Talk 8: Global Constraints

pp. 185-200, sections 6.3.2-6.5.3

Leo Moisio

T-79.4001 Seminar on Theoretical Computer Science, 2009

Outline

Outline

Why Global Constraints

- modelling benefits
  - human readable
  - compact encoding
- performance benefits
  - improved constraint propagation effectivity and efficiency
- implementation benefits
  - modular, black box

Outline

Global Constraints and Arc Consistency

Definition (Global Constraint)

- Global Constraint $C(X) \subseteq \times_{x \in X} D(x)$
- |X| is not fixed
- $(d_1, \ldots, d_n) \in C$ a *solution* to C
- assignment of value d_i to variable $x_i \in X \forall i \in [1, |X|]$ satisfies C
- $C = \emptyset \Rightarrow C$ is *inconsistent*.

Definition (Generalized Arc Consistency)

$C(X)$ is *arc consistent* iff: $\forall x_i \in X \forall v \in D(x) \exists t \in C : t[i] = v$.

Outline

T-79.1001 Recap: DFA

Definition

Deterministic Finite Automaton $M = (Q, \Sigma, \delta, q_0, F)$

- finite set of *states* Q
- finite *alphabet* Σ
- *transition function* $\delta : Q \times \Sigma \rightarrow Q$
- *initial state* $q_0 \in Q$
- set of *final states* $F \subseteq Q$

Definition

- $(q, w) \vdash_M (q', w') \Leftrightarrow w = \sigma w', \delta_M(q, \sigma) = q'$
- $\vdash_M^\star$ reflexive transitive closure of $\vdash_M$
- $L_M = \{w \mid (q_0, w) \vdash_M^\star (q_f, \varepsilon), q_f \in F\}$

The Constraint

Definition

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

- DFA $M = (Q, \Sigma, \delta, q_0, F)$
- $X = \{x_1, \ldots, x_n\}, \quad \forall i\ D(x_i) \subseteq \Sigma$
- language L_M accepted by M

Example

$x_1 \in \{a, b, c\}, x_2 \in \{a, b, c\}, x_3 \in \{a, b, c\}, x_4 \in \{a, b, c\}$,
regular(x_1, x_2, x_3, x_4, M)
(M from previous example, whiteboard)

Representing the Constraint as a Digraph

Definition

Digraph $\mathcal{R} = (V, A)$

- vertex set $V = \bigcup_{i=1}^{n+1} V_i, \quad \forall i \; V_i = \{q_k^i \mid q_k \in Q\}$
- arc set $A = \bigcup_{i=1}^{n} A_i$
  $\forall i \; A_i = \{(q_k^i, q_l^{i+1}) \mid \delta(q_k, d) = q_l, d \in D(x_i)\ \}$

Theorem

A solution to regular(X,M) corresponds to a directed path in $\mathcal{R}$ from $q_0^1 \in V_1$ to $q_f^{n+1} \in V_{n+1}, q_f \in F$.

Proof.

$\mathcal{R}$ represents $M \cap M_n$, where $L_{M_n} = \{s \mid |s| = n\ \}$. □

Arc Consistency for the Constraint

Corollary

regular(X,M) is arc consistent iff $\forall x_i \in X, d \in D(x_i) \exists$ *arc* $a = (q_k^i, q_l^{i+1})$ *st.* $\delta(q_k, d) = q_l$ *and a belongs to a directed path from* q_0^1 *to a final state in* V_{n+1}*.*

Example

Digraph $\mathcal{R}$ for regular(X,M) from previous example, identify paths representing solutions (whiteboard)

Algorithm for Arc Set A of Digraph $\mathcal{R}$

Input: $n = |X|, M = (Q, \Sigma, \delta, q_0, F)$
Output: A
$A \leftarrow \emptyset$
foreach $i \in \{1, \dots, n\}$ **do**
  foreach $q_j \in Q$ **do**
    if $\delta^{in}(q_j^i) \neq \emptyset \vee q_j^i = q_0^1$ **then**
      foreach $d \in D(x_i)$ **do**
        if $\delta(q_j, d) = q_k$ **then**
          $A \leftarrow A \cup \{(q_j^i, q_k^{i+1})\}$
foreach $i \in \{n+1, \dots, 2\}$ **do**
  foreach $q_k \in Q$ **do**
    if $(i = n + 1 \wedge q_k \in Q \setminus F) \vee (i \leq n \wedge \delta^{out}(q_k^i) = \emptyset)$ **then**
      $A \leftarrow A \setminus \delta^{in}(q_k^i)$
return A

Algorithm for Complete Filtering

Input: $A, X, M = (Q, \Sigma, \delta, q_0, F)$
Result: modifies $D(x_i) \forall i$
foreach $x_i \in X$ **do**
    foreach $d \in D(x_i)$ **do**
        if $\nexists q_j, q_k \in Q$ *st.* $(\delta(q_j, d) = q_k \wedge (q_j^i, q_k^{i+1}) \in A)$ **then**
            $D(x_i) \leftarrow D(x_i) \setminus \{d\}$

Analysis

- achieves arc consistency
- $\mathcal{O}(|X||Q||\Sigma|)$ time, space
- incremental updating easy

Outline

The Constraint

Definition

$\mathrm{gcc}(X, C, B)=\{(d_1, \ldots, d_n, o_1, \ldots, o_m) \mid \forall i \ d_i \in D(x_i),$
$\forall j \ occ(b_j, (d_1, \ldots, d_n)) = o_j \in D(c_{b_j})\}$

- $X = \{x_1, \ldots, x_n\}, \quad \forall i \ D(x_i) \subseteq \{b_1, \ldots, b_m\} = B$
- $C = \{c_{b_1}, \ldots, c_{b_m}\}, \quad \forall j \ D(c_{b_j}) \subset \mathbb{N}$
- $occ(b, t)$ = # of occurrences of value b in tuple t.

Theorem

Achieving arc consistency in gcc is NP-hard.

Proof.

CNF SAT $\leq_L$, omitted. $\square$

Modified GCC

Definition

$\text{gcc}(X, R, B)=\{(d_1, \ldots, d_n, o_{b_1}, \ldots, o_{b_m}) \mid \forall i\ d_i \in D(x_i),$
$\forall b \in B\ occ(b, (d_1, \ldots, d_n)) = o_b \in r_b\}$

- $X = \{x_1, \ldots, x_n\}, \quad \forall i\ D(x_i) \subseteq \{b_1, \ldots, b_m\} = B$
- $R = \{r_{b_1}, \ldots, r_{b_m}\}, \quad \forall b \in B:\ r_b = [l_b, u_b] \subset \mathbb{N}$ constants
- $occ(b, t)$ = # of occurrences of value b in tuple t.

Flow Theory Primer: Basic Definitions

- digraph $G = (V, A), \quad s, t \in V$
- function $f : A \to \mathbb{R}$ is a *s-t flow*, iff
  - $f(a) \geq 0, \quad \forall a \in A$
  - $f(\delta^{out}(v)) = f(\delta^{in}(v)), \quad \forall v \in V \setminus \{s, t\}$
- $f(S) = \sum_{a \in S} f(a), \quad \forall S \subseteq A$
- $value(f) = f(\delta^{out}(s)) - f(\delta^{in}(s))$
- $c(a) \geq d(a) \geq 0, \quad \forall a \in A$
- feasible flow: $d(a) \leq f(a) \leq c(a), \quad \forall a \in A$

Flow Theory Primer: Residual Graph

- residual graph $G_f = (V, A_f)$
  $\forall (u, v) \in A$:
  - $f(u,v) < c(u,v) \Rightarrow (u,v) \in A_f$,
    $d_f(u,v) = \max\{d(u,v) - f(u,v), 0\}$,
    $c_f(u,v) = c(u,v) - f(u,v)$
  - $f(u,v) > d(u,v) \Rightarrow (v,u) \in A_f$,
    $d_f(v,u) = 0$,
    $c_f(v,u) = f(v,u) - d(v,u)$

Flow Theory Primer: Weighted Flow Network

- function $w : A \to \mathbb{R}$
- $w(P) = \sum_{a \in P} w(a)$
- $weight(f) = \sum_{a \in A} w(a) f(a)$
- min weight flow f: f is feasible and $weight(f) \leq weight(f')$ for all feasible flows f'

F. T. Primer: Finding Min Weight Feasible Flow

Input: $G = (V, A), d, c, w$
Output: f
$f \leftarrow \vec{0}$
$A \leftarrow A \cup \{(t, s)\}$
$[d, c, w, f](t, s) \leftarrow [0, \infty, 0, 0]$
while $\exists(u, v) \in A : f(u, v) < d(u, v)$ **do**
  find directed v-u path P in G_f minimizing $w(P)$
  if $\nexists P$ **then**
    fail: no feasible flow
  else
    directed circuit $C \leftarrow P, u, v$
    $\varepsilon \leftarrow \max\{\varepsilon \mid \vec{0} \leq f + \varepsilon\chi^P \leq \vec{c}, f(u, v) + \varepsilon \leq d(u, v)\}$
    $f \leftarrow f + \varepsilon\chi^C$
return f

Analysis

- pseudo-polynomial runtime
- faster algorithms exist
- chosen for simplicity

Rerouting Flow Through an Arc in a Min Weight Flow

Theorem

Let f be a min weight s-t flow of value ϕ in $G = (V, A)$ with $f(a) = 0$ for some $a \in A$. Let C be a directed circuit in G_f with $a \in C$ minimizing $w(C)$. Then $f' = f + \varepsilon\chi^C$,where ε is subject to $d \leq f + \varepsilon\chi^C \leq c$, has min weight among all s-t flows g in G with value$(g) = \phi$ and $g(a) = \varepsilon$. $\nexists C \Rightarrow \nexists f'$. Otherwise, weight$(f') =$weight$(f) + \varepsilon \cdot w(C)$.

proof idea.

For a min weight flow f in G, G_f does not contain directed circuits with negative weight. $\square$

Representing the Constraint as a Flow Network

Definition (Augmented Value Graph $G = (V, A)$)

- $V = X \cup B \cup \{s, t\}$
- $A = A_s \cup A_X \cup A_t$
- $A_s = \{(s, x) \mid x \in X\}, [d, c](s, x) = [1, 1]$
- $A_X = \{(x, b) \mid x \in X, b \in D(x)\}, [d, c](x, b) = [0, 1]$
- $A_t = \{(b, t) \mid b \in B\}, [d, c](b, t) = [l_b, u_b]$

Theorem

Let C=gcc(X, R, B) and G as above. Then there is 1-1 correspondence between solutions of C and integral feasible s-t flows in G.

Conditions for Arc Consistency of GCC

Corollary

Let C=$gcc(X, R, B)$ and G the augmented value graph representing C. Then C is arc consistent iff every variable-value arc $(x, b) \in A$ belongs to some feasible integral flow in G.

Theorem

Let G be a graph and f a feasible flow in G. An arc belongs to some feasible flow in G iff it belongs to f or both of its endpoints belong to the same strongly connected component of the residual graph of G with respect to f.

Algorithm Sketch for Complete Filtering

- Find a feasible integral flow f in G.
- Identify variable-value arcs in f.
- Compute the residual graph G_f of G with respect to f.
- Identify strongly connected components in G_f.
- Identify variable-value arcs in SCCs.
- most of above covered in T-106.4100 Design and Analysis of Algorithms

Analysis

- in P
- incremental updating easy
- achieves arc consistency

Outline

Definition (Constraint Optimization Problem)

- A COP on X is a CSP P on X with an objective function $f : \times_{x \in X} D(x) \to \mathbb{Q}$.
- An *optimal solution* to a min(max) CSP problem is a solution d to P that minimizes(maximizes) $f(d)$.

How it used to be done:

- include variable z and constraint $f(d) = z$, minimize z
- when solution d to P is found, calculate $opt \leftarrow f(d)$, add constraint $z < opt$.
- deficiency: no inference from $D(z)$ to $D(x)$

This deficiency is addressed by optimization constraints by embedding z.

Outline

The Constraint

Definition

cost_gcc(X, R, B, z, w)$= \{(d_1, \ldots, d_n, o_{b_1}, \ldots, o_{b_m}, d) \mid$
$(d_1, \ldots, d_n, o_{b_1}, \ldots, o_{b_m}) \in \mathrm{gcc}(X, R, B)$,
$d \in D(z), \sum_{i=1}^{n} w(x_i, d_i) \leq d\}$

- basic gcc as earlier (constant intervals version)
- function $w : X \times B \rightarrow \mathbb{Q}$

The Graph Representation

Definition (Augmented Weighted Value Graph $CG = (V, A)$)

- $V = X \cup B \cup \{s, t\}$
- $A = A_s \cup A_X \cup A_t$
- $A_s = \{(s, x) \mid x \in X\}, [d, c](s, x) = [1, 1]$
- $A_X = \{(x, b) \mid x \in X, b \in D(x)\}, [d, c](x, b) = [0, 1]$
- $A_t = \{(b, t) \mid b \in B\}, [d, c](b, t) = [l_b, u_b]$
- $w(s, x) = 0, w(b, t) = 0, w(x, b) = w(x, b)$

Arc Consistency for the Constraint

Theorem

cost_gcc(X, R, B, z, w) *is arc consistent iff*

- $\forall x \in X \forall d \in D(x) \exists$ *an integral feasible s-t flow f in CG with* $f(x,d) = 1, weight(f) \leq \max D(z)$ *and*
- $weight(f) \leq \min D(z)$ *for some integral feasible s-t flow f in CG*

Algorithm Sketch for Complete Filtering

- build CG
- compute feasible min weight flow f in CG
- $\min D(z) \leftarrow \max(\min D(z), weight(f))$
- for each arc $a = (u, v) \in A_x, f(a) = 0$ representing (x, d), reroute flow through this graph using a min weight directed cycle, resulting in flow f'
- if $weight(f') > \max D(x)$, remove d from $D(x)$

Analysis

- achieves arc consistency
- can be done in polynomial time
- can be made incremental

Outline

The Constraint

Definition

soft_alldifferent(X, z, μ)= $\{(d_1, \ldots, d_n, d) \mid \forall i d_i \in D(x_i), d \in D(z), \mu(d_1, \ldots, d_n) \leq d\}$

- $X = \{x_1, \ldots, x_n\}$
- $\mu_{dec}(x_1, \ldots, x_n) = |\{(i, j) \mid \forall i < j : x_i = x_j\}|$
- μ_{var} minimum number of variables that need to change value to satisfy the constraint

Algorithm for u_{dec} is studied, u_{var} is not studied further.

Representation as a Digraph

Definition

digraph $S = (V, A)$

- $V = \{s, t\} \cup X \cup D(X)$
- $A = A_s \cup A_X \cup A_t$
- $A_s = \{(s, x) \mid x \in X\}$
- $A_X = \{(x, d) \mid x \in X, d \in D(x)\}$
- $A_t = \{(d, t) \mid x \in X, d \in D(x)\} \quad (d, t)_0, (d, t)_1, \ldots$
- $[d, c](a) = [1, 1], a \in A_s$
- $[d, c](a) = [0, 1], a \in A \setminus A_s$
- $w(a) = 0, a \in A \setminus A_t$
- $w(a) = i, a = (d, t)_i \in A_t$

Arc Consistency for the Constraint

Theorem

soft_alldifferent(X, z, μ_{dec}) *is arc consistent iff*

- $\forall a \in A_X$: *a belongs to some feasible integral flow f in S with weight*$(f) \leq \max D(z)$
- $\min D(z) \geq$ *weight*(f) *for a min weight s-t flow f in S*

Algorithm for this uses the same ideas as the previous ones.

Outline

Definitions of Bound Consistencies

Definition

- Constraint $C(X)$ is *lower bound consistent* iff:
  $\forall x \in X : \exists t \in C : t[i] = \min D(x)$
- *upper bound consistency* similarly
- Constraint is *bound consistent*, if it is both lower and upper bound consistent

Outline

Hardness of Achieving Arc Consistency

Theorem

Let C be a constraint. If there is a polynomial-time algorithm that computes arc consistency for C, then there is a polynomial-time algorithm that finds a single solution to C.

Theorem

Let $C(X)$ be a constraint. If there is an algorith A that, for any $x \in X, d \in D(x)$, determines whether there is a solution to the constraint $C \wedge (x \leftarrow d)$, then there is a polynomial-time algorityhm that computer arc consistency for C.

Summary

- Global constraints improve modelling and performance.
- Efficient complete filtering algorithms exist for many global constraints.
- Partial filtering helps when complete filtering is too expensive.

For Further Reading I

- N. Beldiceanu et al.
  Global Constraint Catalog.
  http://www.emn.fr/x-info/sdemasse/gccat/