Alldifferent constraint solver in SMT

Milan Banković   Filip Marić
{milan,filip}@matf.bg.ac.rs

Department of Computer Science
Faculty of Mathematics
University of Belgrade

8th International Workshop on Satisfiability Modulo Theories.
Edinburgh 2010

Outline

1. Introduction to alldifferent constraint
2. The alldifferent theory
3. Our alldifferent *SMT* solver
4. Experimental evaluation
5. Future work and conclusions

Outline

Definition of alldifferent constraint

Definition

Given a set of variables $x_1, x_2, \dots, x_n$, where each variable x_i takes values from its corresponding finite domain $D(x_i)$, then $\mathit{alldiff}\,(x_1, x_2, \dots, x_n)$ means that every two different variables must take different values ($i \neq j \Rightarrow x_i \neq x_j$).

Applications

Broad variety of problems can be reduced to alldifferent:

- Puzzle solving (Sudoku, Latin Square, Eight Queens).
- Scheduling and timetabling.

Example of alldifferent based problem

Example (Latin square 5×5)

	3	4		
3	4	5		
4	5			
5				

Notes

- Each cell should be filled in with a value from 1 to 5.
- Each row and each column is constrained by alldifferent constraint.
- Some values are already given.

Example of alldifferent based problem

Example (Latin square 5×5)

1	2	3	4	5
2	3	4	5	1
3	4	5	1	2
4	5	1	2	3
5	1	2	3	4

Notes

- Each cell should be filled in with a value from 1 to 5.
- Each row and each column is constrained by alldifferent constraint.
- Some values are already given.

Representation of problem

$$
\begin{gathered}
D(x_{11}) = \{1, \ldots, 5\} \\
D(x_{12}) = \{1, \ldots, 5\} \\
\ldots \\
D(x_{55}) = \{1, \ldots, 5\} \\
alldiff(x_{11}, x_{12}, x_{13}, x_{14}, x_{15}) \\
alldiff(x_{21}, x_{22}, x_{23}, x_{24}, x_{25}) \\
\ldots \\
alldiff(x_{15}, x_{25}, x_{35}, x_{45}, x_{55}) \\
x_{51} = 5 \\
x_{41} = 4 \\
\ldots \\
x_{23} = 4
\end{gathered}
$$

Notes

- For each cell one variable is introduced.
- For each variable, its domain consists of values $\{1, \ldots, 5\}$.
- For each row and each column – one alldifferent constraint.
- For each given value – one equality constraint.

Representation of solution

$$x_{11} = 1, x_{12} = 2, \ldots, x_{15} = 5$$
$$x_{21} = 2, x_{22} = 3, \ldots, x_{25} = 1$$
$$\ldots$$
$$x_{51} = 5, \ldots, x_{52} = 1, x_{55} = 4$$

Notes

Solution of the problem is an assignment of values to variables satisfying all the constraints

Related work

Related work

- Algorithm implemented within CP solvers
- Arc-consistency for alldifferent – (Régin 1994)
- Explanation generating procedures in CP (Gent et al, 2010)
- Infinite and large domains (Quimper and Walsh, 2004)
- Bitvector alldifferent consistency checking (Biere and Brummayer, 2008)
- Usage of alldifferent within SMT (Nieuwenhuis et al, 2007)

What is our goal?

The goal is to...

- express a single alldifferent constraint as a first order theory
- construct a $DPLL(T)$-compliant theory solver (AD-solver) for such theory
- combine multiple AD-solvers to solve problems with multiple alldifferent constraints (Delayed Theory Combination)

Outline

The alldifferent theory

Signature and axioms

- The signature consists of constant symbols $\overline{x}_i$ for each variable x_i and constant symbols $\overline{d}_j$ for each value $d_j \in \bigcup_i D(x_i)$
- Axioms of sanity: $\overline{d}_i \neq \overline{d}_j$, for $i \neq j$ (different constants represent different values)
- Domain axioms: $\bigvee_{d \in D(x)} \overline{x} = \overline{d}$ for each variable x (x should take value from its domain)
- Axioms of difference: $\overline{x}_i \neq \overline{x}_j$ for $i \neq j$ (ensure alldifferent is satisfied)

Outline

AD-solver functionality:

- Conflict detection (based on optimal matchings)
- Theory propagation (based on Régin's algorithm)
- Propagation and conflict explaining (based on Minimal Obstacle Sets (MOS), contributed)

Matching problem and alldifferent

AD-solver

- *AD*-solver is based on the matching problem in bipartite graphs
- A bipartite graph is assigned to the alldifferent constraint (called its value graph)

Example of value graph

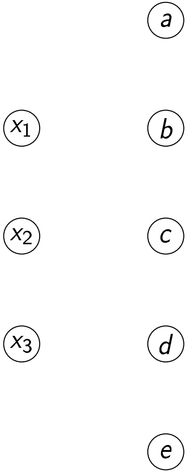
Example

$D(x_1) = \{a, c, d\}$, $D(x_2) = \{b, d, e\}$,
$D(x_3) = \{a, b\}$, $alldiff(x_1, x_2, x_3)$

Notes

- Each vertex at the left side corresponds to one variable.

Example of value graph



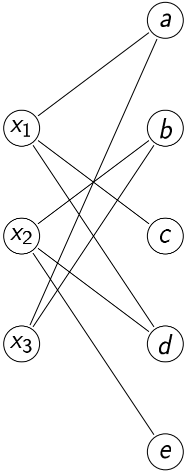
Example

$D(x_1) = \{a, c, d\}$, $D(x_2) = \{b, d, e\}$,
$D(x_3) = \{a, b\}$, $alldiff(x_1, x_2, x_3)$

Notes

- Each vertex at the left side corresponds to one variable.
- Each vertex at the right side corresponds to one value.

Example of value graph



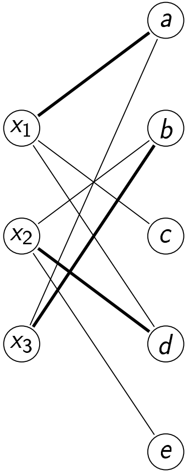
Example

$D(x_1) = \{a, c, d\}$, $D(x_2) = \{b, d, e\}$, $D(x_3) = \{a, b\}$, $alldiff(x_1, x_2, x_3)$

Notes

- Each vertex at the left side corresponds to one variable.
- Each vertex at the right side corresponds to one value.
- Each variable is connected to values from its domain.

Example of value graph



Example

$D(x_1) = \{a, c, d\}$, $D(x_2) = \{b, d, e\}$,
$D(x_3) = \{a, b\}$, $alldiff(x_1, x_2, x_3)$

Notes

- Each vertex at the left side corresponds to one variable.
- Each vertex at the right side corresponds to one value.
- Each variable is connected to values from its domain.
- Solution corresponds to matching that covers left side vertices.

Matching problem and alldifferent

Important concepts

- Alternating path – path containing edges that alternatively belong to the current matching
- Augmenting path – alternating path connecting unmatched vertices at opposite sides
- Directed value graph – the value graph with edges oriented from left to right if they belong to the matching, and from right to left otherwise

Conflict detection

Conflict detection in alldifferent

- Optimal matchings – matchings with maximal cardinality
- alldifferent is satisfiable iff an optimal matching covers left side vertices
- An optimal matching construction – augmenting paths based algorithms (e.g. Hopcroft&Karp)
- The current matching is incrementally augmented using found augmenting paths until an optimal matching is reached (incremental property)

Theory propagations

Classification of edges in the value graph

- Vital edge – belongs to all optimal matchings (equality propagated)
- Inconsistent edge – doesn't belong to any optimal matching (disequality propagated)
- Alternating edge – neither vital nor inconsistent

Theory propagations

Régin's algorithm

- Searching for alternating edges:
  - search for edges reachable from unmatched vertices (BFS, using alternating paths)
  - search for edges that belong to alternating cycles (Tarjan's algorithm for strongly connected components)
- Non-alternating (vital or inconsistent) edges cause propagations

Conflict and propagations explaining

Algorithm for finding minimal explanation

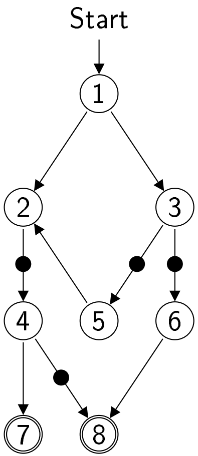
- The explanation generating procedure is the main contribution of our work
- The algorithm we propose finds explanations that are minimal in sense of inclusion.
- The problem of propagation and conflict explaining is reduced to the Minimal Obstacle Set Problem (MOS), also introduced and solved in our work

Conflict and propagations explaining

MOS problem

- MOS problem is the problem of finding subcut of a given cut in a directed graph that is minimal in sense of inclusion
- Edges belonging to the cut are called obstacles in our terminology (hence the name of the problem)
- Efficiency: the algorithm executes in one graph traversal

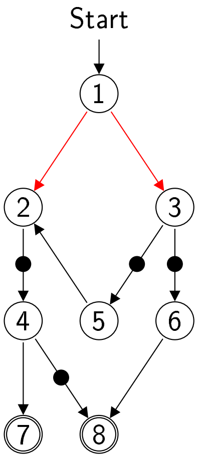
Conflict and propagations explaining



Notes

- Start vertex 1, final vertices $\{7, 8\}$.

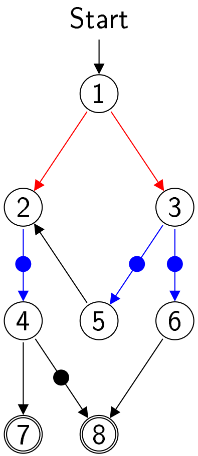
Conflict and propagations explaining



Notes

- Start vertex 1, final vertices $\{7, 8\}$.
- Check for reachable obstacles.

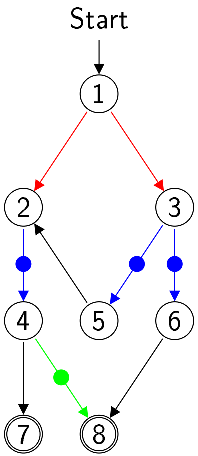
Conflict and propagations explaining



Notes

- Start vertex 1, final vertices $\{7, 8\}$.
- Check for reachable obstacles.

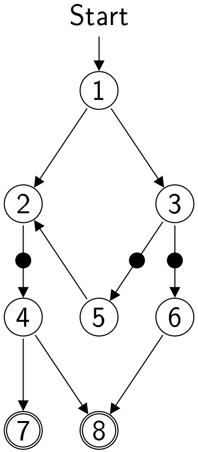
Conflict and propagations explaining



Notes

- Start vertex 1, final vertices $\{7, 8\}$.
- Check for reachable obstacles.
- Obstacle $(4, 8)$ is unreachable. It may be released.

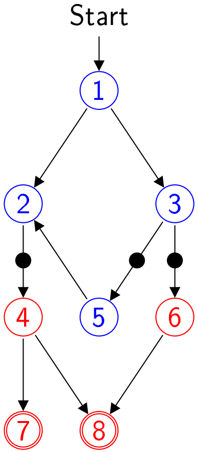
Conflict and propagations explaining



Notes

- Start vertex 1, final vertices $\{7, 8\}$.
- Check for reachable obstacles.
- Obstacle $(4, 8)$ is unreachable. It may be released.

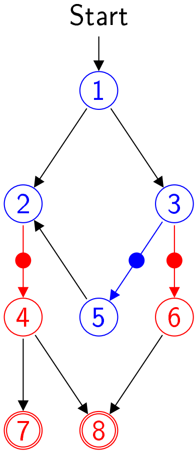
Conflict and propagations explaining



Notes

- Start vertex 1, final vertices $\{7, 8\}$.
- Check for reachable obstacles.
- Obstacle $(4, 8)$ is unreachable. It may be released.
- Blue vertices are blocked. Red vertices are unblocked.

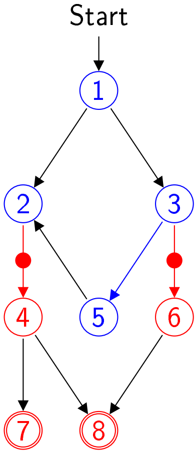
Conflict and propagations explaining



Notes

- Start vertex 1, final vertices $\{7, 8\}$.
- Check for reachable obstacles.
- Obstacle $(4, 8)$ is unreachable. It may be released.
- Blue vertices are blocked. Red vertices are unblocked.
- Obstacle $(3, 5)$ may be released, because 3 and 5 are both blocked.

Conflict and propagations explaining



Notes

- Start vertex 1, final vertices $\{7, 8\}$.
- Check for reachable obstacles.
- Obstacle $(4, 8)$ is unreachable. It may be released.
- Blue vertices are blocked. Red vertices are unblocked.
- Obstacle $(3, 5)$ may be released, because 3 and 5 are both blocked.
- Minimal set of obstacles: $(2, 4)$ and $(3, 6)$.

Conflict and propagations explaining

Reduction to MOS problem

- Obstacle set: edges removed from the value graph due to literals in the partial model
- Conflict explaining: Non-optimal matching cannot be augmented – unmatched vertices at the opposite sides are separated by the obstacles
- Propagation explaining: Edges became non-alternating – they are separated from the unmatched vertices by the obstacles (and/or alternating cycles containing them are broken by the obstacles)

Outline

Experimental evaluation

Experiments

- Sudoku instances 25×25, randomly generated with around 40% cells filled in
- 200 instances, time limit: 2 minutes per instance
- Compared solvers: argoalldiff (with and without explanations), argosat, minion, yices (EUF encoding, SMT-LIB's distinct)

Experimental evaluation

Solver	**Solved instances**	**Average time on solved instances**
argoalldiff (with expl.)	194	8.8s
minion	174	10s
argosat	172	18s
argoalldiff (without expl.)	169	18s
yices (using `distinct`)	0	-
yices (using EUF)	0	-

Experimental evaluation

Experimental evaluation

- Average number of conflicts: 460 (argoalldiff) vs 33000 (argosat)
- Reduction of explanations size: 27% for theory propagations, 9% for conflicts
- Explanation procedures time consumption: only 3% of the overall execution time

Outline

Future work

Future work

- Efficiency improvement of our prototypical implementation
- Possible application: timetabling (teaching timetable for Faculty of Mathematics)

Conclusions

Conclusions

- Alldifferent constraint: expressed as a first order theory
- *SMT*-approach: reducing alldifferent based problems to *SMT*.
- Efficient theory solver based on the matching problem.
- Conflict and propagation explaining: new efficient algorithm is proposed.

THANK YOU :)