Global Constraints

Hemmo Nieminen

March 18, 2009

General

Basic notations and definitions from the previous lectures hold (CSP, COP, variables, domains of variables etc.).

General

Basic notations and definitions from the previous lectures hold (CSP, COP, variables, domains of variables etc.).

Let C be a constraint on the variables $x_1, ..., x_n$ with respective domains $D(x_1), ..., D(x_n)$. We say that C is generalized arc consistent (hyper-arc consistent) if for every $1 \leq i \leq k$ and $\forall_{v \in D(x_i)}$, there exists a tuple $(d_1, ..., d_k) \in C$ such that $d_i = v$. A CSP is arc consistent if each of its constraints is arc consistent.

Filtering is a type of propagation in which one tries to remove values from unassigned variables' domains without affecting the possible solutions of the CSP. Filtering is *complete* if no more values can be removed from variables' domains without affecting the possible solutions of the CSP. Note that if no values can be removed from the domains of the variables, the CSP is hyper-arc consistent.

Graphs

An (undirected) graph is a pair $G = (V, E)$, where V is finite set of vertices and $E \subseteq V \times V$ is a multiset of edges (an edge is $\{u, v\}$, $u \in V$, $v \in V$). G is *bipartite* if there exists a partition $S \cup T$ of V such that $E \subseteq S \times T$, denoted $G = (S, T, E)$.

Graphs

An (undirected) graph is a pair $G = (V, E)$, where V is finite set of vertices and $E \subseteq V \times V$ is a multiset of edges (an edge is $\{u, v\}$, $u \in V$, $v \in V$). G is *bipartite* if there exists a partition $S \cup T$ of V such that $E \subseteq S \times T$, denoted $G = (S, T, E)$.

A *walk* in a graph G is an alternating sequence of consecutive edges and vertices. A walk is a *path* if all of its vertices are distinct. A *circuit* is a path but it has the same vertex as its starting and ending vertex.

Graphs

An (undirected) graph is a pair $G = (V, E)$, where V is finite set of vertices and $E \subseteq V \times V$ is a multiset of edges (an edge is $\{u, v\}$, $u \in V$, $v \in V$). G is *bipartite* if there exists a partition $S \cup T$ of V such that $E \subseteq S \times T$, denoted $G = (S, T, E)$.

A *walk* in a graph G is an alternating sequence of consecutive edges and vertices. A walk is a *path* if all of its vertices are distinct. A *circuit* is a path but it has the same vertex as its starting and ending vertex.

A *directed* walk/graph/path is a walk/graph/path in which its edges go only in one direction, thus $\{u, v\} \neq \{v, u\}$.

Matching Theory

A matching in a graph $G = (V, E)$ is a set $M \subseteq E$ of disjoint edges. A matching is said to cover a vertex v, if v belongs to some edge in M. A set of vertices is covered by M, if its every vertex is covered by M. A vertex is said to be *M-free* if it's not covered by M. M's cardinality is the number of edges in it ($|M|$).

Matching Theory

A matching in a graph $G = (V, E)$ is a set $M \subseteq E$ of disjoint edges. A matching is said to cover a vertex v, if v belongs to some edge in M. A set of vertices is covered by M, if its every vertex is covered by M. A vertex is said to be *M-free* if it's not covered by M. M's cardinality is the number of edges in it ($|M|$).

Let M be a matching in G. A path P in G is called *M-augmenting*, if P has odd length, its ends are M-free and its edges are alternating in and out of M.

Matching Theory

A matching in a graph $G = (V, E)$ is a set $M \subseteq E$ of disjoint edges. A matching is said to cover a vertex v, if v belongs to some edge in M. A set of vertices is covered by M, if its every vertex is covered by M. A vertex is said to be *M-free* if it's not covered by M. M's cardinality is the number of edges in it ($|M|$).

Let M be a matching in G. A path P in G is called *M-augmenting*, if P has odd length, its ends are M-free and its edges are alternating in and out of M.

A value graph of a set of variables X is a bipartite graph $G = (X, D(X), E)$, where X is a set of variables and $E = \{(x, d) \mid x \in X, d \in D(x)\}$.

Theorem

Let $G = (V, E)$ be a graph, and let M be a matching in G. Then M is a maximum-cardinality matching if and only if there does not exist an M-augmenting path in G.

Theorem

Let $G = (V, E)$ be a graph, and let M be a matching in G. Then M is a maximum-cardinality matching if and only if there does not exist an M-augmenting path in G.

Hence, maximum-cardinality matching can be found by repeatedly finding M-augmenting paths. On a bipartite graph G M-augmenting paths can be found easily from a directed bipartite graph $G_M = (U, W, A)$, created by orienting all edges in M from W to U and all other edges from U to W.

Global Constraints

A *global constraint* is a constraint that captures a relation between a non-fixed number of variables. Typically global constraints can be expressed as a conjunction of several simpler constraints. Global constraints however simplify the programming task and definitions.

In the following we'll go through some well known global constraints. There are numerous others defined, but these are meant to give you an idea of the types of global constraints one can create and use.

sum

Let $c_i \in \mathbb{Q}$ for $1 \leq i \leq n$ and $D(z) \subseteq \mathbb{Q}$.

$sum(x_1, ..., x_n, z, c) = \{(d_1, ..., d_n, d) \mid \forall i \ d_i \in D(x_i), d \in D(z), d = \Sigma_{i=1}^{n} c_i d_i\}$

We can also write $z = \Sigma_{i=1}^{n} c_i x_i$.

knapsack

A variant of the sum constraint. Let domain of z now be between upper bound u and lower bound l, $D(z) = [l, u]$.

$$knapsack(x_1, ..., x_n, z, c) =$$
$$\{(d_1, ..., d_n, d) \mid \forall i \ di_i \in D(z_i), d \in D(z), d \leq \Sigma_{i=1}^{n} c_i d_i\} \cap$$
$$\{(d_1, ..., d_n, d) \mid \forall i \ di_i \in D(z_i), d \in D(z), d \geq \Sigma_{i=1}^{n} c_i d_i\}$$

We can also write $l \leq \Sigma_{i=1}^{n} c_i d_i \leq u$.

element

Let y be an integer variable, z a variable with finite domain, and c an array of variables, i.e., $c = [x_1, ..., x_n]$. The element constraint states that z is equal to the y-th variable in c, or $z = x_y$.

$$element(y, z, x_1, ..., x_n) = \{(e, f, d_1, ..., d_n) \mid e \in D(y), f \in D(z), \forall i \ d_i \in D(x_i), f = d_e\}$$

$$(z = x_1 \wedge y = 1) \vee (z = x_2 \wedge y = 2) \vee ...$$

alldifferent

The AllDifferent constraint states that within the set $\{x_1, ..., x_n\}$ whenever $i \neq j, x_i \neq x_j$.

$$alldifferent(x_1, ..., x_n) = \{(d_1, ..., d_n) \mid \forall_i d_i \in D(x_i), \forall_{j \neq \text{I}} d_i \neq d_j\}.$$

$$(x_1 \neq x_2) \wedge (x_1 \neq x_3) \wedge (x_2 \neq x_3) \wedge ...$$

gcc

The *global cardinality constraint* is a generalization of alldifferent. Let there be a set of variables $X = \{x_1, ..., x_n\}$ and let $\{v_1, ..., v_m\} = D(X)$. A variable c_{v_i} is equal to the number of variables in X that will be assigned the value v_i.

$$gcc(x_1, ..., x_n, c_{v_1}, ..., c_{v_m}) = \{(w_1, ..., w_n, o_1, ..., o_m) \mid \forall j\ w_j \in D(x_j), \forall i\ occ(v_i, (w_1, ..., w_n)) = o_i \in D(c_{v_i})\}$$

$occ(m, X) =$ the number of occurences of m in the set X.

cost_gcc

The *global cardinality constraint with costs* combines a *gcc* and a variant of the *sum* constraint. In addition to variables used with *gcc* we now have a function w that associates each pair $(x, d) \in X \times D(X)$ a "cost" $w(x, d) \in \mathbb{Q}$ and a variable z with domain $D(z)$. Assuming we want to *minimize* z:

$$cost\_gcc(x_1, ..., x_n, c_{v_1}, ..., c_{v_m}, z, w) = \{(d_1, ..., d_n, o_1, ...o_m, d) \mid (d_1, ..., d_n, o_1, ..., o_m) \in gcc(x_1, ..., x_n, v_{v_1}, ..., c_{v_n}), \forall i \ d_i \in D(x_i), d \in D(z), \Sigma_{i=1}^{n} w(x_i, d_i) \leq d\}.$$

cumulative

We are given a collection $T = t_1, ..., t_n$ of task, such that each task t_i is associated with four variables: its *release time* r_i, its *deadline* d_i, its *processing time* p_i and its *capacity requirement* c_i. In addition, we are given the capacity C of some resource. A solution is a schedule, i.e., a starting time s_i for each task t_i such that $r_i \leq s_i \leq d_i - p_i$, and in addition $\forall u \Sigma_{i|s_i \leq u \leq s_i + p_i} c_i \leq C$.

$cumulative(\{r_1, ..., r_n\}, \{d_1, ..., d_n\}, \{p_1, ..., p_n\}, \{c_1, ..., c_n\}, C)$

regular

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA, L(M) is the language M accepts and let $X = \{x_1, x_2, ..., x_n\}$ be a set of variables with $D(x_i) \subseteq \Sigma$ for $1 \leq i \leq n$.

$$regular(X, M) = \{(d_1, ..., d_n) \mid \forall i \ d_i \in D(x_i), d_1 d_2 ... d_n \in L(M)\}$$

circuit

Let $X = \{x_1, x_2, ..., x_n\}$ be a set of variables with respective domains $D(x_i) \subseteq \{1, 2, ..., n\}$ for $i = 1, 2, ..., n$. Then

$circuit(x_1, ..., x_n) = \{(d_1, ..., d_n) \mid \forall i \ di_i \in D(x_i), d_1, ..., d_n$ is cyclic$\}$.

soft_alldifferent

The soft AllDifferent constraint, as opposed to a traditional hard constraint may be violated. Instead, we measure the amount of violation and try to minimize it. The *violation measure* of a soft constraint $C(x_1, ..., x_n)$ is a function $\mu : D(x_1) \times ... \times D(x_n) \rightarrow \mathbb{Q}$, which is represented by a variable z. For the soft AllDifferent constraint we consider two violation measures; μ_{var} (variable based) and μ_{dec} (decomposition based). μ_{var} counts the minimum number of variables that need to change in order to satisfy the constraint and μ_{dec} counts the number of constraints in its binary decomposition that are violated.

soft_alldifferent

For $alldifferent(x_1, ..., x_n)$:

$$\mu_{dec}(x_1, ..., x_n) = \mid \{(i,j) \mid \forall_{i<j} x_i = x_j\} \mid.$$

$$soft\_alldifferent(x_1, ..., x_n, z, \mu) =$$
$$\{(d_1, ..., d_n, d) \mid \forall i \ d_i \in D(x_i), d \in D(z), \mu(d_1, ...d_n) \leq d\}.$$

Filtering Algorithms

A *filtering algorithm* for a constraint C is an algorithm that filters the domains of variables with respect to C. If the algorithm achieves complete filtering the algorithm itself is said to be complete (wrt. partial).

In general, establishing arc consistency for non-binary constraint (or global constraint) is NP-hard, however, for a number of global constraints it is possible to establish arc consistency quite efficiently. In the following we'll go through a complete filtering algorithm for the AllDifferent global constraint.

Backgrounds

Theorem

Let $X = \{x_1, ..., x_n\}$ be a set of variables and let G be the value graph of X. Then $(d_1, ..., d_n) \in alldifferent(x_1, ..., x_n)$ if and only if $M = \{\{x_1, d_1\}, ..., \{x_n, d_n\}\}$ is a matching in G.

Backgrounds

Theorem

Let $X = \{x_1, ..., x_n\}$ be a set of variables and let G be the value graph of X. Then $(d_1, ..., d_n) \in alldifferent(x_1, ..., x_n)$ if and only if $M = \{\{x_1, d_1\}, ..., \{x_n, d_n\}\}$ is a matching in G.

Corollary

Let G be the value graph of a set of variables $X = \{x_1, ..., x_n\}$. The constraint $alldifferent(x_1, ..., x_n)$ is arc consistent if and only if every edge in G belongs to a matching in G covering X.

Theorem

Let G be a graph and M a maximum-cardinality matching in G. An edge e belongs to some maximum-cardinality matching in G if and only if $e \in M$, or e is on an even length M-alternating path starting at an M-free vertex, or e is on an even-length M-alternating circuit.

Proof.

Let M be a maximum-cardinality matching in G. Suppose edge e belongs to a maximum-cardinality matching N, and $e \notin M$. The graph $G' = (V, M \otimes N)$ consists of even-length paths and circuits with edges alternatingly in M and N. If the paths are not of even length, either M or N can be made larger by interchanging edges in M and N along this path (a contradiction because they are of maximum-cardinality).

Proof.

Let M be a maximum-cardinality matching in G. Suppose edge e belongs to a maximum-cardinality matching N, and $e \notin M$. The graph $G' = (V, M \otimes N)$ consists of even-length paths and circuits with edges alternatingly in M and N. If the paths are not of even length, either M or N can be made larger by interchanging edges in M and N along this path (a contradiction because they are of maximum-cardinality).

Conversely, let M be a maximum-cardinality matching in G and let P be and even-length M-alternating path starting at an M-free vertex or an M-alternating circuit. Let e be and edge such that $e \in P \setminus M$. Then $M \otimes P$ is a maximum-cardinality matching that contains e. □

The Algorithm

First we compute a maximum-cardinality matching M in the value graph $G = (X, D(X), E)$. Next we identify the even M-alternating paths starting at an M-free vertex, and the even M-alternating circuits in G and mark the vertices belonging to those paths and circuits as "used". For all edges $\{x, d\}, x \in X, d \in D(X)$, if the edge is not in M and it is not marked as "used" we update $D(x) = D(x) \setminus \{d\}$. This leads to a hyper-arc consistent graph satisfying the alldiff constraint.

To find the desired paths and circuits from G, we define the directed bipartite graph $G_M = (X, D(X), A)$ with arc set $A = \{(x, d) \mid x \in X, d \in D(X), \{x, d\} \in M\} \cup \{(d, x) \mid x \in X, d \in D(X), \{x, d\} \in E \setminus M\}$.

By searching the strongly connected components from G_M we find the even-length circuits in G. Vertices part of a M-alternating path starting from a M-free vertex in G_M are also part of a M-alternating path starting from M-free vertex in G. There are sufficient algorithms for finding these and the overall filtering algorithm can be finished in polynomial time.

Summary

- There are lots of different global constraints and they are great for expressing more complicated relations between larger sets of variables.
- By filtering, we reduce the amount of needed computation for solving CSPs by removin useless values from the CSP's constraints' variables' domains.
- Filtering algorithm tightens the domains of a constraint's variables.
- Complete filtering algorithm produces hyper-arc consistent constraints.
- Few complete filtering algorithms can be run in polynomial time for non-binary (global) constraints.