The AllDifferent Constraint: Efficiency Measures

Ian P. Gent, Ian Miguel, Peter Nightingale
**AllDifferent**

- A vector of variables must take distinct values
- Very widely used – very important
- Examples:
  - A class of students must have lectures at distinct times
  - In a sports schedule, the teams playing on a particular week are all distinct
  - No pair of golfers play together more than once
  - Sudoku
Van Hoeve surveys various strengths of inference.

In order of increasing strength:
- Weak and fast pairwise decomposition (AC) -- $O(r)$
- Bound consistency – find Hall intervals (as described by Toby) and prune bounds – $O(r \log r)$
- Range consistency – find Hall intervals and prune
- Generalised arc consistency (GAC) – $O(k^{0.5} r d)$

We focus on GAC algorithm by Régin.
GAC AllDifferent

• One expensive pass achieves consistency
• Traditionally has large incremental, backtracked data structure
• Traditionally low priority
• Triggered on any domain change
  – But many changes are processed together
• No paper that we are aware of comprehensively investigates implementation decisions
Our approach

• Investigate optimizations in literature (tried to find everything!)

• Trigger only on relevant values
  – It is not necessary to trigger on all domain removals
  – Identify $O(2r+d)$ trigger values

• Partition the constraint dynamically
  – Algorithm already identifies independent sub-constraints
  – Store and re-use this partition
  – Run expensive algorithm only on sub-constraint
Régin's Algorithm

- Find a maximum matching $M$ from variables to values.
  - Corresponds to a satisfying tuple of the constraint
- If $|M| < r$, the constraint is unsatisfiable
- Construct residual graph $R$ (as described later)
- Edges not in $M$, and in no cycle in $R$, correspond to values to prune
Régin's Algorithm

- Described in terms of flow, Ford-Fulkerson BFS algorithm
- Alternative is bipartite graph matching, Hopcroft-Karp or other algorithm
Régin's Algorithm

- Find maximum flow from s to t
- Ford-Fulkerson algorithm
Régin's Algorithm

- Find maximum flow from s to t
- Ford-Fulkerson algorithm
Régis's Algorithm

- Find maximum flow from $s$ to $t$
- Ford-Fulkerson algorithm
Régin's Algorithm

- Find maximum flow from s to t
- Ford-Fulkerson algorithm
Régis's Algorithm

- Find maximum flow from s to t
- Ford-Fulkerson algorithm
Régin's Algorithm

- Find maximum flow from s to t
- Ford-Fulkerson algorithm
Régin's Algorithm

- Find maximum flow from s to t
- Ford-Fulkerson algorithm
Régin's Algorithm

- Find maximum flow from s to t
- Ford-Fulkerson algorithm
Régin's Algorithm

- Completed maximum flow from s to t
- Covers all variables (constraint is satisfiable)
- One of 24
Régin's Algorithm

• Find strongly-connected components
Régin's Algorithm

- Strongly-connected components (SCCs)
  - Vertices \(i\) and \(j\) in same SCC iff:
    - Path from \(i\) to \(j\) and from \(j\) to \(i\) in digraph
  - Found by Tarjan's algorithm
    - DFS
  - SCC='Maximal set of cycles'
Régis's Algorithm

- Find strongly-connected components
Régis's Algorithm

- Cycle within SCC
- Apply cycle to find different maximum flow
- No cycles between SCCs
Régin's Algorithm

- Cycle within SCC
- Apply cycle to find different maximum flow
- No cycles between SCCs
Régis's Algorithm

- No cycles between SCCs
- No maximum flows involving \( x_3=2 \) or \( x_4=2 \)
Régin's Algorithm

- Remove edges which are:
  - Between SCCs
  - Not in flow
- Corresponds to theorem by Berge, 1973
Implementation

• Key assumption: don't maintain the graph, discover it as you traverse
  – Domain queries cheap in Minion
  – Alternative: maintain and BT adjacency lists, size $O(rd)$
  – We claim this is better without experiment
  – If Patrick reads the paper, I'm in trouble!
  – If the assumption is not true, our experiments are somewhat less reliable, but the big results should still hold
Optimizations in Literature

- Incremental matching (Régin)
- Priority Queue
  - Execute at low priority and with no duplicate events
- Staged propagation (Schulte & Stuckey)
  - Do simple propagation at high priority, GAC at low priority
- Domain counting (Quimper & Walsh)
- Fixpoint reasoning (Schulte & Stuckey)
  - Solves the 'Double Call Problem'
- Advisors (Lagerkvist & Schulte)
Priority Queue

Instance Families
- contrived
- golomb
- langford
- quasigroup
- n queens
- QWH
- social golfers
- sports scheduling
Incremental Matching
FF-BFS vs HK

- FF is also much easier to implement!
Staged propagation

- Very simple, deals with assigned vars at high priority
Triggering

• Trigger only on relevant values (Dynamic Triggers)
  – It is not necessary to trigger on all domain removals
  – Identify t ≤ 2r+d trigger values from rd
  – Doesn't work on our instances!
  – Ratio not low enough
Triggering

- Domain counting (Lagerkvist & Schulte, variant of Quimper & Walsh)
  - Only trigger when domain size less than \( r \)
  - Very cheap but has almost no effect
- Fixpoint reasoning and advisors
  - No claim in original papers that these are useful for AllDifferent
  - DT results suggest fixpoint reasoning is useless
  - We have something like advisors (although more general) – the variable event queue!
Partitioning the constraint

- Partition by SCCs
  - Each SCC corresponds to an independent sub-constraint
  - Store and re-use this partition (of the variables)
  - Run expensive algorithm only on sub-constraint
Partitioning the constraint

- Small incremental data structure which backtracks efficiently

<table>
<thead>
<tr>
<th>setElements:</th>
<th>setElementIndex:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6</td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5 6</td>
<td></td>
</tr>
</tbody>
</table>

Partition this set into \(\{1,3,5\}\),\(\{2,4,6\}\)

<table>
<thead>
<tr>
<th>setElements:</th>
<th>setElementIndex:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3 5 4 2 6</td>
<td></td>
</tr>
<tr>
<td>1 4 2 5 3 6</td>
<td></td>
</tr>
</tbody>
</table>

\(\text{splitPoint}[3]=true\) indicates that adjacent elements 5 and 2 are in different subsets in the partition.
Partitioning the constraint
Partitioning the constraint

- Worth considering for other large constraints
  - GAC GCC partitions in the same way
  - Graph connectivity partitions when you find a 'bridge'
  - Sequence constraint?
  - Regular/Slide partition when variables are assigned in middle
Pairwise AllDifferent

- Trigger only on assignment of a variable
- Remove assigned value from all other variables
- Extremely cheap
- Equivalent to AC on pairwise not-equal constraints
- This is no straw man!
Comparing to Pairwise
Comparing to Pairwise

- GAC AllDifferent never slows down search by more than 2.34 times
- Can be 100,000 times faster
- Most AllDifferent constraints here are tight

<table>
<thead>
<tr>
<th>Instance Families</th>
<th>contrived</th>
<th>golomb</th>
<th>langford</th>
<th>quasigroup</th>
<th>n queens</th>
<th>QWH</th>
<th>social golfers</th>
<th>sports scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+</td>
<td>×</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>
Modelling with AllDifferent

- Golomb Ruler
  - Triangular table representing all pairs
  - One AllDifferent constraint
  - Optimization tightens AllDiff
  - Implied constraints

<table>
<thead>
<tr>
<th>Ruler:</th>
<th>0</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>... (monotonic)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diffs:</td>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B-A</td>
<td>C-A</td>
<td>D-A</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C-B</td>
<td>D-B</td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D-C</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Modelling with AllDifferent

• Langford's problem with 2 instances of each number
  – Model due to Rendl
  – Permutation
  – Represent the indices rather than the actual Langford sequence

For Langford sequence of length n with n/2 numbers.
Pos[1..n] -- AllDifferent
// 1 from the second instance of 1
...
Modelling with AllDifferent

- Quasigroup and QWH
  - Similar to Sudoku (without the sub squares)
  - \( n \times n \) matrix of variables with domain 1..n
  - AllDifferent on each row and each column
  - QWH has some values filled in already
    - Well known to show off GAC AllDifferent
  - Quasigroup has various properties (e.g. associativity, idempotence)
    - Colton & Miguel's model and implied constraints
Modelling with AllDifferent

- N Queens problem
  - Model 1
    - Three vectors representing queen position in row, the number of the leading diagonal, and the number of the secondary diagonal
    - These vectors are all different
  - Model 2
    - One vector representing queen position in row (all different)
    - Constraints to forbid diagonals
    - Tailor creates 30 auxiliary variables for n=16
Modelling with AllDifferent

- Sports scheduling
  - Two viewpoints
    - For each week, a vector of the teams (all different)
    - Vector of games (all different)
    - Channelling constraints between the two (table)
    - Symmetry breaking constraints (< for each game, lex on weeks, lex on stadiums)
    - Stadium constraints (each team plays no more than twice in one stadium)

<table>
<thead>
<tr>
<th>Stadium 1</th>
<th>Stadium 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1:</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Modelling with AllDifferent

- **Social Golfers**
  - Very similar to sports scheduling
  - Two viewpoints
    - For each week, a vector of the golfers (all different)
    - Vector of pairs who played together (all different but not necessarily a permutation)
    - Channelling constraints between the two (table)
    - Symmetry breaking constraints (< within the groups, lex on weeks, lex between groups)

```
Week 1:
1  2  4  5  3  6  7  9  ...
```
Modelling with AllDifferent

- As you can see, AllDifferent is widely used! 7 example problems.
- The AllDifferent is tight in all examples
  - In a lot of cases it is worth doing GAC, but not all
  - I think it does depend on tightness, but also on other constraints surrounding the AllDifferent
  - I refuse to offer any advice!
Conclusions

- A bag of useful tricks from the literature
- One new trick which worked: partitioning the constraint
  - Perhaps this is general!
- One new trick which didn't: dynamic triggers from SCC algorithm
- The only modelling advice is to try a couple of different propagators!
Thank You

• Any Questions?