X_2$ and $X_1 + X_2 = X_3$ and alldifferent($[X_1, X_2, X_3]$)



Fails 8 times!

Backtracking Search

- Backtracking suffers from thrashing ☹ :
  - performs checks only with the current and past variables;
  - search keeps failing for the same reasons.

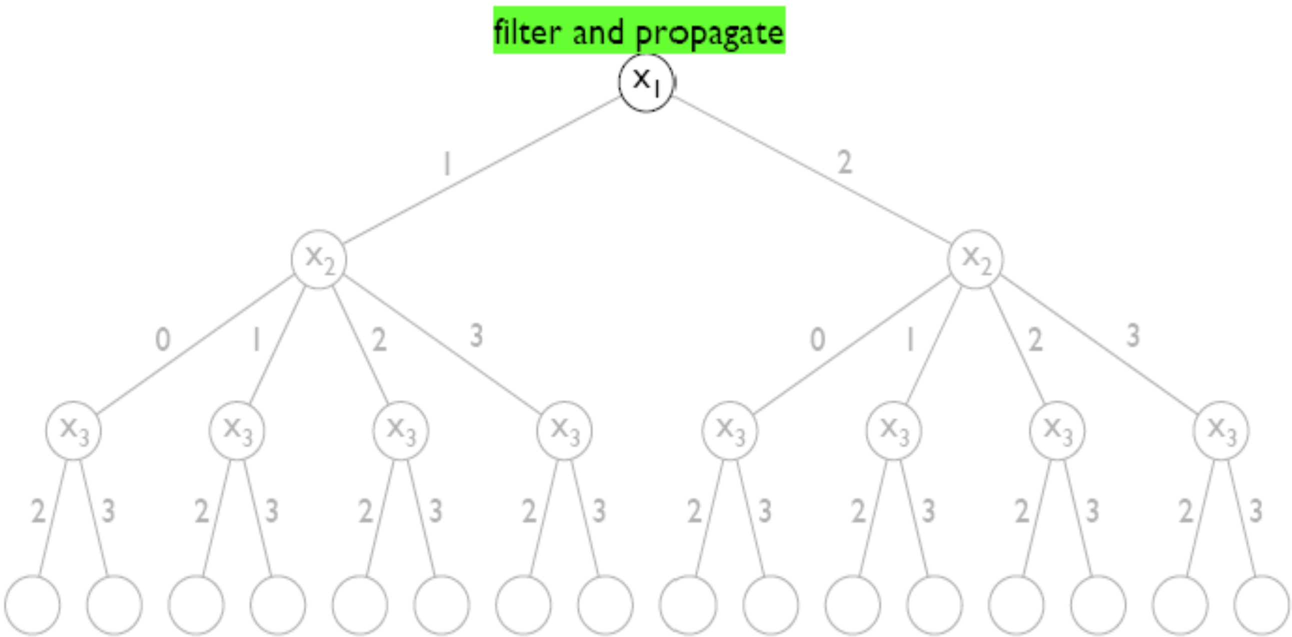


Local Consistency & Propagation

- We can reason about the properties of constraints and their effect on their variables.
- Some values can be filtered from some domains, reducing the backtracking search space significantly!

BS + Local Consistency & Propagation

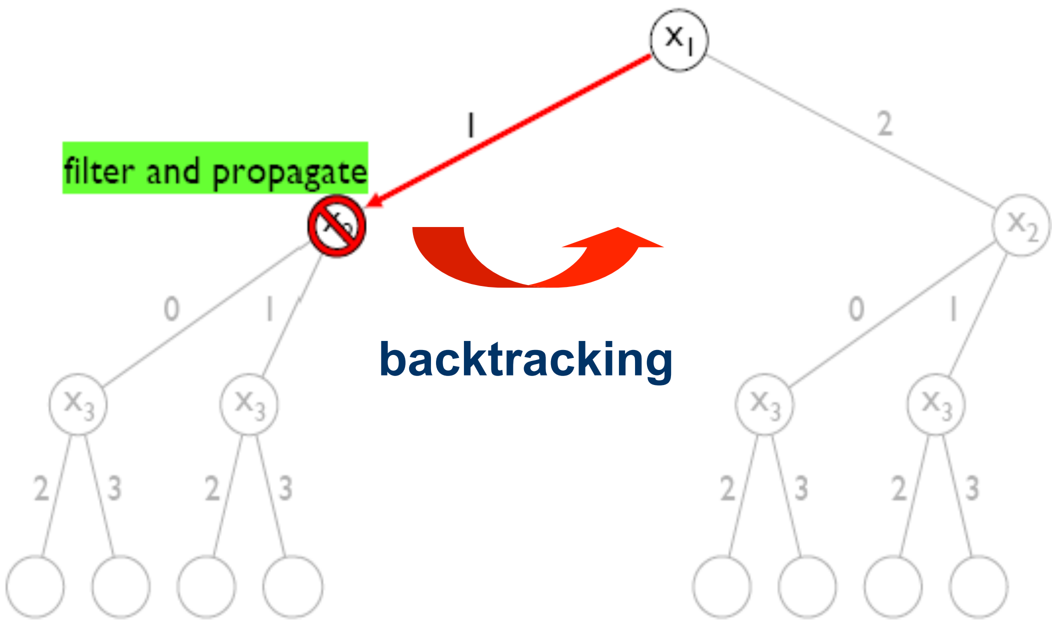
- $X_1 \in \{1,2\}$ $X_2 \in \{0,1,\not{2},\not{3}\}$ $X_3 \in \{2,3\}$
- $X_1 > X_2$ and $X_1 + X_2 = X_3$ and alldifferent($[X_1, X_2, X_3]$)



BS + Local Consistency & Propagation

- $X_1 \in$ {1,~~2~~} $X_2 \in$ {0,~~1~~} $X_3 \in$ {~~2~~,~~3~~}
- $X_1 > X_2$ and $X_1 + X_2 = X_3$ and alldifferent([X_1, X_2, X_3])

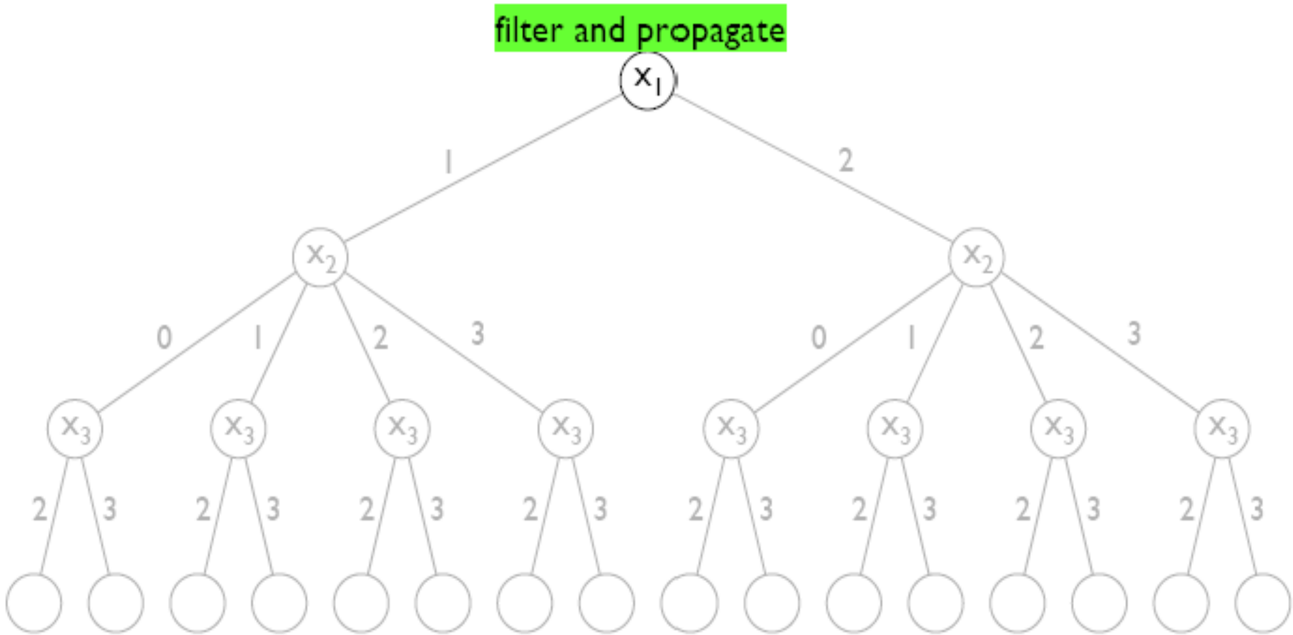
  Backtracking search + local consistency/propagation



BS + Local Consistency & Propagation

- $X_1 \in \{1,2\}$ $X_2 \in \{0,1,\cancel{2},\cancel{3}\}$ $X_3 \in \{2,3\}$
- $X_1 > X_2$ and $X_1 + X_2 = X_3$ and alldifferent($[X_1, X_2, X_3]$)

Backtracking search + local consistency/propagation

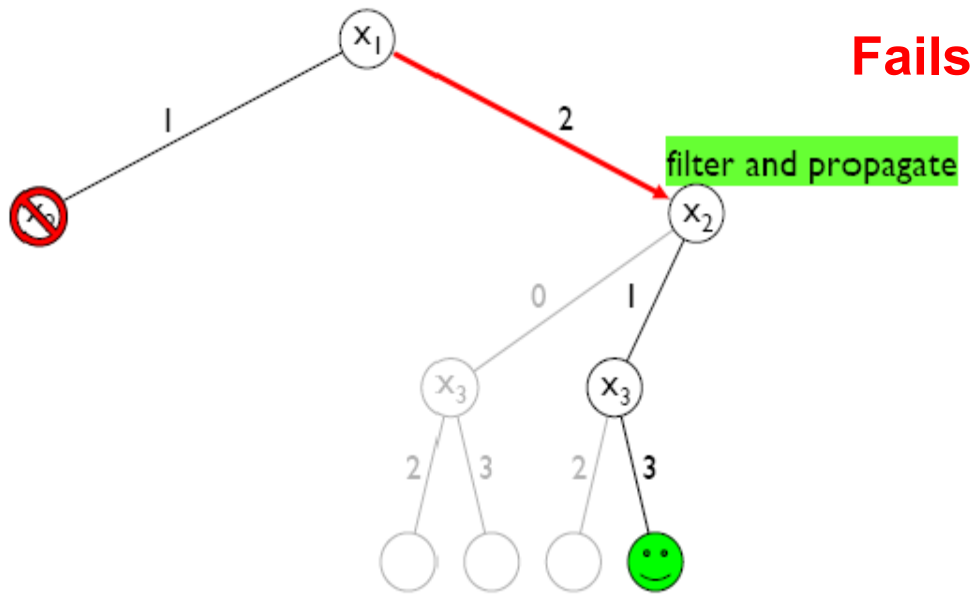


BS + Local Consistency & Propagation

- $X_1 \in \{\not{1},2\}$ $X_2 \in \{\not{0},1\}$ $X_3 \in \{\not{2},3\}$
- $X_1 > X_2$ and $X_1 + X_2 = X_3$ and alldifferent($[X_1, X_2, X_3]$)

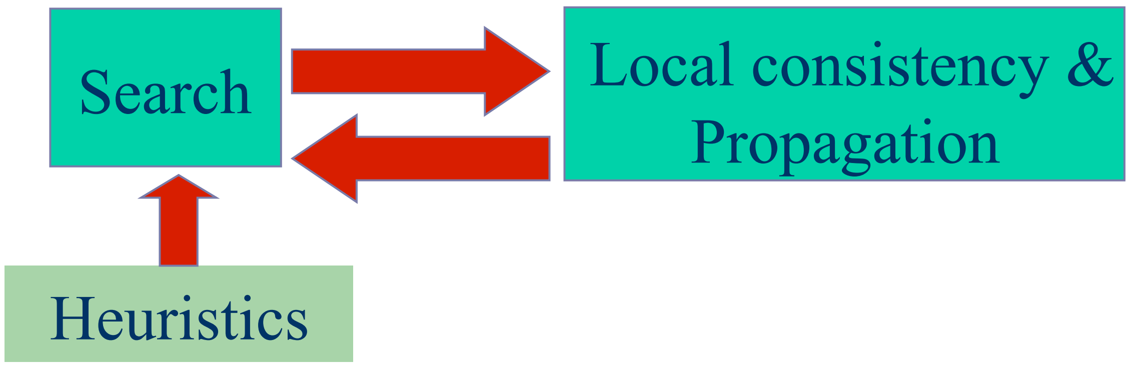
  Backtracking search + local consistency/propagation

Fails only once!



Local Consistency & Propagation

- Central to the process of solving CSPs which are inherently intractable.



PART II: Local Consistency & Constraint Propagation

AIMMS

- Modeling language with an interface to CP and MIP solvers (http://www.aimms.com/cp)
- Student license
- GUI support available only in the Windows version
  - Create a virtual machine with Windows OS in your computer
  - Virtual Box (https://www.virtualbox.org/)
- Extensive documentation
  - One-hour tutorial (general introduction)
  - Chapter 21 – "Constraint Programming" of Language Reference