Dynamically Detecting and Exploiting Symmetry in Finite Constraint Satisfaction Problems

Amy M. Davis
Constraint Systems Laboratory
Department of Computer Science and Engineering
University of Nebraska-Lincoln
amydavis@cse.unl.edu

April 16, 2002

Supported in part by NASA Nebraska Space Grant WBS # 26-0511-0025-001 and the Center for Communication and Information Science (CCIS).

Subject of Thesis

Area Constraint Satisfaction for combinatorial problems

Questions How to detect and use **dynamic** symmetry to

1. Find families of solutions
2. Improve performance of problem solving

Methods Theoretical and empirical evaluations in search

Results

1. Establish superiority and effectiveness of approach
2. Dispel fears/doubts about dynamic symmetries
3. Identify directions for practical applications and theoretical investigations

Constraint Satisfaction Problem (CSP): Definition

Given $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$:

- $\mathcal{V}$ a set of variables
$$\mathcal{V} = \{V_1, V_2, \ldots, V_n\}$$
- $\mathcal{D}$ a set of variable domains (values that a variable may take)
$$\mathcal{D} = \{D_{V_1}, D_{V_2}, \ldots, D_{V_n}\}$$
- $\mathcal{C}$ a set of constraints
$$\mathcal{C}_{V_a, V_b, \ldots, V_i} = \{(x, y, \ldots, z)\} \subset D_{V_a} \times D_{V_b}, \times \ldots \times D_{V_i}$$

Question: Can we assign a value to each variable such that all constraints are satisfied?

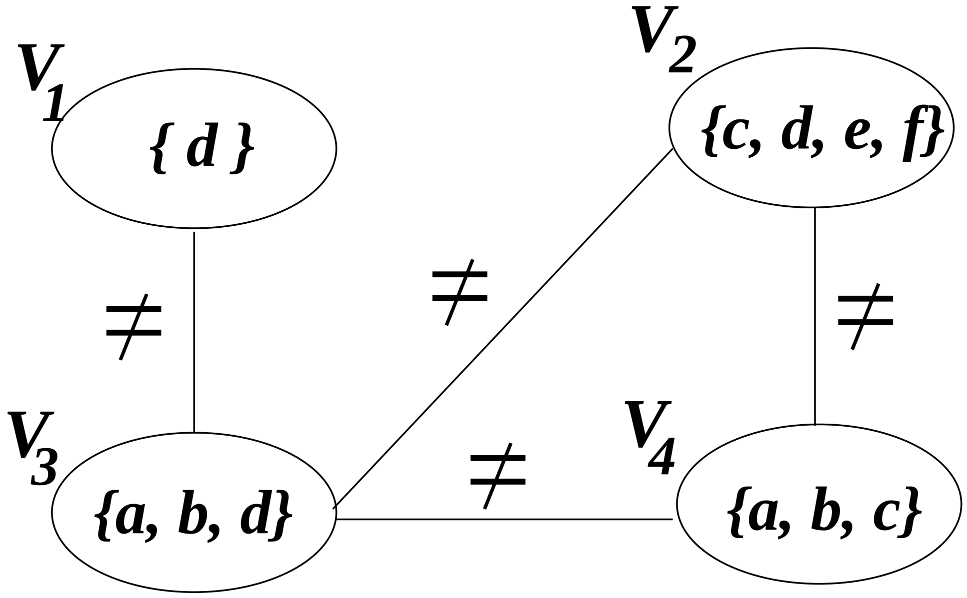
In general, **NP**-complete problem

CSP: Representation

Variable $\longrightarrow$ node

Domain $\longrightarrow$ node label

Constraint $\longrightarrow$ edge between nodes



CSP: Applications

- Scheduling [Fox 1987]
- Resource allocation [Choueiry 1994]
- AI planning [Kambhampati 1999]
- Collaborative problem-solving [Petrie 1996]
- Database [Kanellakis et al. 1990]
- Bioinformatics [Backofen 2001]

Solving a CSP

Basic method

— Backtrack search, exponential

Most effective improvements

— Lookahead filtering

— Dynamic variable-value ordering

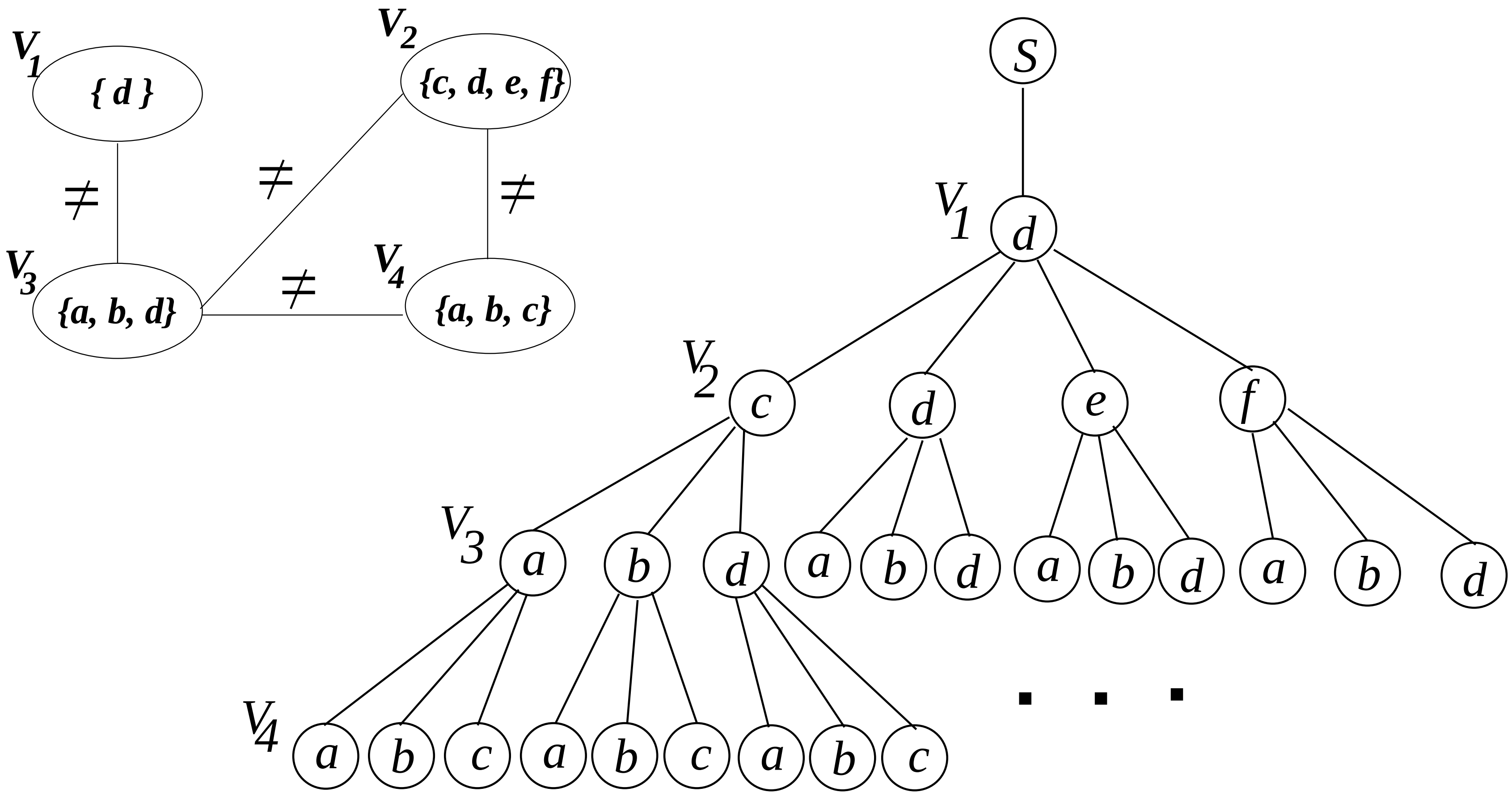
Exploit structure (problem specific)

— Real-life problems have a non-random structure

— Example: topology of graph, semantics of constraints

— **Our choice: symmetry**

Search Space for Backtracking Search

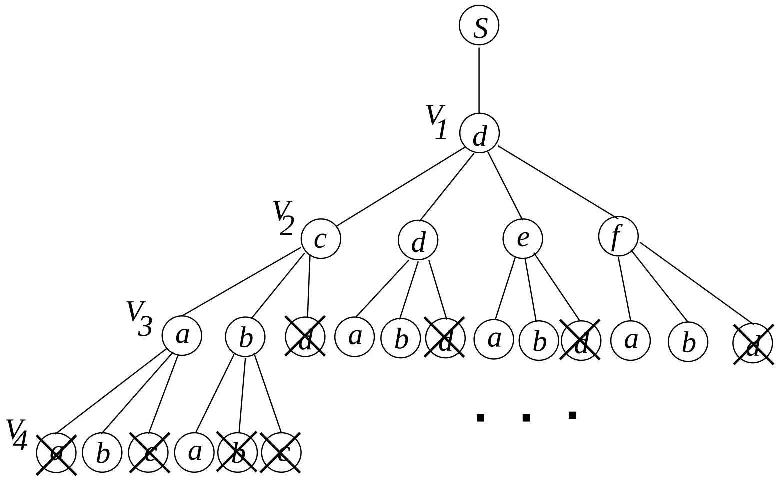


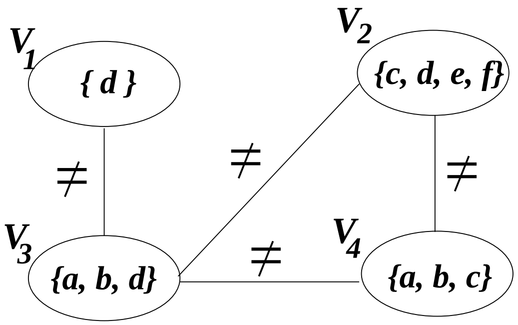
Lookahead Filtering

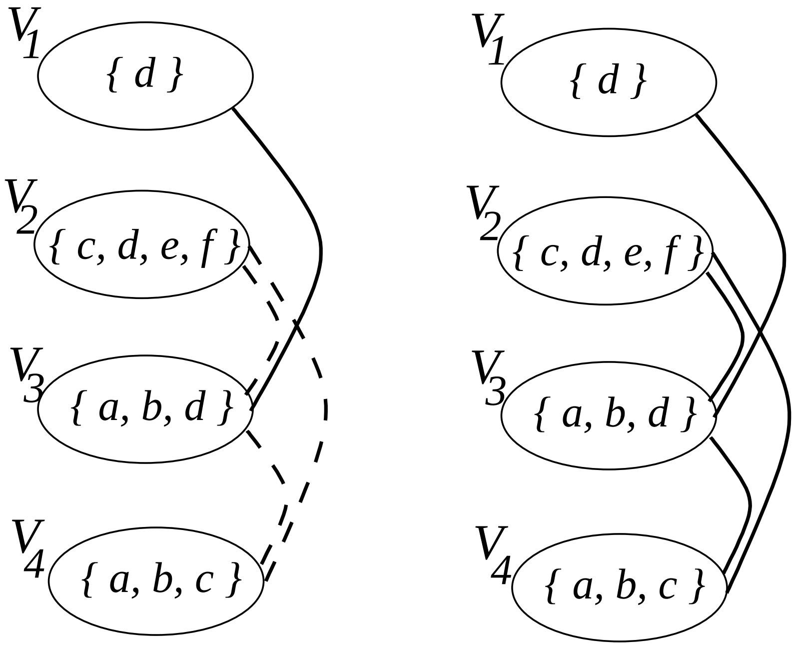
– Partial lookahead: Forward checking (FC)

– Full lookahead: Maintaining arc consistency (MAC)

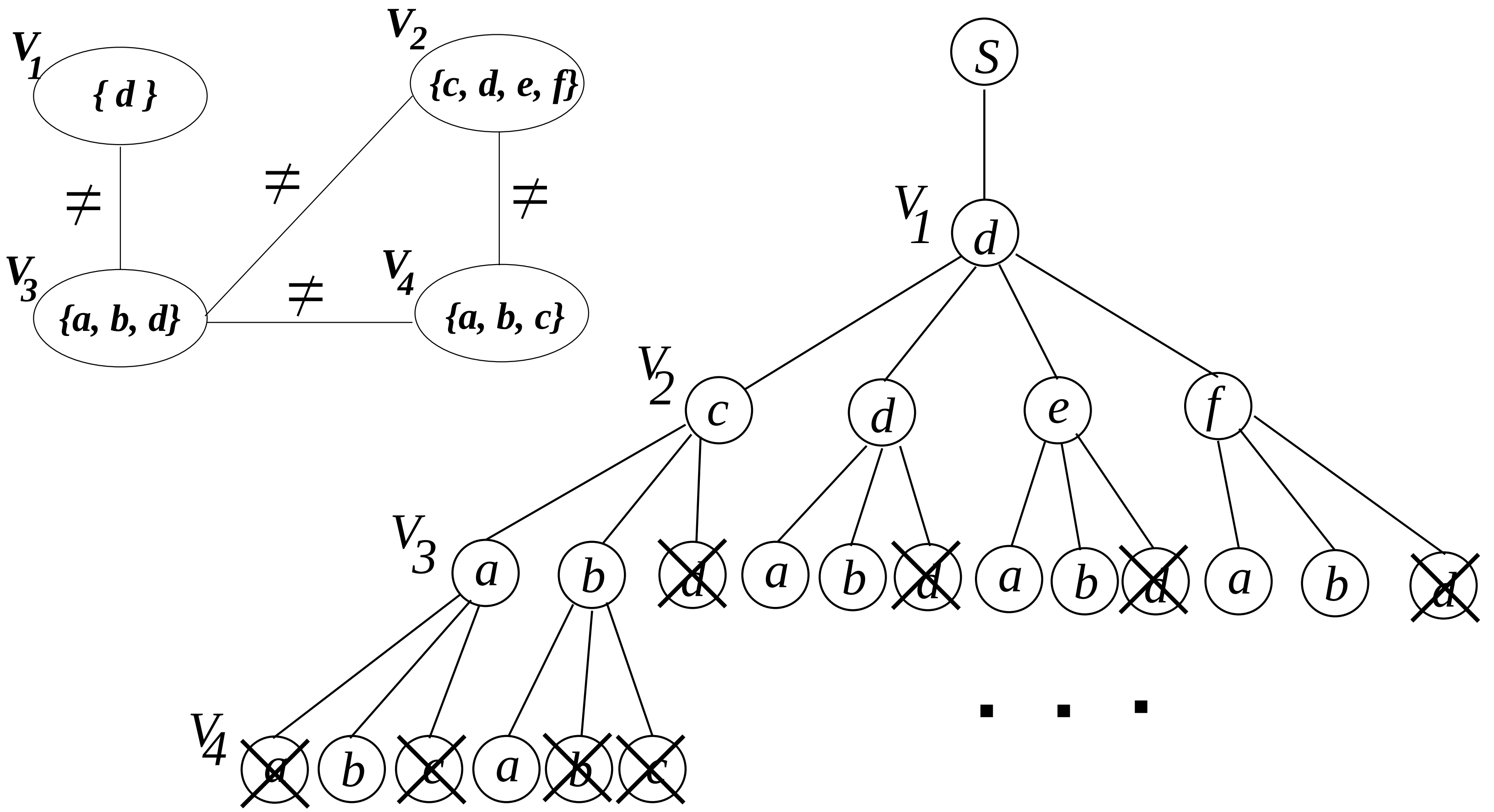
[Sabin and Freuder, 1994]







Order of Variables and Values During Search



Variable-Value Ordering

Goal:	Move through the search space faster Backtrack less
Variable ordering:	Fail first principle
Value ordering:	Find the most promising value
Ordering:	Static Dynamic
Tested here:	Static Least Domain, `SLD` Dynamic Least Domain, `DLD` Promise, [Geelen 1992] **New** ordering strategies

Exploit Problem Structure: Symmetries

Declare symmetries: [Glaisher 1874; Ellman 1993]

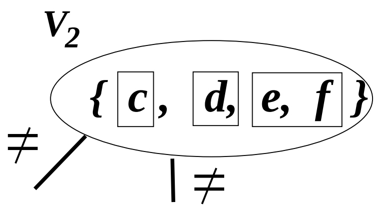
- Applies to specific class of problems
- Not general enough to apply to **any** problem
- Requires that the user or designer know the structure

Discover symmetries: [Freuder 1991]

- Solver **discovers** the symmetries, called **interchangeability**
- Global interchangeability may be **NP**-hard.
  Strong (perfect) interchangeability may be rare.
- Local interchangeability is tractable.
  Weak (imperfect) interchangeability is common.

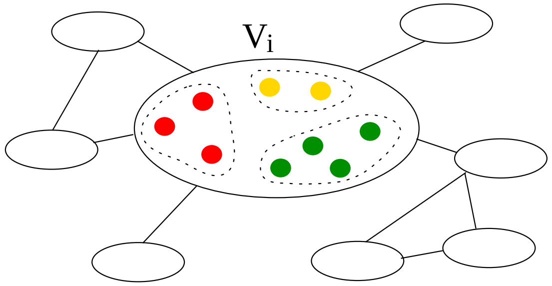
Interchangeability: The Basics

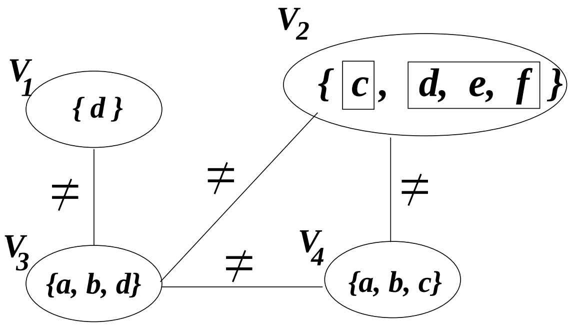
– Two interchangeable values for a variable behave the same
  * globally (likely intractable), or
  * locally (tractable).
– Interchangeable values are redundant.
– Interchangeable values can be placed in a **bundle**, and treated as a single value.



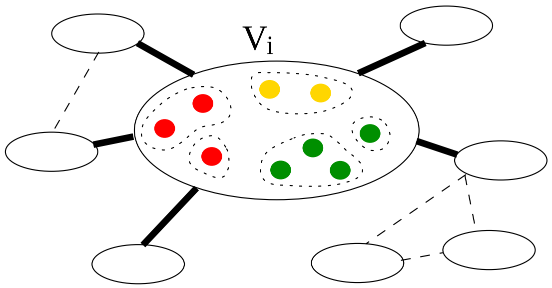
Interchangeability: Definitions

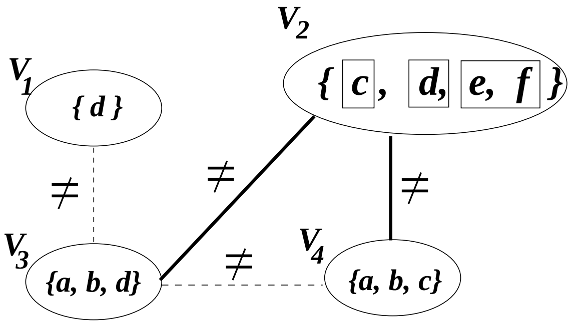
Full





Neighborhood, restricted to Neigh($\{V_i\}$)





Neighborhood Interchangeability according to one Constraint (NI$_C$)

[Haselböck 93]

Neighborhood Partial Interchangeability (NPI) [Choueiry & Noubir 98]

Interchangeability in Search

Neighborhood Interchangeability according to one Constraint $\mathbf{NI}_C$

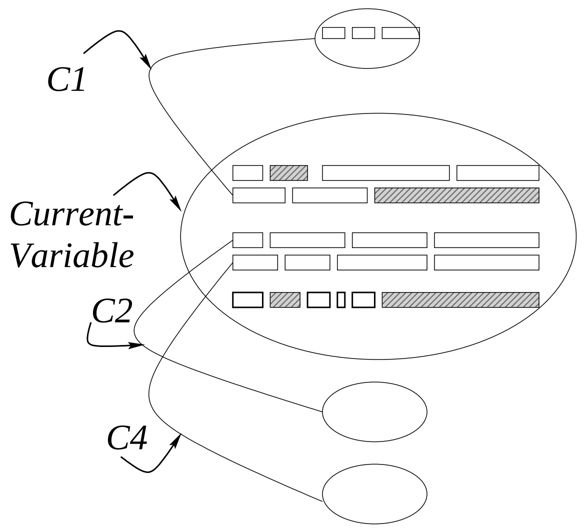
[Haselböck 1993]

– Compute all locally interchangeable values for all variables.

– Store these interchangeable values.

– Perform search, substituting each set of interchangeable values with one representative.

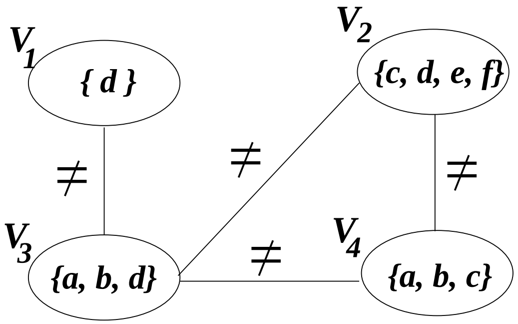
– Perform a faster search.

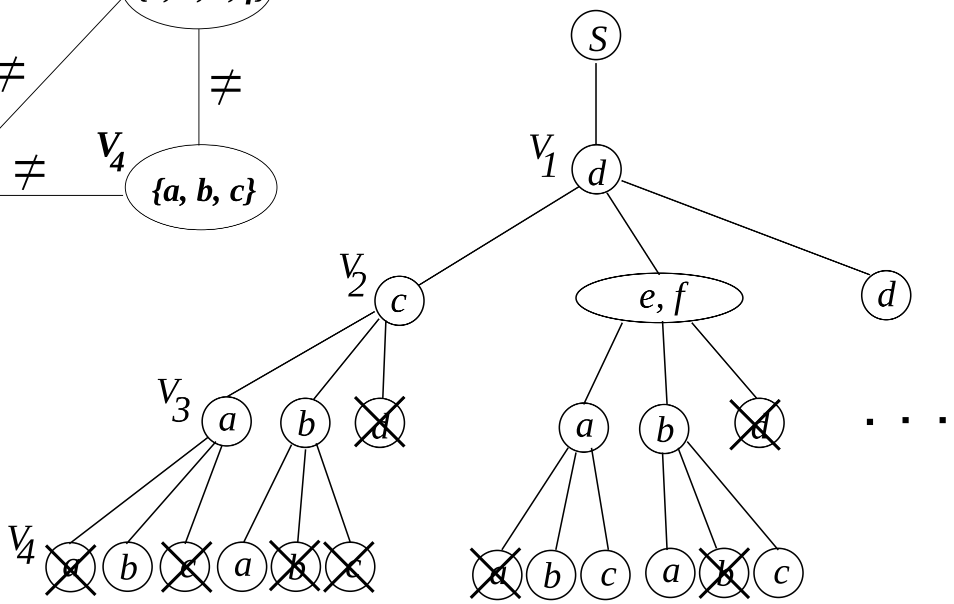


Interchangeability

— Reduces search space

— Creates **bundled** solutions





V_1 $\boxed{(d)}$

V_2 $\boxed{(e, f)}$

V_3 $\boxed{(a)}$

V_4 $\boxed{(b, c)}$

Advantages of Bundling

Bundling partitions the solution space.

– Compact representation of the solution space

– Each bundle is a set of robust solutions

Static or Dynamic Interchangeability?

Static bundling NI_C [Haselböck 1993]

– Before search, find interchangeable values

– Store interchangeable values for duration of search

Dynamic bundling (using NPI) [Freuder 1991]

– Re-compute interchangeable values at each step of search

– Exploit values that become interchangeable **during** search

– Too expensive to be practical? [Meseguer 1989]

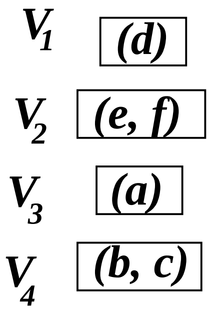
Evaluation criteria

– Nodes Visited `NV`

– Constraints Checked `CC`

– Size of Solution Bundles `SB`

Empirical tests also measured CPU Time.



Dynamic Bundling: DNPI

Introducing: Dynamic Neighborhood Partial Interchangeability

- **Guaranteed** no more expensive than forward checking
- **Guaranteed** no more expensive than static interchangeability
- **Guaranteed** to produce larger solution bundles static interchangeability

Guarantees hold for interchangeability integrated into search for all solutions with a static variable and value ordering.

Theoretical Comparisons

Number of Nodes Visited

$$\mathrm{FC} \;\overset{\geqslant}{\overline{\hspace{4em}}}\; \mathrm{NI}_C \;\overset{\geqslant}{\overline{\hspace{4em}}}\; \mathrm{DNPI}$$

Number of Constraints Checks

$$\mathrm{FC} \;\overset{\mathrm{NI}_C}{\overset{\geqslant}{\overline{\hspace{4em}}}}\; \mathrm{DNPI}$$

Size of Solution Bundles

$$\mathrm{FC} \;\overset{\leqslant}{\overline{\hspace{4em}}}\; \mathrm{NI}_C \;\overset{\leqslant}{\overline{\hspace{4em}}}\; \mathrm{DNPI}$$

Search Improvement Techniques: Summary

Lookahead filtering:

— Forward checking (FC)

— Maintaining arc consistency (MAC)

Dynamic variable-value ordering:

— Static Least Domain (`SLD`)

— Dynamic Least Domain (`DLD`)

— Promise variable-value ordering (`promise`)

— **New:** dynamic variable-bundle ordering (`LD-MB`)
Choose largest bundle in variable with smallest domain.

Exploitation of interchangeability:

— Static interchangeability (NI_C)

— Dynamic interchangeability (DNPI)

Search strategies: Hybrids

– **Hybrids with FC:**

$$\left\{ \text{——}, \mathrm{NI}_C, \mathrm{DNPI} \right\} \times \mathrm{FC} \times \left\{ \texttt{SLD}, \texttt{DLD}, \texttt{promise}, \texttt{LD-MB} \right\}$$

– **Hybrids with MAC:**

$$\left\{ \mathrm{DNPI} \right\} \times \mathrm{MAC} \times \left\{ \texttt{SLD}, \texttt{DLD}, \texttt{LD-MB} \right\}$$

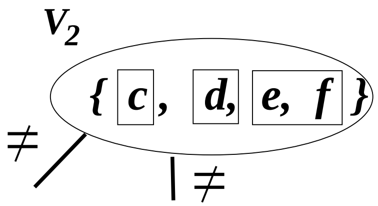
Empirical Evaluations

– When computationally feasible, find all solutions,
Otherwise, for a first solution.

– Toy problems, puzzles, real-world application (PARC)

– Random problems, varying

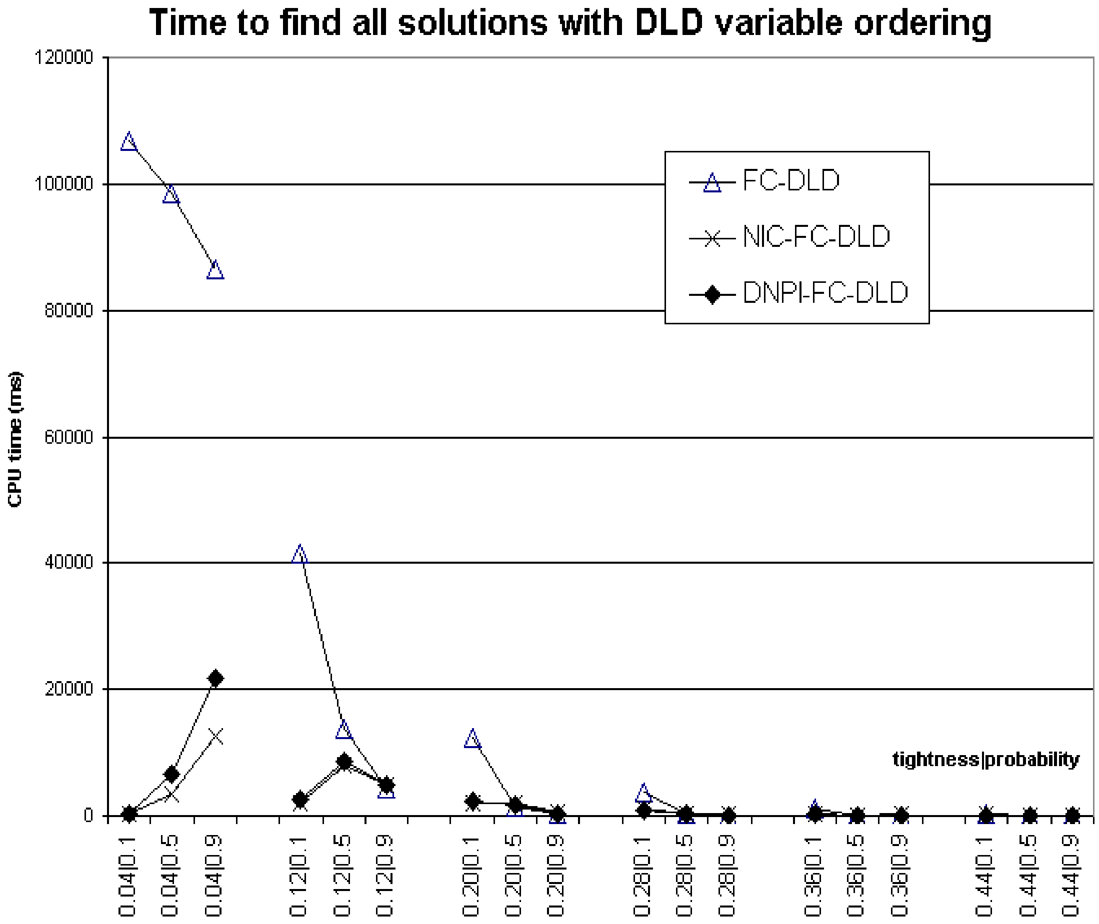
* tightness: loose $\cdots$ tight problems
* density: sparse $\cdots$ dense problems
* Level of interchangeability
Induced Domain Fragmentation `IDF` [Zou et al. 2001]



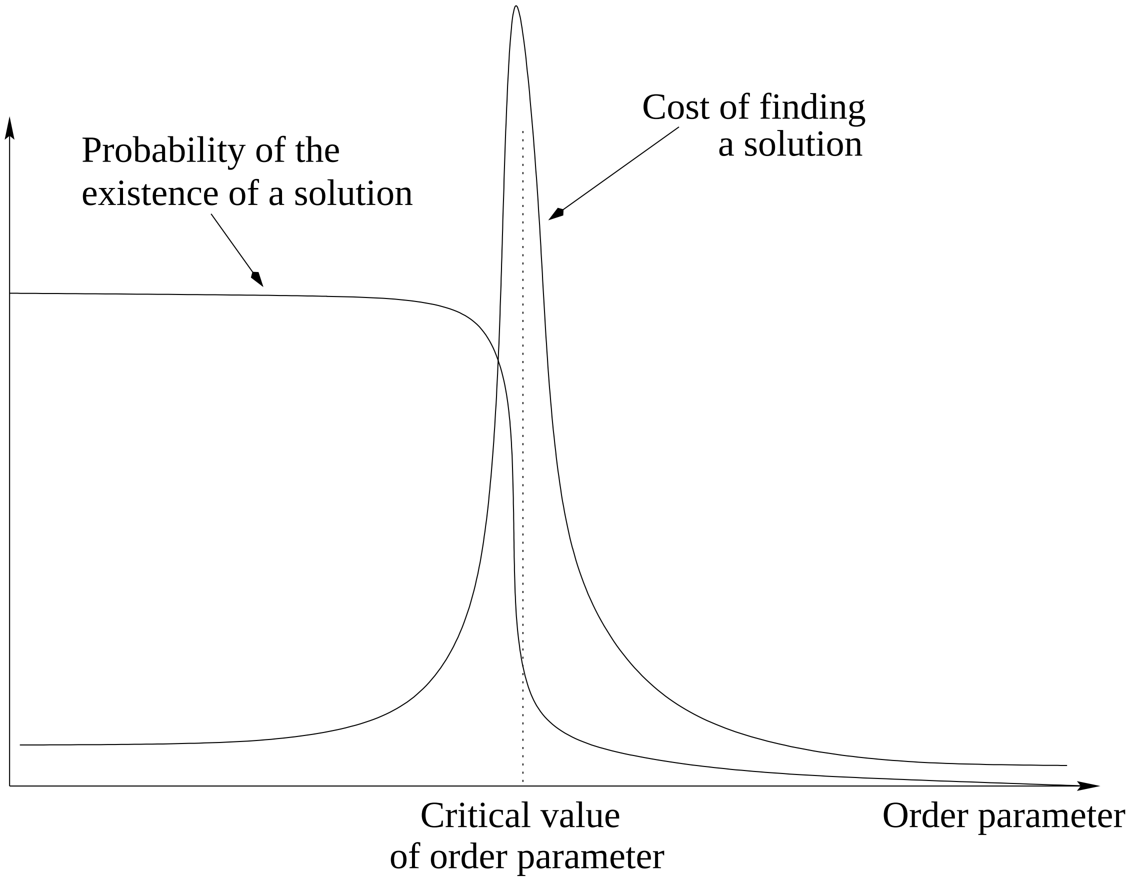
Hybrids: when are they profitable?

Int.	Lookahead	Ordering	Profitable?
DNPI	FC	`SLD`	yes, guaranteed theoretically **(!)** for all solutions
NI_C	FC	`SLD`	sometimes
DNPI	FC	`DLD`	yes **(!)**
DNPI	FC	`promise`	yes, for one solution
DNPI	FC	`LD-MB`	yes for one solution **(!)**
DNPI	MAC	`SLD`	yes
DNPI	MAC	`DLD`	no **(!)**

Evidence: All Solutions (varying: tightness, density)

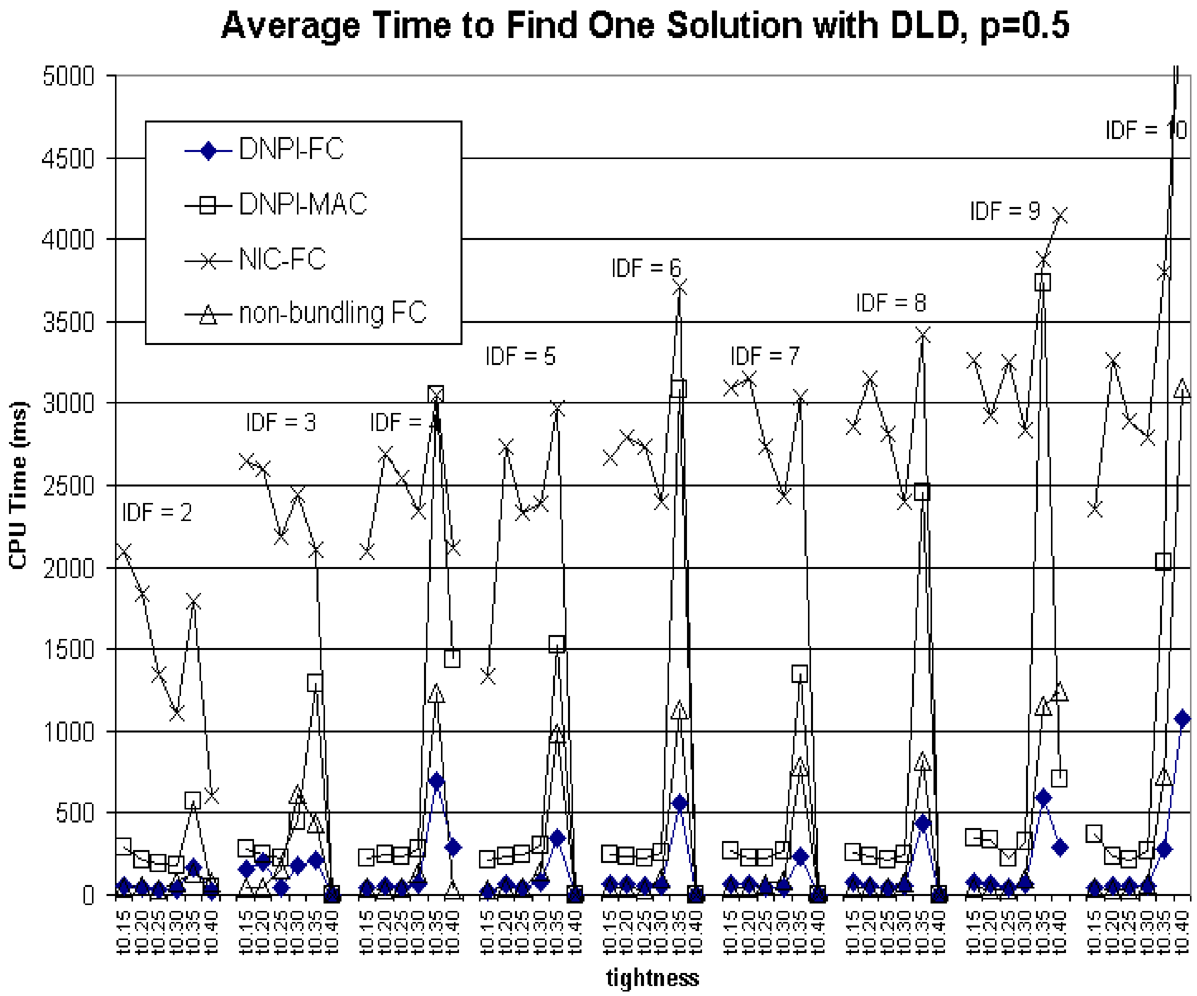


Phase Transition

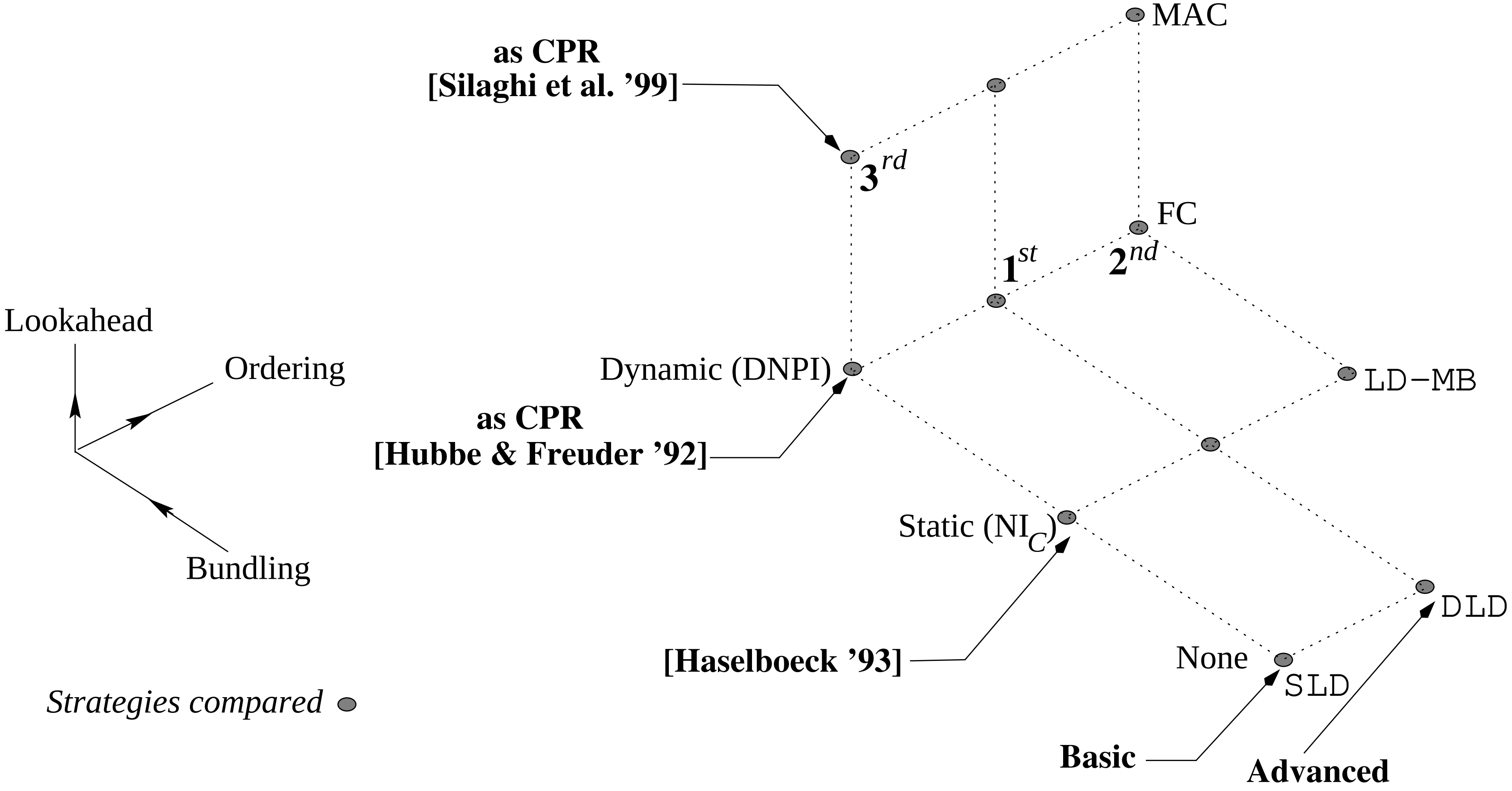


[Cheeseman *et al.* 1991]

Evidence: Phase Transition (varying: tightness, IDF)



Ranking of Best Strategies



DNPI-FC-DLD: the Best Overall

Empirically better than all others when:

– Toy problems, puzzles, real-world application

– Finding one solution and all solutions

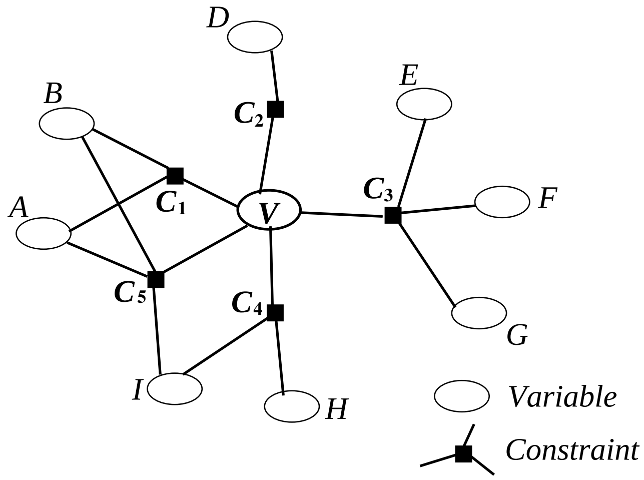
– Random problems with varying tightness, density, `IDF`

– **Problems residing in phase transition**

Novelty: reduce phase transition while finding multiple robust solutions

Extending Interchangeability: Non-binary CSPs

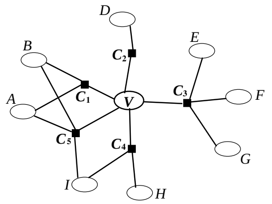
– Real problems are more naturally modeled with non-binary constraints. E.g., tables in relational databases.



– Research on non-binary problems still at infancy stage (no standard FC yet).

– DNPI easily extends to non-binary problems.

Dynamic Bundling for Non-binary CSPs



A	B	D	E	F	G	H	I	V
(1)	(3)	(4 5 6)	(1)	(3)	(1)	(2 4)	(1)	(5 6)
(1)	(3)	(1 2)	(1)	(3)	(1)	(4)	(1)	(2)
(1)	(3)	(1 2)	(1)	(3)	(1)	(1 2 3)	(1)	(1 2)
(1)	(3)	(4 5 6)	(1)	(3)	(1)	(1 2 3 4)	(3)	(5 6)
(1)	(3)	(4 5 6)	(1)	(3)	(3)	(1 2 3 4)	(3)	(5 6)
(1)	(3)	(4 5 6)	(1)	(3)	(3)	(2 4)	(1)	(5 6)
(1)	(3)	(1 2)	(1)	(3)	(3)	(1 2 3 4)	(1)	(2)
(1)	(3)	(1 2)	(1)	(2)	(2)	(1 2 3)	(1)	(1)
(1)	(3)	(4 5 6)	(1)	(2)	(1 3)	(1 2 3 4)	(3)	(5 6)
(1)	(3)	(4 5 6)	(1)	(2)	(1 3)	(2 4)	(1)	(5 6)
(1)	(3)	(1 2)	(1)	(2)	(1 3)	(4)	(1)	(2)
(1)	(3)	(1 2)	(1)	(2)	(1 3)	(1 2 3)	(1)	(1 2)
(1)	(3)	(4 5 6)	(2)	(2)	(1 3)	(1 2 3 4)	(3)	(5 6)
(1)	(3)	(4 5 6)	(2)	(2)	(1 3)	(2 4)	(1)	(5 6)
(1)	(3)	(1 2)	(2)	(2)	(1 3)	(1 2 3 4)	(1)	(2)
(1)	(3)	(4 5 6)	(3)	(2)	(1)	(1 2 3 4)	(3)	(5 6)
(1)	(3)	(4 5 6)	(3)	(2)	(1)	(2 4)	(1)	(5 6)
(1)	(3)	(1 2)	(3)	(2)	(1)	(1 2 3 4)	(1)	(2)
(1)	(2)	(4 5 6)	(1)	(3)	(1)	(1)	(2)	(4)
(1)	(2)	(4 5 6)	(1)	(3)	(1)	(1 2 3)	(1)	(4)
(1)	(2)	(4 5 6)	(1)	(2)	(1 2 3)	(1)	(2)	(4)
(1)	(2)	(4 5 6)	(1)	(2)	(1 2 3)	(1 2 3)	(1)	(4)

Dynamic Bundling on Non-binary CSPs

		Finding all solutions					
		NV		CC		Time	
$t \setminus p$		**0.2**	**0.5**	**0.2**	**0.5**	**0.2**	**0.5**
0.10	FC-SLD	995887	33200	5077855	974843	498831	89282
	DNPI-FC-SLD	452349	28957	4755347	970641	156106	47965
	FC-DLD	793922	17983	3945387	624972	419552	47109
	DNPI-FC-DLD	283744	14318	3687240	624531	101375	27175
0.20	FC-SLD	109453	1590	951677	175151	71019	11397
	DNPI-FC-SLD	75417	1588	936371	175147	36768	7215
	FC-DLD	69579	561	667370	97101	35710	6571
	DNPI-FC-DLD	41236	560	658819	97106	26249	4297
0.30	FC-SLD	9387	202	196539	36703	10326	2682
	DNPI-FC-SLD	8588	202	194580	36703	7076	1670
	FC-DLD	3791	70	108778	24504	5071	1841
	DNPI-FC-DLD	3323	70	108689	24504	3801	1312
0.40	FC-SLD	1774	63	74466	20021	3066	1499
	DNPI-FC-SLD	1738	63	74407	20021	2520	997
	FC-DLD	521	30	40500	14064	1647	1024
	DNPI-FC-DLD	511	30	40496	14064	1319	528

Results: Dynamic Bundling is Worthwhile!

Cost-wise beneficial (almost always):

– Never takes more constraint checks than FC

– Never visits more nodes than FC

– Almost always takes less time than FC

Works very well with dynamic variable ordering:

– Not provable theoretically, but empirically obvious

– DNPI with `DLD` best of all combinations

Remains superior in more difficult situations:

– Finding one solution

– Puzzles

– Phase transition

– No interchangeability in problem

Better with FC than with MAC (an overkill)

Future Work

Applications in:

– Relational Database

– AI Planning

– Human Computer Interaction

Broader symmetries:

– Functional interchangeability [Freuder 1991, Benhamou 1994]

– Continuous CSPs

– Relationship to backbone and SAT

Others:

– Random generator that guarantees solutions

– Phase transitions in non-binary CSPs