DM841

Discrete Optimization

Part II

Lecture 1

Introduction to Constraint Programming

Combination

Simplification


Contradiction

Redundancy

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Outline

Schedule and Material

- Schedule:
  - Wednesday 10:15-12:00
  - Friday 8.15-10:00
  - Tuesday 14:15-16:00 (only from week 48)
  - Officially last lecture in Week 51, Friday, 19th December, 2014
- Communication tools
  - Public Course Webpage (Wp)
    http://www.imada.sdu.dk/~marco/DM841/
  - In Blackboard (Bb):
    - Announcements
    - Documents (Photocopies)
  - Personal email
  - You are welcome to visit me in my office in working hours.

Contents

- The technology behind Constraint Programming
- CP in gecode
- Modelling and solving constrained optimization problems

Evaluation

- Two obligatory assignments (pass/fail)
- Final assignment (50% of final grade)
  - Model
  - Implementation
  - Report (Max 10 pages)

References

- Main References:

RBW F. Rossi, P. van Beek and T. Walsh (ed.), Handbook of Constraint Programming, Elsevier, 2006

MPG C. Schulte, G. Tack, M.Z. Lagerkvist, Modelling and Programming with Gecode 2013

- Photocopies (Bb)
- Articles from the Webpage
- Lecture slides
- Assignments

- Active participation

Software

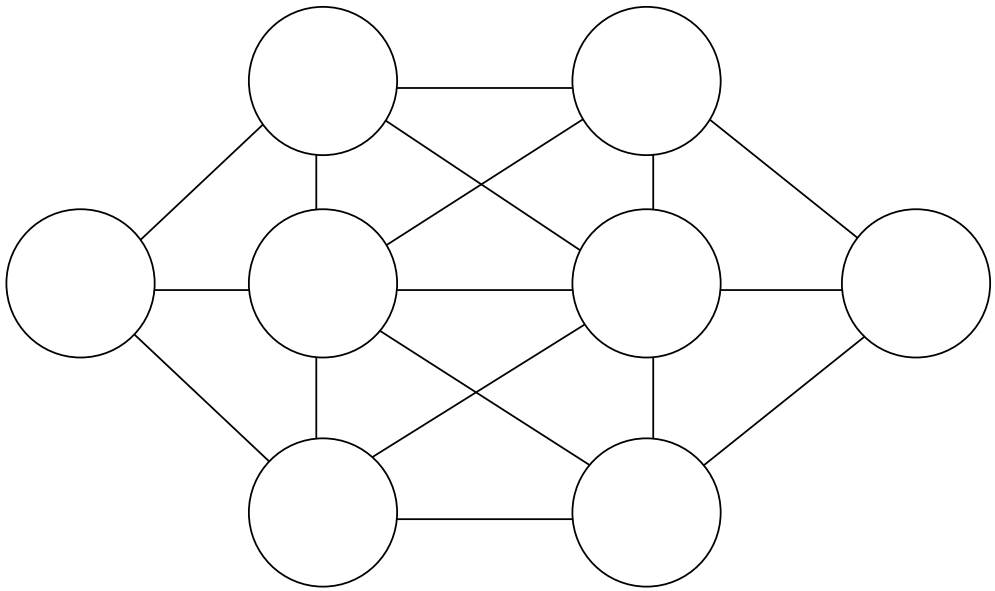
Under development:
http://www.minizinc.org/challenge2014/results2014.html

Here, we will use free and open-source software:

- Gecode (C++) – MIT license

Outline

Outline



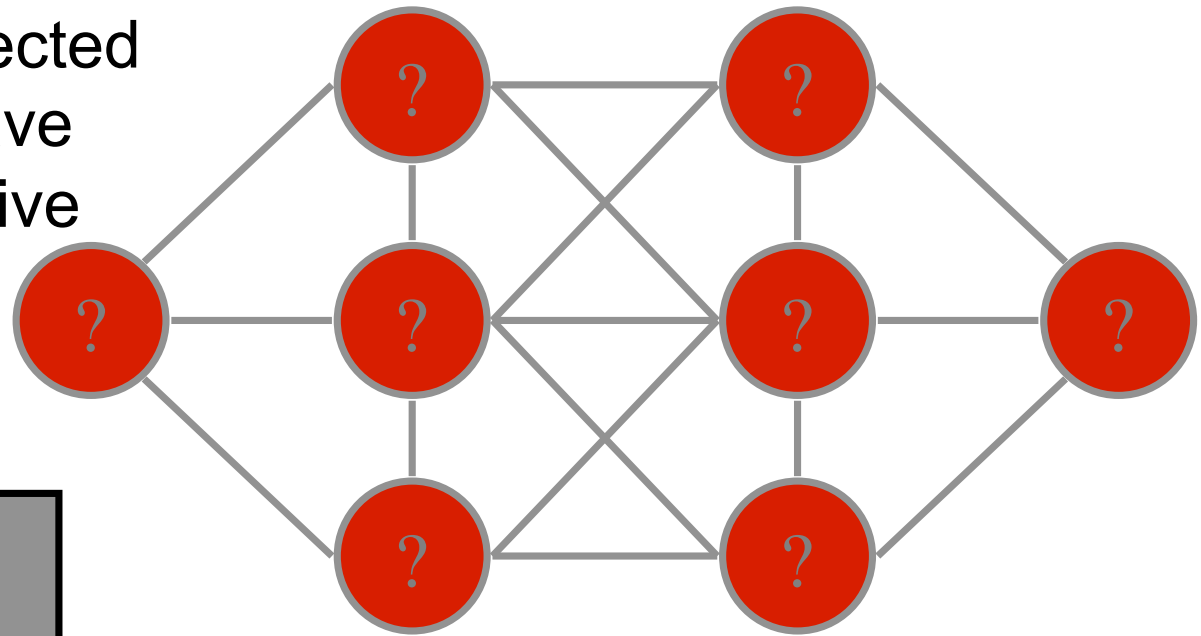
Put a different number in each circle (1 to 8) such that adjacent circles cannot take consecutive numbers

Constraint Programming
An Introduction
by example

Patrick Prosser
with the help of Toby Walsh, Chris Beck,
Barbara Smith, Peter van Beek, Edward Tsang, ...

A Puzzle

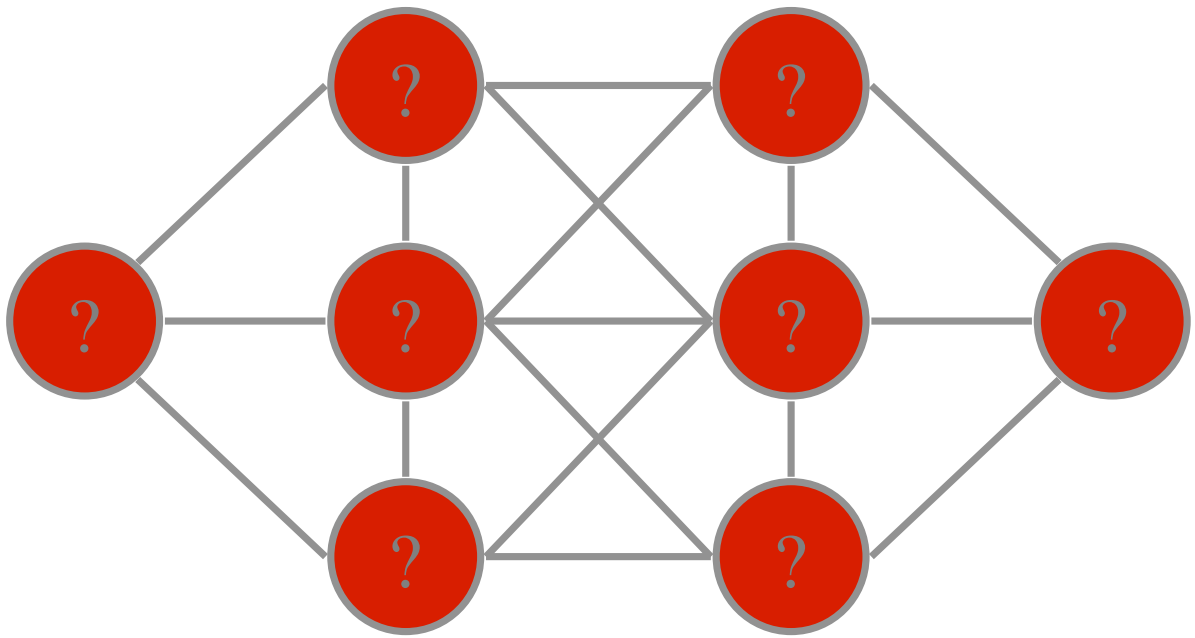
- Place numbers 1 through 8 on nodes
  - Each number appears exactly once
  - No connected nodes have consecutive numbers



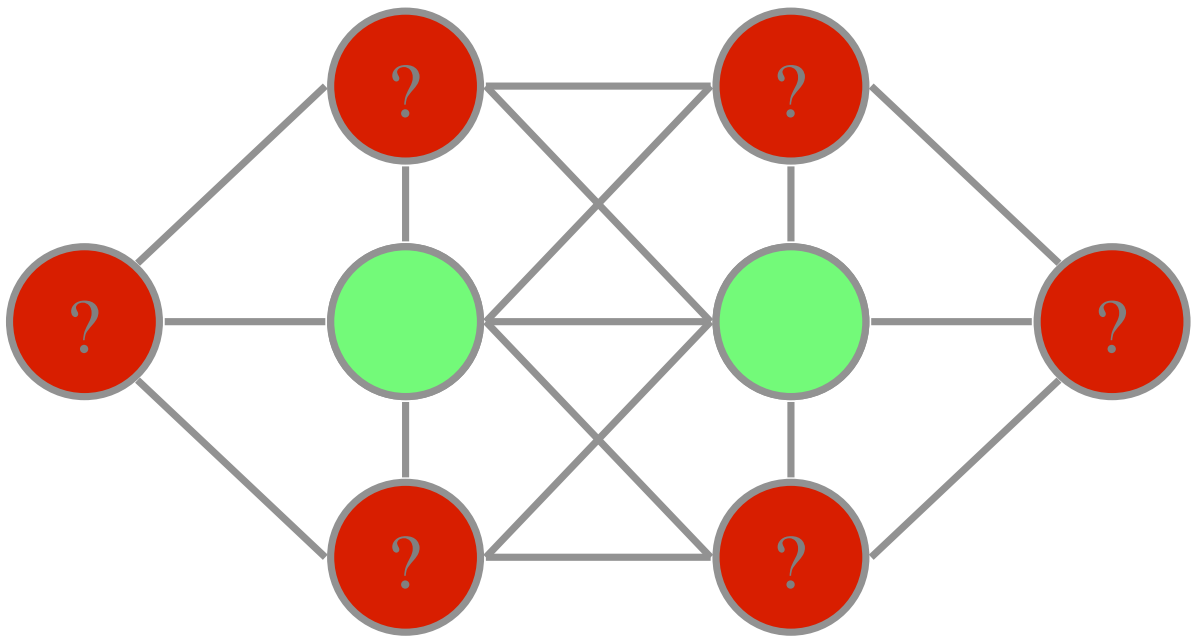
You have
8 minutes!

Heuristic Search

Which nodes are hardest to number?

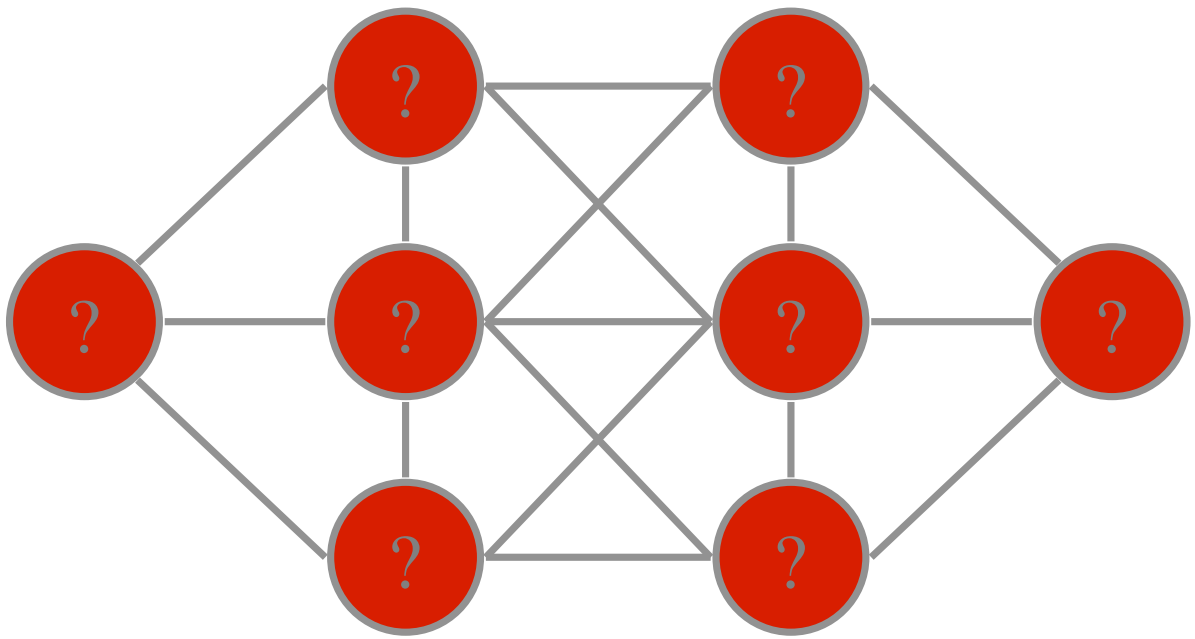


Heuristic Search



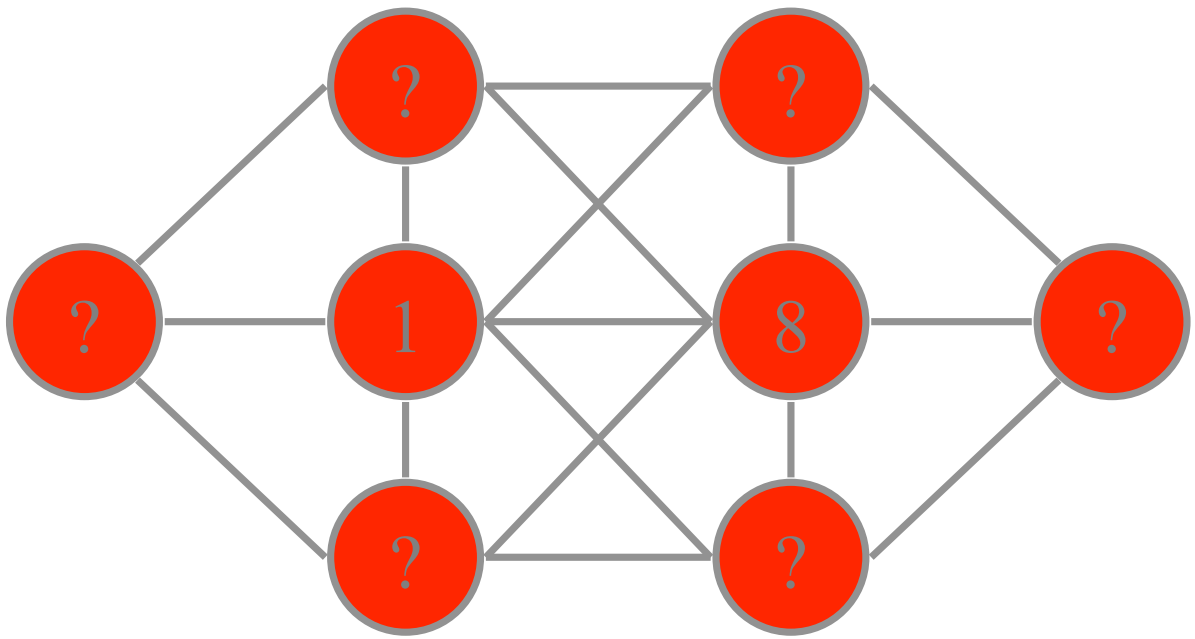
Heuristic Search

Which are the least constraining values to use?



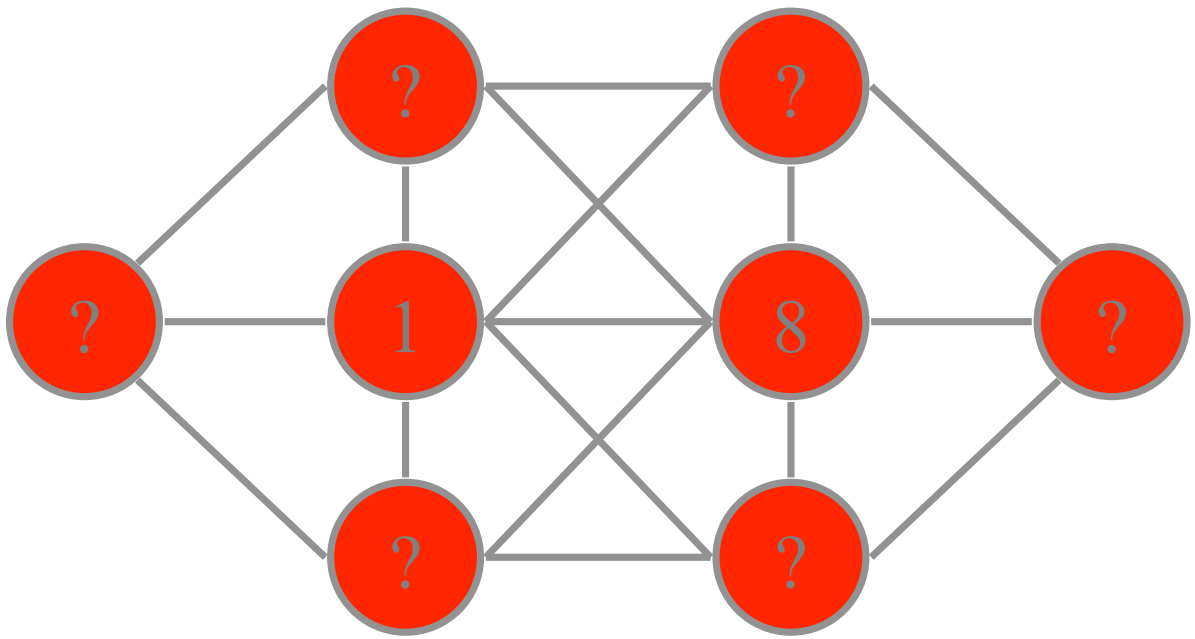
Heuristic Search

Values 1 and 8



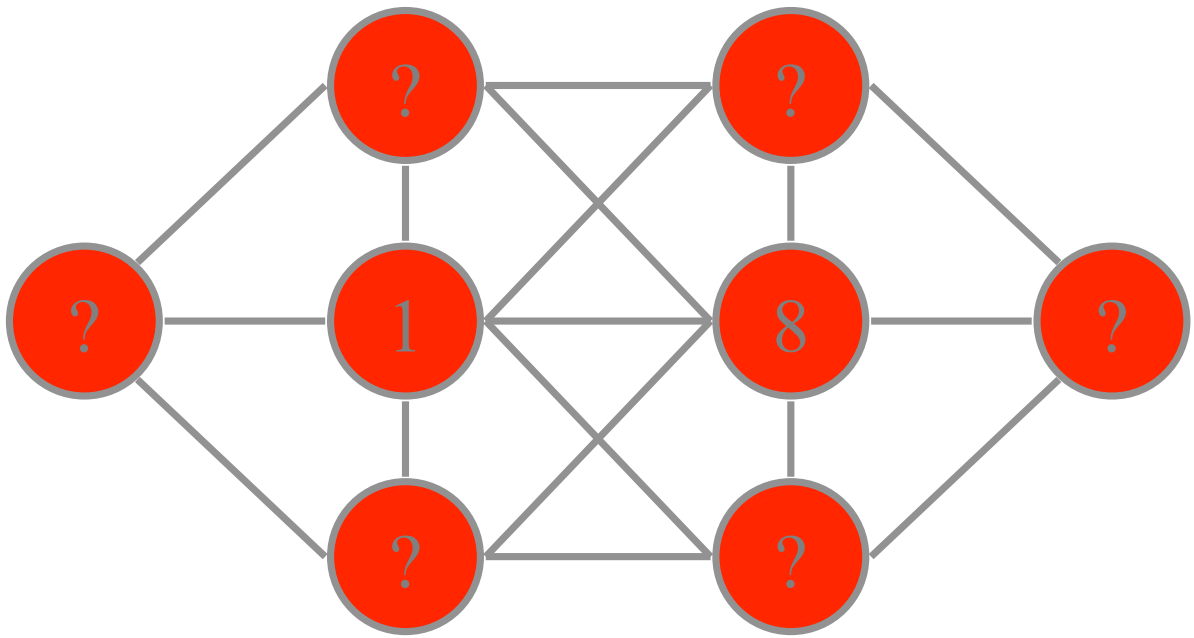
Heuristic Search

Values 1 and 8



Symmetry means we don't need to consider: 8 1

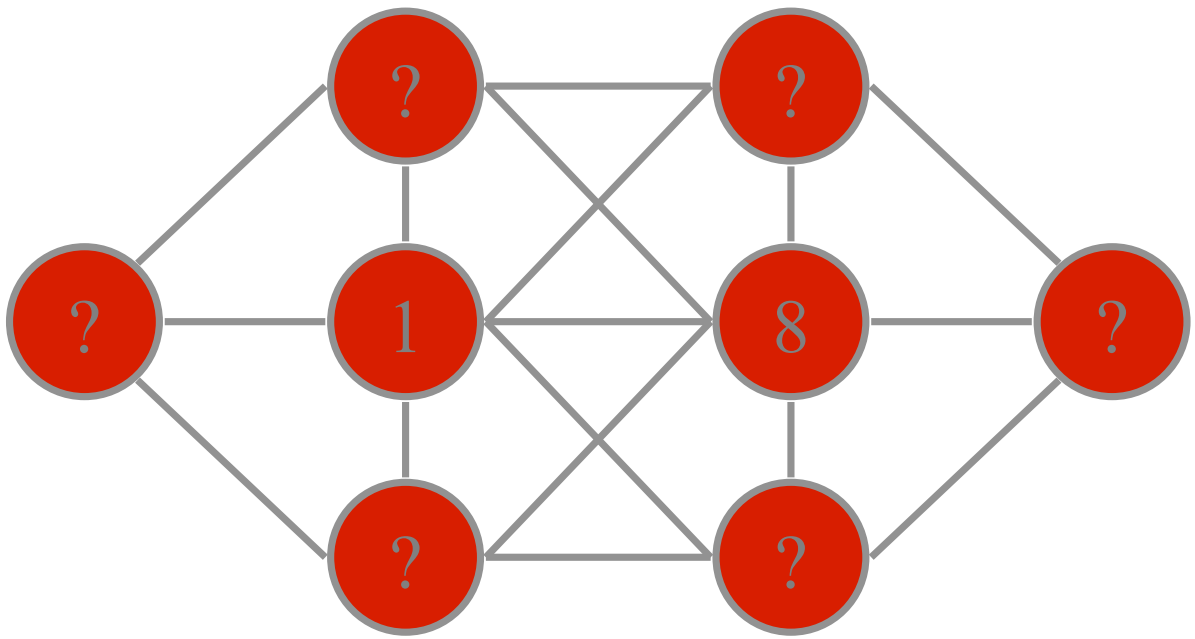
Inference/propagation



We can now eliminate many values for other nodes

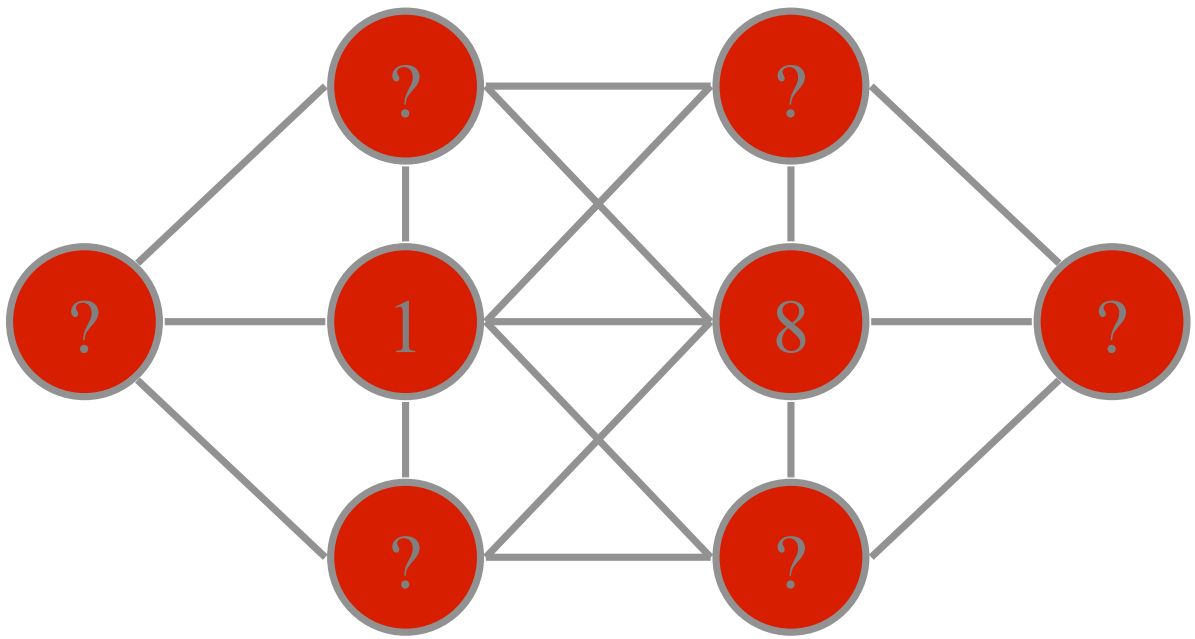
Inference/propagation

{1,2,3,4,5,6,7,8}



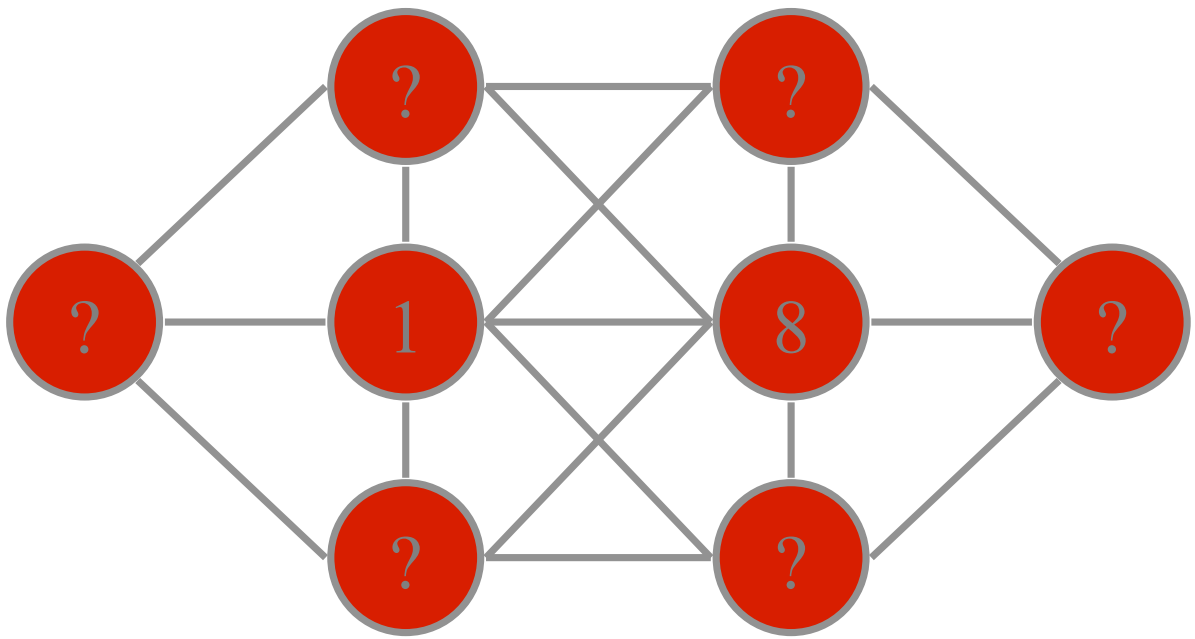
Inference/propagation

{2,3,4,5,6,7}

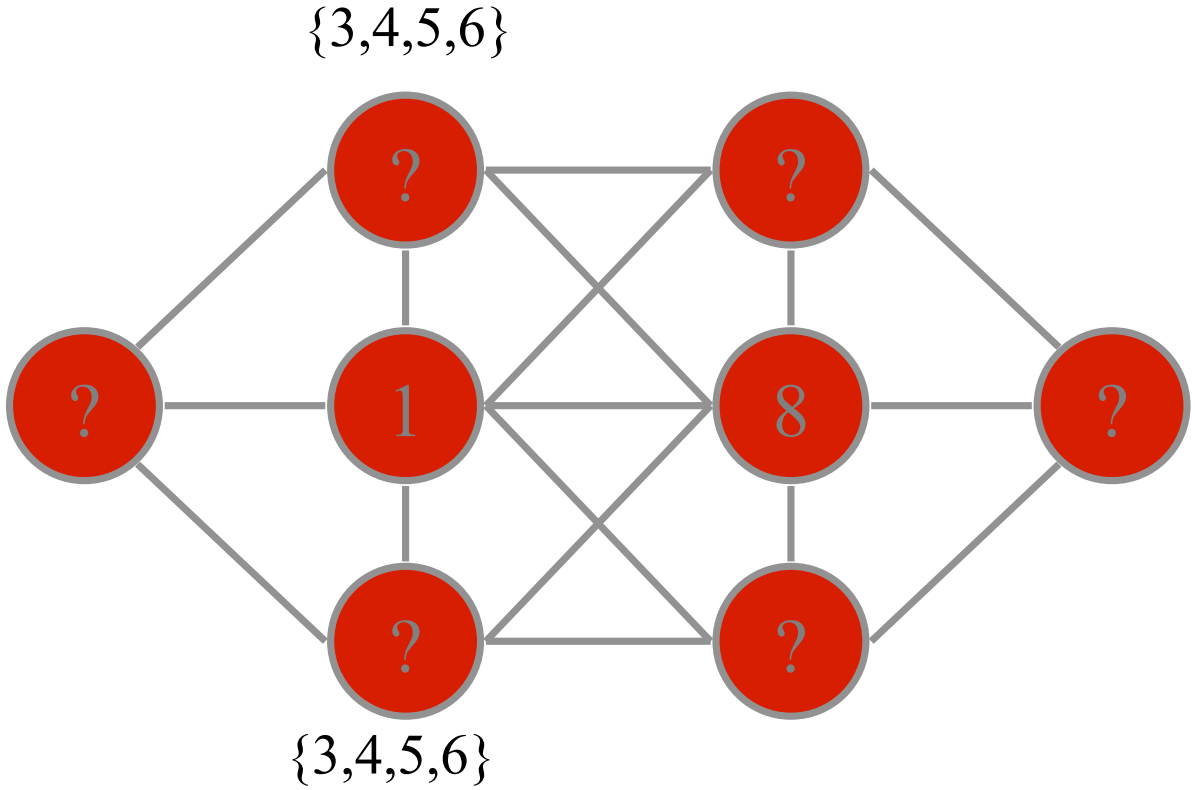


Inference/propagation

{3,4,5,6}

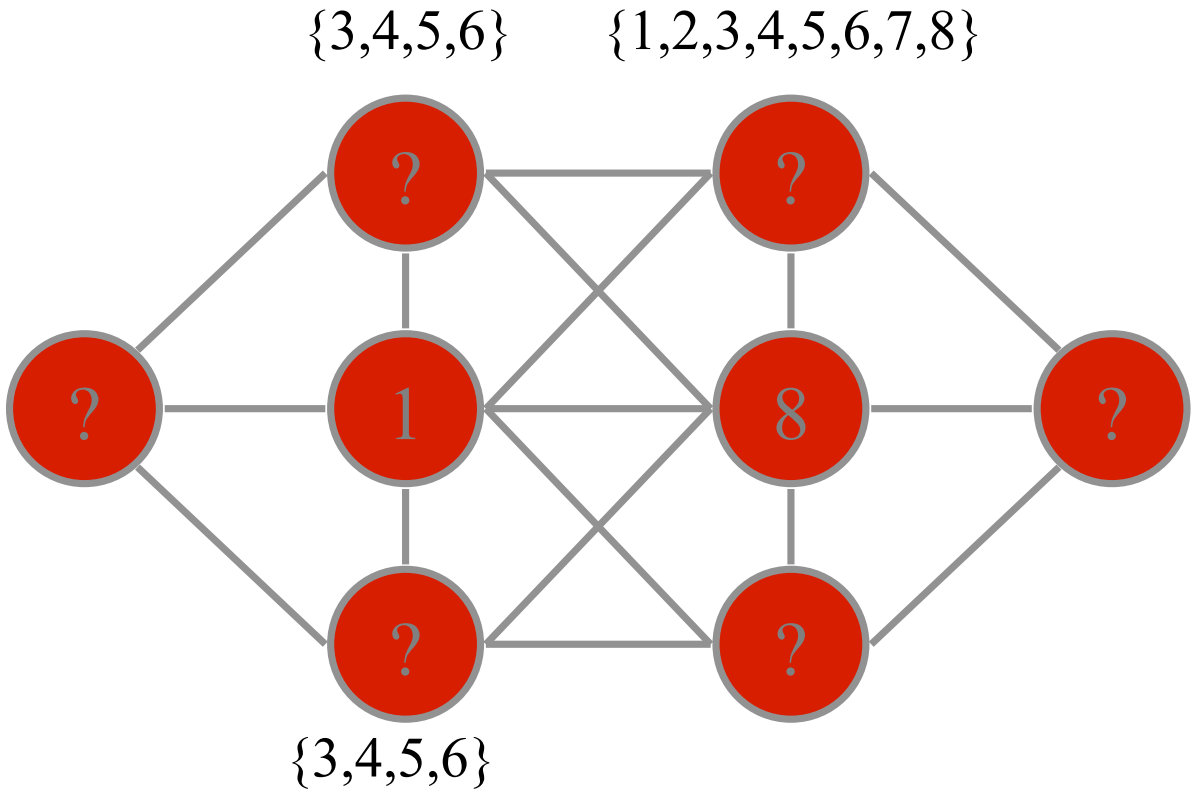


Inference/propagation

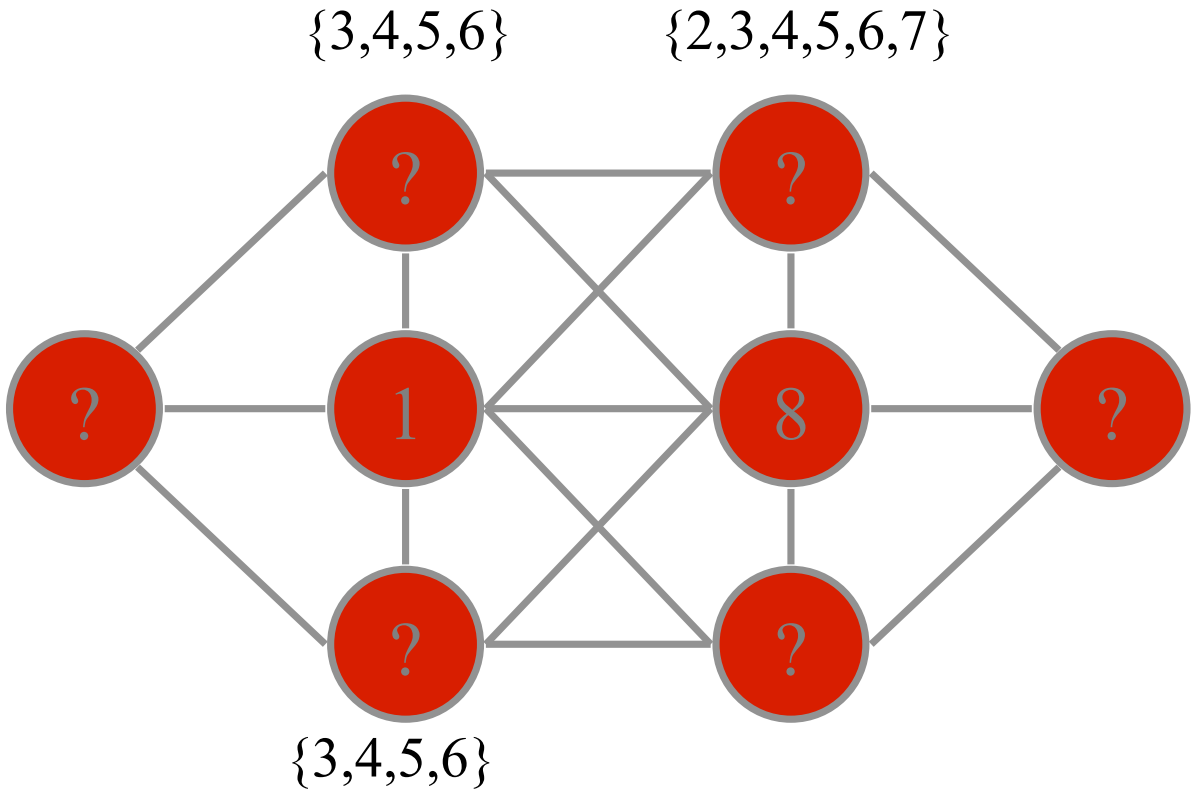


By symmetry

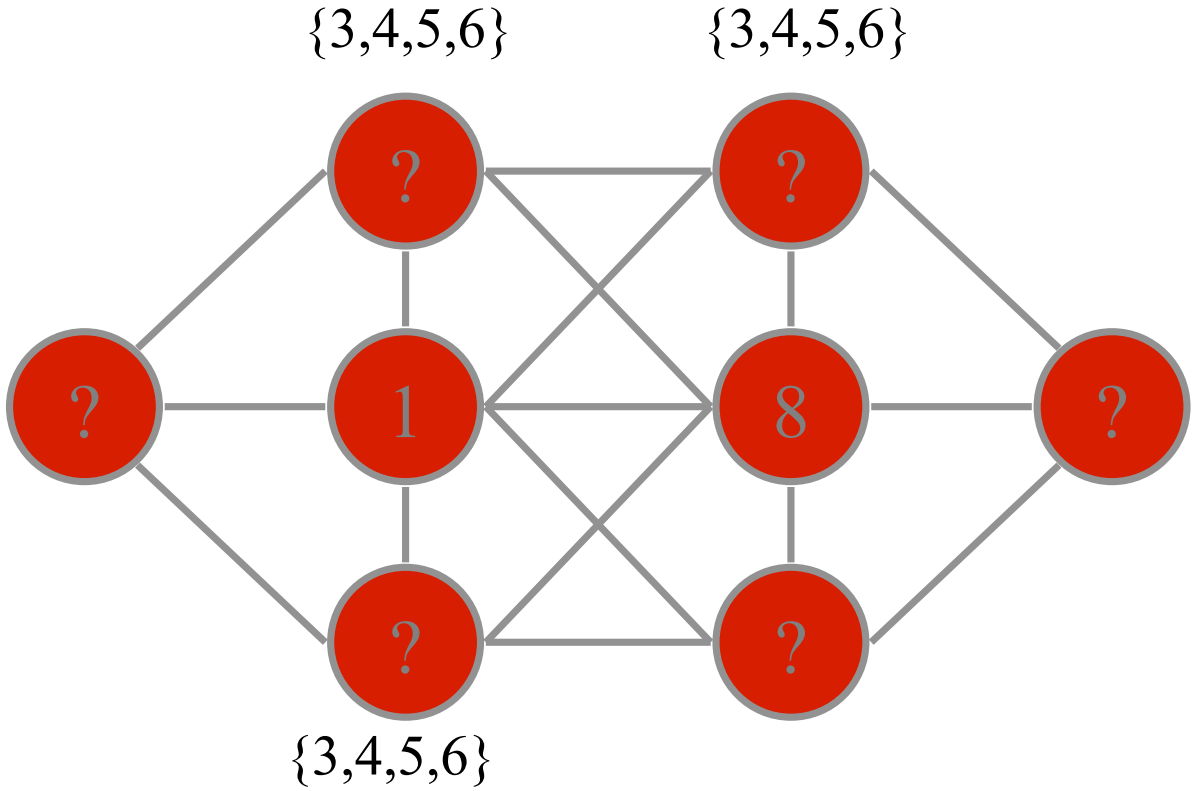
Inference/propagation



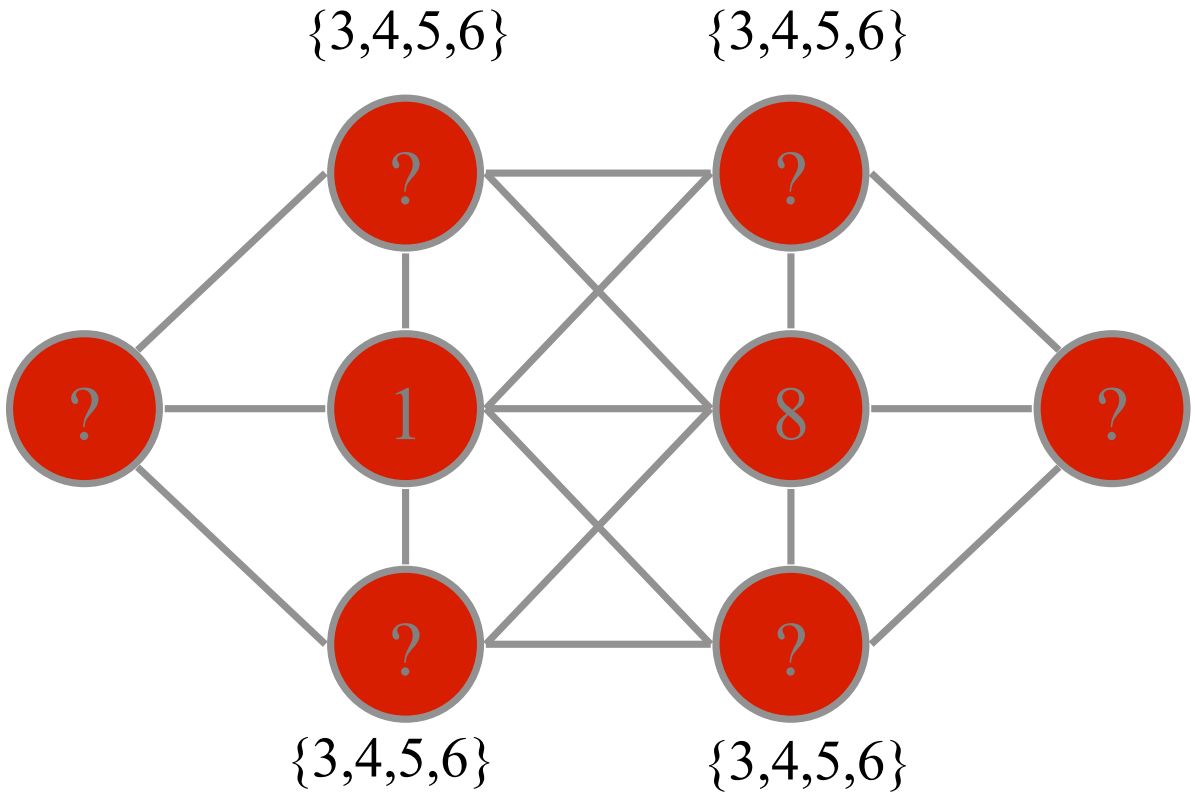
Inference/propagation



Inference/propagation

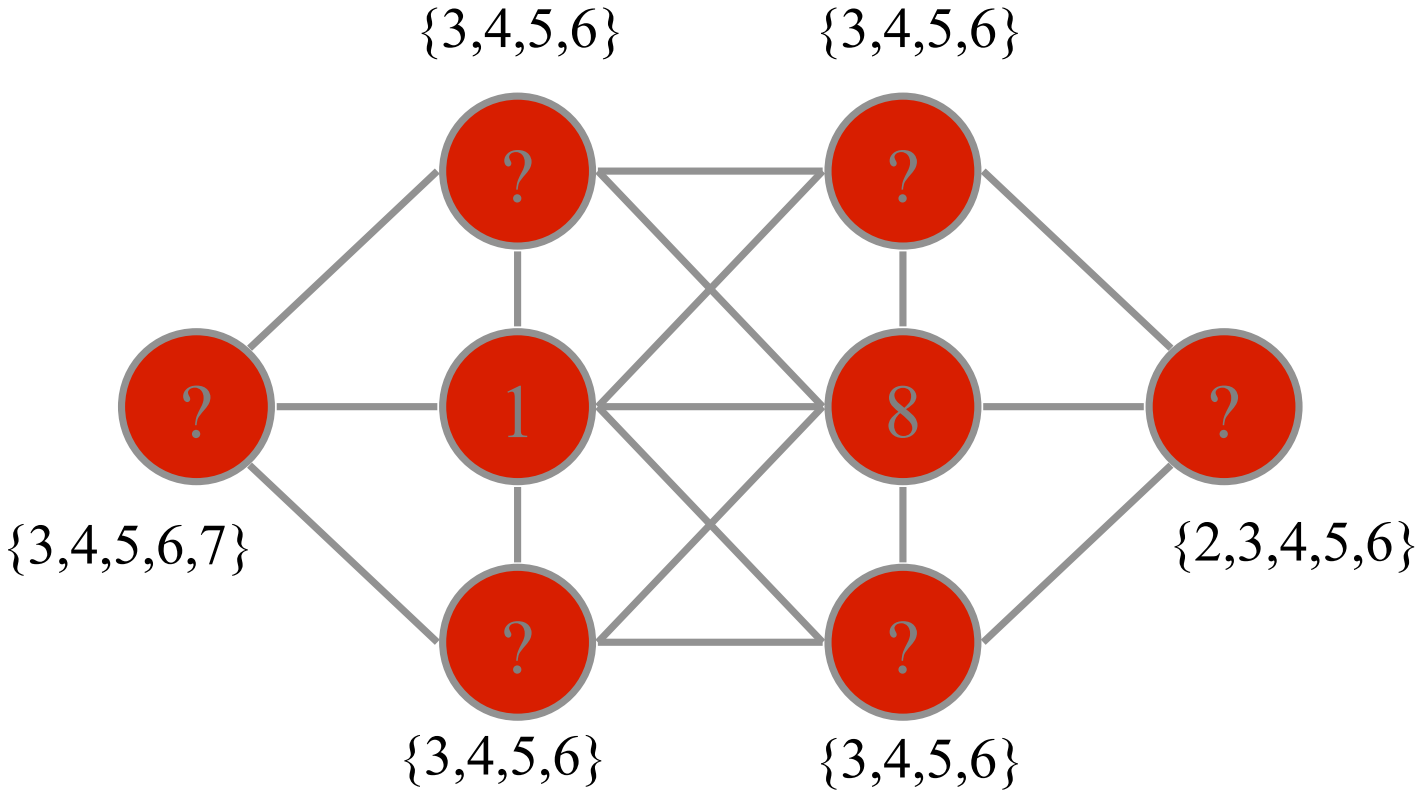


Inference/propagation

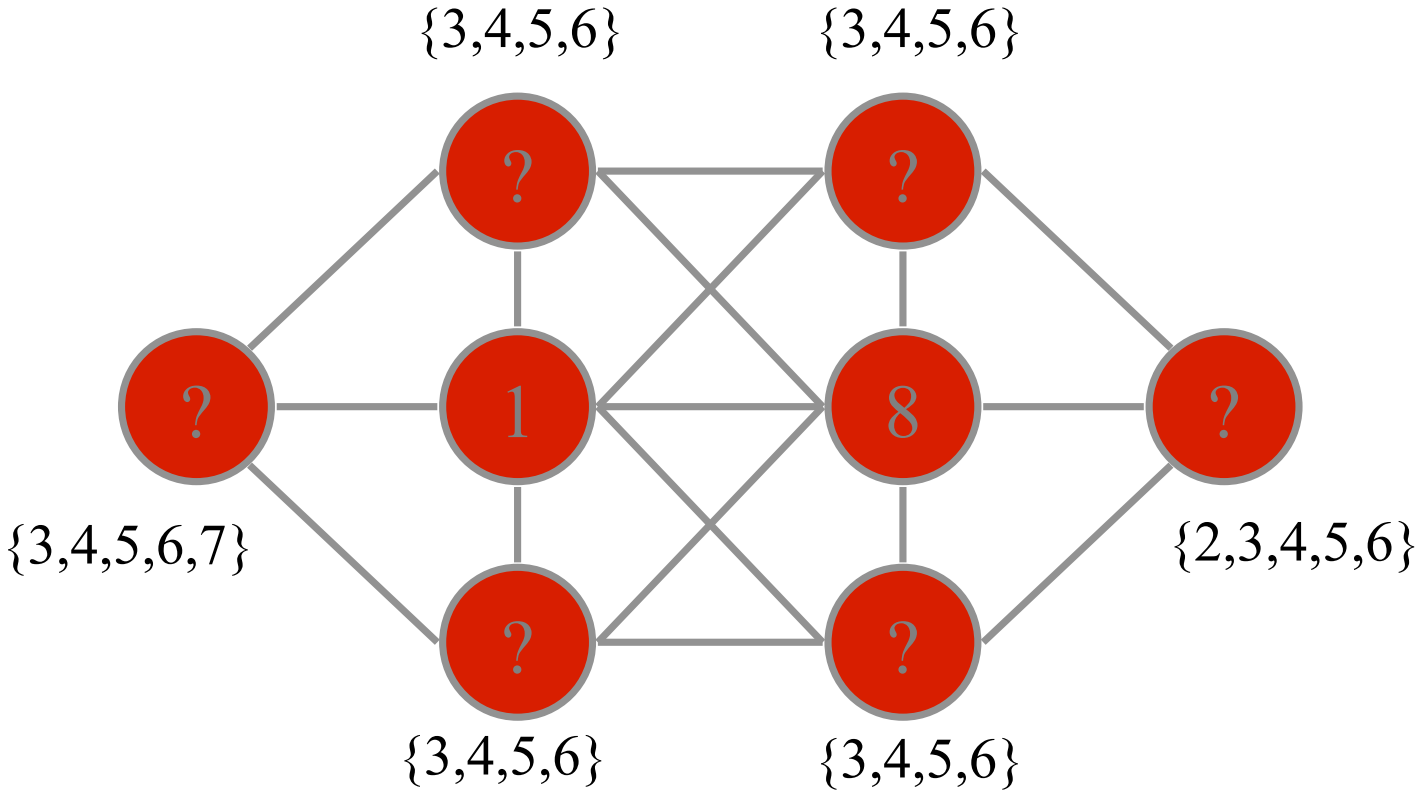


By symmetry

Inference/propagation

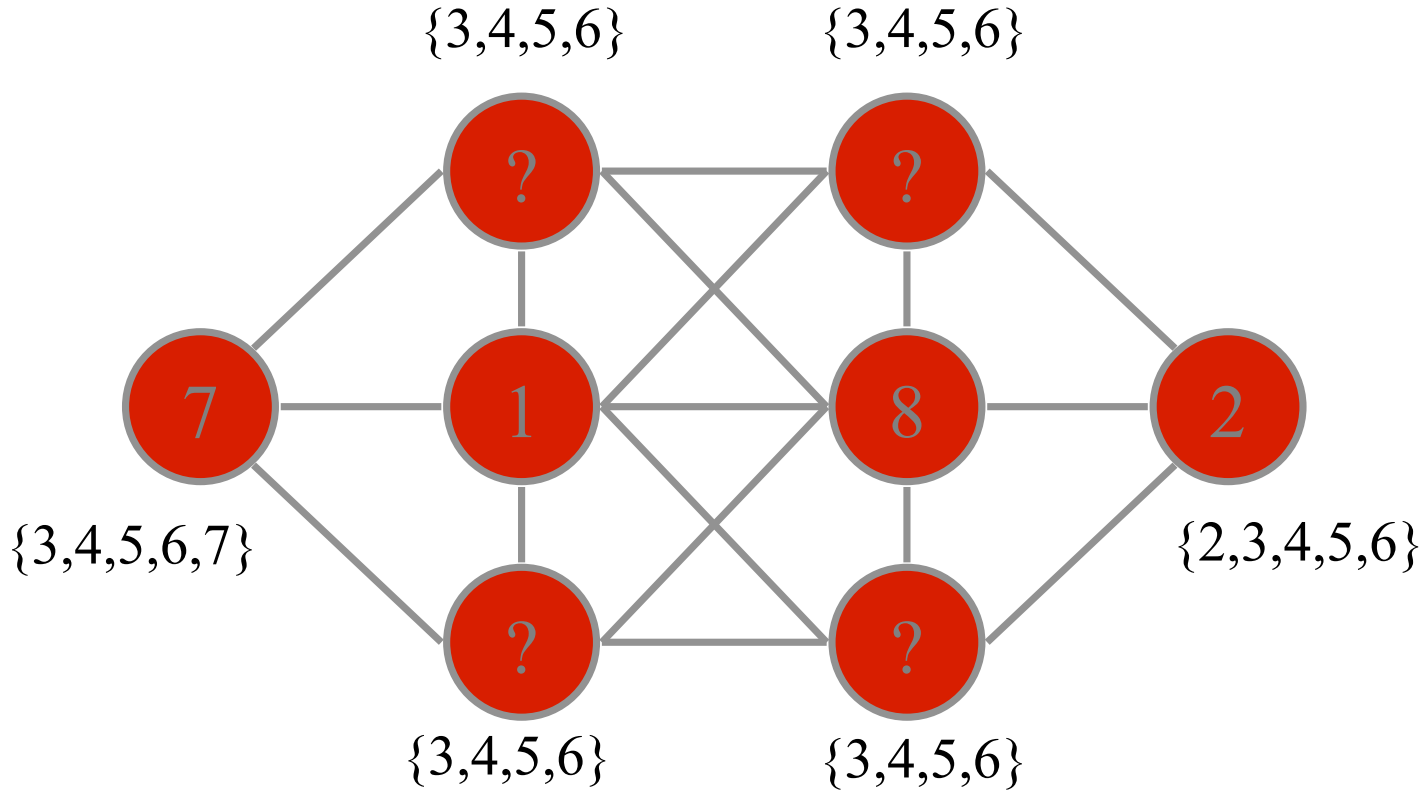


Inference/propagation



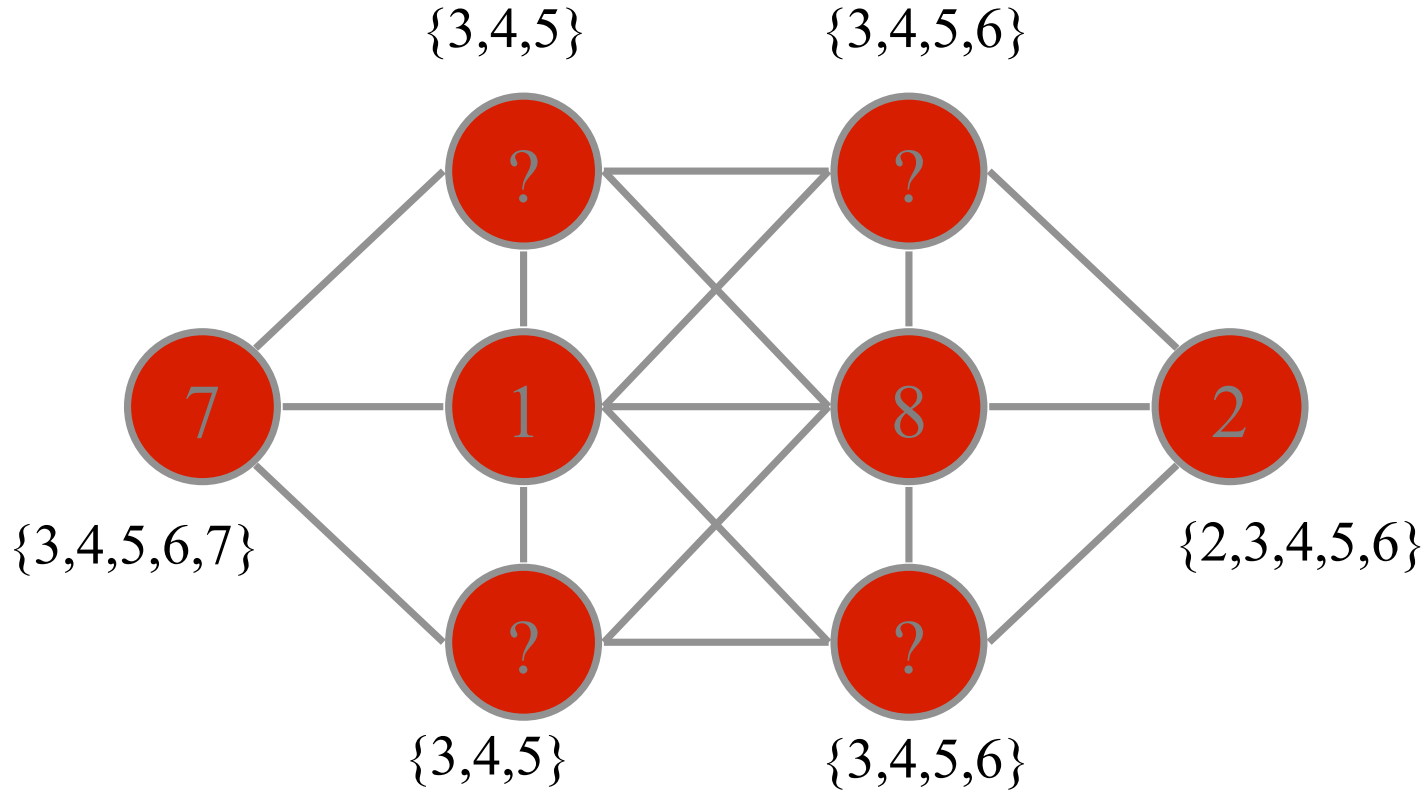
Value 2 and 7 are left in just one variable domain each

Inference/propagation



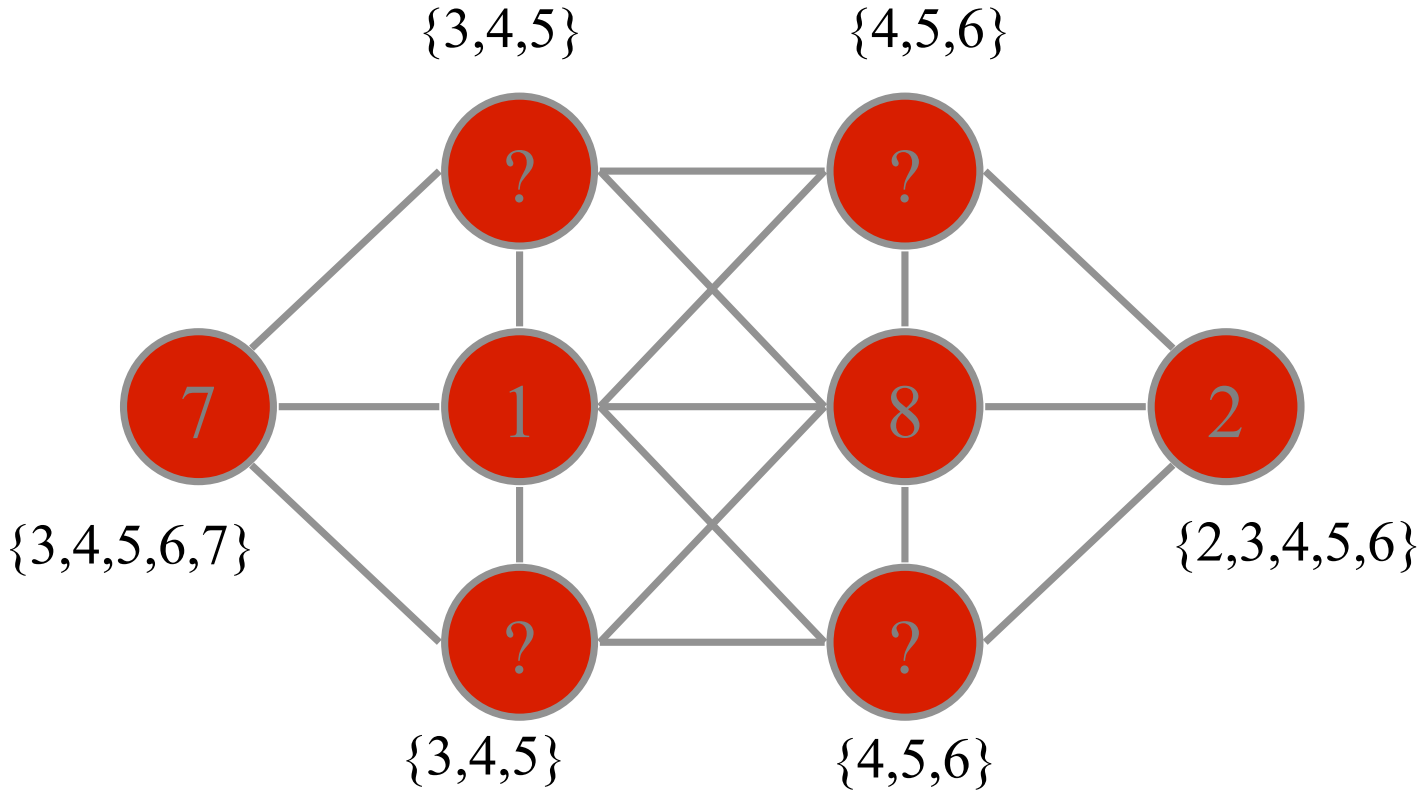
And propagate …

Inference/propagation



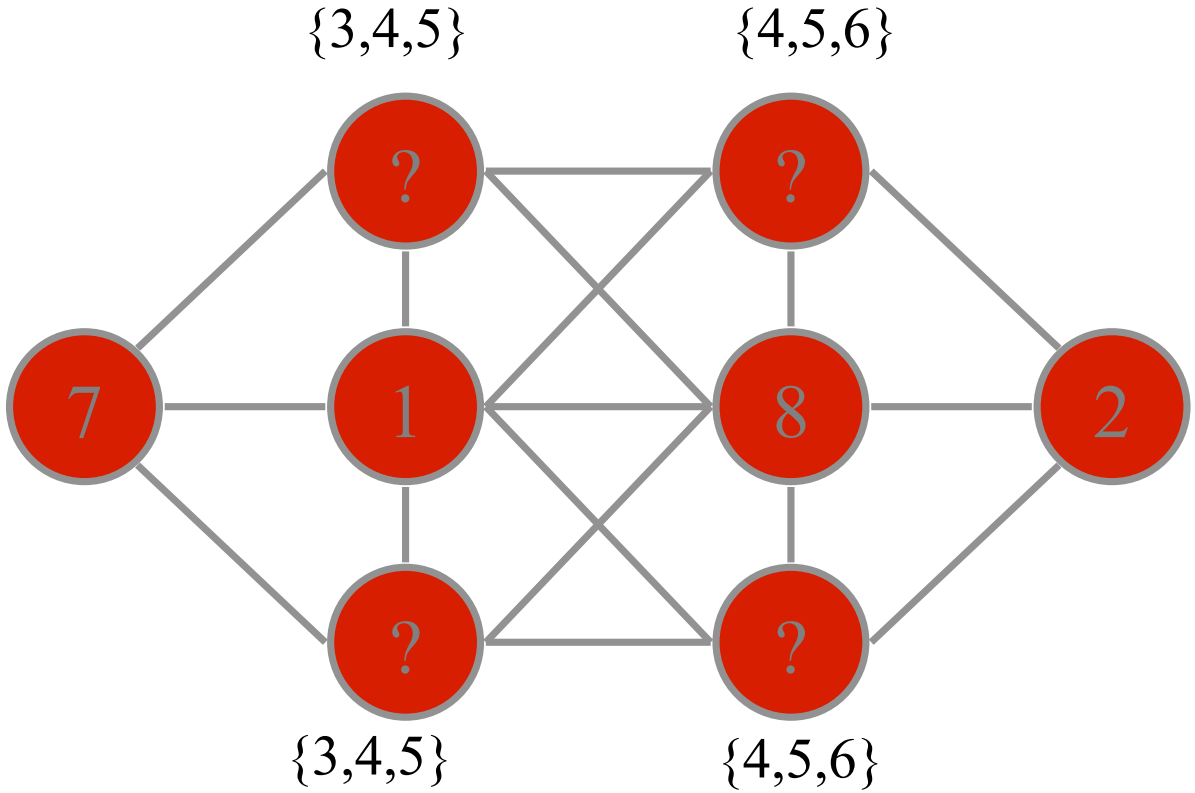
And propagate …

Inference/propagation



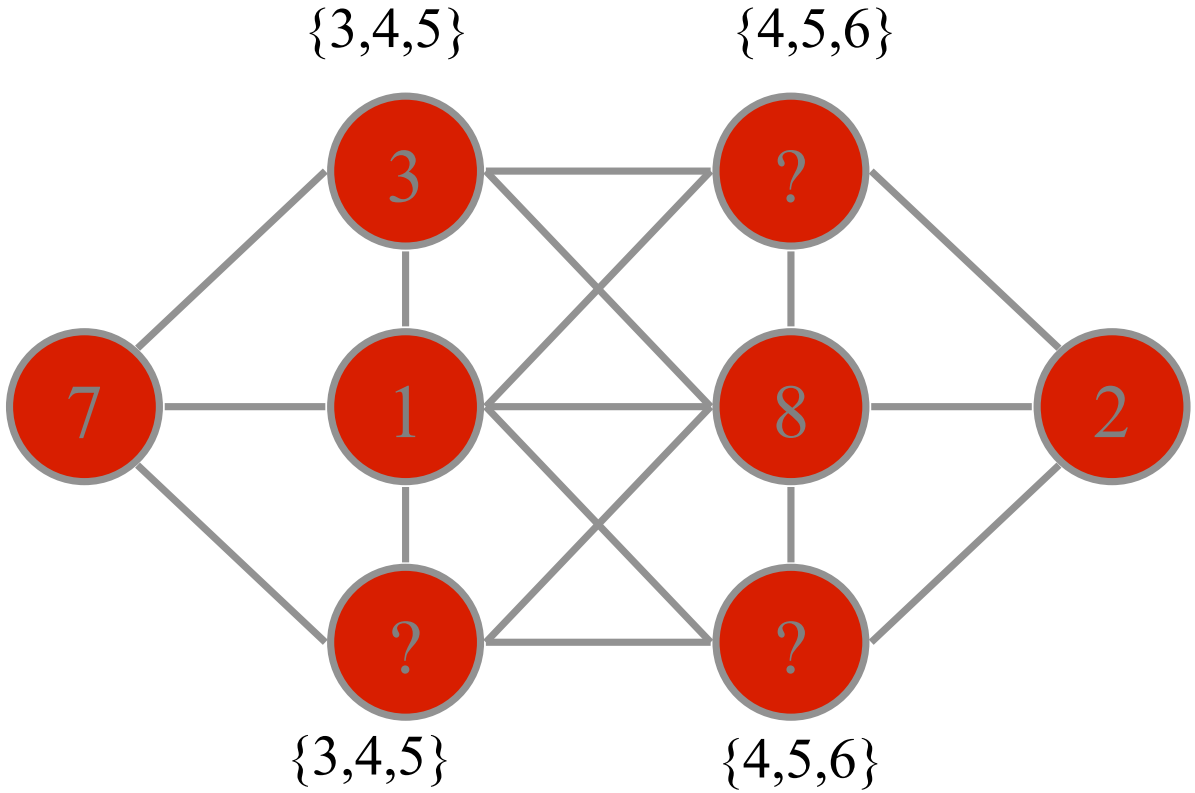
And propagate …

Inference/propagation



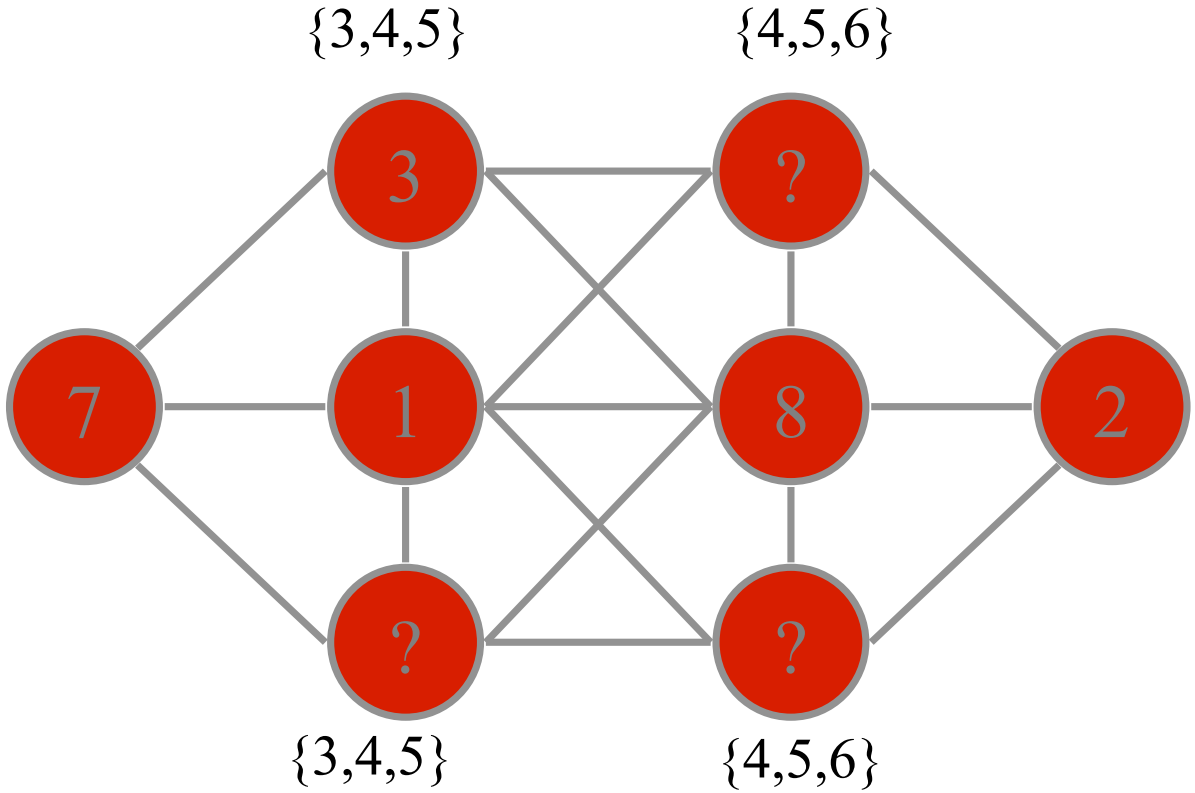
Guess a value, but be prepared to backtrack …

Inference/propagation



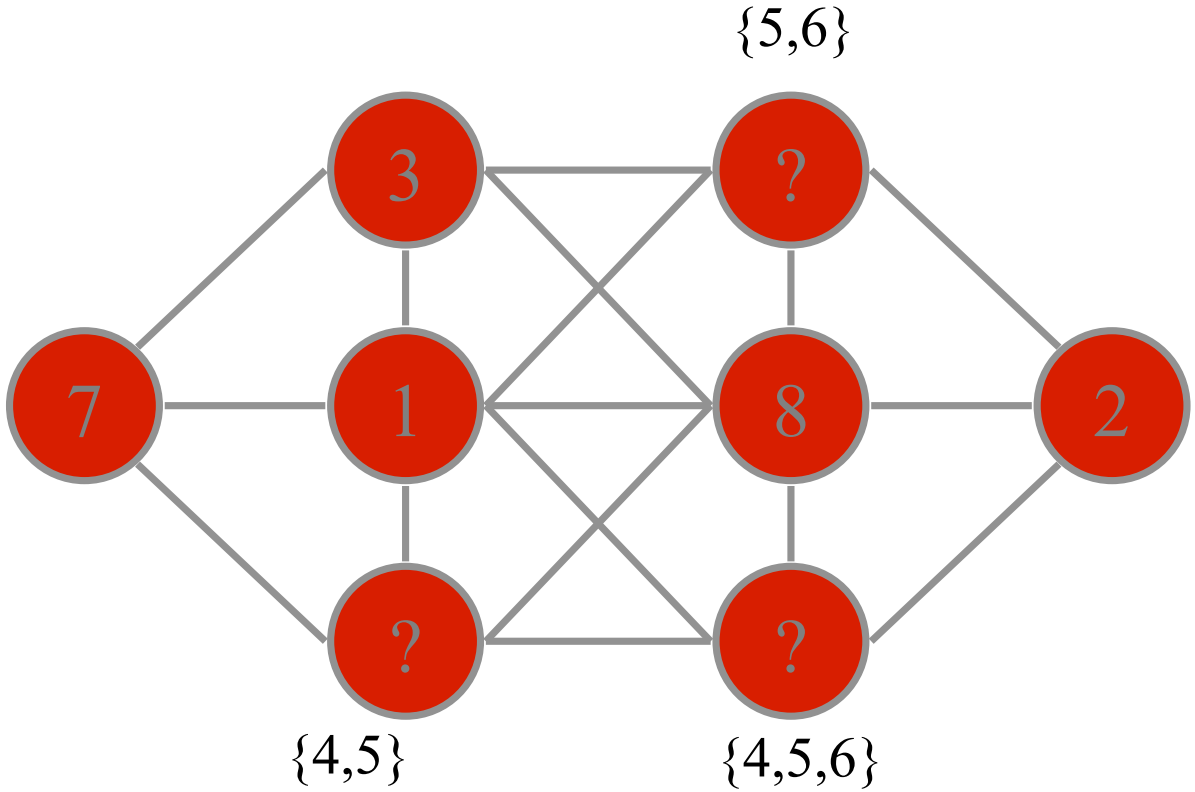
Guess a value, but be prepared to backtrack …

Inference/propagation



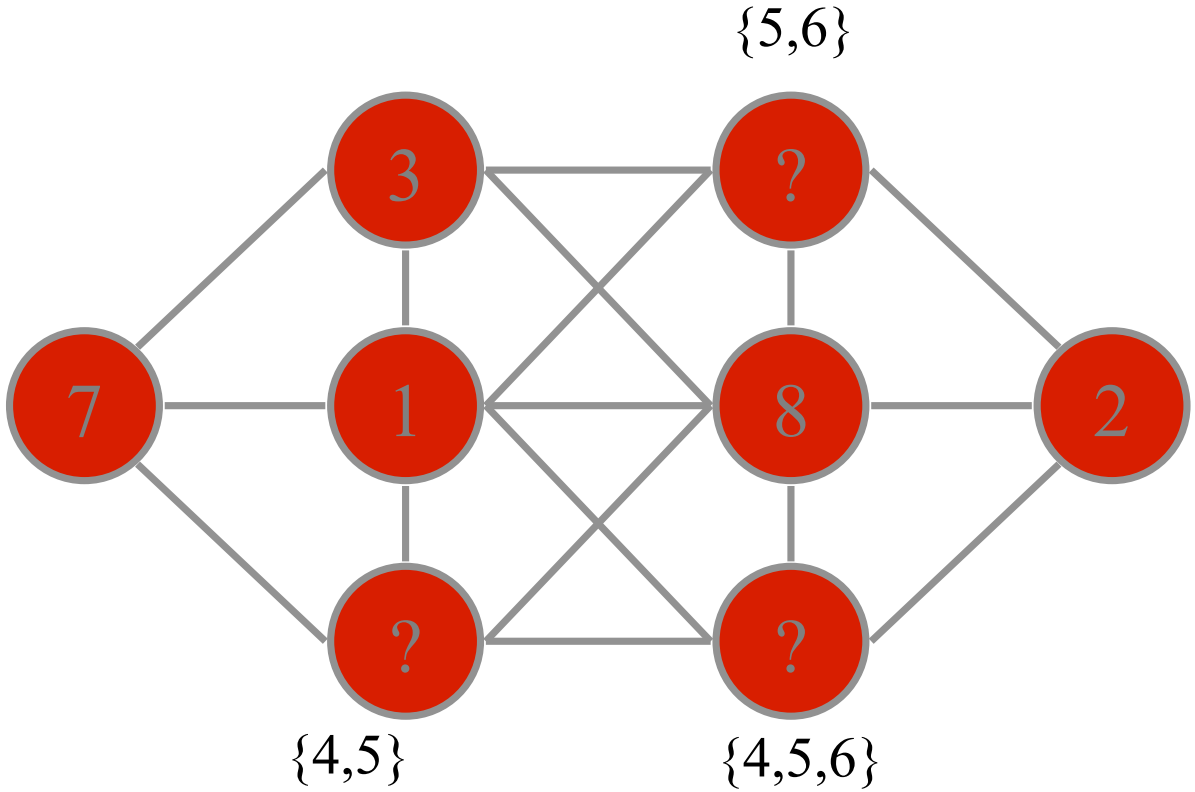
And propagate …

Inference/propagation



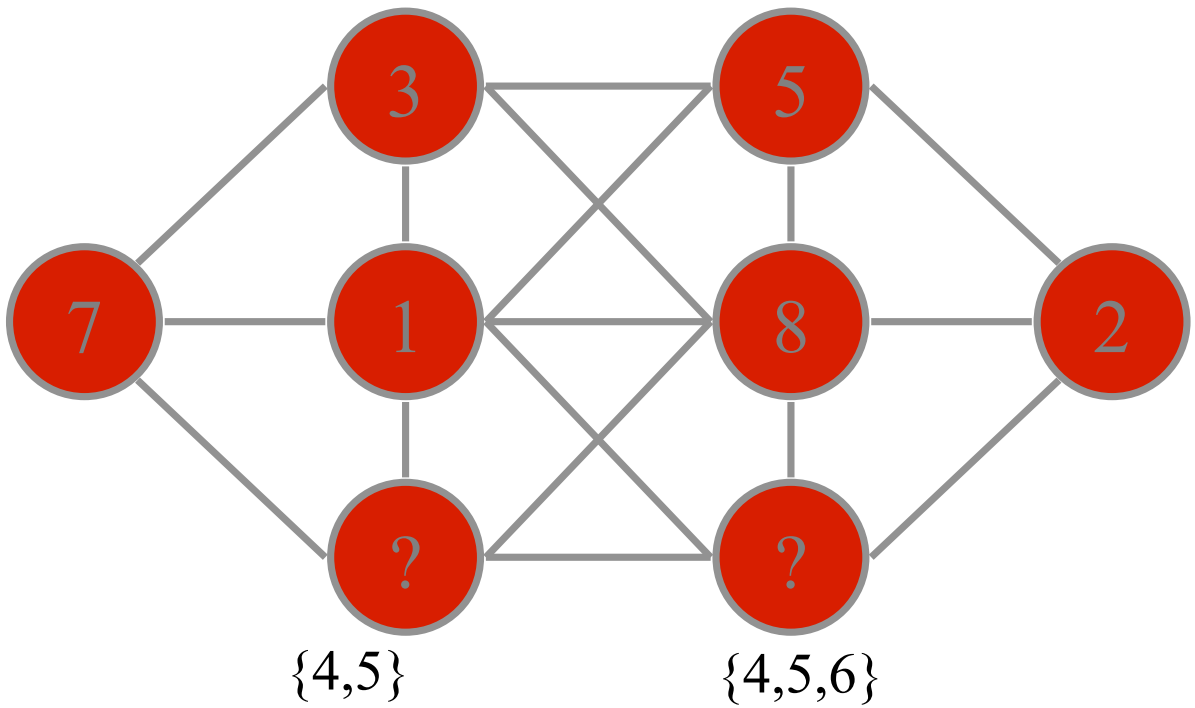
And propagate …

Inference/propagation



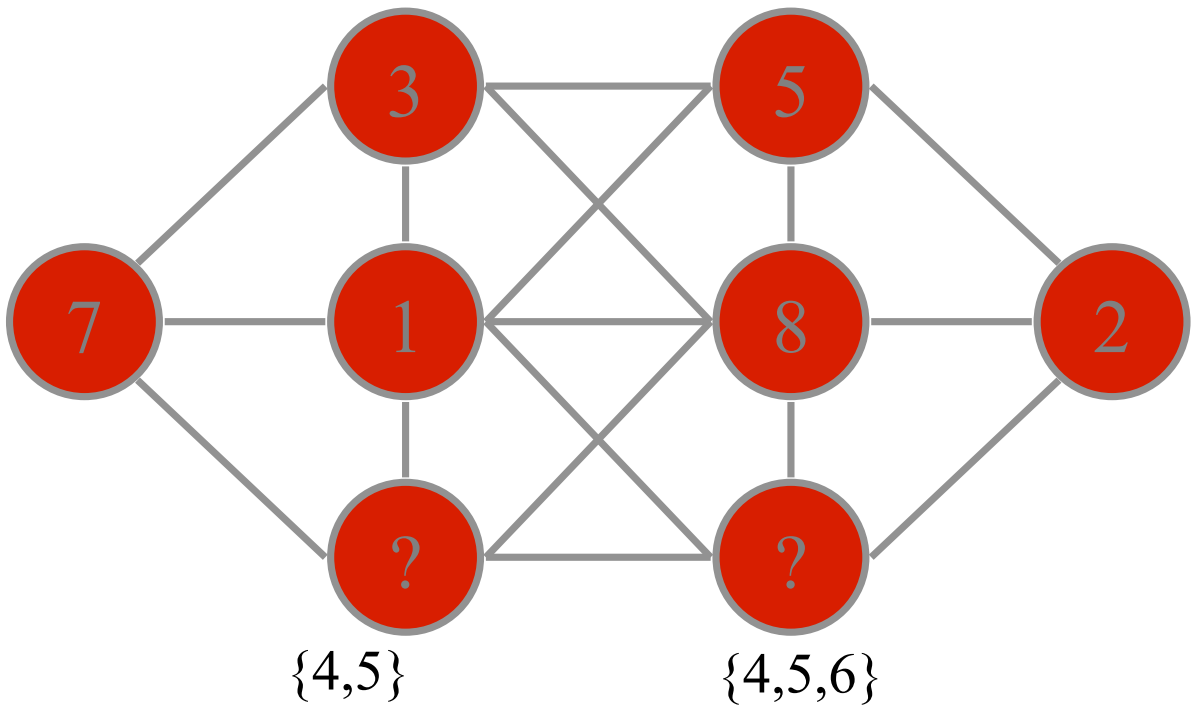
Guess another value …

Inference/propagation



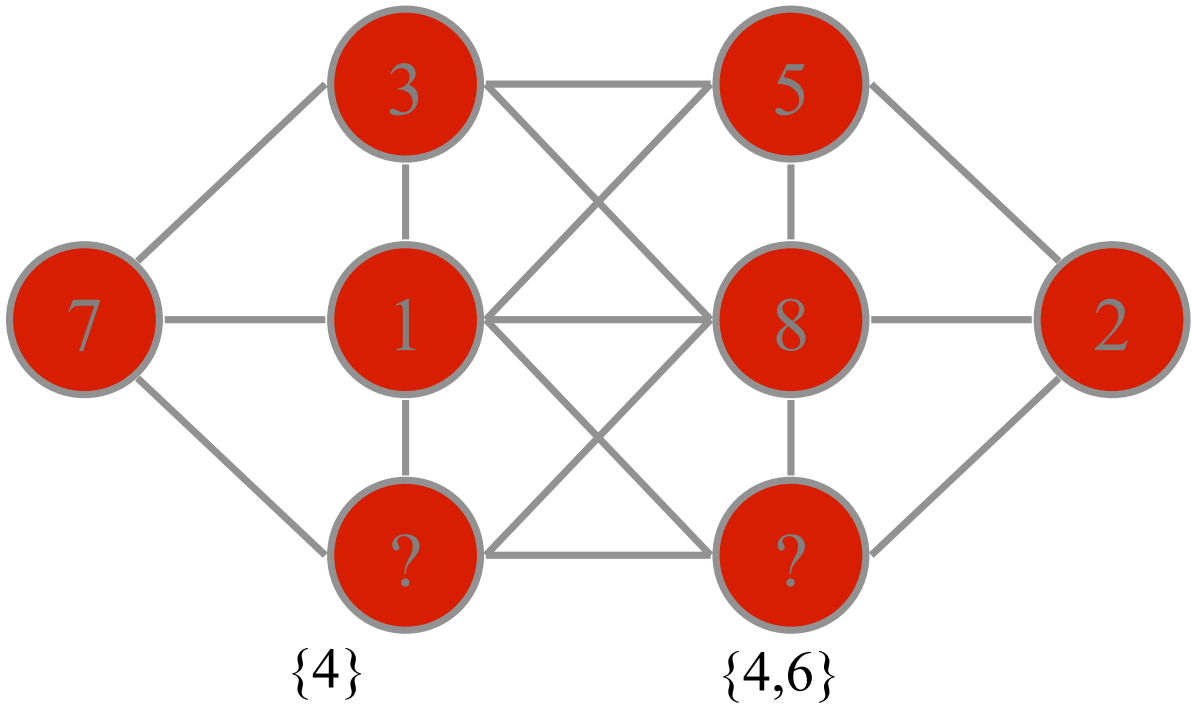
Guess another value …

Inference/propagation



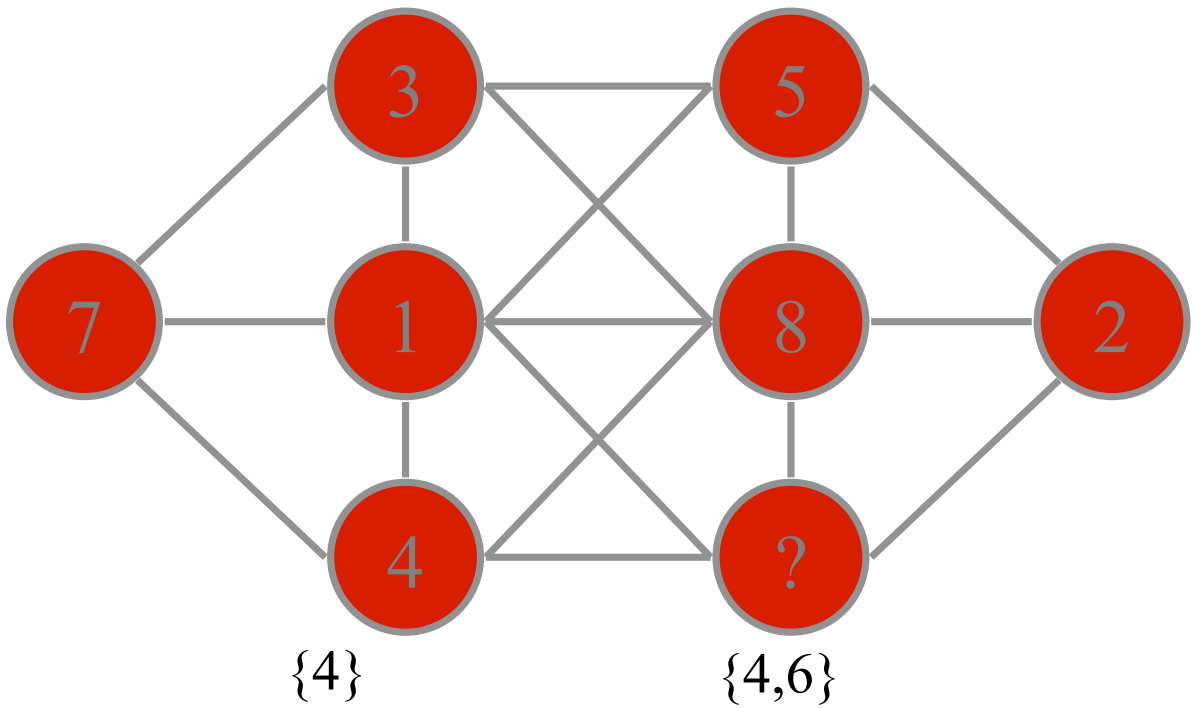
And propagate …

Inference/propagation



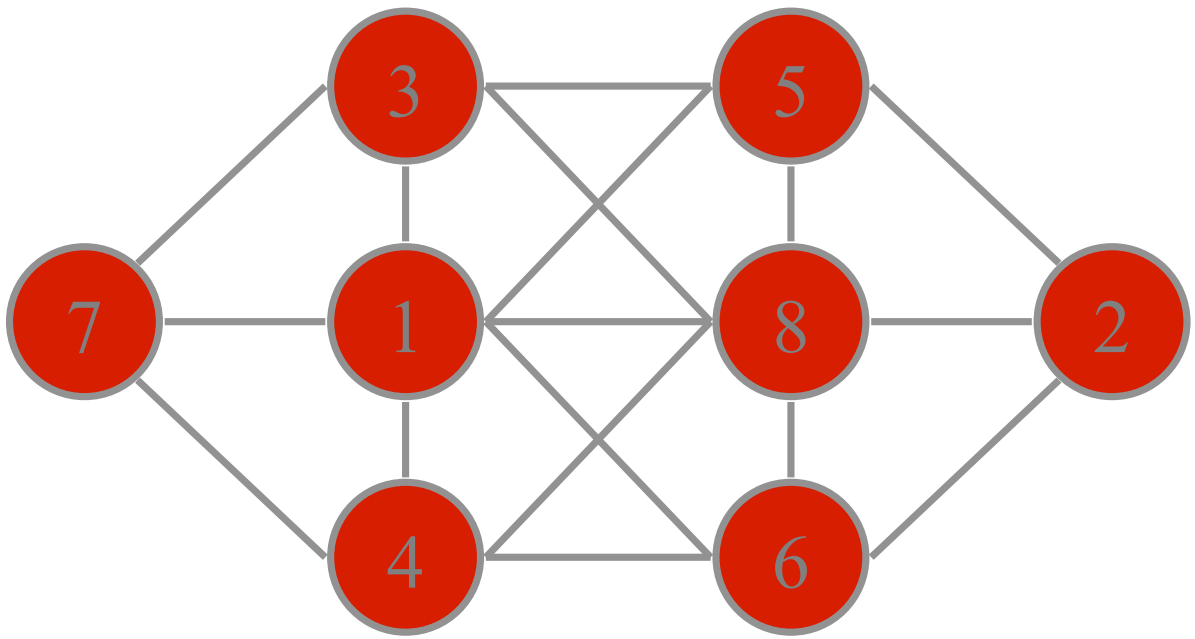
And propagate …

Inference/propagation



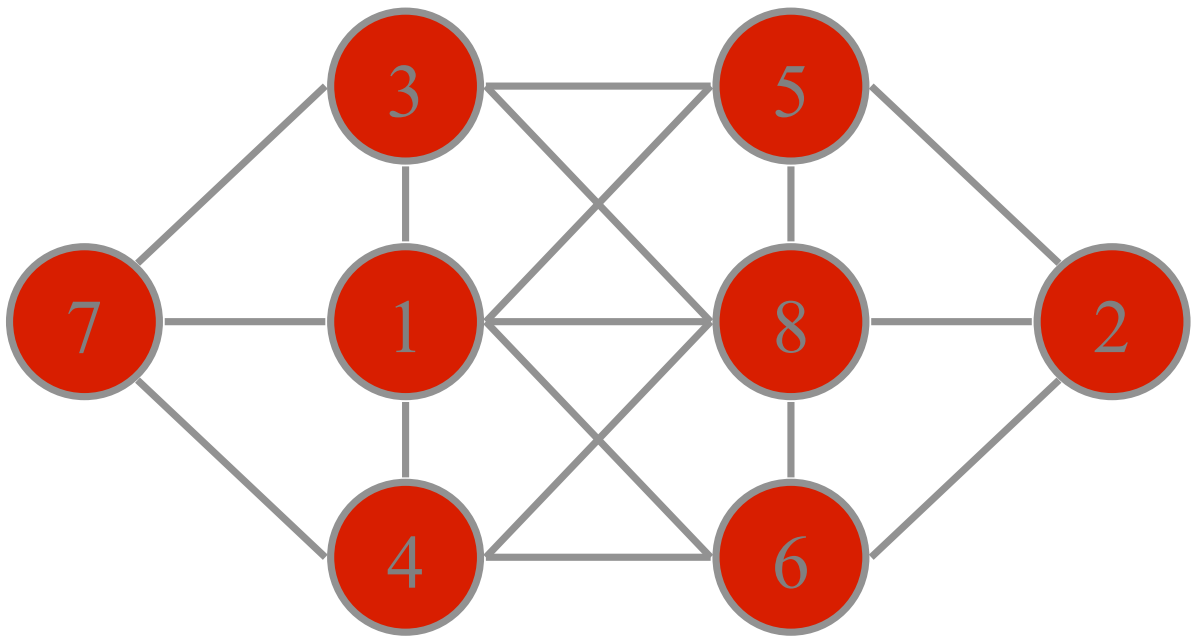
One node has only a single value left …

Inference/propagation



{6}

Solution

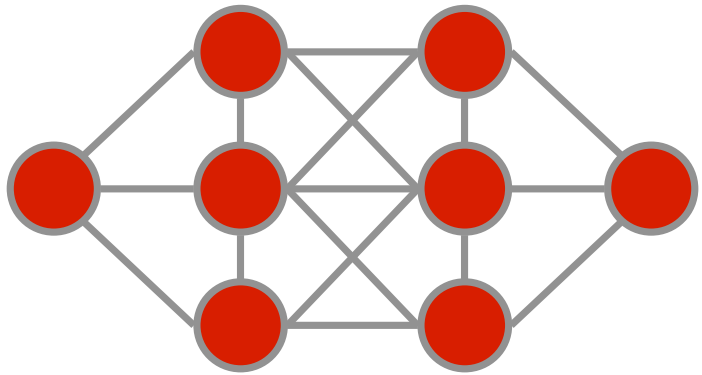


The Core of Constraint Computation

- Modelling
  - Deciding on variables/domains/constraints
- Heuristic Search
- Inference/Propagation
- Symmetry
- Backtracking

Hardness

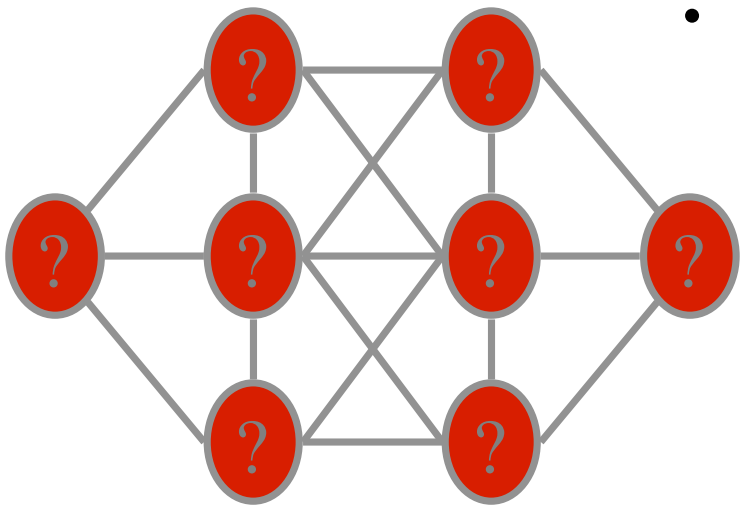
- The puzzle is actually a hard problem
  - NP-complete



Constraint programming

- Model problem by specifying constraints on acceptable solutions
  - define variables and domains
  - post constraints on these variables
- Solve model
  - choose algorithm
    - incremental assignment / backtracking search
    - complete assignments / stochastic search
  - design heuristics

Example CSP



- Variable, v_i for each node
- Domain of {1, …, 8}
- Constraints
  - All values used

    allDifferent(v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8)

  - No consecutive numbers for adjoining nodes

    $|v_1 - v_2| > 1$

    $|v_1 - v_3| > 1$

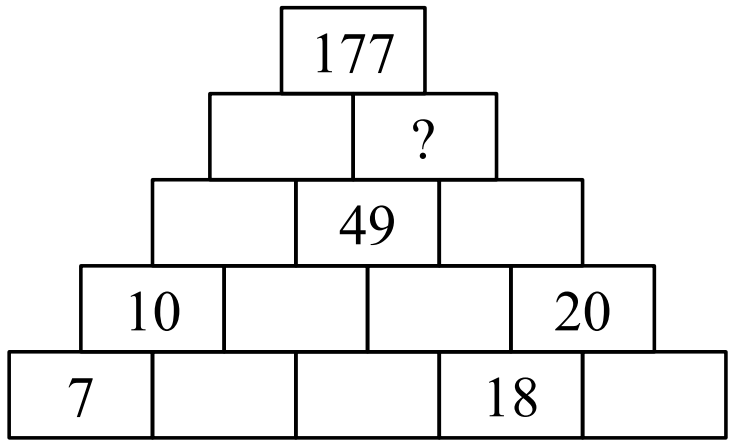
    …

more examples?

Do you know any constraint satisfaction problems?

To a man with a hammer, everything looks like a nail.

Scotsman 4/12/2003



In the pyramid above, two adjacent bricks added together give the value of the brick above. Find the value for the brick marked ?

Crossword puzzle generation

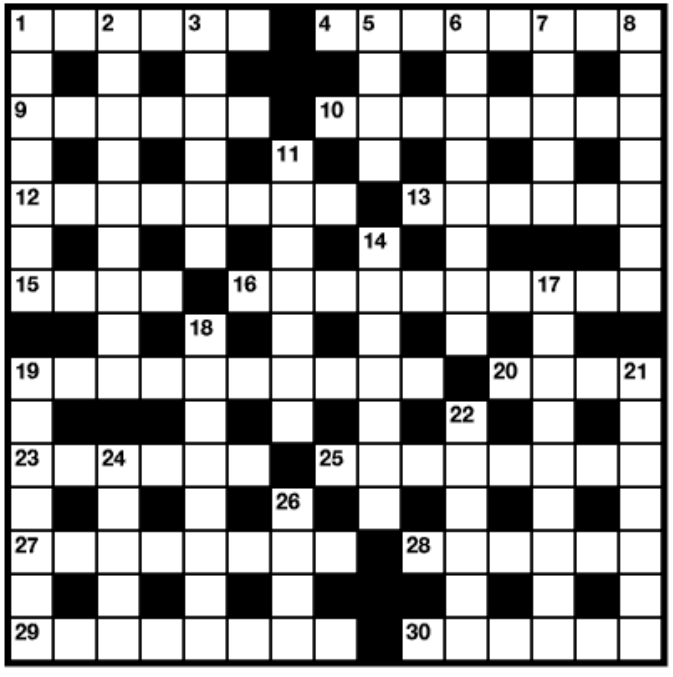
Printer Friendly Format - - Windows Internet Explorer

http://www.heraldandtimeslabs.com/dev/crosswords/print-crossword.php

File Edit View Favorites Tools Help

Page Tools

heraldscotland



Monday, 28 Sep 2009

Across

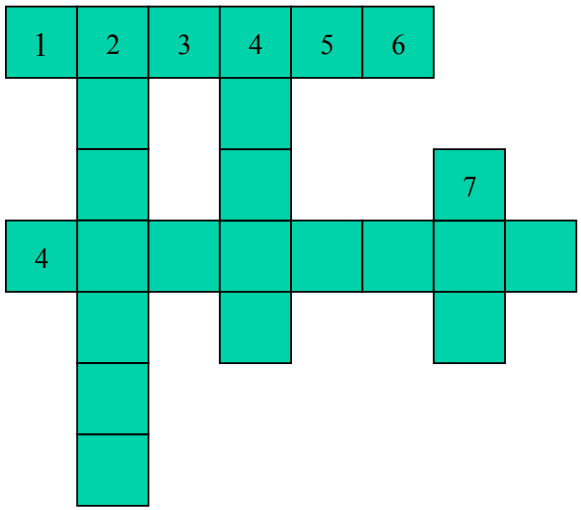
1 Mass-produced legislation? [3,3]

Down

1 Not subject put on in advance [7]

start slides 3 Internet Explo... Microsoft PowerP... 10:39

An example

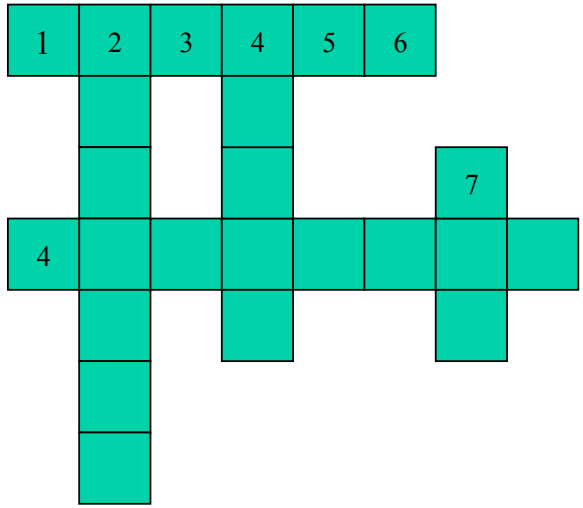


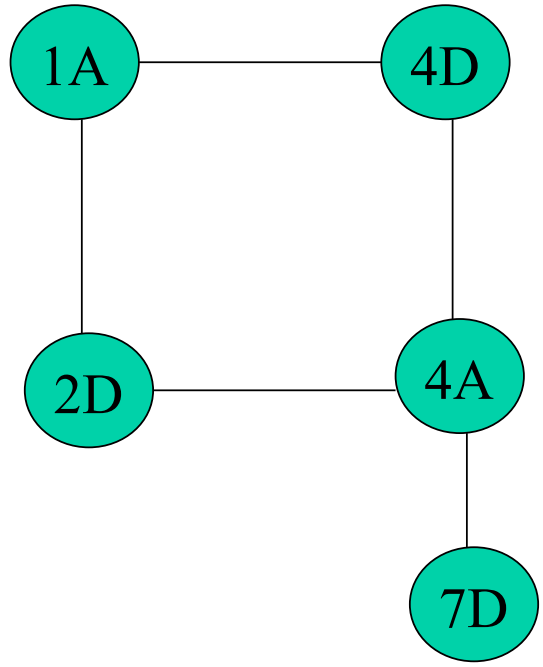
Make a crossword puzzle!

Given the above grid and a dictionary, fill it.

Then go get the clues (not my problem)

An example

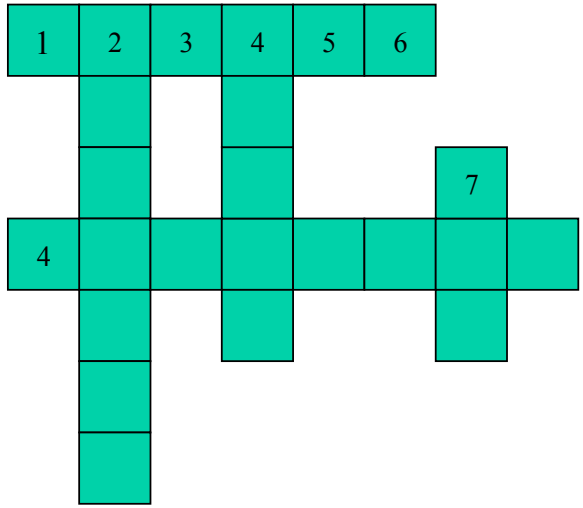


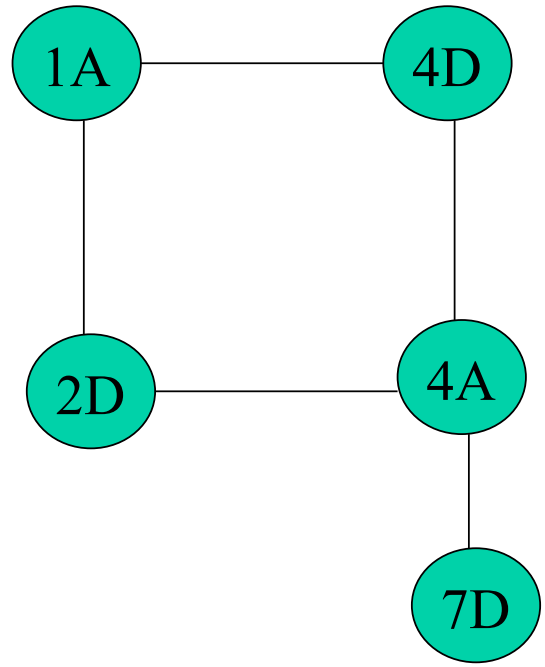


1A	1 across
4D	4 down
2D	2 down
4A	4 across
7D	7 down

Variables

An example





1A-4D: 4th of 1A equals 1st of 4D

1A-2D: 2nd of 1A equals 1st of 2D

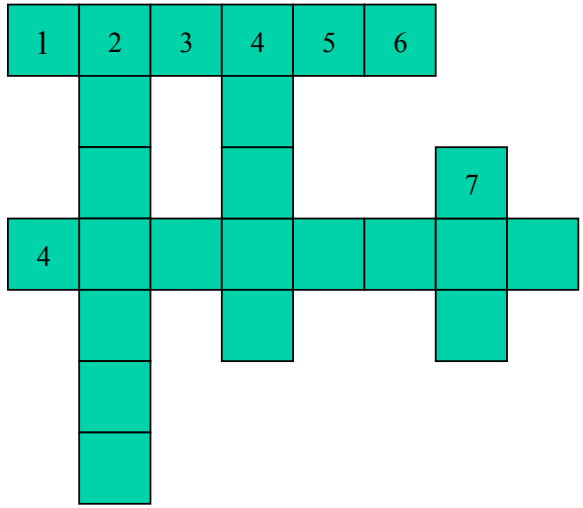
2D-4A: 4th of 2D equals 2nd of4D

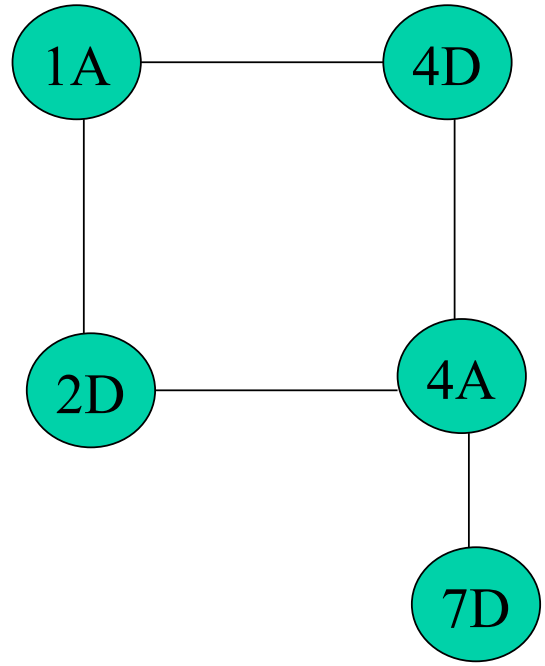
4D-4A: 4th of 4A equals 4th of 4D

4A-7D: 7th of 4A equals 2nd of 7D

Constraints

An example





1A: any 6 letter word

4A: any 8 letter word

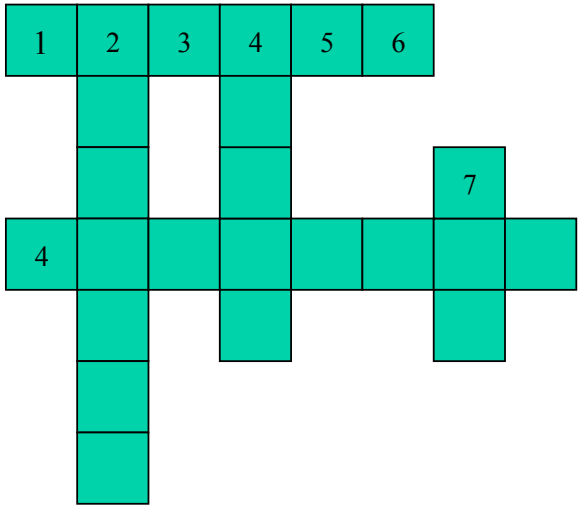
4D: any 5 letter word

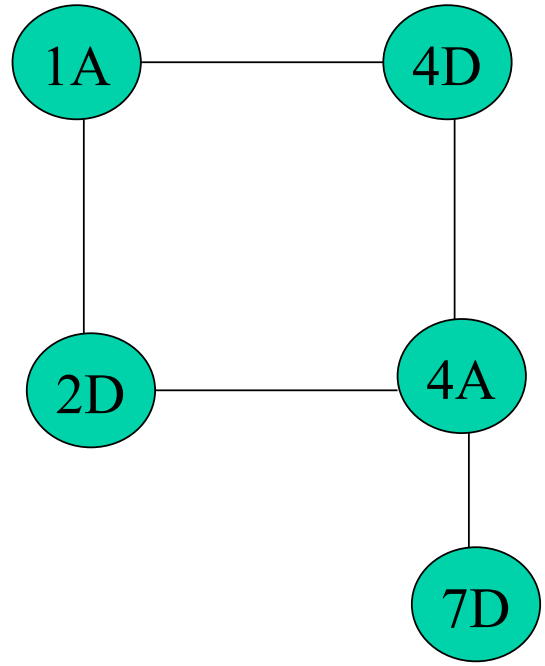
2D: any 7 letter word

7D: any 3 letter word

Domains (also unary constraints!)

An example

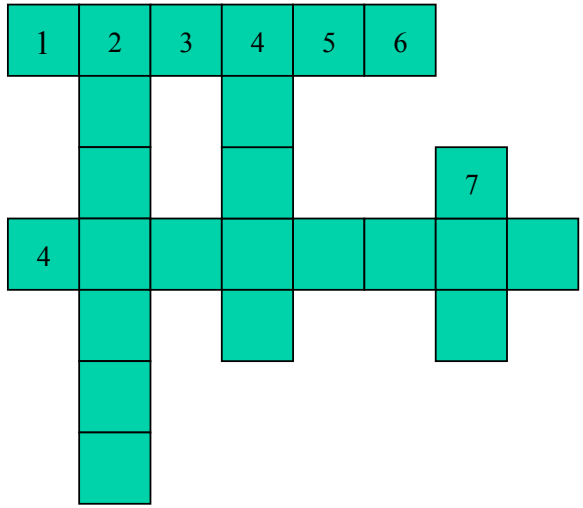


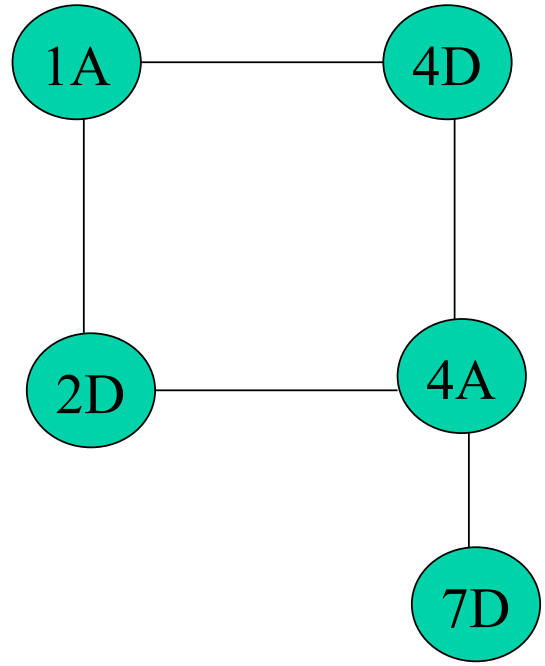


Find an assignment of values to variables, from their domains, such that the constraints are satisfied (or show that no assignment exists)

A CSP!

An example





Choose a variable

Assign it a value

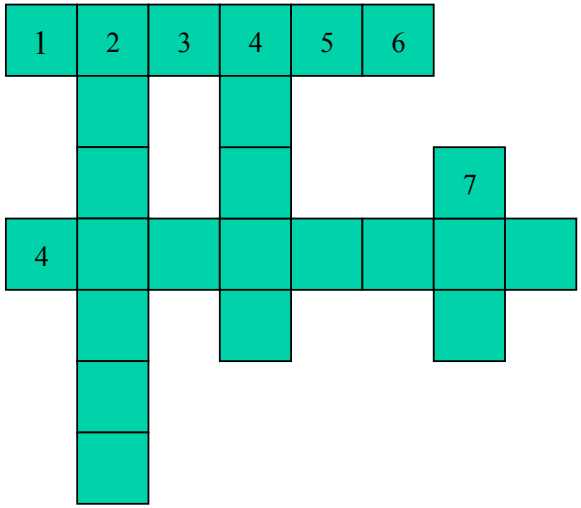
Check compatibility

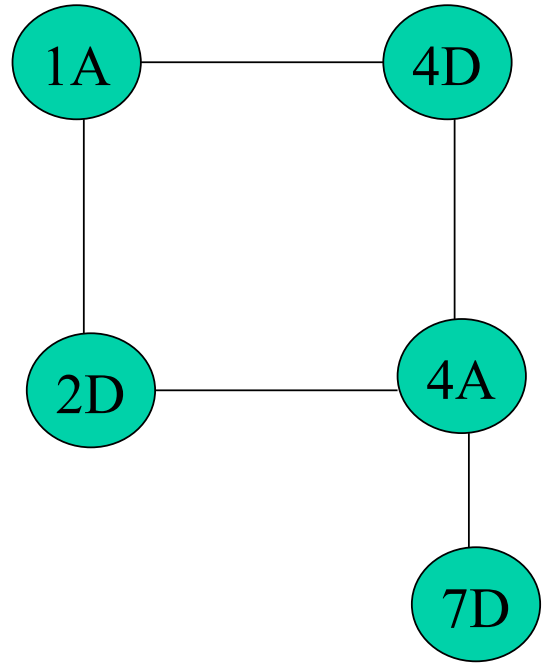
If not compatible try a new value

If no values remain re-assign previous variable

Good old fashioned BT!

Questions?





What variable should I choose?

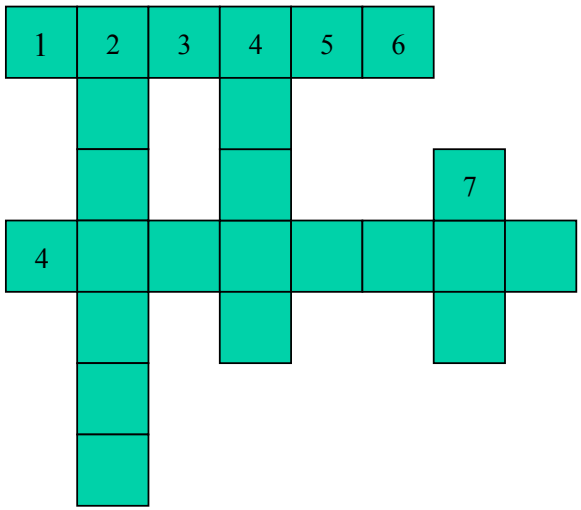
What value should I choose?

What reasoning can I do when making an assignment?

What reasoning can I do on a dead end?

Decisions, decisions!

An example



Is there an alternative representation?

Problems of interest to CP

These are some of the problems that have been tackled by CP

- factory scheduling (JSSP)
- vehicle routing (VRP)
- packing problems (NumPart and BinPack)
- timetabling (exams, lectures, trains)
- configuration and design (hardware)
- workforce management (call centres, etc)
- car sequencing (assembly line scheduling)
- supertree construction (bioinformatics)
- network design (telecoms problem)
- gate arrival (at airports)
- logistics (Desert Storm an example)
- aircraft maintenance schedules
- aircraft crew scheduling (commercial airlines)
- air cover for naval fleet

CP

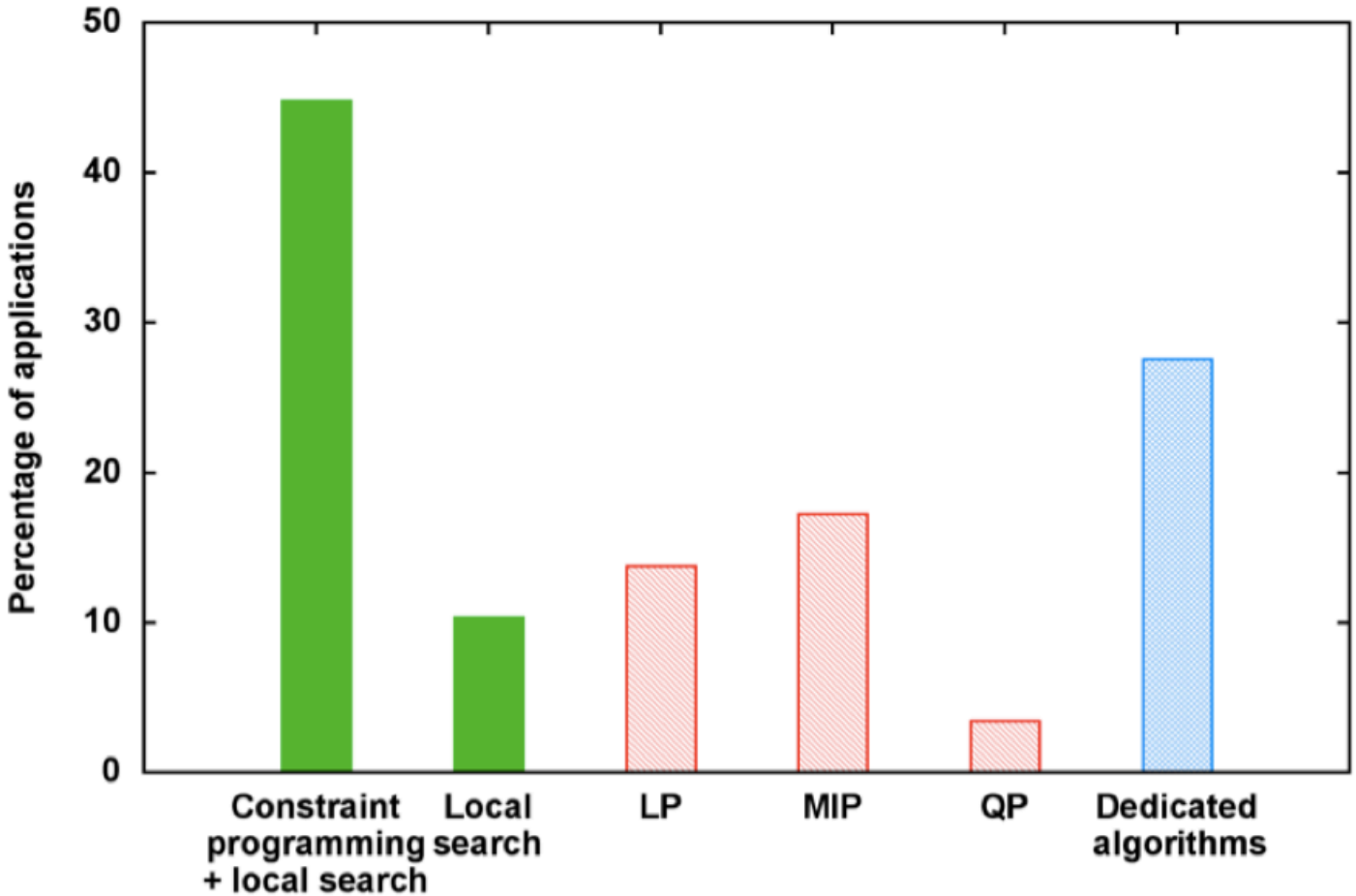
Constraint Programming = model (representation) +
propagation (reasoning, inference) +
search (reasoning, inference)

Applications

- Operation research (optimization problems)
- Graphical interactive systems (to express geometrical correctness)
- Molecular biology (DNA sequencing, 3D models of proteins)
- Finance
- Circuit verification
- Elaboration of natural languages (construction of efficient parsers)
- Scheduling of activities
- Configuration problem in form compilation
- Generation of coerent music programs **?**.
- Data bases
- ...
- http://hsimonis.wordpress.com/

Applications

Distribution of technology used at Google for optimization applications developed by the operations research team



[Slide presented by Laurent Perron on OR-Tools at CP2013]

List of Contents

Outline

Constraint Programming

The **domain** of a variable x, denoted $D(x)$, is a finite set of elements that can be assigned to x.

A **constraint** C on X is a subset of the Cartesian product of the domains of the variables in X, i.e., $C \subseteq D(x_1) \times \cdots \times D(x_k)$. A tuple $(d_1, \ldots, d_k) \in C$ is called a solution to C.
Equivalently, we say that a solution $(d_1, ..., d_k) \in C$ is an assignment of the value d_i to the variable $x_i, \forall 1 \leq i \leq k$, and that this assignment satisfies C.
If $C = \emptyset$, we say that it is inconsistent.

Extensional: specifies the good (or bad) tuples (values)
Intensional: specifies the characteristic function

Constraint Programming

Constraint Satisfaction Problem (CSP)

A CSP is a finite set of variables X, together with a finite set of constraints C, each on a subset of X. A **solution** to a CSP is an assignment of a value $d \in D(x)$ to each $x \in X$, such that all constraints are satisfied simultaneously.

Constraint Optimization Problem (COP)

A COP is a CSP P defined on the variables $x_1, \ldots, x_n$, together with an objective function $f : D(x_1) \times \cdots \times D(x_n) \rightarrow Q$ that assigns a value to each assignment of values to the variables. An **optimal solution** to a minimization (maximization) COP is a solution d to P that minimizes (maximizes) the value of $f(d)$.

Task:

- determine whether the CSP/COP is consistent (has a solution):
- find one solution
- find all solutions
- find one optimal solution
- find all optimal solutions

Solving CSPs

- Systematic search:
  - choose a variable x_i that is not yet assigned
  - create a choice point, i.e. a set of mutually exclusive & exhaustive choices, e.g. $x_i = v$ vs $x_i \neq v$
  - try the first & backtrack to try the other if this fails
- Constraint propagation:
  - add $x_i = v$ or $x \neq v$ to the set of constraints
  - re-establish local consistency on each constraint
    $\rightsquigarrow$ remove values from the domains of future variables that can no longer be used because of this choice
  - fail if any future variable has no values left

Representing a Problem

- If a CSP $\mathcal{P} =< \mathcal{X}, \mathcal{DE}, \mathcal{C} >$ represents a problem P, then every solution of $\mathcal{P}$ corresponds to a solution of P and every solution of P can be derived from at least one solution of $\mathcal{P}$
- More than one solution of P can be represented by the same solution of $\mathcal{P}$, if modelling removes symmetry
- The variables and values of $\mathcal{P}$ represent entities in P
- The constraints of $\mathcal{P}$ ensure the correspondence between solutions
- The aim is to find a model $\mathcal{P}$ that can be solved as quickly as possible (Note that shortest run-time might not mean least search!)

Interactions with Search Strategy

Whether a model is better than another can depend on the search algorithm and search heuristics

- Let's assume that the search algorithm is fixed
  although different level of consistency can also play a role
- Let's also assume that choice points are always $x_i = v$ vs $x_i \neq v$
- Variable (and value) order still interact with the model a lot
- Is variable & value ordering part of modelling?

  In practice it is.
  but it depends on the modeling language used

Global Constraint: `alldifferent`

Global constraint:

set of more elementary constraints that exhibit a special structure when considered together.

`alldifferent` constraint

Let $x_1, x_2, \ldots, x_n$ be variables. Then:

$$\texttt{alldifferent}(x_1, ..., x_n) = \{(d_1, ..., d_n) \mid \forall i, d_i \in D(x_i), \quad \forall i \neq j,\ d_i \neq d_j\}.$$

Constraint arity: number of variables involved in the constraint

Note: different notation and names used in the literature

Global Constraint Catalog

http://www.emn.fr/z-info/sdemasse/gccat/sec5.html

Global Constraint Catalog

Corresponding author: Nicolas Beldiceanu nicolas.beldiceanu@emn.fr

Online version: Sophie Demassey sophie.demassey@emn.fr

Google Search ○ Web ◉ Catalog
○ all formats ◉ html ○ pdf

Global Constraint Catalog
html / 2009-12-16

Search by:

NAME Keyword Meta-keyword Argument pattern Graph description
Bibliography Index

Keywords (ex: *Assignment*, *Bound consistency*, *Soft constraint*,...) can be searched by **Meta-keywords** (ex: *Application area*, *Filtering*, *Constraint type*,...)

About the catalogue

The catalogue presents a list of 348 global constraints issued from the literature in constraint programming and from popular constraint systems. The semantic of each constraint is given together with a description in terms of graph properties and/or automata.

Outline

Computational Models

Three main Computational Models to solve (combinatorial) constrained optimization problems:

- Mathematical Programming (LP, ILP, QP, SDP, ...)
- Constraint Programming (CSP as a model, SAT as a very special case)
- Local Search (... and Meta-heuristics)
- Others? Dynamic programming, dedicated algorithms, satisfiability modulo theory, etc.

Modeling

Modeling:

1. identify:
   - variables and domains
   - constraints
   - objective functions

   that formulate the problem

2. express what in point 1) in a way that allows the solution by available software

Variables

In MILP: real and integer (mostly binary) variables

In LS: integer variables

In CP:

- finite domain integer (including Booleans),
- continuos with interval constraints
- structured domains: finite sets, multisets, graphs, ...

Constraint Programming vs LS

- In LS modelling, we define: i) variables and domains, ii) objective function and iii) neighborhood. Constraints entered in the objective function. In CP constraints are declared explicitly.
- In LS, constraints handled indirectly either by solution representation or relaxed in the objective function. They can be violated. More constraints may be bad.
- In CP handled directly, used to remove values from variable domanis, they cannot be violated. More constraints may be good.

Constraint Programming vs MILP

- In MILP we formulate problems as a set of linear inequalities
- In CP we describe substructures (so-called global constraints) and combine them with various combinators.
- Substructures capture building blocks often (but not always) comptuationally tractable by special-purpose algorithms
- CP models can:
  - be solved by the constraint engine
  - be linearized and solved by their MIP solvers;
  - be translated in CNF and sovled by SAT solvers;
  - be handled by local search
- In MILP the solver is often seen as a black-box
  In CP and LS solvers leave the user the task of programming the search.
- CP = model + propagation + search
  constraint propagation by domain filtering $\rightsquigarrow$ inference
  search = backtracking or branch and bound (local search)

Example: Sudoku

How can you solve the following Sudoku?

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

Sudoku: LS model

Model:

- Varibales: $X_{ij} \in N$, restrict to be permutation over rows
- Evaluation function: count number of conflicts in the subsquares and on columns
- Neighborhood: exchange two vlaues on a row

Search: eg, iterative improvement, stochastic local search, tabu search, etc.

Sudoku: ILP model

Let y_{ijt} be equal to 1 if digit t appears in cell (i,j). Let N be the set $\{1,\ldots,9\}$, and let J_{kl} be the set of cells (i,j) in the 3×3 square in position k,l.

$$\sum_{j \in N} y_{ijt} = 1, \qquad \forall i, t \in N,$$

$$\sum_{j \in N} y_{jit} = 1, \qquad \forall i, t \in N,$$

$$\sum_{i,j \in J_{kl}} y_{ijt} = 1, \qquad \forall k, l = \{1,2,3\}, t \in N,$$

$$\sum_{t \in N} y_{ijt} = 1, \qquad \forall i, j \in N,$$

$$y_{i,j,a_{ij}} = 1, \qquad \forall i, j \in \text{ given instance.}$$

Sudoku: CP model

Model:

$$
\begin{aligned}
& X_{ij} \in N, && \forall i, j \in N, \\
& X_{ij} = a_{ij}, && \forall i, j \in \text{ given instance}, \\
& \text{alldifferent}([X_{1i}, \ldots, X_{9i}]), && \forall i \in N, \\
& \text{alldifferent}([X_{i1}, \ldots, X_{i9}]), && \forall i \in N, \\
& \text{alldifferent}(\{X_{ij} \mid ij \in J_{kl}\}), && \forall k, l \in \{1, 2, 3\}.
\end{aligned}
$$

Search: backtracking

Sudoku: CP model (revisited)

$$
\begin{aligned}
& X_{ij} \in N, && \forall i,j \in N,\\
& X_{ij} = a_t, && \forall i,j \in \text{ given instance},\\
& \text{alldifferent}([X_{1i},\ldots,X_{9i}]), && \forall i \in N,\\
& \text{alldifferent}([X_{i1},\ldots,X_{i9}]), && \forall i \in N,\\
& \text{alldifferent}(\{X_{ij} \mid ij \in J_{kl}\}), && \forall k,l \in \{1,2,3\}.
\end{aligned}
$$

Redundant Constraint:

$$
\begin{aligned}
& \sum_{j \in N} X_{ij} = 45, && \forall i \in N,\\
& \sum_{j \in N} X_{ji} = 45, && \forall i \in N,\\
& \sum_{ij \in J_{kl}} X_{ij} = 45, && k,l \in \{1,2,3\}.
\end{aligned}
$$

Hybrid Methods?

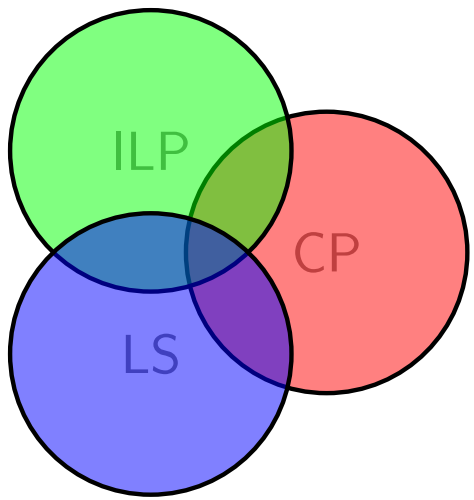
Strengths:

- CP is excellent to explore highly constrained combinatorial spaces quickly
- Math programming is particulary good at deriving lower bounds
- LS is particualry good at derving upper bounds

How to combine them to get better "solvers"?

- Exploiting OR algorithms for filtering
- Exploiting LP (and SDP) relaxation into CP
- Hybrid decompositions:

  1. Logical Benders decomposition
  2. Column generation
  3. Large-scale neigbhrohood search



Integrated Modeling

Models interact with solution process hence models in CP and IP are different.

To integrate one needs:

- to know both sides
- to have available a modelling language that allows integration (python, C++, MiniZinc)

There are typcially alternative ways to formulate a problem. Some may yield faster solutions.

Typical procedure:

- begin with a strightforward model to solve a small problem instance
- alter and refine the model while scaling up the instances to maintain tractability

CP and LS as alternative approaches

- Constraint Programming
  1. Declarative modeling language (model&solve paradigm)
  2. Global constraints and filtering algorithm
  3. Search tree
- Constraint-based Local Search
  1. Declarative modeling language (model&solve paradigm)
  2. Global constraints and penalty functions
  3. Local Search and Meta-heuristic (incremental evaluation and commitment of moves)
- Hybrid methods (... once there was Comet)

Resume

- LS models
  - largest flexibility but everything to implement
- MILP models
  - impose modelling rules: linear inequalities and objectives
  - emphasis on tightness and compactness of LP, strength of bounds (remove dominated constraints)
- CP models
  - a large variety of algorithms communicating with each other: global constraints
  - more expressiveness
  - emphasis on exploiting substructres, include redundant constraints