Combinatorial Decision Making & Optimization – Mod I

2023/2024

Second cycle degree/two year Master's in Artificial Intelligence
Dept of Computer Science and Engineering (DISI)
University of Bologna

Syllabus

- **CP**
  - February 22: Introduction to the course and CP
  - February 23: Modelling
  - February 29: Local consistency, constraint propagation, global constraints
  - March 1: Search
  - March 7, 8: Exercises in MiniZinc
- **SAT**
  - March 14: Introduction to SAT, encoding decision problems in SAT
  - March 15: Basic solving techniques (resolution, unit propagation, DPLL)
  - March 21: Conflict-driven clause learning SAT solvers, hybrid CP-SAT solvers
  - March 22: Exercises in Z3, SAT encodings
  - April 4: SAT encodings, Exercises in Z3

Syllabus

- **Invited speaker**
  - April 5: Filippo Focacci
    - Founder and CEO of Decision Brain.
    - PhD at the University of Ferrara, Italy.
    - Technical talk on London's bike hiring scheme developed using IBM decision optimization tools.

Introduction

- Why with Constraint Programming (CP)?
- Overview of CP.
- Resources.

Popularity of Constraint Programming

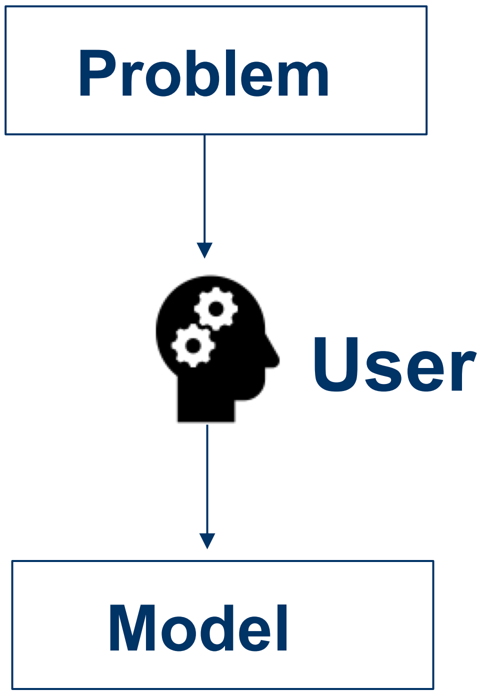
- An important and growing area of AI.
  - Universities, research centers and companies (such as IBM, Google) around the world contribute to the advancement of the state-of-the-art.
  - Many companies are applying CP successfully.
    - Including IBM, Google, Ericsson, Siemens, Renault, Oracle, Sap, Intel, Tacton
- Technology of choice in logistics, scheduling, planning…
- A useful asset on the job market!

Covid-19 Test Scheduling

- Ocado Retail Ltd is one of the world's biggest online-only grocery retail businesses.
- Employs over 15K people, many of them performing frontline roles such as packing in the warehouses, order deliveries, providing customer service in the call centers.
- With the pandemic, the company decided to test all frontline employees on a weekly basis, which required scheduling the employees at each site subject to various constraints.
  - Proved difficult to solve manually.
- Data Science team developed a CP-based solution, which was successfully used to schedule up to 3,500 employees across 4 sites (IFORS news, vol. 15, number 4, December 2020)

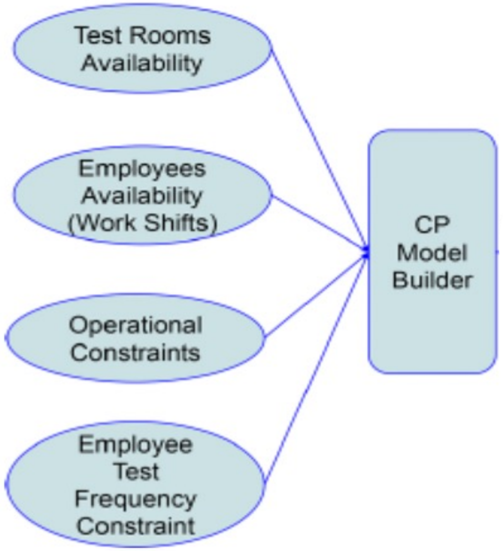
What is Constraint Programming?

- A declarative programing paradigm for stating and solving combinatorial optimization problems.
  - User models a decision problem by formalizing:
    - the unknowns of the decision → decision variables ($X_1,...,X_n$).
    - possible values for unknowns → domains ($D_1,...,D_n$ with each $D_i(X_i) = \{v_{i1},...,v_{id}\}$.
    - relations between the unknowns → constraints ($C_1,\dots,C_m$ with each $C_i(X_j, \dots, X_k)$).



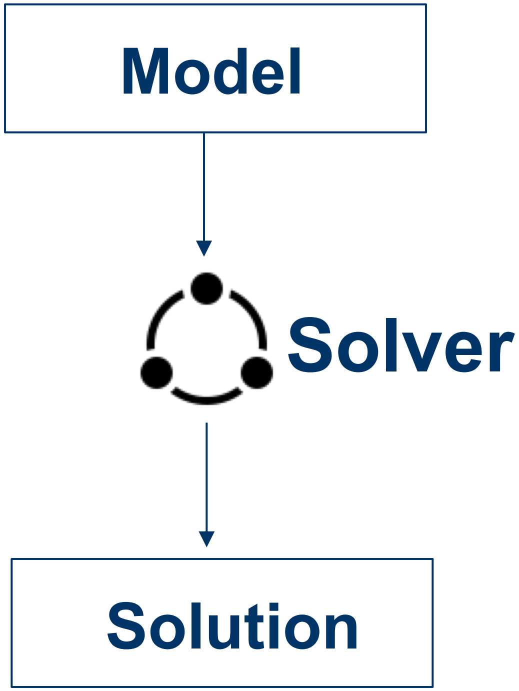
Covid-19 Test Scheduling

- Availability Constraints
  - Testing room, tester, and employee availabilities.
- Frequency constraints
  - The spacing between tests performed on the same employee should be within given bounds.
- Operational constraints
  - Each employee should be tested within their working shift.
  - Only a limited share of employees from the same work area should be scheduled for a test on the same day.

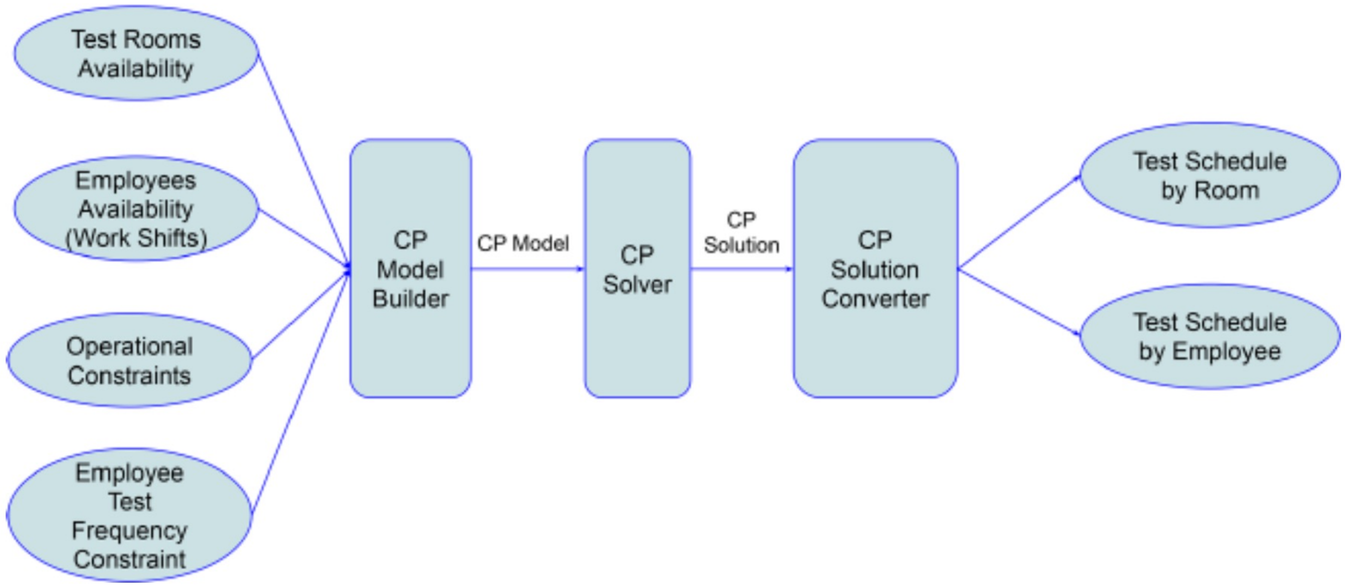


What is Constraint Programming?

- A declarative programing paradigm for stating and solving combinatorial optimization problems.
  - A constraint solver finds a solution to the model (or proves that no solution exists) via a search algorithm by assigning a value to every variable ($X_i = v_{ij}$) such that all constraints are satisfied.



Covid-19 Test Scheduling



Why Constraint Programming?

- CP provides a rich language for expressing constraints and defining search procedures.
  - Easy modelling.
    - Fast prototyping with a variety of constraints.
    - Easy to maintain programs.
    - Extensibility.
  - Easy control of search.
    - Experimentation with advanced search strategies.

Orthogonal and Complementary Approaches to CDMO

- ILP from OR
  - Modeling with linear inequalities.
  - Numerical calculations.
  - Focus on objective function and optimality.
    - Bounding → elimination of suboptimal values from domains.
  - Exploits global structure.
    - Relaxations, cutting planes, and duality theory.
- CP from AI
  - Rich language for modeling and search procedures.
  - Logical processing.
  - Focus on constraints and feasibility.
    - Propagation → elimination of infeasible values from domains.
  - Exploits local structure.
    - Domain reductions based on individual constraints.

Strengths of CP

- Success on irregular problems!
  - Timetabling, sequencing, scheduling allocation, rostering, etc.
  - Contain messy constraints non-linear in nature.
  - Contain multiple disjunctions which result in poor information returned by a linear relaxation of the problem.

Weaknesses and Opportunities of CP

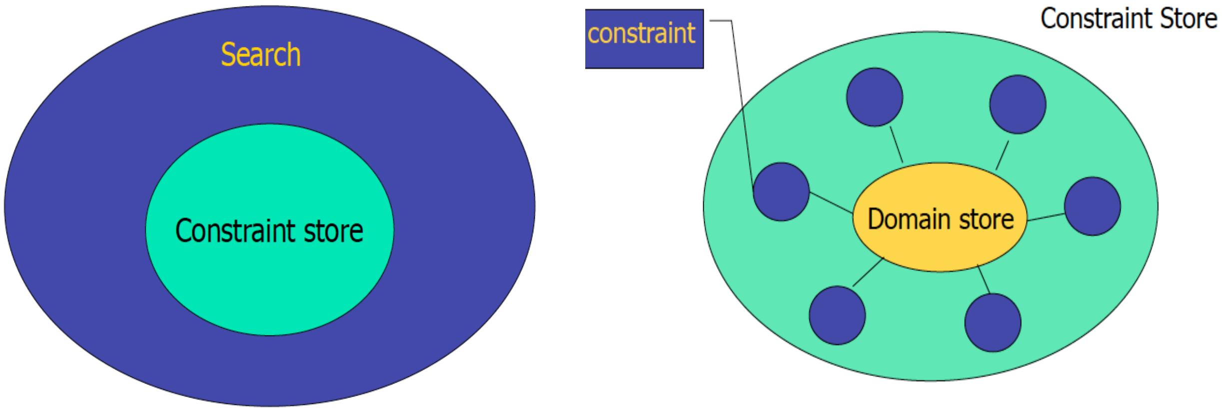
- Optimality
  - CP: no special focus on objective function and optimality ☹
  - ILP: scales up on loosely constrained optimization problems.
  - HS: is effective in finding quickly good-quality solutions.
- Best optimality approaches are often hybrids of CP, ILP and HS.
  - CP is a suitable framework for hybridization ☺

Overview of CP

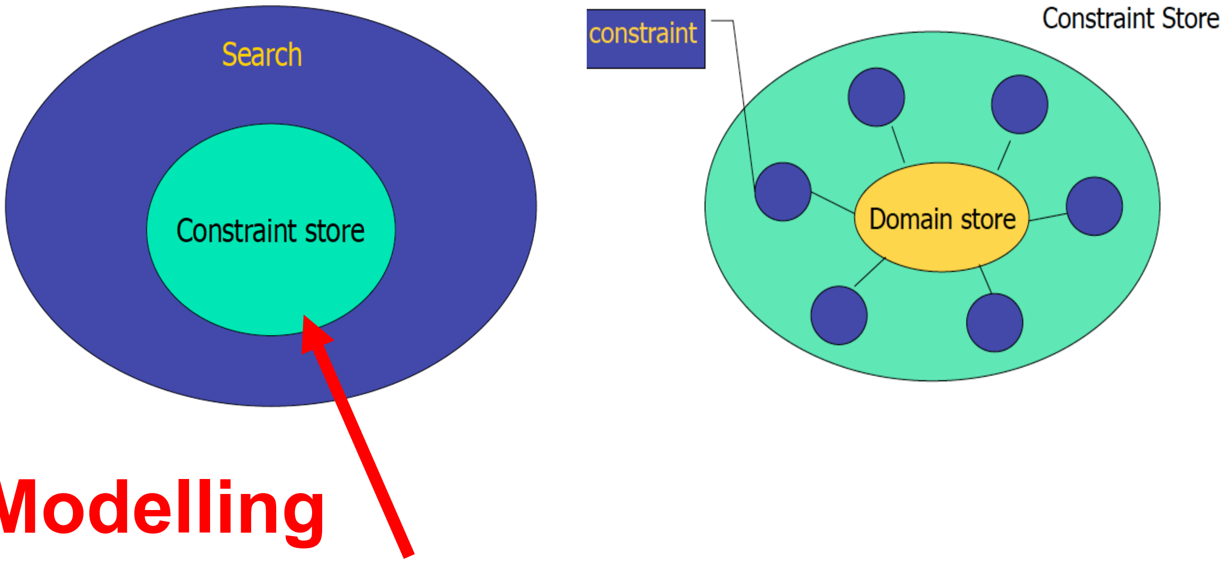
Constraint Solver

- Enumerates all possible variable-value combinations via a systematic backtracking tree search.
  - Guesses a value for each variable.
- During search, examines the constraints to remove incompatible values from the domains of the future (unexplored) variables, via propagation.
  - Shrinks the domains of the future variables.

Constraint Programming



Constraint Programming



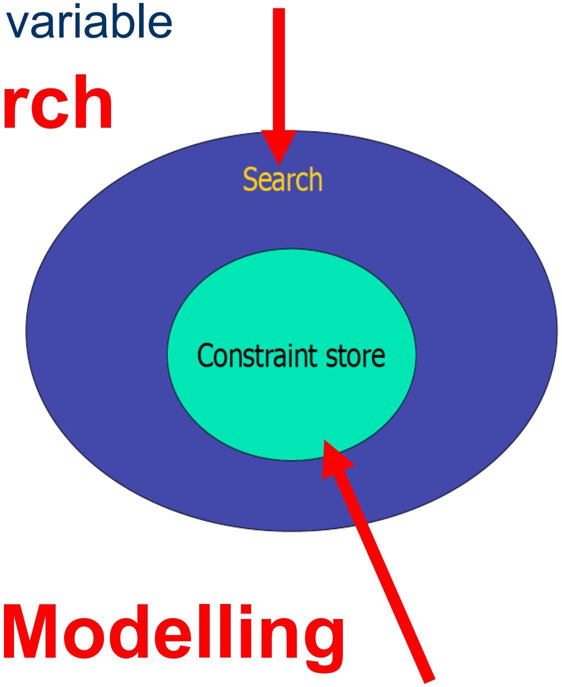
Modelling

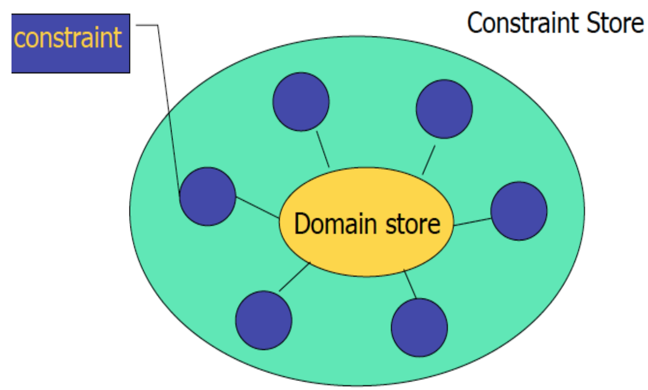
User expresses the problem

Constraint Programming

Solver uses a backtracking tree search algorithm to guess a value for each variable

Search





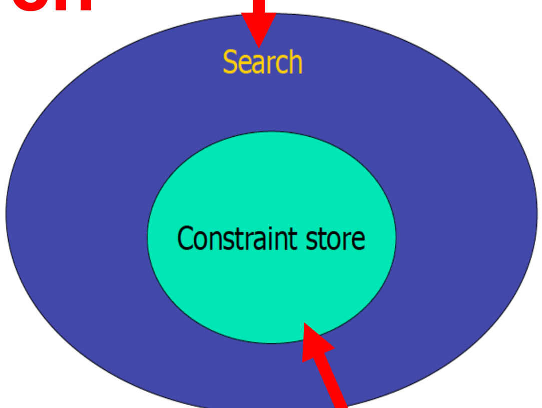
Modelling

User expresses the problem

Constraint Programming

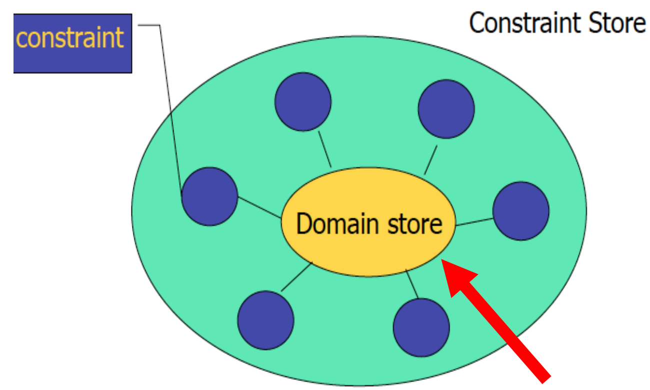
Solver uses a backtracking tree search algorithm to guess a value for each variable

Search



Modelling

User expresses the problem



Propagation

Solver uses algorithms to examine each constraint to reduce the domains of the future variables

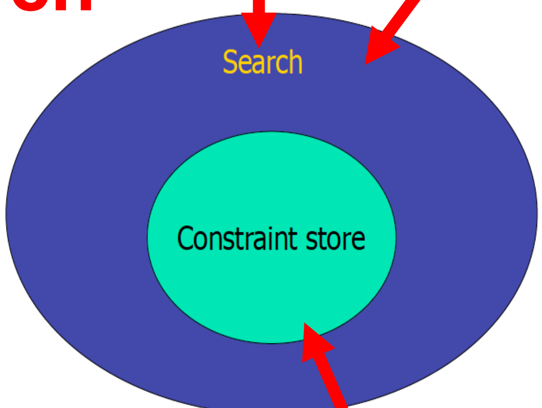
Constraint Programming

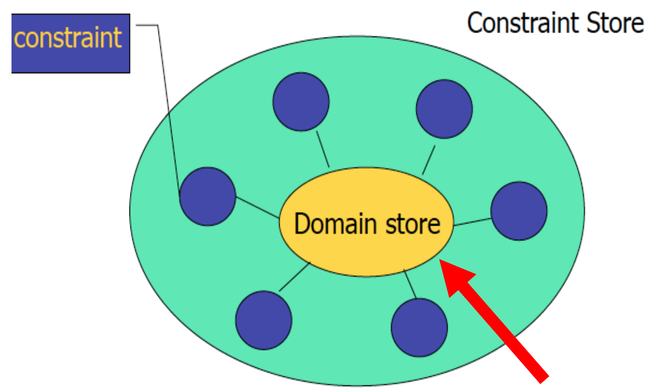
Solver uses a backtracking tree search algorithm to guess a value for each variable

Search

Solver exploits the current search state and problem specific knowledge to guide the search

Search heuristics





Modelling

User expresses the problem

Propagation

Solver uses algorithms to examine each constraint to reduce the domains of the future variables

Dual Role of a Model

- Captures combinatorial substructures.
- Enables solver to reduce the search space.
  - Constraints act as propagation algorithms.
  - Variables' domains act as communication mechanism.

Search and Propagation

- Search decisions and propagation are interleaved.

Propagation

$\Downarrow$

$X_i = v_{ij}$

$\Downarrow$

Propagation

$\Downarrow$

$X_k = v_{kj}$

$\Downarrow$

Propagation

…

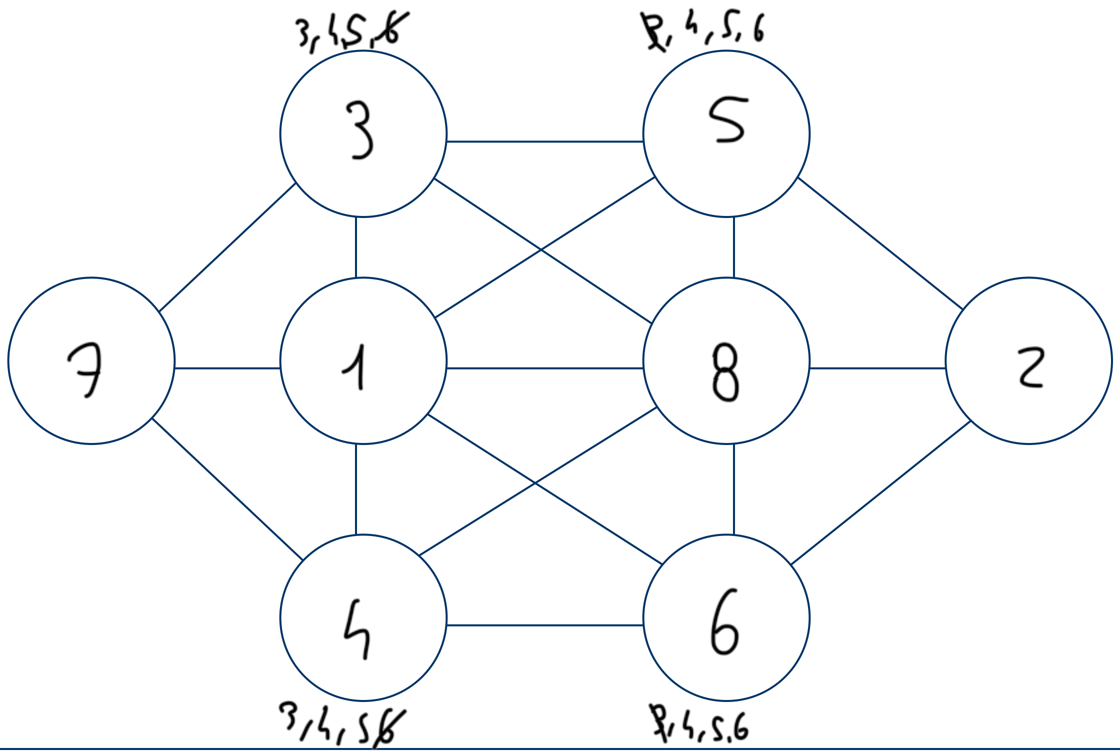
Expectation from CP

- Declarative programming
  - The user declaratively models the problem.
  - An underlying solver returns a solution with its default search.

Reality in CP

- Modelling is critical!
  - The user often has to use advanced modelling techniques for strong propagation.
- Default search of the solver is usually not enough!
  - The user often has to program the search strategy (search algorithm, search heuristics,…)

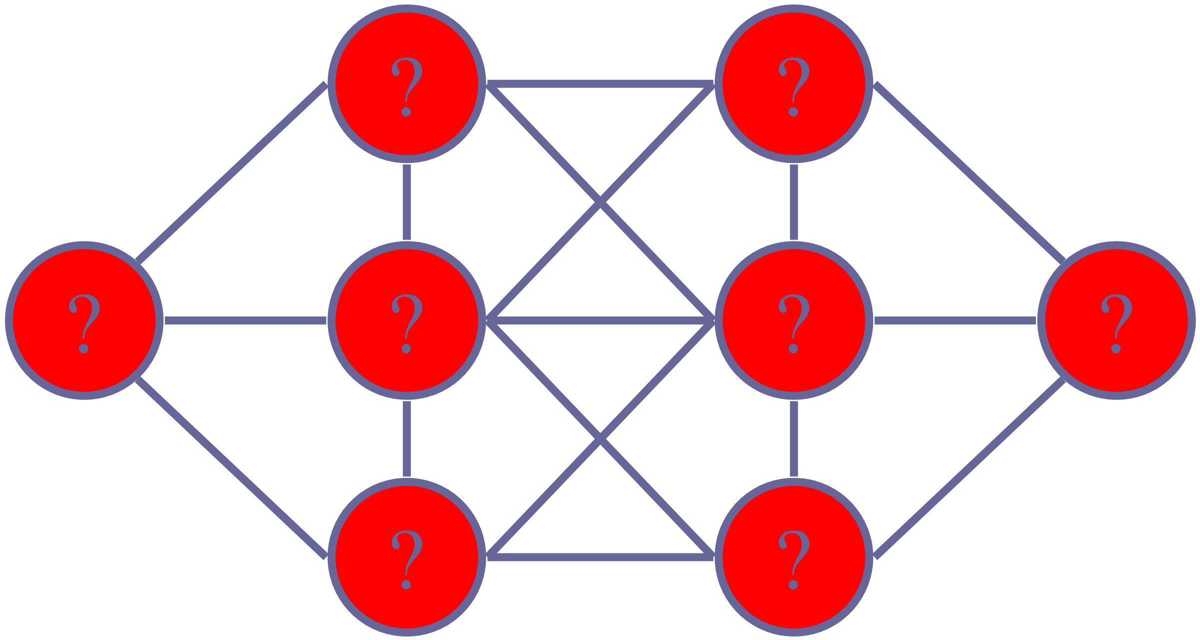
A Puzzle



Place a different number in each node (1 to 8) such that adjacent nodes cannot take consecutive numbers

A Puzzle

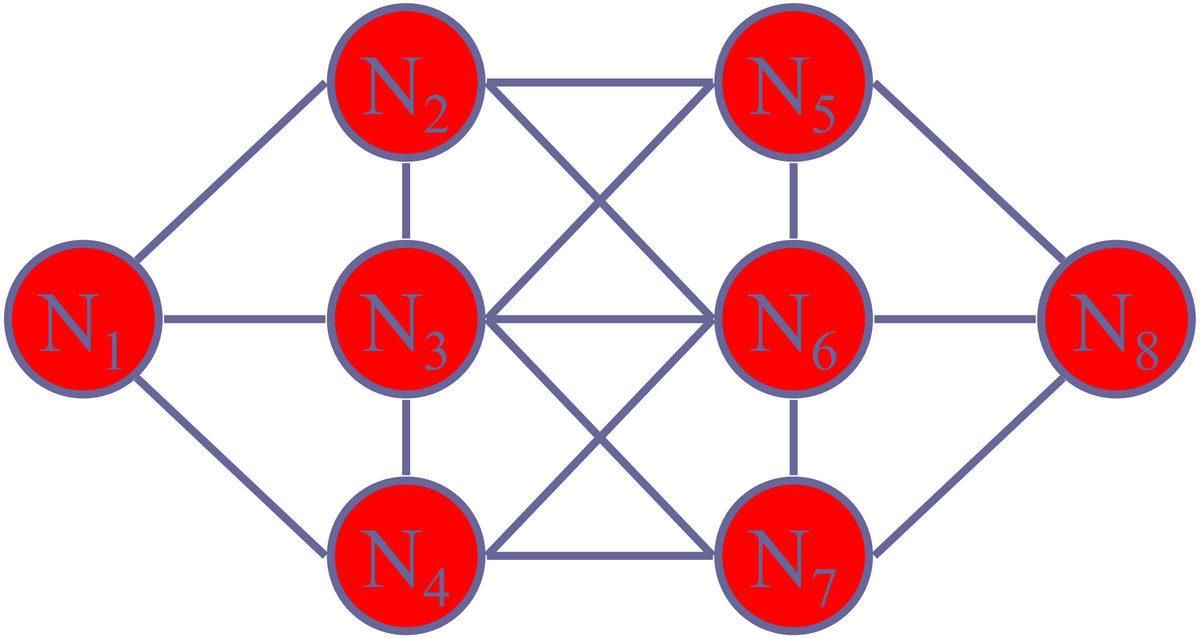
- Place numbers 1 through 8 on nodes, s.t.:
  - each number appears exactly once;
  - no connected nodes have consecutive numbers.



Modelling

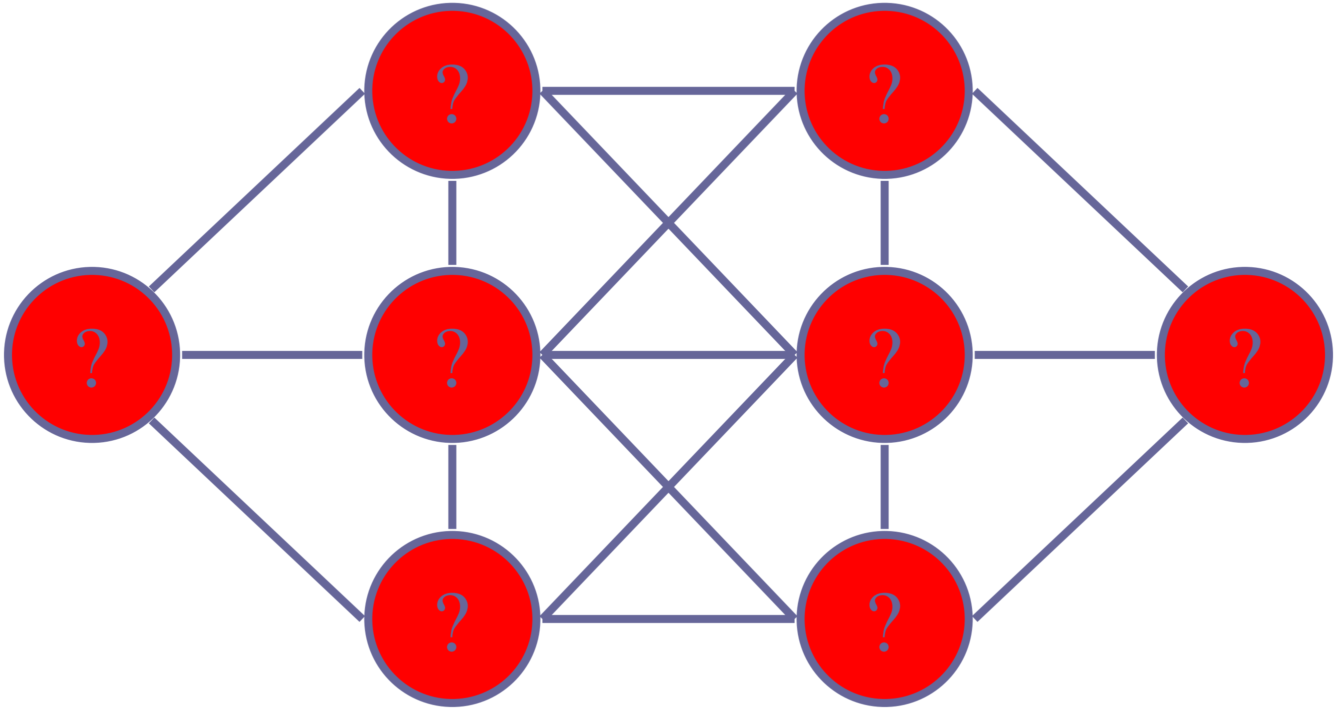
- Variables: $N_1 \ldots N_8$ that represent the nodes
- Domains: the set of values {1,2,3,4,5,6,7,8} that $N_1 .. N_8$ can take
- Constraints: for all $i < j$ s.t. N_i and N_j are adjacent $|N_i - N_j| > 1$

  for all $i < j$ $N_i \neq N_j$



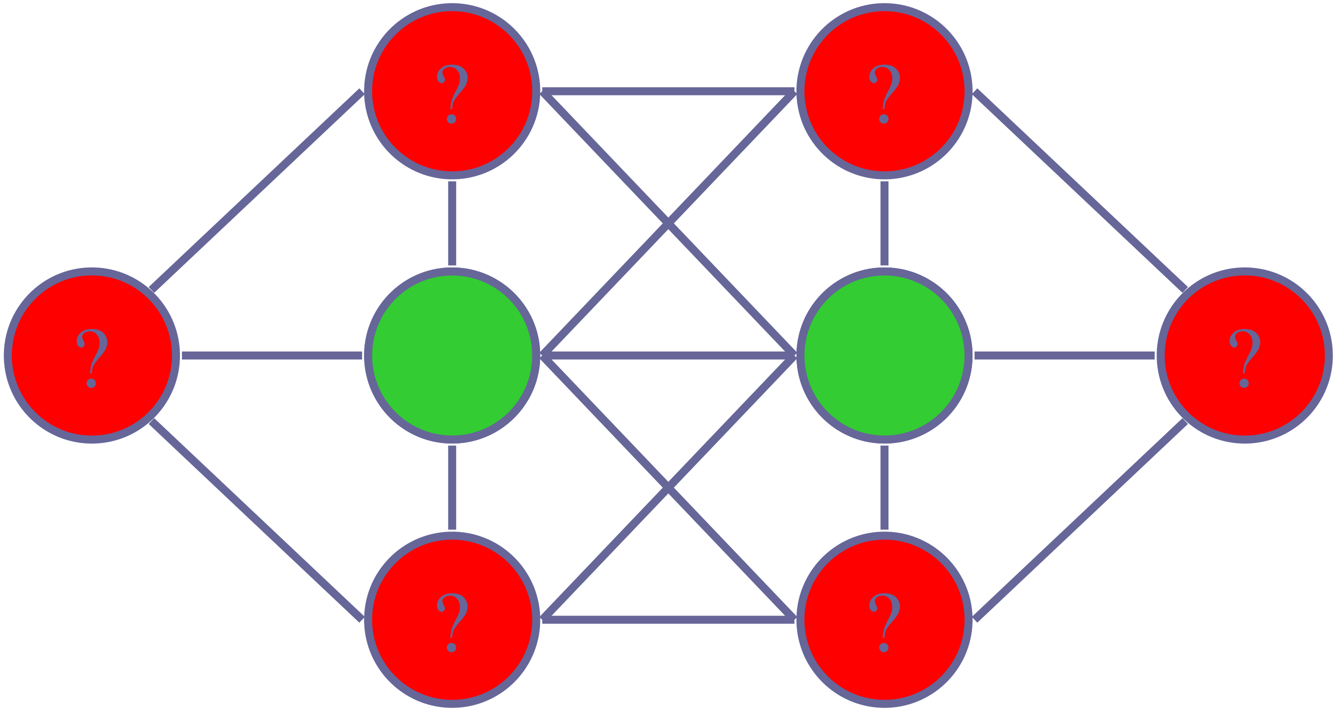
Backtracking Search + Heuristics

- Guess a value for a variable!



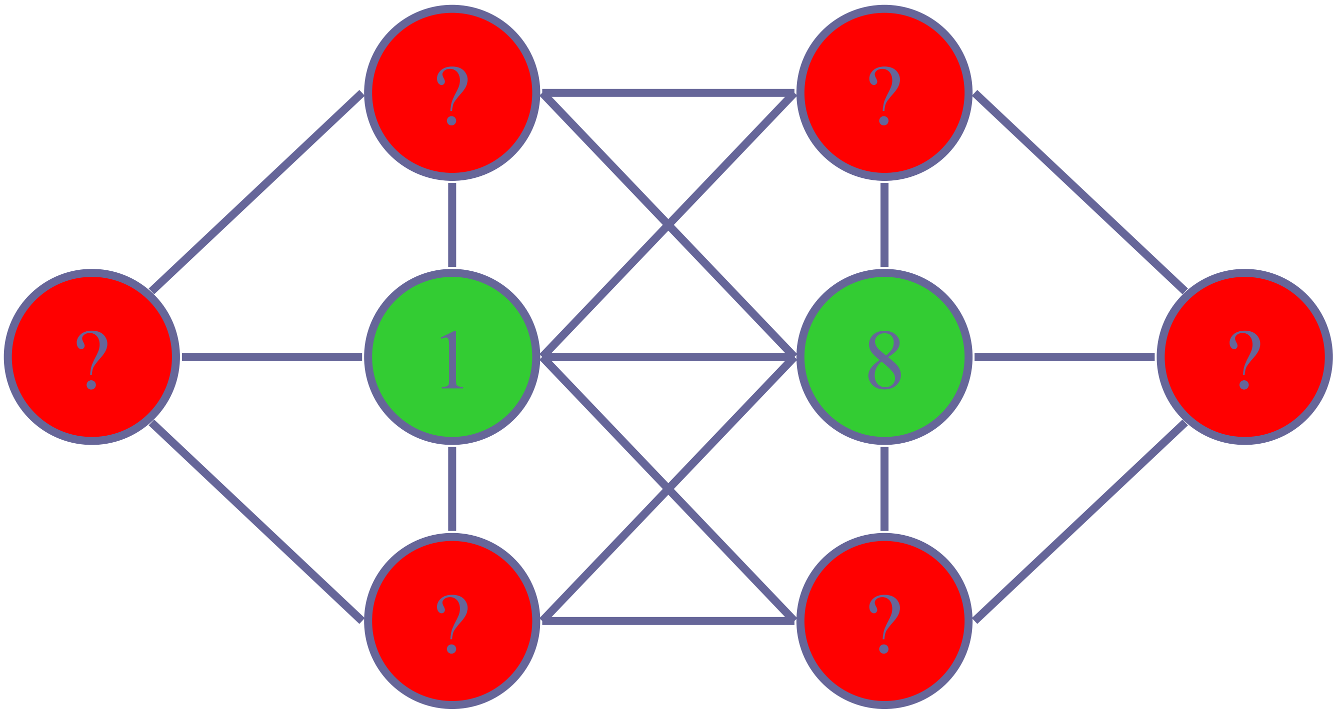
Backtracking Search + Heuristics

- Guess a value for a variable!
  - We start with the hardest variables.



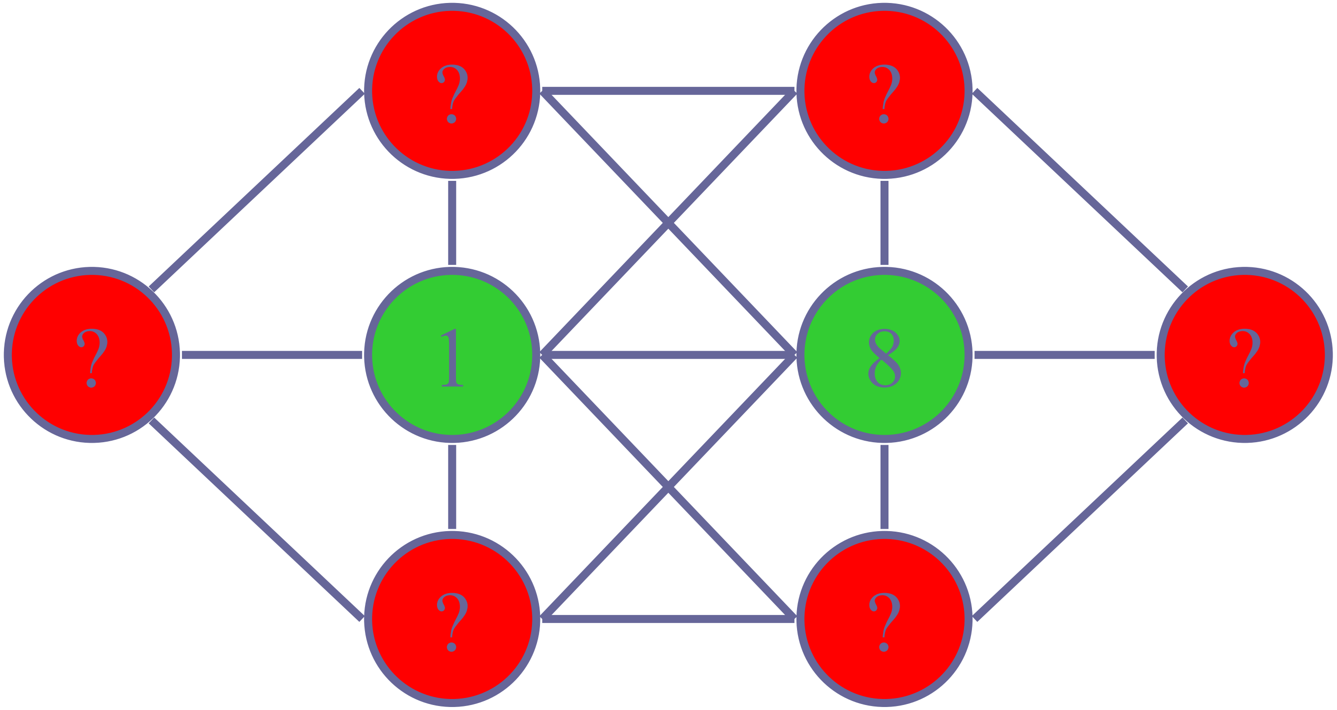
Backtracking Search + Heuristics

- Guess a value for a variable!
  - We assign them the safest values.

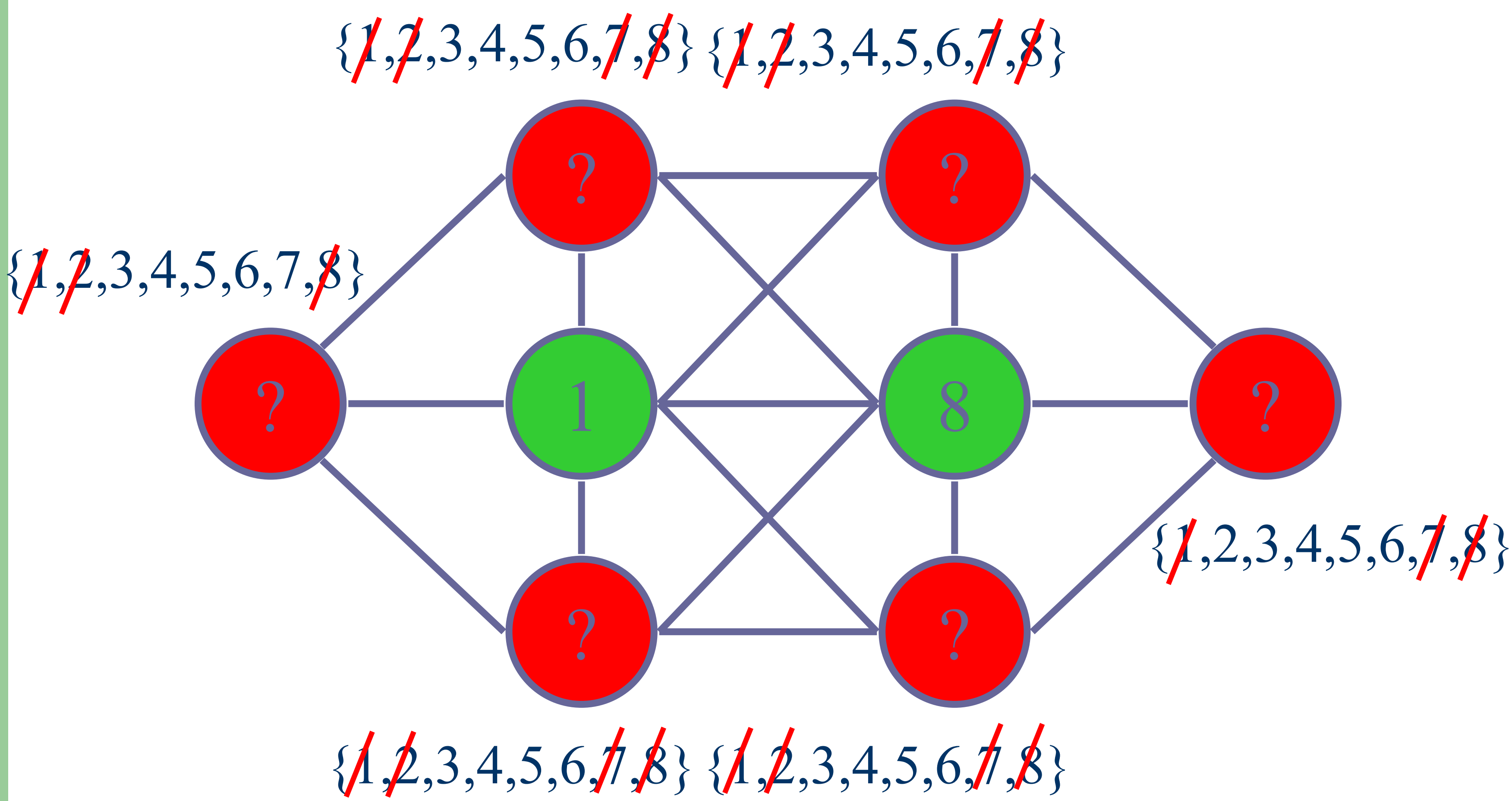


Propagation

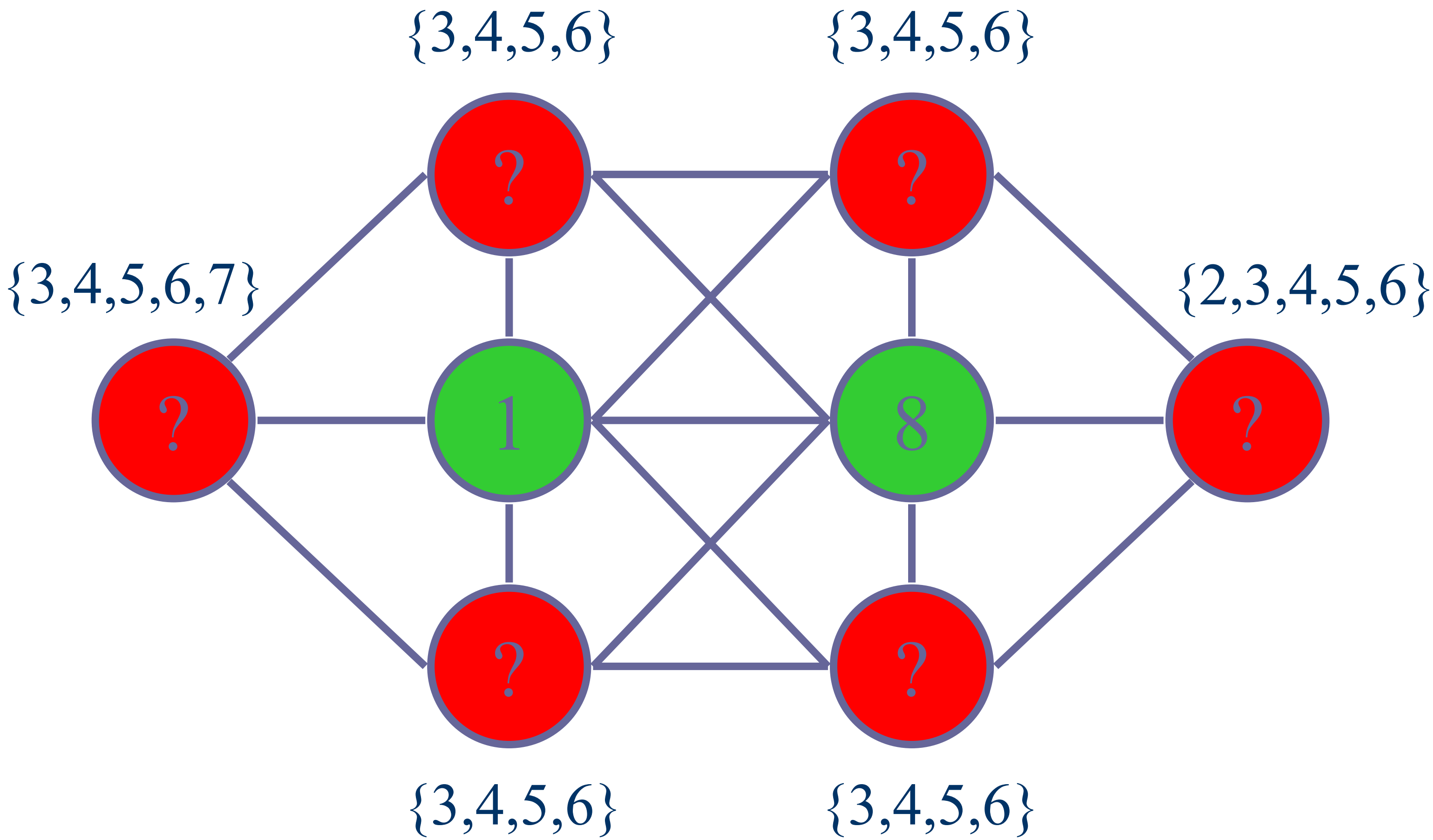
- We now examine the constraints.



Propagation

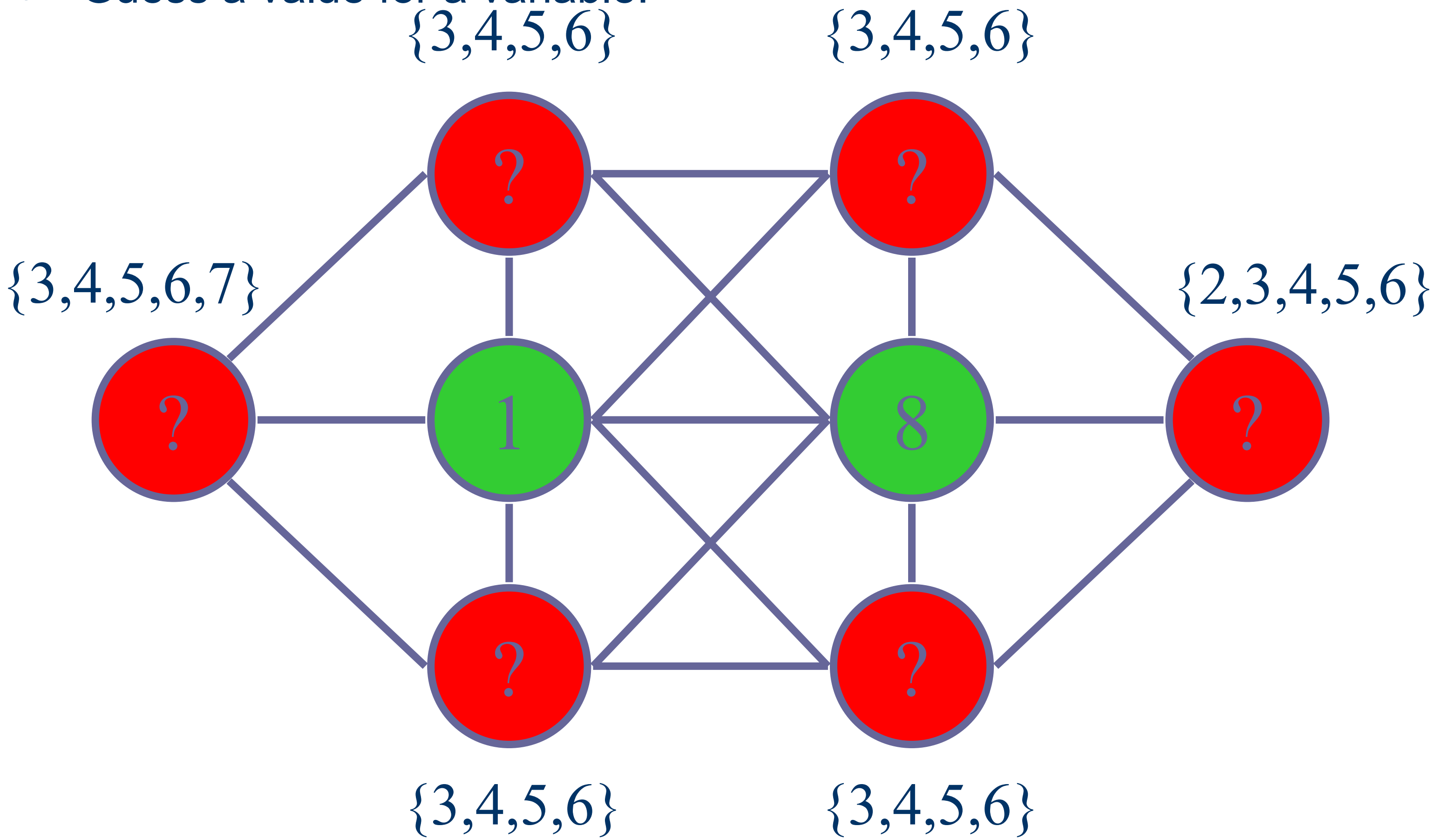


Propagation



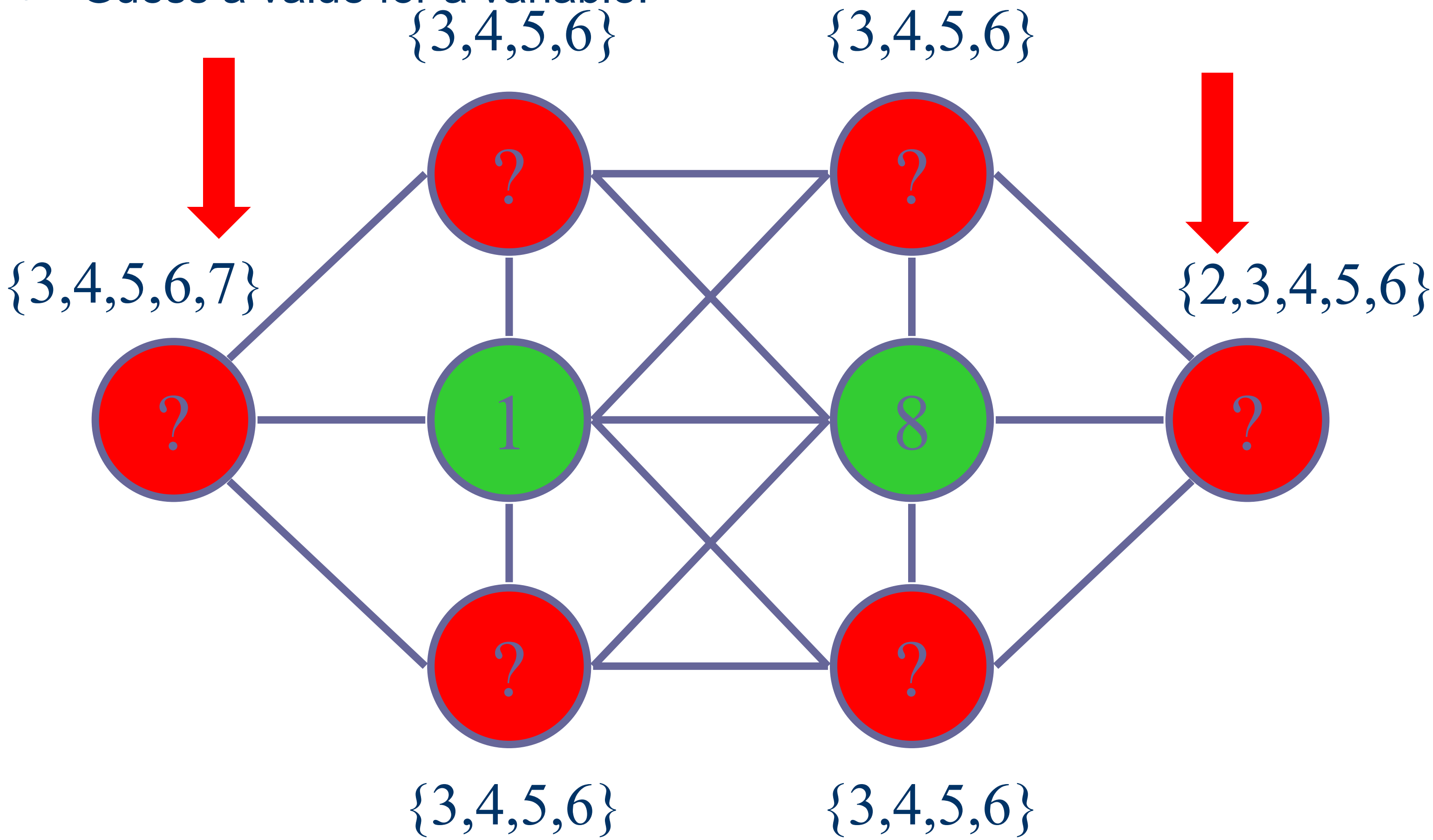
Backtracking Search + Heuristics

- Guess a value for a variable!

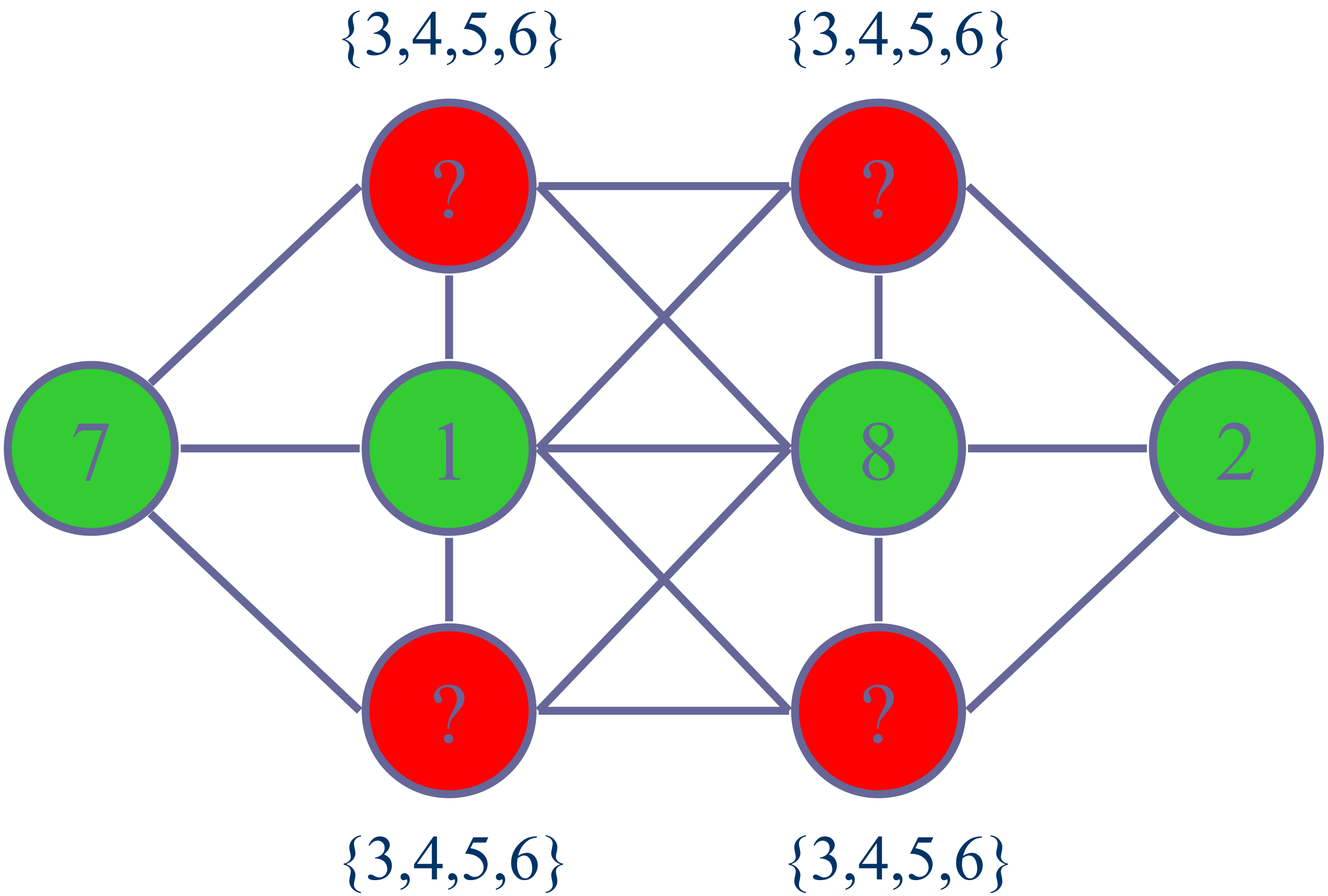


Backtracking Search + Heuristics

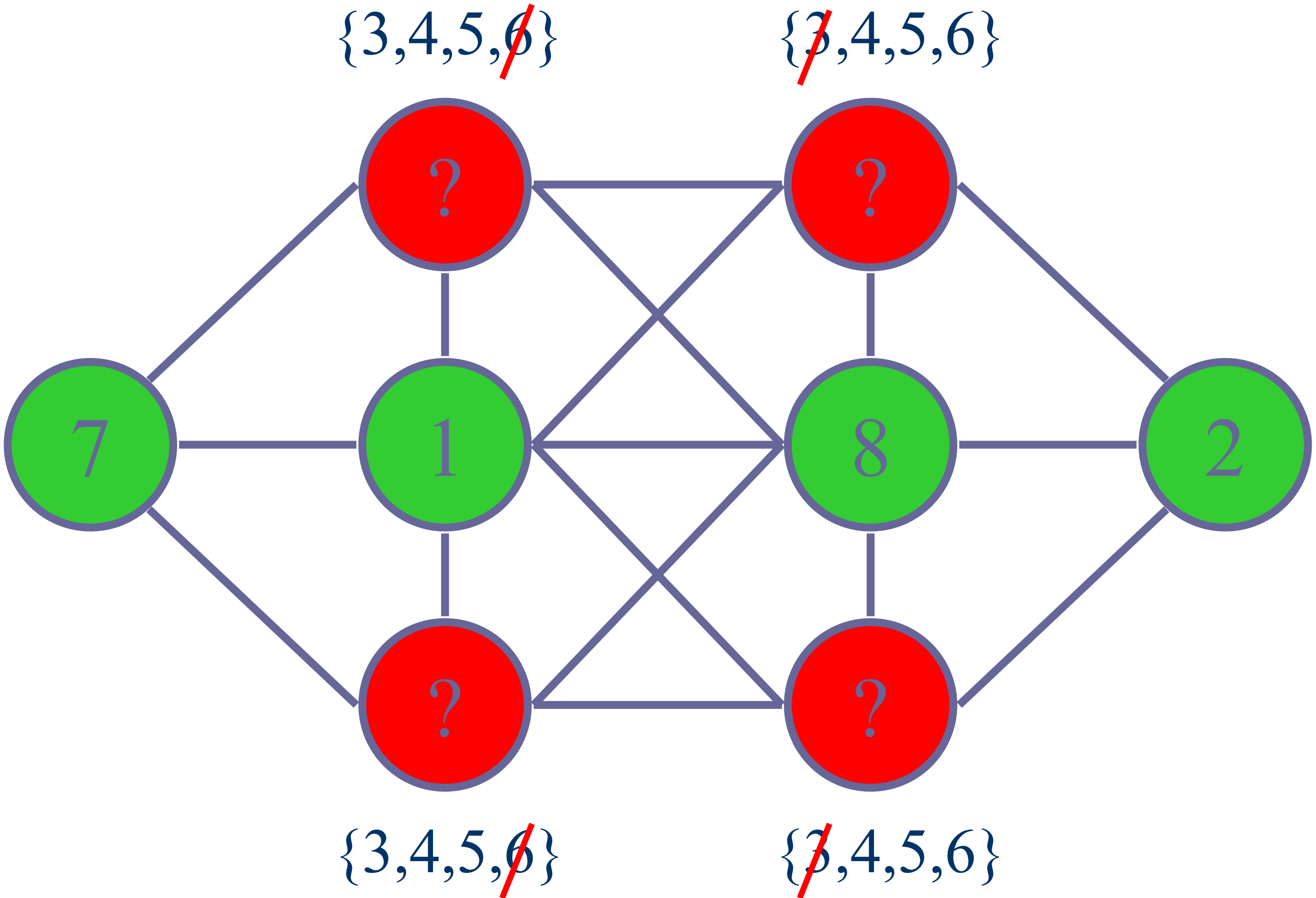
- Guess a value for a variable!



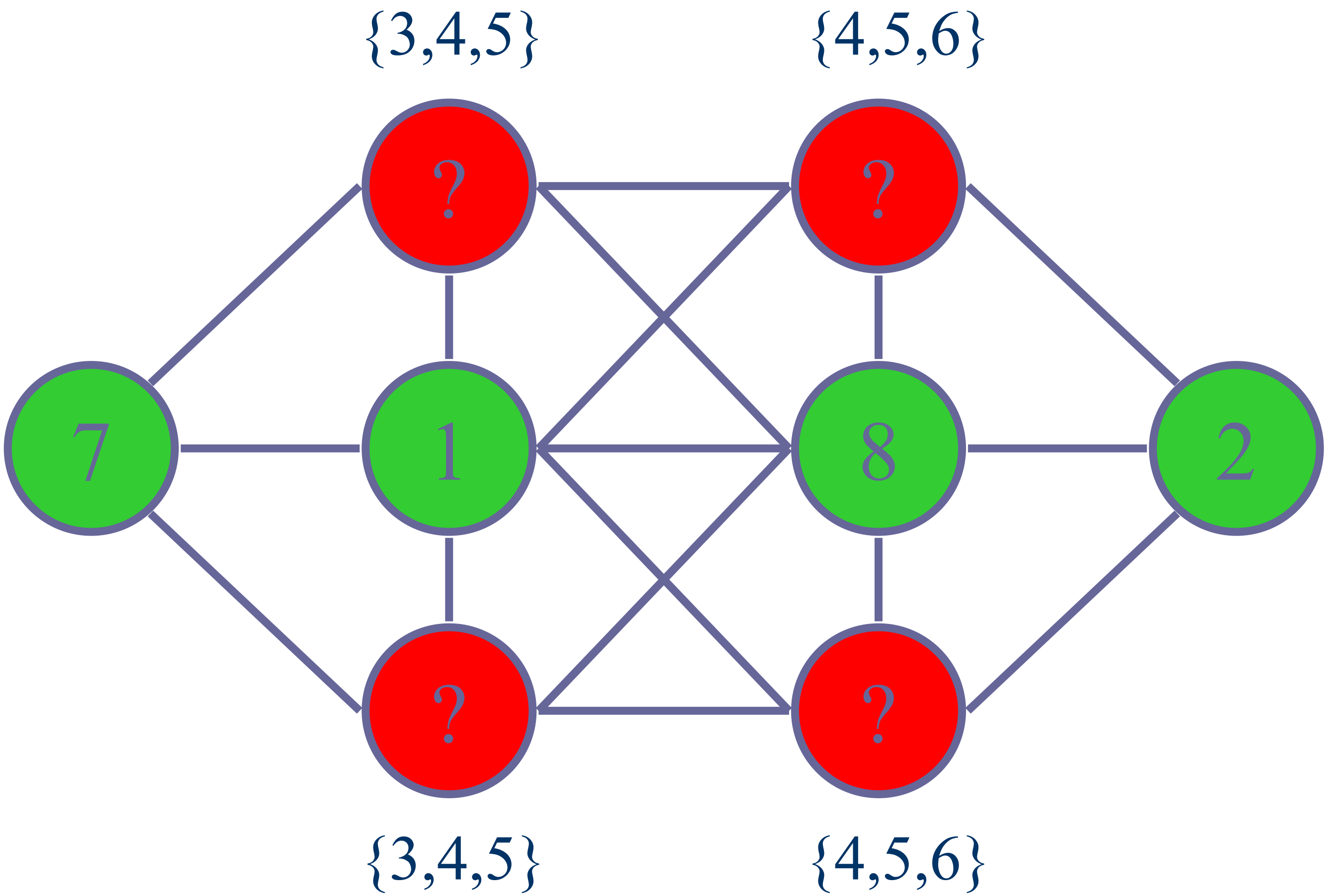
Backtracking Search + Heuristics



Propagation

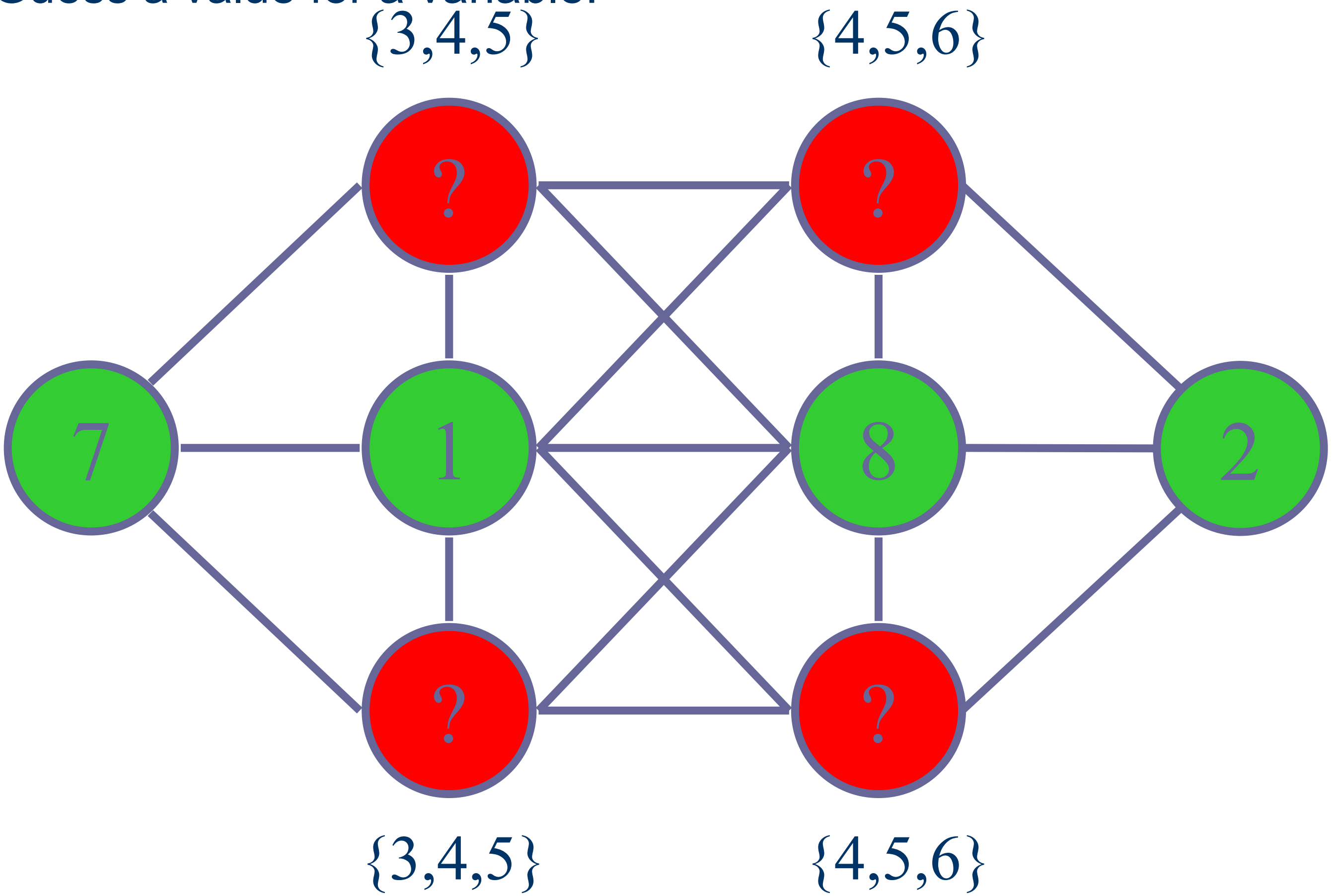


Propagation

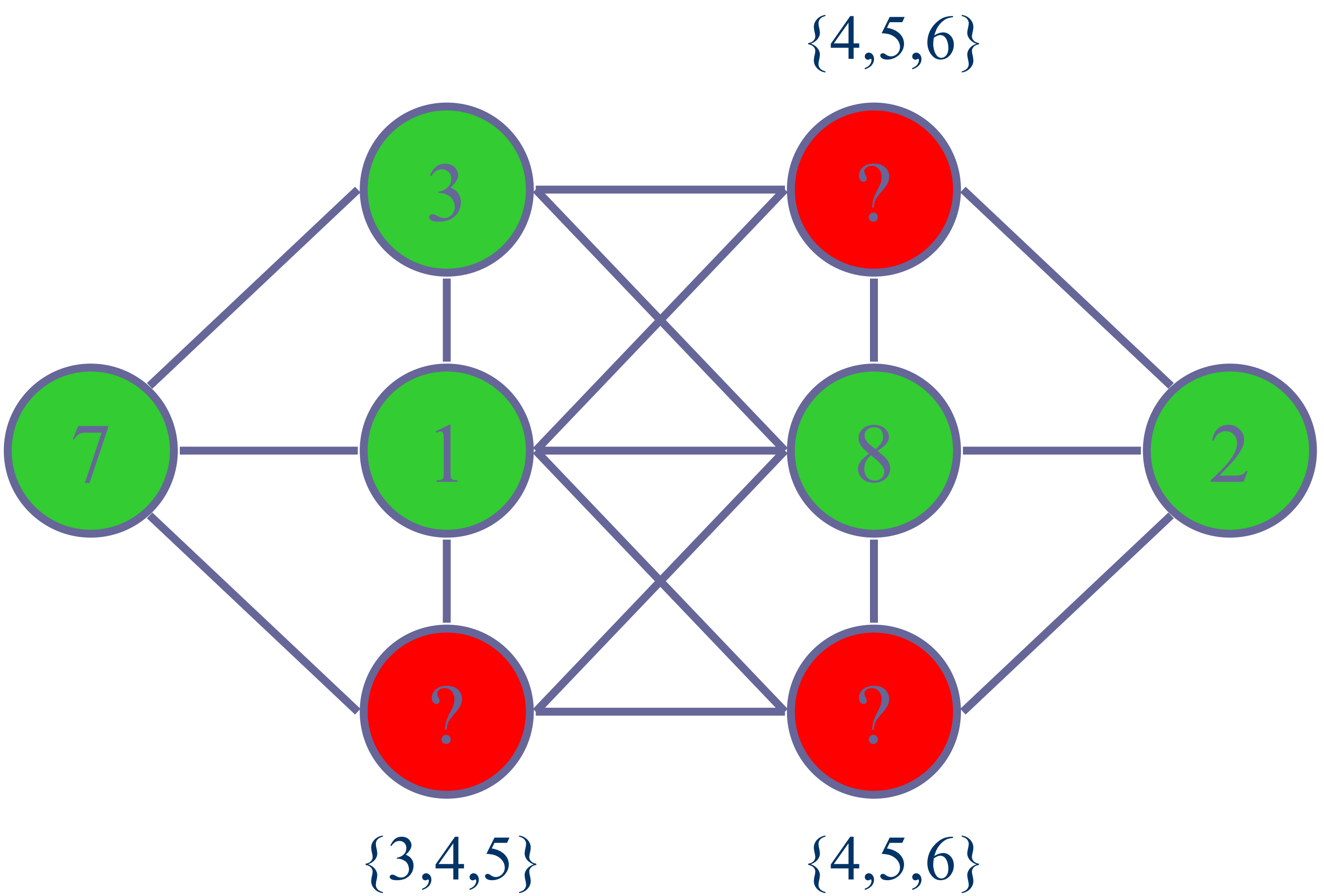


Backtracking Search + Heuristics

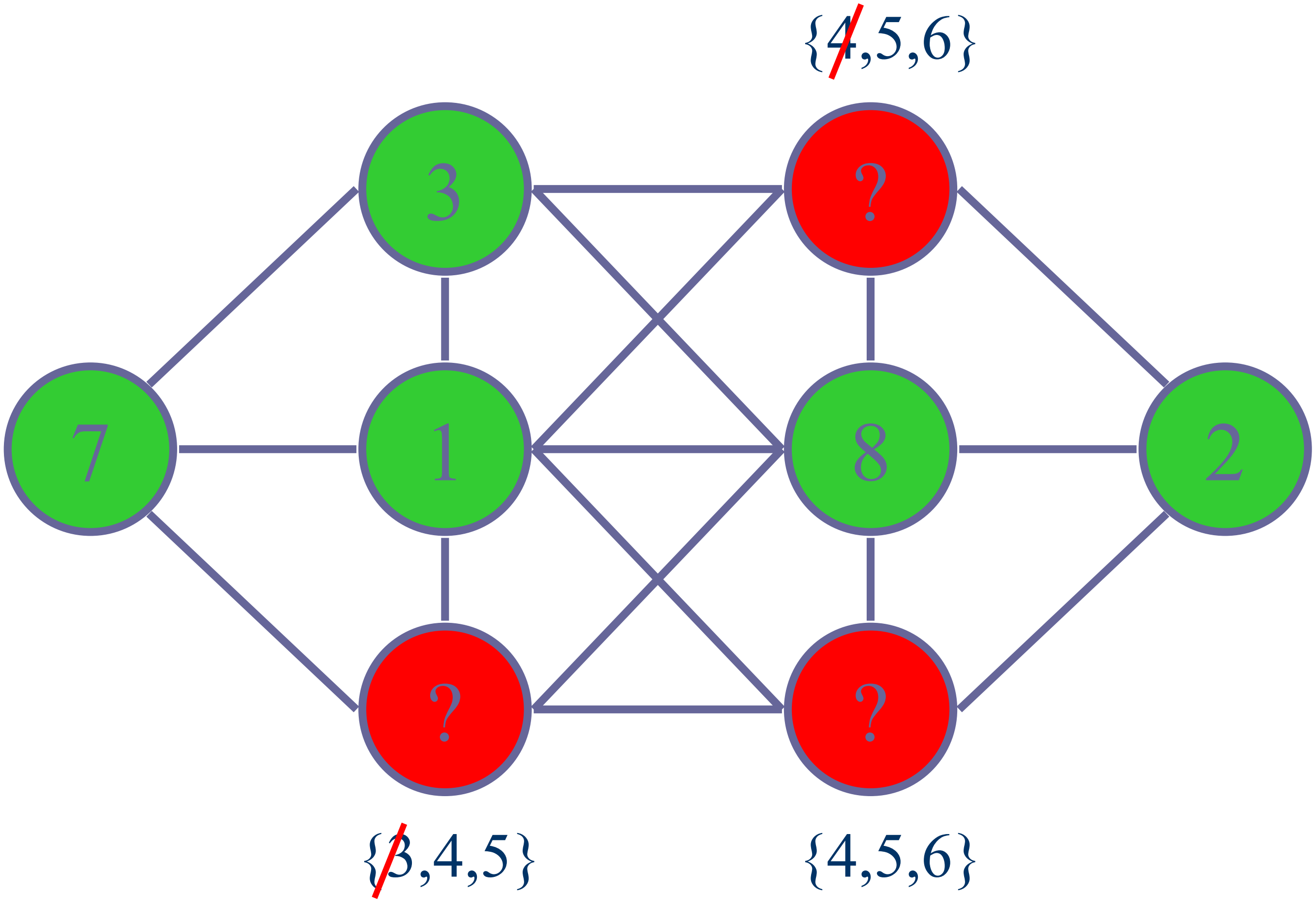
- Guess a value for a variable!



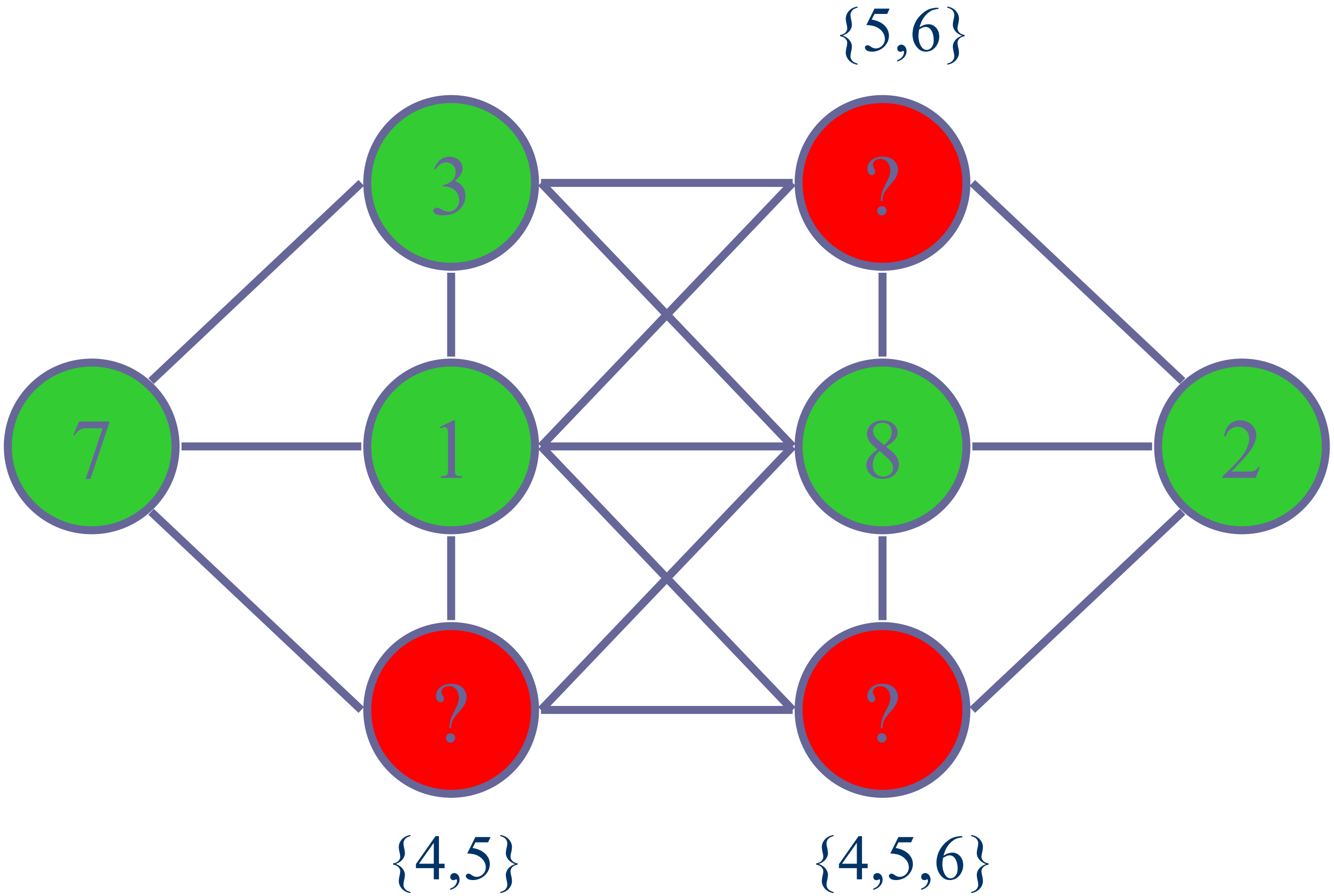
Backtracking Search + Heuristics



Propagation

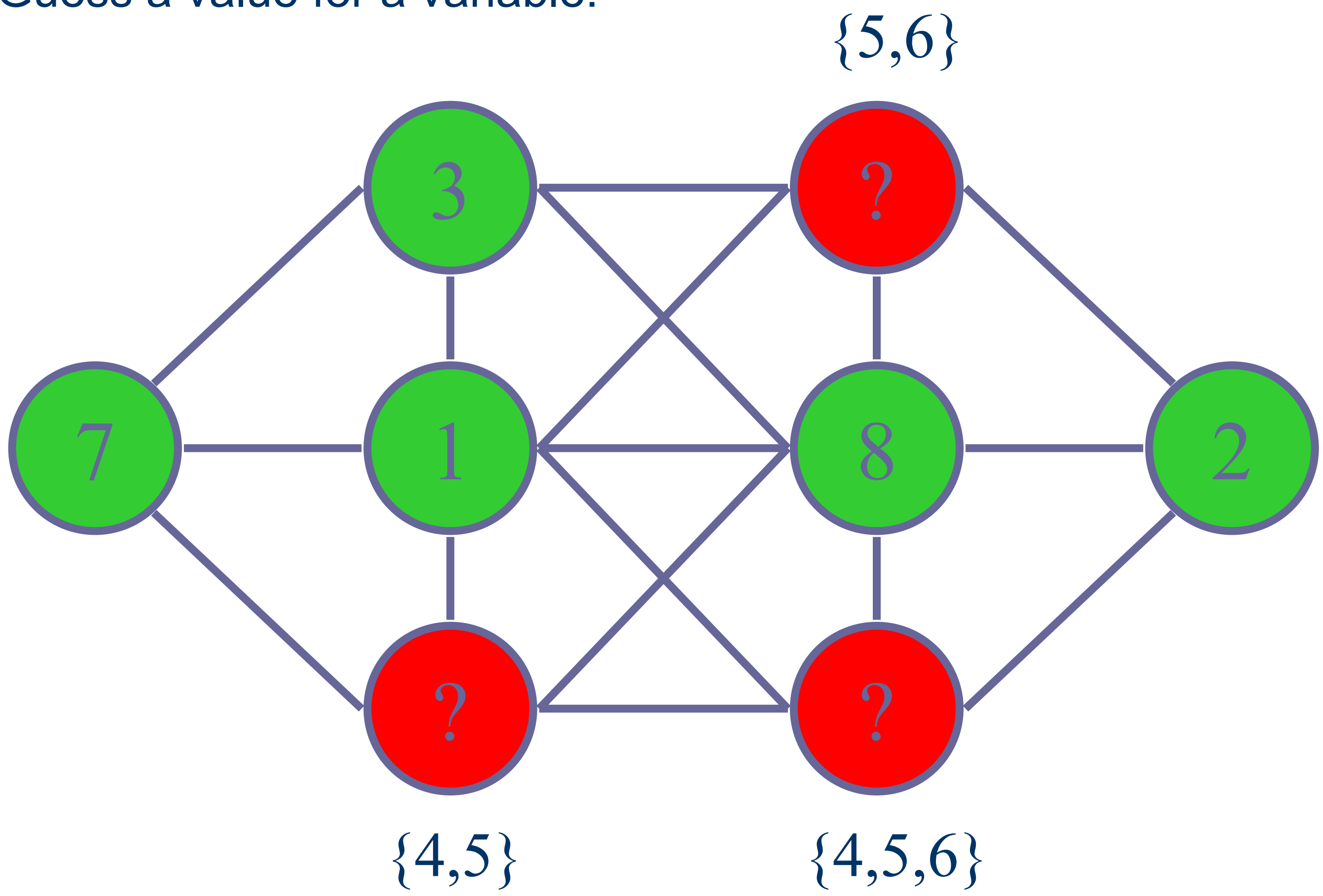


Propagation

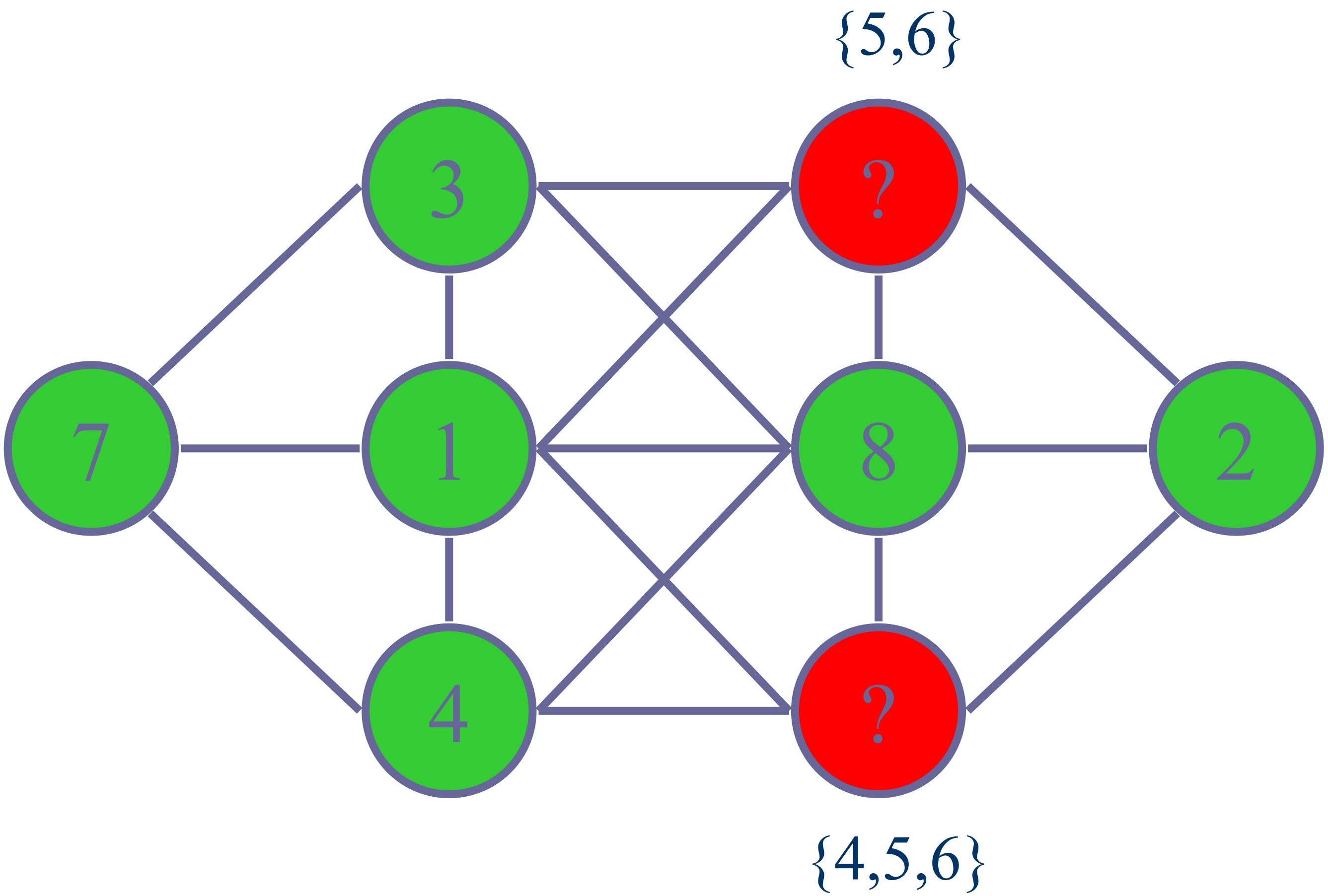


Backtracking Search + Heuristics

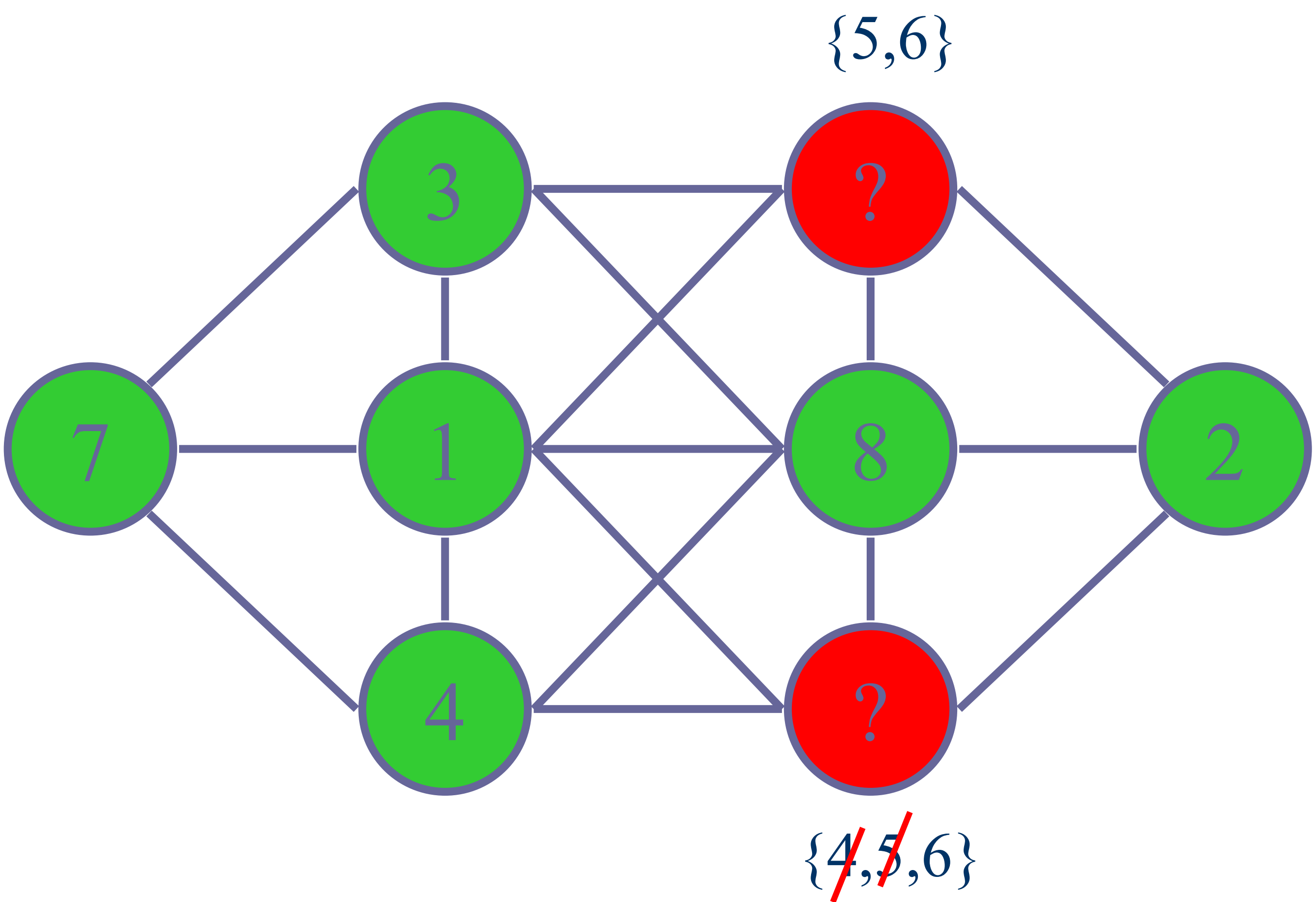
- Guess a value for a variable!



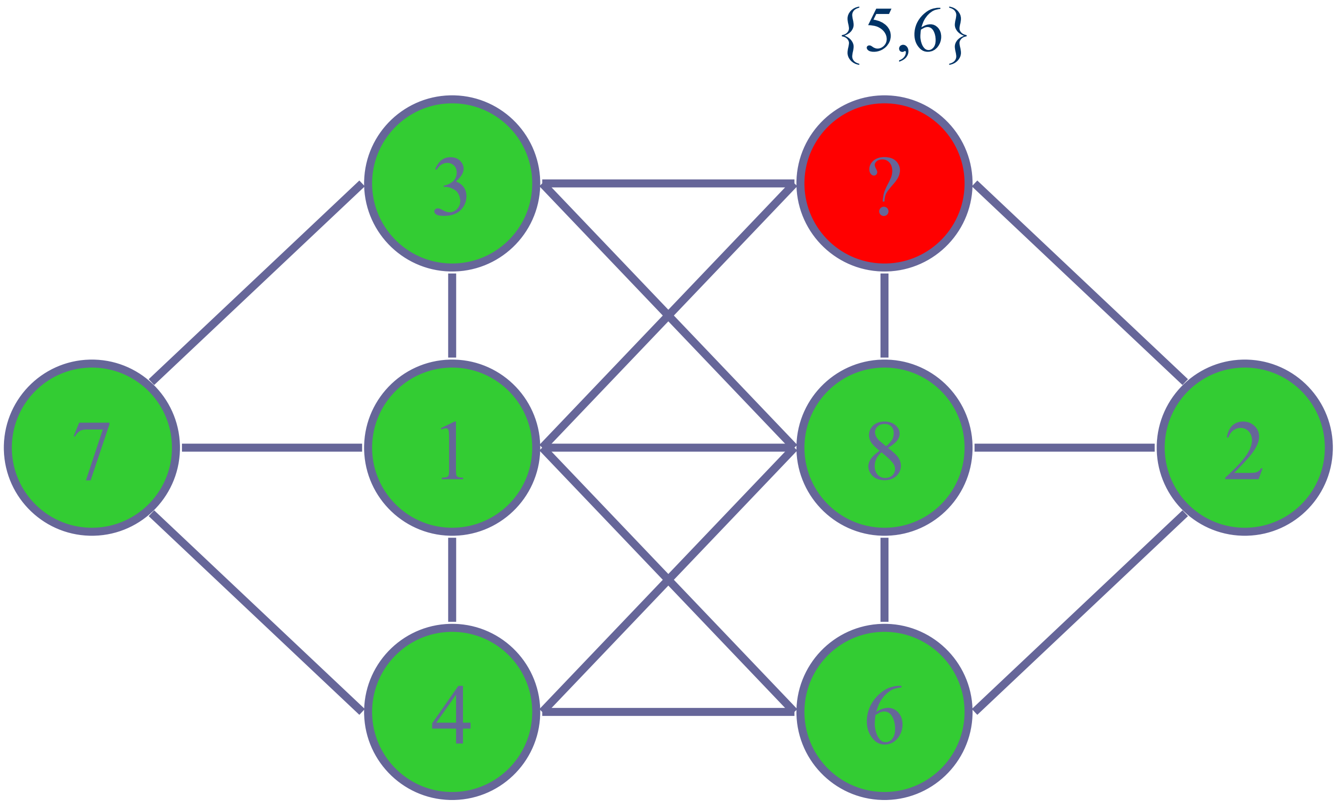
Backtracking Search + Heuristics



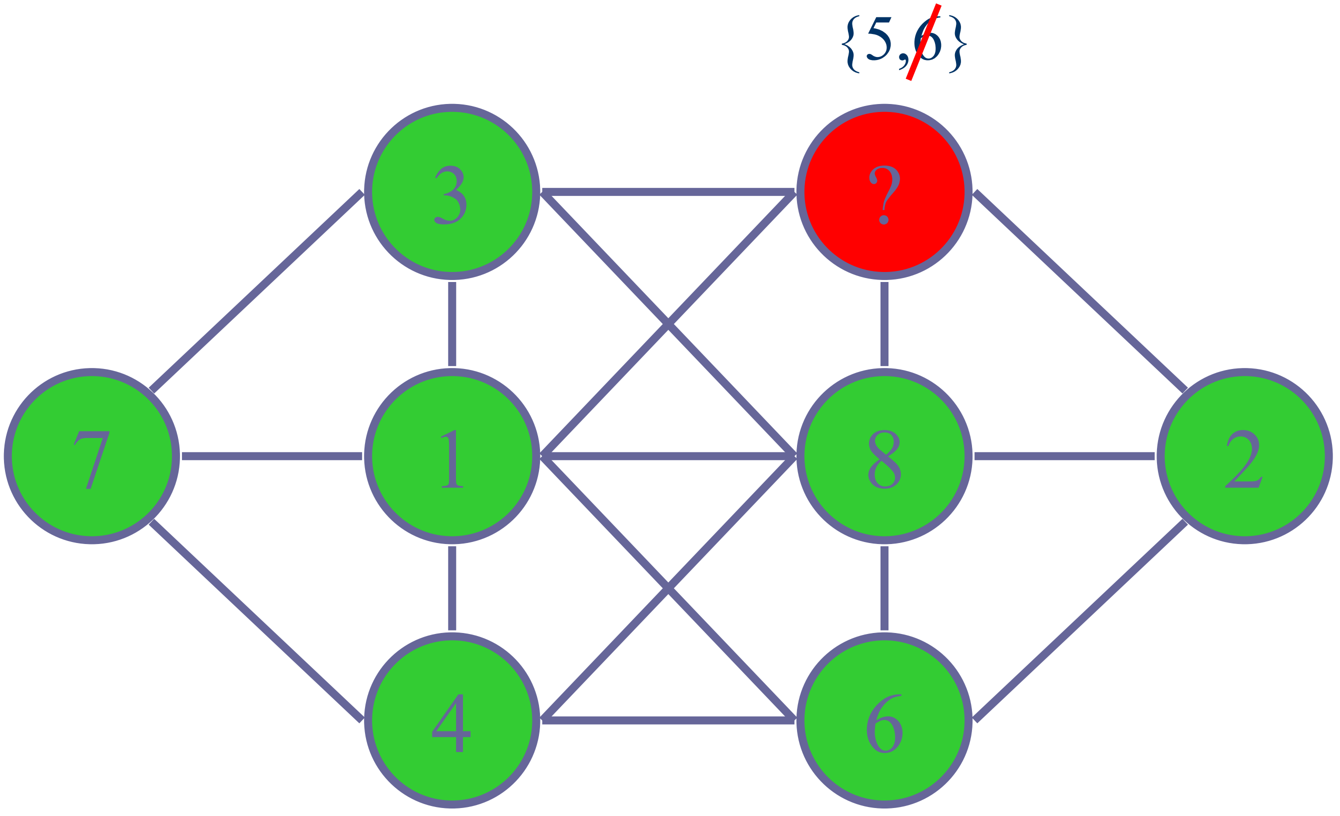
Propagation



Propagation

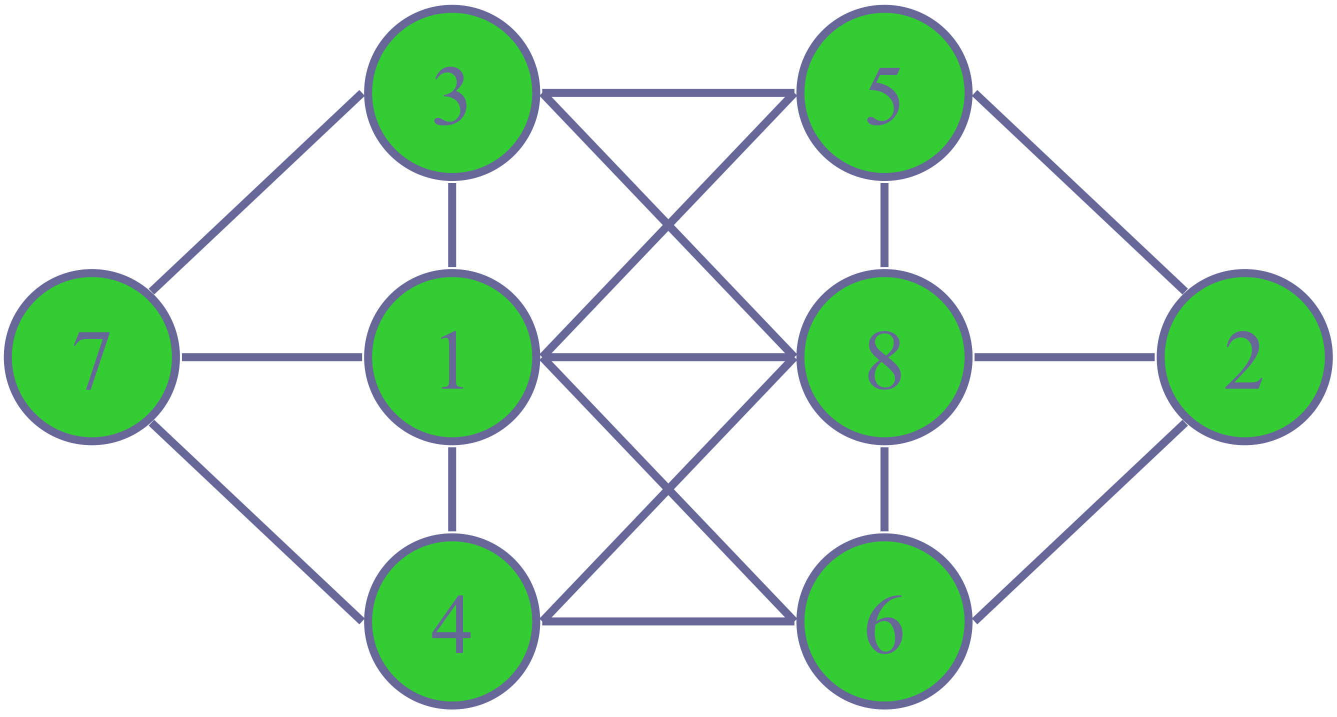


Propagation

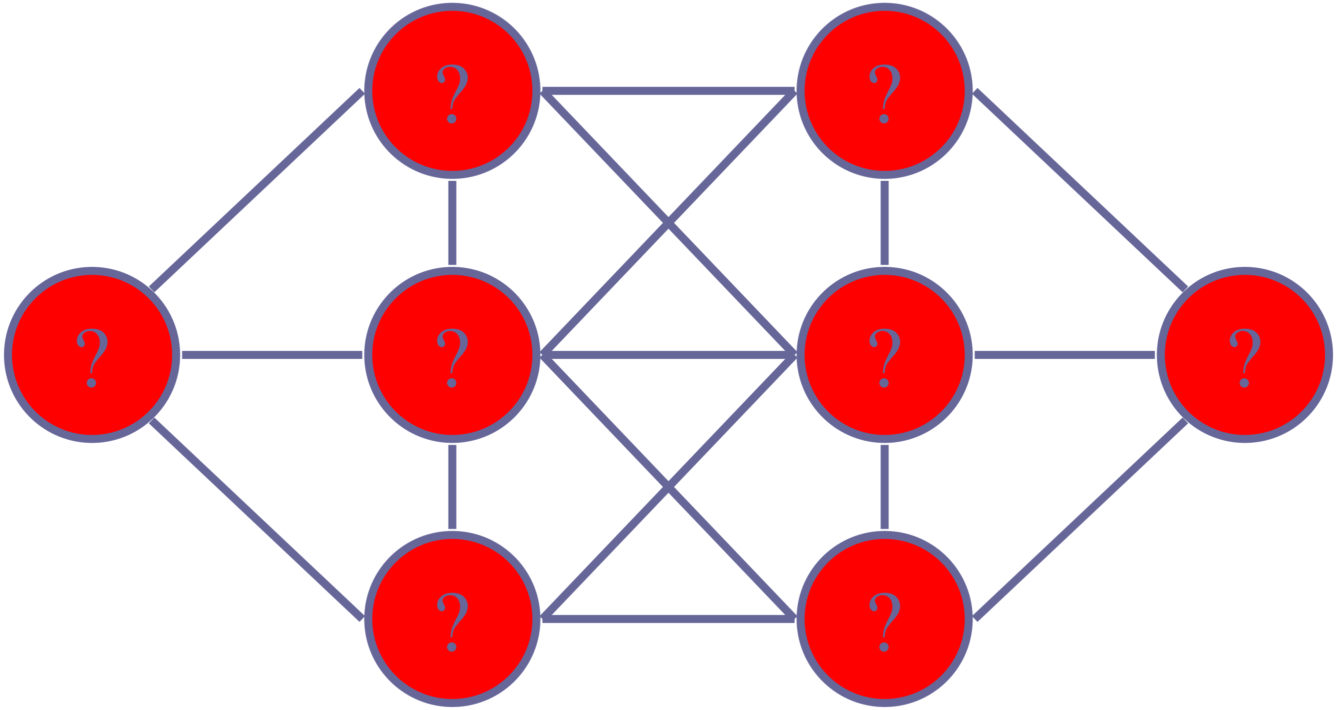


Solution

- 8 guesses, without any backtracking!

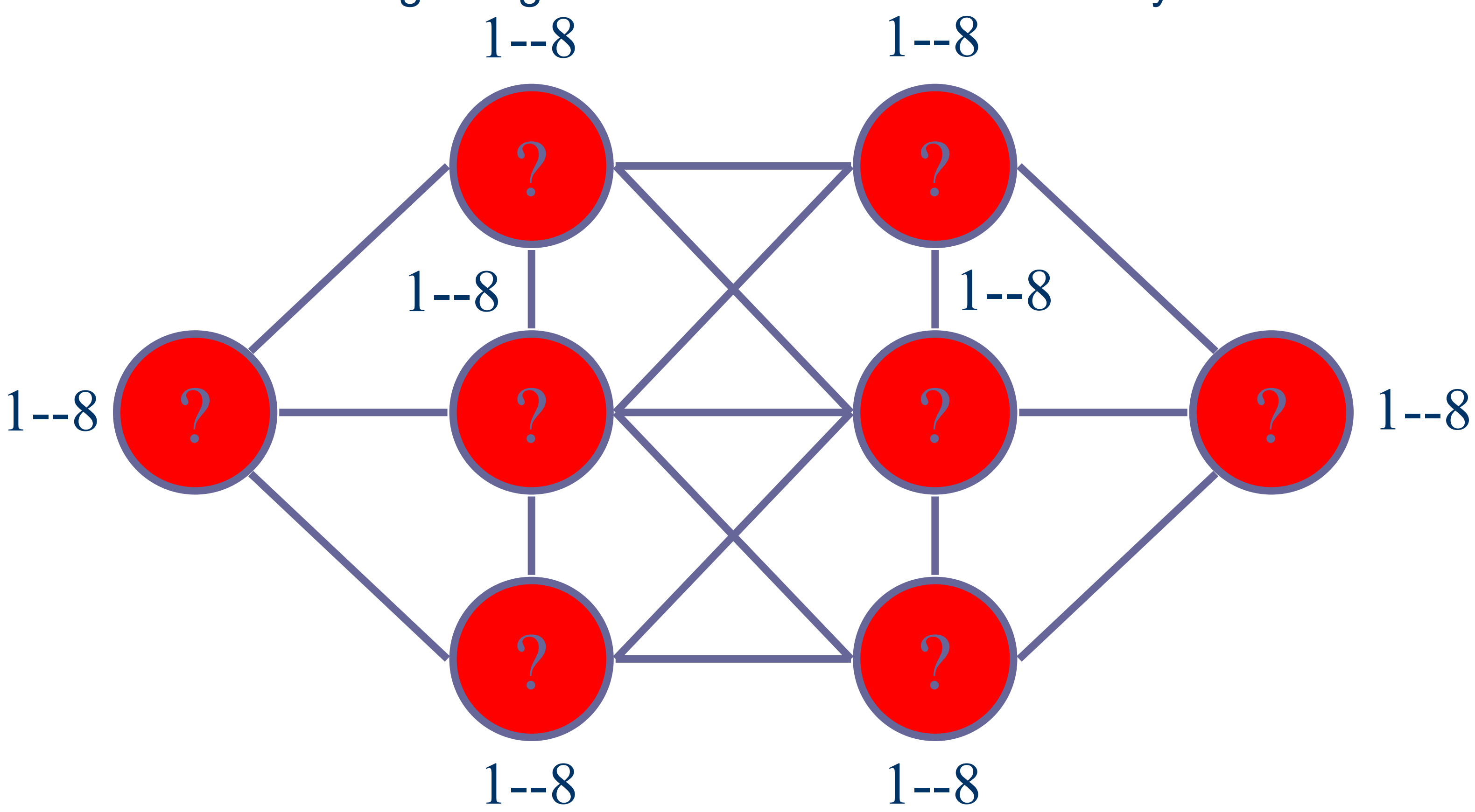


Backtracking Search without Heuristics



Backtracking

- Back to the beginning after 45 backtracks without any solution ☹

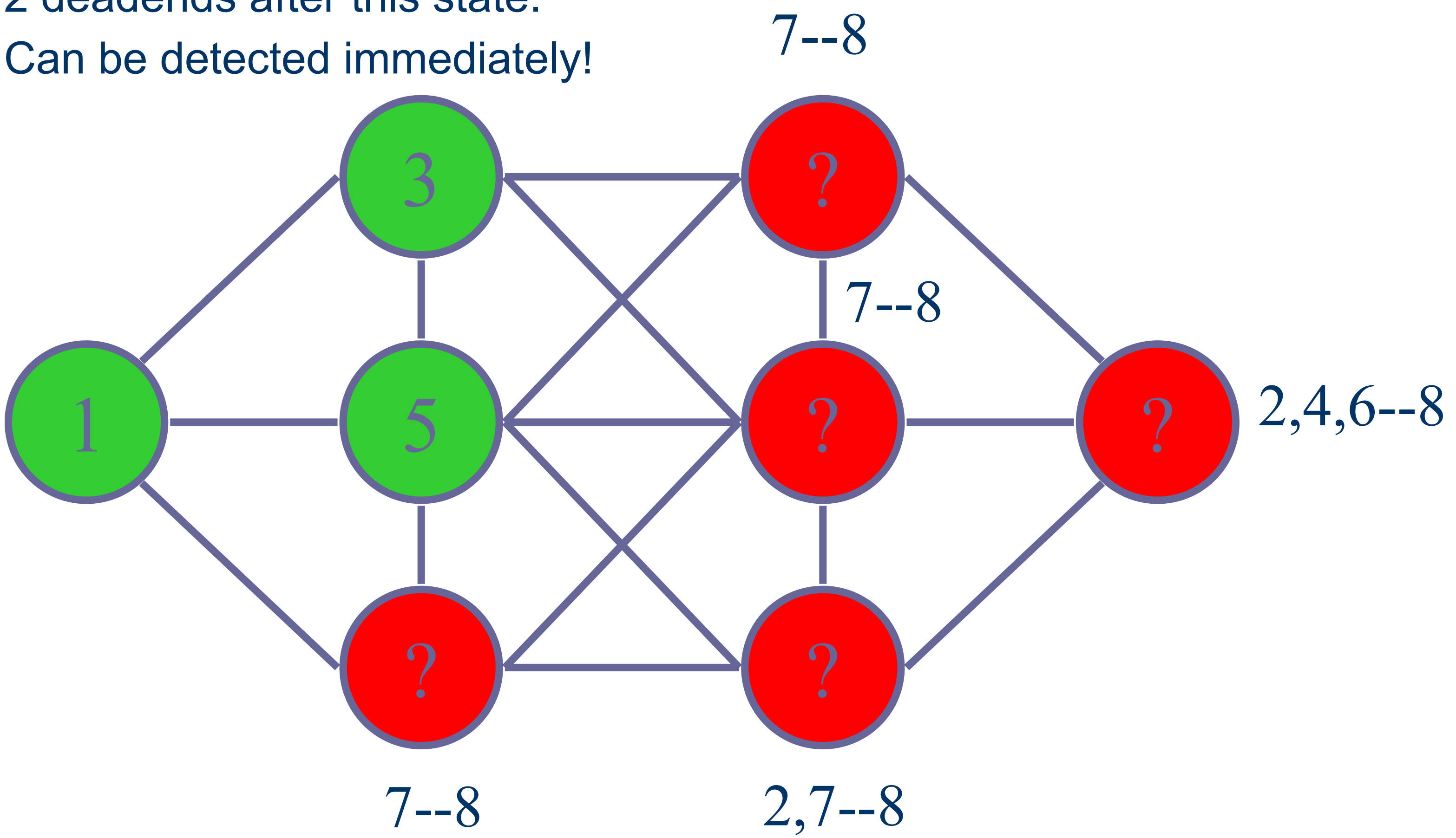


What’s going on?

- Bad choice of variables, bad assignment of values.
  → Good heuristic choice is very important!
- Good heuristics are always possible?
  - Yes and no 🙃
- What can we do then?
  - Apply stronger form of propagation during search!

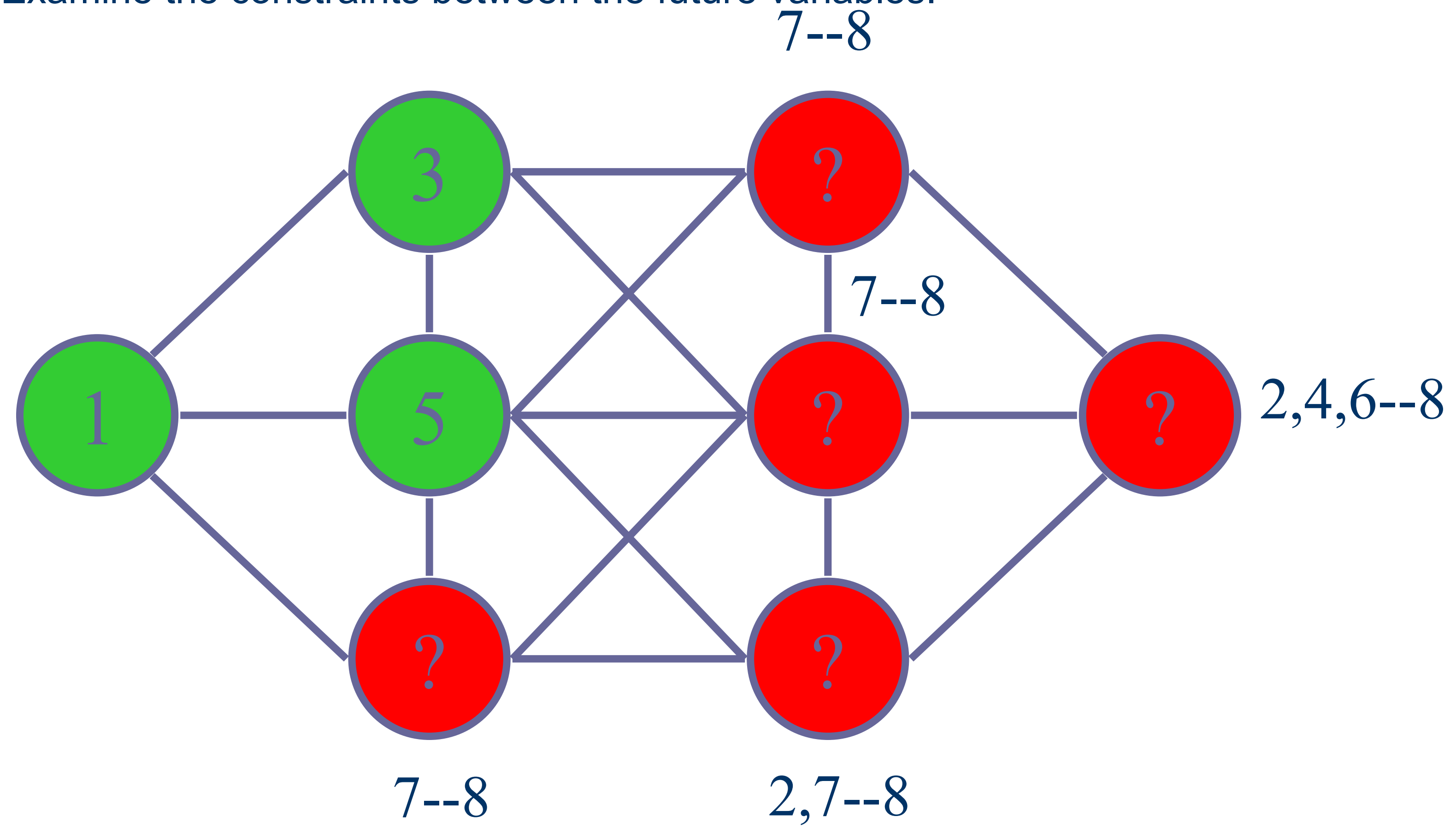
A State During Search

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



A State During Search

- Examine the constraints between the future variables.

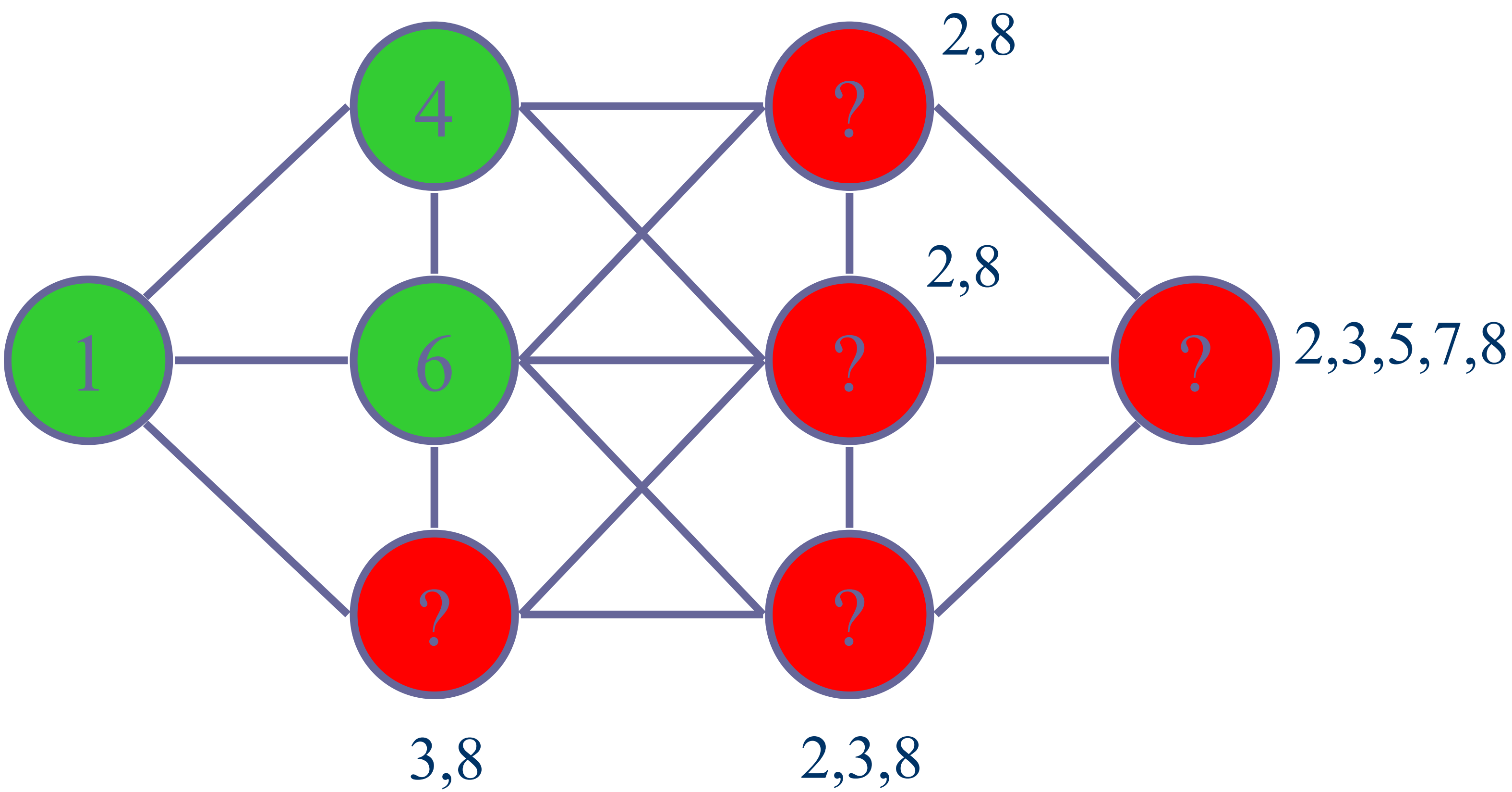


What’s going on?

- Bad choice of variables, bad assignment of values.
  - → Good heuristic choice is very important!
- Good heuristics are always possible?
  - Yes and no 🙃
- What can we do then?
  - Apply stronger form of propagation during search!
- Is that all?
  - Better modelling can result in stronger form of propagation.

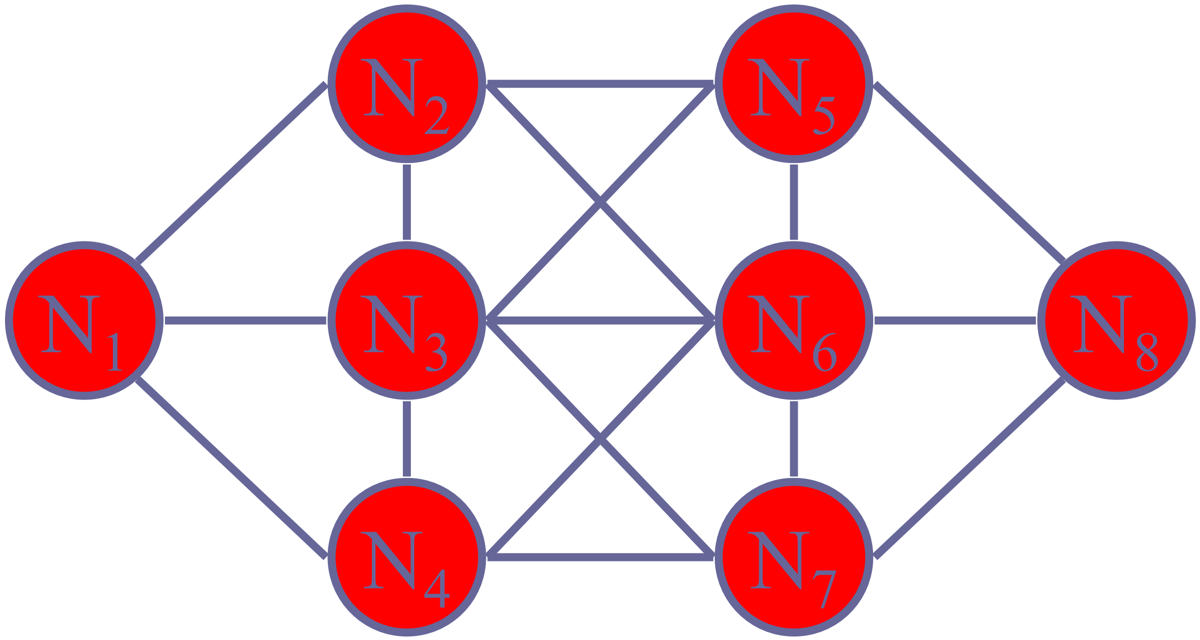
Another State

- Cannot detect the inconsistency of N_3= 6.
  - Future variables are fine wrt the constraints.



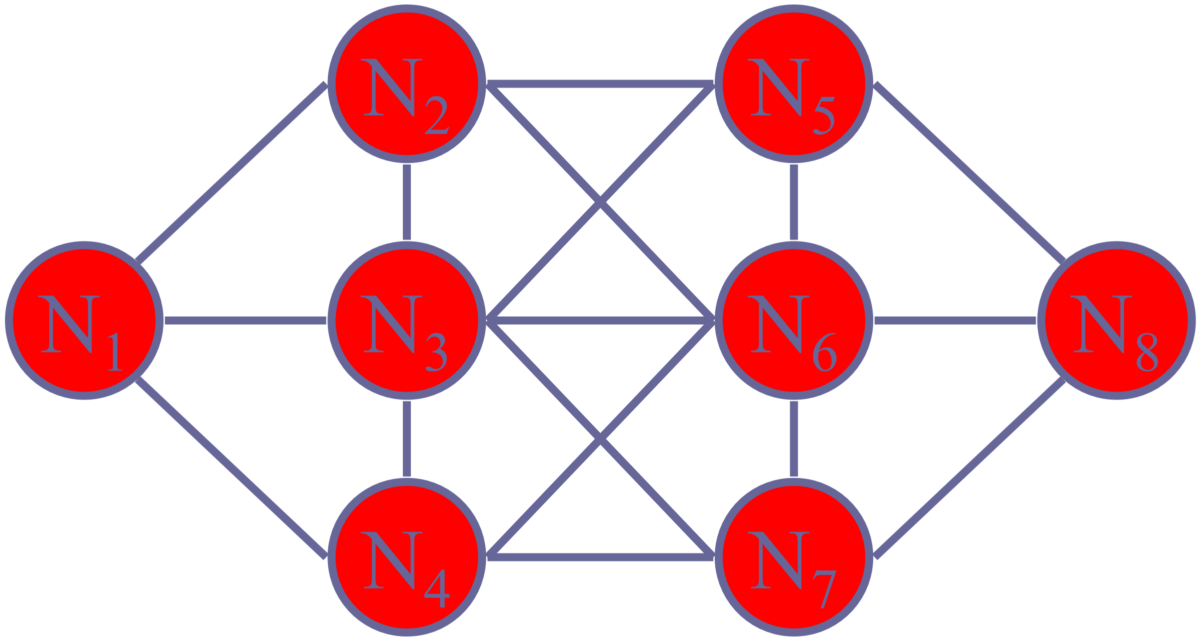
Initial Model

- Constraints:
  - for all $i < j$ s.t. N_i and N_j are adjacent $|N_i - N_j| > 1$
  - for all $i < j$ $N_i \neq N_j$



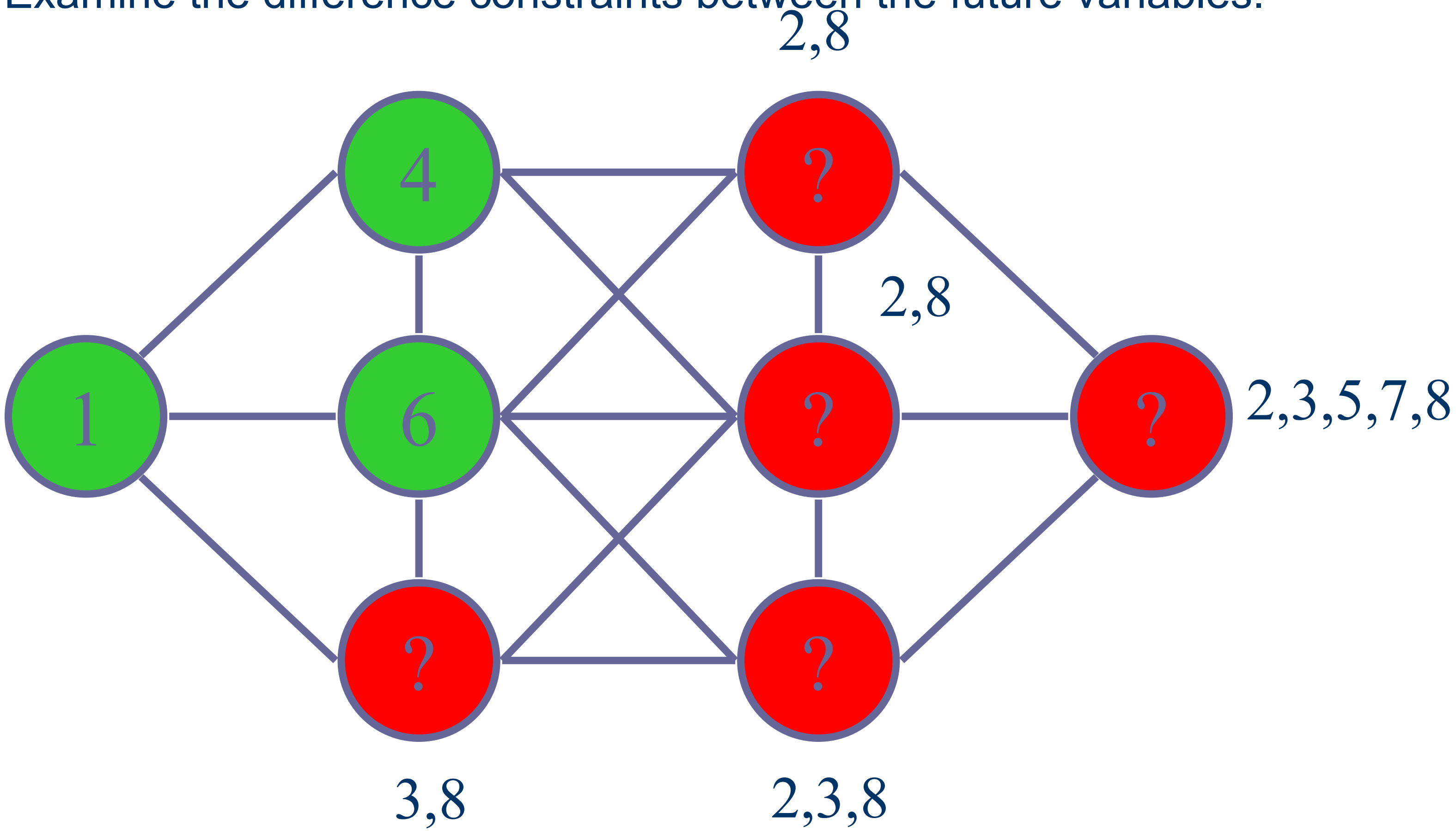
Better Model

- Constraints:
  - for all $i < j$ s.t. N_i and N_j are adjacent $|N_i - N_j| > 1$
  - alldifferent($[N_1, N_2, N_3, N_4, N_5, N_6, N_7, N_8]$)



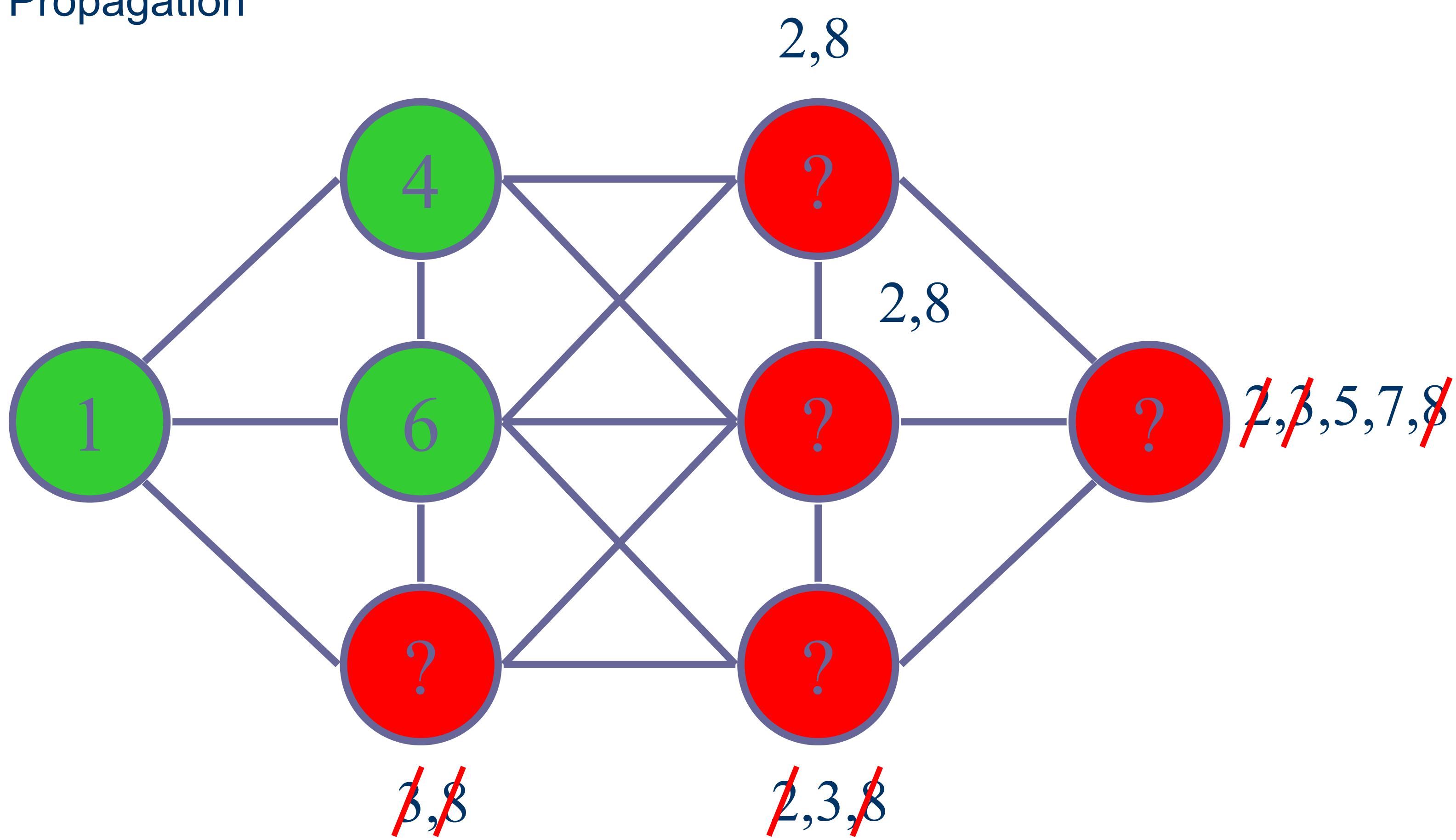
Another State

- Examine the difference constraints between the future variables.



Another State

- Propagation



Constraint Programming

- For an efficient CP solving, we need:
  - effective and efficient constraint propagation algorithms;
  - a model with such constraints;
  - effective and efficient search algorithm and heuristics.
- Attention!
  - Intelligent reasoning comes with a cost.
  - Need a good balance.

Constraint Programming

- Declarative programming:
  - the user models the problem;
  - an underlying search-based solver returns a solution.
- Computer programming:
  - the user needs to program a strategy to search for a solution
    - search algorithm, heuristics, …
  - otherwise, solving process can be inefficient.