Foundations of Artificial Intelligence

27. Constraint Satisfaction Problems: Constraint Graphs

Martin Wehrle

Universität Basel

April 18, 2016

Constraint Satisfaction Problems: Overview

Chapter overview: constraint satisfaction problems

- 22.–23. Introduction
- 24.–26. Basic Algorithms
- 27.–28. Problem Structure
  - 27. Constraint Graphs
  - 28. Decomposition Methods

Constraint Graphs

Motivation

- To solve a constaint 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 between u and v iff R_{uv} is a nontrivial constraint.

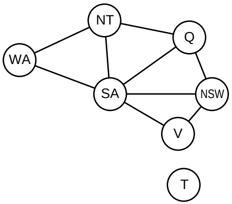
Constraint Graphs

Definition (constraint graph)

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

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

example: coloring of the Australian states and territories



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 $\mathcal{C}$.

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.

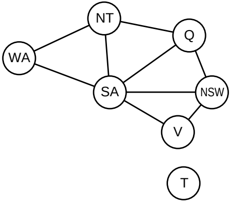
The union of the solutions of these subproblems is a solution for $\mathcal{C}$.

Proof.

A total assignment consisting of combined subsolutions satisfies all constraints that occur within the subproblems.
From the definitions of constraint graphs and connected components, all nontrivial constraints are within a subproblem. □

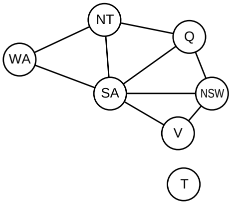
Unconnected Constraint Graphs: Example

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



Unconnected Constraint Graphs: Example

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



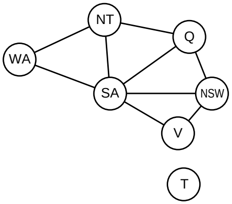
further example:

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

savings?

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 $2^{30} = 1073741824$

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: $k = 5, n = 10$
$\rightsquigarrow k^n = 9765625, \; nk^2 = 250$

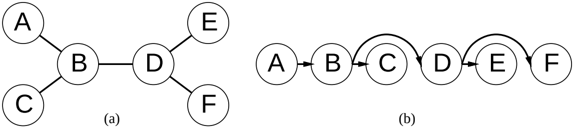
Trees as Constraint Graphs: Algorithm

algorithm for trees:

- Build a directed tree for the constraint graph.
  Select an arbitrary variable as the root.
- Order variables $v_1, \ldots, v_n$ such that parents are ordered before their children.
- For $i \in \langle n, n-1, \ldots, 2 \rangle$: 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 $v_1, \ldots, v_n$ and forward checking.
  $\rightsquigarrow$ solution is found without backtracking steps

Trees as Constraint Graphs: Example

constraint network $\rightsquigarrow$ directed tree + order:



revise steps:

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

finding a solution:

backtracking with order $A \prec B \prec C \prec D \prec E \prec F$

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