FoPra: Implementation and Evaluation of Advanced Propagation Algorithms for Global Constraints

Patrick Pekczynski

Supervisors: Dipl.-Inf. Guido Tack, MSc Marco Kuhlmann

Programming Systems Lab
Department of Computer Science
Saarland University, Saarbrücken

28.10.2004

Motivation

Example (Latin Square)

- 4×4-matrix of 16 variables x_i ranging over $D_i = [1..4]$
- each possible value occurs exactly once in each row and each column

Variables

$$
\begin{matrix}
x_1 & x_2 & x_3 & x_4 \\
x_5 & x_6 & x_7 & x_8 \\
x_9 & x_{10} & x_{11} & x_{12} \\
x_{13} & x_{14} & x_{15} & x_{16}
\end{matrix}
$$

Motivation

Example (Latin Square)

- 4×4-matrix of 16 variables x_i ranging over $D_i = [1..4]$
- each possible value occurs exactly once in each row and each column

Variables

x_1	x_2	x_3	x_4
x_5	x_6	x_7	x_8
x_9	x_{10}	x_{11}	x_{12}
x_{13}	x_{14}	x_{15}	x_{16}

$\Longrightarrow$

Values

1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

Motivation

Example (Latin Square)

- 4×4-matrix of 16 variables x_i ranging over $D_i = [1..4]$
- each possible value occurs exactly once in each row and each column

Variables

x_1	x_2	x_3	x_4
x_5	x_6	x_7	x_8
x_9	x_{10}	x_{11}	x_{12}
x_{13}	x_{14}	x_{15}	x_{16}

$\Longrightarrow$

Values

1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

Constraints

- $\forall$ rows r_i: $\mathit{Alldifferent}(r_i)$
- $\forall$ columns c_i: $\mathit{Alldifferent}(c_i)$

What is Constraint Programming?

Terminology (Constraint Programming)

1. *Emerged from the work in artificial intelligence (AI).*
2. *Basic idea: Combine*
    - *existing search-methods (backtracking, branch-and-bound, ...)*
    - *constraint propagation techniques*
3. *CP is applied in wide-spread areas:*
    - *artificial intelligence*
    - *operations research*
    - *genome sequencing*
    - *combinatorial optimization*
    - *electrical engineering*
    - *computer algebra*
    - *natural language processing*
    - ...

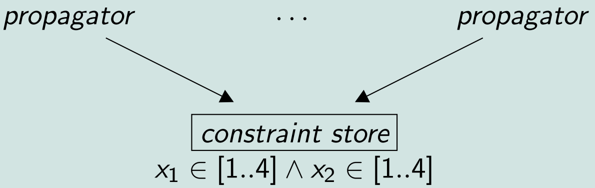
$\Rightarrow$ *method for modeling and solving many types of problems*

What are Propagators?

Terminology (Constraint Propagators)

- *fundamental concept in CP*
- *reduce search space of constraint problem (cf. "filtering,narrowing, pruning, ...").*
- *essential component of a computation space*

Computation Store



Let's propagate!

Computation Space

Propagator 1

$x + y = 9$

Propagator 2

$2x + 4y = 24$

Constraint Store

x	0	1	2	3	4	5	6	7	8	9
y	0	1	2	3	4	5	6	7	8	9

Let's propagate!

Computation Space

Propagator 1

$x + y = 9$

Propagator 2

$2x + 4y = 24$

Constraint Store

x	0	1	2	3	4	5	6	7	8	9
y	0	1	2	3	4	5	6	7	8	9

Let's propagate!

Computation Space

Propagator 1

$x + y = 9$

Propagator 2

$2x + 4y = 24$

Constraint Store

x	0	1	2	3	4	5	6	7	8	9
y	0	1	2	3	4	5	6	7	8	9

Let's propagate!

Computation Space

Propagator 1

$x + y = 9$

Propagator 2

$2x + 4y = 24$

Constraint Store

x	0	1	2	3	**4**	**5**	**6**	7	8	9
y	0	1	2	**3**	**4**	5	6	7	8	9

Let's propagate!

Computation Space

Propagator 1

$x + y = 9$

Propagator 2

$2x + 4y = 24$

Constraint Store

x	0	1	2	3	4	**5**	**6**	7	8	9
y	0	1	2	**3**	**4**	5	6	7	8	9

Let's propagate!

Computation Space

Propagator 1

$x + y = 9$

Propagator 2

$2x + 4y = 24$

Constraint Store

x	0	1	2	3	4	5	**6**	7	8	9
y	0	1	2	**3**	4	5	6	7	8	9

What are Propagators? - ctd.

Terminology (Constraint Propagators)

- *inference rule for finite domain problems*
- *implement constraint-classes (relations) C on variables $x_i \in X$ ranging over domains $D_i \in D$*
- *narrow D_i until*
  1. *failure*
  2. *entailment*
  3. *success*
- *independent*
- *variables x_i only common communication channel*

Local vs. Global

Example

Consider the following constraint satisfaction problem:

- **local** constraint:

$x \neq y$	$y \neq z$	$z \neq x$
$x \in \{1,2\}$	$y \in \{1,2\}$	$z \in \{1,2\}$

Relation: $\{(1,2),(2,1)\} \Rightarrow$ reduction **impossible**

- **global** constraint:

Alldifferent(x,y,z)		
$x \in \{1,2\}$	$y \in \{1,2\}$	$z \in \{1,2\}$

$\Rightarrow$ CSP is **inconsistent**

- stronger pruning (including earlier failure recognition)
- algorithms far more efficient w.r.t. clever theory behind
- saves posting (one *Alldifferent* constraint replaces $\binom{n}{2}$ basic constraints)

Domain vs. Bounds Consistency

Consistency Levels

Consider propagation for:

$2 \cdot x = y$	
$x \in [1 \ldots 10]$	$y \in [1 \ldots 7]$

1. **Domain consistency**:

   domain propagation narrows the domains as much as possible:

$x \in [1 \ldots 3]$	$y \in \{2, 4, 6\}$

2. **Bounds consistency**:

   interval propagation only narrows the bounds (min,max)

   $\Rightarrow$ *faster pruning*

$x \in [1 \ldots 3]$	$y \in [2 \ldots 6]$

Overview - Algorithms

Constraints

The aim of this FoPra is the implementation of the following constraints:

- *Sortedness*
- *PermSort*
- *Global Cardinality*

Framework

The implementation of these constraints will be based on the **Gecode** *framework.*

Bounds Consistent Algorithm for Sortedness

Definition (Sortedness)

Sortedness $(x_1, \ldots, x_n ; y_1, \ldots, y_n)$

1. Input: 2 sequences of n variables x_i and y_i
2. Output: Is 2^{nd} sequence obtained by sorting 1^{st} in non-decreasing order?

Example (Sortedness)

Sortedness		
$Sortedness(1,3,1;1,1,3)$	holds	✓
$Sortedness(5,2,3;3,2,5)$	violated	↯

Bounds Consistent Algorithm for Sortedness

Algorithmic Background

Efficient Algorithms for Constraint Propagation and for Processing Tree Descriptions, PhD Sven Thiel, 2004 [Thi04]

- *oriented intersection graph*
- *matching in convex bipartite graphs (Glover)*
- *strongly connected components (Mehlhorn)*

$\Rightarrow$ *complexity* $O(n+t)$, $n = |X|$ $\quad t =$ *time for sorting*

Bounds Consistent Algorithm for Global Cardinality

Definition (Global Cardinality)

$\mathrm{GCC}(x_1, \ldots, x_n ; l_1, \ldots, l_d ; u_1, \ldots, u_d)$

1. generalization of the *Alldifferent* constraint:
   - *Alldifferent*$(x_1, \ldots, x_n)$= $\mathrm{GCC}(x_1, \ldots, x_n, ; l_1, \ldots, l_d ; u_1, \ldots, u_d)$ where $\forall i \in \{1, \ldots, d\} : l_i = 0 \wedge u_i = 1$
2. Input: a sequence of n variables x_i, defined on a set of values $D = \{v_1, \ldots, v_d\}$ and for each value v_i a pair $[l_i, u_i]$
3. Output: Is it possible to narrow the domains of the variables x_i, s.t.: $\forall i \in \{1, \ldots, d = |D|\} \; \forall v_i \in D : l_i \leq \#v_i \leq u_i$?

Example (Global Cardinality)

Global Cardinality		
$GCC(2, [1..2], [2..3], [2..3], [1..4], [3..4] ; 1, 1, 1, 2 ; 3, 3, 3, 3)$	holds	✓
$GCC(2, [1..2], [2..3], [2..3], [1..4], [3..4] ; 1, 1, 1, 1 ; 1, 1, 1, 1)$	violated	↯

Bounds Consistent Algorithm for Global Cardinality

Algorithmic Background

An Efficient Bounds Consistency Algorithm for the Global Cardinality Constraint, van Beek et. al. [qui03]

- *Modification of the bounds consistency algorithm for the Alldifferent constraint*
- *Theory of Hall-Intervalls*
- *alternative implementation of the lbc constraint based on union-find datastructure*

$\Rightarrow$ *complexity* $O(t+n)$, $n = |X|$ $t =$ *sorting time*

Domain Consistent Algorithm for Global Cardinality

Algorithmic Background

Improved Algorithms for the Global Cardinality Constraint Constraint, van Beek et. al. [imp04]

- *matching in a bipartite graph*
- *strongly connected components*
- *alternating paths*

$\Rightarrow$ *complexity* $O(n * d),\ n = |X|\ \ d = |D|$

References I

[BGC00] Noëlle Bleuzen-Guernalec and Alain Colmerauer.
Optimal Narrowing of a Block of Sortings in Optimal Time.
Constraints: An International Journal, 5(1/2):85–118m, Januar 2000.

[HK73] John E. Hopcroft and Richard M. Karp.
An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs.
SIAM: Journal of Computing, 2(4):225–231, December 1973.

[imp04] *Improved Algorithms for the Global Cardinality Constraint*, volume 3528, Toronto, Canada, September 2004.

References II

[I.S04] I.S.Laboratory.
SICStus Prolog user's manual, 3.11.1 Technical Report.
Swedish Institute of Computer Science, 2004.
Download PDF-File.

[lop03] *A Fast and Simple Algorithm for Bounds Consistency of the Alldifferent Constraint*, Acapulco, Mexico, August 2003.

[LOQTvB03] Alejandro López-Ortiz, Claude-Guy Quimper, John Tromp, and Peter van Beek.
A Fast and Simple Algorithm for Bounds Consistency of the Alldifferent Constraint, Technical Report.
Acapulco, Mexico, 2003.
Download PS-File.

References III

[Meh84] Kurt Mehlhorn.
Data Structures and Algorithms, volume 2 Graph Algorithms and NP-Completeness of *EATCS Monographs*.
Springer Verlag, 1984.

[OSvE95] W. J. Older, G. M. Swinkels, and M. H. van Emden.
Getting to the real problem: Experience with bnr prolog in or.
In *Proc.of the Third International Conference on the Practical Application of Prolog*, pages 465–478, Paris, 1995.

[qui03] *An Efficient Bounds Consistency Algorithm for the Global Cardinality Constraint*, volume 2833, Kinsale, Ireland, September 2003.

References IV

[QvBLO+03] Claude-Guy Quimper, Peter van Beek, Alejandro López-Ortiz, Alexander Golynski, and Sayyed Bashir Sadjad. *An Efficient Bounds Consistency Algorithm for the Global Cardinality Constraint, Technical Report*. 2003. Download PS-File.

[Ŕ96] J-C. Régin. Generalized arc consistency for global cardinality constraint. In *Proceedings of the 13th National Conference on AI (AAAI/IAAI'96)*, volume 1, pages 209–215, Portland, August 1996.

[Reg94] *A filtering algorithm for constraints of difference in CSPs*, volume 1, Seattle, July 31 - August 4 1994.

References V

[S.A00] ILOG S.A.
ILOG Solver 5.0:Reference Manual.
2000.

[SS04] Christian Schulte and Gert Smolka.
Finite Domain Constraint Programming in Oz. A Tutorial, 1.3.0 edition.
2004.
Download PDF-File.

[Thi04] Sven Thiel.
Efficient Algorithms for Constraint Propagation and for Processing Tree Descriptions, PhD Thesis.
Universität des Saarlandes, Saarbrücken, Germany, 2004.
Downloadpdf: PDF-File.

References VI

[WNJ97] Mark Wallace, Stefano Novello, and JoachimSchimpf.
Eclipse: A platform for constraint logic programming.
Technical report.
IC-Parc, Imperial College, London, UK, 1997.
Online-Version.

[Zho73] Jianyang Zhou.
A permutation-based-approach for solving the job-shop problem.
Constraints: An International Journal, 2(2):185–213, October 1973.

Bounds Consistent Algorithm for PermSort

Definition (PermSort)

PermSort($x_1, \ldots, x_n$; $y_1, \ldots, y_n$; $p_1, \ldots, p_n$)

1. equals *Sortedness* with respect to permutation variables
2. Input: 3 sequences of n variables x_i, y_i and p_i
3. Output: Is 2^{nd} sequence obtained by sorting 1^{st} in non-decreasing order AND is 3^{rd} sequence a permutation, s.t.:
   - $\forall i \in \{1, \ldots, n\} : p_i(x_i) = y_i$

Example (PermSort)

PermSort		
PermSort$(1, 3, 1; 1, 1, 3; 1, 3, 2)$	holds	✓
PermSort$(5, 2, 3; 2, 3, 5; 1, 2, 3)$	violated	↯

Bounds Consistent Algorithm for PermSort

Algorithmic Background

A Permutation-Based Approach for Solving the Job-Shop Problem, Jianyang Zhou, Constraints an International Journal 1997 [Zho73]

- *alternative to an extension of Thiel's algorithm for the Sortedness constraint*
- *using Alldifferent propagator*

$\Rightarrow$ *complexity* $O(n^2),\ n = |X|$