

# Constraint Satisfaction Problems

## Constraint Networks

**Bernhard Nebel, Julien Hué, and Stefan Wöfl**

Albert-Ludwigs-Universität Freiburg

May 7, 2012

# Constraint Satisfaction Problems

May 7, 2012 — Constraint Networks

1 Constraint Networks

2 Projection Networks

3 Constraint Networks and Graphs

4 Solving Constraint Networks

## 1 Constraint Networks

- Constraint networks
- Solution
- Normalized Constraint Networks
- Deduction
- Minimal Network

## Constraint networks

### Definition

A **constraint network** is a triple

$$N = \langle V, \text{dom}, C \rangle$$

where:

- ▶  $V$  is a non-empty and finite set of **variables**;
- ▶  $\text{dom}$  is a function that assigns to each variable  $v \in V$  a non-empty set  $\text{dom}(v)$  ( $\text{dom}(v)$  is called the **domain** of  $v$ , elements of  $\text{dom}(v)$  are called **values**);
- ▶  $C$  is a set of relations over variables of  $V$  (called **constraints**), i.e., each constraint is a relation  $R_{x_1, \dots, x_m}$  over some scheme  $S = (x_1, \dots, x_m)$  of variables in  $V$ .

The set of constraint schemes  $\{S_1, \dots, S_t\}$  is called **network scheme**.

## Constraint networks

If we assume an ordering of the variables in  $V$ , we can write networks more compactly:

### Definition

A **constraint network** is a triple

$$N = \langle V, D, C \rangle$$

where:

- ▶  $V = (v_1, \dots, v_n)$  is a non-empty and finite sequence of **variables**;
- ▶  $D = (D_1, \dots, D_n)$  is a sequence of **domains** for  $V$  ( $D_i$  is the domain of variable  $v_i$ );
- ▶  $C$  is a set of **constraints**  $R_{\bar{x}}$  where  $\bar{x} = (v_{i_1}, \dots, v_{i_m})$  is a scheme of variables in  $V$  and  $R \subseteq D_{i_1} \times \dots \times D_{i_m}$ .

## Example: 4-queens problem

The 4-queens problem can be represented as single constraint network. For example, consider variables  $v_1, \dots, v_4$  (each associated to a column of the  $4 \times 4$ -chess board).

Each variable  $v_i$  has as its domain  $D_i = \{1, \dots, 4\}$  (conceived of as the row positions of a queen in column  $i$ ).

	$v_1$	$v_2$	$v_3$	$v_4$
1				
2				
3				
4				

Define then binary constraints (thus encoding "non-attacking queen positions"):

$$R_{v_1, v_2} := \{(1, 3), (1, 4), (2, 4), (3, 1), (4, 1), (4, 2)\}$$

$$R_{v_1, v_3} := \{(1, 2), (1, 4), (2, 1), (2, 3), (3, 2), (3, 4), (4, 1), (4, 3)\}$$

...

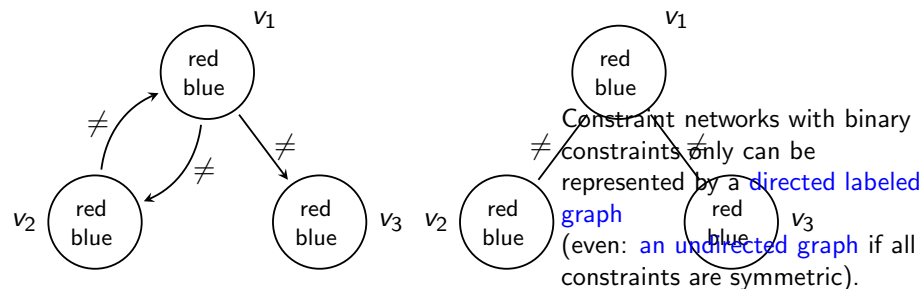
## Example: Graph colorability

$k$ -Colorability of a graph  $G$  can be represented as a constraint network of the following form:

$$V = \{v_i : v_i \text{ is a vertex in } G\}$$

$$D_i = \{1, \dots, k\} \quad (v_i \in V)$$

$$C = \{((v_i, v_j), \neq) : \{v_i, v_j\} \text{ is an edge of } G\}$$



## Solution of a constraint network

### Definition

A **solution** of a constraint network  $N = \langle V, D, C \rangle$  is a **(variable) assignment**

$$a: V \rightarrow \bigcup_{i: v_i \in V} D_i$$

such that

- (a)  $a(v_i) \in D_i$ , for each  $v_i \in V$ ,
- (b)  $(a(x_1), \dots, a(x_m)) \in R$  for each constraints  $R_{x_1, \dots, x_m}$  in  $C$ .

$N$  is called **solvable** (or: **satisfiable**) if  $N$  has a solution.

$Sol(N)$  denotes the set of all solutions of  $N$ .

## Instantiation, partial solution

Let  $N = \langle V, D, C \rangle$  be a constraint network.

### Definition

- (a) An **instantiation** of a subset  $V'$  of  $V$  is an assignment  $a : V' \rightarrow \bigcup_{i: v_i \in V'} D_i$  with  $a(v_i) \in D_i$ .
- (b) An instantiation  $a$  of  $V'$  is called **partial solution** if  $a$  satisfies each constraint  $R_S$  in  $C$  with  $S \subseteq V'$ .  
We also say:  $a$  is **consistent relative** to  $N$ .
- (c) For an instantiation  $a$  of a subset  $V' = \{x_1, \dots, x_m\}$  and a constraint  $R_S$  with scope  $S \subseteq V'$ , let

$$\bar{a}[S] := (a(x_1), \dots, a(x_m)).$$

Hence a solution is an instantiation of all variables in  $V$  that is consistent relative to  $N$ .

## Instantiation, solution

Note:

- (a) An instantiation of variables in  $V' \subseteq V$ ,  $a$ , is a partial solution (consistent relative to  $N$ ) iff

$$\bar{a}[S] \in R, \quad \text{for each constraint } R \text{ with scope } S \subseteq V'.$$

- (b) Not every partial solution is part of a (full) solution, i.e., there may be partial solutions of a constraint network that cannot be extended to a solution. For the 4-queens problem, for example,

	$v_1$	$v_2$	$v_3$	$v_4$
1	q			
2			q	
3				
4		q		

## Normalized constraint network

Let  $N = \langle V, D, C \rangle$  be a constraint network.

Due to our definition it is possible that  $C$  contains constraints

$$R_{v_{i_1}, \dots, v_{i_k}} \quad \text{and} \quad S_{v_{j_1}, \dots, v_{j_k}}$$

where  $(j_1, \dots, j_k)$  is just a permutation of  $(i_1, \dots, i_k)$ .

Without changing the set of solutions, we can simplify the network by deleting  $S_{v_{j_1}, \dots, v_{j_k}}$  from  $C$  and rewriting  $R_{v_{i_1}, \dots, v_{i_k}}$  as follows:

$$R_{v_{i_1}, \dots, v_{i_k}} \leftarrow R_{v_{i_1}, \dots, v_{i_k}} \cap \pi_{v_{i_1}, \dots, v_{i_k}}(S_{v_{j_1}, \dots, v_{j_k}}).$$

Given a fixed order on the set of variables  $V$ , we can systematically delete-and-refine constraints. This results in a constraint network that contains **at most one constraint for each subset of variables**. Such a network is called a **normalized constraint network**.

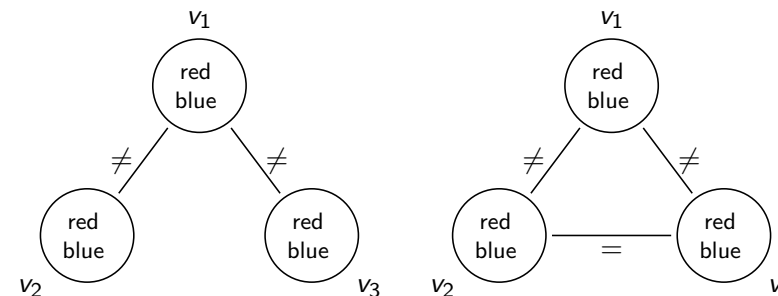
## Equivalence

Let  $N$  and  $N'$  be constraint networks on the same set of variables and on the same domains for each variable.

### Definition

$N$  and  $N'$  are called **equivalent** if they have the same set of solutions.

Example:



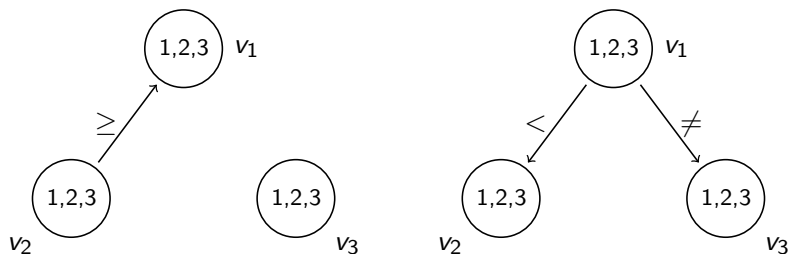
## Tightness

Let  $N$  and  $N'$  be (normalized) constraint networks on the same set of variables and on the same domains for each variable.

### Definition

$N$  is **as tight as**  $N'$  if for each constraint  $R_S$  of  $N$ ,

- (a)  $N'$  has no constraint with the same scope as  $R_S$ , or
- (b)  $R \subseteq \pi_S(R'_{S'})$ , where  $R'_{S'}$  is the constraint of  $N'$  with the same scope as  $R_S$ .



- ▶ Clearly, if  $N'$  is as tight as  $N$ , then  $\text{Sol}(N') \subseteq \text{Sol}(N)$ .
- ▶ Constraint tightness has a large influence on the efficiency of

## Intersection of networks

### Definition

The **intersection** of  $N$  and  $N'$ ,  $N \cap N'$ , is the network defined by intersecting for each scope the constraints  $R_S \in C$  and  $R'_S \in C'$  with the same scope, i.e., modulo a suitable permutation of the constraint schemes,

$$R''_S := R_S \cap R'_S.$$

If for a scope  $S$  only one of the networks contains a constraint, then we set:

$$R''_S := R_S \quad (\text{or } := R'_S, \text{ resp.})$$

### Lemma

If  $N$  and  $N'$  are equivalent networks, then  $N \cap N'$  is equivalent to both networks and as tight as both networks.

## Minimal network

### Definition

Let  $N_0$  be a constraint network and let  $N_1, \dots, N_k$  be the set of **all** constraint networks (defined on the same set of variables and the same domains) that are equivalent to  $N_0$ .

$$\bigcap_{1 \leq i \leq k} N_i$$

is called the **minimal network** of  $N_0$ .

### Lemma

The minimal network is equivalent to and as tight as all the constraint networks  $N_i$ . There is no network equivalent to  $N_0$  that is tighter than the minimal network.

## 2 Projection Networks

## Projecting constraints

Let  $R_S$  be a constraint with scheme  $S = (x_1, \dots, x_m)$  (we can think of  $R_S$  as a constraint network ...).

### Definition

The **projection network** of  $R_S$ ,  $\text{Proj}(R_S)$ , is the constraint network defined by:

$$V := S, \quad D_i := \pi_{x_i}(R_S), \quad R'_{x_i, x_j} := \pi_{x_i, x_j}(R_S)$$

for all variables  $x_i$  and variable pairs  $x_i, x_j$ .

Consider  $R_{x,y,z}$  with  $R = \{(a, a, b), (a, b, b), (a, b, a)\}$ .

Then  $\text{Proj}(R_{x,y,z})$  consists of the following constraints:  $R'_{x,y} = \{(a, a), (a, b)\}$ ,

$R'_{x,z} = \{(a, b), (a, a)\}$ , and

$R'_{y,z} = \{(a, b), (b, b), (b, a)\}$ .

**In this case:**  $\text{Sol}(\text{Proj}(R_{x,y,z})) = R_{x,y,z}$ .

## Projecting constraints

The projection network is an upper approximation by **binary networks** in the following sense:

### Lemma

Any solution of  $R_S$  (as a network) defines a solution of  $\text{Proj}(R_S)$ , i.e.,

$$R_S \subseteq \text{Sol}(\text{Proj}(R_S)).$$

□

### Lemma

$\text{Proj}(R_S)$  is the “tightest” upper approximation of  $R_S$  by binary constraint networks, i.e., there is no binary constraint network  $N'$  defined on the variables of  $R_S$  such that:

$$R \subseteq \text{Sol}(N') \subsetneq \text{Sol}(\text{Proj}(R_S)).$$

## Binary representation

### Definition

A relation  $R_S$  with scope  $S$  **has a binary representation** if the relation (conceived of as a network) is equivalent to  $\text{Proj}(R_S)$ .

From the fact that a relation has a binary representation, it does not follow that all its projections have binary representations as well (Exercise!).

### Definition

A relation  $R_S$  with scope  $S$  is **binary decomposable** if the relation itself and all its projections to subsets of  $S$  (with at least 3 elements) have a binary representation.

## 3 Constraint Networks and Graphs

- Primal Constraint Graphs
- Dual Constraint Graph
- Constraint Hypergraph

## Primal constraint graphs

Let  $N = \langle V, D, C \rangle$  be a (normalized) constraint network.

### Definition

The **primal constraint graph** of a network  $N = \langle V, D, C \rangle$  is the undirected graph

$$G_N := \langle V, E_N \rangle$$

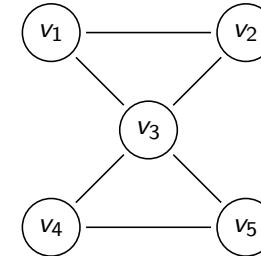
where

$$\{u, v\} \in E_N \iff \{u, v\} \text{ is a subset of the scope of some constraint in } N.$$

## Primal constraint graph: Example

Consider a constraint network with variables  $v_1, \dots, v_5$  and two ternary constraints  $R_{v_1, v_2, v_3}$  and  $S_{v_3, v_4, v_5}$ .

Then the primal constraint graph of the network has the form:



Absence of an edge between two variables/nodes means that there is no **explicit** constraint in which both variables participate.

## Dual constraint graphs

### Definition

The **dual constraint graph** of a constraint network  $N = \langle V, D, C \rangle$  is the labeled graph

$$D_N := \langle V', E_N, I \rangle$$

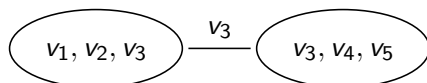
with

$$X \in V' \iff X \text{ is the scope of some constraint in } N$$

$$\{X, Y\} \in E_N \iff X \cap Y \neq \emptyset$$

$$I: E_N \rightarrow 2^V, \{X, Y\} \mapsto X \cap Y$$

In the example above, the dual constraint graph is:



## Constraint hypergraph

### Definition

The **constraint hypergraph** of a constraint network  $N = \langle V, D, C \rangle$  is the hypergraph

$$H_N := \langle V, E_N \rangle$$

with

$$X \in E_N \iff X \text{ is the scope of some constraint in } N.$$

In the example above (constraint network with variables  $v_1, \dots, v_5$  and two ternary constraints  $R_{v_1, v_2, v_3}$  and  $S_{v_3, v_4, v_5}$ ) the hyperedges of the constraint hypergraph are:

$$E_N = \{ \{v_1, v_2, v_3\}, \{v_3, v_4, v_5\} \}.$$

## 4 Solving Constraint Networks

## Simple solution strategy: Backtracking search

**Backtracking:** search systematically for consistent partial instantiations in a depth-first manner:

- ▶ **forward phase:** extend the current partial solution by assigning a consistent value to some new variable (if possible)
- ▶ **backward phase:** if no consistent instantiation for the current variable exists, we return to the previous variable.

## Backtracking algorithm

**Backtracking**( $N, a$ ):

*Input:* a constraint network  $N = \langle V, D, C \rangle$  and  
a partial assignment  $a$  of  $N$   
(e.g., the empty instantiation  $a = \{\}$ )

*Output:* a solution of  $N$  or “inconsistent”

**if**  $a$  is not consistent with  $N$ :

**return** “inconsistent”

**if**  $a$  is defined for all variables in  $V$ :

**return**  $a$

select **some variable**  $v_i$  for which  $a$  is not defined

**for each value**  $x$  from  $D_i$ :

$a' := a \cup \{v_i \mapsto x\}$

$a'' \leftarrow \text{Backtracking}(N, a')$

**if**  $a''$  is not “inconsistent”:

**return**  $a''$

**return** “inconsistent”

## Literature



Rina Dechter.  
Constraint Processing,  
Chapter 2, Morgan Kaufmann, 2003