Reformulating Constraint Satisfaction Problems with Application to Geospatial Reasoning

Ken Bayer

Constraint Systems Laboratory
Department of Computer Science & Engineering
University of Nebraska-Lincoln

Supported by NSF CAREER award #0133568 & AFOSR grants FA9550-04-1-0105, FA9550-07-1-0416

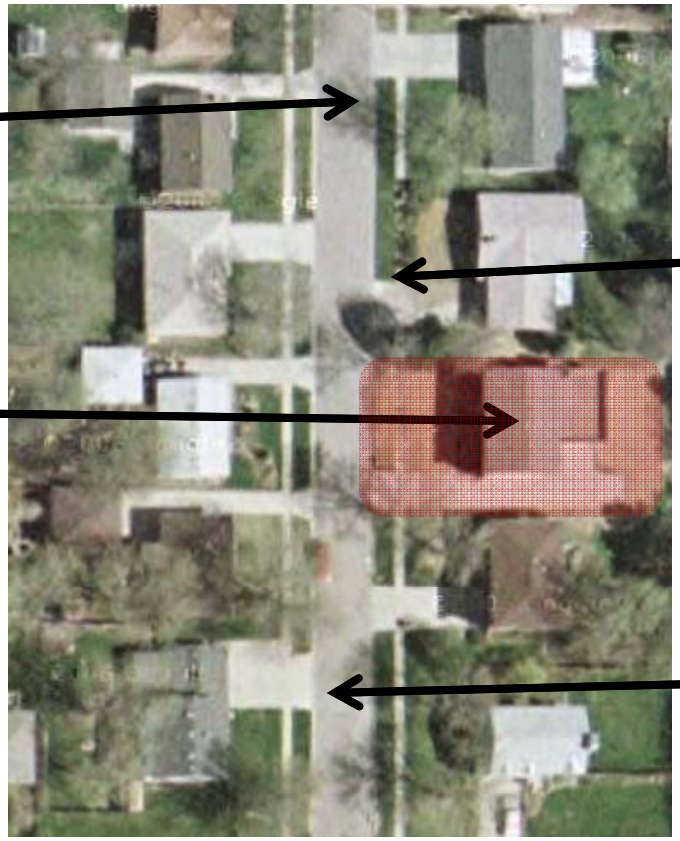
Main contributions

1. Four new reformulation techniques for CSPs
   - Query reformulation
   - Domain reformulation
   - Constraint relaxation
   - Reformulation via symmetry detection
2. BID as a CSP [Michalowski & Knoblock, AAAI 05]
   - Improved constraint model
   - Showed that original BID is in **P**
   - Custom solver
3. Application of the reformulations to BID problem

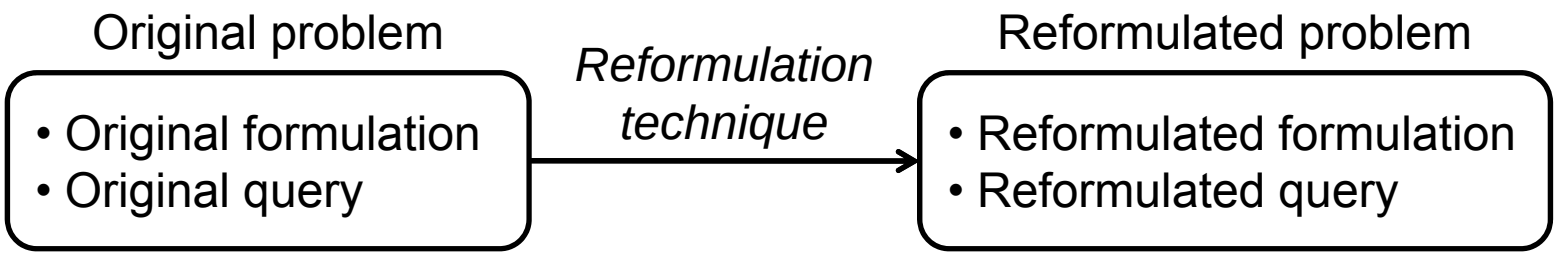
Outline

- **Background**
- BID model & custom solver
- Reformulation techniques
  - Description
  - General use in CSPs
  - Application to BID
  - Evaluation on real-world BID data
- Conclusions & future work

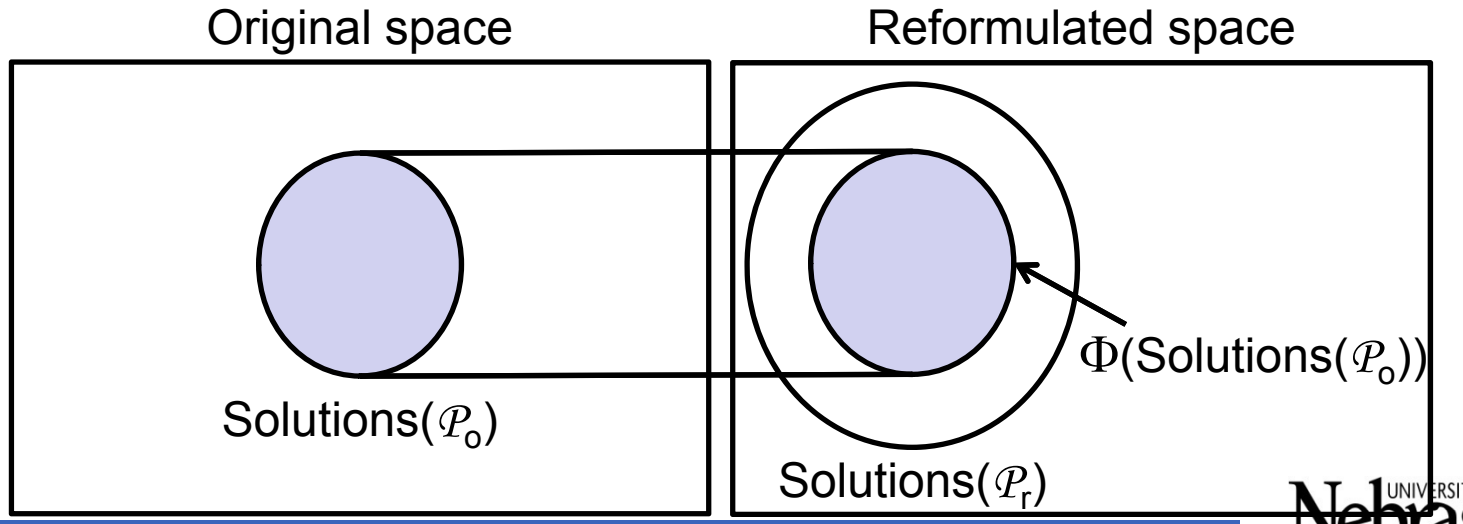
Motivation: finding my house



Abstraction & Reformulation

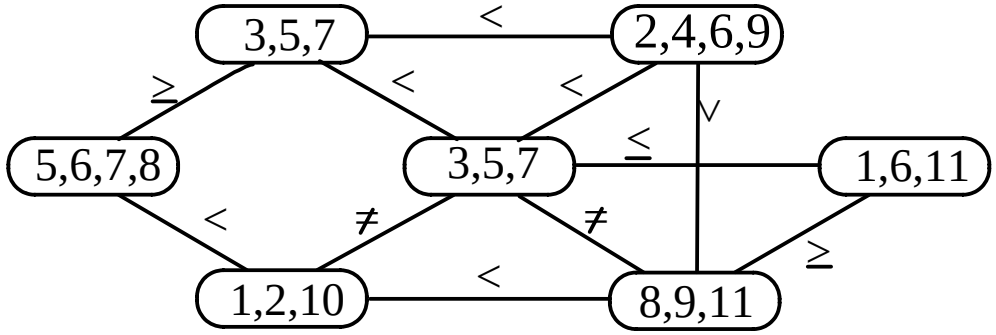


… may be an approximation



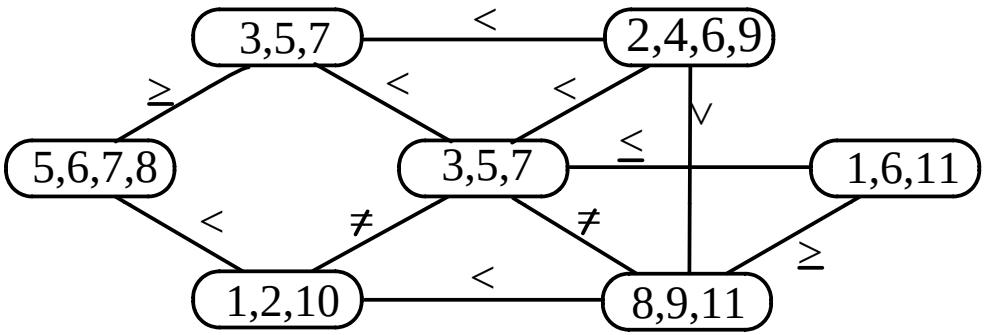
Constraint Satisfaction Problems

- Formulation: $\mathcal{F} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$
  - $\mathcal{V}$ = set of variables
  - $\mathcal{D}$ = set of their domains
  - $\mathcal{C}$ = set of constraints restricting the acceptable combination of values for variables
- Query: All consistent solutions, a single solution, *etc.*
- Solved with
  - Constraint propagation
  - Search

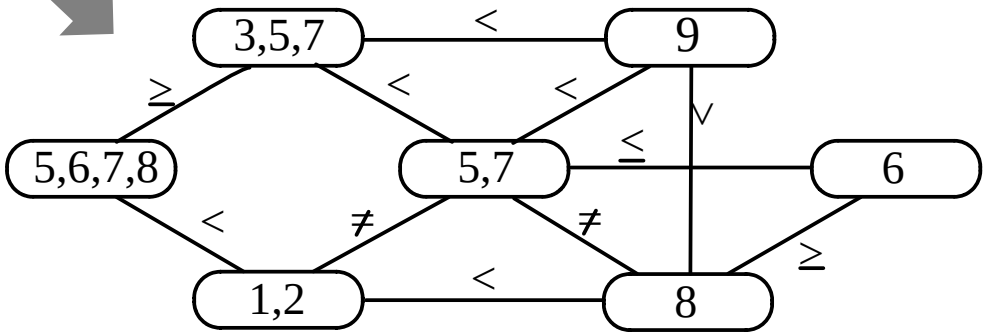


Constraint propagation

Remove values that cannot appear in **any** solution

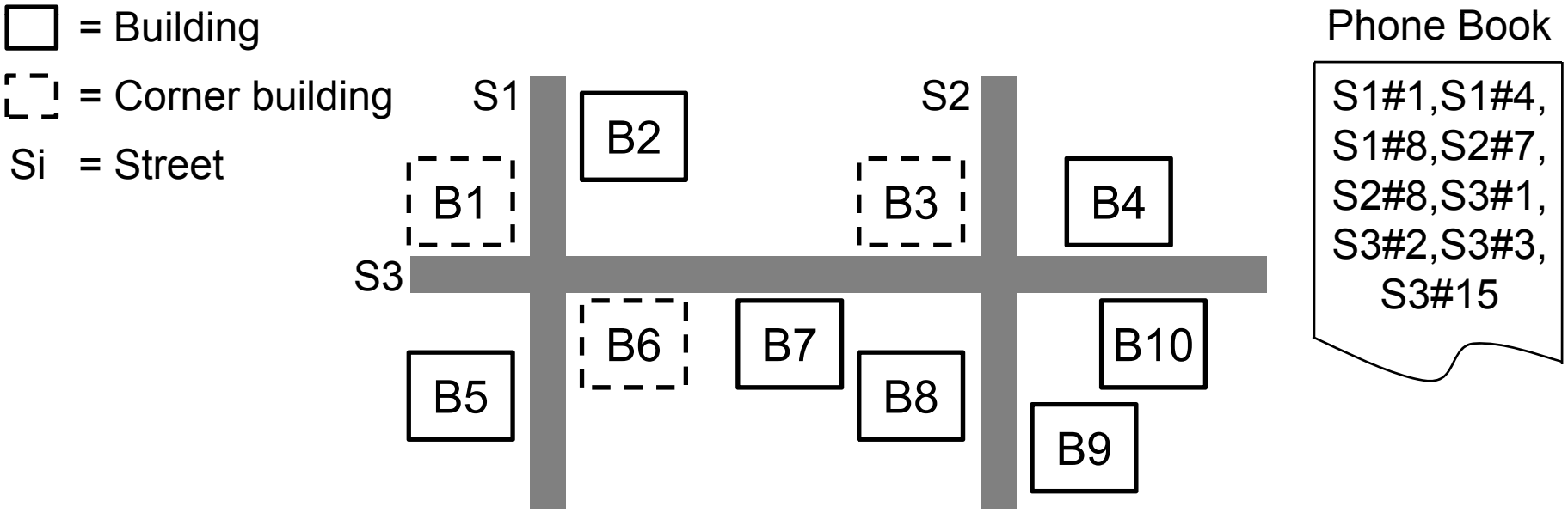


Arc consistency



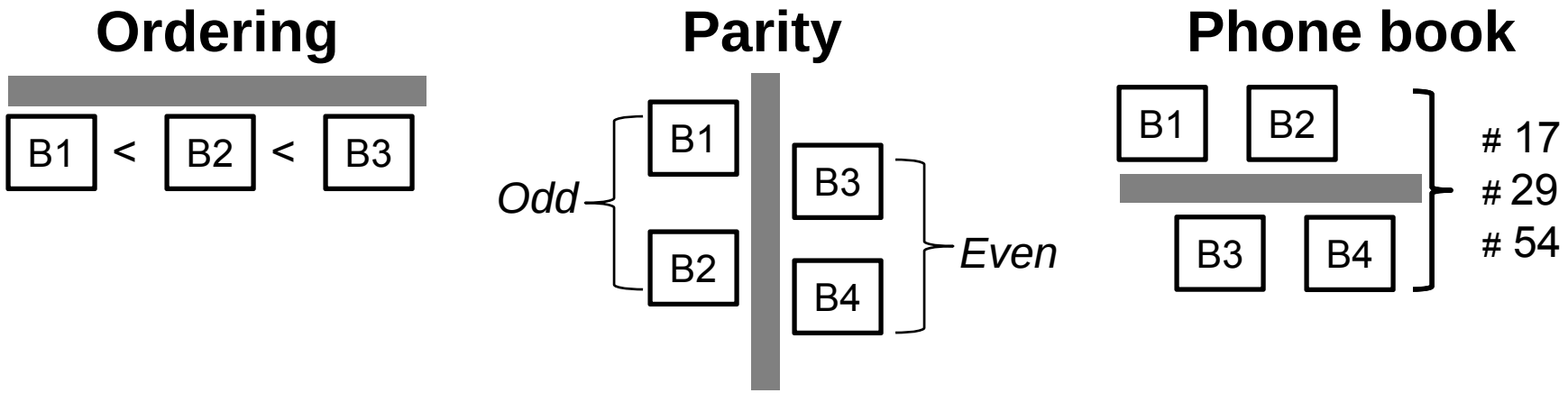
Building Identification (BID) problem

- Data input: map layout + phone book
- Basic numbering rules
- Additional information



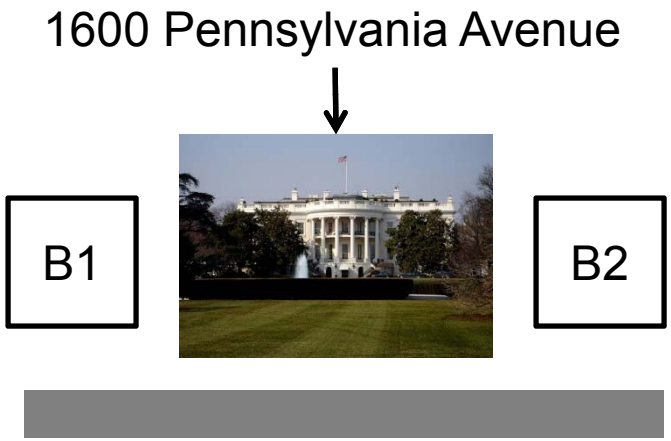
Basic numbering rules

- Ordering: Increasing/decreasing numbers along a street
- Parity: Odd/even numbers on opposing sides of a street
- Phone book: Complete/incomplete
  - Assumption: all addresses in phone-book must be used

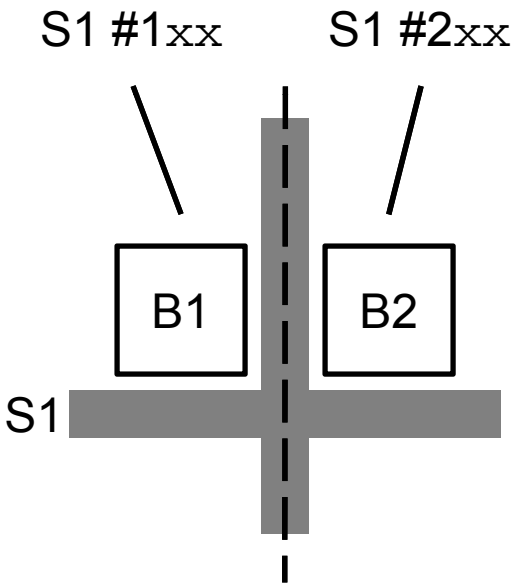


Additional information

Landmarks

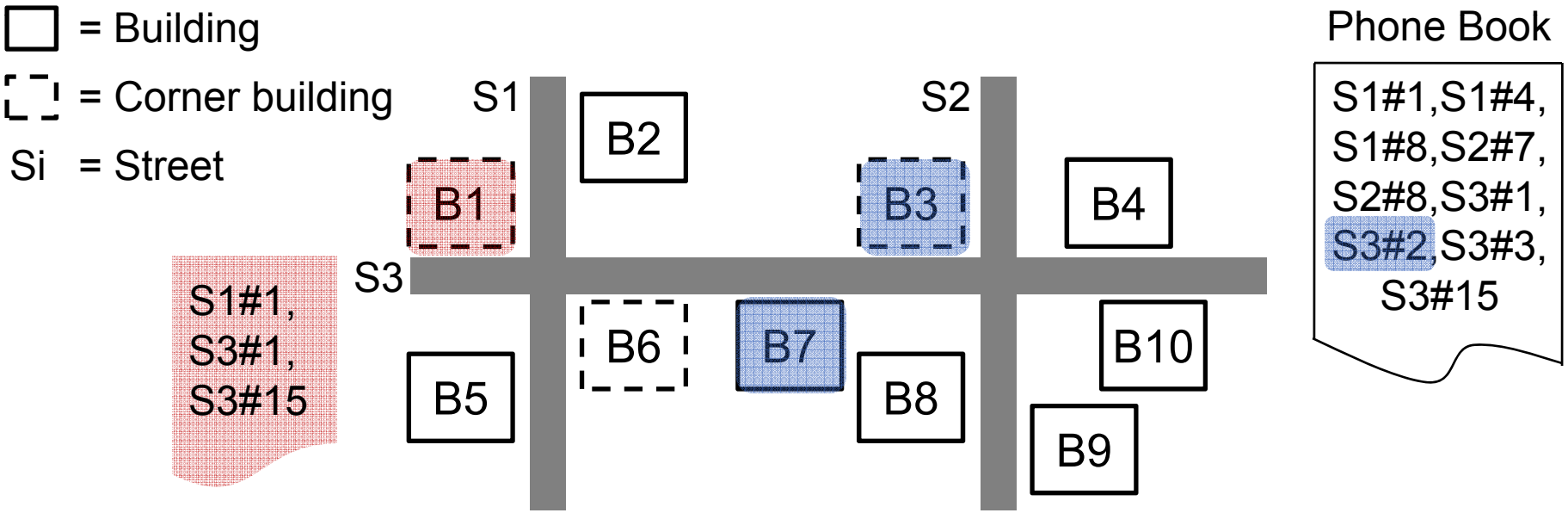


Gridlines



Query

1. Given an address, what buildings could it be?
2. Given a building, what addresses could it have?

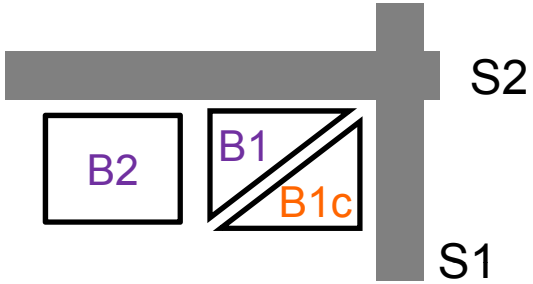


Outline

- Background
- **BID model & custom solver**
- Reformulation techniques
- Conclusions & future work

CSP model: variables

- Orientation variables
  - OddOnNorth, OddOnEast, IncreasingNorth, IncreasingEast
  - Domains: {true, false}
- Corner variables
  - Corner buildings
  - Domains: set of streets
- Building variables
  - Corner & non-corner buildings
  - Domains: set of addresses



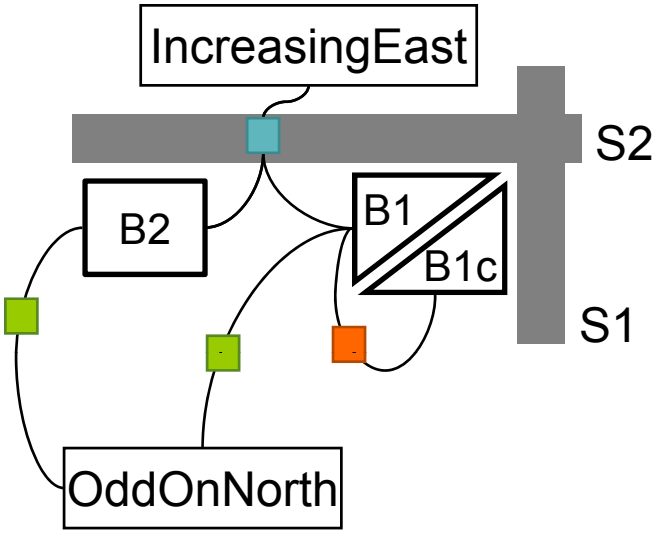
D_{B1c} = {S1, S2}

D_{B1} = {S1#2, S1#4,…, S1#228, S2#5, S2#9,…, S2#25}

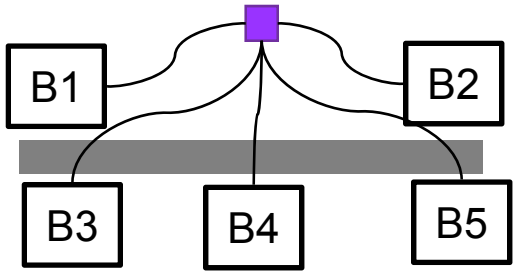
D_{B2} = { S2#5, S2#9,…, S2#25}

CSP model: constraints

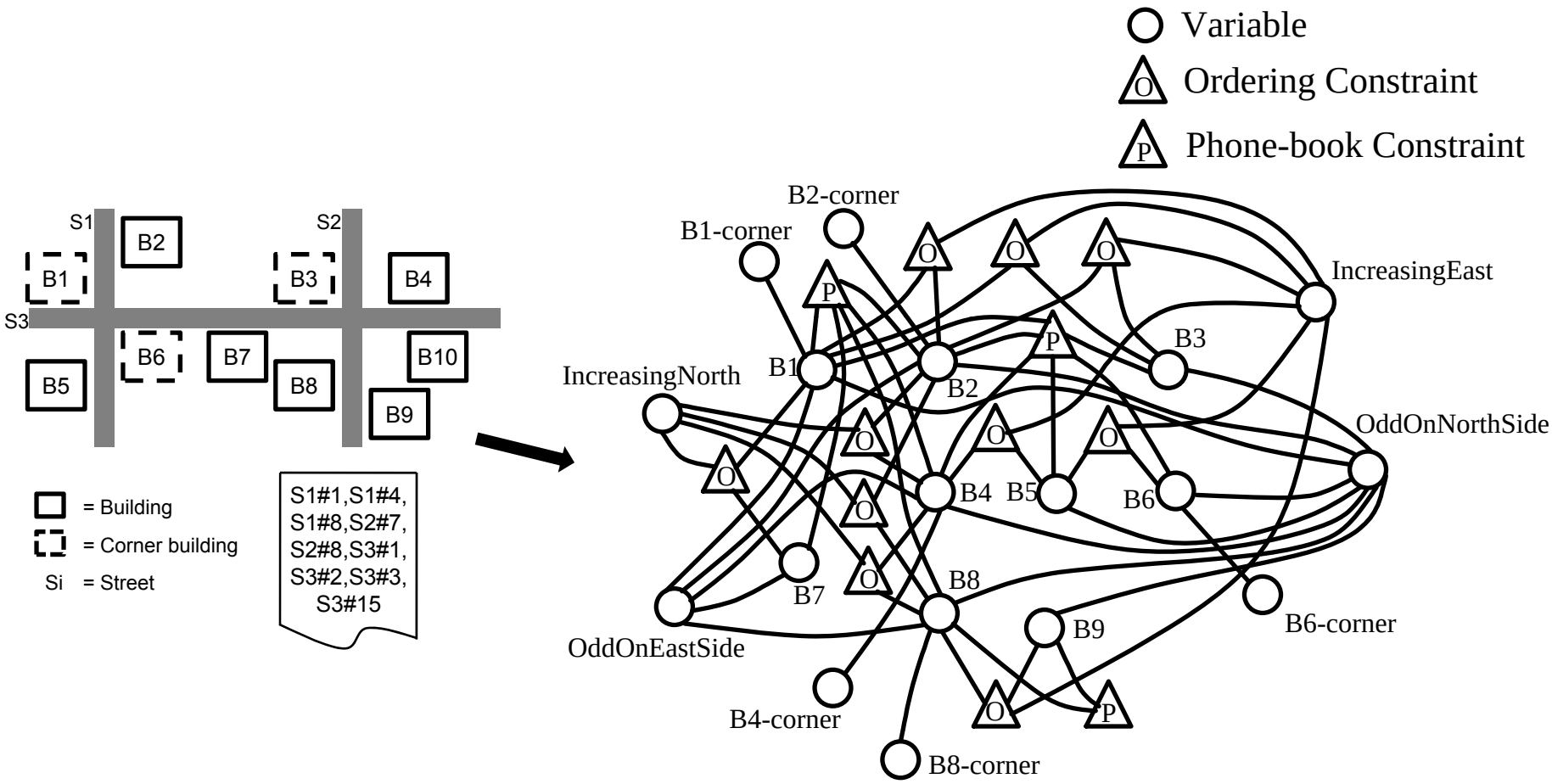
- **Parity constraints**
- **Ordering constraints**
- **Corner constraints**



- **Phone-book constraints**
- Optional: grid constraints



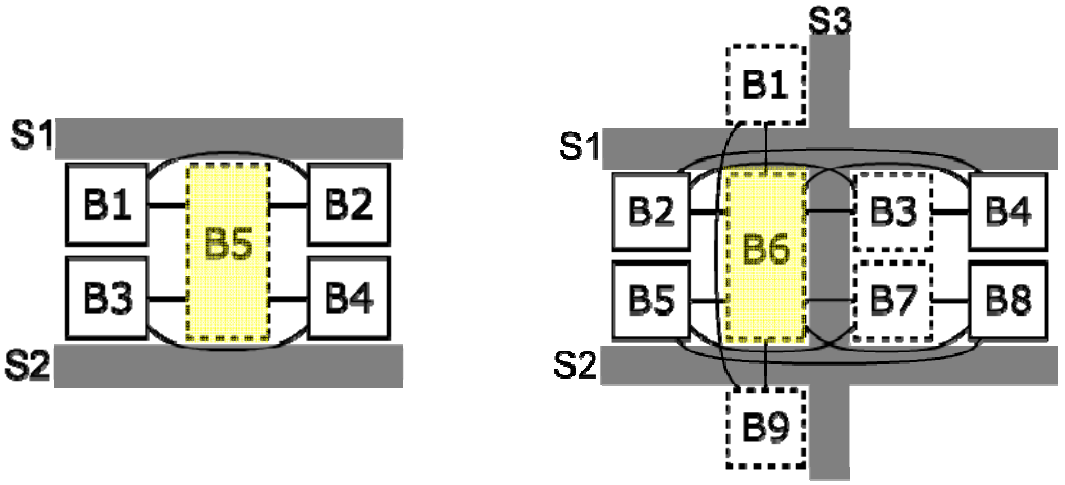
Example constraint network



Special configurations

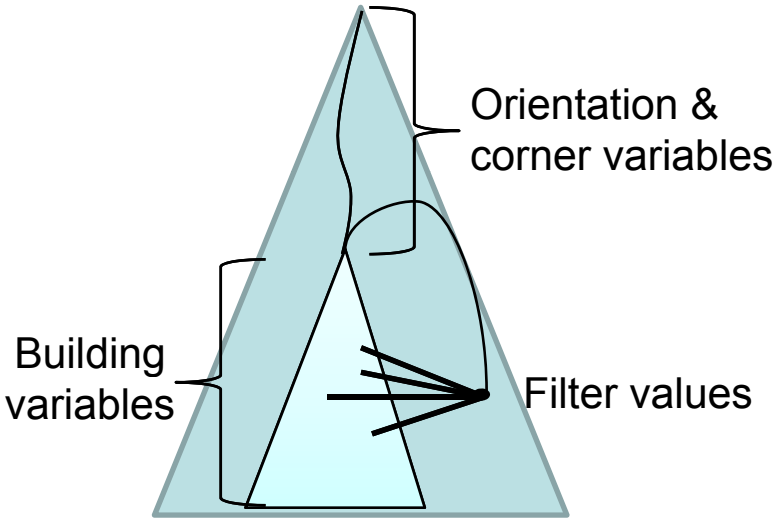
1. Orientations vary per street (e.g., Belgrade)
2. Non-corner building on two streets
3. Corner building on more than two streets

→ All gracefully handled by the model



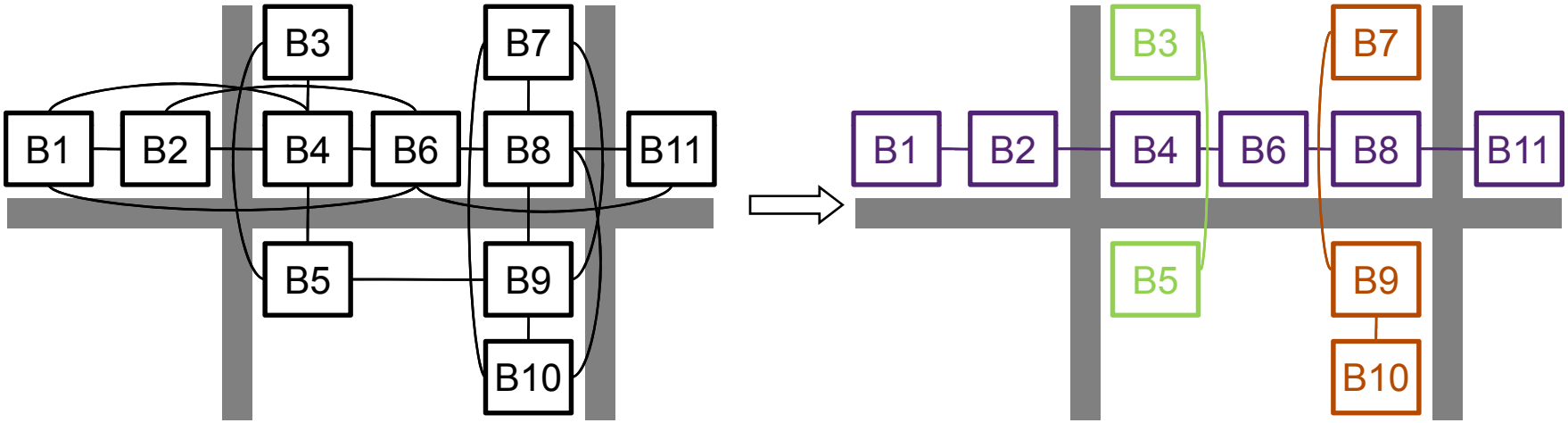
Custom solver

- Backtrack search
- Conflict-directed backjumping
- Forward checking (nFC3)
- Domains implemented as intervals (box consistency)
- Variable ordering
  1. Orientation variables
  2. Corner variables
  3. Building variables
- Backdoor variables
  - Orientation + corner variables



Backdoor variables

- We instantiate **only** orientation & corner variables



- We guarantee solvability **without** instantiating building variables

Features of new model & solver

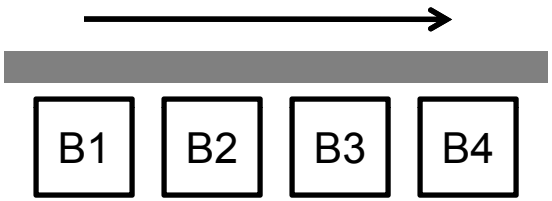
- Model
  - Reflects topology
  - Reduces number of variables
  - Reduces constraint arity
  - Constraints can be declared locally & in restricted ‘contexts’ (feature important for Michalowski’s work)
- Solver
  - Exploits structure of problem
  - Implements domains as possibly infinite intervals
  - Incorporates all reformulations (to be introduced)
- Improvement over previous work [Michalowski+, 05]

Outline

- Background
- BID model & custom solver
- Reformulation techniques
  - **Query reformulation**
  - AllDiff-Atmost & domain reformulation
  - Constraint relaxation
  - Reformulation via symmetry detection
- Conclusions & future work

Query in the BID

- Problem: BID instances have many solutions

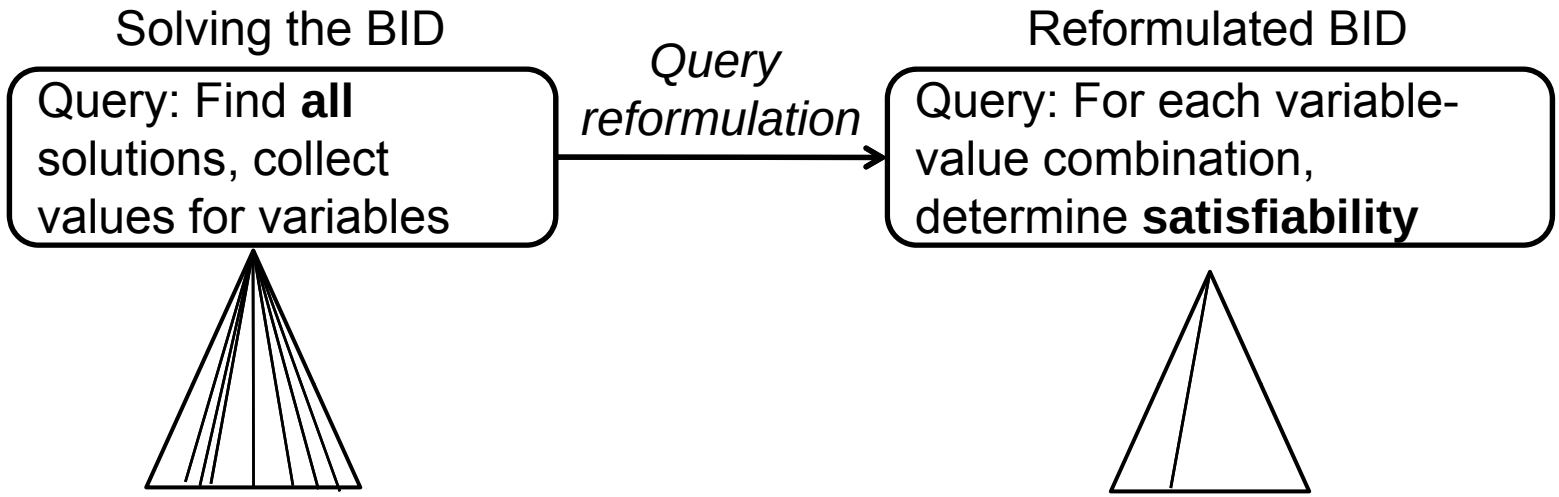


Phone book: {4,8}

B1	B2	B3	B4
2	4	6	8
2	4	8	10
2	4	8	12
4	8	10	12
4	6	8	10
4	6	8	12

We **only** need to know which values appear in ***at least one*** solution

Query reformulation



Original query	Reformulated query
Single counting problem	Many satisfiability problems
All solutions	Per-variable solution
Exhaustive search	One path
Impractical when there are many solutions	Costly when there are few solutions

Evaluations: real-world data from El Segundo

[Shewale]

Case study	Phone book completeness	Number of… buildings	corner bldgs	blocks
NSeg125-c	100.0%	125	17	4
NSeg125-i	45.6%			
NSeg206-c	100.0%	206	28	7
NSeg206-I	50.5%			
SSeg131-c	100.0%	131	36	8
SSeg131-i	60.3%			
SSeg178-c	100.0%	178	46	12
SSeg178-i	65.6%			

Previous work did not scale up beyond 34 bldgs, 7 corner bldgs, 1 block

All techniques tested return same solutions

Evaluation: query reformulation

Incomplete phone book → many solutions → better performance

Case study	Original query	New query [s]
NSeg125-i	>1 week	744.7
NSeg206-i	>1 week	14,818.9
SSeg131-i	>1 week	66,901.1
SSeg178-i	>1 week	119,002.4

Complete phone book → few solutions → worse performance

Case study	Original query [s]	New query [s]
NSeg125-c	1.5	139.2
NSeg206-c	20.2	4,971.2
SSeg131-c	1123.4	38,618.4
SSeg178-c	3291.2	117,279.1

Generalizing query reformulation

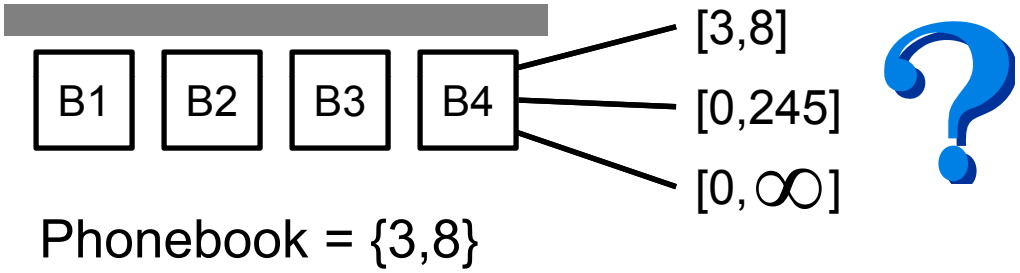
- Relational (i,m)-consistency, R(i,m)C
  - Given m constraints, let s be the size of their scope
  - Compute **all solutions** of length s
  - To generate tuples of length i (i.e., constraints of arity i)
  - Space: $O(d^s)$
- Query reformulation
  - For each combination of values for i variables
  - Try to extend to **one** solution of length s
  - Space: $O(\binom{s}{i} d^i)$, $i < s$
- Per-variable solution computes R(1,$|C|$)C

Outline

- Background
- BID model & custom solver
- Reformulation techniques
  - Query reformulation
  - **AllDiff-Atmost & domain reformulation**
  - Constraint relaxation
  - Reformulation via symmetry detection
- Conclusions & future work

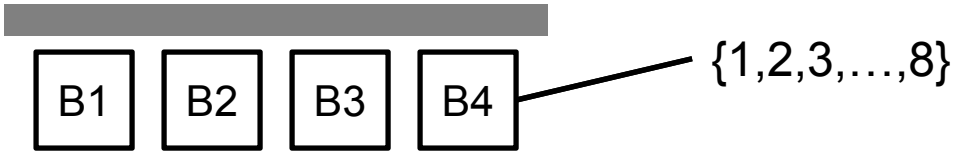
Domain reformulation

- Domains in the BID are large
- Min/max value?



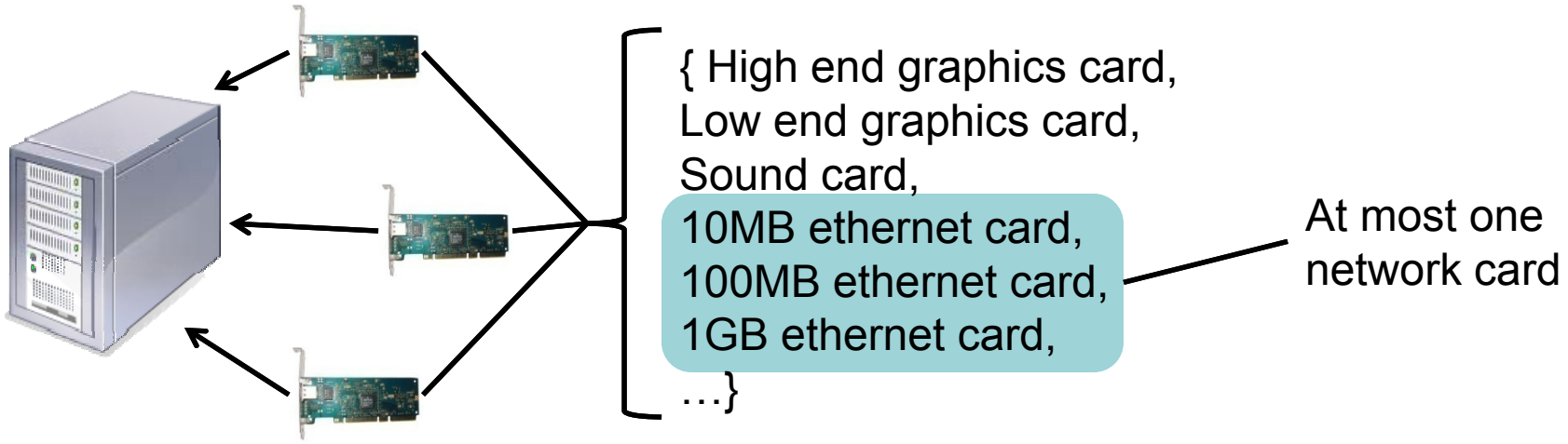
Phonebook = {3,8}

- Enumerate?



AllDiff-Atmost constraint

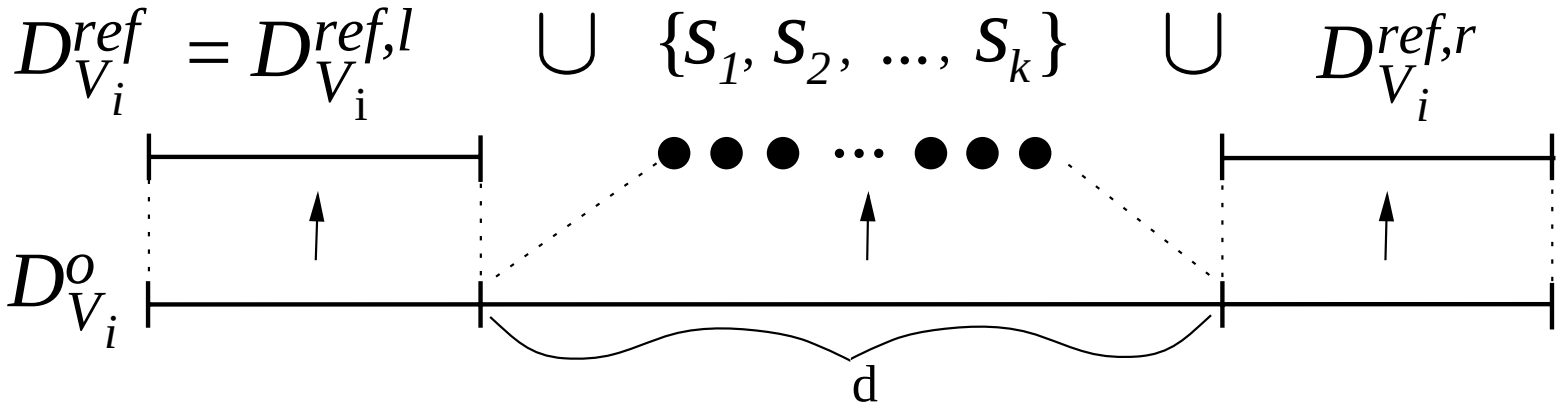
- AllDiff-Atmost($\mathcal{A}$,*k*,*d*)
  - For a set of variables $\mathcal{A}$, ensure that the variables in $\mathcal{A}$ collectively are not assigned more than *k* values from the set *d*



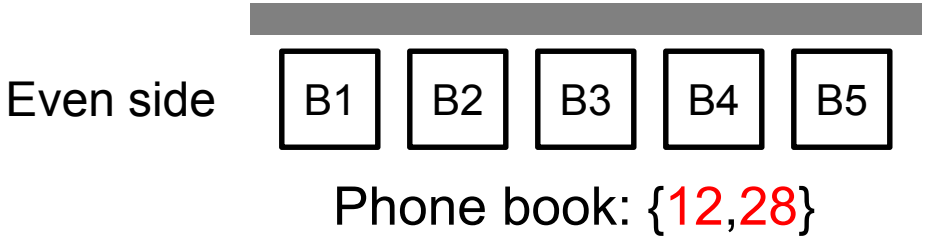
AllDiff-Atmost reformulation

Replaces

- interval *d* of values (potentially infinite)
- with *k* **symbolic values**

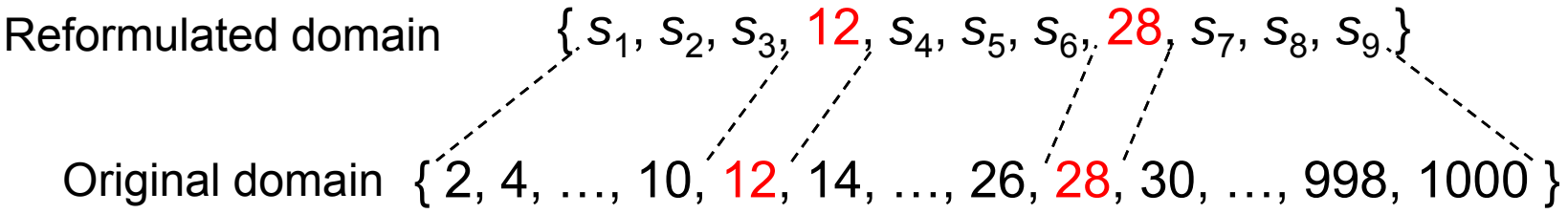


AllDiff-Atmost in the BID



12	**28**	30	32	34
12	14	16	**28**	36
10	**12**	14	20	**28**
2	4	6	**12**	**28**
…	…	**12**	**28**	**...**

- *Cannot use more than*
  - ***3** addresses less than 12*
  - ***3** addresses in (12,28)*
  - ***3** addresses more than 28*



Evaluation: domain reformulation

- Reduced domain size → improved search performance

Case study	Phone-book completeness	Average domain size		Runtime [s]	
		Original	Reformulated	Original	Reformulated
NSeg125-i	45.6%	1103.1	236.1	2943.7	744.7
NSeg206-i	50.5%	1102.0	438.8	14,818.9	5533.8
SSeg131-i	60.3%	792.9	192.9	67,910.1	66,901.1
SSeg178-i	65.6%	785.5	186.3	119,002.4	117,826.7

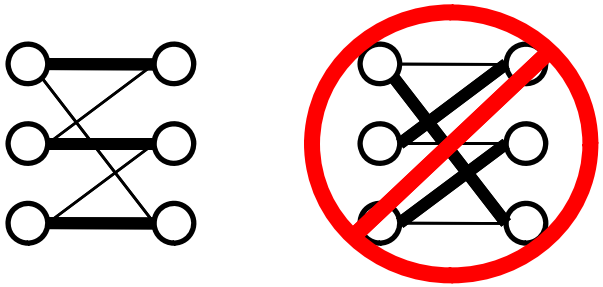
Outline

- Background
- BID model & custom solver
- Reformulation techniques
  - Query reformulation
  - AllDiff-Atmost & domain reformulation
  - **Constraint relaxation**
  - Reformulation via symmetry detection
- Conclusions & future work

Constraint relaxation

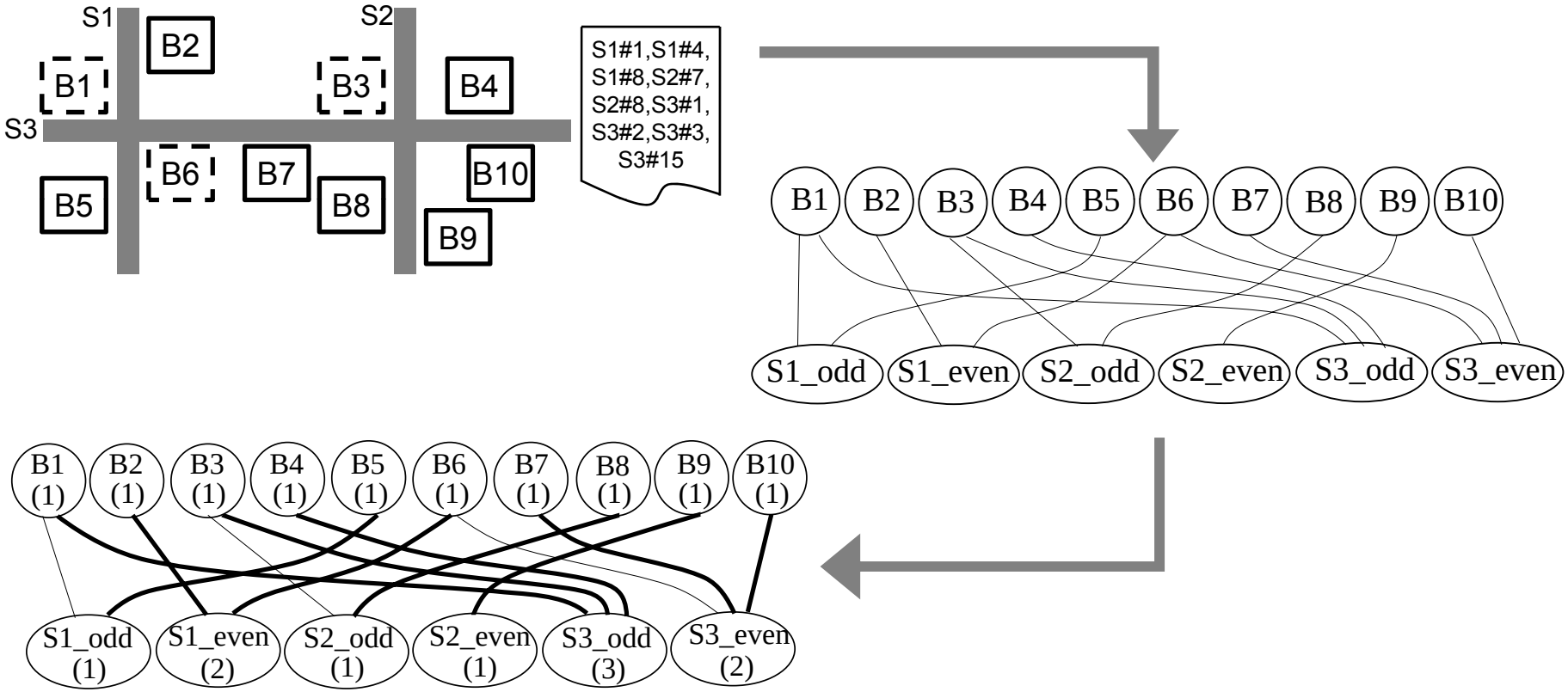
- Resource allocation problems are often matching problems with additional constraints

Additional constraint: We don't want Bob and Charlie both working sales at the same time



BID as a matching problem

- Assume we have no grid constraints



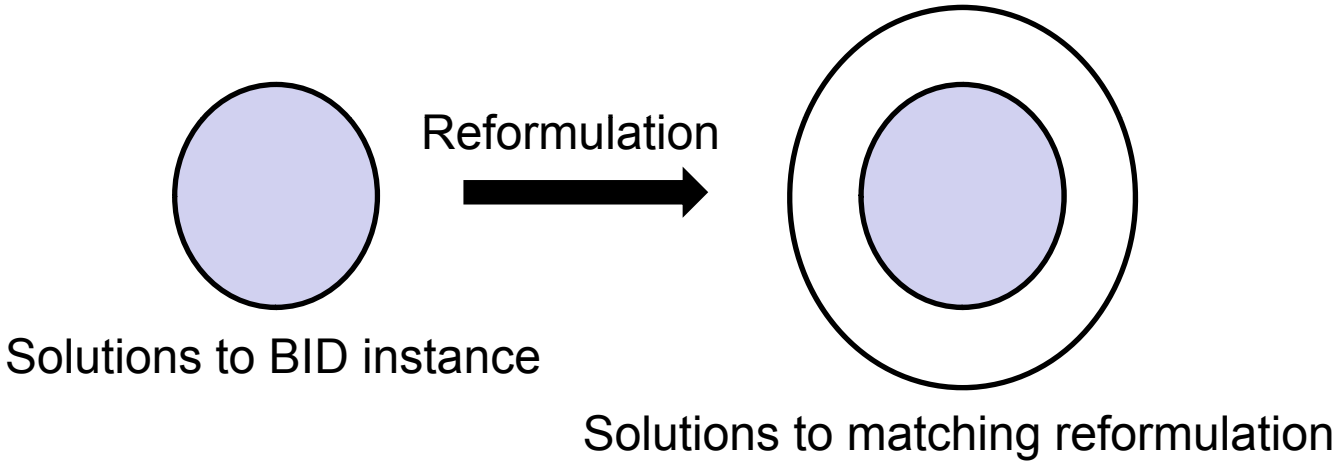
BID w/o grid constraints

- BID instances without grid constraints can be solved in ***polynomial time***

Case study	Runtime [s]	
	BT search	Matching
NSeg125-c	139.2	4.8
NSeg206-c	4971.2	16.3
SSeg131-c	38618.3	7.3
SSeg178-c	117279.1	22.5
NSeg125-i	744.7	2.5
NSeg206-i	5533.8	8.5
SSeg131-i	38618.3	7.3
SSeg178-i	117826.7	4.9

BID w/ grid constraints

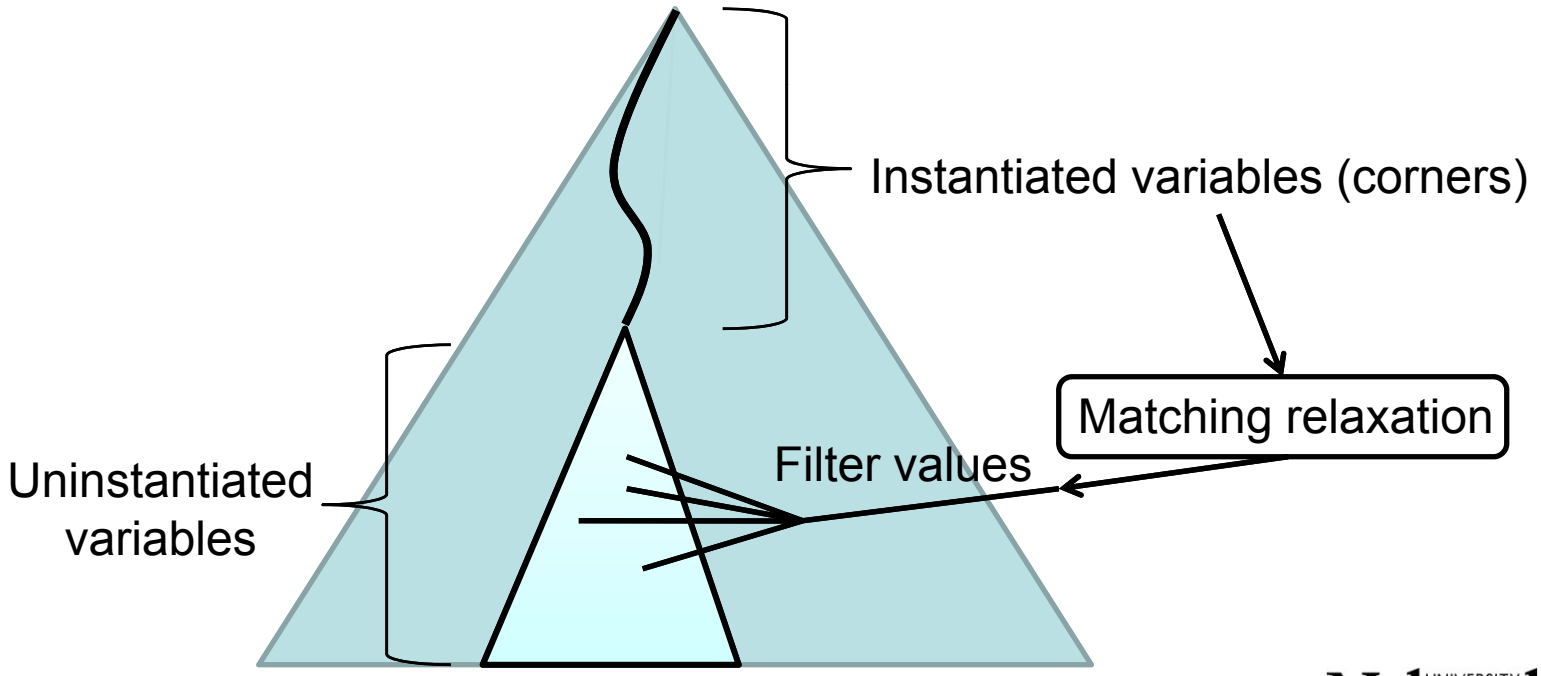
- The matching reformulation is a *necessary approximation*



No solution to matching reformulation → no solution to the original BID

Lookahead: relaxation during search

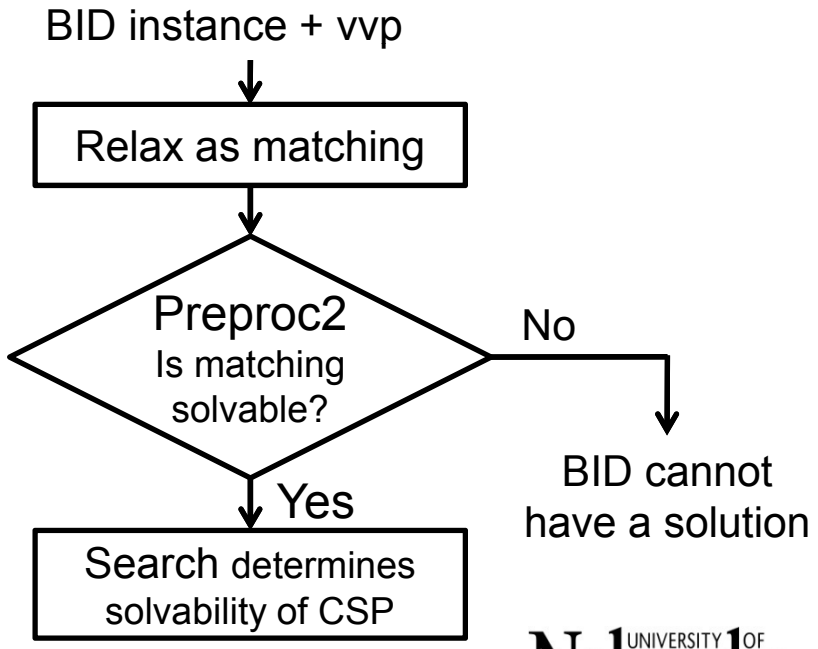
- Filter vvps that do not appear in any maximum matching [Régin, 1994]



Using the relaxation prior to search

- Preproc1
  - Prior to running per-variable solution loop
  - Remove vvps that don't appear in any max. matching [Régin, 1994]

- Preproc2
  - Prior to testing a CSP for solvability
  - Necessary approximation determines unsolvability



Evaluation: constraint relaxation

- Generally, improves performance

Case Study	BT	Preproc2 +BT	% (from BT)	Lkhd +BT	% (from BT)	Lkhd +Preproc1&2 + BT	% (from Lkhd+BT)
NSeg125-i	1232.5	1159.1	**6.0%**	726.6	**41.0%**	701.1	**3.5%**
NSeg206-c	2277.5	614.2	**73.0%**	1559.2	**31.5%**	443.8	**71.5%**
SSeg178-i	138404.2	103244.7	**25.4%**	121492.4	**12.2%**	85185.9	**29.9%**

- Rarely, the overhead exceeds the gains

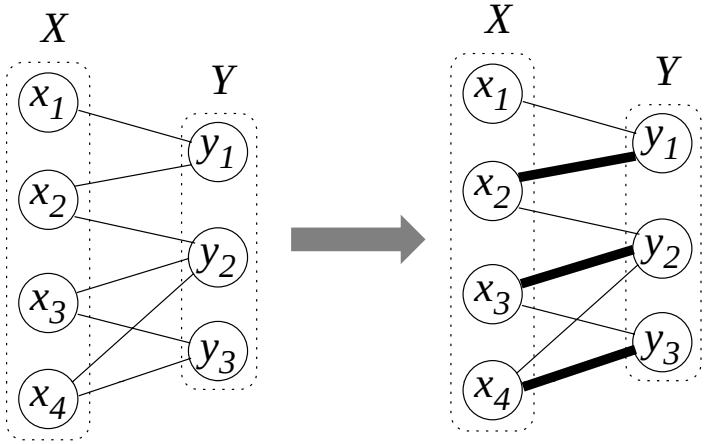
Case Study	BT	Preproc2 +BT	% (from BT)	Lkhd +BT	% (from BT)	Lkhd +Preproc1&2 + BT	% (from Lkhd+BT)
NSeg125-i	100.8	33.2	**67.1%**	**140.2**	-39.0%	39.6	**71.8%**
NSeg206-c	114405.9	114141.3	**0.2%**	107896.3	**5.7%**	**108646.6**	-0.7%

Outline

- Background
- BID model & custom solver
- Reformulation techniques
  - Query reformulation
  - AllDiff-Atmost & domain reformulation
  - Constraint relaxation
  - **Reformulation via symmetry detection**
- Conclusions & future work

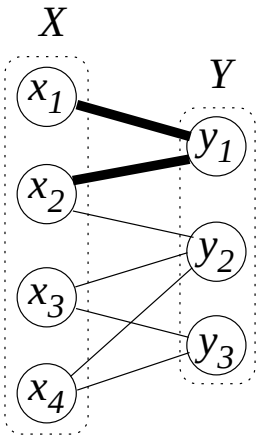
Find all maximum matchings

Find one maximum matching [Hopcroft+Karp, 73]

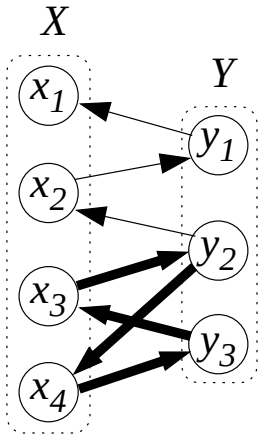


Identify… [Berge, 73]

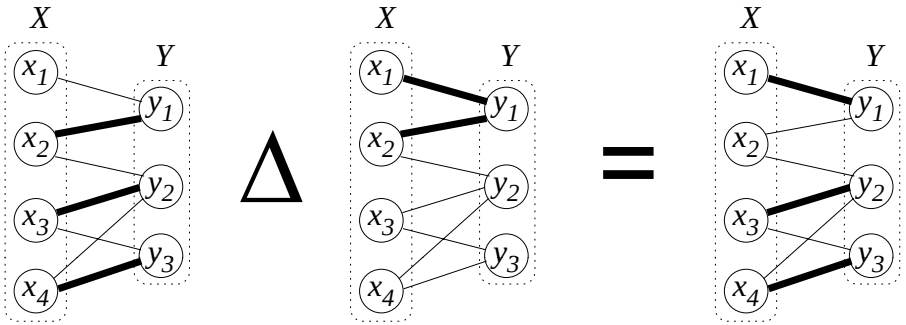
… even alternating paths starting @ free vertex:



… alternating cycles:

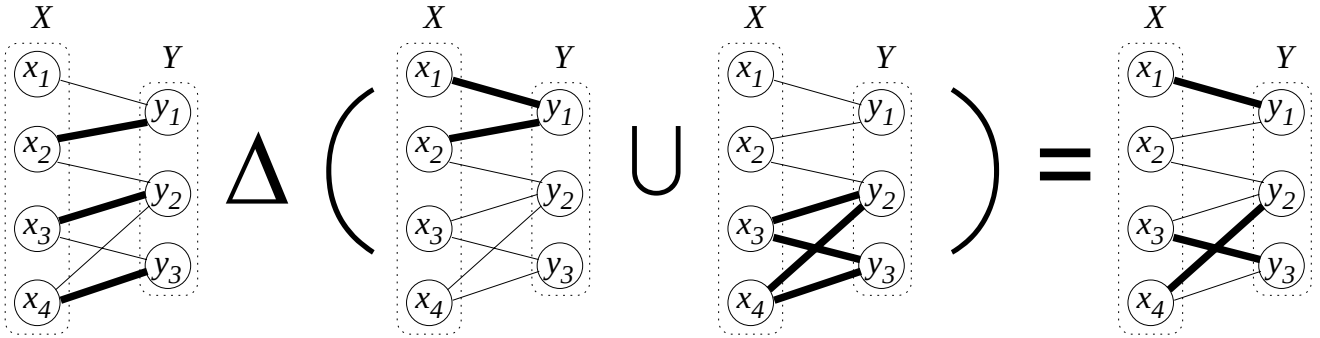


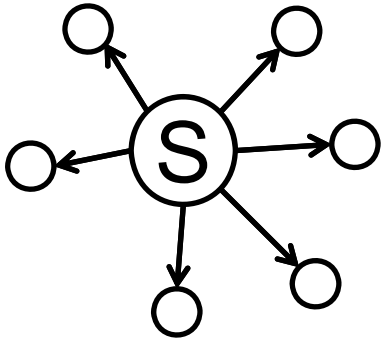
Symmetric maximum matchings



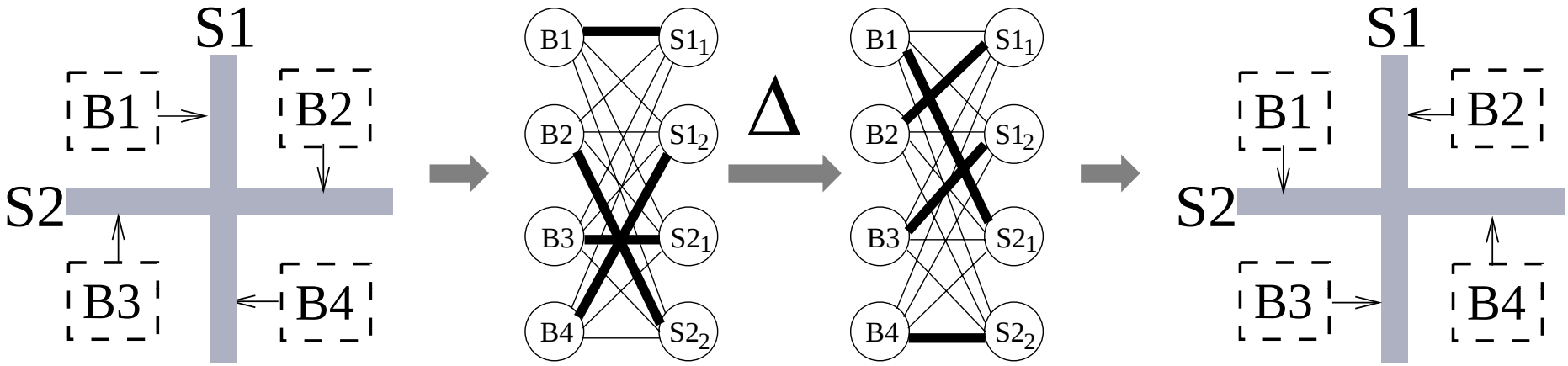
All matchings can be produced using the sets of disjoint alternating paths & cycles

→ Compact representation





Symmetric matchings in BID



- Some symmetric solutions do not break the grid constraints
  - Use symmetry breaking constraints to avoid exploring them
- Some do, we do not know how to use them…

Conclusions

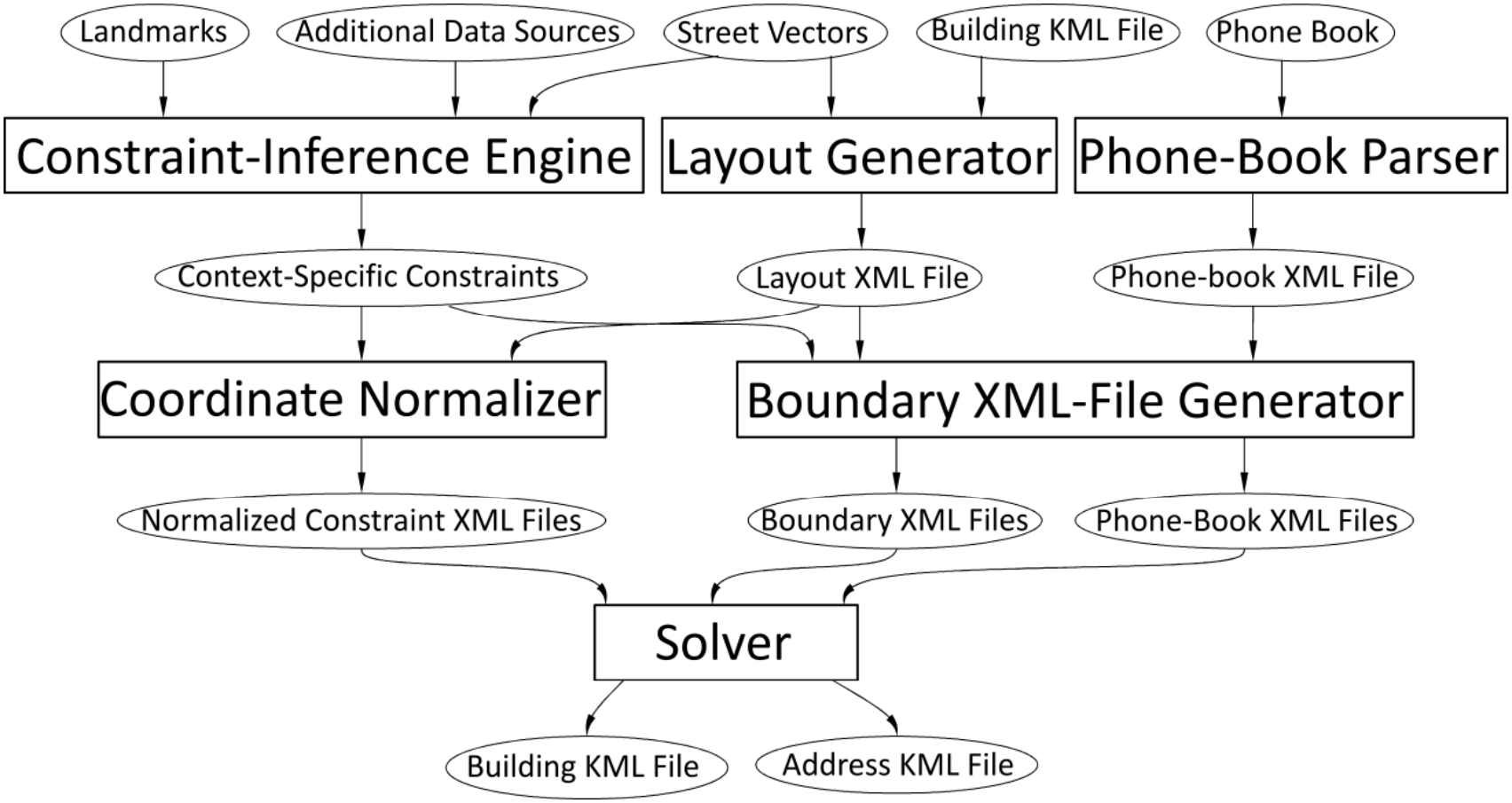
- We proposed four reformulation techniques
- We described their usefulness for general CSPs
- We demonstrated their effectiveness on the BID
- **Lesson: reformulation is an effective approach to improve the scalability of complex systems**

Future work

- Empirically evaluate our new algorithm for relational (*i*,*m*)-*consistency*
- Identify other problems that benefit from a matching relaxation
- Exploit the symmetries we identified
- Enhance the model by inferring more constraints [Michalowski]

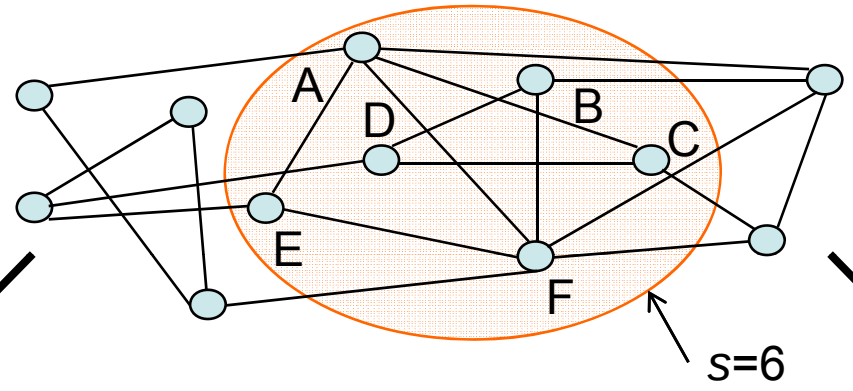
Questions?

Overall architecture



Using query reformulation

Goal: Compute R(2,7)C with i=2, m=7



A	B	C	D	E	F
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...

$RC_{(i,m)}$: Space $O(d^s)$

New algor.: Space $O(\binom{s}{i} d^i)$

Note: $i < s$

A	C
...	...
...	...
...	...

A	B
...	...
...	...
...	...

D	B
...	...
...	...

F	B
...	...
...	...
...	...

E	C
...	...
...	...
...	...