Stochastic Constraint Programming and Global Stochastic Constraints

A modelling and solution framework for effective decision making under uncertainty

Dr. Roberto Rossi

The University of Edinburgh Business School
The University of Edinburgh
roberto.rossi@ed.ac.uk

Friday, November the 29th, 2013

What is Constraint Programming (CP)

A subfield of **Artificial Intelligence**

It represents an **efficient means for finding solutions** to combinatorial problems in areas such as:

- timetabling
- scheduling
- design & configuration
- ...

Closely related to a number of disciplines:

- logic programming & inference
- graph theory
- combinatorial optimization
- ...

Who cares about Constraints

- IBM acquired ILOG in 2009, a leading vendor of constraint technology.
- CISCO acquired the ECLiPSe constraint logic programming system.
- St Andrews Minion solver is used to schedule the CB1000 Nanoproteomic Analysis System `dx.doi.org/10.1007/978-3-642-04244-7_8`.
- In Scotland there is a fairly large academic community: there are constraint programers in Edinburgh, Glasgow, St Andrews and Dundee.

Constraints

Constraints represent a natural **means of knowledge representation**.

Examples

"x+y=30"

"The helicopter can carry one passenger"

University timetabling:

- "no student can attend two lectures at once"
- "George Square lecture theatre has a capacity of 400 students"
- "art history lectures require a slide projector"

Constraints

Constraint programming offers many **expressive constraints** that can be used to build a constraint model, e.g.

$$\text{allDifferent}(x_1, \ldots, x_n)$$

Global constraint catalog

The catalogue presents a list of 354 global constraints issued from the literature in constraint programming and from popular constraint systems.

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

Solving problems with constraints

Two phases:

1. **Describe the problem** to be solved as a **constraint model**, a format suitable for input to a constraint solver.
2. **Search** (automatically) for solutions to the model with a constraint solver (e.g. IBM ILOG, Choco, ECLiPse,. . .).

The (finite domain) Constraint Satisfaction Problem

Given

- a finite set of **decision variables**
- for each decision variable, a finite **domain** of potential values
- a finite set of **constraints** on the decision variables

Find an **assignment** of values to variables such that all constraints are satisfied.

The (finite domain) Constraint Satisfaction Problem

Decision variable

A decision variable corresponds to a **choice** that must be made in solving a problem.

Domains

Values in the domain of a decision variable correspond to the **options** for a particular choice.

Constraints

Represent **relations** that identify which assignments of values to variables are allowed.

Constraint solving

Typically interleaves two components

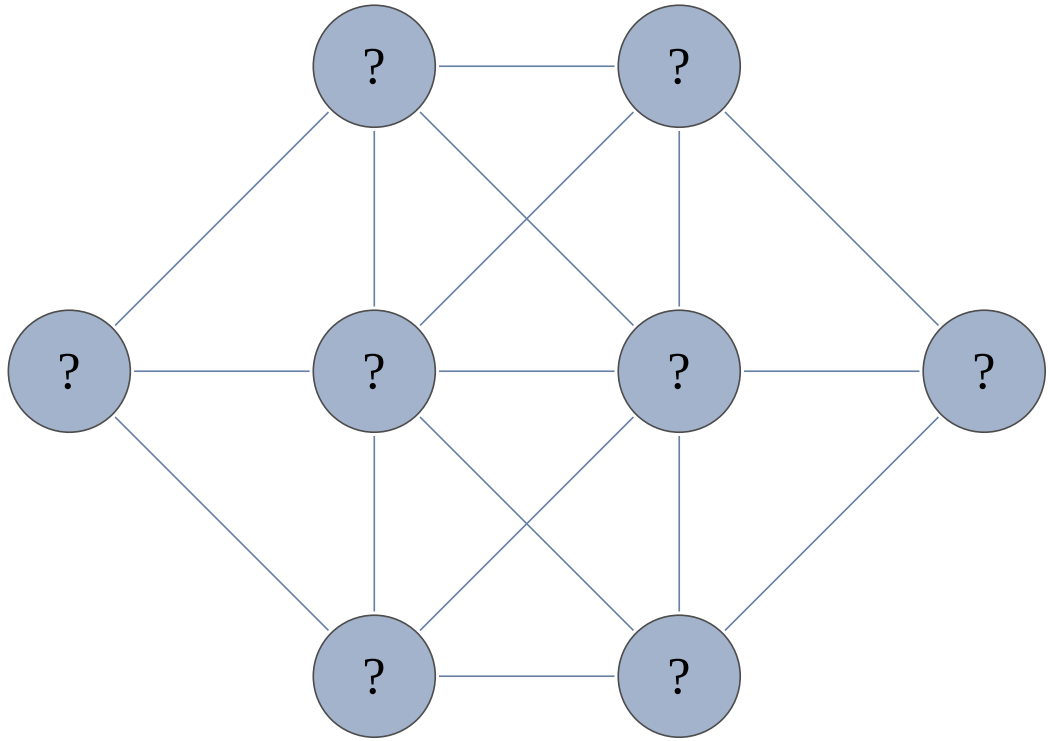
- systematic **search** through a space of partial assignments
  - extend an assignment to a subset of the variables incrementally
  - backtrack (i.e. "undo") if establish that current partial assignment cannot be extended to a solution
- **constraint propagation**
  - logical inference based on constraints and current domains
  - aims to reduce domains

The Crystal Maze Puzzle

Thanks to **Patrick Prosser**
Computing Science
Glasgow University
17 Lilybank Gardens
Glasgow G12 8RZ

Crystal Maze

Given the network



Place numbers $1, \ldots, 8$ on nodes by making sure that

- each number appears exactly once
- consecutive numbers do not appear on adjacent nodes

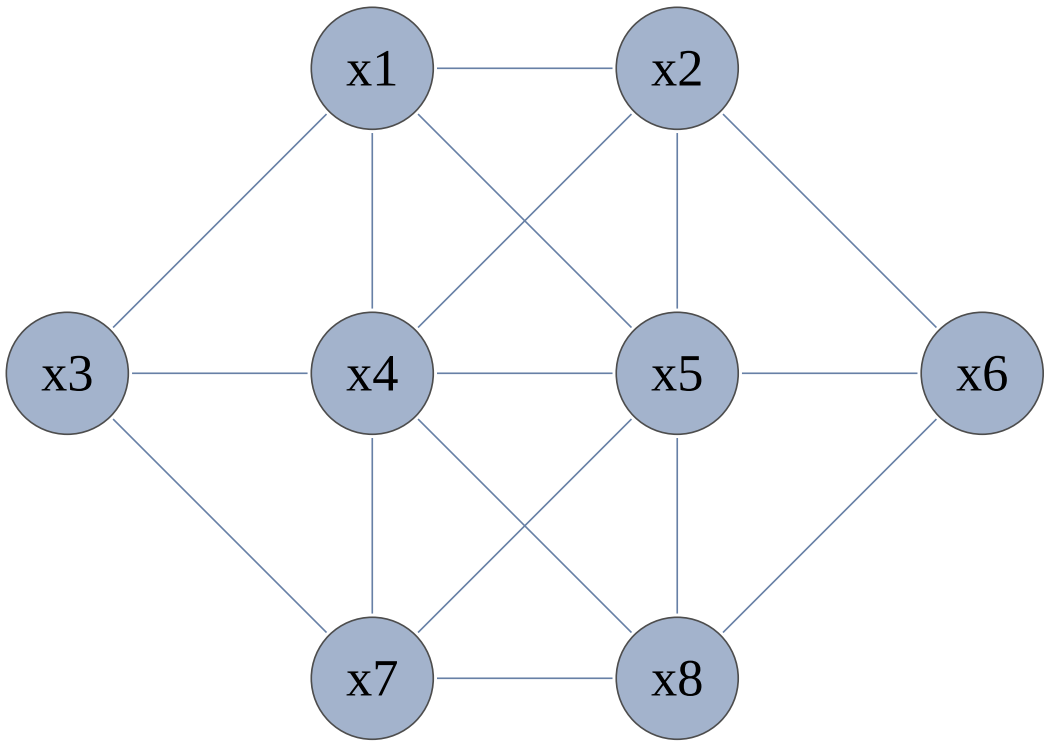
Crystal Maze as a CSP

The Crystal Maze puzzle can be modelled as a CSP.

A constraint solver can then make the inferences you just performed by hand, automatically.

Crystal Maze as a CSP

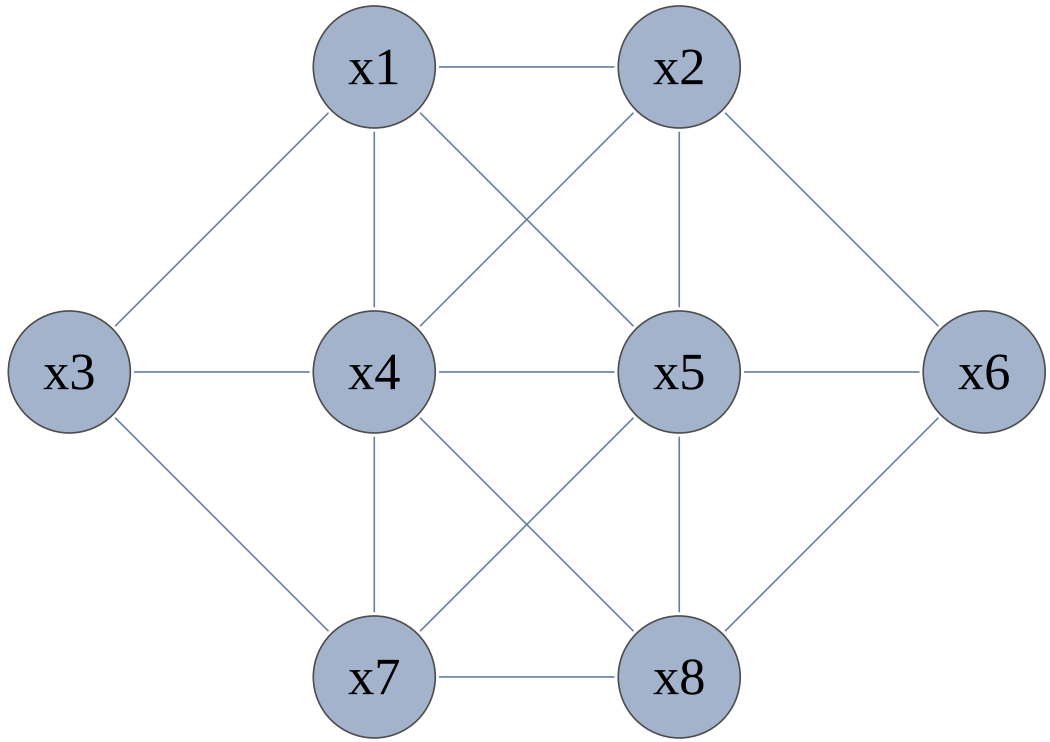
Variables



The indexing adopted is irrelevant.

Crystal Maze as a CSP

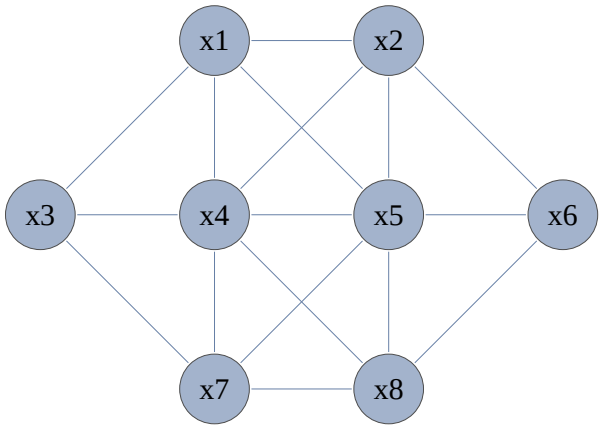
Domains



All variables have the same domain: $\{1, 2, 3, 4, 5, 6, 7, 8\}$.

Crystal Maze as a CSP

Constraints



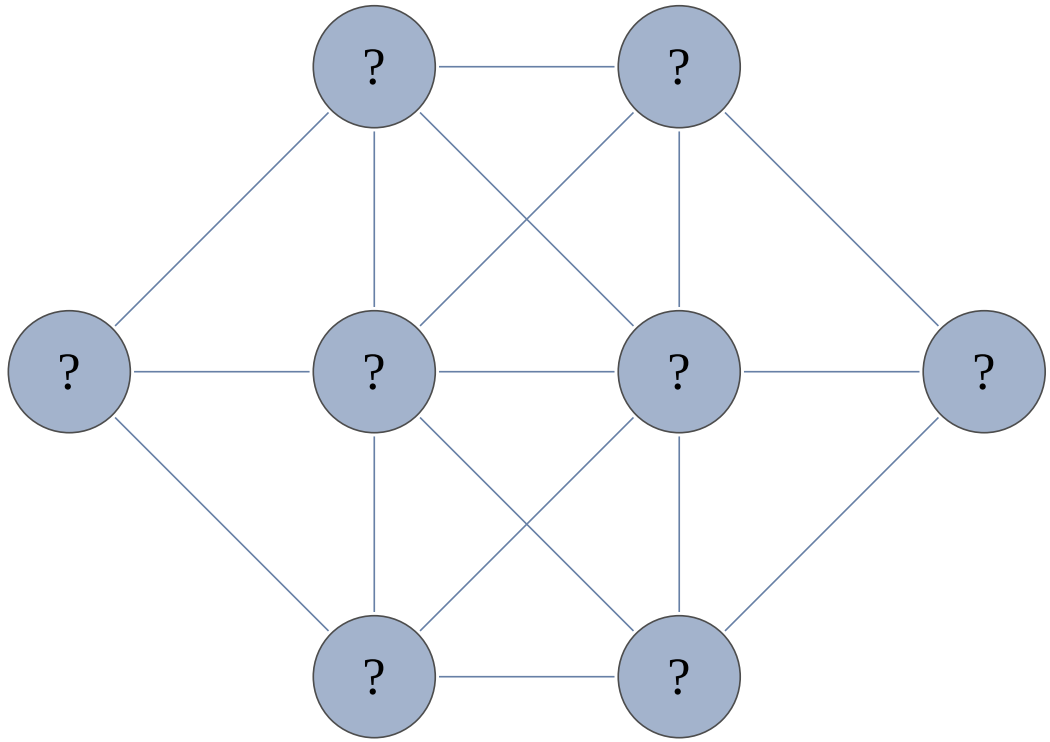
- All values in the domains should be used: alldifferent$(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$
- No consecutive numbers on adjacent nodes:
  $|x_1 - x_2| > 1$
  $|x_1 - x_3| > 1$
  ...

Solving the Crystal Maze Puzzle

But how would a constraint solver reason about it?

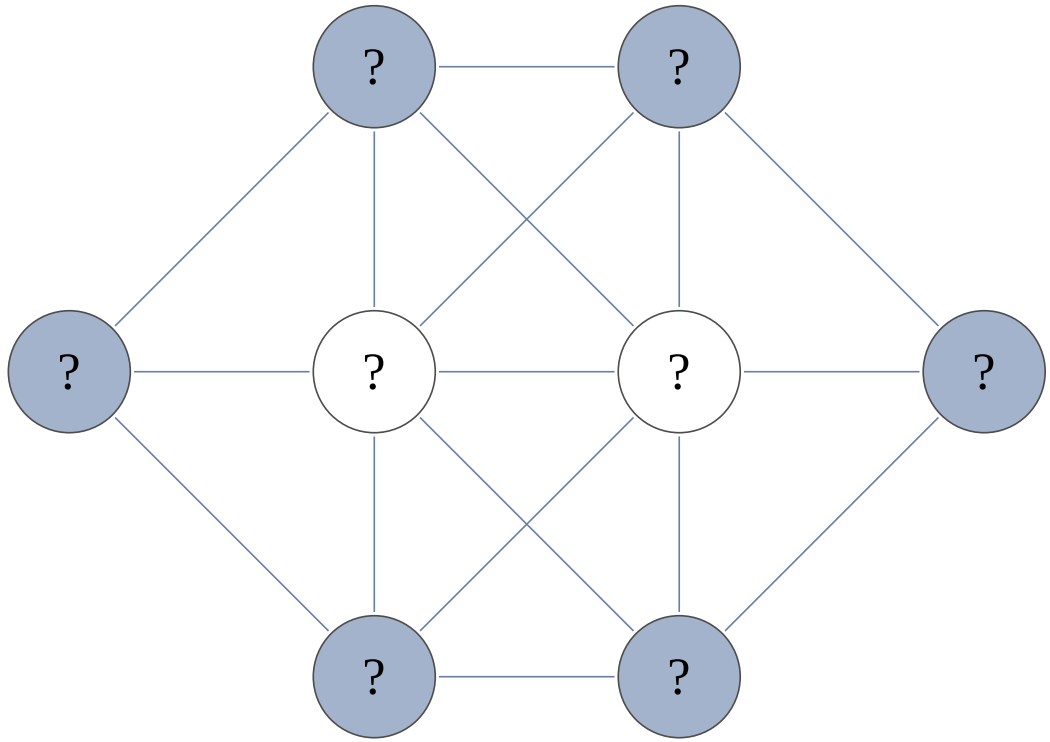
Solving the Crystal Maze Puzzle

Which nodes are the hardest to label?



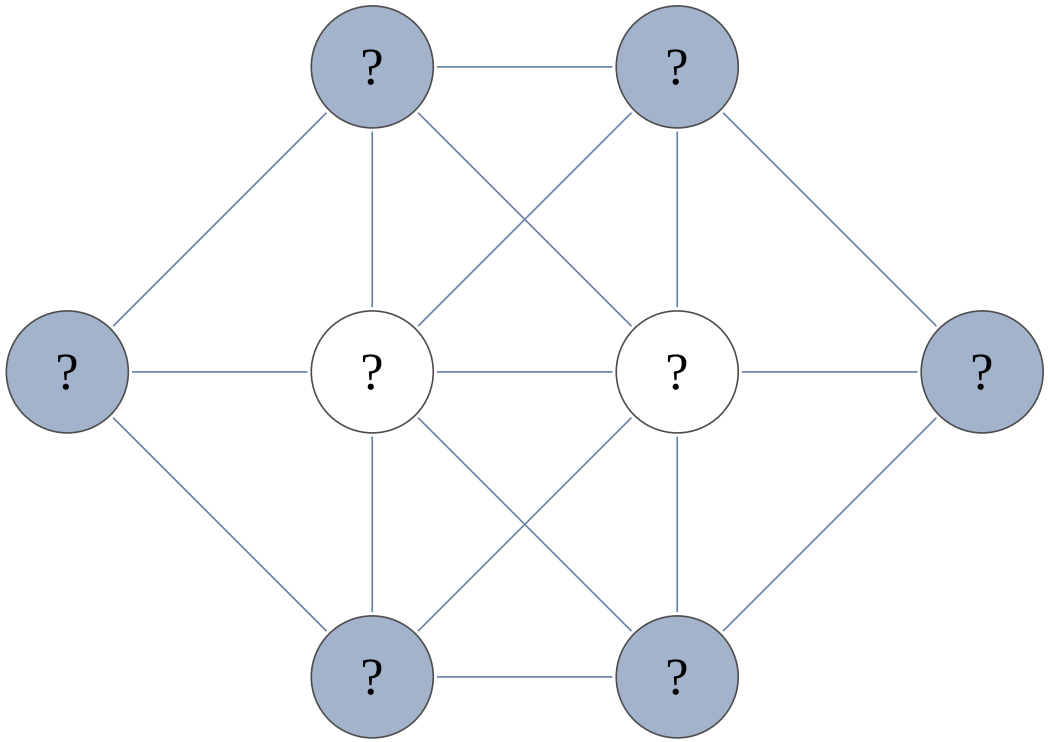
Solving the Crystal Maze Puzzle

Which nodes are the hardest to label?



Solving the Crystal Maze Puzzle

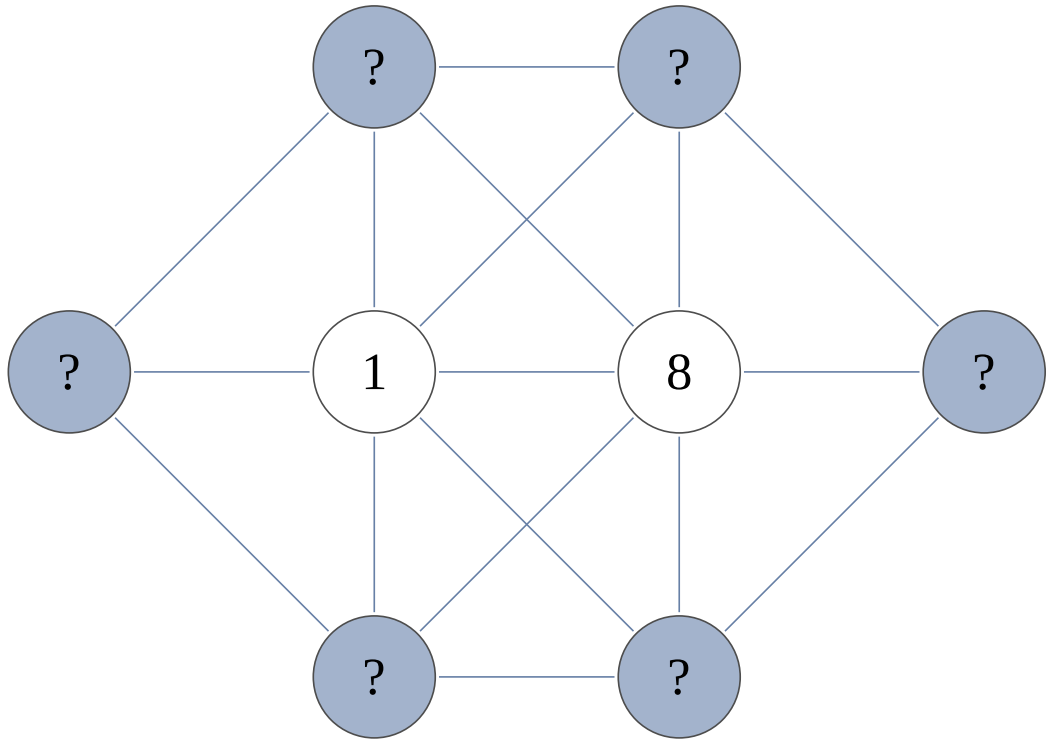
Which are the least constraining values to use?



Recall that domains are $\{1, 2, 3, 4, 5, 6, 7, 8\}$.

Solving the Crystal Maze Puzzle

Which are the least constraining values to use?

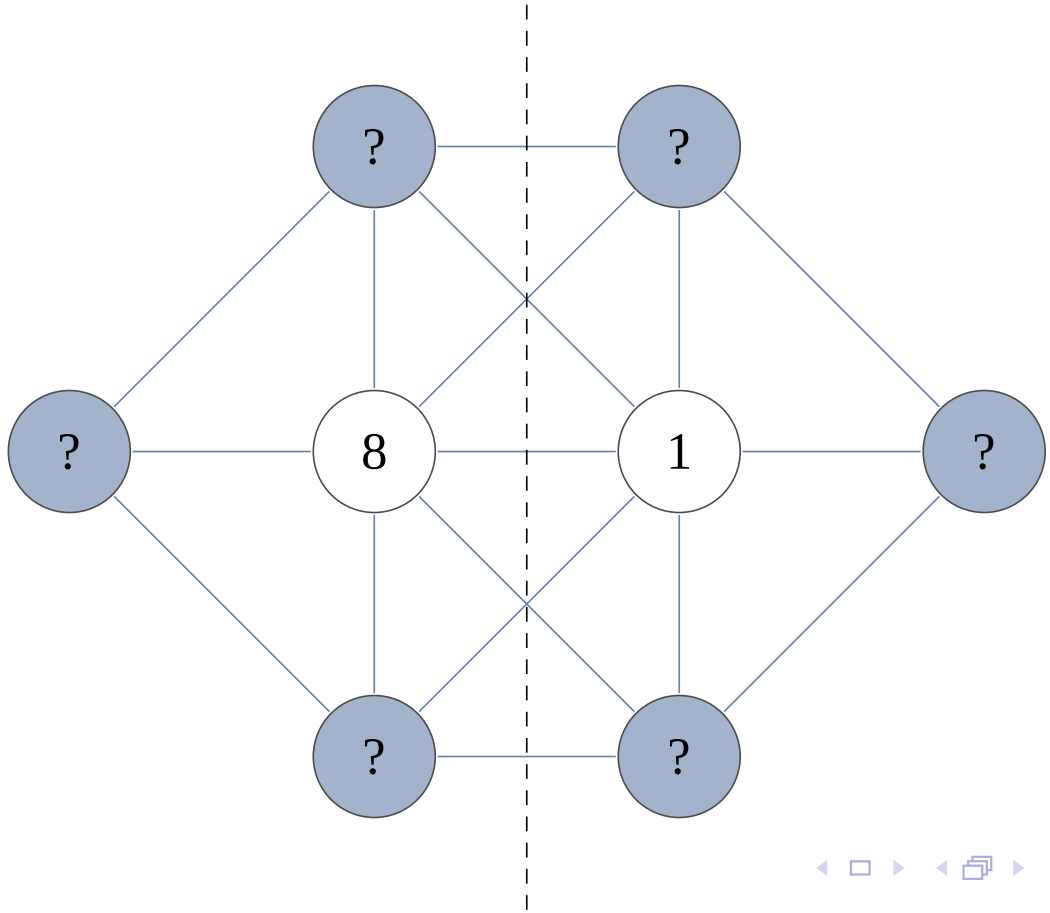


Recall that domains are $\{1, 2, 3, 4, 5, 6, 7, 8\}$.

Solving the Crystal Maze Puzzle

Exploit symmetry!

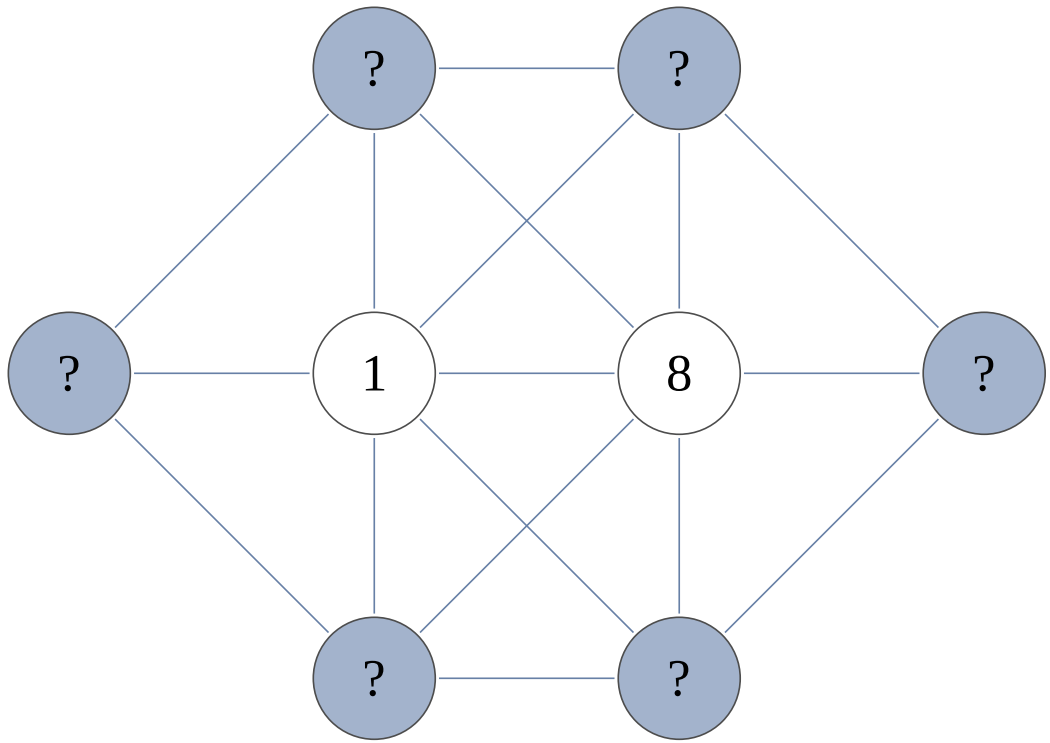
It would be unnecessary to consider the following symmetric assignment.



Solving the Crystal Maze Puzzle

Constraint propagation

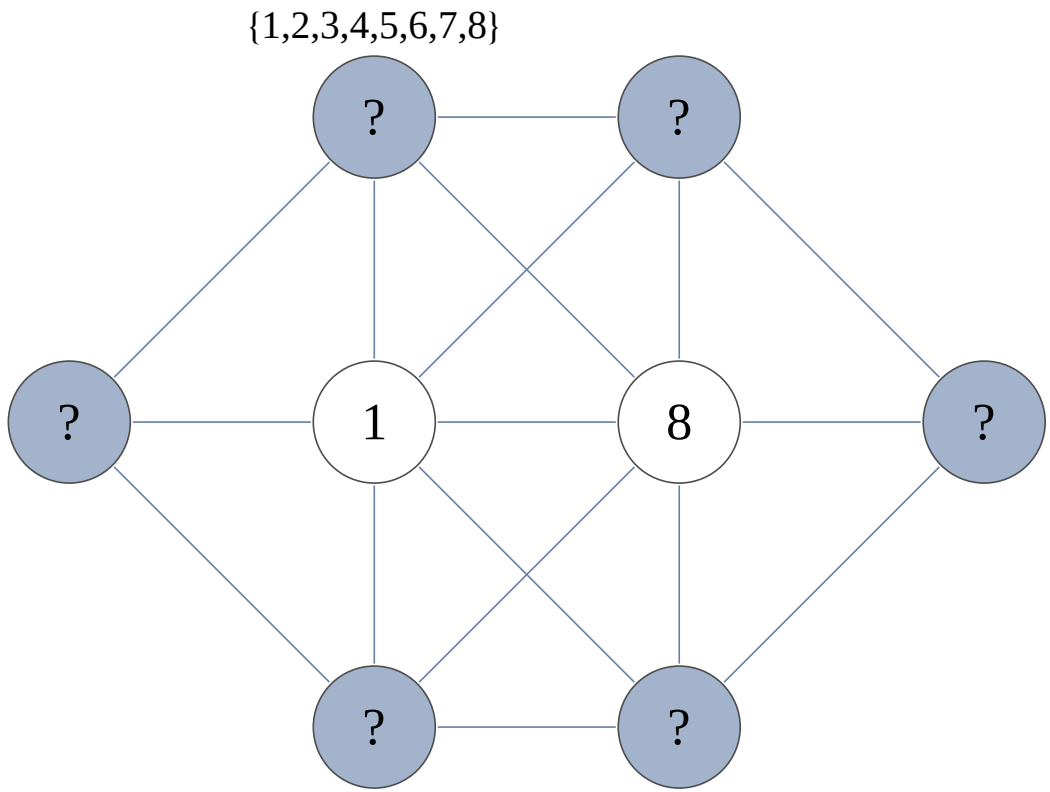
Exploit constraint propagation to eliminate infeasible values from domains associated with other nodes (i.e. decision variables).



Solving the Crystal Maze Puzzle

Constraint propagation

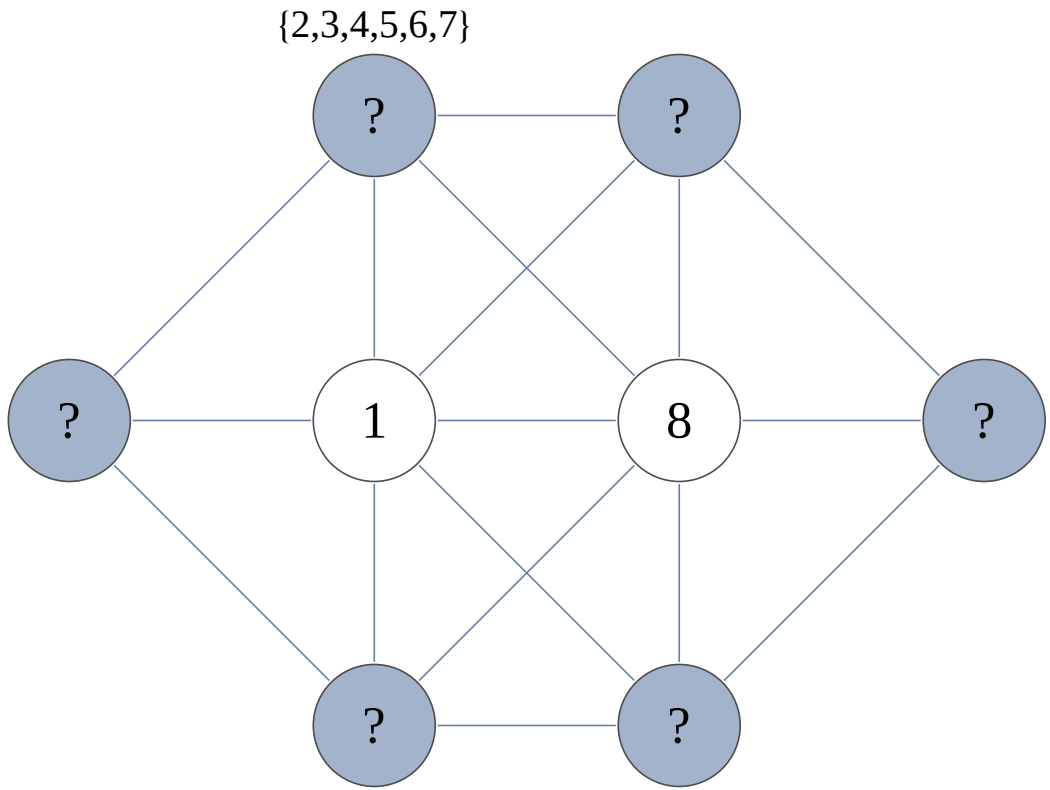
Exploit constraint propagation to eliminate infeasible values from domains associated with other nodes (i.e. decision variables).



Solving the Crystal Maze Puzzle

Constraint propagation

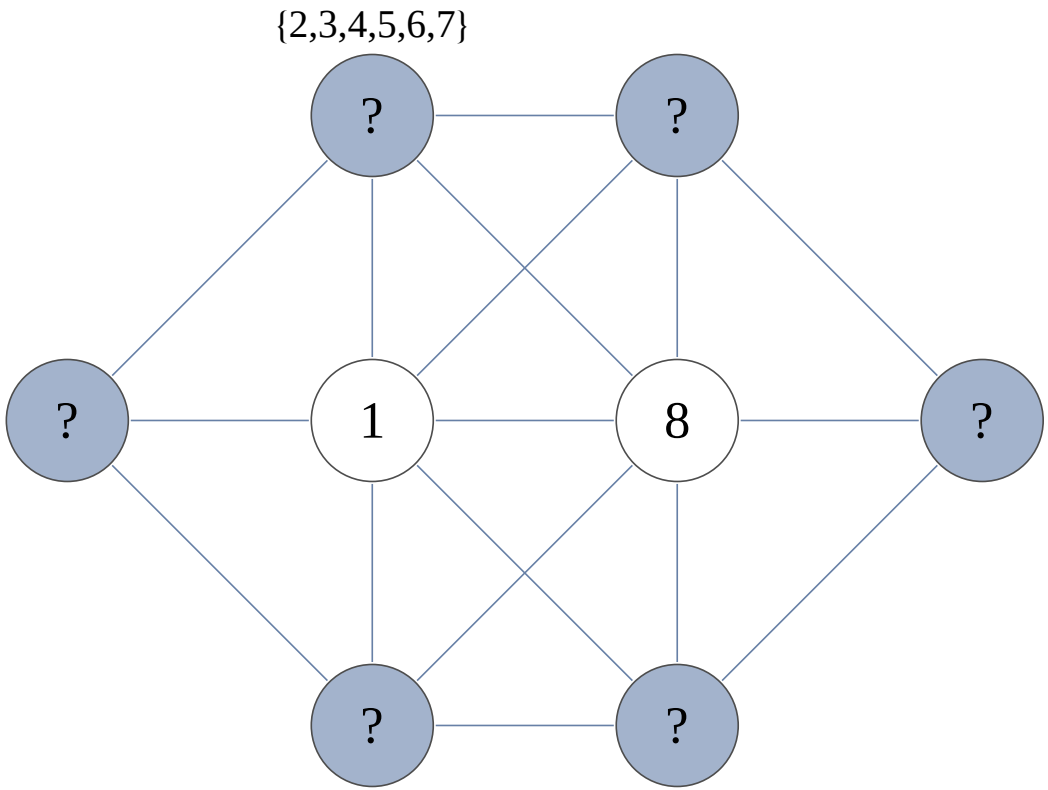
Values 1 and 8 appear exactly once.



Solving the Crystal Maze Puzzle

Constraint propagation

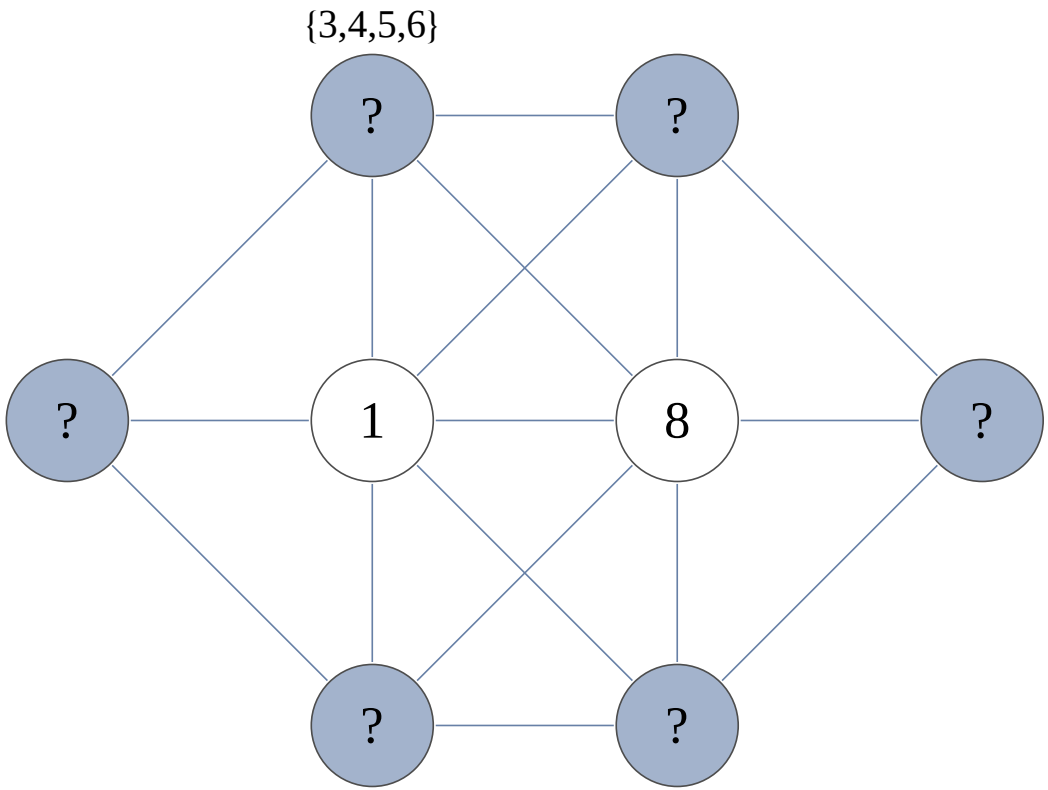
Enforce adjacency restrictions.



Solving the Crystal Maze Puzzle

Constraint propagation

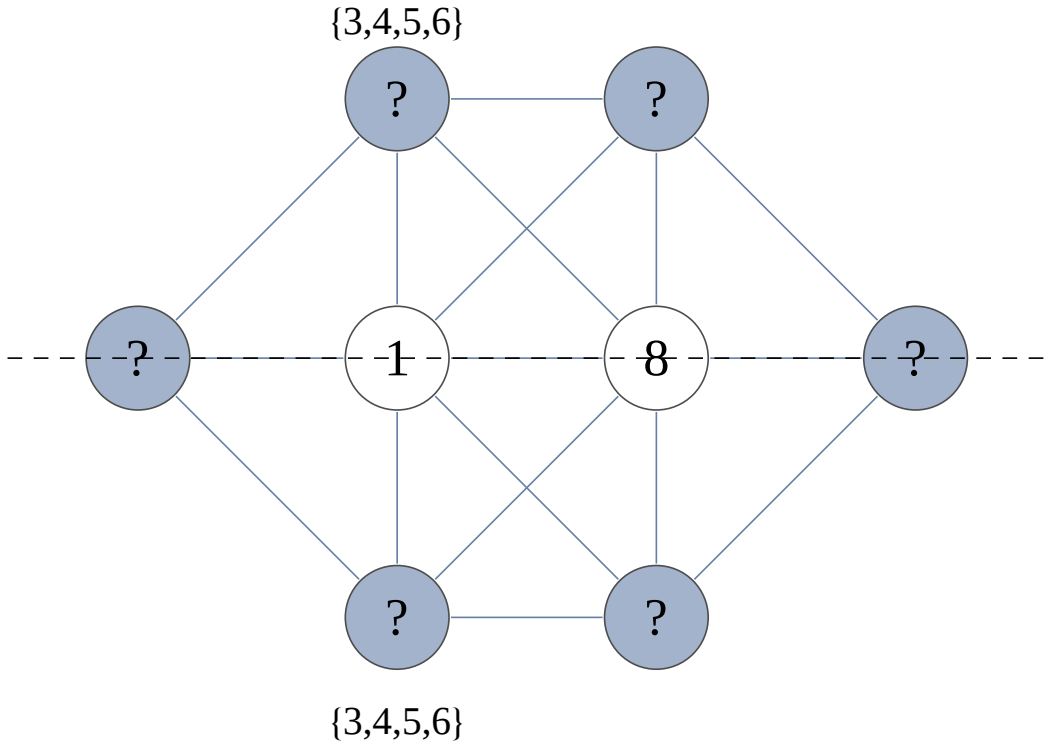
Enforce adjacency restrictions.



Solving the Crystal Maze Puzzle

Constraint propagation

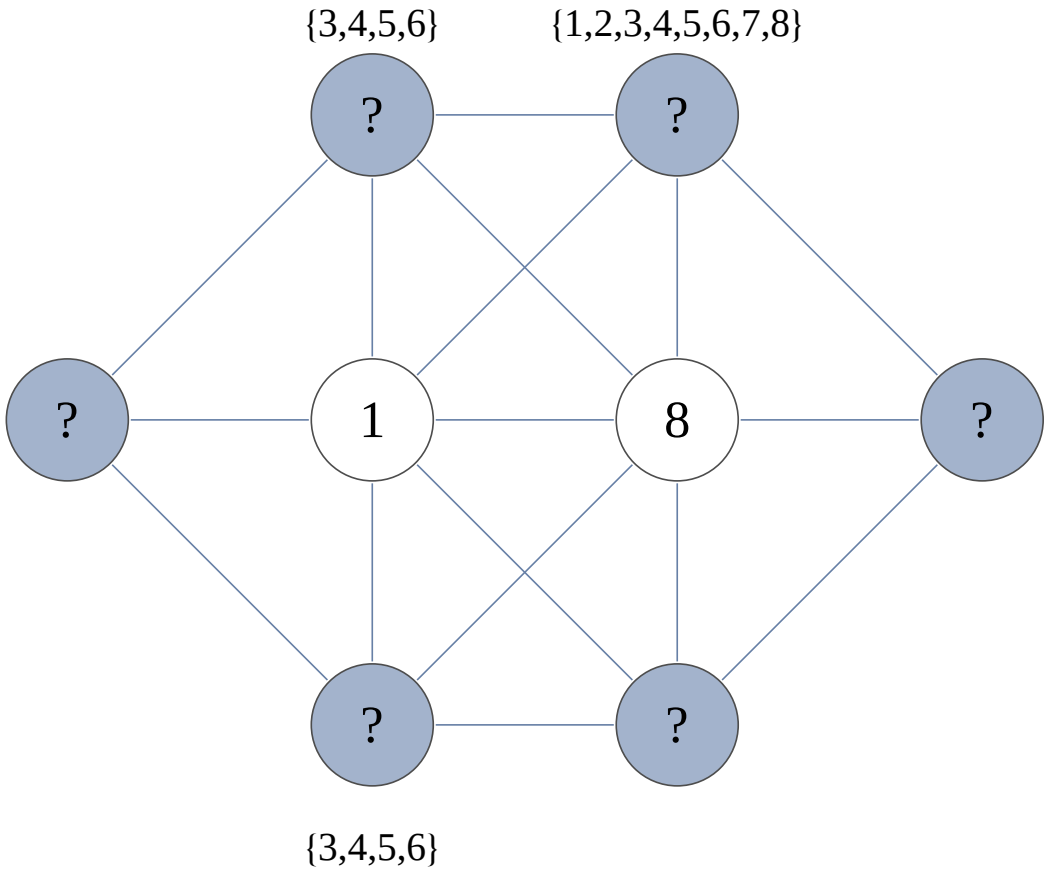
Symmetry again. . .



Solving the Crystal Maze Puzzle

Constraint propagation

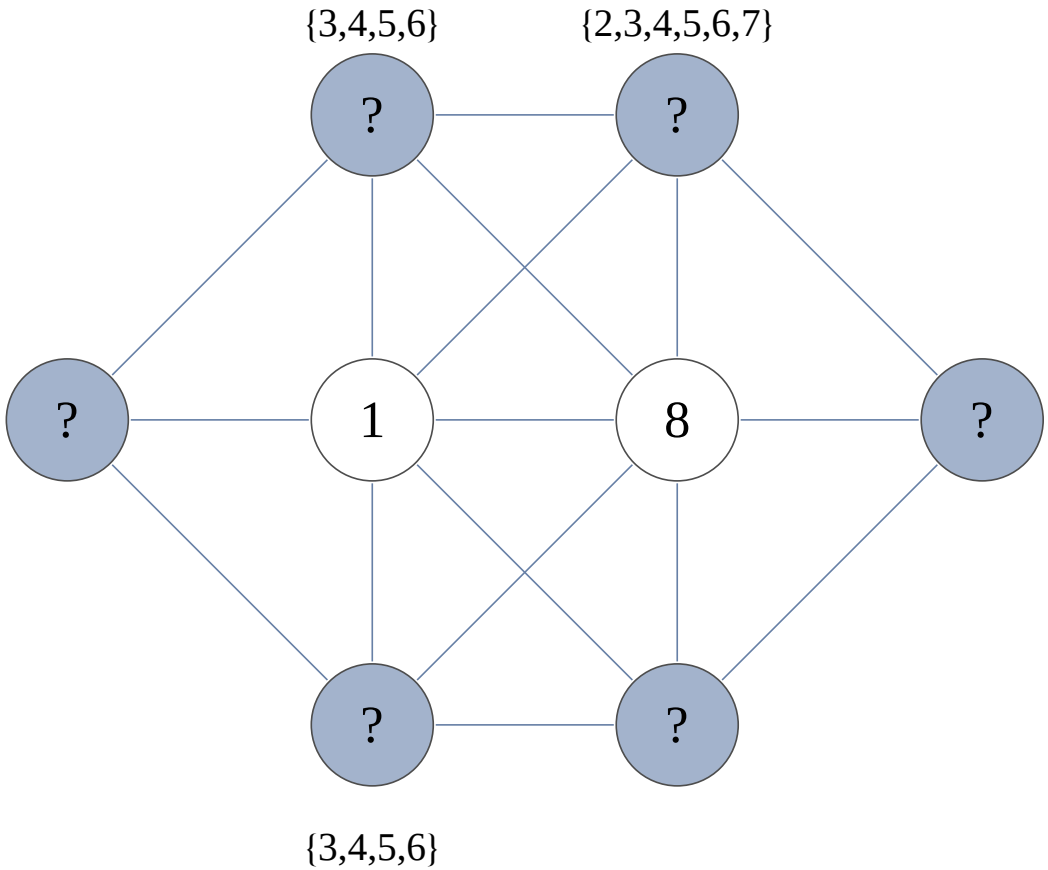
Values 1 and 8 appear exactly once.



Solving the Crystal Maze Puzzle

Constraint propagation

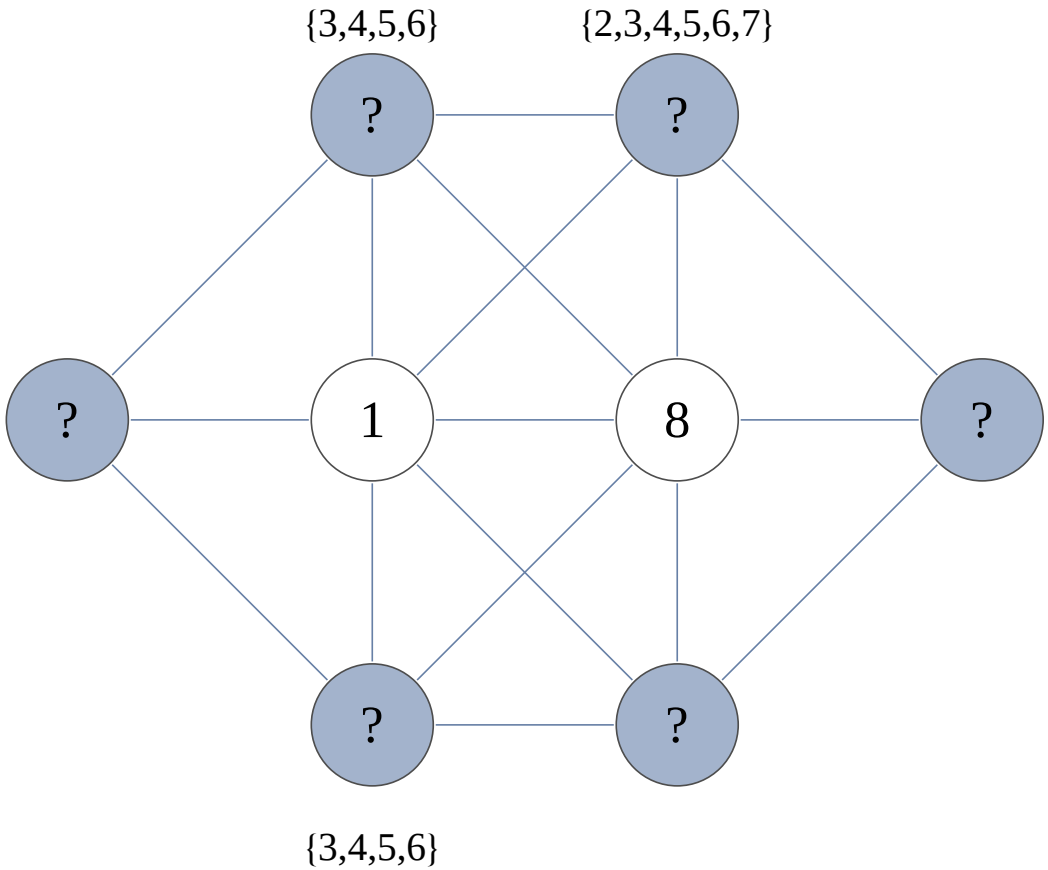
Values 1 and 8 appear exactly once.



Solving the Crystal Maze Puzzle

Constraint propagation

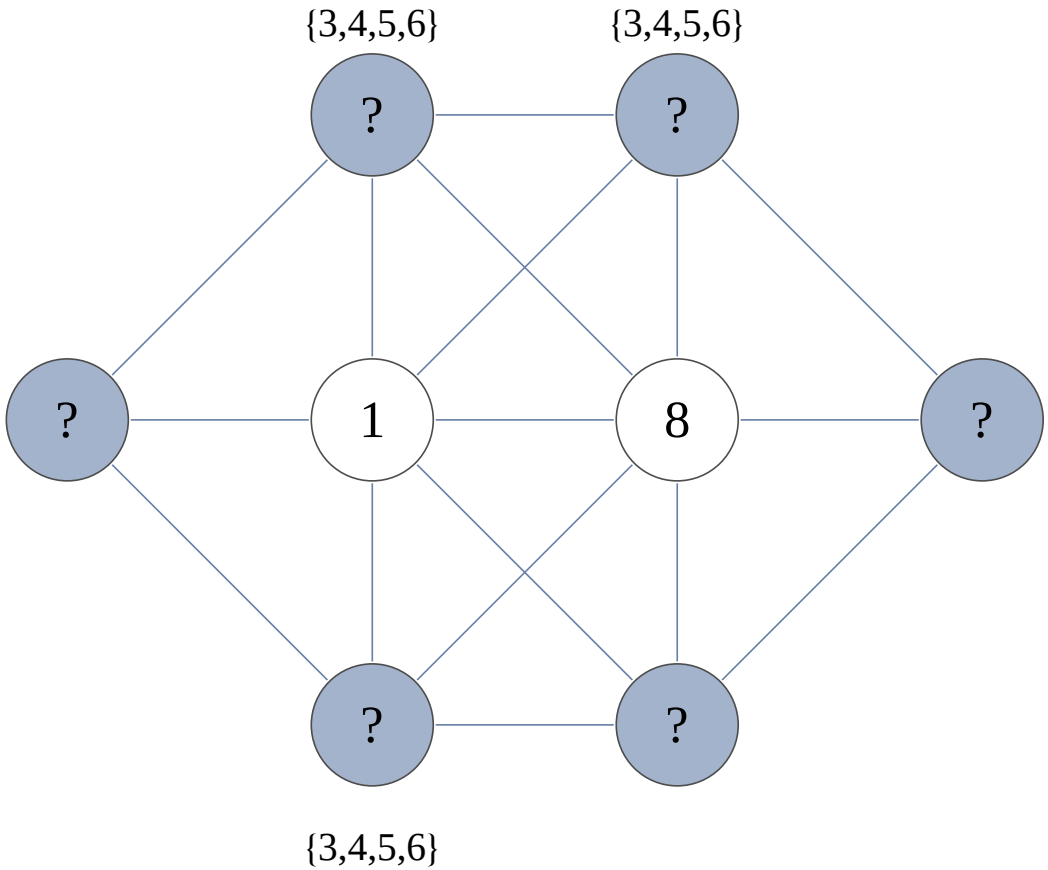
Enforce adjacency restrictions.



Solving the Crystal Maze Puzzle

Constraint propagation

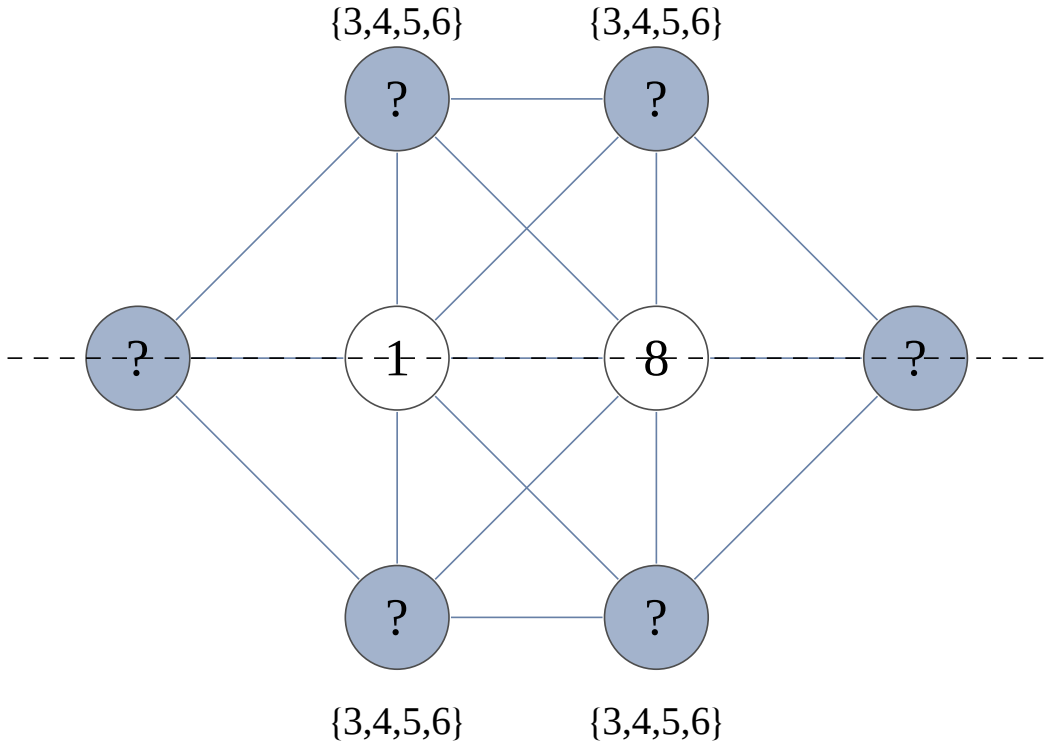
Enforce adjacency restrictions.



Solving the Crystal Maze Puzzle

Constraint propagation

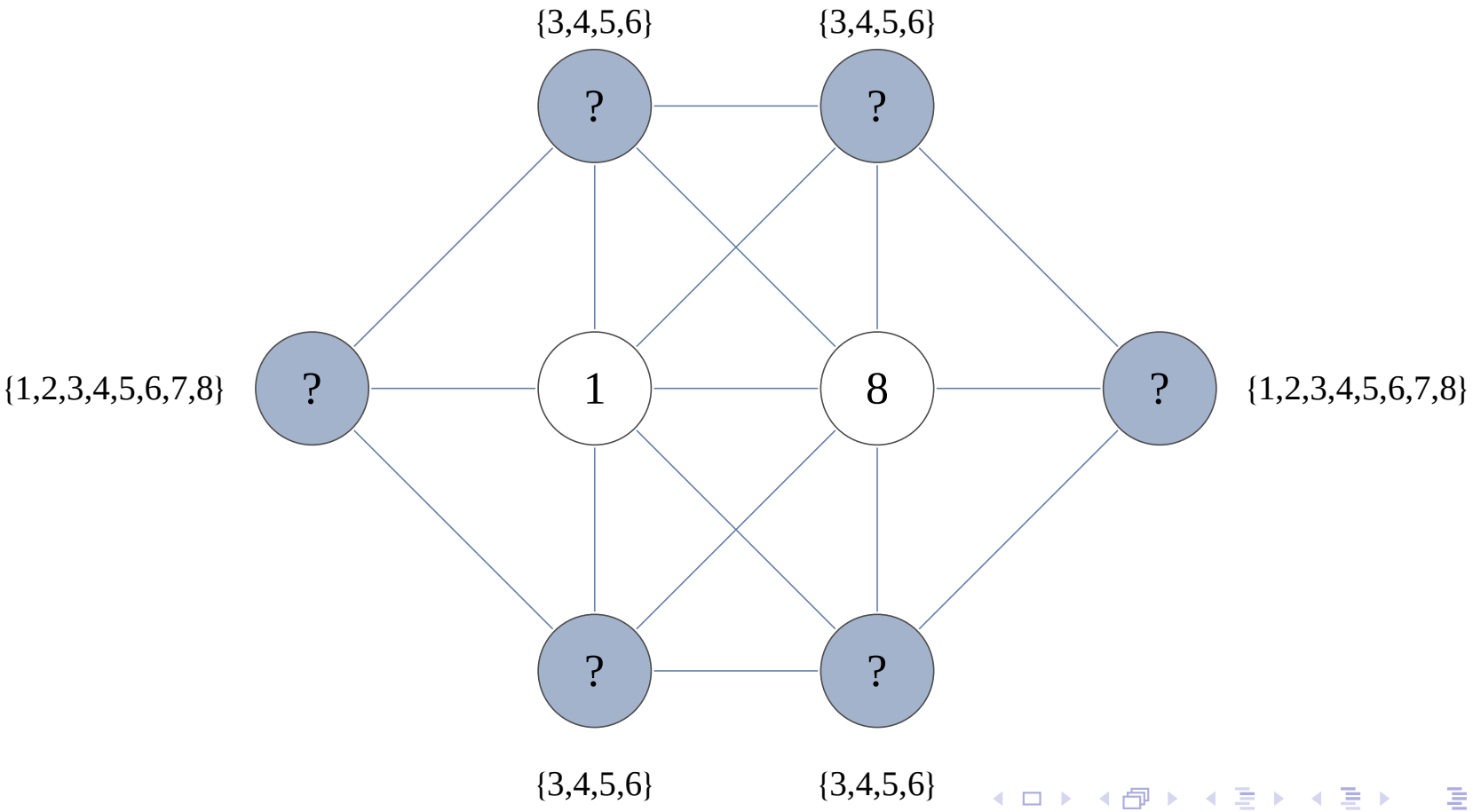
Symmetry. . .



Solving the Crystal Maze Puzzle

Constraint propagation

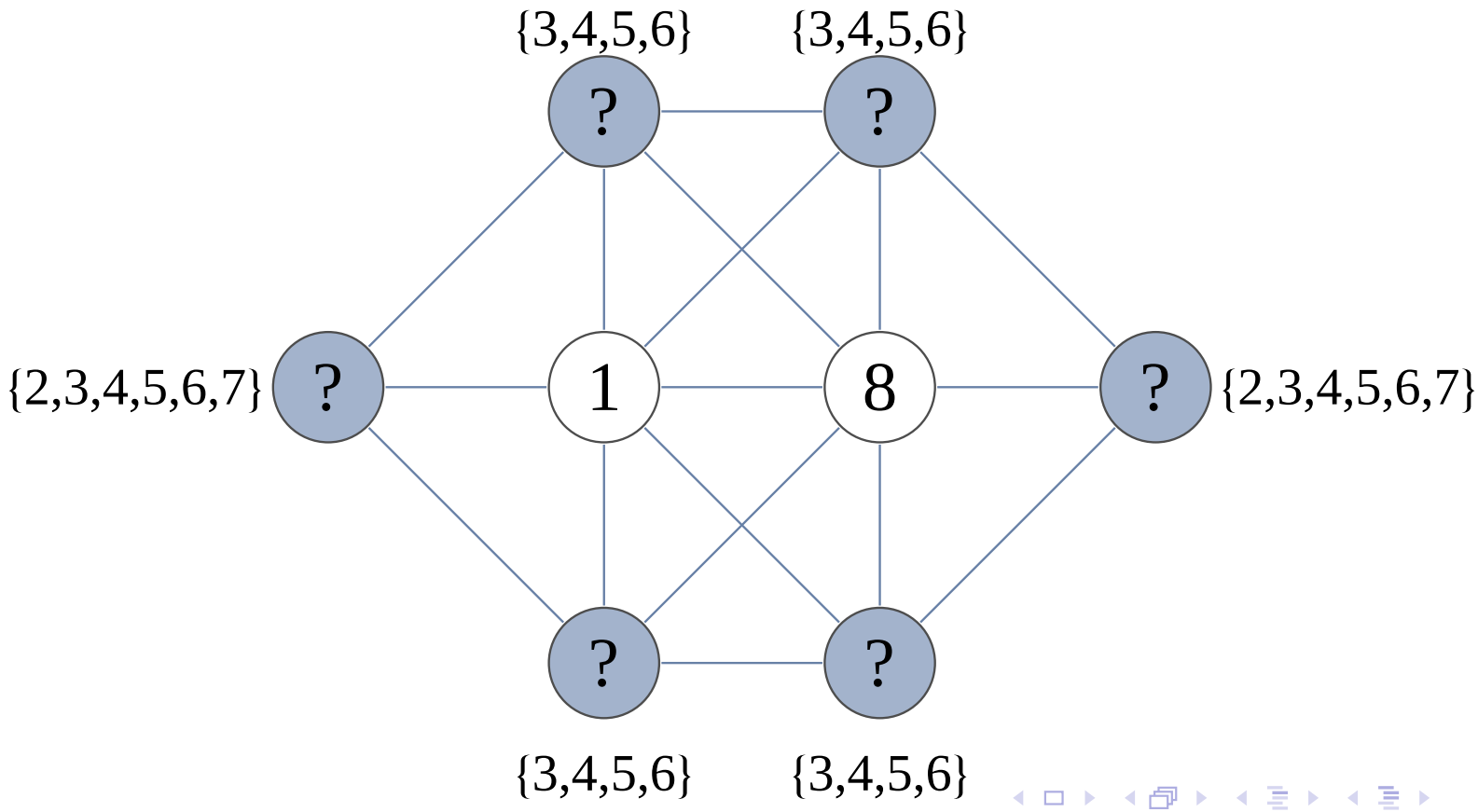
Values 1 and 8 appear exactly once.



Solving the Crystal Maze Puzzle

Constraint propagation

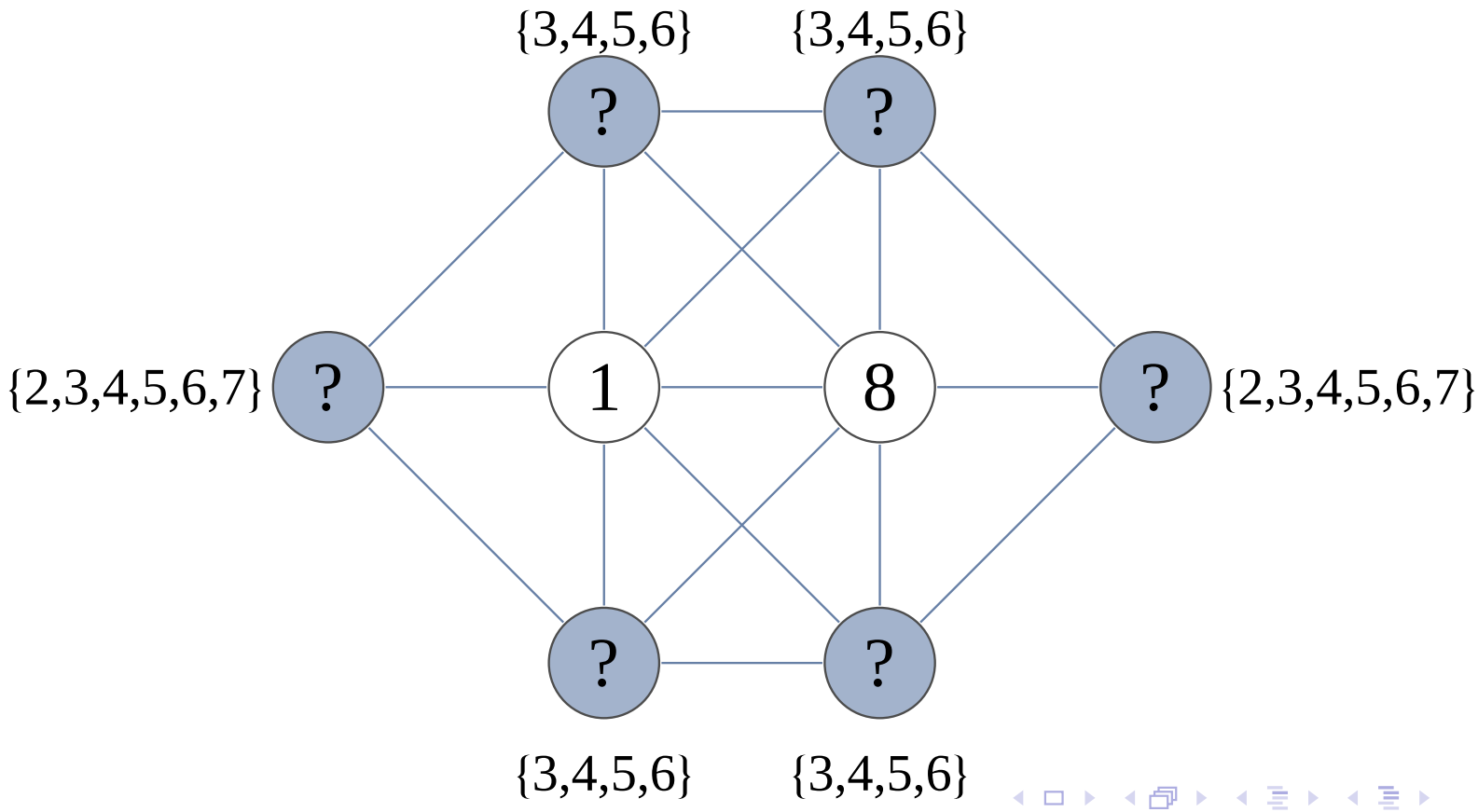
Values 1 and 8 appear exactly once.



Solving the Crystal Maze Puzzle

Constraint propagation

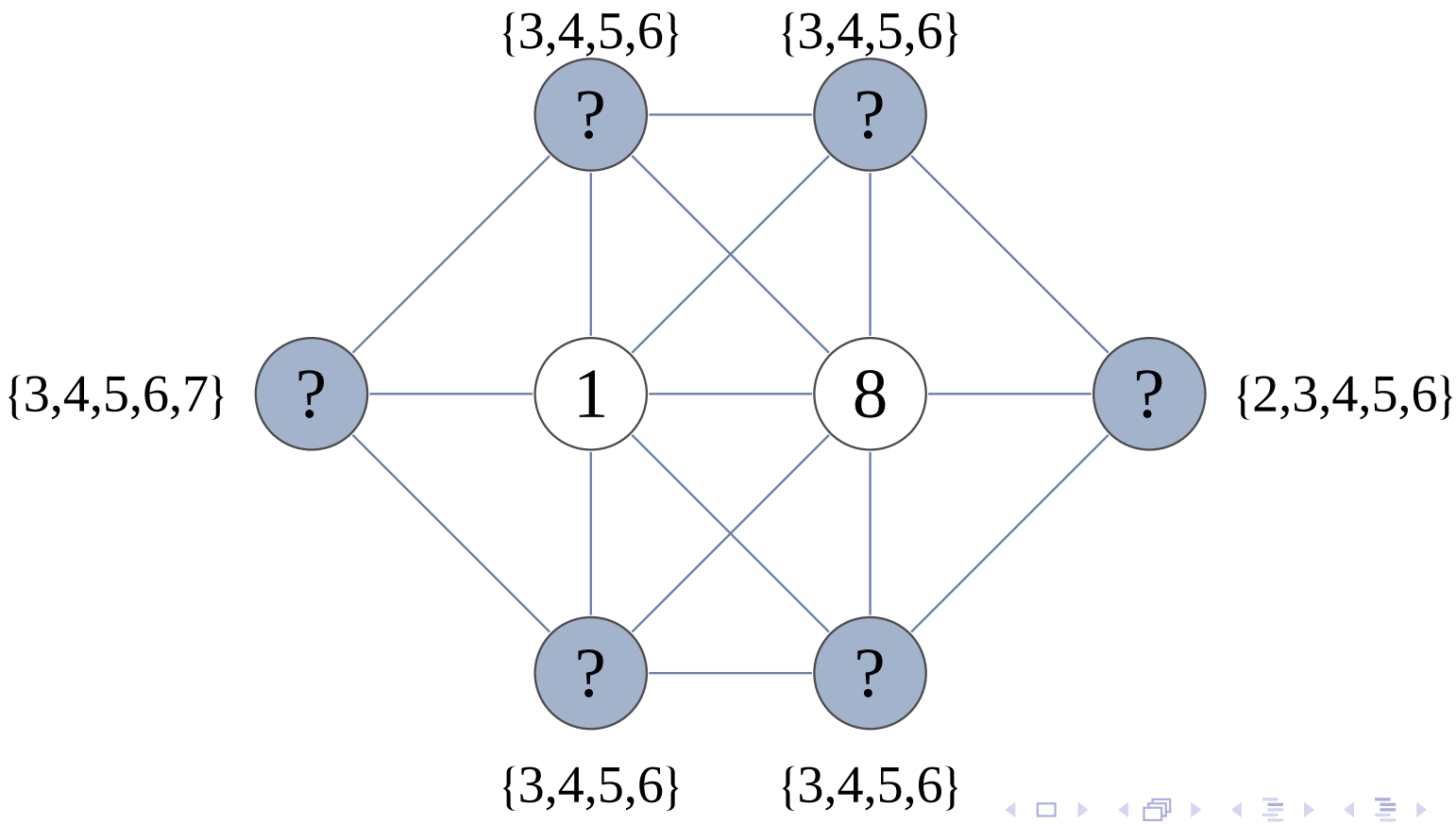
Enforce adjacency restrictions.



Solving the Crystal Maze Puzzle

Constraint propagation

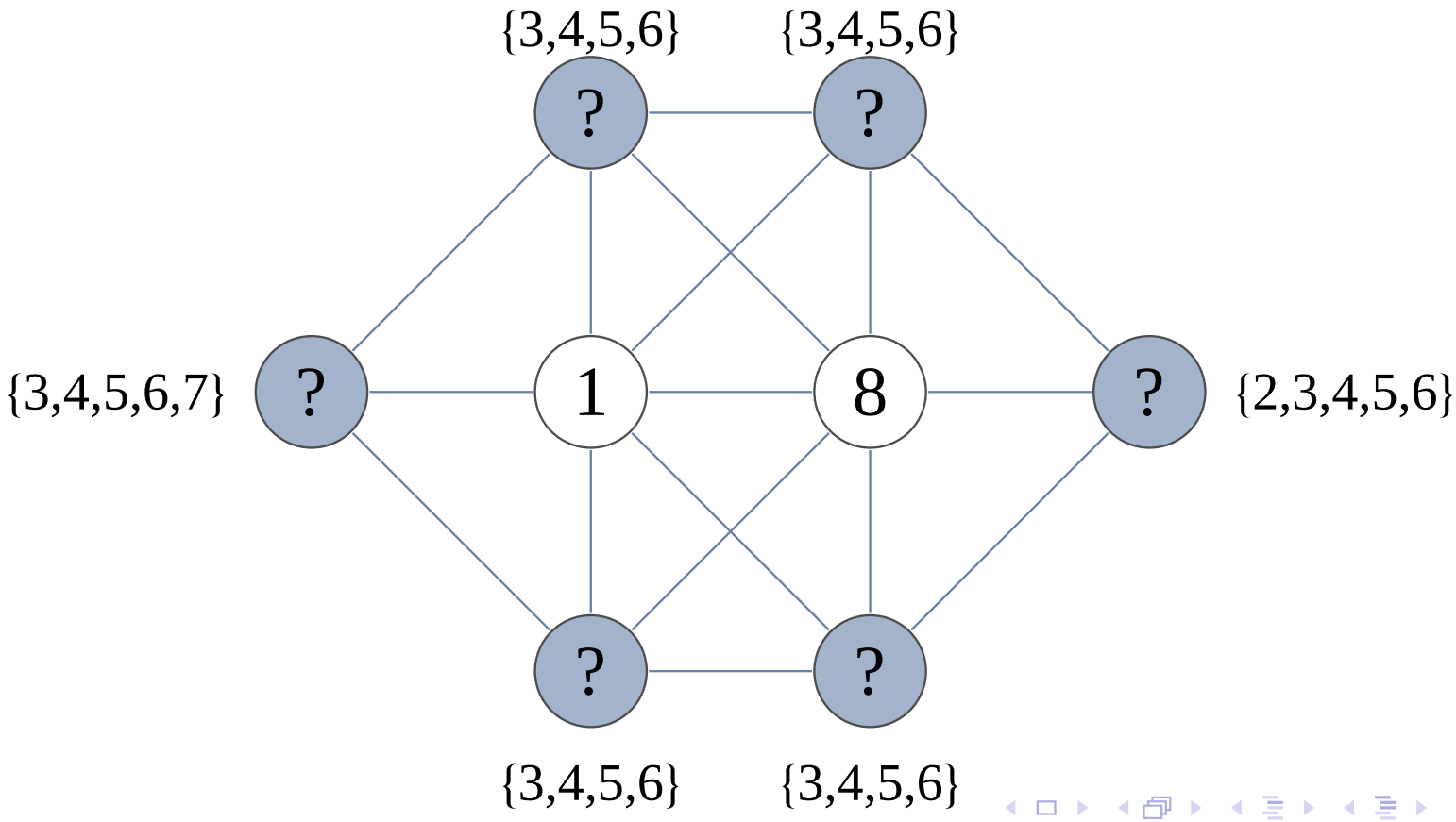
Enforce adjacency restrictions.



Solving the Crystal Maze Puzzle

Constraint propagation

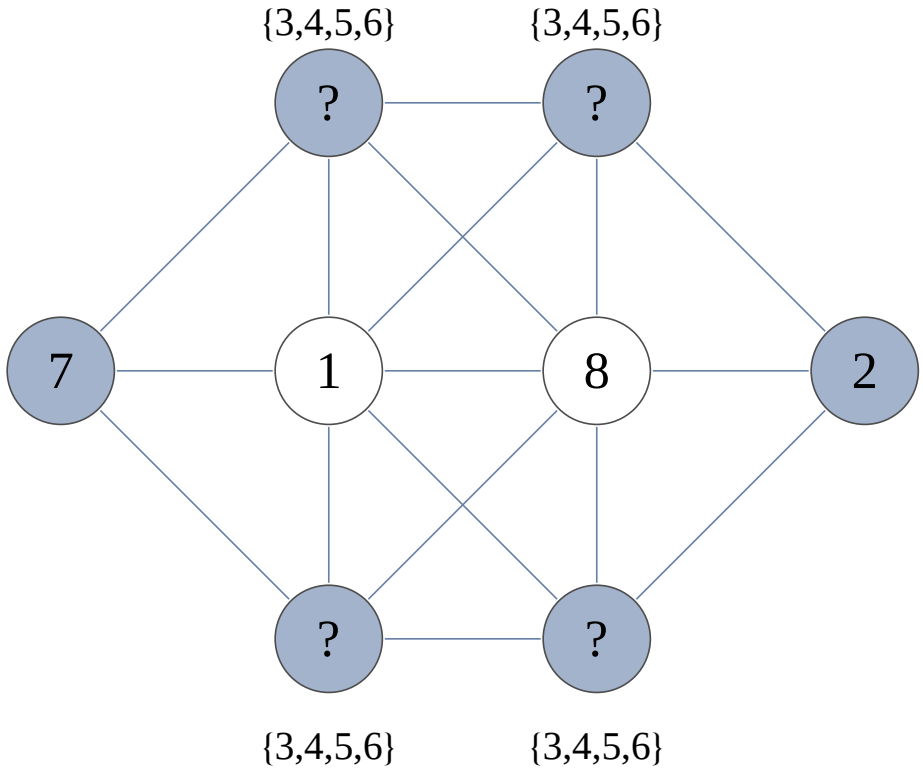
Values 1 and 8 appear exactly once.



Solving the Crystal Maze Puzzle

Constraint propagation

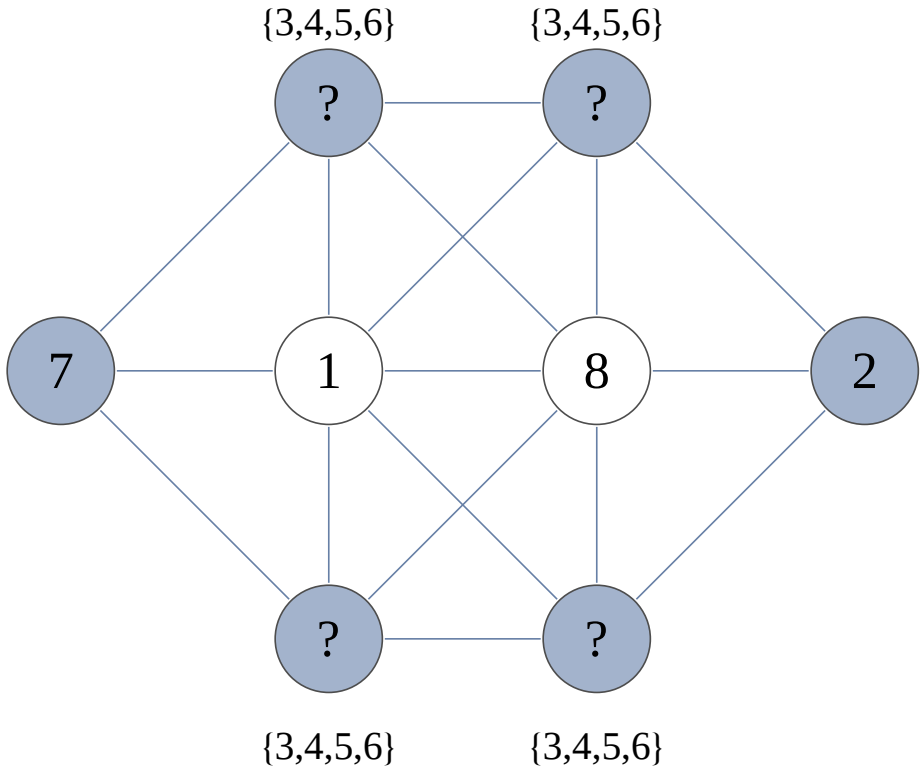
Values 1 and 8 appear exactly once.



Solving the Crystal Maze Puzzle

Constraint propagation

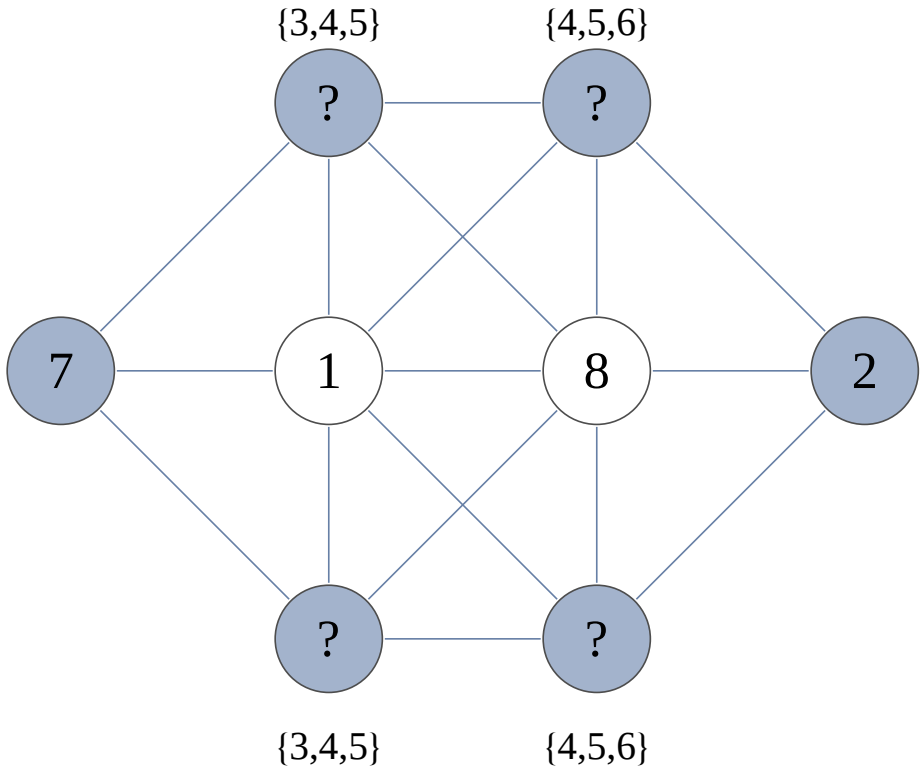
Enforce adjacency restrictions.



Solving the Crystal Maze Puzzle

Constraint propagation

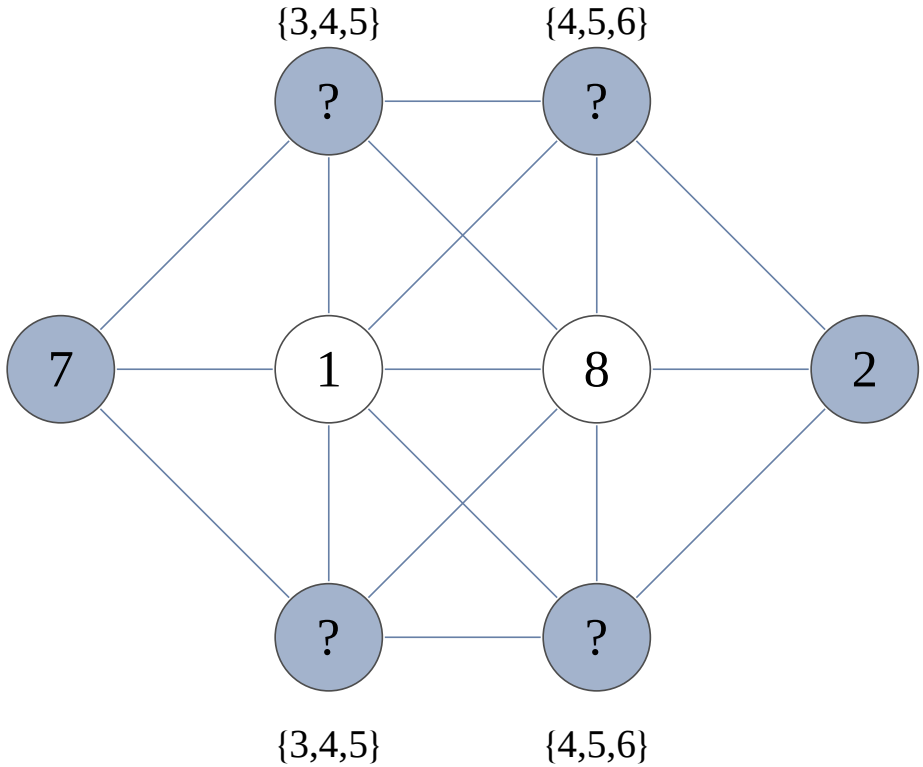
Enforce adjacency restrictions.



Solving the Crystal Maze Puzzle

Search

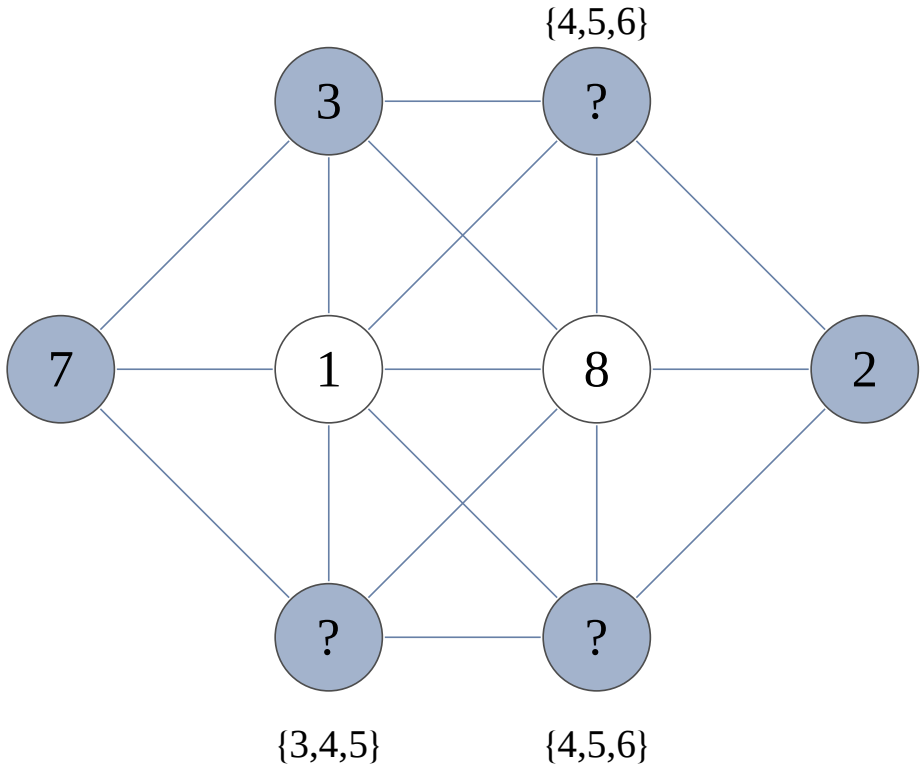
Guess, but be prepared to backtrack. . .



Solving the Crystal Maze Puzzle

Search

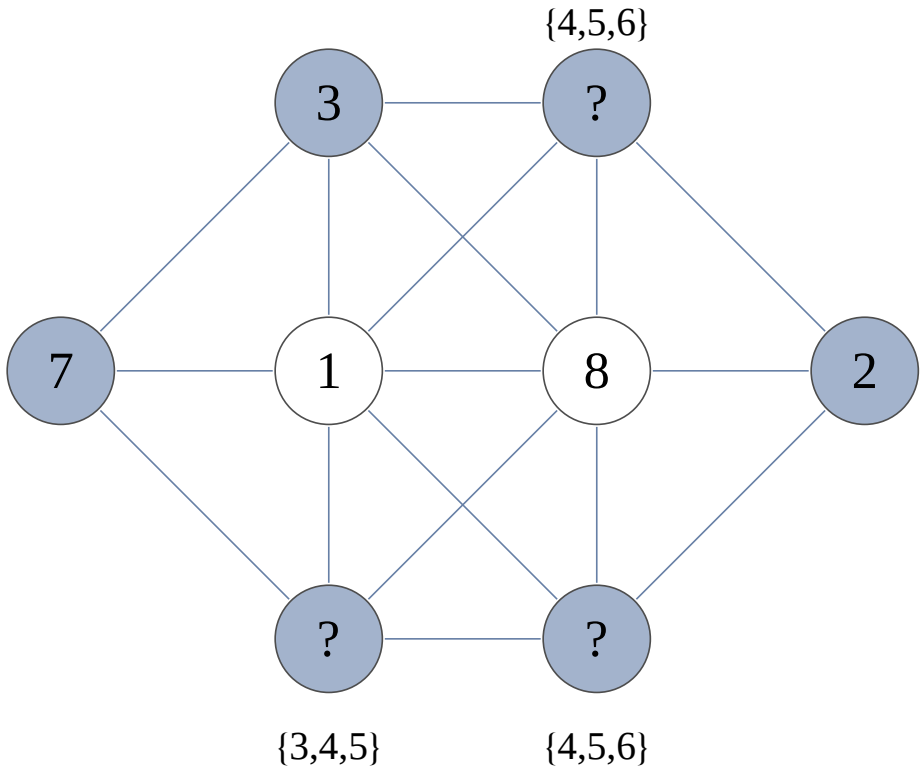
Guess, but be prepared to backtrack. . .



Solving the Crystal Maze Puzzle

Constraint propagation

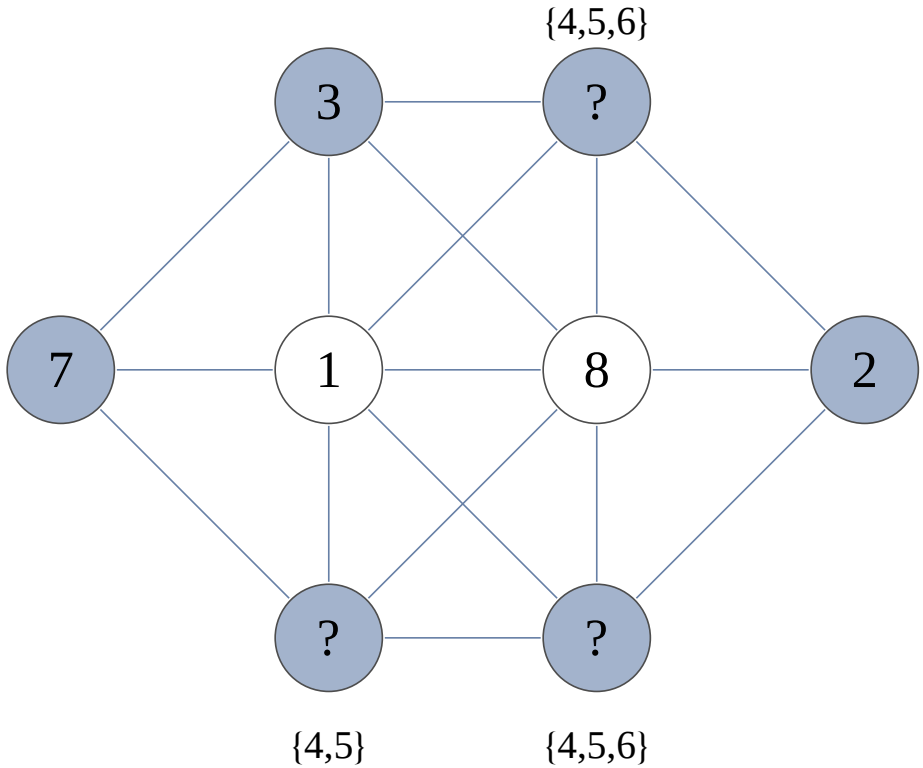
Values 1 and 8 appear exactly once.



Solving the Crystal Maze Puzzle

Constraint propagation

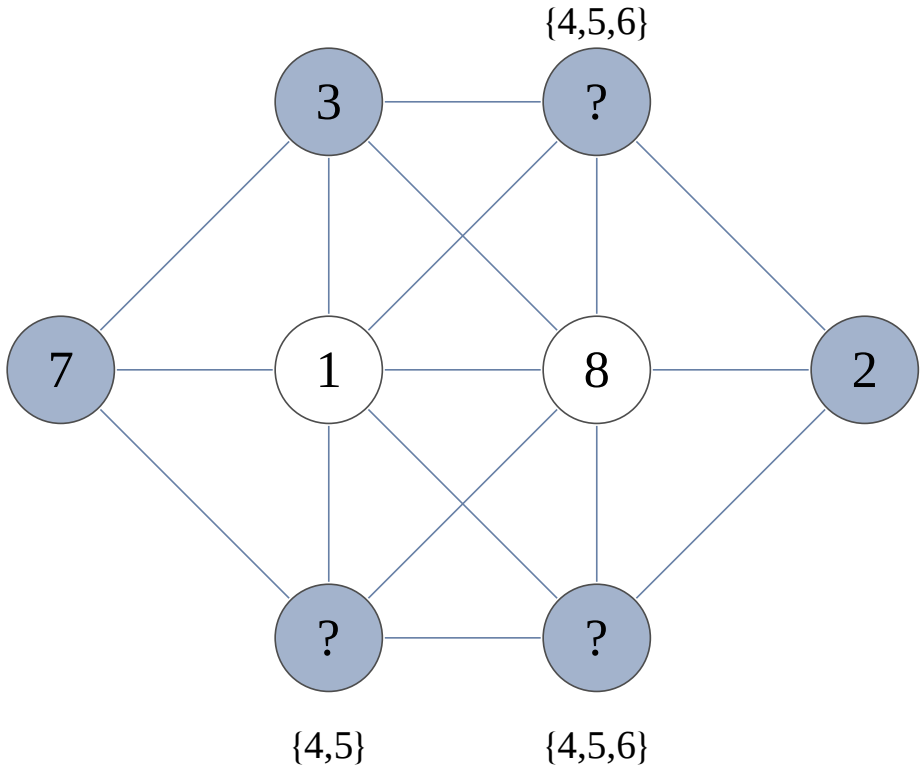
Values 1 and 8 appear exactly once.



Solving the Crystal Maze Puzzle

Constraint propagation

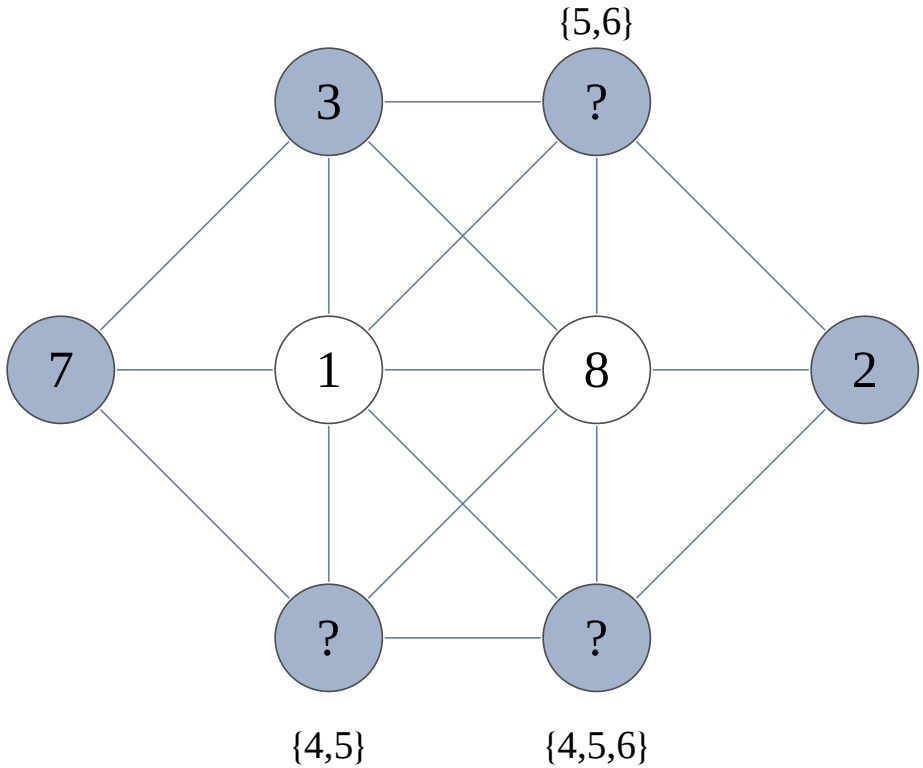
Enforce adjacency restrictions.



Solving the Crystal Maze Puzzle

Constraint propagation

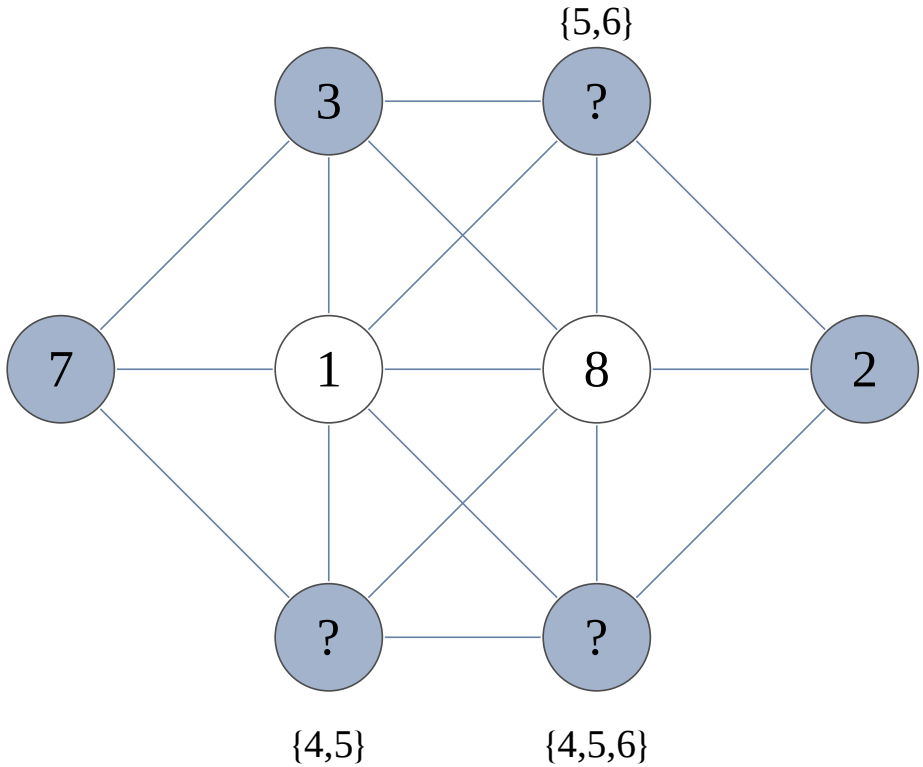
Enforce adjacency restrictions.



Solving the Crystal Maze Puzzle

Search

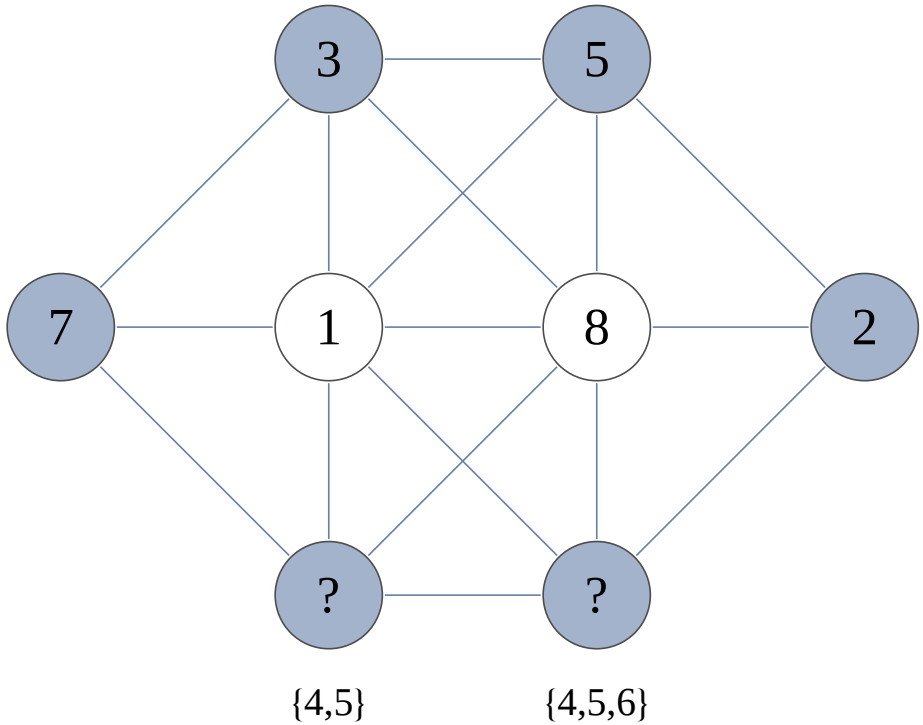
Guess, but be prepared to backtrack. . .



Solving the Crystal Maze Puzzle

Search

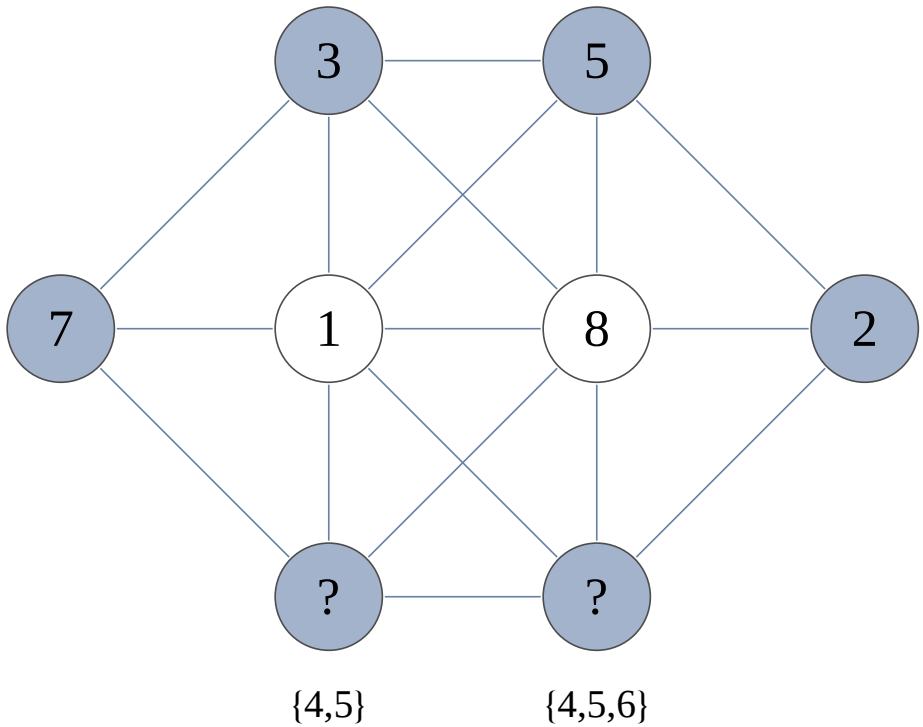
Guess, but be prepared to backtrack. . .



Solving the Crystal Maze Puzzle

Constraint propagation

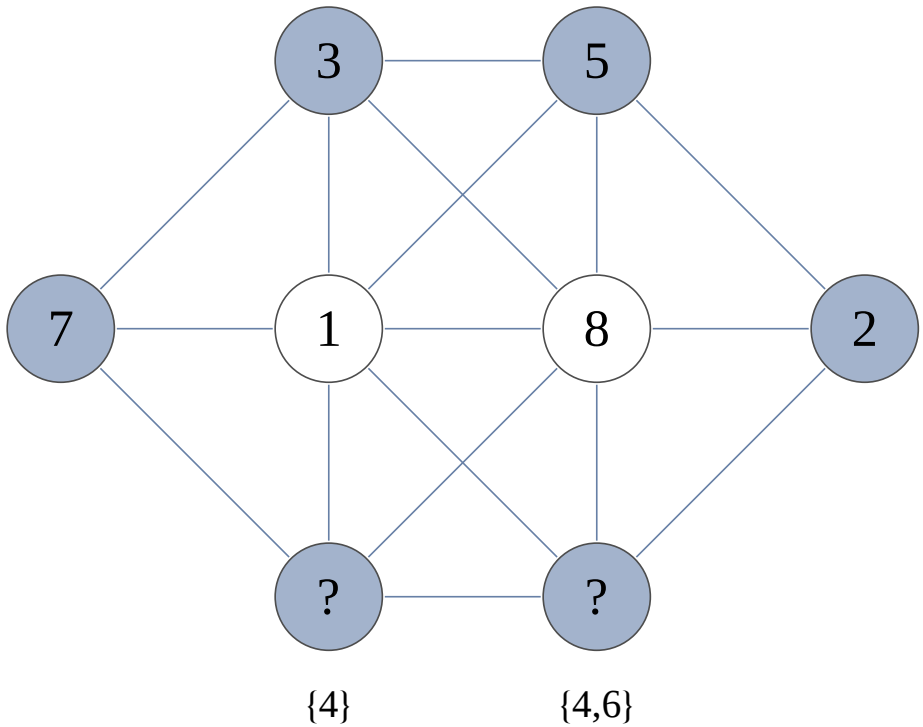
Values 1 and 8 appear exactly once.



Solving the Crystal Maze Puzzle

Constraint propagation

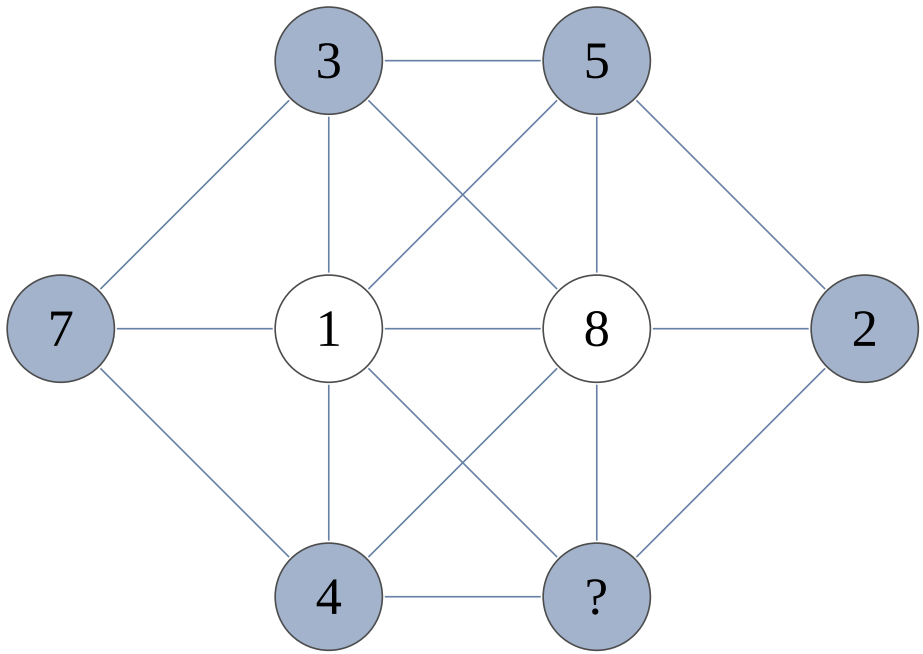
Values 1 and 8 appear exactly once.



Solving the Crystal Maze Puzzle

Constraint propagation

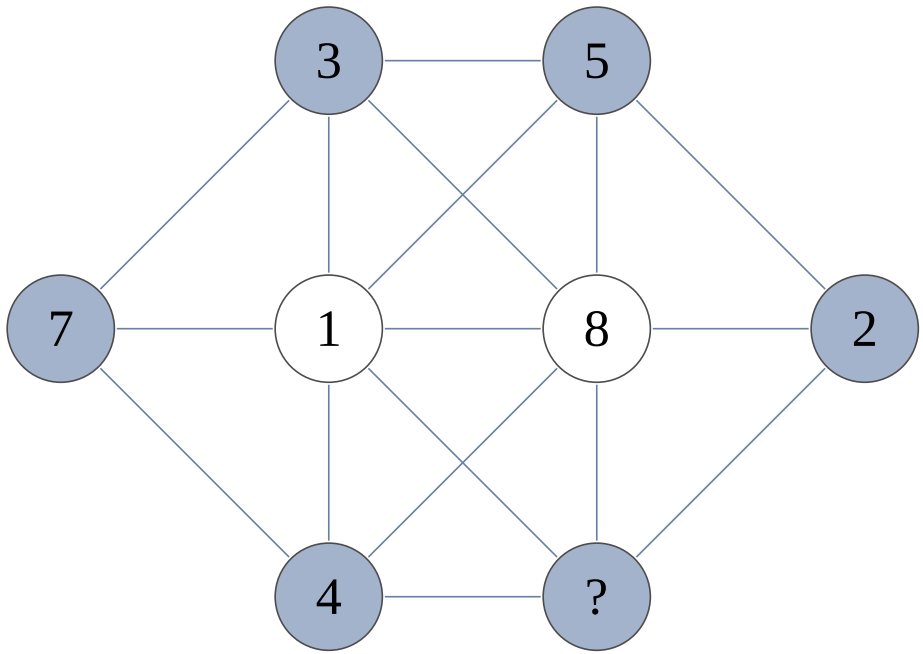
Values 1 and 8 appear exactly once.



Solving the Crystal Maze Puzzle

Constraint propagation

Values 1 and 8 appear exactly once.

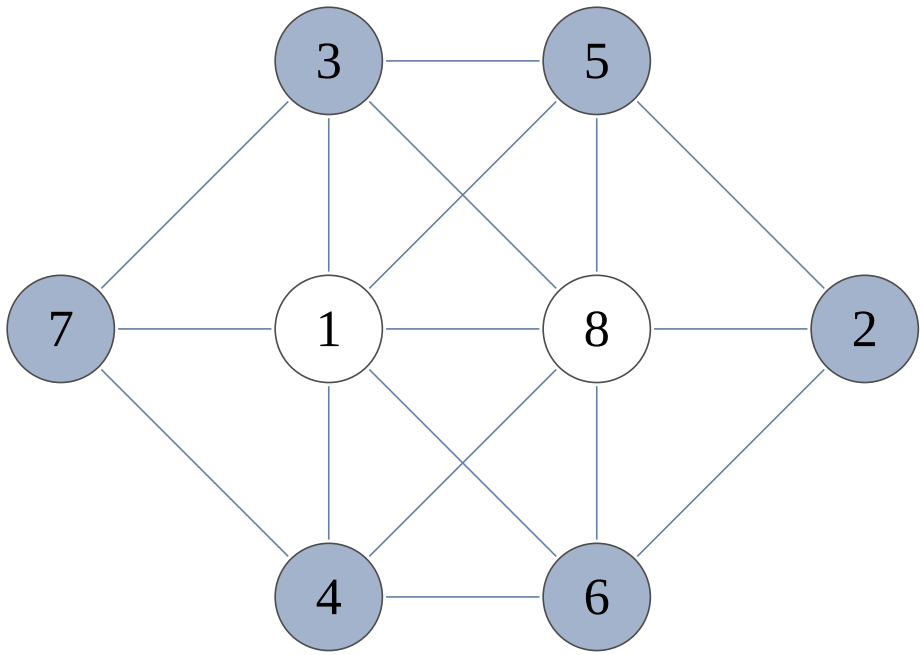


{6}

Solving the Crystal Maze Puzzle

Constraint propagation

Solution.



Solving the Crystal Maze Puzzle

Generally, it doesn't go as smooth as this.

Search will often involve backtracking.

Search

Search is often carried out by using "standard" depth-first search (DFS) procedures.

At each node of the search tree two heuristics operate:

- **variable selection**: what variable do we branch on (e.g. smallest domain, most constrained, ...)?
- **value selection**: what value in the domain do we try first (e.g. increasing, decreasing, ...)?

In general, there is no easy way to decide what strategies are good for a given problem!

Propagation

Propagation operates by exploiting dedicated filtering algorithms that enforce some degree of "consistency" by removing provably infeasible value from the domains of the constrained variables.

Possible "levels" of consistency (from weaker to stronger) are:

- arc consistency
- bound consistency
- range consistency
- ...
- hyper-arc consistency

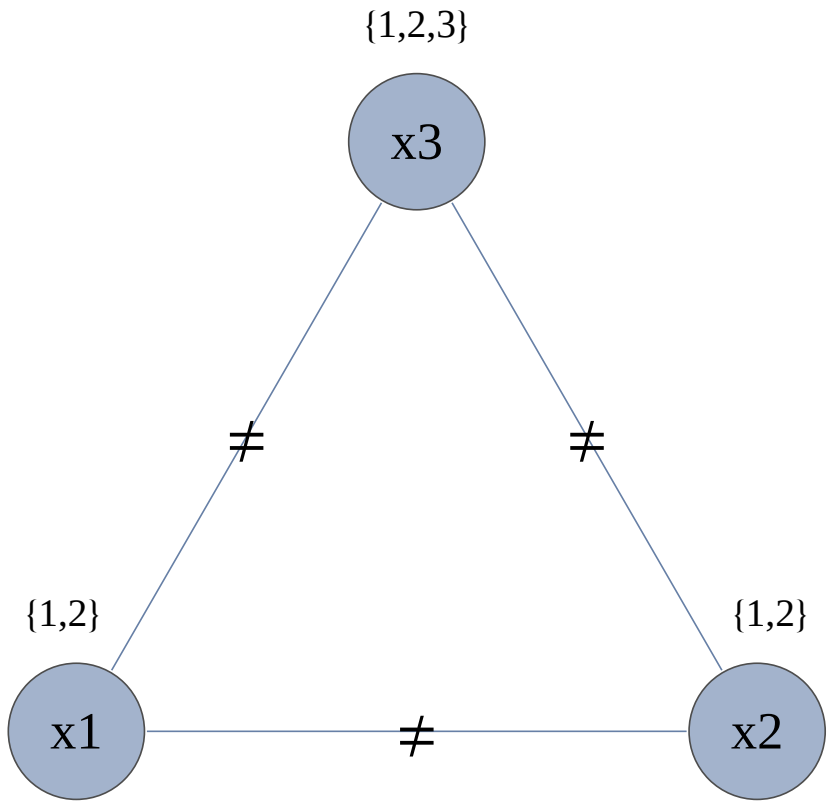
Propagation

Arc consistency

Arc consistency is the simplest (and weakest) possible form of consistency that can be enforced; it is enforced by checking each pair of variables against a given constraint that constrains them — as we did in the Crystal Maze example!

Propagation

Arc consistency: example



Propagation

Bound consistency

Bound consistency operates by pruning edges of domains, a typical example involves linear inequalities.

Propagation

Bound consistency: example

Variables & Domains:

$$x_1 \in \{1, \ldots, 3\}$$
$$x_2 \in \{5, \ldots, 9\}$$
$$x_3 \in \{1, \ldots, 2\}$$

Constraints:

$$x_1 + x_2 + x_3 \geq 15$$

Reasoning:

$$x_1 \geq 15 - \bar{x}_2 - \bar{x}_3$$

where $\bar{x}_i$ denotes the maximum value in the domain of x_i.

Propagation

Bound consistency: example

Variables & Domains:

$$x_1 \in \{1, \ldots, 3\}$$
$$x_2 \in \{5, \ldots, 9\}$$
$$x_3 \in \{1, \ldots, 2\}$$

Constraints:

$$x_1 + x_2 + x_3 \geq 15$$

Reasoning:

$$x_1 \geq 15 - 9 - 2 = 4 \notin \{1, \ldots, 3\} \quad \text{inconsistent!}$$

where $\bar{x}_i$ denotes the maximum value in the domain of x_i.

Propagation

Hyper-arc consistency

This is the strongest possible level of consistency.

Informally speaking, it guarantees that it is always possible to extend the current partial assignment to a solution via backtrack-free search.

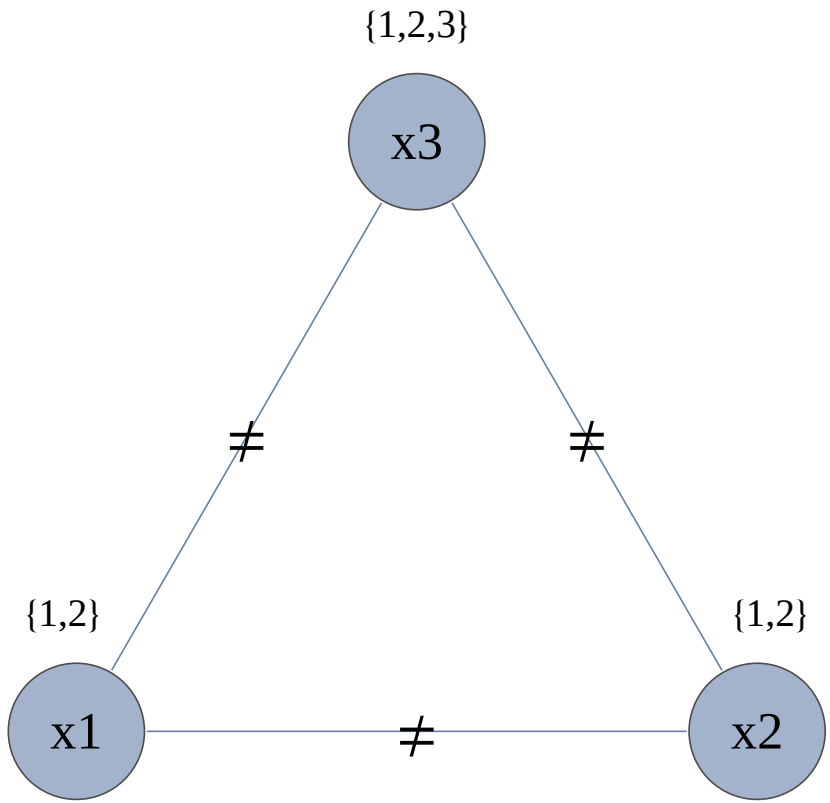
If the problem of interest is NP-hard, enforcing hyper-arc consistency on it is also NP-hard.

We shall demonstrate how it is possible to enforce hyper-arc consistency for the alldifferent$(x_1, \ldots, x_n)$ constraint, which ensures that $x_1, \ldots, x_n$ take all different values.

Jean-Charles Régin: A Filtering Algorithm for Constraints of Difference in CSPs. AAAI 1994: 362-367

Propagation

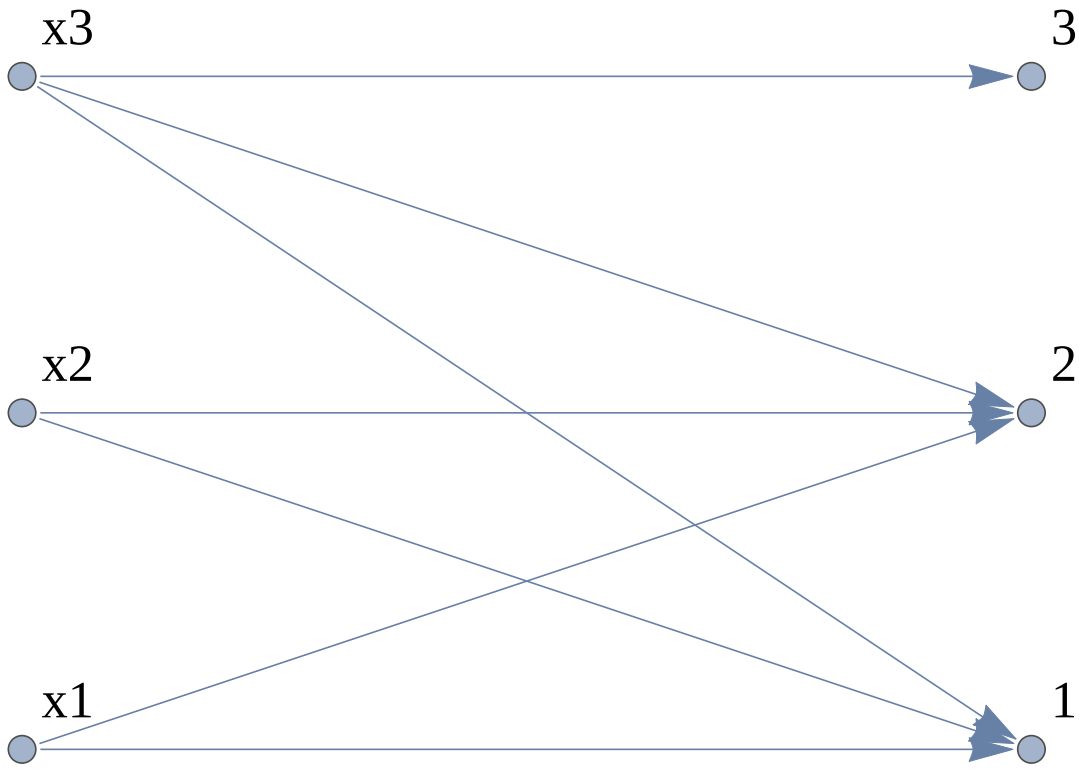
Hyper-arc consistency: example



Propagation

Hyper-arc consistency: example

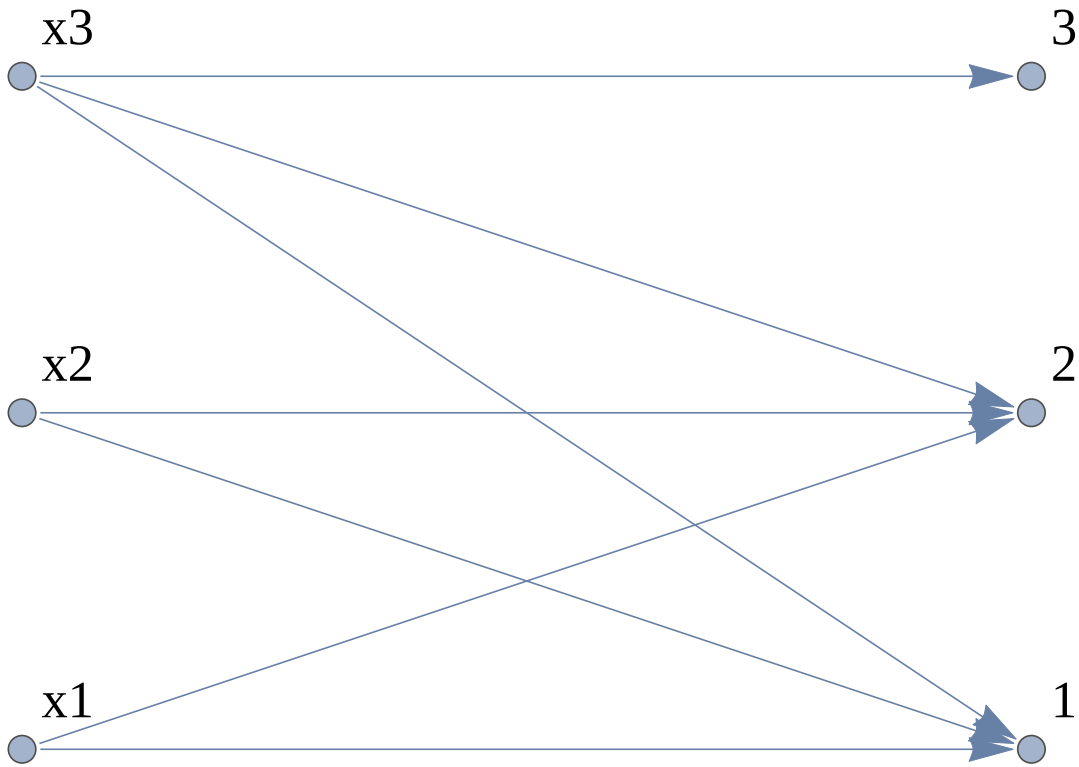
Reformulate the problem as that of finding a Maximum Matching in a Bipartite Graph — known to be solvable in polynomial time.



Propagation

Hyper-arc consistency: example

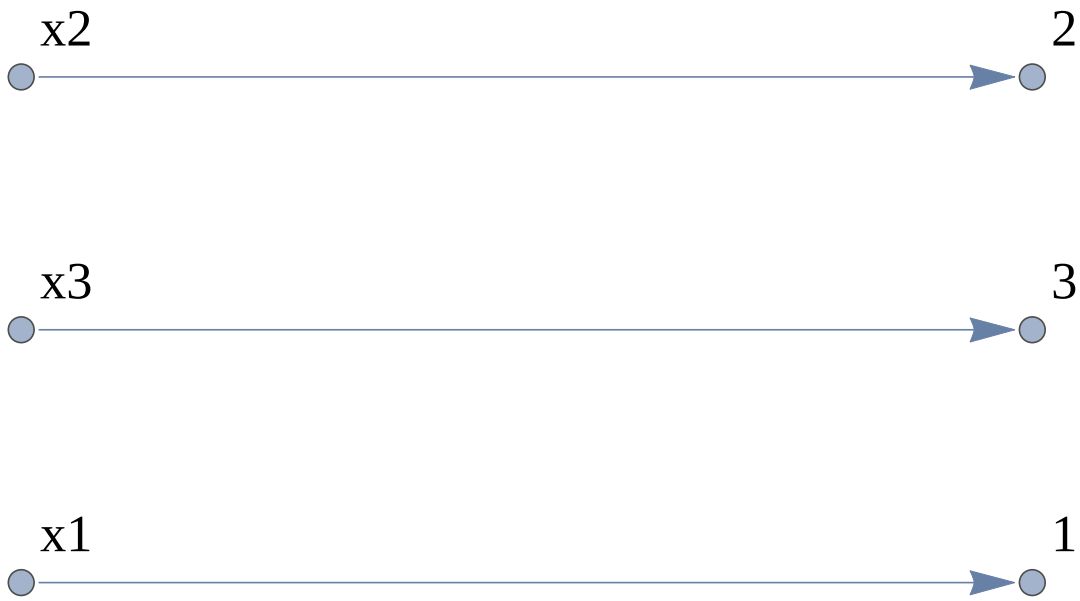
Find a maximum matching. If the cardinality of the maximum matching is less than the number of variables, then there is no feasible assignment; hence we remove all values from domains.



Propagation

Hyper-arc consistency: example

In this case there is a maximum matching of size 3, therefore there is at least one feasible assignment (displayed).



Propagation

Hyper-arc consistency: example

However, to enforce hyper-arc consistency, in principle we should be able to identify **all** possible maximum matchings of size 3 — so that we can retain every values in decision variable domains, which belongs at least to a maximum matching.

Propagation

Hyper-arc consistency: example

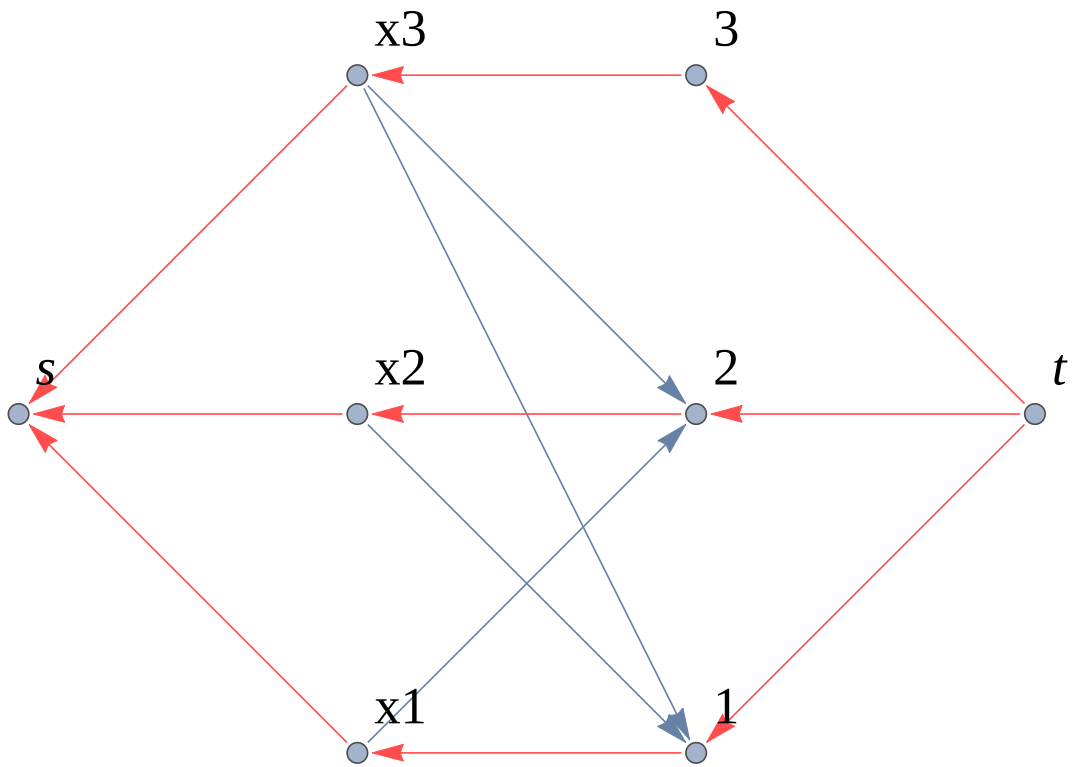
However, to enforce hyper-arc consistency, in principle we should be able to identify **all** possible maximum matchings of size 3 — so that we can retain every values in decision variable domains, which belongs at least to a maximum matching.

Fortunately, to find out if a values in a decision variable domain belongs at least to a maximum matching there is no need to enumerate all maximum matchings.

Propagation

Hyper-arc consistency: example

To understand why, we can look at the extended graph used by the Ford-Fulkerson algorithm — which can be used to determine a maximum matching — and focus on its strongly connected components (SCC).

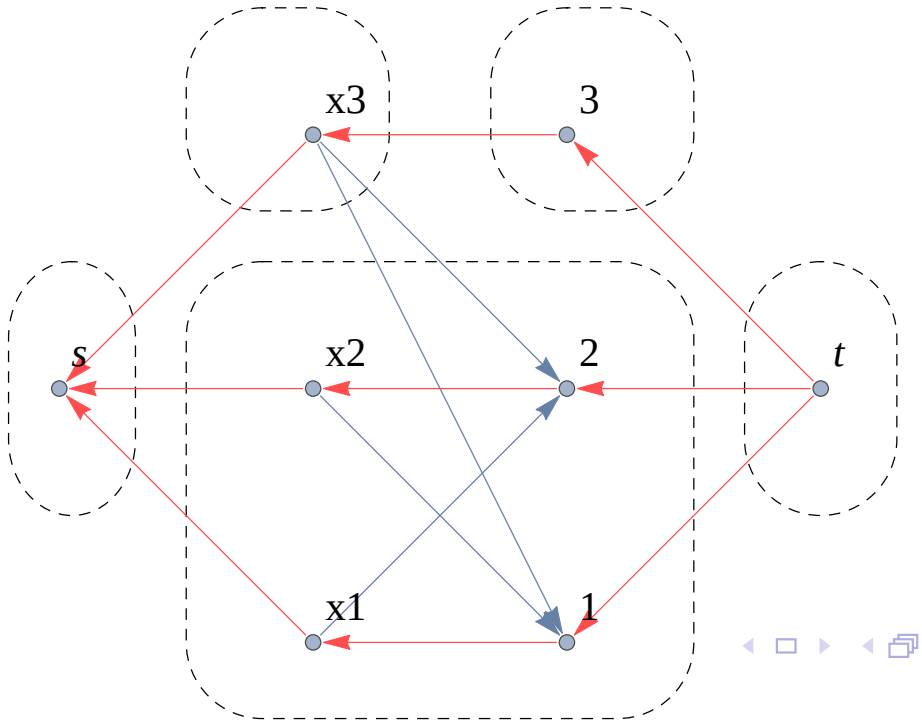


Propagation

Hyper-arc consistency: example

A directed graph is said to be strongly connected, if every vertex is reachable from every other vertex.

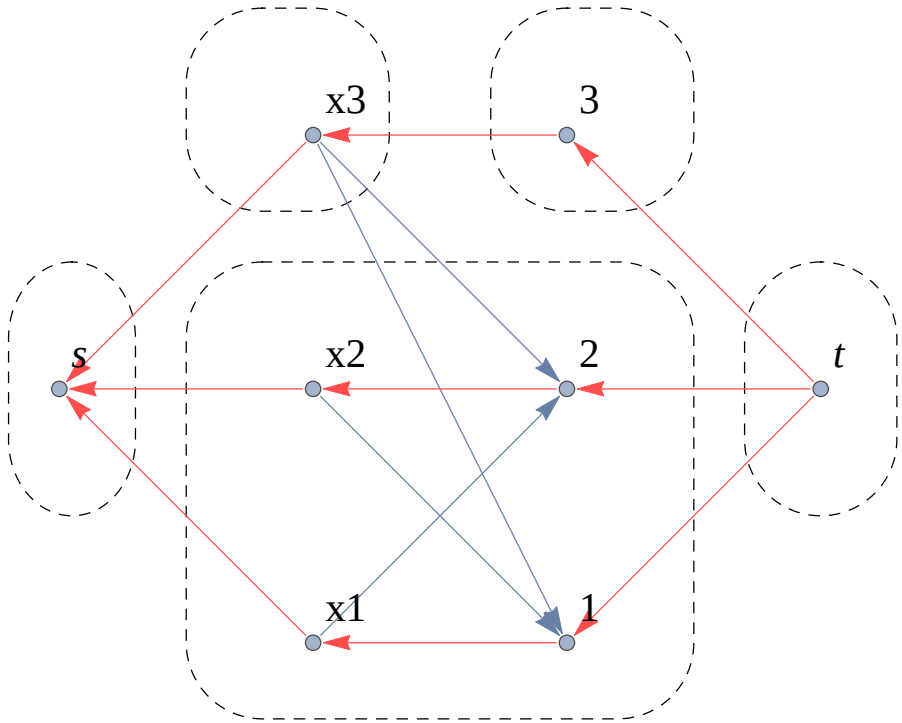
The strongly connected components of an arbitrary directed graph form a partition into subgraphs that are strongly connected; these can be computed in linear time.



Propagation

Hyper-arc consistency: example

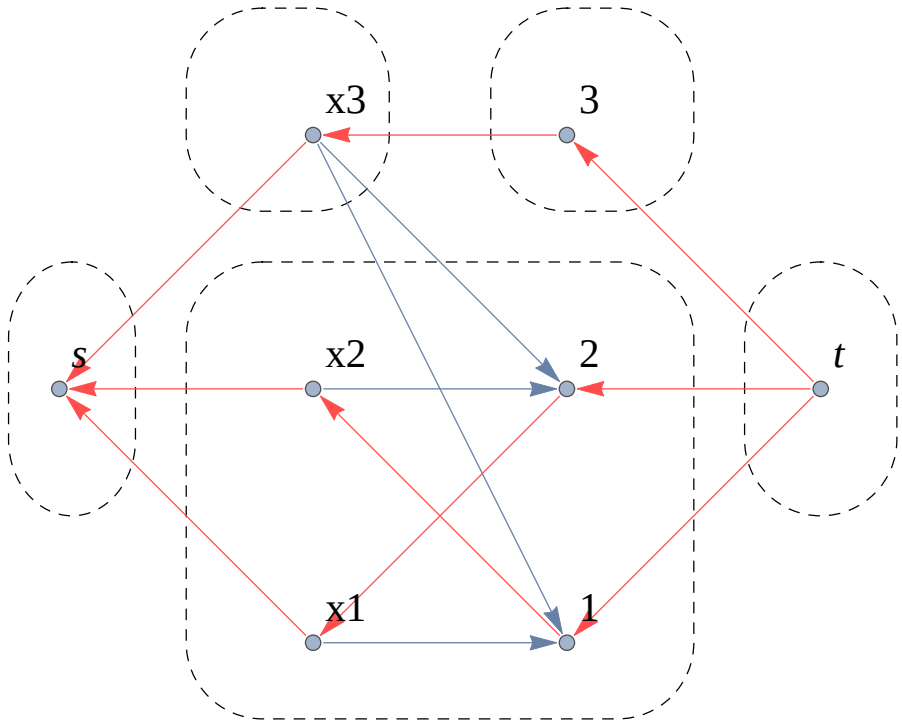
Observation: by flipping every edge in a strongly connected component we obtain a new maximum matching.



Propagation

Hyper-arc consistency: example

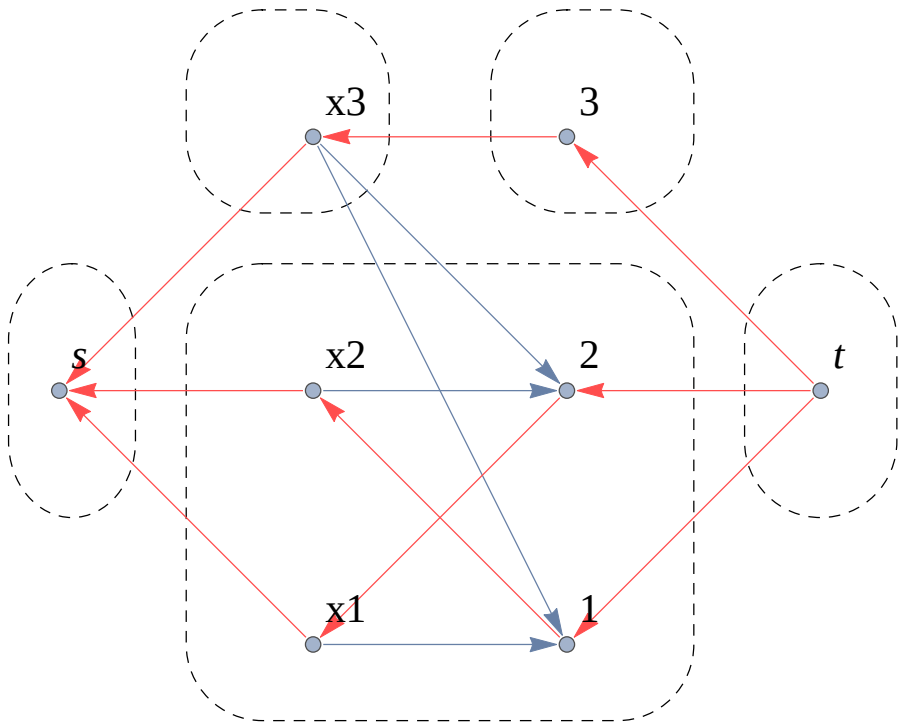
Observation: by flipping every edge in a strongly connected component we obtain a new maximum matching.



Propagation

Hyper-arc consistency: example

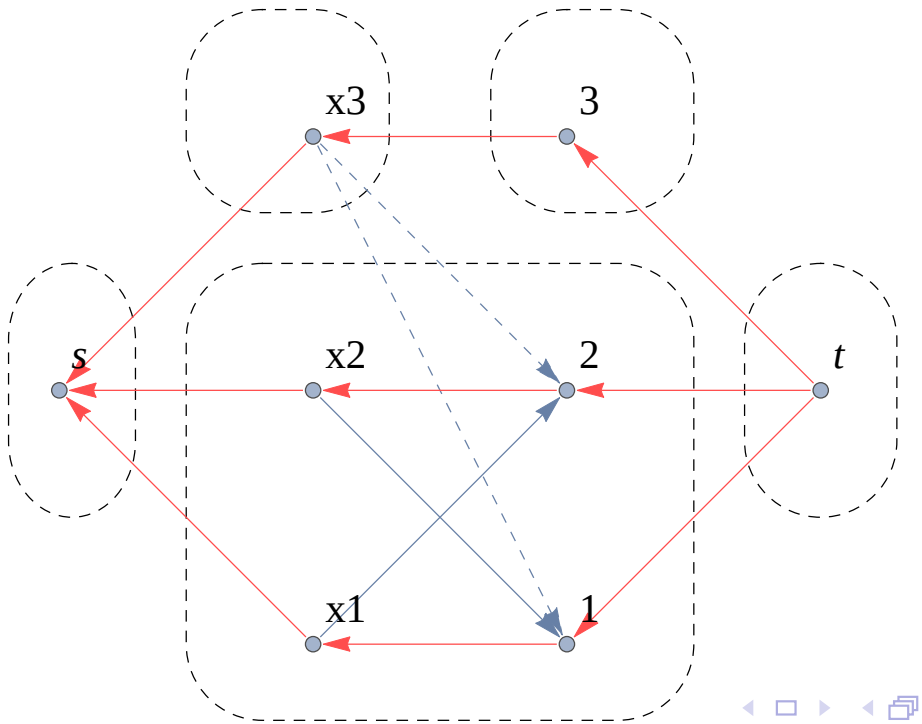
This means that all edges that connect nodes within a strongly connected component also belong to a maximum matching; hence the respective values in decision variable domains should be retained.



Propagation

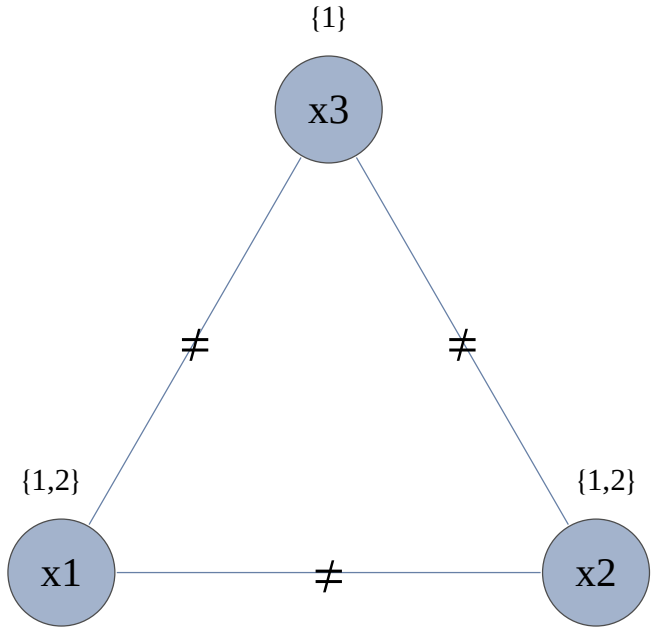
Hyper-arc consistency: example

Conversely, all edges that connect nodes belonging to different strongly connected component and that do not belong to the current matching, can not belong to any maximum matching (note that strongly connected components are all saturated); hence the respective values in decision variable domains should be removed.



Propagation

Hyper-arc consistency: example



Decision making under uncertainty

Can we extend constraint programming so that it can be applied to problems of decision making under uncertainty?

Stochastic Constraint Programming

arXiv:0903.1152v1 [cs.AI] 6 Mar 2009

Stochastic Constraint Programming

Toby Walsh[1]

Abstract. To model combinatorial decision problems involving uncertainty and probability, we introduce stochastic constraint programming. Stochastic constraint programs contain both decision variables (which we can set) and stochastic variables (which follow a probability distribution). They combine together the best features of traditional constraint satisfaction, stochastic integer programming, and stochastic satisfiability. We give a semantics for stochastic constraint programs, and propose a number of complete algorithms and approximation procedures. Finally, we discuss a number of extensions of stochastic constraint programming to relax various assumptions like the independence between stochastic variables, and compare with other approaches for decision making under uncertainty.

1 Introduction

Many decision problems contain uncertainty. Data about events in the past may not be known exactly due to errors in measuring or difficulties in sampling, whilst data about events in the future may simply not be known with certainty. For example, when scheduling power stations, we need to cope with uncertainty in future energy demands. As a second example, nurse rostering in an accident and emergency department requires us to anticipate variability in workload. As a final example, when constructing a balanced bond portfolio, we must deal with uncertainty in the future price of bonds. To deal with such situations, we propose an extension of constraint programming called *stochastic constraint programming* in which we distinguish between decision variables, which we are free to set, and stochastic (or observed) variables, which follow some probability distribution.

2 Stochastic constraint programs

We define a number of models of stochastic constraint programming of increasing complexity. In an one stage stochastic constraint satisfaction problem (stochastic CSP), the decision variables are set before the stochastic variables. This models situations where we act now and observe later. For example, we have to decide now which nurses to have on duty and will only later discover the actual workload. We can easily invert the instantiation order if the application demands, with the stochastic variables set before the decision variables. Constraints are defined (as in traditional constraint satisfaction) by relations of allowed tuples of values. Constraints can, however, be implemented with specialized and efficient algorithms for consistency checking. The stochastic variables independently take values with probabilities given by a probability distribution. We discuss later how to relax these assumptions, and how this compares with other frameworks. A one stage stochastic CSP is satisfiable iff there exists values for the decision variables so that, given random values for the stochastic variables, the probability that all the constraints are satisfied equals or exceeds a threshold θ. The probabilistic satisfaction of constraints allows us to ignore worlds (values for the stochastic variables) which are rare. Note that the definition reduces to that of a traditional constraint satisfaction problem if we have no stochastic variables and $\theta = 1$.

In a two stage stochastic CSP, there are two sets of decision variables, V_{d1} and V_{d2}, and two sets of stochastic variables, V_{s1} and V_{s2}. The aim is to find values for the variables in V_{d1}, so that given random values for V_{s1}, we can find values for V_{d2}, so that given random values for V_{s2}, the probability that all the constraints are satisfied equals or exceeds θ. Note that the values chosen for the second set of decision variables V_{d2} are conditioned on both the values chosen for the first set of decision variables V_{d1} and on the random values given to the first set of stochastic variables V_{s1}. This can model situations in which items are produced and can be consumed or put in stock for later consumption. Future production then depends both on previous production (earlier decision variables) and on previous demand (earlier stochastic variables). A m stage stochastic CSP is defined in an analogous way to one and two stage stochastic CSPs.

A stochastic constraint optimization problem (stochastic COP) is a stochastic CSP plus a cost function defined over the decision and stochastic variables. The aim is to find a solution that satisfies the stochastic CSP which minimizes (or, if desired, maximizes) the expected value of the cost function.

3 Production planning example

The following stochastic constraint program models a simple m quarter production planning problem. In each quarter, we will sell between 100 and 105 copies of a book. To keep customers happy, we want to satisfy demand in all m quarters with 80% probability. At the start of each quarter, we decide how many books to print for that quarter. This problem is modelled by a m stage stochastic CSP. There are m decision variables, x_i representing production in each quarter. There are also m stochastic variables, y_i representing demand in each quarter. These take values between 100 and 105 with equal probability. There is a constraint to ensure 1st quarter production meets 1st quarter demand:

$$x_1 \geq y_1$$

There is also a constraint to ensure 2nd quarter production meets 2nd quarter demand plus any unsatisfied demand or less any stock:

$$x_2 \geq y_2 + (y_1 - x_1)$$

And there is a constraint to ensure jth quarter production ($j \geq 2$) meets jth quarter demand plus any unsatisfied demand or less any

[1] Cork Constraint Computation Centre, University College Cork, Ireland. email: tw@4c.ucc.ie. The author thanks the other members of the APES research group, especially Ian Gent for his helpful discussions. This research was supported by an EPSRC advanced research fellowship.

European Conference on Artificial Intelligence, 2002

Constraints (2006) 11: 53–80
DOI 10.1007/s10601-006-6849-7

Stochastic Constraint Programming: A Scenario-Based Approach

S. Armagan Tarim · Suresh Manandhar · Toby Walsh

© Springer Science + Business Media, LLC 2006

Abstract To model combinatorial decision problems involving uncertainty and probability, we introduce scenario based stochastic constraint programming. Stochastic constraint programs contain both decision variables, which we can set, and stochastic variables, which follow a discrete probability distribution. We provide a semantics for stochastic constraint programs based on scenario trees. Using this semantics, we can compile stochastic constraint programs down into conventional (non-stochastic) constraint programs. This allows us to exploit the full power of existing constraint solvers. We have implemented this framework for decision making under uncertainty in stochastic OPL, a language which is based on the OPL constraint modelling language (Van Hentenryck, P., Michael, L., Perron, L., & Regin, J.-C. (1999) Constraint programming in OPL. In G. Nadathur, (Ed.), *Principles and practice of declarative programming*, LNCS 1702, (pp. 97–116). Springer-Verlag). To illustrate the potential of this framework, we model a wide range of problems in areas as diverse as portfolio diversification, agricultural planning and production/inventory management.

Keywords Constraint programming · Constraint satisfaction · Reasoning under uncertainty

S. A. Tarim (✉)
Cork Constraint Computation Centre,
Department of Computer Science,
University College Cork, Cork, Ireland
e-mail: at@4c.ucc.ie

S. Manandhar
Artificial Intelligence Group,
Department of Computer Science,
University of York, York, UK
e-mail: suresh@cs.york.ac.uk

T. Walsh
National ICT Australia and School of Computer Science and Engineering,
University of New South Wales, Sydney, Australia
e-mail: tw@cse.unsw.edu.au

Springer

Constraints, 2006

Stochastic Constraint Programming (Walsh, 2002)

A Stochastic Constraint Satisfaction Problem (SCSP) is a 7-tuple

$$\langle V, S, D, P, C, \theta, L \rangle$$

1. $V = \{v_1, \ldots, v_n\}$ is a set of decision variables
2. D is a function mapping each decision variable in V to a domain of potential values
3. $S = \{s_1, \ldots, s_n\}$ is a set of random variables
4. P is a function mapping each random variable in S to a probability distribution
5. C is a set of (chance) constraints, possibly involving random variables
6. θ_h is a threshold probability associated with chance-constraint h
7. $L = [\langle V_1, S_1 \rangle, \ldots, \langle V_i, S_i \rangle, \ldots, \langle V_m, S_m \rangle]$ is a list of *decision stages*.

by considering an objective function $f(V, S)$ we obtain a Stochastic Constraint Optimization Problem (SCOP).

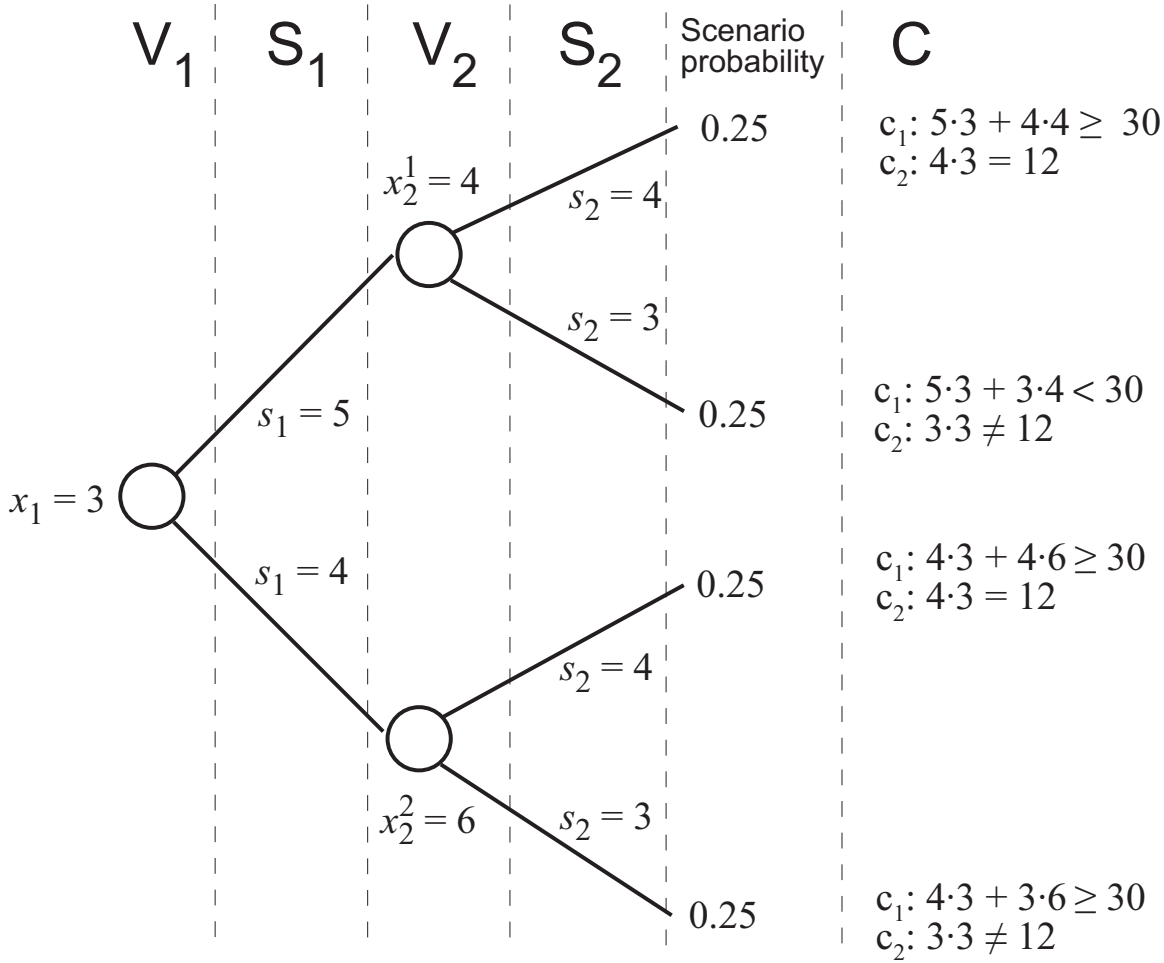
Stochastic Constraint Programming (Walsh, 2002)

An SCSP

- $V_1 = \{x_1\}$,
  $V_2 = \{x_2\}$
- $D(x_1) = \{1, \ldots, 4\}$,
  $D(x_2) = \{3, \ldots, 6\}$
- $S_1 = \{s_1\}$,
  $S_2 = \{s_2\}$
- $P(s_1) = \{(0.5)5, (0.5)4\}$,
  $P(s_2) = \{(0.5)3, (0.5)4\}$
- $C = \left\{ \begin{array}{l} c_1 : \Pr\{s_1x_1 + s_2x_2 \geq 30\} \geq 0.75 \\ c_2 : \Pr\{s_2x_1 = 12\} \geq 0.5; \end{array} \right\}$
- $L = [\langle V_1, S_1 \rangle, \langle V_2, S_2 \rangle]$

Stochastic Constraint Programming (Walsh, 2002)

A solution to the previous SCSP



Stochastic Constraint Programming (Walsh, 2002)

An SCOP: Static Stochastic Knapsack

- $V = \{x_1, \ldots, x_3\}$
- $D(x_i) = \{0, 1\} \quad \forall i \in \{1, \ldots, 3\}$
- $S = \{w_1, \ldots, w_3\}$
- $P(w_1) = \{5(0.5), 8(0.5)\}$,
  $P(w_2) = \{3(0.5), 9(0.5)\}$,
  $P(w_3) = \{15(0.5), 4(0.5)\}$
- $C = \{Pr(w_1x_1 + w_2x_2 + w_3x_3 \leq C) \geq \theta\}$
- $L = [\langle V, S \rangle]$
- $f(x_1, \ldots, x_3) =$
  $\mathbb{E}\left[p_1x_1 + p_2x_2 + p_3x_3 - s \max\left(0, \sum_{i=1}^{3} w_ix_i - 20\right)\right]$

Stochastic Constraint Programming (Walsh, 2002)

An SCOP: Dynamic Stochastic Knapsack

- $V = \{x_1, \ldots, x_3\}$
- $D(x_i) = \{0, 1\} \quad \forall i \in \{1, \ldots, 3\}$
- $S = \{w_1, \ldots, w_3\}$
- $P(w_1) = \{5(0.5), 8(0.5)\}$,
  $P(w_2) = \{3(0.5), 9(0.5)\}$,
  $P(w_3) = \{15(0.5), 4(0.5)\}$
- $C = \{Pr(w_1x_1 + w_2x_2 + w_3x_3 \leq C) \geq \theta\}$
- $L = [\langle\{x_1\}, \{w_1\}\rangle, \langle\{x_2\}, \{w_2\}\rangle, \langle\{x_3\}, \{w_3\}\rangle]$
- $f(x_1, \ldots, x_3) =$
  $\mathbb{E}\left[p_1x_1 + p_2x_2 + p_3x_3 - s \max\left(0, \sum_{i=1}^{3} w_ix_i - 20\right)\right]$

Stochastic OPL (Tarim et al., 2006)

A **language** specifically introduced for **modeling decision problems under uncertainty**.

It captures several **high level concepts** that facilitate the process of modeling uncertainty:

- stochastic variables (independent or conditional distributions)
- several probabilistic measures for the objective function (expectation, variance, etc.)
- chance-constraints
- decision stages
- ...

Stochastic OPL (Tarim et al., 2006)

Static Stochastic Knapsack

```
int N = 3;
int c = 10;
int p = 2;
float θ = 0.2
range Object [1..3];
int value[Object] = [8,15,10];
stoch int weight[Object] = [<5(0.5),8(0.5)>,
                            <3(0.5),9(0.5)>,<15(0.5),4(0.5)>];
var int+ X[Object] in 0..1;
stages = [<X,weight>];
var int+ z;

maximize sum(i in Object) X[i]*value[i] - p*z
subject to{
z = max(0,expected(sum(i in Object) X[i]*weight[i] - c));
prob(sum(i in Object) X[i]*weight[i] - c ≤ 0) ≥ θ;
};
```

Stochastic OPL (Tarim et al., 2006)

Dynamic Stochastic Knapsack

```
int N = 3;
int c = 10;
int p = 2;
float θ = 0.2
range Object [1..3];
int value[Object] = [8,15,10];
stoch int weight[Object] = [<5(0.5),8(0.5)>,
                            <3(0.5),9(0.5)>,<15(0.5),4(0.5)>];
var int+ X[Object] in 0..1;
stages = [<X[1],weight[1]>,<X[2],weight[2]>,<X[3],weight[3]>];
var int+ z;

maximize sum(i in Object) X[i]*value[i] - p*z
subject to{
z = max(0,expected(sum(i in Object) X[i]*weight[i] - c));
prob(sum(i in Object) X[i]*weight[i] - c ≤ 0) ≥ θ;
};
```

Stochastic OPL (Tarim et al., 2006)

By using the approach discussed in

S. A. Tarim, S. Manandhar and T. Walsh,
Stochastic Constraint Programming: A Scenario-Based Approach,
Constraints, Vol.11, pp.53-80, 2006

it is possible to compile any SCSP/SCOP down to a deterministic equivalent CSP.

Stochastic OPL (Tarim et al., 2006)

Static Stochastic Knapsack — deterministic equivalent model

```
int nbWorlds = 8;
range Worlds 1..nbWorlds;
int nbItems = 3;
range Items 1..nbItems;
float s = 0.2
float W[Worlds,Items] =
[[5,3,15],[5,3,4],[5,9,15],[5,9,4],[8,3,15],[8,3,4],[8,9,15],[8,9,4]];
float Pr[Worlds] = [0.125,0.125,0.125,0.125,0.125,0.125,0.125,0.125];

float p[Items] = [8,15,10];
float C = 10;

var float+ z[Worlds];
var int+ w[Worlds] in 0..1;
var int+ x[Items] in 0..1;

maximize sum(i in Items) x[i]*r[i] - p*(sum(j in Worlds)Pr[j]*z[j])
subject to{
forall(j in Worlds) z[j] >= (sum(i in Items)W[j,i]*x[i])-C;
forall(j in Worlds) (sum(i in Items)W[j,i]*x[i] <= C) => w[i]==1;
sum(j in Worlds) Pr[j]*w[j] >= 0.2;
};
```

Stochastic OPL (Tarim et al., 2006)

Advantages

- **Seamless** modeling under uncertainty
- **Stochastic OPL** not necessarily linked to CP

Drawbacks

- **Size** of the compiled model
- **Constraint propagation** not fully supported

Decision making under uncertainty

Can we reuse existing propagation algorithms, e.g. alldifferent, in a stochastic setting?

Global stochastic constraints & Global chance constraints

Synthesizing Filtering Algorithms for Global Chance-Constraints*

Brahim Hnich[1], Roberto Rossi[2], S. Armagan Tarim[3], and Steven Prestwich[4]

[1] Faculty of Computer Science, Izmir University of Economics, Turkey
brahim.hnich@ieu.edu.tr
[2] Logistics, Decision and Information Sciences, Wageningen UR, The Netherlands
roberto.rossi@wur.nl
[3] Operations Management Division, Nottingham University Business School, UK
armtar@yahoo.com
[4] Cork Constraint Computation Centre, University College Cork, Ireland
s.prestwich@4c.ucc.ie

Abstract. Stochastic Constraint Satisfaction Problems (SCSPs) are a powerful modeling framework for problems under uncertainty. To solve them is a P-Space task. The only solution approach to date compiles down SCSPs into classical CSPs. This allows the reuse of classical constraint solvers to solve SCSPs, but at the cost of increased space requirements and weak constraint propagation. This paper tries to overcome some of these drawbacks by automatically synthesizing filtering algorithms for global chance-constraints. These filtering algorithms are parameterized by propagators for the deterministic version of the chance-constraints. This approach allows the reuse of existing propagators in current constraint solvers and it enhances constraint propagation. Experiments show the benefits of this novel approach.

1 Introduction

Stochastic Constraint Satisfaction Problems (SCSPs) are a powerful modeling framework for problems under uncertainty. SCSPs were first introduced in [10] and further extended in [9] to permit multiple chance-constraints and a range of different objectives in order to model combinatorial problems under uncertainty. SCSP is a PSPACE-complete problem [10]. The approach in [9] compiles down SCSPs into deterministic equivalent CSPs. This makes it possible to reuse existing solvers, but at the cost of increased space requirements and of hindering constraint propagation. In this paper we overcome some of these drawbacks by automatically synthesizing filtering algorithms for global chance-constraints. These filtering algorithms are built around propagators for the deterministic version of the chance-constraints. Like the approach in [9], our approach reuses the propagators already available for classical CSPs. But, unlike [9], our approach uses fewer decision variables and strengthens constraint propagation. Our results

* Brahim Hnich is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027.

I.P. Gent (Ed.): CP 2009, LNCS 5732, pp. 439–453, 2009.
© Springer-Verlag Berlin Heidelberg 2009

International Conference on Principles and Practice of Constraint Programming, 2008

Artificial Intelligence 189 (2012) 69–94

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Filtering algorithms for global chance constraints ☆

Brahim Hnich [a], Roberto Rossi [b,*], S. Armagan Tarim [c], Steven Prestwich [d]

[a] Department of Computer Engineering, Izmir University of Economics, Turkey
[b] University of Edinburgh Business School, 29 Buccleuch Place, EH8 9JS, Edinburgh, UK
[c] Department of Management, Hacettepe University, Ankara, Turkey
[d] Cork Constraint Computation Centre, University College Cork, Ireland

ARTICLE INFO

Article history:
Received 8 June 2011
Received in revised form 7 May 2012
Accepted 10 May 2012
Available online 14 May 2012

Keywords:
Stochastic constraint programming
Stochastic constraint satisfaction
Global chance constraints
Filtering algorithms
Stochastic alldifferent

ABSTRACT

Stochastic Constraint Satisfaction Problems (SCSPs) are a powerful modeling framework for problems under uncertainty. To solve them is a PSPACE task. The only complete solution approach to date — scenario-based stochastic constraint programming — compiles SCSPs down into classical CSPs. This allows the reuse of classical constraint solvers to solve SCSPs, but at the cost of increased space requirements and weak constraint propagation. This paper tries to overcome these drawbacks by automatically synthesizing filtering algorithms for global chance constraints. These filtering algorithms are parameterized by propagators for the deterministic version of the chance constraints. This approach allows the reuse of existing propagators in current constraint solvers and it has the potential to enhance constraint propagation. Our results show that, for the test bed considered in this work, our approach is superior to scenario-based stochastic constraint programming. For these instances, our approach is more scalable, it produces more compact formulations, it is more efficient in terms of run time and more effective in terms of pruning for both stochastic constraint satisfaction and optimization problems.
© 2012 Elsevier B.V. All rights reserved.

1. Introduction

In this work we consider problems in which we are required to make decisions under uncertainty. The word *uncertainty* is used to characterize the existence, in these problems, of uncontrollable or "random" variables,[1] which cannot be influenced by the decision maker. In addition to random variables, the problems we consider also comprise controllable or "decision" variables, to which a value from given domains has to be assigned. More specifically, a problem classified as *deterministic* with respect to the degree of uncertainty does not include random variables, while a *stochastic* problem does. Random variables are typically employed to model factors such as the customer demand for a certain product, the crop yield of a given piece of land during a year, the arrival rate of orders at a reservation center and so forth. A continuous or discrete domain of possible values that can be observed is associated with each random variable. A probabilistic measure — typically a probability distribution — over such a domain is assumed to be available in order to fully quantify the likelihood of each value (respectively, range of values in the continuous case) that appears in the domain. The decision making process comprises one or more consecutive *decision stages*. In a decision stage, a decision is taken by the decision

☆ This work is an extended version of Hnich et al. (2009) [11]. S. Armagan Tarim is supported by Hacettepe University (HU-BAB) and the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. 110M500.
* Corresponding author. Tel.: +44 (0) 131 651 4389.
E-mail addresses: brahim.hnich@ieu.edu.tr (B. Hnich), roberto.rossi@ed.ac.uk (R. Rossi), armagan.tarim@hacettepe.edu.tr (S.A. Tarim), s.prestwich@4c.ucc.ie (S. Prestwich).
[1] Alternatively, in the literature, these variables are also denoted as "stochastic".

0004-3702/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.artint.2012.05.001

Artificial Intelligence, 2012

Global stochastic constraints & Global chance constraints

A **global stochastic constraint** (Rossi et al., 2008) represents a relation among a non-predefined number of **decision** and **random** variables.

It implements dedicated filtering algorithms based on

- **feasibility** reasoning
- **optimality** reasoning

Stochastic Programming

$\Pr\left\{\sum_{i=1}^{k} w_i x_i \leq C\right\} \geq \theta$

$x_i \rightarrow$ decision variable

$w_i \rightarrow$ random variable or constant parameter

C random variable or constant parameter

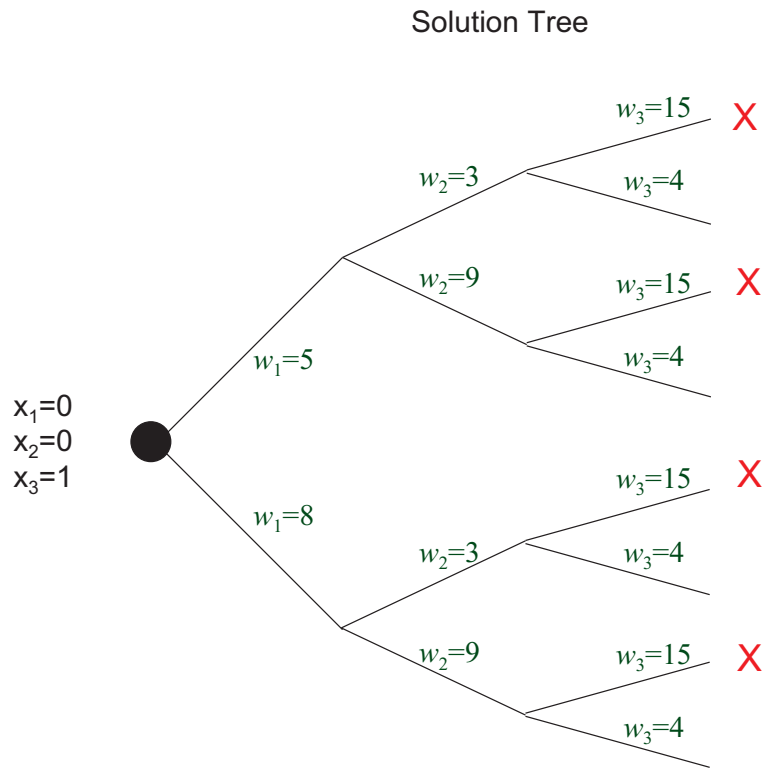
$\theta \rightarrow$ satisfaction probability

Global Chance Constraint

`stochLinIneq`$(x_1, \ldots, x_k, w_1, \ldots, w_k, C, \theta)$;

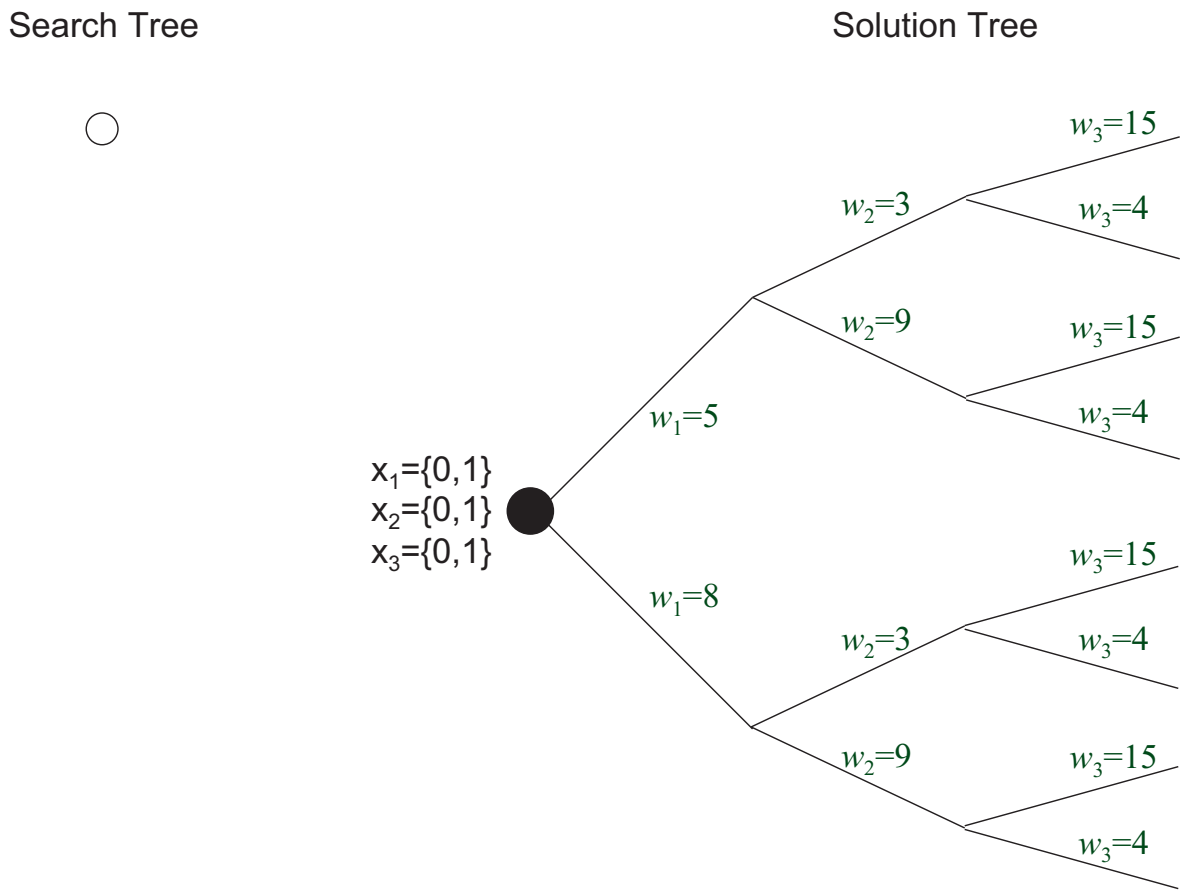
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$



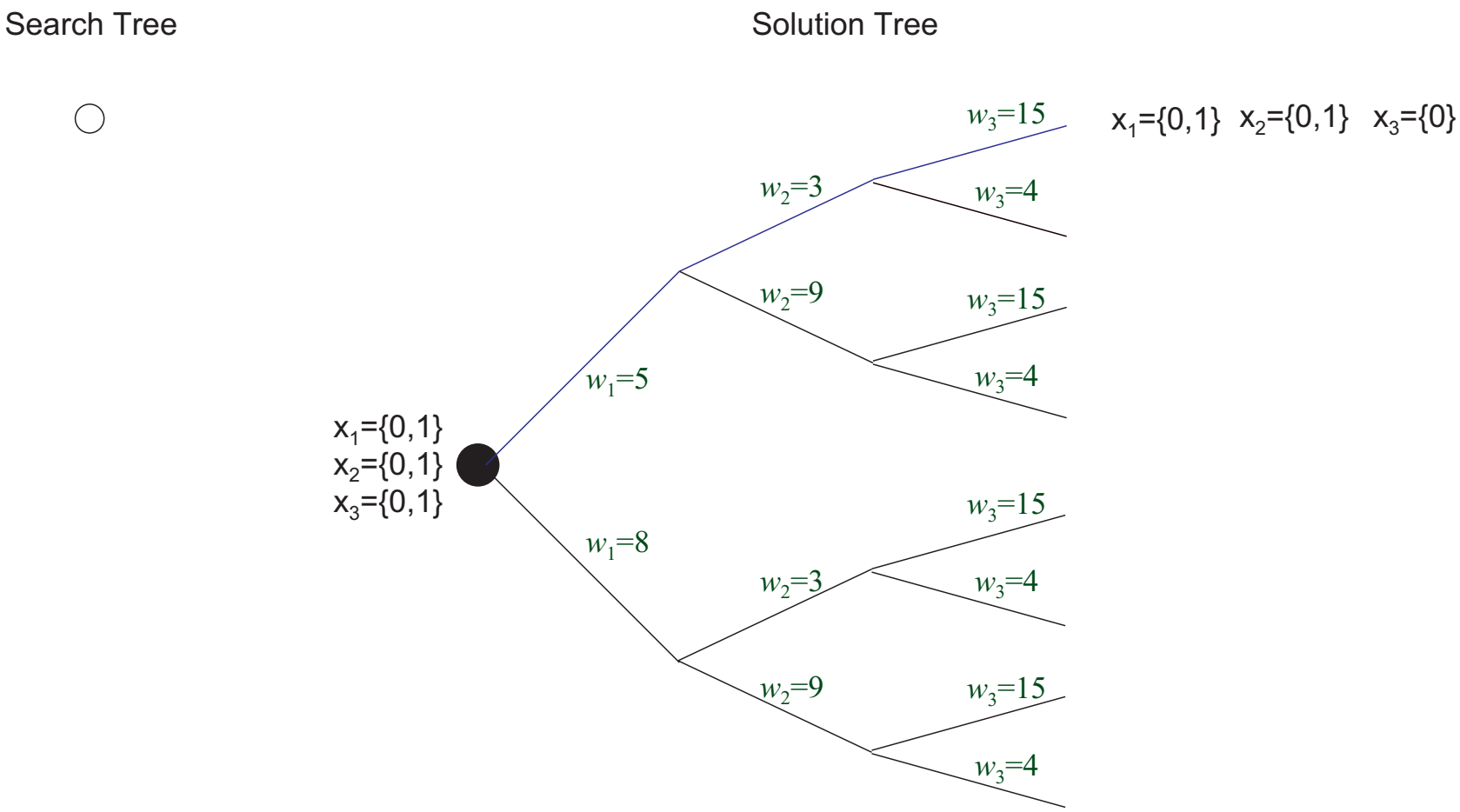
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$



Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$

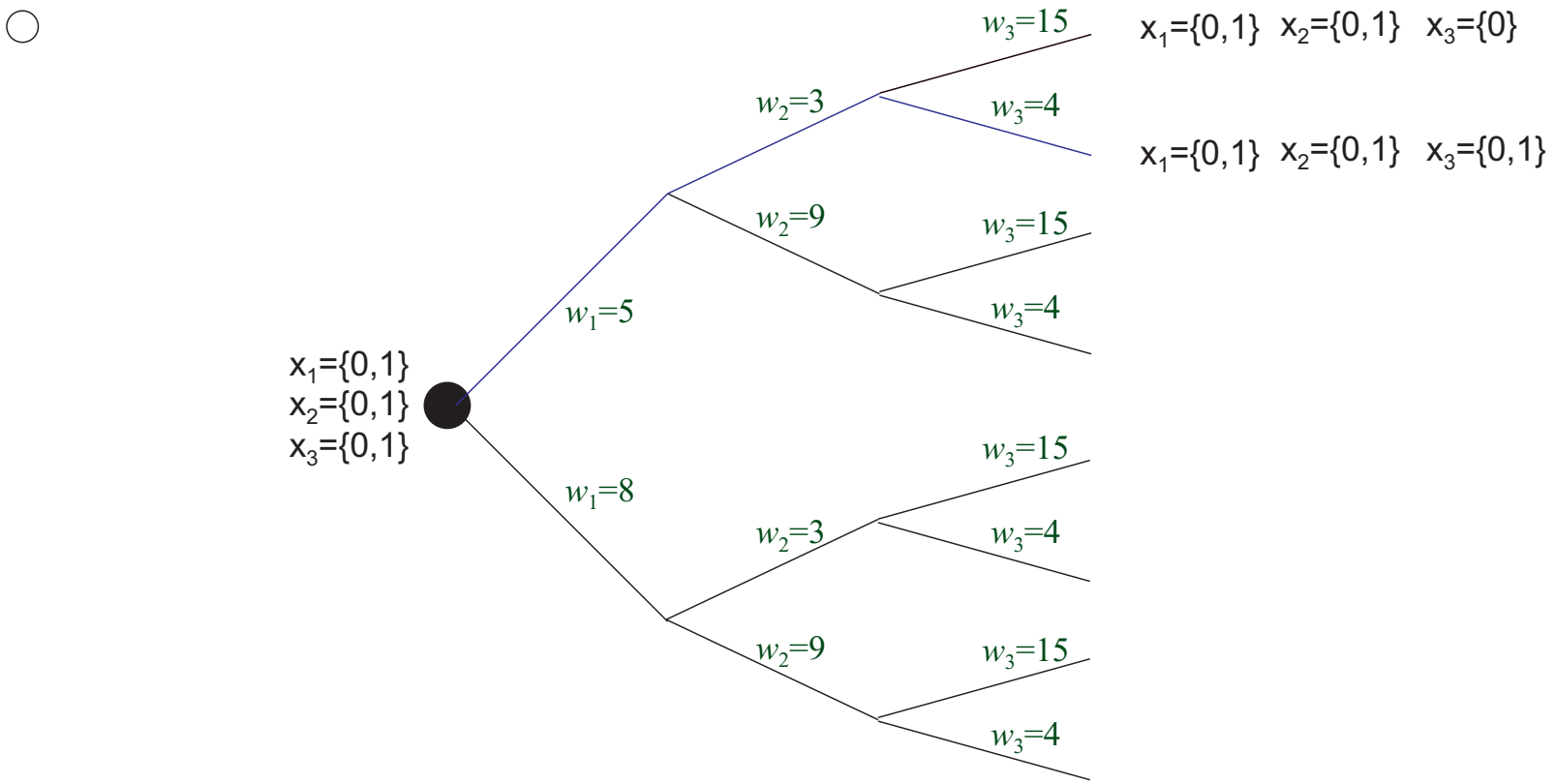


Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$

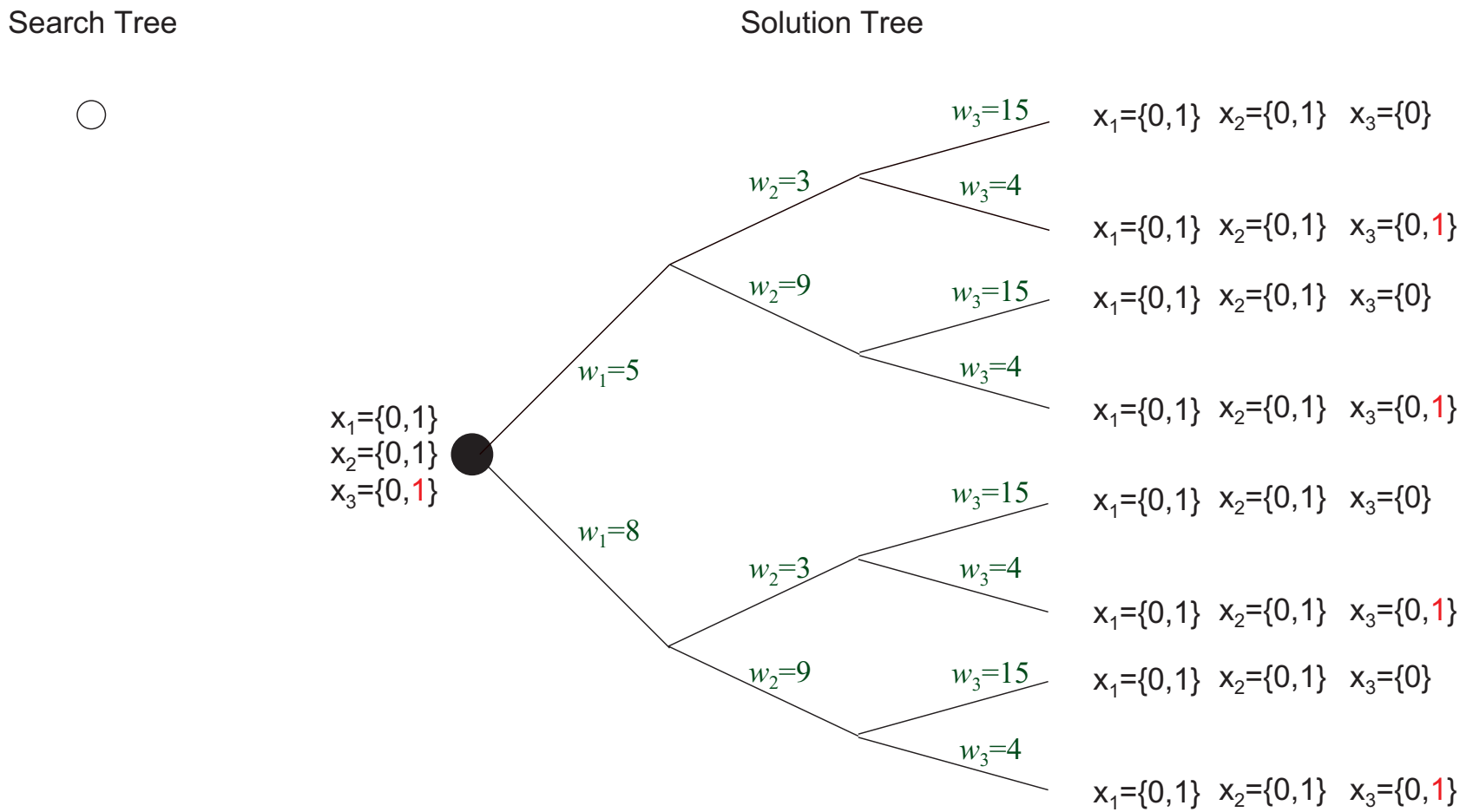
Search Tree

Solution Tree



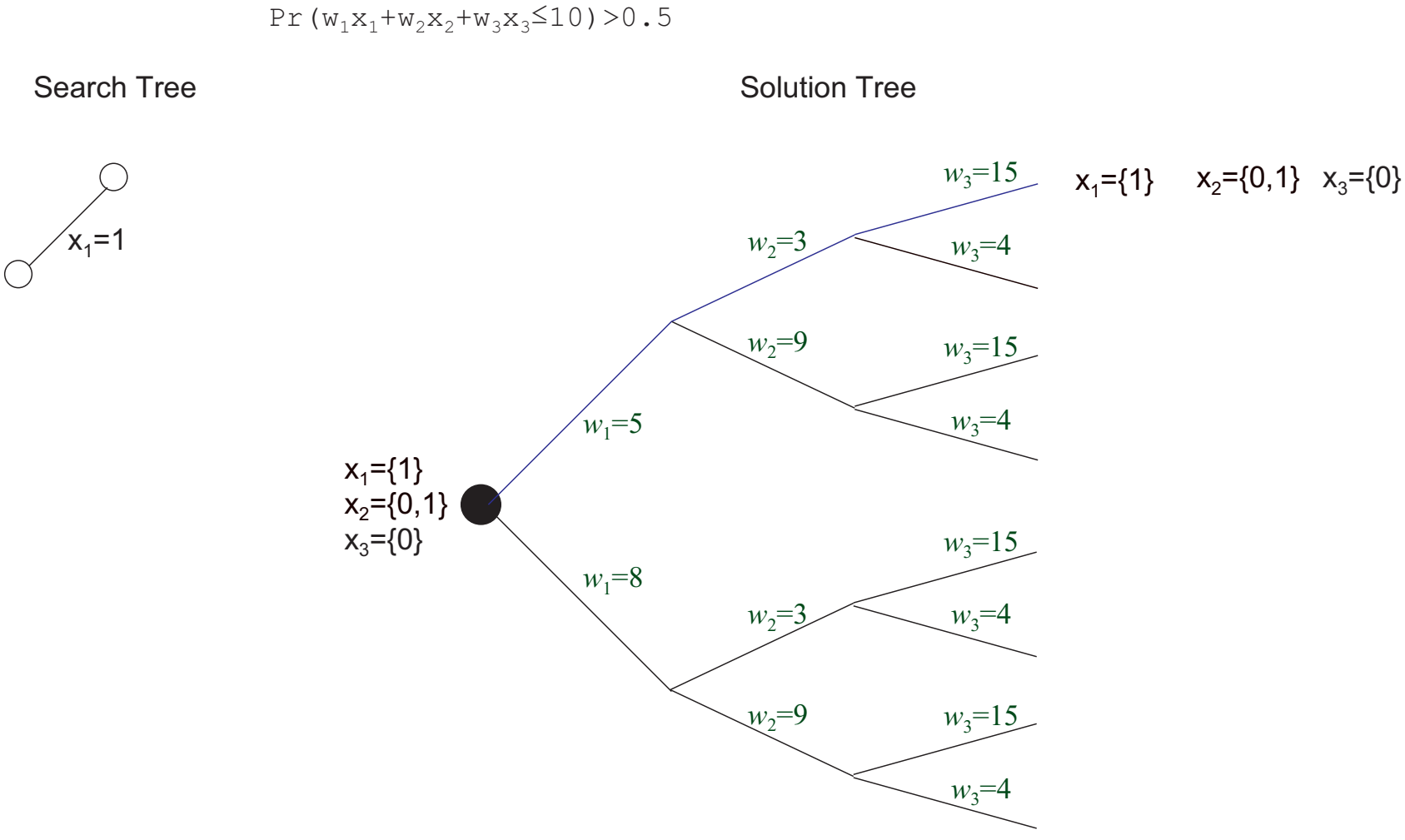
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$



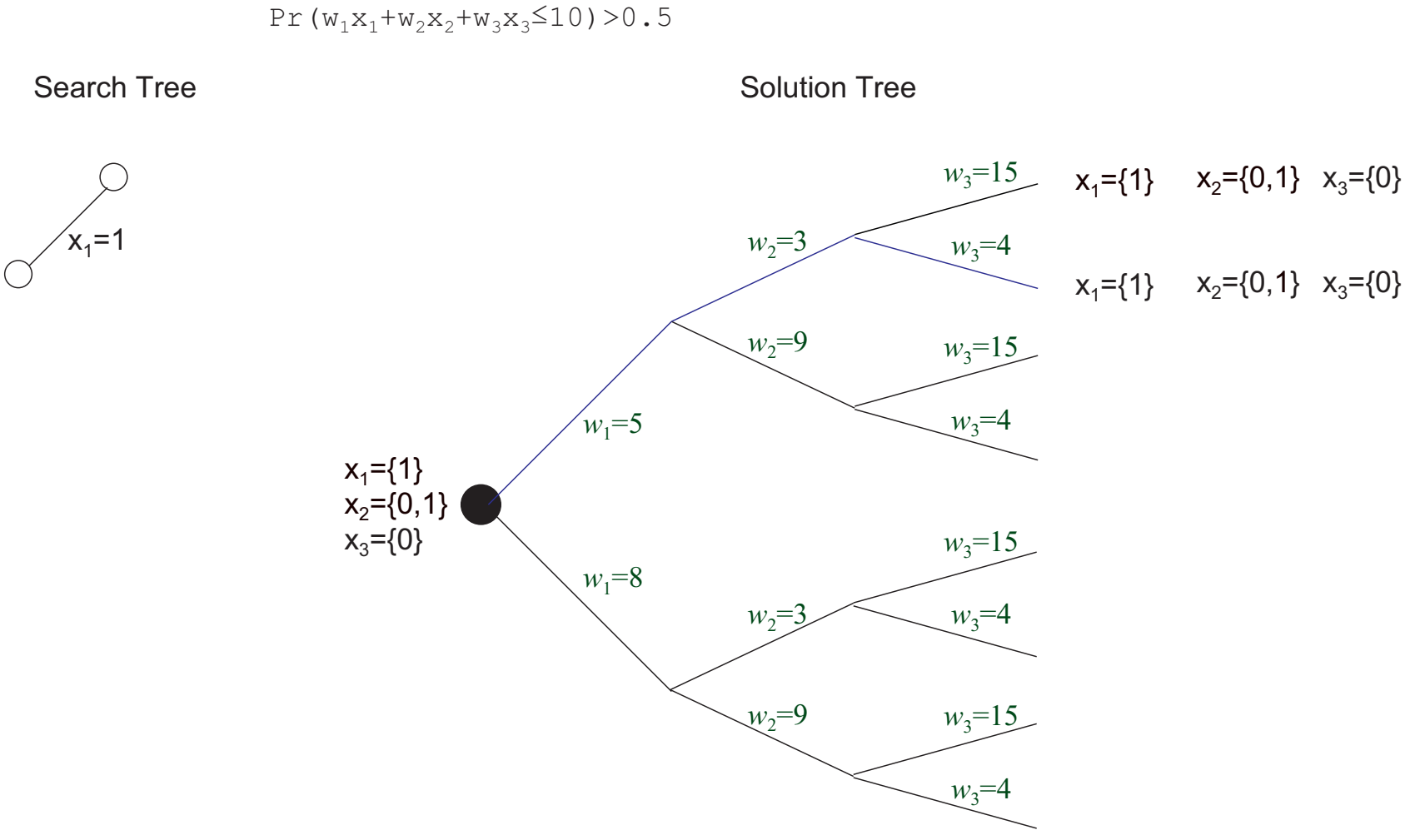
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$



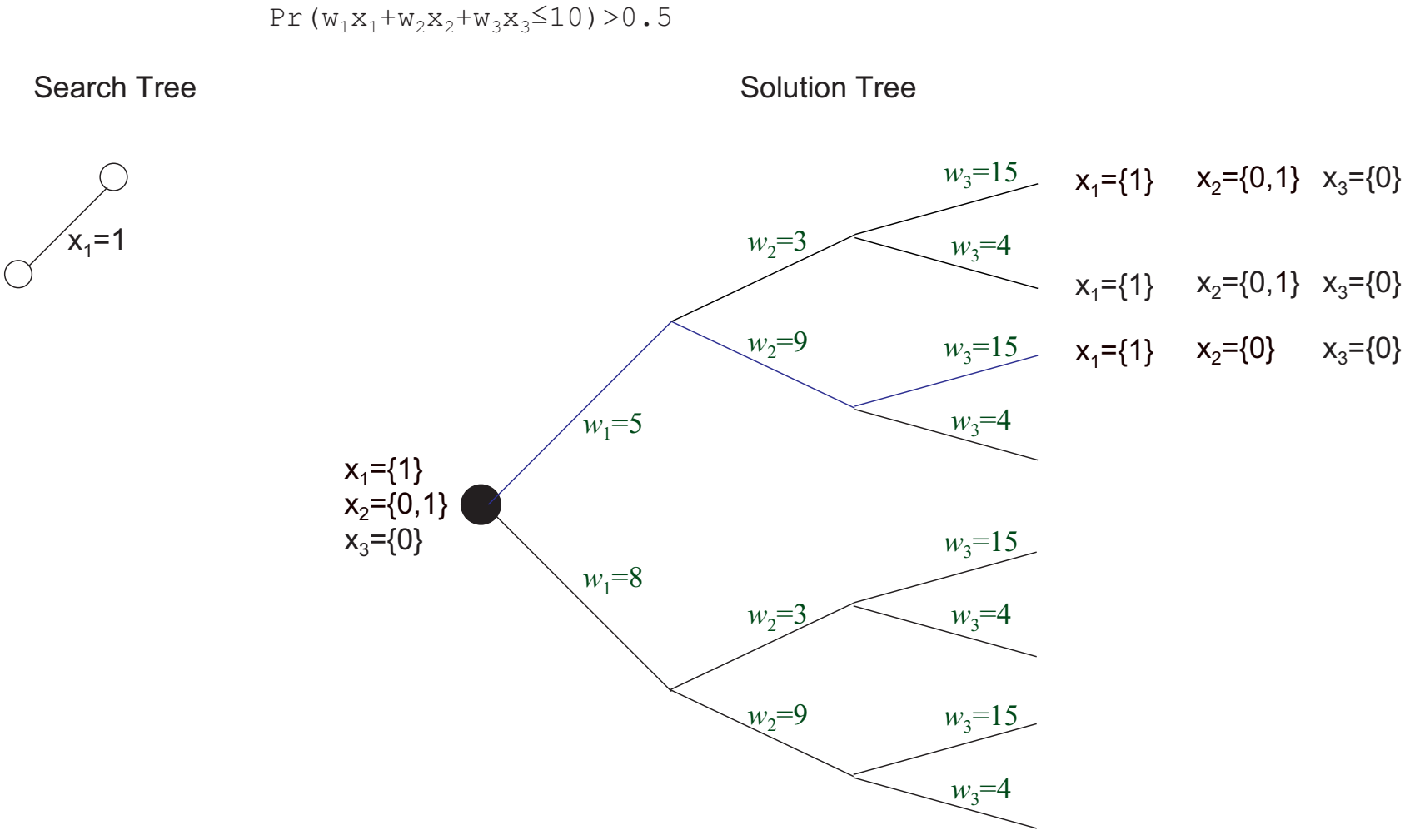
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$



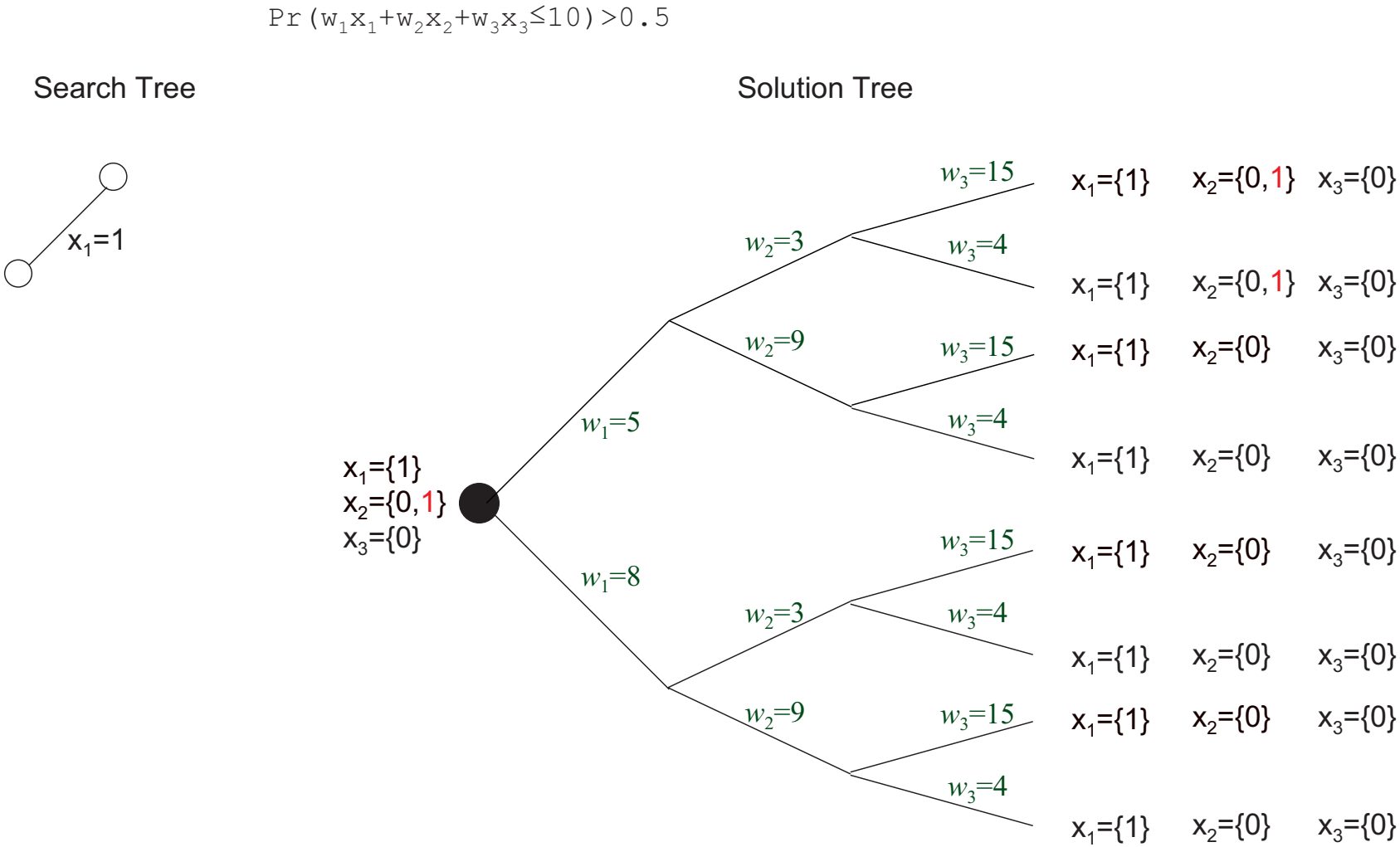
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$



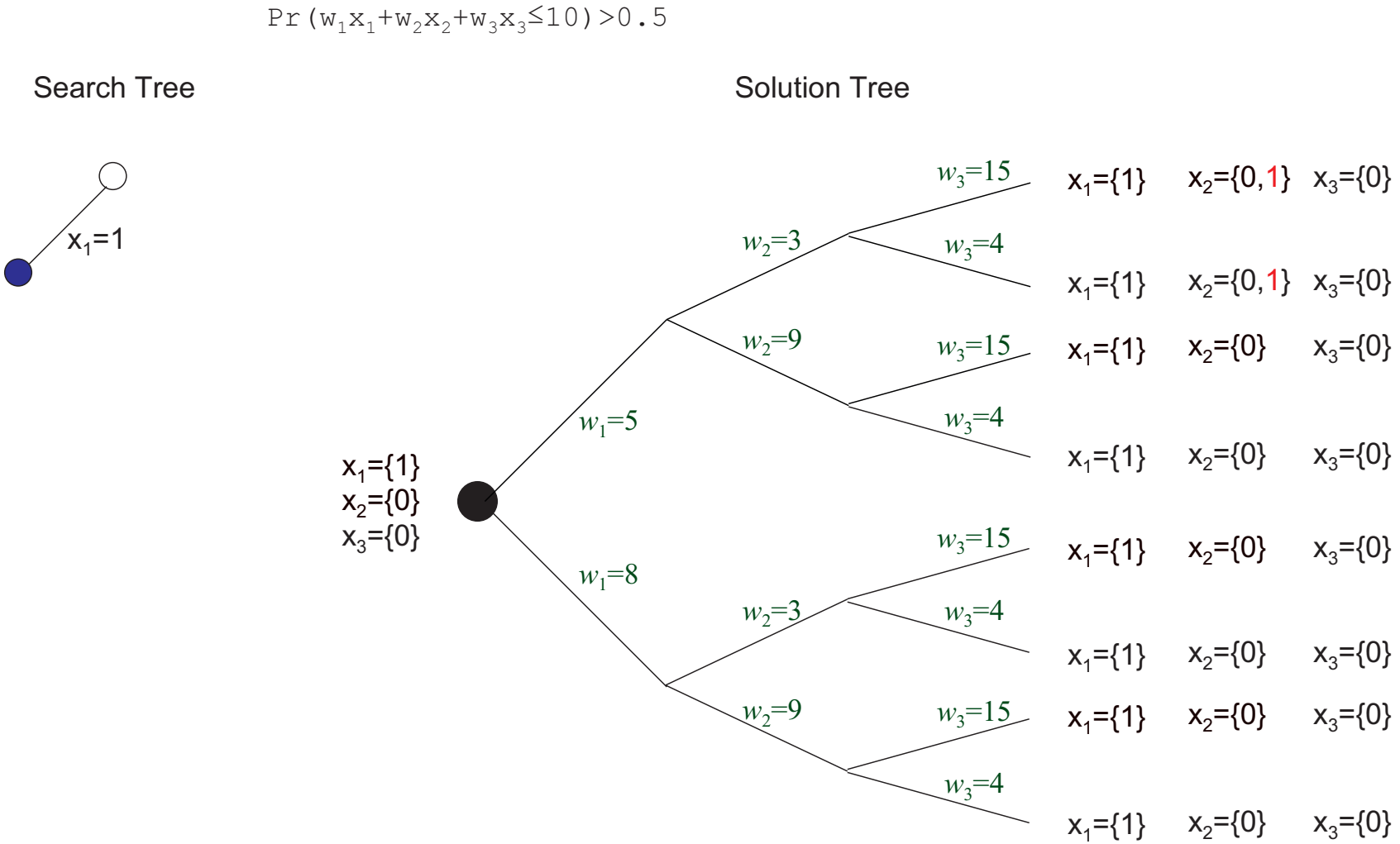
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$



Filtering algorithms for global chance constraints: an example

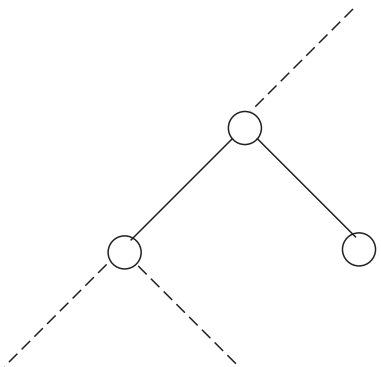
$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$



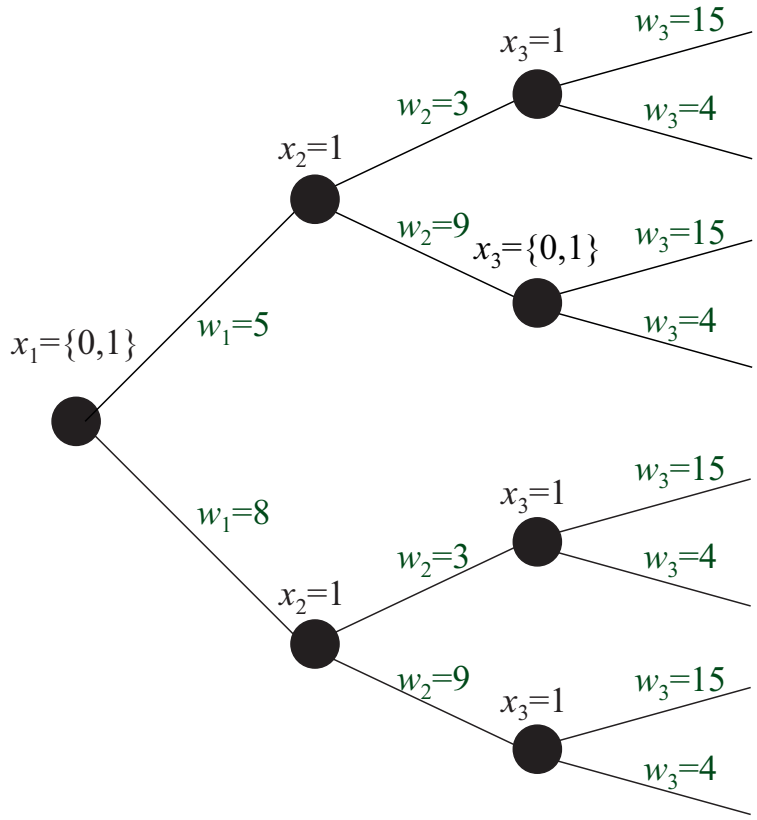
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



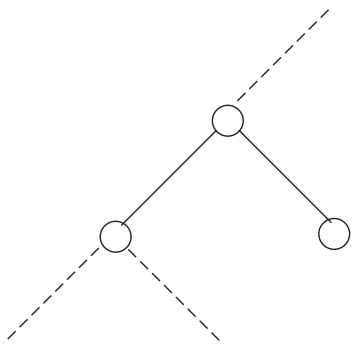
Solution Tree



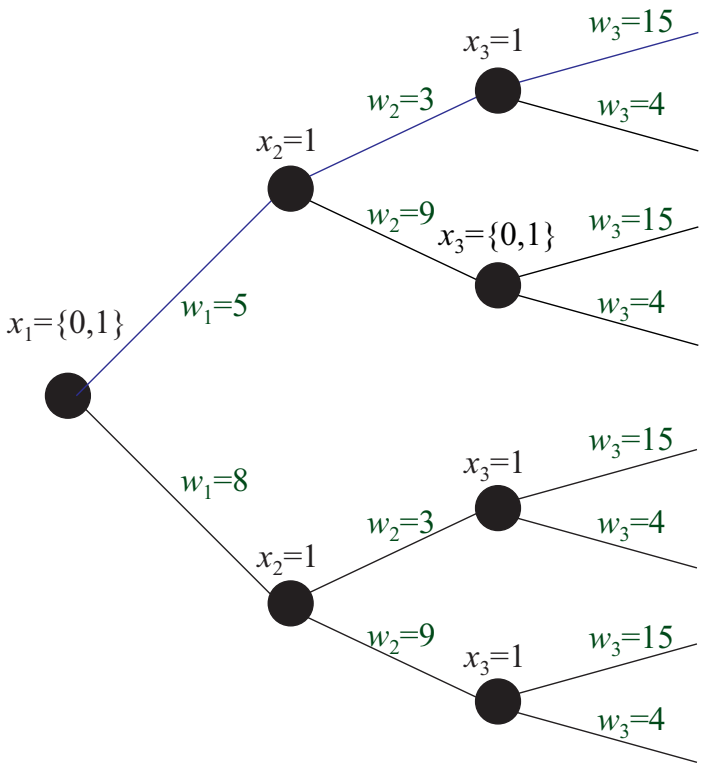
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



Solution Tree

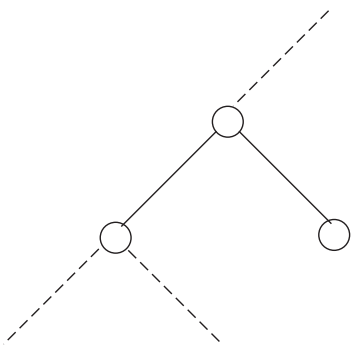


x_1={} x_2={1} x_3={1}

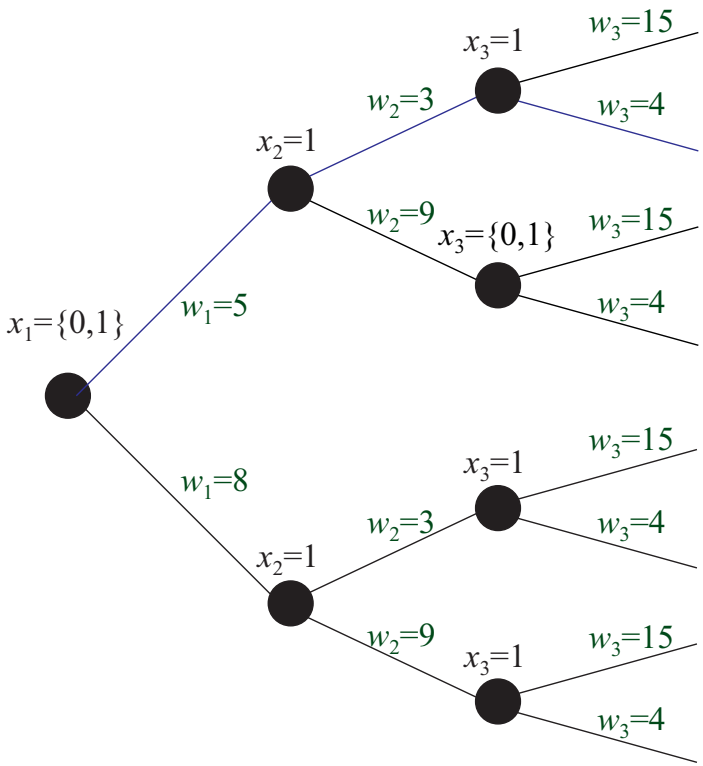
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



Solution Tree



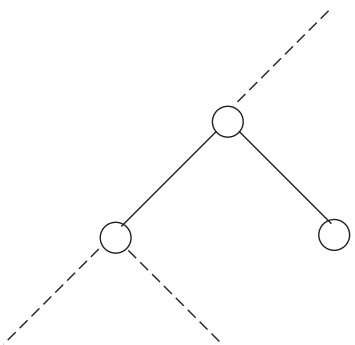
x_1={} x_2={1} x_3={1}

x_1={0} x_2={1} x_3={1}

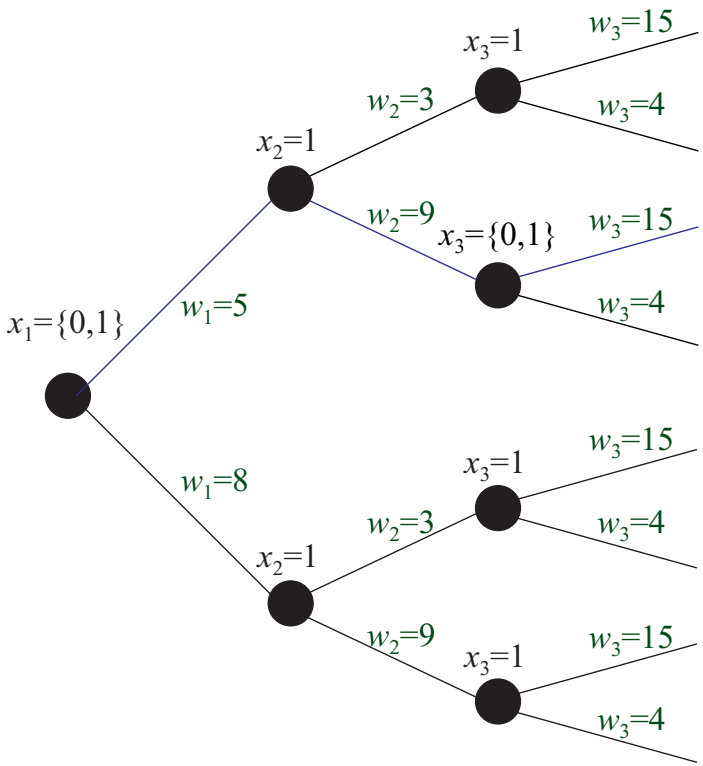
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



Solution Tree



x_1={} x_2={1} x_3={1}

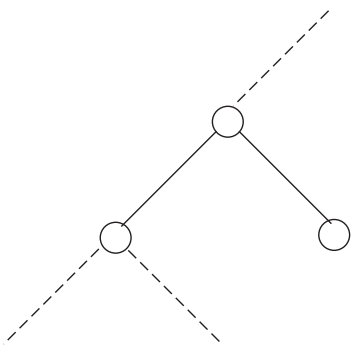
x_1={0} x_2={1} x_3={1}

x_1={0} x_2={1} x_3={0}

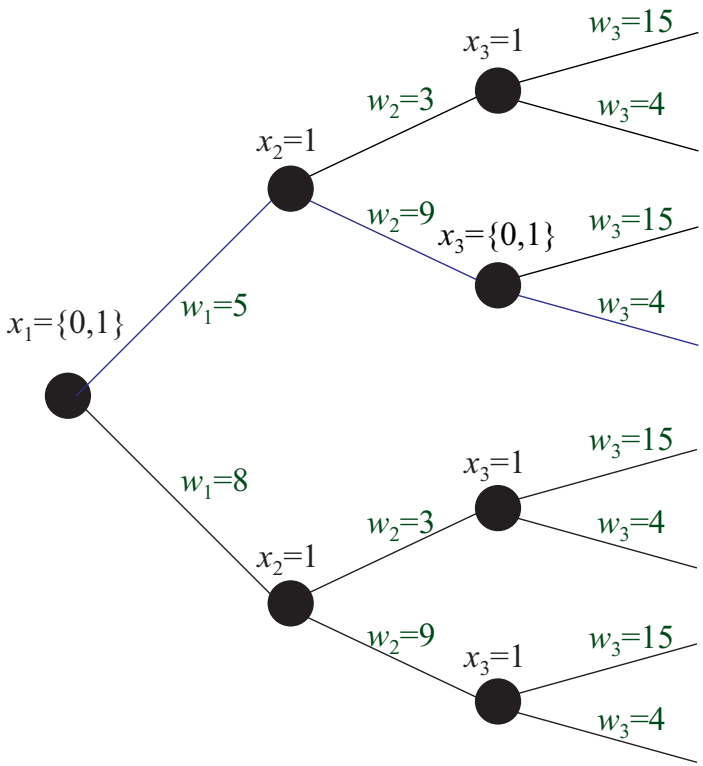
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



Solution Tree



x_1={} x_2={1} x_3={1}

x_1={0} x_2={1} x_3={1}

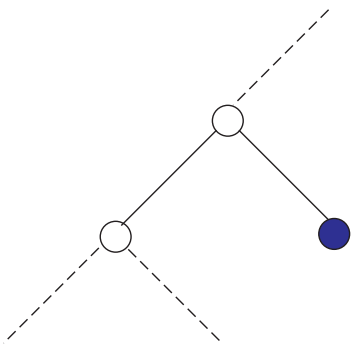
x_1={0} x_2={1} x_3={0}

x_1={0} x_2={1} x_3={0}

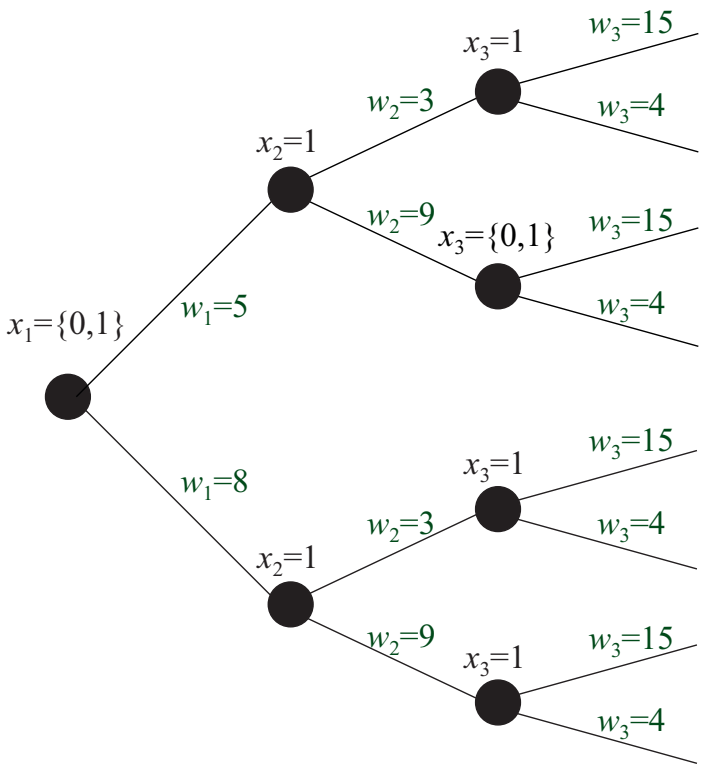
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



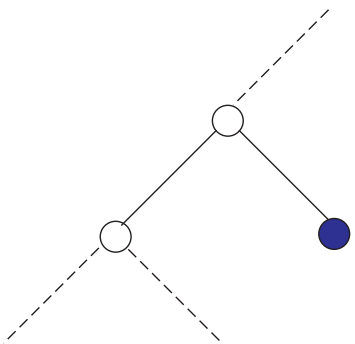
Solution Tree



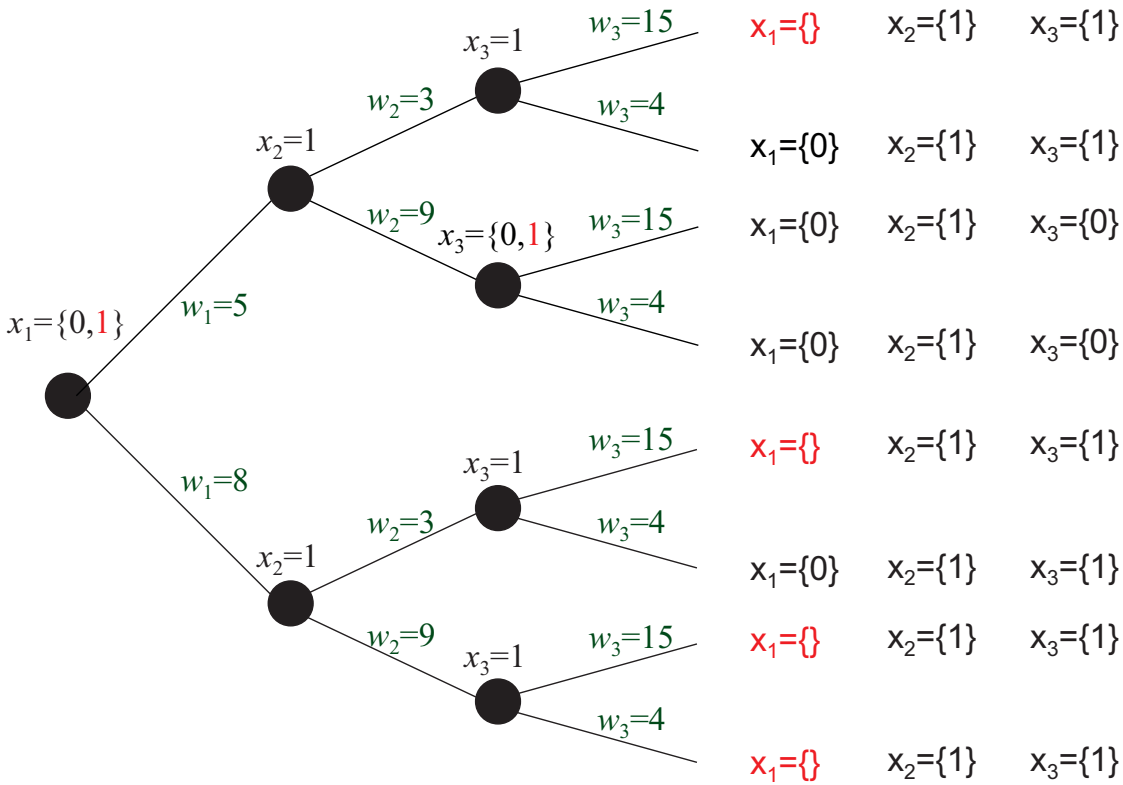
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



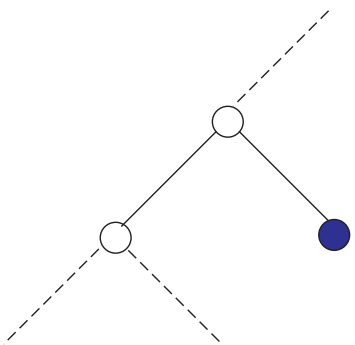
Solution Tree



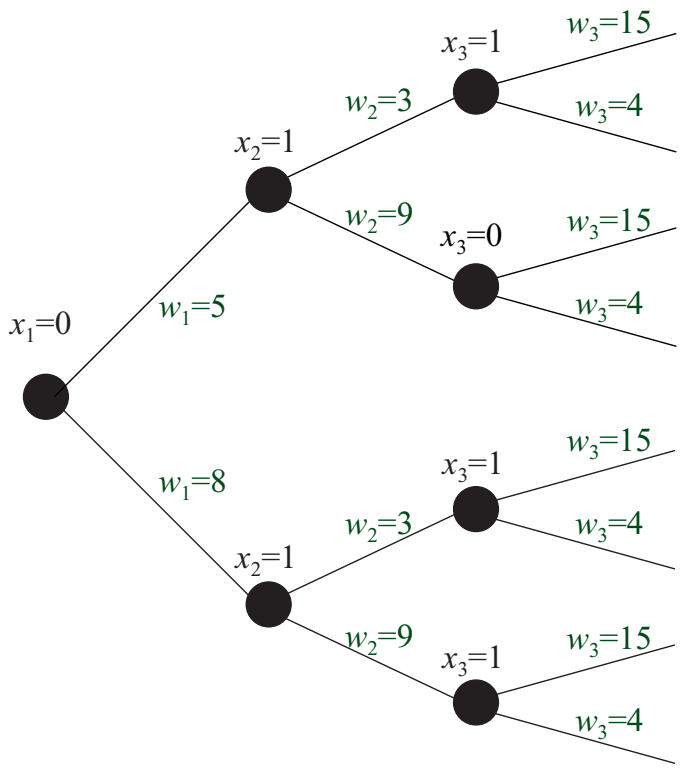
Filtering algorithms for global chance constraints: an example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



Solution Tree



x_1={}	x_2={1}	x_3={1}	
x_1={0}	x_2={1}	x_3={1}	
x_1={0}	x_2={1}	x_3={0}	
x_1={0}	x_2={1}	x_3={0}	
x_1={}	x_2={1}	x_3={1}	
x_1={0}	x_2={1}	x_3={1}	
x_1={}	x_2={1}	x_3={1}	
x_1={}	x_2={1}	x_3={1}	

Global chance constraints vs Scenario-based approach

Random SCSPs in (Hnich et al., 2012)

Global stochastic constraints & Global chance constraints

What are the advantages of modelling problems of decision making under uncertainty via global stochastic constraints?

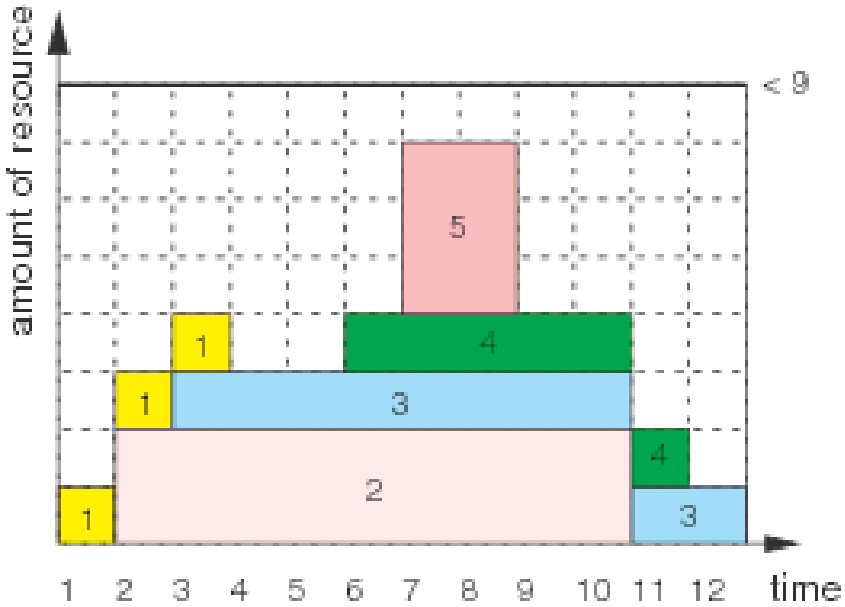
Deterministic multiprocessor scheduling (CSP)

Constraints:

(1) $\text{cumulative}(s, e, t, c, m)$

Decision variables:

$$s_k \in \{r_k, \ldots, d_k\}, \qquad \forall k \in 1, \ldots, |K|$$
$$e_k \in \{r_k, \ldots, d_k\}, \qquad \forall k \in 1, \ldots, |K|$$



Stochastic multiprocessor scheduling (SCSP)

Uncertain processing time

Constraints:

(1) $\Pr\{\text{cumulative}(s, e, t, c, m)\} \geq \theta$

Decision variables:

$s_k \in \{r_k, \ldots, d_k\}, \quad \forall k \in 1, \ldots, |K|$

$e_k \in \{r_k, \ldots, d_k\}, \quad \forall k \in 1, \ldots, |K|$

Stochastic variables:

$t_k \rightarrow$ processing time of order $k \quad \forall k \in 1, \ldots, |K|$

Stage structure:

$V_1 = \{s_1, s_2, \ldots, s_{|K|}\} \quad S_1 = \{t_1, t_2 \ldots, t_{|K|}\}$

$V_2 = \{e_1, e_2, \ldots, e_{|K|}\} \quad S_2 = \{\}$

$L = [\langle V_1, S_1 \rangle, \langle V_2, S_2 \rangle]$

Stochastic multiprocessor scheduling (SCSP)

Uncertain capacity requirements

Constraints:

(1) $\Pr\{\text{cumulative}(s, e, t, c, m)\} \geq \theta$

Decision variables:

$s_k \in \{r_k, \ldots, d_k\}, \quad \forall k \in 1, \ldots, |K|$

$e_k \in \{r_k, \ldots, d_k\}, \quad \forall k \in 1, \ldots, |K|$

Stochastic variables:

$c_k \rightarrow$ capacity requirement of order $k \quad \forall k \in 1, \ldots, |K|$

Stage structure:

$V_1 = \{s_1, s_2, \ldots, s_{|K|}, t_1, t_2 \ldots, t_{|K|}, e_1, e_2, \ldots, e_{|K|}\}$

$S_1 = \{c_1, c_2, \ldots, c_{|K|}\}$

$L = [\langle V_1, S_1 \rangle]$

Stochastic multiprocessor scheduling (SCSP)

Uncertain processing time and capacity requirements

Constraints:

(1) $\Pr\{\text{cumulative}(s, e, t, c, m)\} \geq \theta$

Decision variables:

$s_k \in \{r_k, \ldots, d_k\}, \quad \forall k \in 1, \ldots, |K|$

$e_k \in \{r_k, \ldots, d_k\}, \quad \forall k \in 1, \ldots, |K|$

Stochastic variables:

$t_k \rightarrow$ processing time of order $k \quad \forall k \in 1, \ldots, |K|$

$c_k \rightarrow$ capacity requirement of order $k \quad \forall k \in 1, \ldots, |K|$

Stage structure:

$V_1 = \{s_1, s_2, \ldots, s_{|K|}\} \quad S_1 = \{t_1, t_2 \ldots, t_{|K|}, c_1, c_2, \ldots, c_{|K|}\}$

$V_2 = \{e_1, e_2, \ldots, e_{|K|}\} \quad S_2 = \{\}$

$L = [\langle V_1, S_1 \rangle, \langle V_2, S_2 \rangle]$

Stochastic Plane Landing Scheduling Problem

Uncertain plane delay

Constraints:

(1) $t_i = x_i + d_i \qquad \forall i \in P1, \ldots, P5$

(2) $l_i = x_i + f_i + d_i \qquad \forall i \in P1, \ldots, P5$

(3) $b_i = x_i + d_i \qquad \forall i \in P6, \ldots, P14$

(4) $\Pr\{\text{alldiff}(t_1, l_3, t_5, b_6, b_7, b_8)\} \geq \theta_4$

(5) $\Pr\{\text{alldiff}(l_1, t_2, l_4, b_9, b_{10}, b_{11})\} \geq \theta_5$

(6) $\Pr\{\text{alldiff}(l_2, t_3, t_4, l_5, b_{12}, b_{13}, b_{14})\} \geq \theta_6$

(7) $\Pr\{l_1 \leq t_2\} \geq \theta_7$

(8) $l_3 \leq t_5$

Decision variables:

$x_1 \in \{1, \ldots, 12\}, \quad x_2 \in \{13, 14, 15, 22, 23, 24\},$
$x_3 \in \{1, \ldots, 9\}, \quad x_4 \in \{7, \ldots, 12\},$
$x_5 \in \{16, \ldots, 24\},$
$x_6 = 18, \; x_7 = 10, \; x_8 = 2,$
$x_9 = 15, \; x_{10} = 8, \; x_{11} = 2,$
$x_{12} = 13, \; x_{13} = 5, \; x_{14} = 2,$
$t_i \in \{1, \ldots, 28\} \quad \forall i \in 1, \ldots, 5$
$l_i, \in \{1, \ldots, 40\} \quad \forall i \in 1, \ldots, 5$
$b_i, \in \{1, \ldots, 22\} \quad \forall i \in 6, \ldots, 14$

Stochastic variables:

$d_i \rightarrow$ delay of plane $i \quad \forall i \in 1, \ldots, 14$

Stage structure:

$V_1 = \{x_1, x_2, \ldots, x_{14}\}$
$S_1 = \{d_1, d_2 \ldots, d_{14}\}$
$V_2 = \{t_1, \ldots, t_5, l_1, \ldots, l_5, b_6, \ldots, b_{14}\}$
$S_2 = \{\}$
$L = [\langle V_1, S_1 \rangle, \langle V_2, S_2 \rangle]$

Global stochastic constraints & Global chance constraints

Can we scale these propagation methods when random variable supports are large?

Global stochastic constraints & Global chance constraints

Numerical study

A comprehensive numerical study on several problems

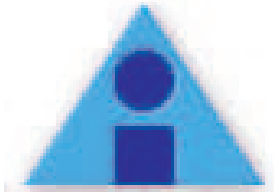
- Random stochastic CSPS
- Stochastic knapsack (SCOP)
- Stochastic plane landing scheduling problem

is presented in (Hnich et al., 2012).

In this numerical study we demonstrates the effectiveness of global chance constraints in modelling and solving problems of decision making under uncertainty.

We also demonstrate that global chance constraints outperform classical scenario-based deterministic equivalent reformulation such as the one discussed in (Tarim et al., 2006).

(α, ϑ)-solutions and Sampled SCSP



22nd International Joint Conference on Artificial Intelligence (IJCAI-11)

Finding (α, ϑ)-solutions via Sampled SCSPs

Roberto Rossi,[1] Brahim Hnich,[2] S. Armagan Tarim[3] and Steven Prestwich[4]
[1]Logistics, Decision and Information Sciences, Wageningen University, the Netherlands, roberto.rossi@wur.nl
[2]Department of Computer Engineering, Izmir University of Economics, Turkey, brahim.hnich@ieu.edu.tr
[3]Department of Management, Hacettepe University, Ankara, Turkey, armagan.tarim@hacettepe.edu.tr
[4]Cork Constraint Computation Centre,University College Cork, Ireland, s.prestwich@4c.ucc.ie

Abstract

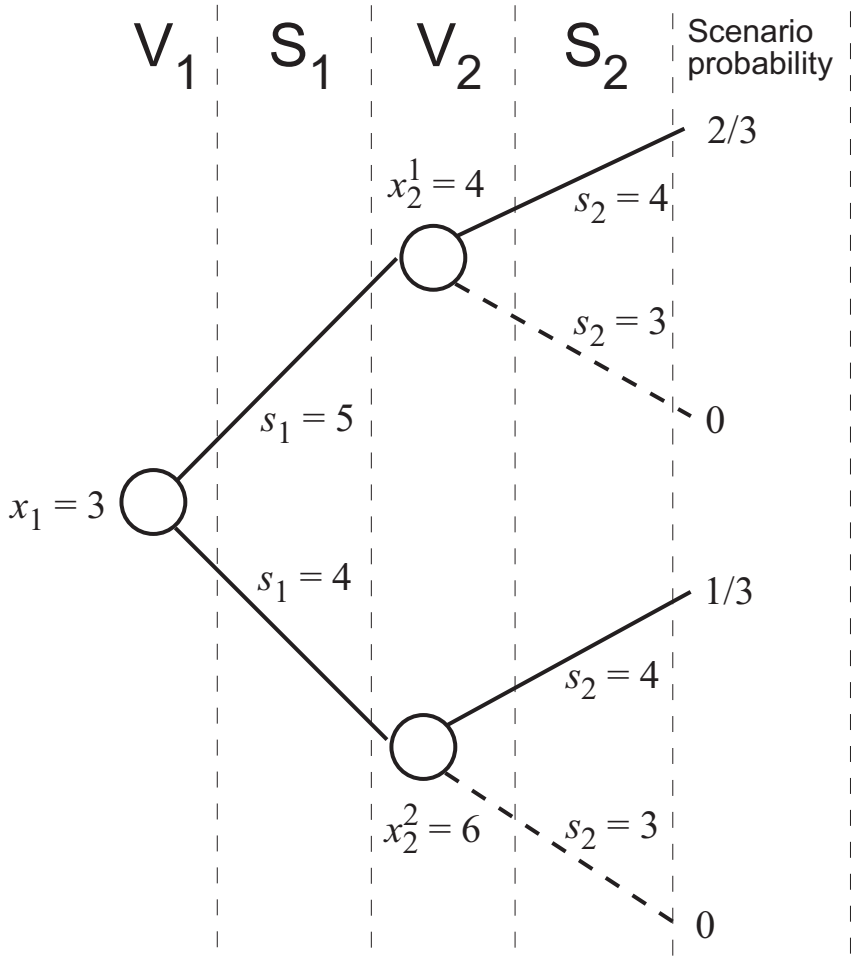
We discuss a novel approach for dealing with single-stage stochastic constraint satisfaction problems (SCSPs) that include random variables over a continuous or large discrete support. Our approach is based on two novel tools: sampled SCSPs and (α, ϑ)-solutions. Instead of explicitly enumerating a very large or infinite set of future scenarios, we employ statistical estimation to determine if a given assignment is consistent for a SCSP. As in statistical estimation, the quality of our estimate is determined via confidence interval analysis. In contrast to existing approaches based on sampling, we provide likelihood guarantees for the quality of the solutions found. Our approach can be used in concert with existing strategies for solving SCSPs.

formally introduce the concept of (α, ϑ)-solution and we apply it to SCSPs that include continuous random variables — i.e. an infinite number of scenarios — and that cannot be solved by exact approaches in the stochastic constraint programming literature.

2 Formal Background

An m-stage SCSP [Walsh, 2002; Tarim *et al.*, 2006; Hnich *et al.*, 2009] is defined as a 7-tuple $\langle V, S, D, P, C, \beta, L \rangle$, where V is a set of decision variables and S is a set of random variables, D is a function mapping each element of V (respectively, S) to a domain (respectively, support) of potential values. In classical SCSPs both decision variable domains and random variable supports are assumed to be finite. P is a function mapping each element of S to a probability distribution for its associated support. C is a set of chance-constraints over a non-empty subset of decision variables and a subset

(α, ϑ)-solutions and Sampled SCSP



Questions