

Constraint Satisfaction Problems

Global Constraints

Bernhard Nebel, Julien Hué, and Stefan Wöfl

Albert-Ludwigs-Universität Freiburg

July 16, 2009

Constraint Satisfaction Problems

July 16, 2009 — Global Constraints

1 Motivation

2 Filtering

1 Motivation

- Global Constraints
- All-different
- Sum and Cardinality
- Circuit

Global constraints

What are global constraints?

- ▶ Type of similar constraint relations . . .
- ▶ . . . differing in the number of variables
- ▶ **Semantically redundant**: same constraint can be expressed by a conjunction of simpler constraints
- ▶ **Similar structure**: can be exploited by constraint solvers

Examples:

- ▶ sum constraint, knapsack constraint, element constraint, all-different constraint, cardinality constraints

All-different constraint

Definition

Let v_1, \dots, v_n be variables each with a domain D_i ($1 \leq i \leq n$).

$$\text{alldifferent}(v_1, \dots, v_n) := \\ \{(d_1, \dots, d_n) \in D_1 \times \dots \times D_n : d_i \neq d_j \text{ for } i \neq j\}$$

The all-different constraint is a simple, but widely used global constraint in constraint programming.

It allows for compact modeling of CSP problems.

Example: n -Queens Problem

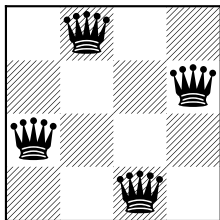


Figure: 4-queens problem

Problem representation:

Variables v_i for each
column $1, \dots, n$;
 v_i can take a “row value”
 $1, \dots, n$.

No-attack constraints:

$$v_i \neq v_j \text{ for } 1 \leq i < j \leq n$$

$$v_i - v_j \neq i - j \text{ for } 1 \leq i < j \leq n$$

$$v_j - v_i \neq i - j \text{ for } 1 \leq i < j \leq n$$

$$\text{alldifferent}(v_1, \dots, v_n)$$

$$\text{alldifferent}(v_1 - 1, \dots, v_n - n)$$

$$\text{alldifferent}(v_1 + 1, \dots, v_n + n)$$

Sum constraint

Let v_1, \dots, v_n, z be variables with subsets of \mathbb{Q} as domain.
 For each v_i , let $c_i \in \mathbb{Q}$ be some fixed scalar, $c = (c_1, \dots, c_n)$.

Definition

The **sum constraint** is defined as:

$$\text{sum}(v_1, \dots, v_n, z; c) := \left\{ (d_1, \dots, d_n, d) \in \left(\prod_{1 \leq i \leq n} D_i \right) \times D_z : d = \sum_{1 \leq i \leq n} c_i d_i \right\}.$$

Global cardinality constraint

v_1, \dots, v_n : “assignment variables” with $D_{v_i} \subseteq \{d_1^*, \dots, d_m^*\}$.

c_1, \dots, c_m : “count variables” with sets of integers as domains.

Definition

The **global cardinality constraint** is defined as:

$$\text{gcc}(v_1, \dots, v_n, c_1, \dots, c_m) :=$$

$$\left\{ (d_1, \dots, d_n, o_1, \dots, o_m) \in \prod_{1 \leq i \leq n} D_{v_i} \times \prod_{1 \leq j \leq m} D_{c_j} : \right.$$

for each j , d_j^* occurs in (d_1, \dots, d_n) exactly o_j times $\left. \right\}$

The global cardinality constraint can be considered a generalization of the all-different constraint.

Circuit constraint

Let $s = (s_1, \dots, s_n)$ be a permutation of $\{1, \dots, n\}$.

Define C_s as the smallest set that contains 1 and with each element i also s_i .

(s_1, \dots, s_n) is called **cyclic** if $C_s = \{1, \dots, n\}$.

Definition

Let v_1, \dots, v_n be variables with domains $D_i = \{1, \dots, n\}$ ($1 \leq i \leq n$).

$$\text{circuit}(v_1, \dots, v_n) := \\ \{(d_1, \dots, d_n) \in D_1 \times \dots \times D_n : (d_1, \dots, d_n) \text{ is cyclic}\}$$

Given an assignment $a = (d_1, \dots, d_n)$, define

$$A := \{(v_i, v_{d_i}) : d_i \in D_i, 1 \leq i \leq n\}.$$

Then, a satisfies $\text{circuit}(v_1, \dots, v_n)$ if and only if (V, A) is a directed cycle (without proper sub-cycles).

Example: Traveling Salesperson Problem

Traveling Salesperson Problem (TSP):

Given a set of n cities and distances c_{ij} between city i and city j , find the shortest route that visits all cities and finishes in the starting city.

TSP is not a constraint satisfaction problem, but a constraint optimization problem ...



Constraint optimization problem

Definition

A **constraint optimization problem** (COP) is a constraint satisfaction problem together with an **objective function** f that assign to each variable assignment a a value $f(a) \in \mathbb{Q}$.

- ▶ **Minimization COP**: Find a solution a that minimizes $f(a)$.
- ▶ **Maximization COP**: Find a solution a that maximizes $f(a)$.
- ▶ **Optimal solution**: Solution to a minimization (maximization) COP.

Decision problem associated to a COP:

Given an instance of a COP, (N, f) , and some threshold $t \in \mathbb{Q}$, is there a solution a of P such that $f(a) \geq t$ ($f(a) \leq t$, resp.)?

The decision problem of TSP

v_i : variable for city i with domain $D_i := \{1, \dots, n\} \setminus \{i\}$
(read as: value of v_i is the city to be visited next)

c_{ij} : distance between cities i and j (may not be symmetric)

t : bound for the total tour length

Then:

$\text{circuit}(v_1, \dots, v_n)$

$$\sum_{1 \leq i \leq n} c_{iv_i} \leq t$$

2 Filtering

- Arc consistency
- All-different Constraint

Filtering

- ▶ Constraint propagation techniques aim at **filtering** variable domains: remove useless values (that cannot participate in any solution) as early as possible.
- ▶ Filtering allows **false-positives** (values are kept though they are useless),
- ▶ but not false-negatives (useful values may not be removed).
- ▶ A constraint is “good” if it allows significant filtering (pruning of domain values) with low computational efforts.
- ▶ Constraint solver may benefit from exploiting the structure of such good constraints.

Filtering by enforcing arc consistency

- ▶ In general, enforcing generalized arc consistency on a constraint network requires exponential time w.r.t. the largest arity of some constraint relation in the network.

Recall: Enforcing generalized arc consistency runs in time

$$\mathcal{O}(erd^r),$$

where e is the number of constraints and r is the largest arity of some constraint in the network,

- ▶ Though general constraints have often high arity, there exist efficient methods to enforce generalized arc consistency.
- ▶ In the following we consider the all-different constraints.

Value graphs

Definition

An undirected graph $G = \langle V, E \rangle$ is **bipartite** if there exists a partition $S \dot{\cup} T$ of V such that for each $\{x, y\} \in E$, $x \in S$ iff $y \in T$.

A directed graph $G = \langle V, A \rangle$ is **bipartite** if there exists a partition $S \dot{\cup} T$ of V such that $A \subseteq (S \times T) \cup (T \times S)$.

G is then written in the form $G = \langle S, T, E \rangle$ (resp. $G = \langle S, T, A \rangle$).

Definition

Let V be a set of variables and D be the union of all domains D_v for $v \in V$.

The **value graph** of V is defined as the following bipartite graph:

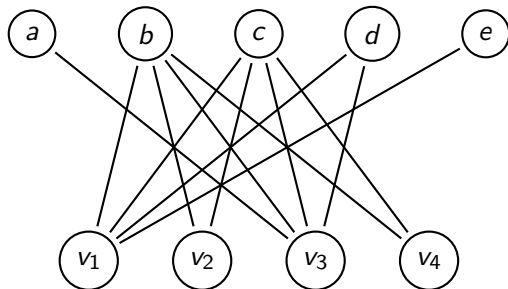
$$G = \langle V, D, E \rangle$$

where $E = \{\{v, d\} : v \in V, d \in D_v\}$.

Example: Value graph

Consider variables v_1, \dots, v_4 with $D_1 = \{b, c, d, e\}$, $D_2 = \{b, c\}$, $D_3 = \{a, b, c, d\}$, $D_4 = \{b, c\}$.

Value graph:



Matchings

Let $G = \langle V, E \rangle$ be an undirected (simple) graph.

Definition

A **matching** in G is a set $M \subseteq E$ of pairwise disjoint edges.

A matching M **covers** a set $S \subseteq V$ if $S \subseteq \bigcup M$, i.e., each $v \in S$ is contained in some edge in M .

$v \in V$ is **M -free** if M does not cover $\{v\}$.

Definition

Let M be a matching in G .

A path $P = v_0, e_1, \dots, e_k, v_k$ in G is **M -alternating** if all the edges e_i are alternatingly out of and in M .

An M -alternating path $P = v_0, e_1, \dots, e_k, v_k$ is called **M -augmenting** if v_0 and v_k are M -free.

Max-cardinality matching

Let $G = \langle V, E \rangle$ be a graph and M be a matching in G .

Theorem (Peterson)

M is a max-cardinality matching (i.e., it is a matching of maximum cardinality) if and only if there is no M -augmenting path in G .

Remark: If M is a matching and v_0, \dots, v_k is an M -augmenting path, then

$$M' := M \oplus \{\{v_i, v_{i+1}\} : 0 \leq i \leq k - 1\}$$

is a matching with $|M'| = |M| + 1$.

Hence a max-cardinality matching can be obtained by repeatedly searching for an M -augmenting path in $G \dots$

Max-cardinality matching on bipartite graphs

Let $G = \langle U, W, E \rangle$ be a bipartite graph and M be some matching in G . Define a directed bipartite graph $G_M = \langle U, W, A \rangle$ by

$$A := \{(w, u) : \{u, w\} \in M, u \in U, w \in W\} \cup \\ \{(u, w) : \{u, w\} \in E \setminus M, u \in U, w \in W\}$$

Each directed path in G_M is M -alternating.

If such a path starts and ends in an M -free vertex (starts in U , ends in W), it is an M -augmenting path in G .

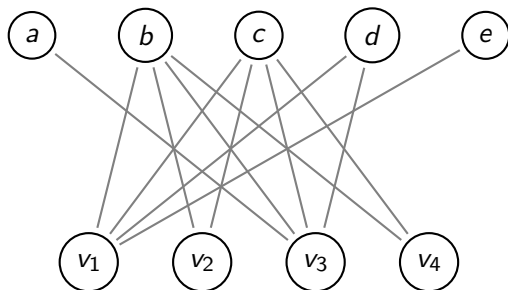
If no M -augmenting path can be found, M is a max-cardinality matching.

This can be used to compute a max-cardinality matching in time

$\mathcal{O}(|U| \cdot |A|)$ (van der Waerden and König)

... can be improved to $\mathcal{O}(\sqrt{|U|} \cdot |A|)$ (Hopcroft and Karp)

Example: Computing a max-cardinality matching



... and max-cardinality matching

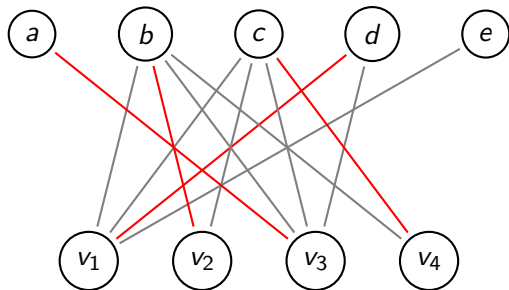
$$M = \{\{v_4, b\}, \{v_2, c\}, \{v_1, e\}, \{v_3, a\}\}$$

All-different constraint and matching

Let $V = \{v_1, \dots, v_n\}$ be a set of variables and G be the value graph of V .
 Let (d_1, \dots, d_n) be a variable assignment.

Lemma

$(d_1, \dots, d_n) \in \text{alldifferent}(v_1, \dots, v_n)$ if and only if
 $M = \{\{v_1, d_1\}, \dots, \{v_n, d_n\}\}$ is a matching in G .



Arc-consistent all-different constraint

Lemma

The constraint $alldifferent(v_1, \dots, v_n)$ is generalized arc-consistent if and only if every edge in G belongs to a matching in G that covers V .

Proof.

Simple (exercise!). □

Edges in max-cardinality matchings

Theorem

Let G be a graph and let M be a max-cardinality matching in G .

An edge e belongs to some max-cardinality matching in G if and only if one of the following conditions holds:

- ▶ $e \in M$.
- ▶ e is on an even-length M -alternating path starting at an M -free vertex;
- ▶ e is on an even-length M -alternating cycle.

Enforcing arc consistency on all-different constraints

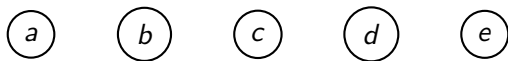
1. Compute a max-cardinality matching M in the value graph of V
(can be done in time $\mathcal{O}(m\sqrt{n})$ where $m = \sum_{1 \leq i \leq n} |D_i|$)
2. Identify the even M -alternating paths starting in an M -free vertex and the M -alternating cycles:
 - 2.1 Define dir. bipartite graph $G_M^* = \langle V, D_V, A \rangle$ with $A = \{(v, d) : v \in V, \{v, d\} \in M\} \cup \{(d, v) : v \in V, \{v, d\} \in E \setminus M\}$
 - 2.2 Compute the strongly connected components in G_M (in time $\mathcal{O}(n + m)$)
 - 2.3 Mark acrs between vertices in the same component as “used”: they belong to an even M -alternating cycle
 - 2.4 Mark arcs as “used” that belong to a M -alternating path in G_M that starts in an M -free vertex (breadth-first search in time $\mathcal{O}(m)$).
3. Update $D_v \leftarrow D_v \setminus \{d\}$ for all edges $\{v, d\}$ where the corresponding arc is not marked as “used”.

Example: Enforcing arc-consistency



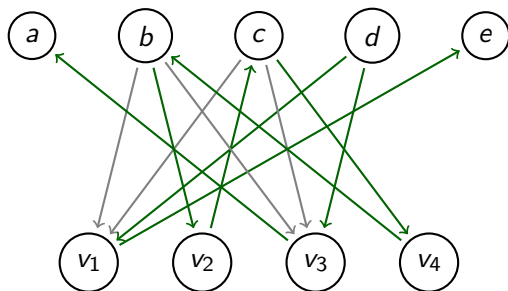
Start from max-cardinality matching

Example: Enforcing arc-consistency



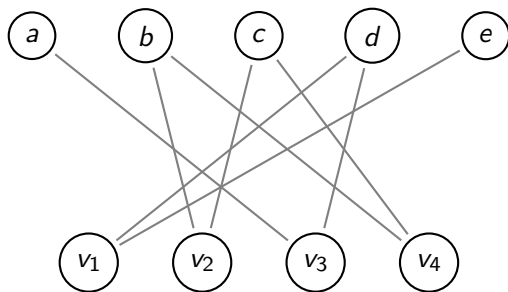
Compute strongly connected components
(e.g. by Kosaraju's algorithm)

Example: Enforcing arc-consistency



Mark "used" arcs

Example: Enforcing arc-consistency



... and remove unused arcs

Example: Enforcing arc-consistency



The all-different constraint is now arc-consistent

Literature



Willem-Jan van Hoeve and Irit Katriel.
Global Constraints,
Handbook of Constraint Programming, Elsevier, 2006