

---

# Reformulating Constraint Satisfaction Problems with Application to Geospatial Reasoning

K. Bayer <sup>1</sup> M. Michalowski <sup>2</sup> B.Y. Choueiry <sup>1,2</sup> C.A. Knoblock <sup>2</sup>

<sup>1</sup> Constraint Systems Laboratory  
University of Nebraska-Lincoln

<sup>2</sup> Information Sciences Institute  
University of Southern California

Supported by NSF CAREER Award #0133568 and  
AFOSR grants FA9550-04-1-0105 and FA9550-07-1-0416

---

*Constraint Systems Laboratory*

UNIVERSITY OF  
**Nebraska**  
Lincoln

# Contributions

---

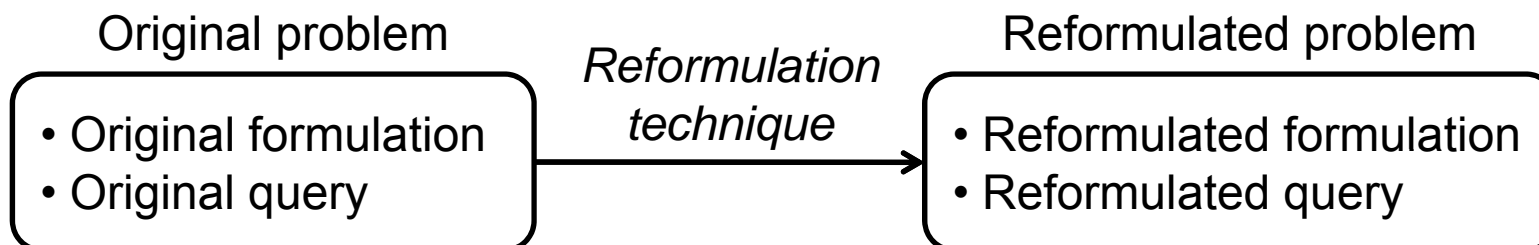
- BID problem as a CSP [Michalowski & Knoblock, AAAI 05]
  - Improved constraint model
  - Showed original BID problem is in **P**
  - Custom solver
- Four new reformulation techniques for CSPs
  1. Query reformulation
  2. Domain reformulation
  3. Constraint relaxation
  4. Reformulation via symmetry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

# Outline

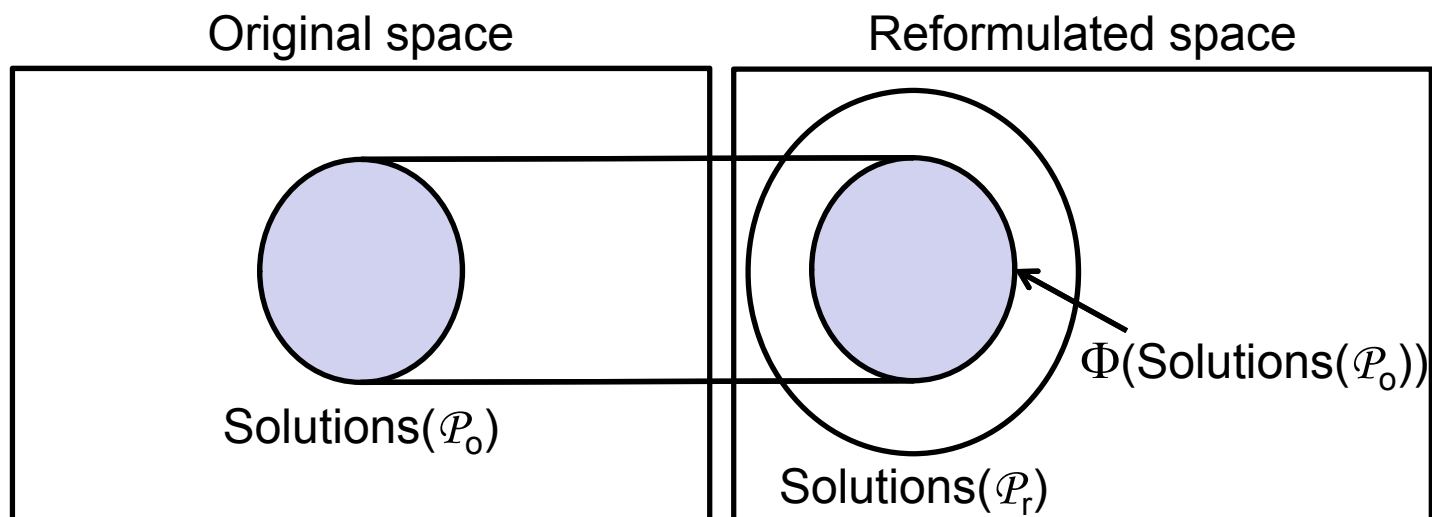
---

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

# Abstraction & Reformulation



... may be an approximation

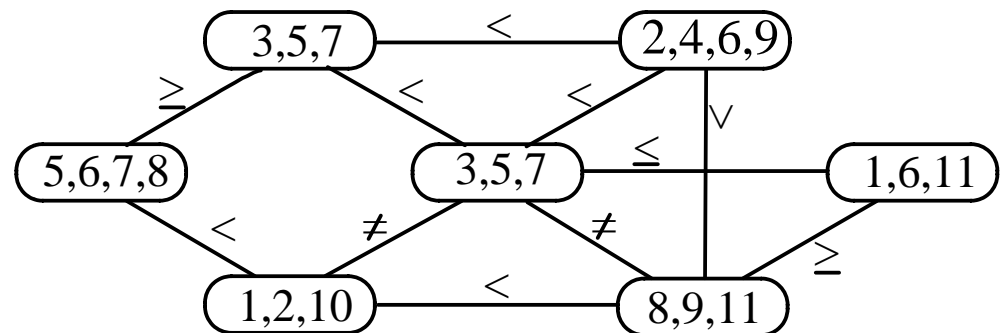


# Constraint Satisfaction Problems

- Formulation:  $F = (\mathcal{V}, \mathcal{D}, C)$ 
  - $\mathcal{V}$  = set of variables
  - $\mathcal{D}$  = set of their domains
  - $C$  = set of constraints restricting the acceptable combination of values for variables

- Query: All solutions, a single solution, *etc.*

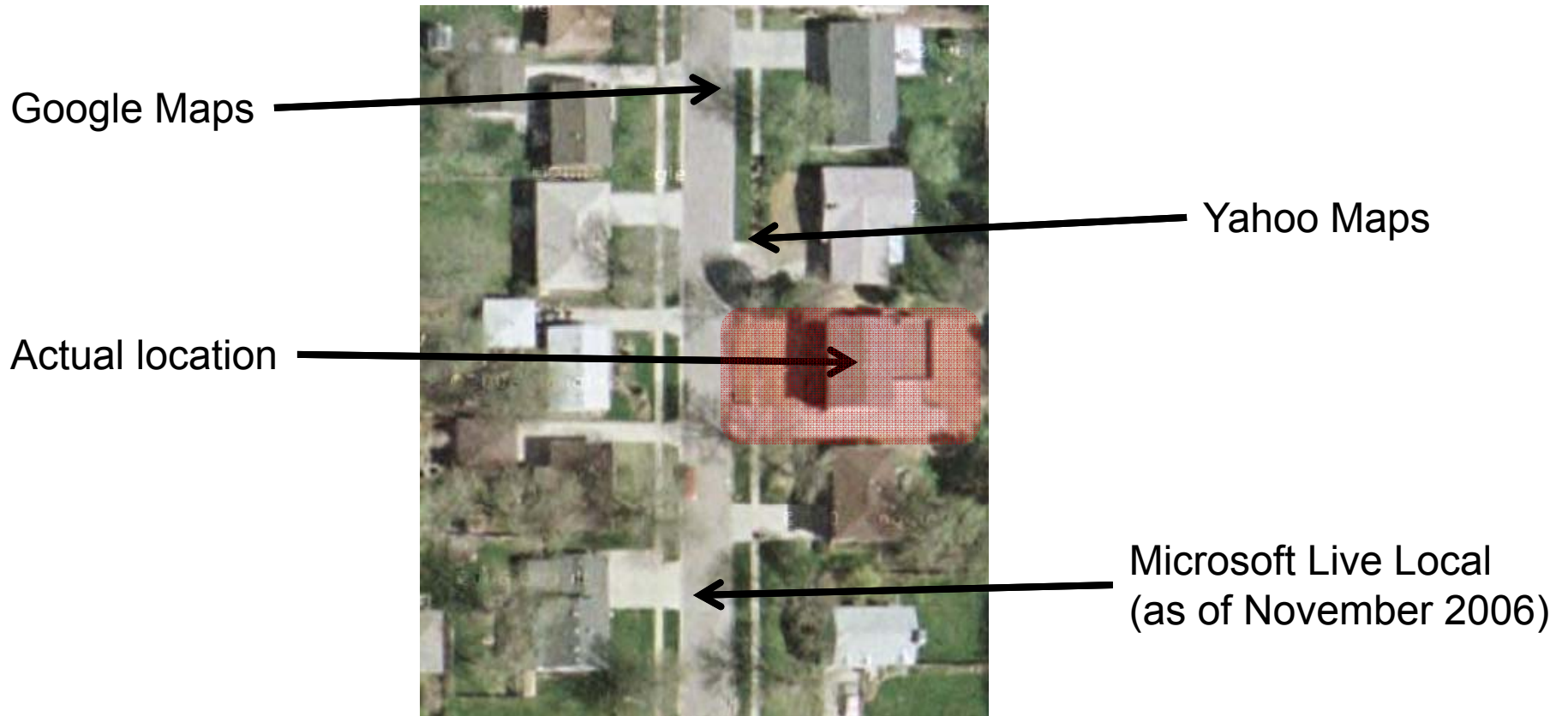
- Solved with
  - Constraint propagation
  - Search



- Term: variable-value pair (vvp)

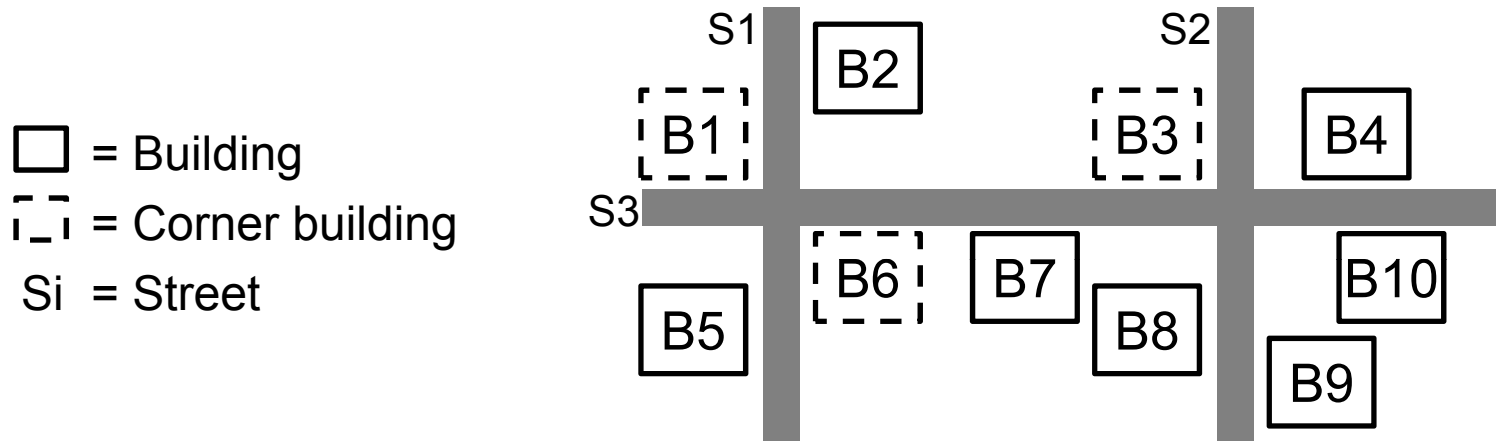
# Issue: finding Ken's house

---



# Building Identification (BID) problem

- Layout: streets and buildings



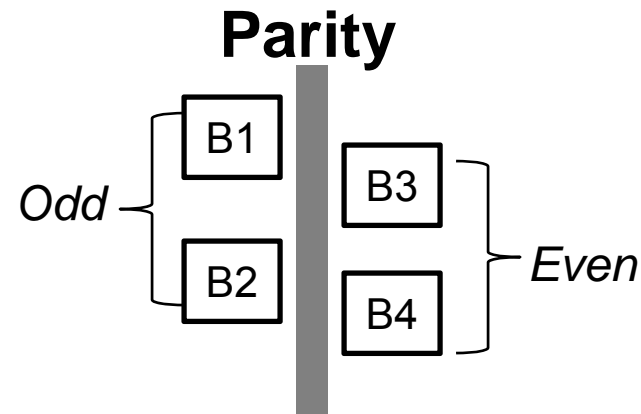
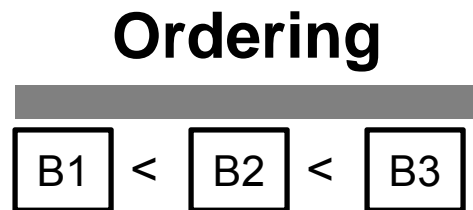
- Phone book
  - Complete/incomplete
  - Assumption: all addresses in phone book must be used

S1#1, S1#4, S1#8,  
S2#7, S2#8, S3#1,  
S3#2, S3#3, S3#15,  
...

# Basic (address numbering) rules

---

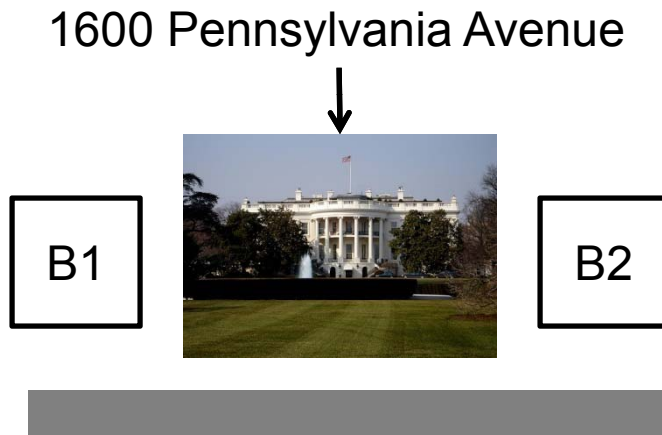
- Ordering
  - Numbers increase/decrease along a street
- Parity
  - Numbers on a given side of a street are odd/even



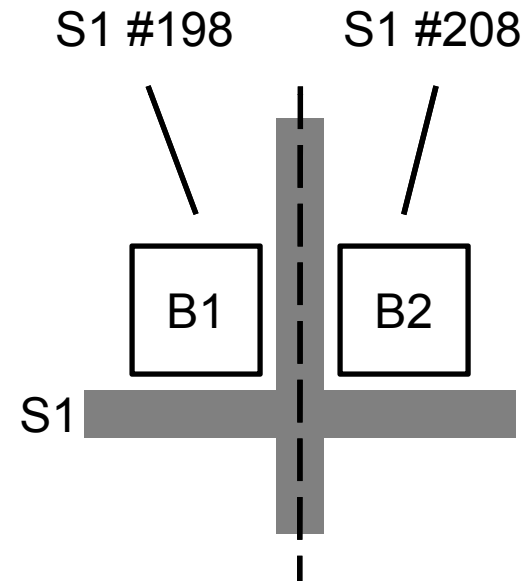
# Additional information

---

## Landmarks



## Gridlines



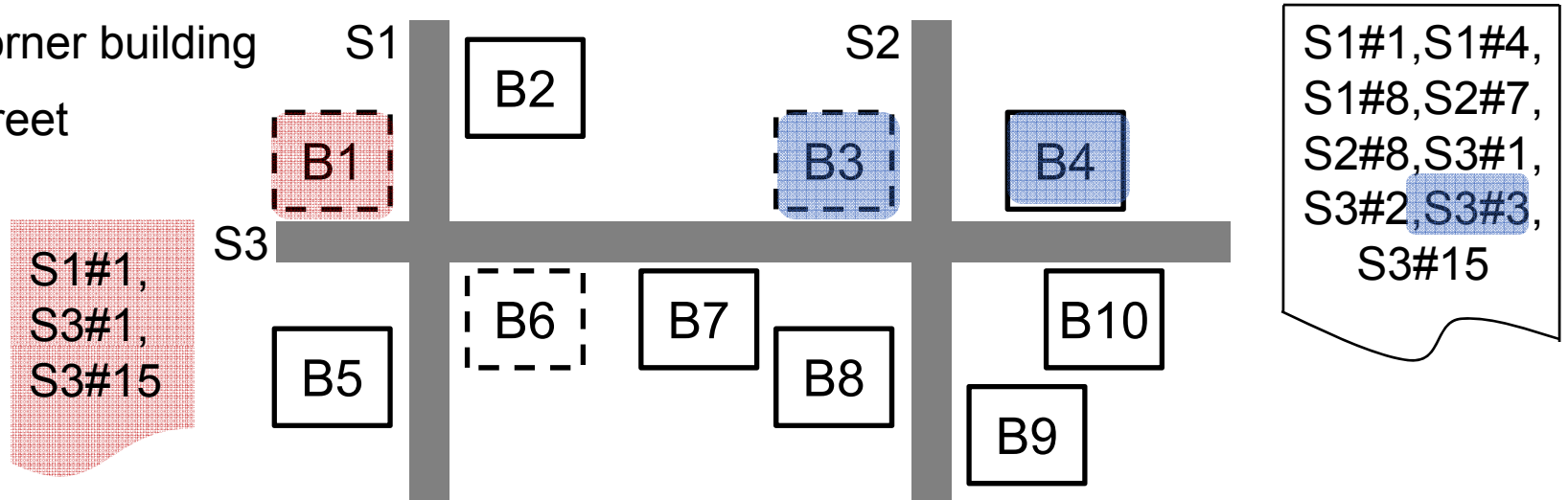
# Query

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

□ = Building

□-□ = Corner building

S<sub>i</sub> = Street



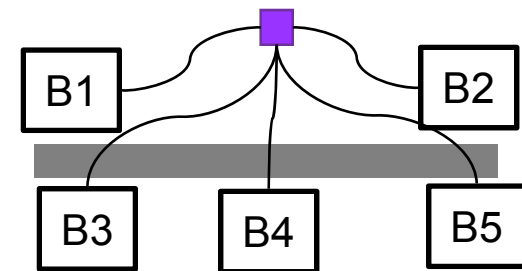
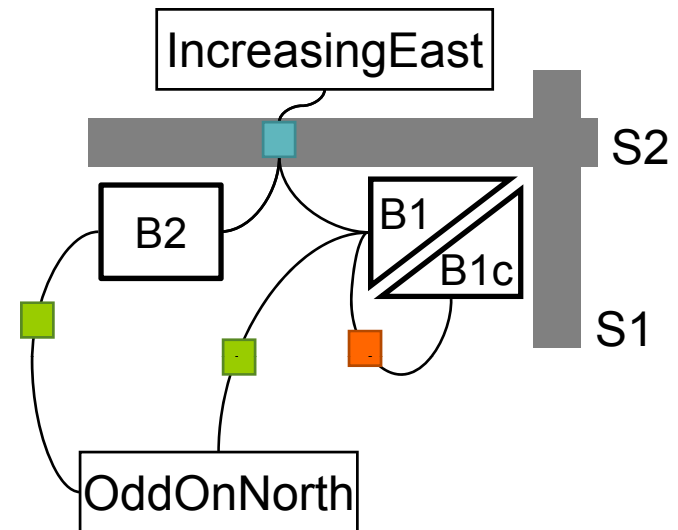
# Outline

---

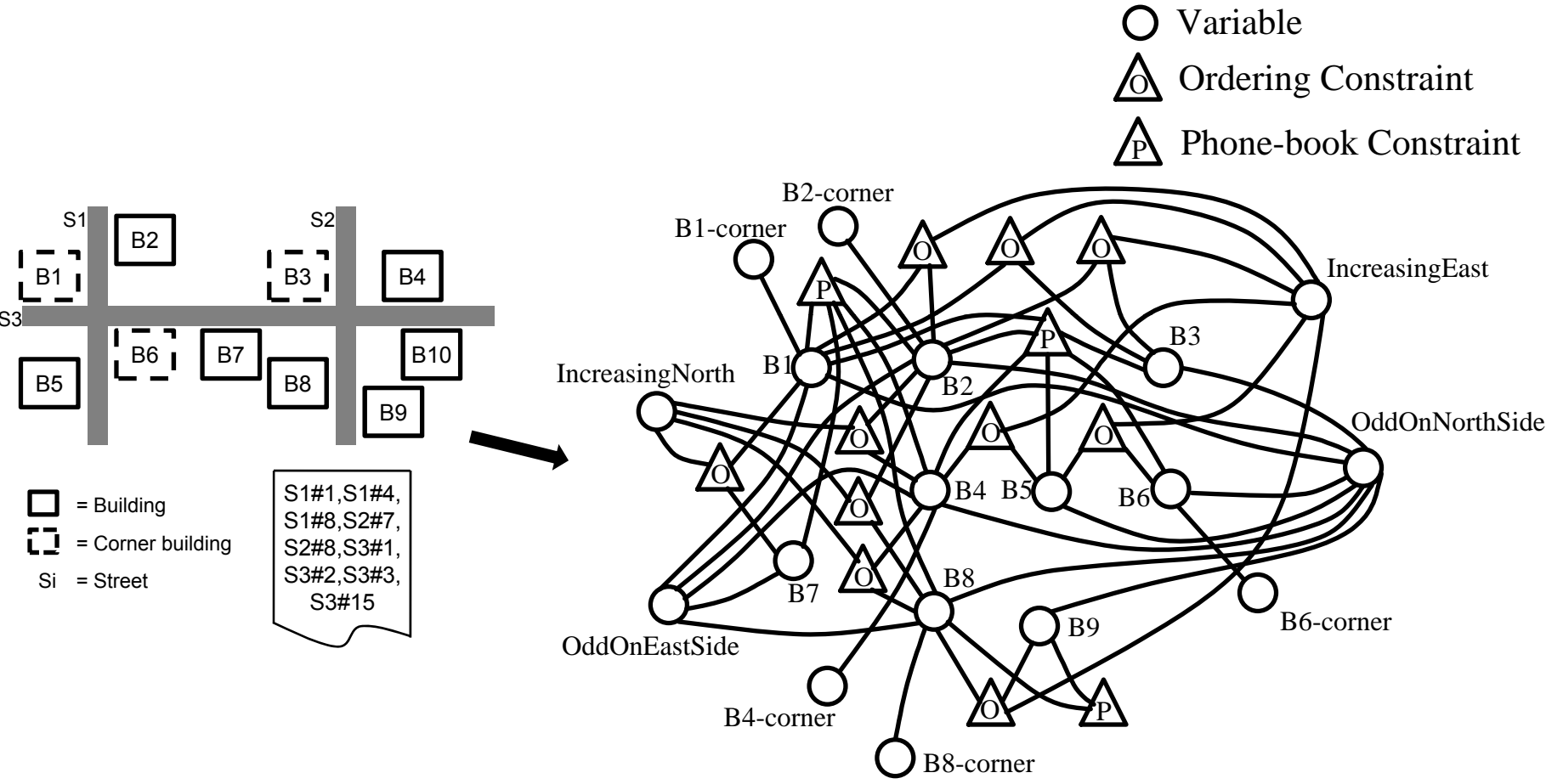
- Background
- **BID model & custom solver**
- Reformulation techniques
- Conclusions & future work

# CSP model

- Parity constraints
- Ordering constraints
- Corner constraints
- Phone-book constraints
- Optional: grid constraints



# Example constraint network



# Features of new model & solver

---

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

# Outline

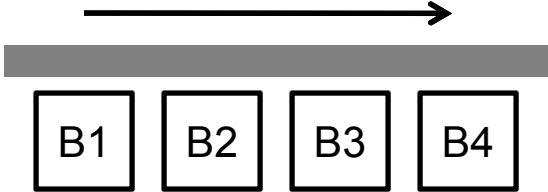
---

- 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

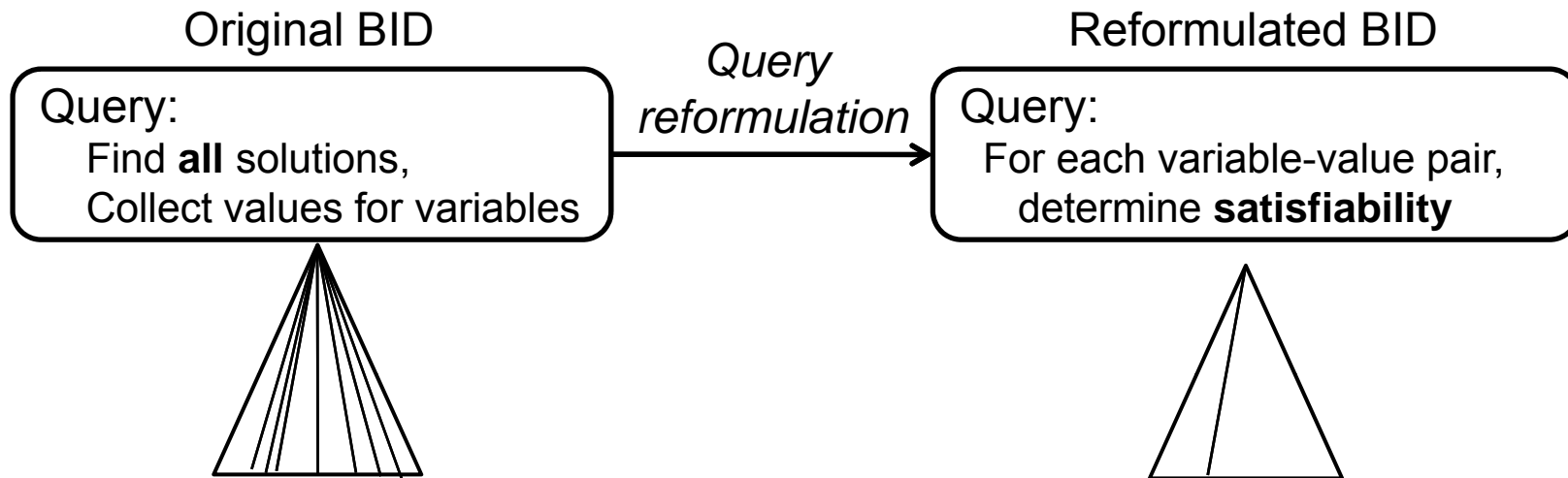


Phone book: {4,8}

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 (address) appear in **at least one** solution for a variable (building)

# 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	Number of...		
	Completeness	Buildings	Corner buildings	Blocks
NSeg125-c	100.0%	125	17	4
NSeg125-i	45.6%			
NSeg206-c	100.0%	206	28	7
NSeg206-l	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

# Evaluation: query reformulation

---

Incomplete phone book → many solutions → better performance

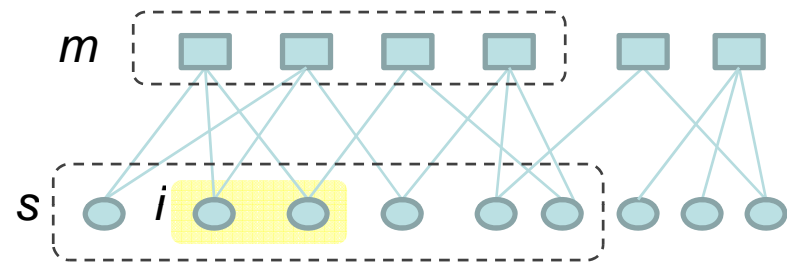
Case study	Original query	New query [s]
NSeg125-i	>1 week	744.7
NSeg206-i	>1 week	14,818.9
SSeg131-i	>1 week	66,901.1
SSeg178-i	>1 week	119,002.4

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, algorithm  $R(i,m)C$ 
  - For every  $m$  constraints
    - Compute **all solutions** of length  $s$
    - To generate tuples of length  $i$
  - Space:  $O(d^s)$



- Reformulated BID query is  $R(1,|C|)C$
- Query reformulation for Relational  $(i,m)$ -consistency
  - For each combination of values for  $i$  variables
    - Try to extend to **one** solution of length  $s$
  - Space:  $O(\binom{s}{i}d^i)$ ,  $i < s$

# Outline

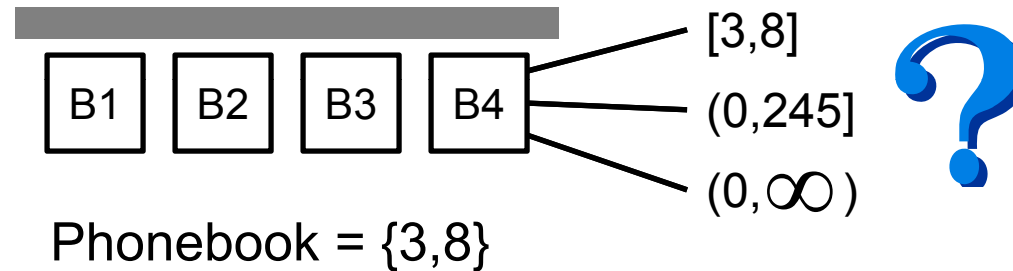
---

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

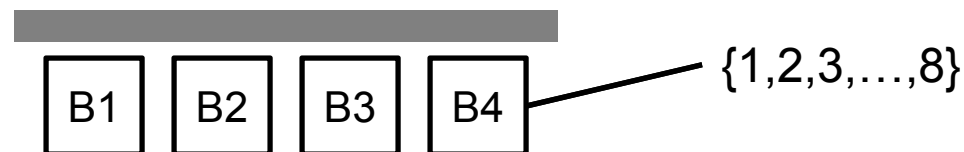
# Domain reformulation

---

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

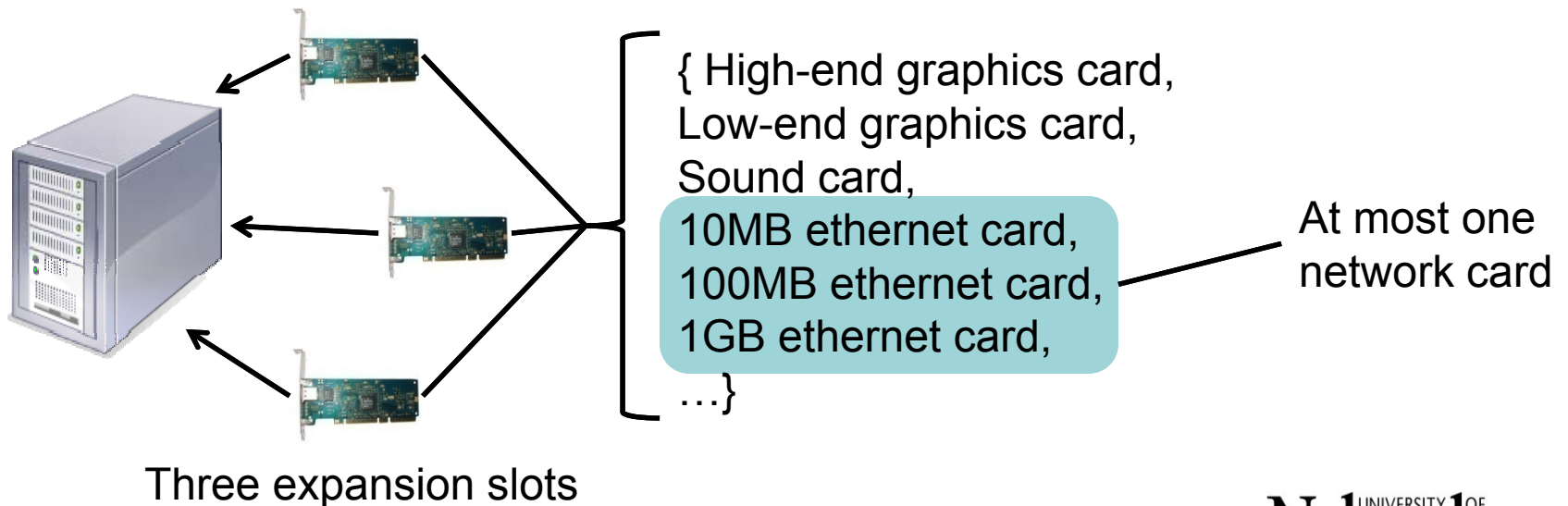


- Enumerate?



# AllDiff-Atmost constraint

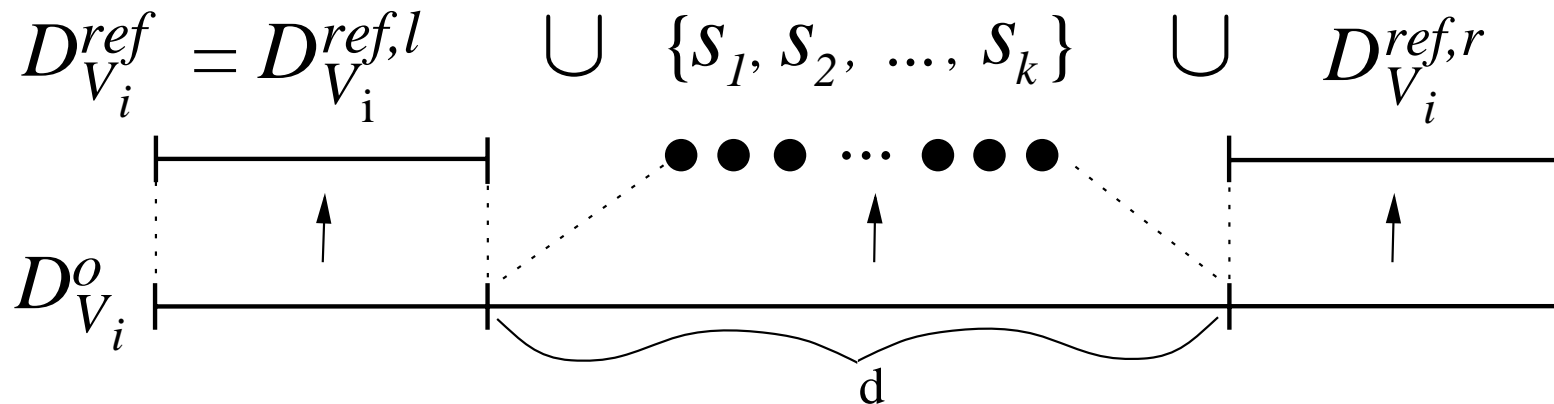
- AllDiff-Atmost( $\mathcal{A}, k, d$ )
  - The variables in  $\mathcal{A}$  can be assigned at most  $k$  values from the set  $d$



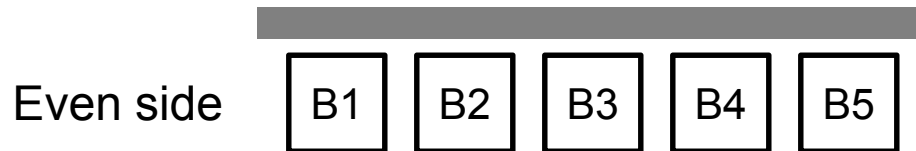
# AllDiff-Atmost reformulation

## Replaces

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



# AllDiff-Atmost in the BID

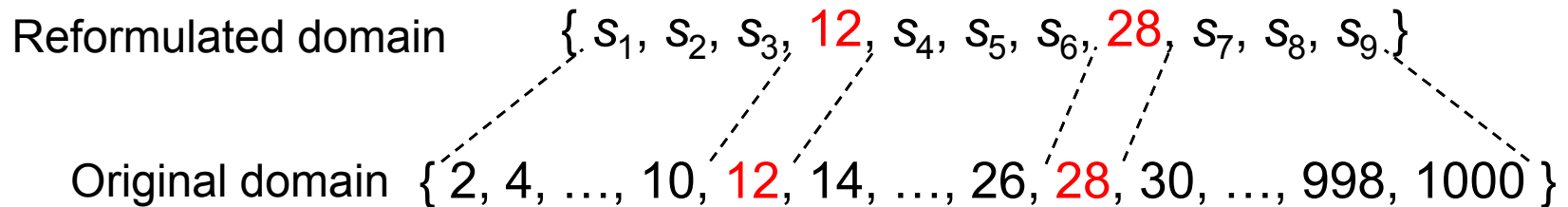


Phone book: {12,28}

Original domain = {2, 4, ..., 998, 1000}

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

- Can use at most
  - 3 addresses in [2,12)
  - 3 addresses in (12,28)
  - 3 addresses in (28,1000]



# 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

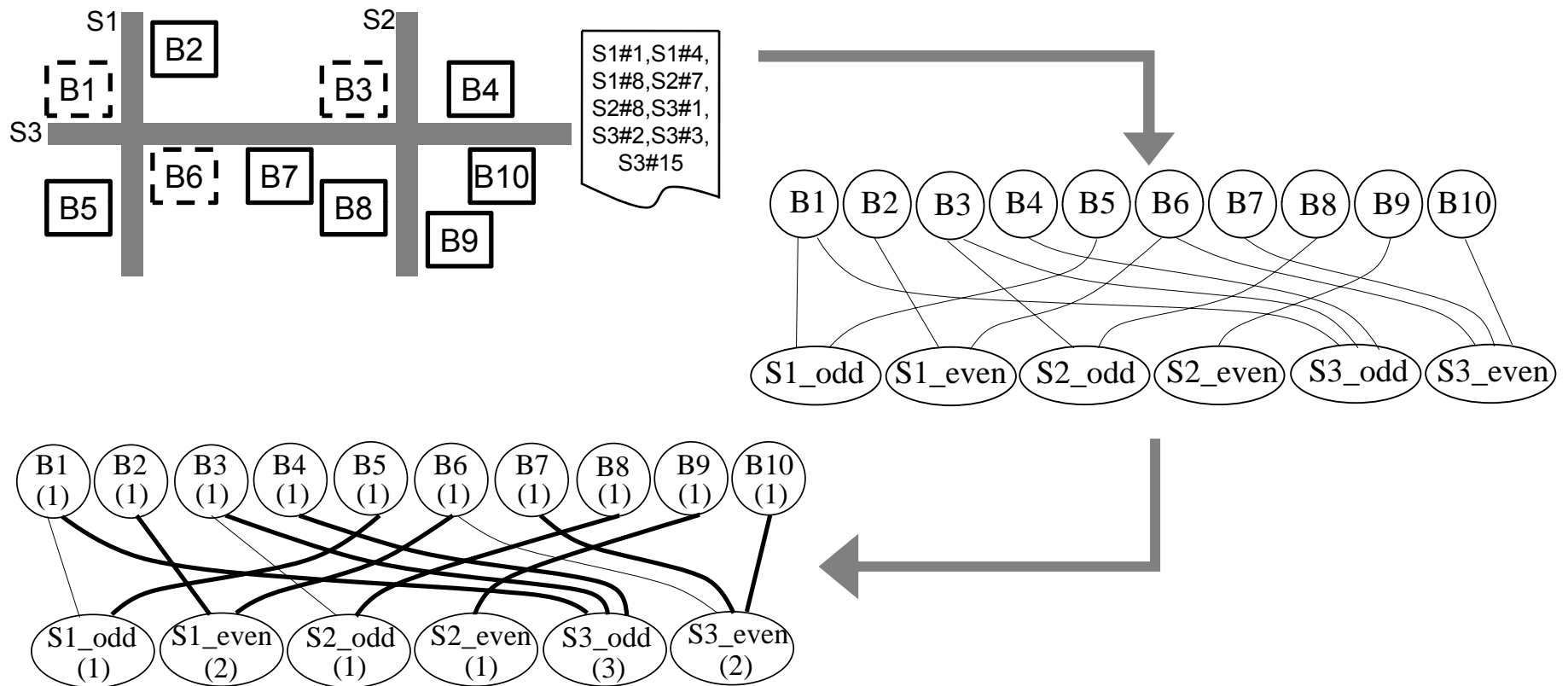
# Outline

---

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

# BID as a matching problem

- Assume we have no grid constraints



# BID w/o grid constraints

---

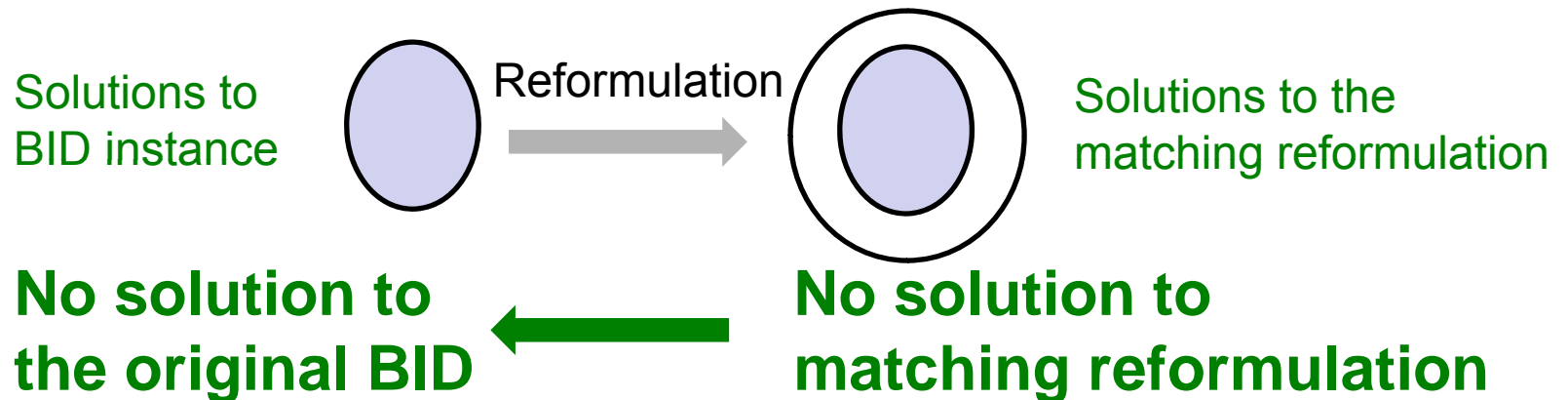
- BID instances without grid constraints can be solved in *polynomial time*

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

---

1. Matching reformulation as a *necessary approximation* of the BID



2. Filtering

[Régis, 1994]

**Remove vvps that do not appear in a maximum matching**

# Matching reformulation in Solver

---

- **Remove vvp's that do not...**      **Preproc1**
- For every vvp
  - Consider CSP + vvp
  - **Is the relaxed CSP solvable?**      **Preproc2**
    - Find one solution using BT search
      - **After each variable instantiation, remove vvp's that do ....**      **Lookahead**

# Evaluation: matching reformulation

- 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	<b>6.0%</b>	726.6	<b>41.0%</b>	701.1	<b>3.5%</b>
NSeg206-c	2277.5	614.2	<b>73.0%</b>	1559.2	<b>31.5%</b>	443.8	<b>71.5%</b>
SSeg178-i	138404.2	103244.7	<b>25.4%</b>	121492.4	<b>12.2%</b>	85185.9	<b>29.9%</b>

- 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-c	100.8	33.2	<b>67.1%</b>	<b>140.2</b>	<b>-39.0%</b>	29.8	<b>78.7%</b>
NSeg131-i	114405.9	114141.3	<b>0.2%</b>	107896.3	<b>5.7%</b>	<b>108646.6</b>	<b>-0.7%</b>

# Outline

---

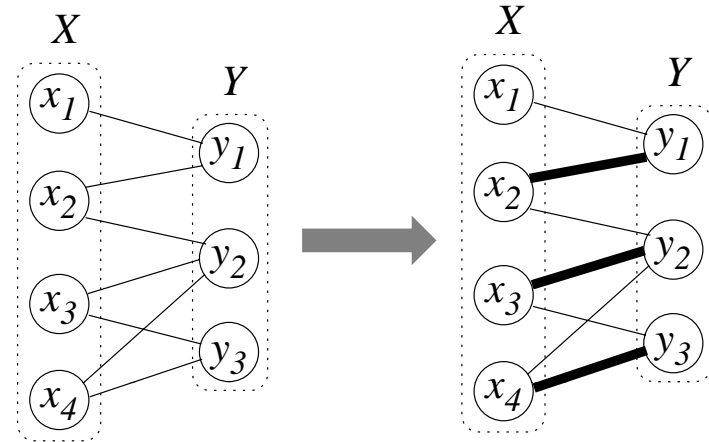
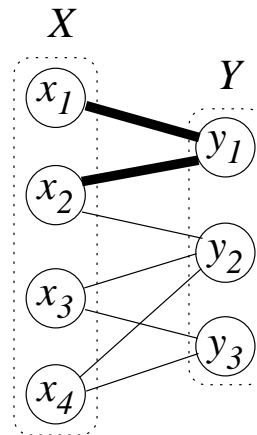
- Background
- BID model & custom solver
- Reformulation techniques
  - Query reformulation
  - AllDiff-Atmost & domain reformulation
  - Constraint relaxation
  - **Reformulation via symmetry detection**
- Conclusions & future work

# Find all maximum matchings

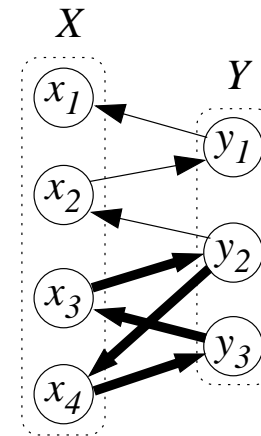
Find one maximum matching [Hopcroft+Karp, 73]

Identify... [Berge, 73]

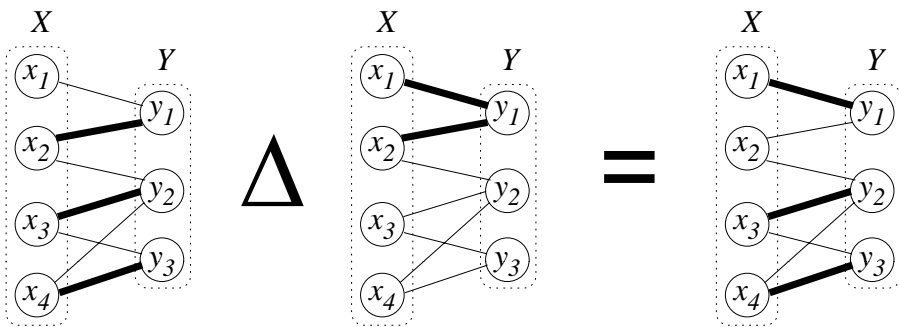
... even alternating paths starting @ a free vertex:



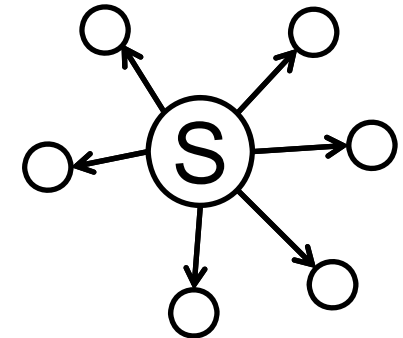
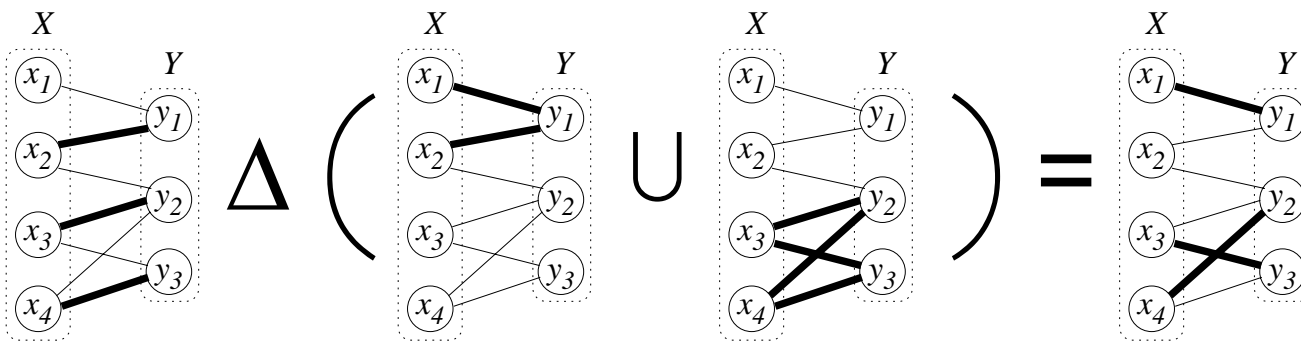
... alternating cycles:



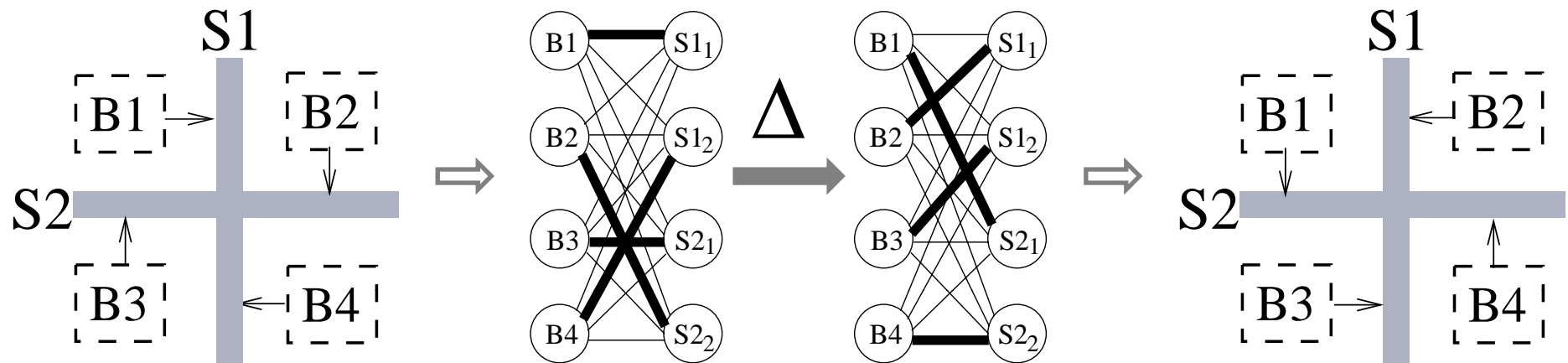
# Symmetric maximum matchings



**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
  - Avoid exploring symmetric solutions during search
- 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
- Exploit the symmetries we identified
- Enhance the model by incorporating new constraints [Michalowski]

---

# Questions?