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

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

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

No-attack constraints:

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

Problem representation:
- Variables $v_i$ for each column $1, \ldots, n$;
- $v_i$ can take a "row value" $1, \ldots, n$. 

Sum constraint

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

The sum constraint is defined as:

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

Global cardinality constraint

Let $v_1, \ldots, v_n$: "assignment variables" with $D_{v_i} \subseteq \{d_1^*, \ldots, d_m^*\}$.
$c_1, \ldots, c_m$: "count variables" with sets of integers as domains.

Definition
The global cardinality constraint is defined as:

$$\text{gcc}(v_1, \ldots, v_n, c_1, \ldots, c_m) :=\{(d_1, \ldots, d_n, a_1, \ldots, a_m) \in \prod_{1 \leq i \leq n} D_{v_i} \times \prod_{1 \leq j \leq m} D_{c_j} :$$

$$\text{for each } j, \ d_j^* \text{ occurs in } (d_1, \ldots, d_n) \text{ exactly } a_j \text{ times}\}$$

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

Circuit constraint
Let \( s = (s_1, \ldots, s_n) \) be a permutation of \( \{1, \ldots, n\} \).
Define \( C_s \) as the smallest set that contains 1 and with each element \( i \) also \( s_i \).
\((s_1, \ldots, s_n)\) is called cyclic if \( C_s = \{1, \ldots, n\} \).

Definition
Let \( v_1, \ldots, v_n \) be variables with domains \( D_i = \{1, \ldots, n\} \) \((1 \leq i \leq n)\).
\[
circuit(v_1, \ldots, v_n) := \{(d_1, \ldots, d_n) \in D_1 \times \cdots \times D_n : (d_1, \ldots, d_n) \text{ is cyclic}\}
\]
Given an assignment \( a = (d_1, \ldots, d_n) \), define
\[
A := \{(v_i, v_{d_i}) : d_i \in D_i, 1 \leq i \leq n\}.
\]
Then, \( a \) satisfies \( \circuit(v_1, \ldots, v_n) \) if and only if \( (V, A) \) is a directed cycle (without proper sub-cycles).

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 \( f(a) \geq t \) \( (f(a) \leq t, \text{ resp.})? \)

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 ...
Filtering

Arc consistency
All-different Constraint

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 $O(erd')$,
    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 $S \cup T$ 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 $S \cup T$ 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 $v_1, \ldots, v_4$ with $D_1 = \{b, c, d, e\}$, $D_2 = \{b, c\}$, $D_3 = \{a, b, c, d\}$, $D_4 = \{b, c\}$.

Value graph:

```
+---+---+---+---+
|   |   |   |   |
| a | b | c | d |
+---+---+---+---+
|   |   |   |   |
|    | v_1 | v_2 | v_3 | v_4 |
```

Matchings

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

Definition

A matching in $G$ is a set $M \subseteq E$ of pairwisely 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 $P = v_0, e_1, \ldots, e_k, v_k$ in $G$ is $M$-alternating if all the edges $e_i$ are alternately out of and in $M$. An $M$-alternating path $P = v_0, e_1, \ldots, e_k, v_k$ is called $M$-augmenting if $v_0$ and $v_k$ are $M$-free.

Max-cardinality matching

Let $G = (V, E)$ 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 $v_0, \ldots, v_k$ 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 $G$.

Max-cardinality matching on bipartite graphs

Let $G = (U, W, E)$ be a bipartite graph and $M$ be some matching in $G$. Define a directed bipartite graph $G_M = (U, W, A)$ 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 $O(|U| \cdot |A|)$ (van der Waerden and König)...

...can be improved to $O(\sqrt{|U|} \cdot |A|)$ (Hopcroft and Karp)
Example: Computing a max-cardinality matching

\[ \begin{align*}
 a & \quad b & \quad c & \quad d & \quad e \\
 v_1 & \quad v_2 & \quad v_3 & \quad v_4 \\
 \end{align*} \]

... and max-cardinality matching
\[ M = \{\{v_4, b\}, \{v_2, c\}, \{v_1, e\}, \{v_3, a\}\} \]

All-different constraint and matching

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

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

Arc-consistent all-different constraint

Lemma
The constraint \( \text{alldifferent}(v_1, \ldots, v_n) \) 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.
Filtering All-different Constraint

1. Compute a max-cardinality matching $M$ in the value graph of $V$ (can be done in time $O(m\sqrt{n})$ 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 $G_M = (V, D_V, A)$ 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 $O(n + m)$)
   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 $O(m)$).
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

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

Mark "used" arcs
Example: Enforcing arc-consistency

\[ \begin{align*}
   a & \rightarrow b \\
   a & \rightarrow c \\
   a & \rightarrow d \\
   a & \rightarrow e \\
   b & \rightarrow v_1 \\
   b & \rightarrow v_2 \\
   b & \rightarrow v_3 \\
   b & \rightarrow v_4 \\
   c & \rightarrow v_1 \\
   c & \rightarrow v_2 \\
   c & \rightarrow v_3 \\
   c & \rightarrow v_4 \\
   d & \rightarrow v_1 \\
   d & \rightarrow v_2 \\
   d & \rightarrow v_3 \\
   d & \rightarrow v_4 \\
   e & \rightarrow v_1 \\
   e & \rightarrow v_2 \\
   e & \rightarrow v_3 \\
   e & \rightarrow v_4 \\
\end{align*} \]

... and remove unused arcs

The all-different constraint is now arc-consistent

Reference: