

Foundations of Artificial Intelligence

D6. Constraint Satisfaction Problems: Constraint Graphs

Malte Helmert

University of Basel

April 15, 2024

Foundations of Artificial Intelligence

April 15, 2024 — D6. Constraint Satisfaction Problems: Constraint Graphs

D6.1 Constraint Graphs

D6.2 Unconnected Graphs

D6.3 Trees

D6.4 Summary

Constraint Satisfaction Problems: Overview

Chapter overview: constraint satisfaction problems

- ▶ D1–D2. Introduction
- ▶ D3–D5. Basic Algorithms
- ▶ D6–D7. Problem Structure
 - ▶ D6. Constraint Graphs
 - ▶ D7. Decomposition Methods

D6.1 Constraint Graphs

Motivation

- ▶ To solve a constraint network consisting of n variables and k values, k^n assignments must be considered.
- ▶ Inference can alleviate this combinatorial explosion, but will not always avoid it.
- ▶ Many practically relevant constraint networks are efficiently solvable if their **structure** is taken into account.

Constraint Graphs

Definition (constraint graph)

Let $\mathcal{C} = \langle V, \text{dom}, (R_{uv}) \rangle$ be a constraint network.

The **constraint graph** of \mathcal{C} is the graph whose vertices are V and which contains an edge $\{u, v\}$ iff R_{uv} is a nontrivial constraint.

Constraint Graphs: Running Example

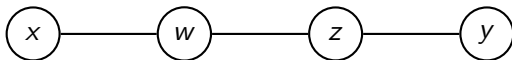
Nontrivial Constraints of Running Example

$$R_{wx} = \{\langle 2, 1 \rangle, \langle 4, 2 \rangle\}$$

$$R_{wz} = \{\langle 1, 2 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle\}$$

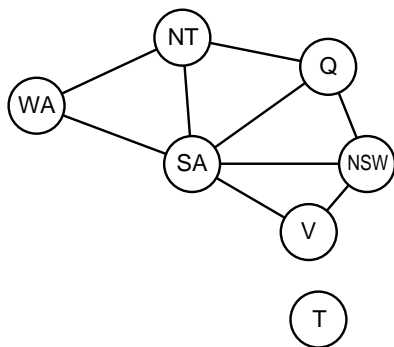
$$R_{yz} = \{\langle 2, 1 \rangle, \langle 3, 1 \rangle, \langle 3, 2 \rangle, \langle 4, 1 \rangle, \langle 4, 2 \rangle, \langle 4, 3 \rangle\}$$

Resulting Constraint Graph:



Constraint Graphs: Better Example

Coloring of the Australian states (plus Northern Territory)



D6.2 Unconnected Graphs

Unconnected Constraint Graphs

Proposition (unconnected constraint graphs)

If the constraint graph of \mathcal{C} has multiple connected components, the subproblems induced by each component can be solved separately.

The union of the solutions of these subproblems is a solution for \mathcal{C} .

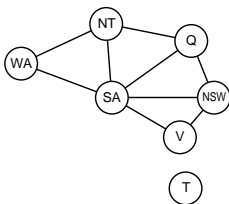
Proof.

A total assignment consisting of combined subsolutions satisfies all constraints that occur **within** the subproblems.

All constraints **between** two subproblems are trivial (follows from the definitions of constraint graphs and connected components). □

Unconnected Constraint Graphs: Example

example: Tasmania can be colored independently from the rest of Australia.



further example:

network with $k = 2$, $n = 30$ that decomposes into three components of equal size

savings?

only $3 \cdot 2^{10} = 3072$ assignments instead of $2^{30} = 1073741824$

D6.3 Trees

Trees as Constraint Graphs

Proposition (trees as constraint graphs)

Let \mathcal{C} be a constraint network with n variables and maximal domain size k whose constraint graph is a *tree* or *forest* (i.e., does not contain cycles).

Then we can solve \mathcal{C} or prove that no solution exists in time $O(nk^2)$.

example: $k = 5, n = 10$

$\rightsquigarrow k^n = 9765625, nk^2 = 250$

Trees as Constraint Graphs: Algorithm

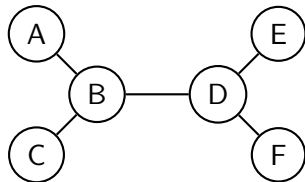
algorithm for trees:

- ▶ Build a **directed** tree for the constraint graph.
Select an arbitrary variable as the root.
- ▶ Order variables v_1, \dots, v_n such that parents are ordered before their children.
- ▶ For $i \in \langle n, n-1, \dots, 2 \rangle$: call `revise($v_{\text{parent}(i)}, v_i$)`
 \rightsquigarrow each variable is arc consistent with respect to its children
- ▶ If a domain becomes empty, the problem is unsolvable.
- ▶ Otherwise: solve with `BacktrackingWithInference`, variable order v_1, \dots, v_n and forward checking.
 \rightsquigarrow solution is found **without backtracking steps**

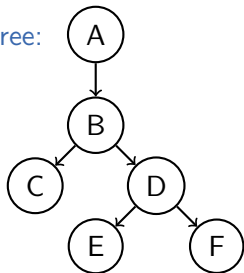
proof: \rightsquigarrow exercises

Trees as Constraint Graphs: Example

1. constraint graph:



2. directed tree:



4. revise steps:

- ▶ revise(D, F)
- ▶ revise(D, E)
- ▶ revise(B, D)
- ▶ revise(B, C)
- ▶ revise(A, B)

3. order:



5. finding a solution:

backtracking with forward checking and order

$A \prec B \prec C \prec D \prec E \prec F$

D6.4 Summary

Summary

- ▶ Constraint networks with **simple structure** are easy to solve.
- ▶ **Constraint graphs** formalize this structure:
 - ▶ **several connected components**:
solve **separately** for each component
 - ▶ **tree**: algorithm **linear** in number of variables