Neighborhood Interchangeability and Dynamic Bundling for Non-binary CSPs

Anagh Lal  
*University of Nebraska-Lincoln, alal@cse.unl.edu*

Berthe Y. Choueiry  
*University of Nebraska-Lincoln, choueiry@cse.unl.edu*

Eugene C. Freuder  
*University College Cork, e.freuder@cs.ucc.ie*

Follow this and additional works at: [http://digitalcommons.unl.edu/cseconfwork](http://digitalcommons.unl.edu/cseconfwork)

Part of the [Computer Sciences Commons](http://digitalcommons.unl.edu/cseconfwork)

[http://digitalcommons.unl.edu/cseconfwork/168](http://digitalcommons.unl.edu/cseconfwork/168)

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Conference and Workshop Papers by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.
Interchangeability partitions domains in a memory
Intersecting the NI sets from the nb
Haselböck, A: Exploiting Interchangeabilities in Constraint Satisfaction Problems. IJCAI 93.

- Tested random CSPs, Model B, 1000 instances per sample
- Finding multiple, robust solutions.

Problems mostly
Dynamic bundling
Maximum no-
Compaction rate achieved on a random data

- Assist in query size estimation
- Use new join algorithm when materializing join queries.

Large FBS: 33 at
Reduces number of tuples compared in the main memory.

Low tightness
- Large FBS: 33 at \( t = 0.35 \)
- Small bundling overhead.

Phase transition
- Multiple solutions exist.
- Maximum no-good bundling yields max savings in NV, \& CC.

High tightness
- Problems mostly unsolvable.
- Minimal bundling overhead.

Increasing a, we note in the phase-transition area:
- FBS increases (more chances for asymmetry).
- CPU time decreases (better no-good bundling).

The benefits of DynBndl increase with increasing domain size; Use DynBndl in database applications where large domains are typical.

Contribution

1. **Interchangeability**: An algorithm for computing interchangeability in non-binary CSPs.
2. **Dynamic bundling**: Integration of the above with backtrack search for solving non-binary CSPs.
3. **Experiments** demonstrating the benefits of dynamic bundling
   - Finding multiple, robust solutions.
   - Decreasing computational cost of search.

**Constraint Satisfaction Problems**

- A Constraint Satisfaction Problem (CSP) is a combinatorial decision problem defined by a set of variables, a set of domain values for these variables, and a set of constraints restricting the allowable combinations of values for variables, where the task is to find a solution (i.e., an assignment of a value to each variable satisfying all constraints), or to find all such solutions.

- **Interchangeability** identifies values equivalent in all solutions of a CSP [2].
- Full Interchangeability (FI): \( d, e, f \) / \( c \) can be swapped for \( v \) in any solution.
- Neighborhood Interchangeability (NI): finds \( e \) / \( f \) but misses \( d \). It efficiently approximates FI.

- **Bundling = Search + neighborhood interchangeability** [3]
- Dynamic bundling = Search + dynamic NI [1]

- Bundling = Search + neighborhood interchangeability [3]
- Dynamic bundling = Search + dynamic NI [1]

- **Interchangeability in non-binary CSPs**

- We show how to compute NI for non-binary constraints by:
  1. Building the non-binary discrimination tree, nb-DT(\( V, C \)), a data-structure that determines the NI sets of a variable \( V \) given a constraint \( C \) defined on \( V \).
  2. Intersecting the NI sets from the nb-DTs of a set of constraints, which yields the domain partition of the variable \( V \) given the constraints.

- **Future Research Directions**

- Use new join algorithm when materializing join queries.
- Explore bundled results in data-analysis/data-mining packages.
- Assist in query size estimation
- Improve accuracy of sampling operators