DM826 – Spring 2011

Modeling and Solving Constrained Optimization Problems

Lecture 6
Filtering Algorithms

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

[*Based on slides by Stefano Gualandi, Politecnico di Milano*]

Declarative and Operational Semantic

- Declarative Semantic: specify **what** the constraint means. Evaluation criteria is expressivity.
- Operational Semantic: specify **how** the constraint is computed, i.e., is kept *consistent* with its declarative semantic. Evaluation criteria are efficiency and effectiveness.

Example

So far, we have defined only the Declarative Semantic of the `alldifferent` constraint, not its Operational Semantic.

Domain Consistency

Definition

A constraint C on the variables $x_1, \ldots, x_m$ with respective domains $D_1, \ldots, D_m$ is called domain consistent (or hyper-arc consistent) if for each variable x_i and each value $d_i \in D_i$ there exists compatible values in the domains of all the other variables of C, that is, there exists a tuple $(d_1, \ldots, d_i, \ldots, d_k) \in C$.

Consistency and Filtering Algorithms

- Different level of consistency (arc, bound, range, domain) are maintained by different filtering algorithms, which must be able to:
  1. Check consistency of C w.r.t. the current variable domains
  2. Remove inconsistent values from the variable domains

- The stronger is the level of consistency, the higher is the complexity of the filtering algorithm.

... again the `alldifferent` case

There exists in literature several filtering algorithms for the `alldifferent` constraints.

Domain consistency for alldifferent

1. build value graph $G = (X, D(X), E)$
2. compute maximum matching M in G
3. **if** $|M| < |X|$ **then return false**
4. mark all arcs in G_M that are not in M as unused
5. compute SCCs in G_M and mark all arcs in a SCC as used
6. perform breadth-first in G_M search starting from M-free vertices, and mark all traversed arcs as used if they belong to an even path
7. **for all** arcs (x_i, d) in G_M marked as unused **do**
   $D(x_i) := D(x_i) \setminus d$
   if$D(x_i) = \emptyset$ **then return false**
8. **return true**

Overall complexity: $O(n\sqrt{m} + (n + m) + m)$
It can be updated incrementally if other constraints remove some values.

Relaxed Consistency

Definition

A constraint C on the variables $x_1, \ldots, x_m$ with respective domains $D_1, \ldots, D_m$ is called bound(Z) consistent if for each variable x_i and each value $d_i \in \{\min(D_i), \max(D_i)\}$ there exists compatible values between the min and max domain of all the other variables of C, that is, there exists a value $d_j \in [\min(D_i), \max(D_i)]$ for all $j \neq i$ such that $(d_1, \ldots, d_i, \ldots, d_k) \in C$.

Definition

A constraint C on the variables $x_1, \ldots, x_m$ with respective domains $D_1, \ldots, D_m$ is called range consistent if for each variable x_i and each value $d_i \in D_i$ there exists compatible values between the min and max domain of all the other variables of C, that is, there exists a value $d_j \in [\min(D_i), \max(D_i)]$ for all $j \neq i$ such that $(d_1, \ldots, d_i, \ldots, d_k) \in C$.

Bound Consistency [Mehlorn&Thiel2000]

Definition (Convex Graph)

A bipartite graph $G = (X, Y, E)$ is convex if the vertices of Y can be assigned distinct integers from $[1, |Y|]$ such that for every vertex $x \in X$, the numbers assigned to its neighbors form a subinterval of $[1, |Y|]$.

In convex graph we can find a matching in linear time.

Survey of complexity: effectiveness and efficiency

Consistency	Idea	Complexity	Amort.	Reference(s)
arc		$O(n^2)$		[VanHentenryck1989]
bound	Hall	$O(n \log n)$		[Puget1998]
	Flows			[Mehlhorn&Thiel2000]
	Hall			[Lopez&All2003]
		$O(n)$		[Mehlhorn&Thiel2000]
				[Lopez&All2003]
range	Hall	$O(n^2)$		[Leconte1996]
domain	Flows	$O(n\sqrt{m})$	$O(n\sqrt{k})$	[Régin1994],[Costa1994]

Where $n =$ number of variables, $m = \sum_{i \in 1 \ldots n} |D_i|$, and
$k =$ number of values removed.

References

van Hoeve W. and Katriel I. (2006). **Global constraints**. In *Handbook of Constraint Programming*, chap. 6. Elsevier.