A General Approach to Using Problem Instance Data for Model Refinement in Constraint Satisfaction Problems

Martin Michalowski

Doctoral Defense

July 22nd, 2008

Outline

- Motivation
  - Building Identification Problem
  - Constraint model refinement
  - Modeling: Constraint Satisfaction Problems (CSPs)
- Constraint-inference framework
  - Outline core components
  - Experimental evaluation
- Contributions and Related Work
- Future Work

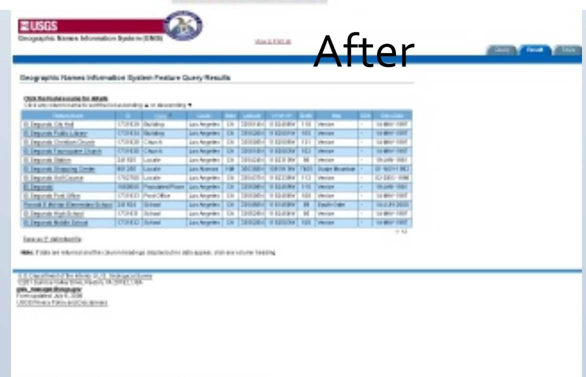
Motivation

Building Identification (BID) Problem

Traditional Sources

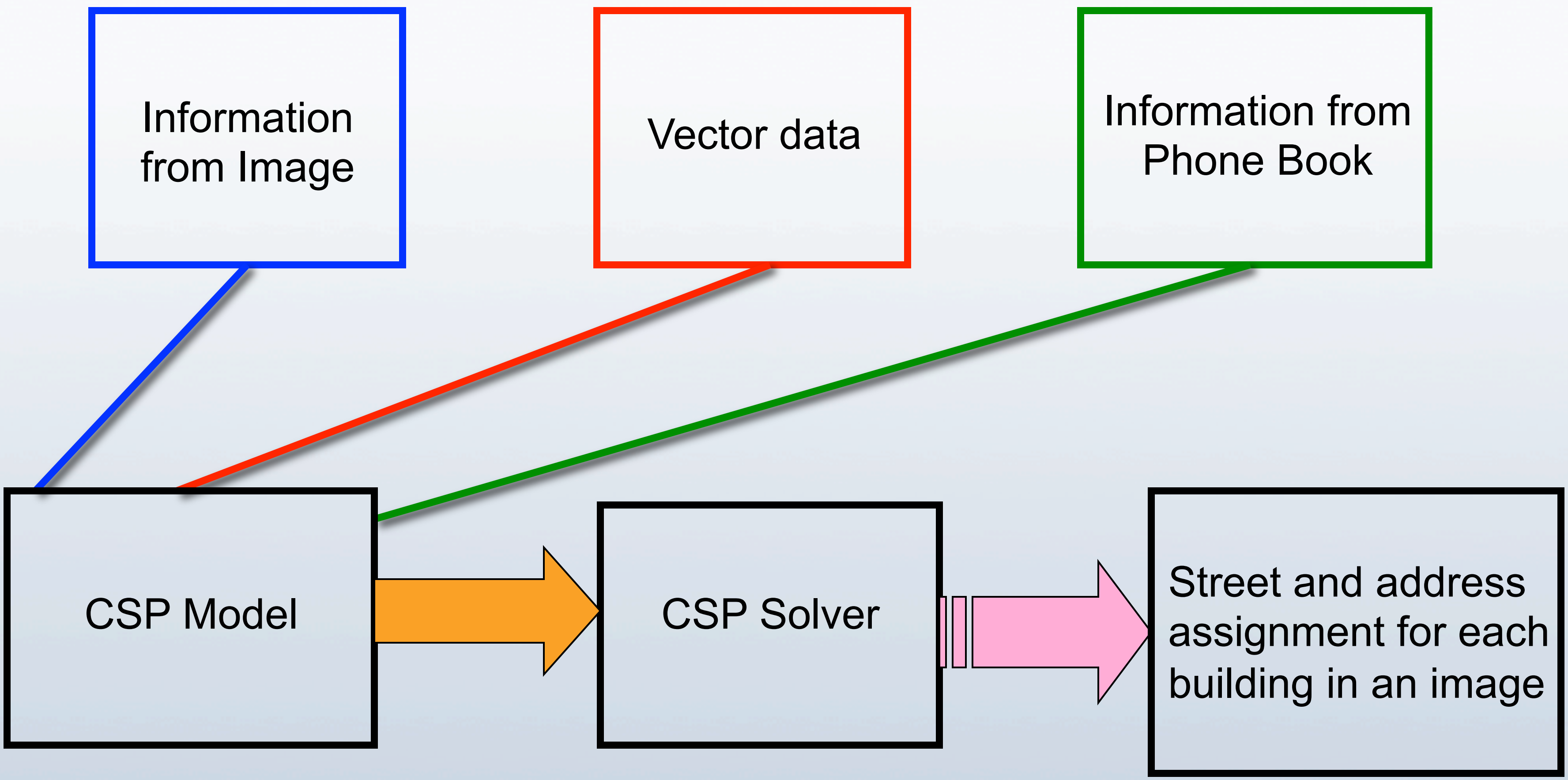
Non-traditional Sources

Before



After

BID Problem as a CSP

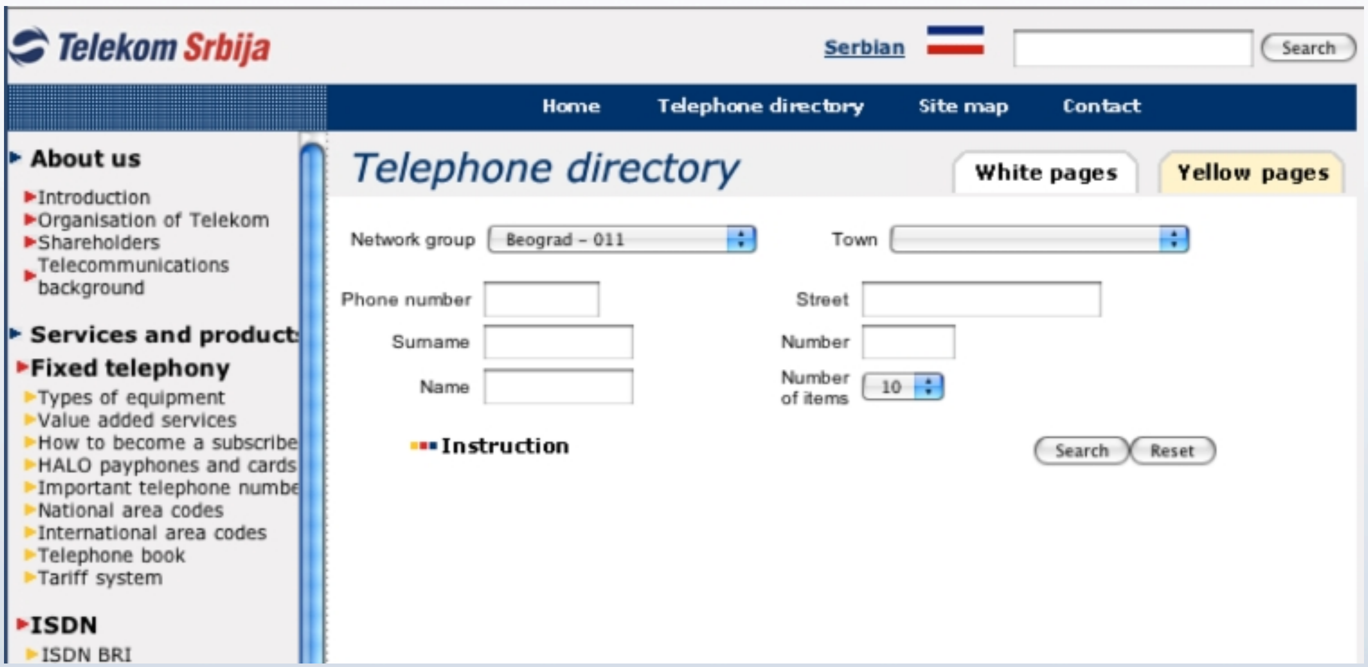


[Michalowski & Knoblock 2005]

Public Information

- Set of street names
- Set of buildings
    - Potential street(s) it is on
    - Side of street it is on
    - Order for a given street
- Additional information
    - Side of street where even numbers lie
    - Ascending addresses direction
- Helpful but not required
    - Constrains the problem

Public Information



Phone book

- Set of known addresses for all streets in image (vector data)

Example Constraints

<u>Parity Constraint</u>

Assures all these buildings will be even or odd, not a mix

Example Constraints

Ascending Constraint

Assures that address > address because we know numbers ascend in south direction on N/S running streets

Key Ideas

- Use both explicit and implicit information in publicly available data sources.
  - Challenge: combining this information
  - Solution: use a constraint satisfaction framework
- Leverage common properties of streets and addresses
  - Cannot be deduced from any individual source but require the combination of data from multiple sources.
  - Represent as constraints

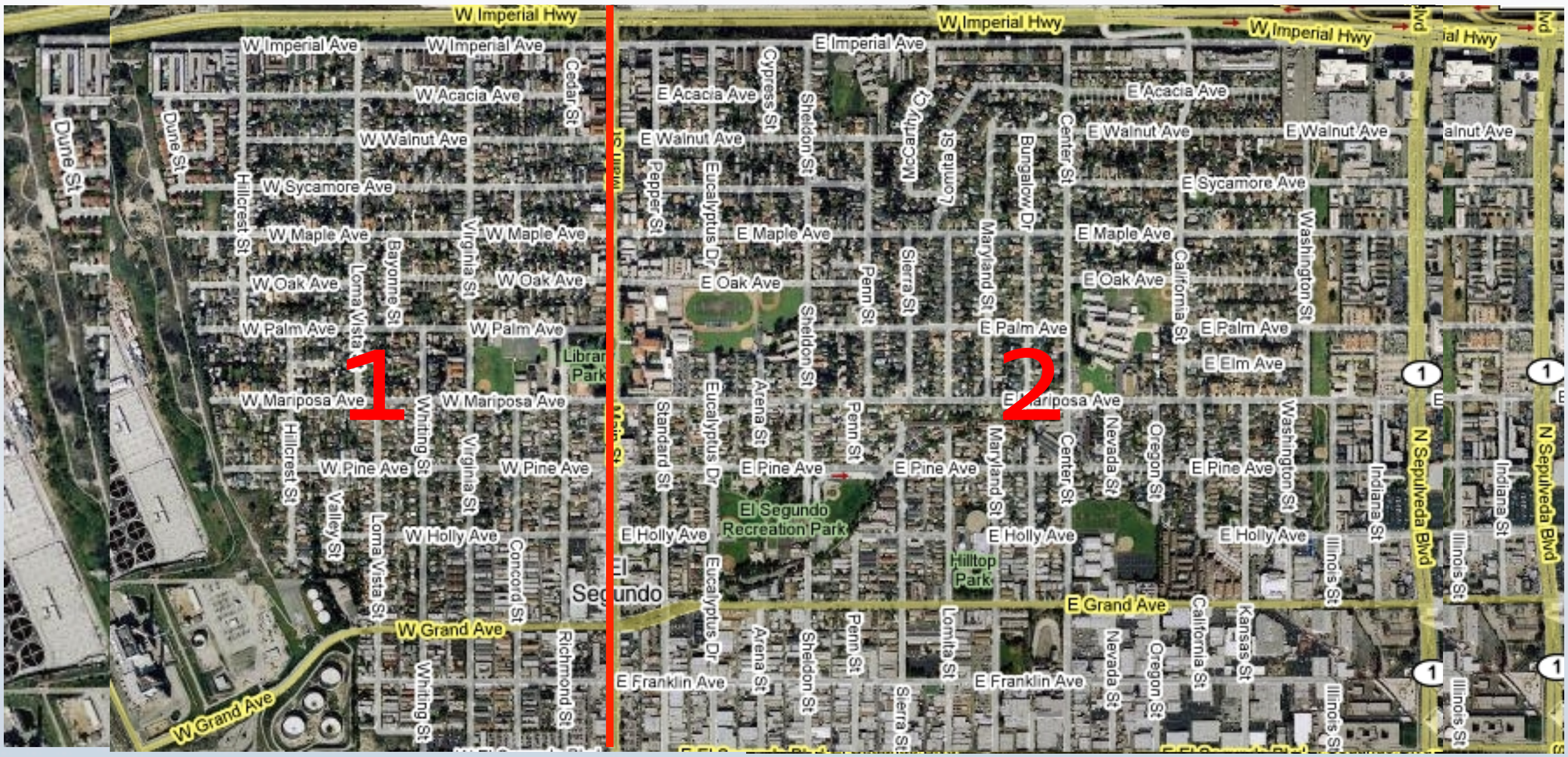
Challenges

- Varying addressing schemes exist
  - Static models don't work
- Single areas are non-homogeneous
  - Further complicates the problem
- Generating models for all possible scenarios
  - BAD! Lots of work, tedious, difficult to account for everything,…

Applicability of Constraints

Block Numbering

Non-Homogenous Areas

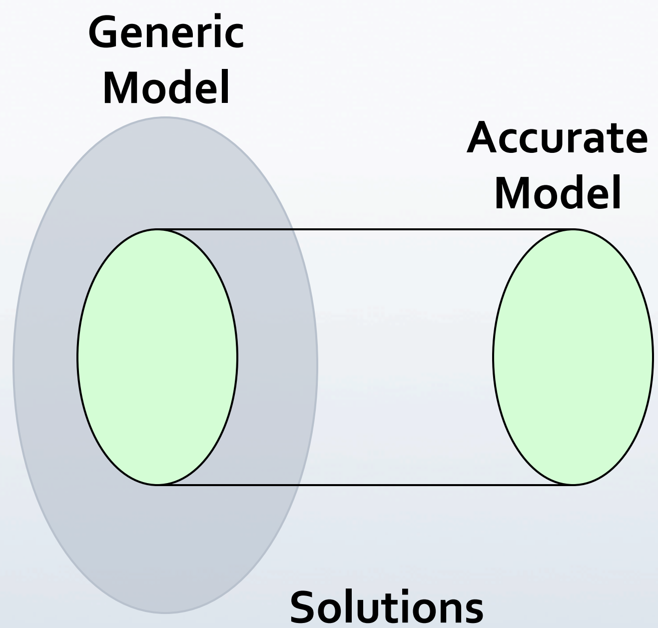


Constraints have different scopes

Addresses increase West **‘Addresses increase West’** **CONFLICT** **‘Addresses increase East’** **Addresses increase East**

Observations

- Instances exhibit variations
- Using the same generic model for all instances yields under-constrained problems



- The scope of constraints can differ in a problem instance
- Input data provides useful information
- Solution = Model refinement

Constraint Satisfaction Problems

- Definition of a CSP
  - **Given** $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$
    - $\mathcal{V}$ is a set of variables, $\mathcal{V} = \{V_1, V_2, \ldots, V_n\}$
    - $\mathcal{D}$ is a set of variable domains (domain values)

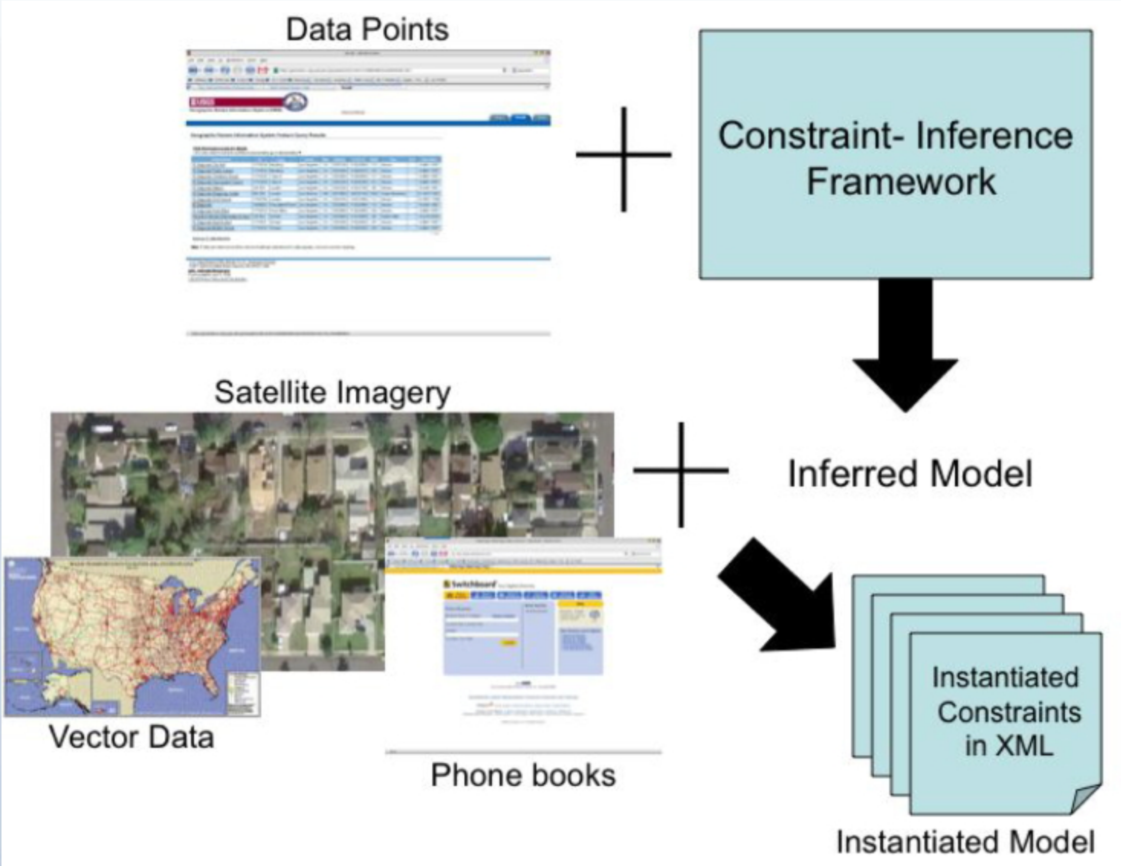
      $\mathcal{D} = \{D_{V1}, D_{V2}, \ldots, D_{Vn}\}$
    - $\mathcal{C}$ is a set of constraints, $\mathcal{C} = \{C_1, C_2, \ldots, C_l\}$

      $C_{Va,Vb,\ldots,Vi} = \{(x,y,\ldots,z)\} \subseteq D_{Va} \times D_{Vb} \times \ldots \times D_{Vi}$
  - **Query**: can we find a value for each variable such that all constraints are satisfied?

- Useful for modeling & solving combinatorial problems

Model Generation

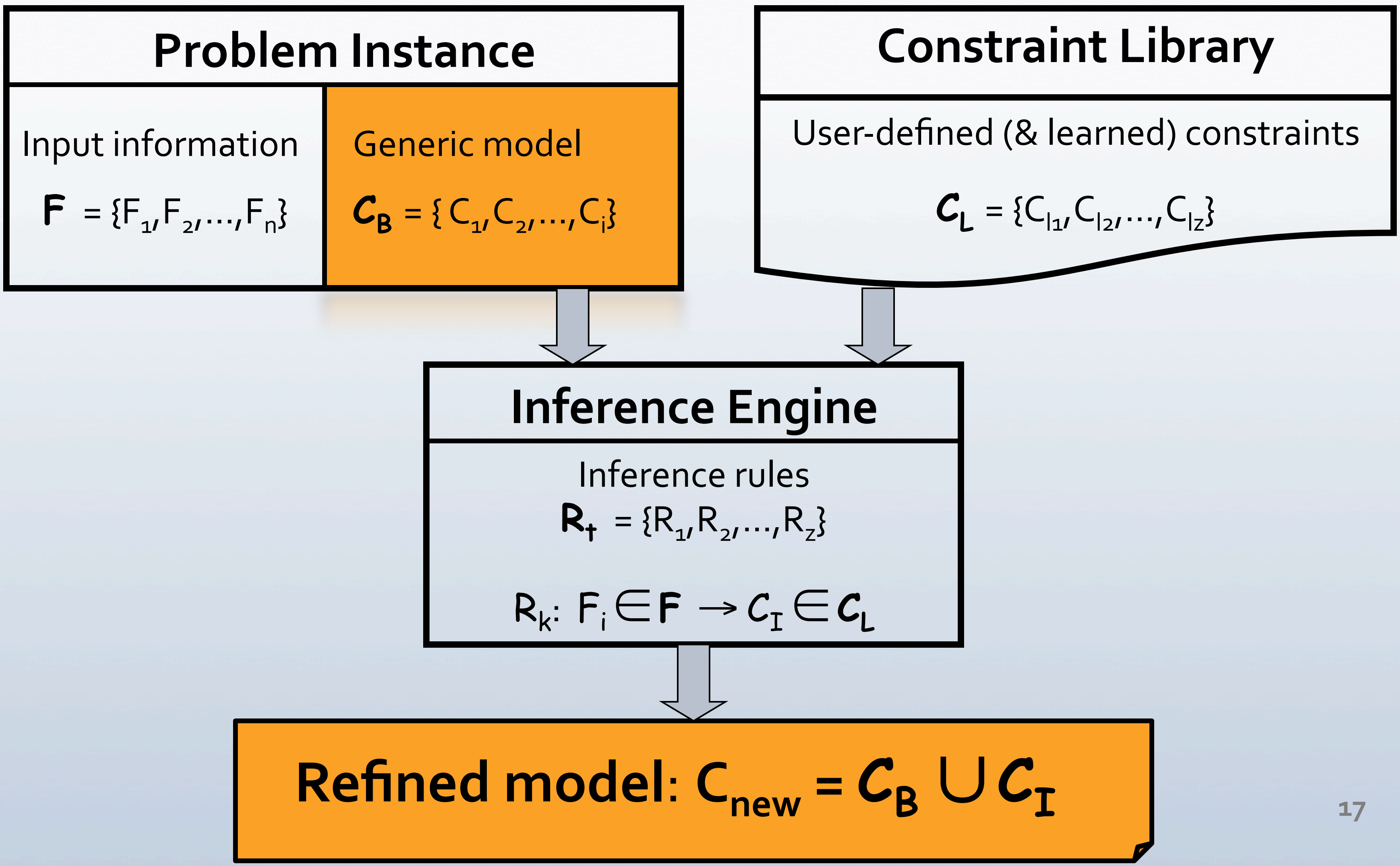
- Creating an accurate model is difficult
  - Thesis work focuses on the modeling problem



Outline

- Motivation
  - Building Identification Problem
  - Constraint model refinement
  - Modeling: Constraint Satisfaction Problems (CSPs)
- Constraint-inference framework
  - Outline core components
  - Experimental evaluation
- Contributions and Related Work
- Future Work

Constraint-Inference Framework



Input Information

- Describes a particular instance of the problem
  - Data points D_i characterized by a set of features F_{Di}
  - Framework exploits other types of information

Generic Model

- Constraints that capture the general characteristics of the problem class

Corner building can only be on one street

A single address per building

Library of Constraints

- Constraints that capture *some* characteristic of a problem instance
  - User-defined (or learned)

BID Problem Sample Library

Inference Rules

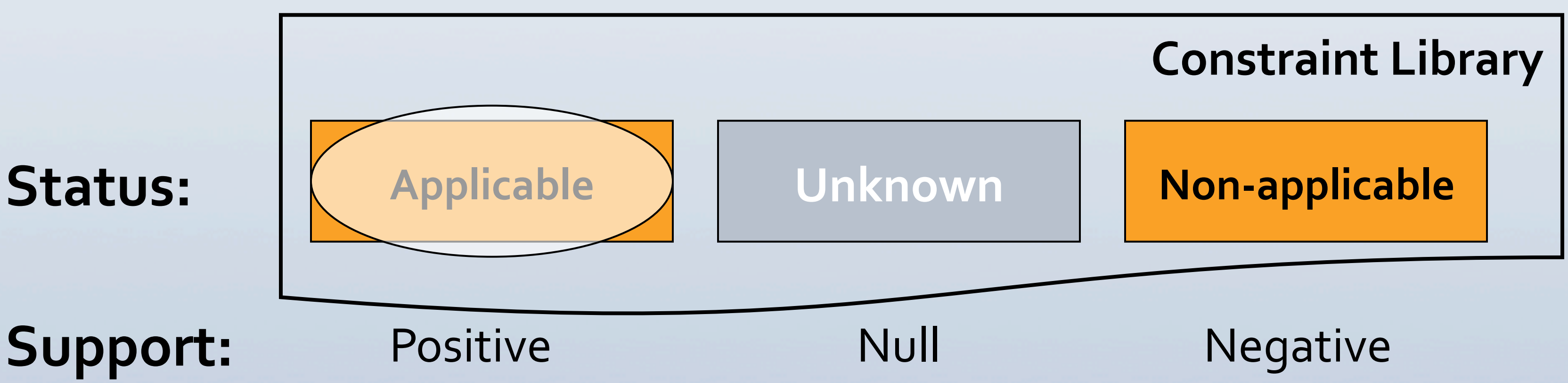
- Map the features of D_i to the constraints of the library
  - Determine constraints governing the instance
  - Rule language supports any programmable predicate expressions

BID Applicability Rule: Odd on North

IF(B_1 and B_2 are on E/W-running street $\wedge$ B_1, B_2 are on N side of street)
$\wedge$ addr(B_1) and addr(B_2) are odd
THEN increment positive support of constraint 'Odd on North'
ELSE increment negative support of constraint 'Odd on North'

Selecting Constraints

- Inference rules are evaluated using data points
  - Supports (+,-) provided for the constraints
- Constraints are partitioned based on their level of support
  - Status: Applicable, Unknown, Non-applicable
- Applicable constraints added to generic model



Selecting Constraints: Algorithm

```
CONSTRAINT-INFERENCE(D, finalSet)
finalSet ← {}
constraints ← constraintLibrary
buckets ← CREATEBUCKETS(D)
for i ← 0 to size[buckets]
        do B ← buckets[i]
            constraints ← EVALUATERULES(B)
for i ← 0 to size[constraints]
        do C ← constraints[i]
            if POSSUPPORT(C) > NEGSUPPORT(C)
                then finalSet ← finalSet ∪ C
```

- Grouping of data points based on feature values

- Evaluation of inference rules to provide support for constraints

- Inference of applicable constraints based on their level of support

Solving Complex Instances

- Large areas may lack expressiveness in input data
  - Incorrect inferences
- Introducing scope complicates the problem
  - Determining scope should be domain independent
- Domain expert shouldn’t play a large role

Incorrect Inferences

- Caused by noisy or weak support

A Solution

Support Level

$f\,(\text{support}^{+}_{C_i}, \text{support}^{-}_{C_i}, C_i)$

Expresses a level of confidence in the inference of a constraint

Support Levels

- Increase confidence in inference by increasing the supports provided
  - Augment the set of inference rules
  - Support a n-to-1 mapping of rules to constraints
- But…
  - More general rules can lead to incorrect (noisy) support
- Non-binary support levels handle this

Example: BID problem, *Increasing North* rules

1. $((\mathrm{sType}(B1) = \mathrm{sType}(B2) = \mathrm{NS})\ \&\ (\mathrm{sSide}(B1) = \mathrm{sSide}(B2)))\ \&\ (\mathrm{addr}(B1) > \mathrm{addr}(B2))\ \&\ (\mathrm{lat}(B1) > \mathrm{lat}(B2))$
2. $((\mathrm{sType}(B1) = \mathrm{sType}(B2) = \mathrm{NS})\ \&\ (\mathrm{sSide}(B1) = \mathrm{sSide}(B2)))\ \&\ (\mathrm{addr}(B1) < \mathrm{addr}(B2))\ \&\ (\mathrm{lat}(B1) < \mathrm{lat}(B2))$

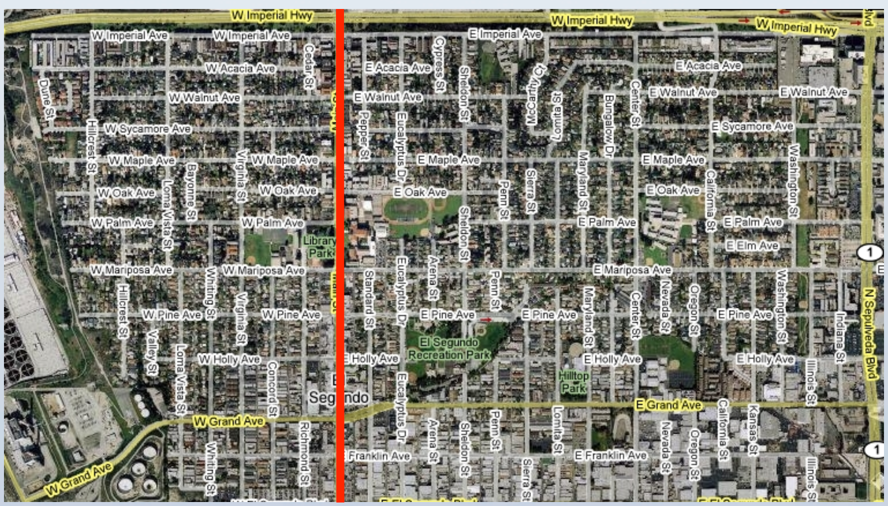
More general (not limited to same side of the street)

3. $((\mathrm{sType}(B1) = \mathrm{sType}(B2) = \mathrm{NS}))\ \&\ (\mathrm{addr}(B1) > \mathrm{addr}(B2))\ \&\ (\mathrm{lat}(B1) > \mathrm{lat}(B2))$
4. $((\mathrm{sType}(B1) = \mathrm{sType}(B2) = \mathrm{NS}))\ \&\ (\mathrm{addr}(B1) < \mathrm{addr}(B2))\ \&\ (\mathrm{lat}(B1) < \mathrm{lat}(B2))$

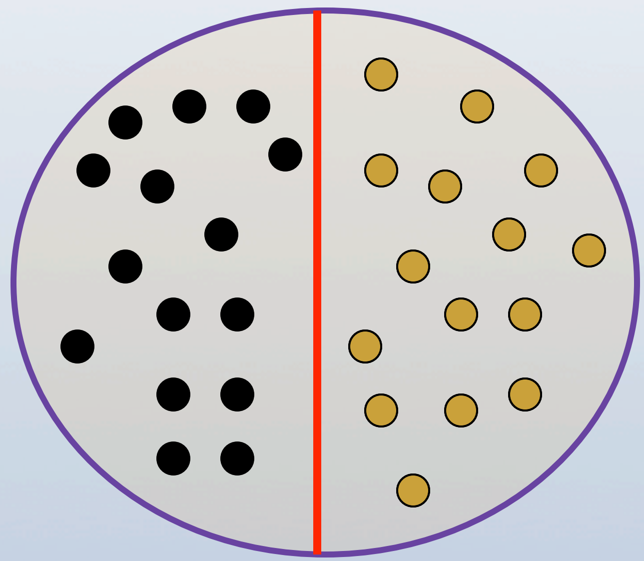
Determining the Scope

- Finding a constraint's scope shouldn't be domain specific
- Assume a *spatial* boundary
- Introduce a *variable* boundary

Spatial Separation



Problem Space



Support Vector Machines [Vapnik, 1995]

Determining the Scope

Domain-independent solution

Inferred Model

Constraint 1
Constraint 2 Conflict
Constraint 3...

Data Points

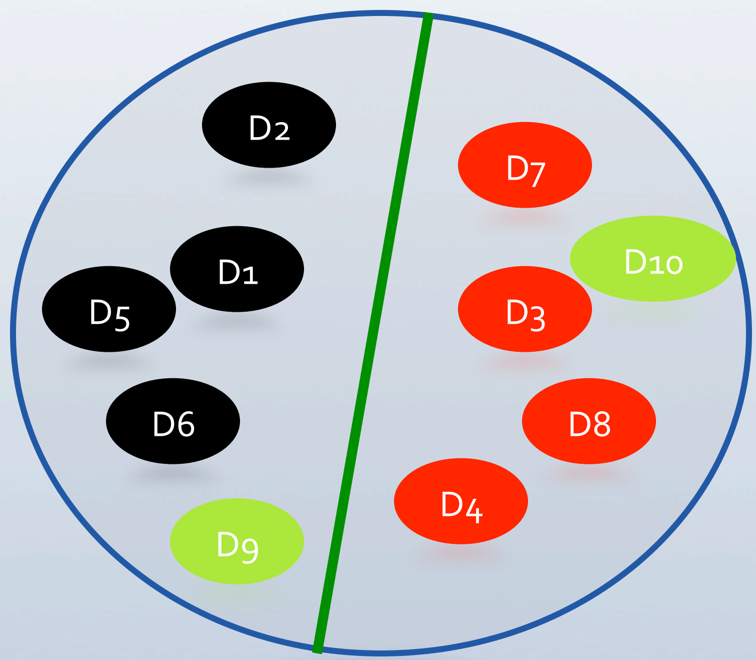
$\{D_{1,2,3,4,5,6,7,8,9,10}\}$

$D_{1,2,5,6} \rightarrow$ Constraint 1

$D_{3,4,7,8} \rightarrow$ Constraint 2

$D_{9,10} \rightarrow$?

Class Labels: **Constraint 1**
Constraint 2



Classify unknown data points

Support Vector Machine Model

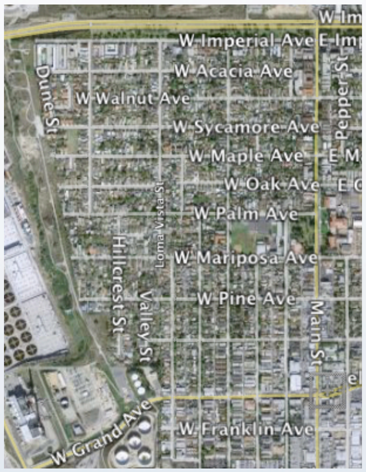
Automating the Process

- Instantiate the model
  - Data points ∈ All variables
  - Augment the scope of applicable constraints
- Represent the model in a recognized format
  - XCSP representation for the BID problem
- Solve automatically
  - Customized solver [Bayer+ CP'07]

Selecting Constraints: Homogenous Areas

Area	Data Points
1. El Segundo CA	(a) 38 points west of Main St. (b) 660 geocoded points (c) 12 USGS gazetteer points East of Main Street (schools and churches)
2. Downtown Los Angeles	7 hotels from an online hotels data source
3. San Francisco CA	16 USGS gazetteer points (schools and churches)
4. Boulder CO	7 USGS gazetteer points (schools only)
5. New Orleans LA	21 USGS gazetteer points (churches and schools)
6. Belgrade Serbia	85 points from a government planning website

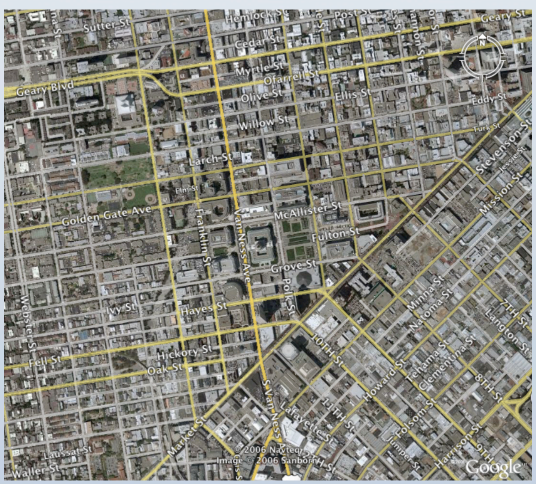
Selecting Constraints: Homogenous Areas



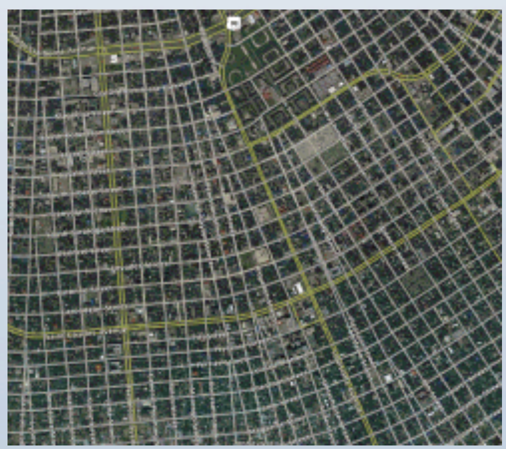
El Segundo CA

Downtown Los Angeles

Belgrade Serbia



San Francisco CA



New Orleans LA

Selecting Constraints: Homogenous Areas

Area	Parity	Block $k = 100$	Increasing North	Increasing East	Prec.	Recall
El Segundo CA(38)	✓	✓	✓	✓	100.00%	100.00%
El Segundo CA(660)	✓	✓	✓	✓	100.00%	100.00%
El Segundo CA(12)	✓	✓	×	✓	100.00%	89.90%
Downtown LA(7)	✓	✓	✓	×	100.00%	87.50%
San Francisco CA(16)	✓	✓	✓	✓	100.00%	100.00%
Boulder CO(7)	✓	N/A	×	✓	100.00%	76.45%
New Orleans LA(21)	✓	×	✓	×	100.00%	64.92%
Belgrade Serbia(85)	✓	N/A	✓	✓	100.00%	100.00%

✓ correctly inferred × not inferred N/A not applicable

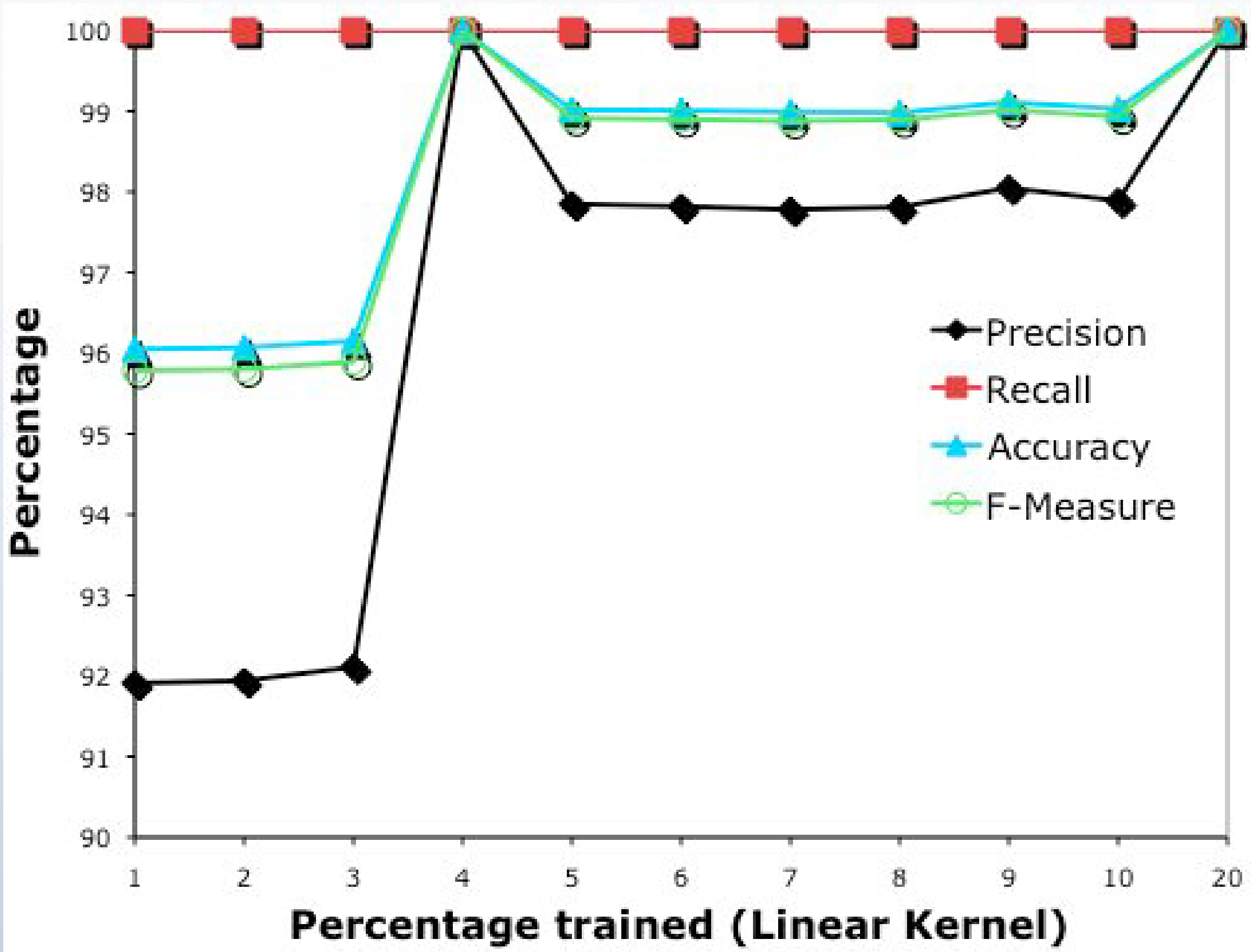
Slightly incomplete but accurate models

Complete and accurate models

Putting It All Together

- Solve non-homogeneous areas
- Enhance the set of inference rules
- Use support levels to deal with noisy data and support
- Infer scopes using SVMs

Determining the Scope: Evaluation



Very accurate with only 4% (~66 points) training data

Selecting Constraints: More Complex Areas

Area	Data Points
1. El Segundo CA	(a) 38 points west of Main St. (b) 1650 geocoded points (East & West of Main) (c) 20 USGS gazetteer points East & West of Main (schools and churches)
2. Downtown Los Angeles	7 hotels from an online hotels data source
3. San Francisco CA	16 USGS gazetteer points (schools and churches)
4. Boulder CO	7 USGS gazetteer points (schools only)
5. New Orleans LA	66 USGS gazetteer points (churches and schools)
6. Belgrade Serbia	88 points from a government planning website
7. Jakarta Indonesia	20 hotels from an online Indonesian source

Not solvable without added enchancements

Selecting Constraints: More Complex Areas

Area	Odd On North/East	Block $k = 100$	Increasing North	Increasing East	Prec.	Recall
El Segundo (38)	✓	✓	✓	✓	100.00%	100.00%
El Segundo (1750)	✓	✓	Scope 1: ✓ Scope 2: ✓	Scope 1: ✓ Scope 2: ✓	98.99%	100.00%
El Segundo (20)	✓	✓	Scope 1: ✓ Scope 2: ✓	Scope 1: ✓ Scope 2: ×	98.73%	89.90%
Downtown LA (7)	✓	✓	✓	×	100.00%	87.50%
San Francisco (16)	✓	✓	✓	✓	100.00%	100.00%
Boulder (7)	✓	N/A	×	✓	100.00%	76.45%
New Orleans (66)	✓	✓	Scope 1: ✓ Scope 2: ✓	Scope 1: ✓ Scope 2: ✓	97.67%	100.00%
Belgrade (88)	Scope 1: ✓ Scope 2: ✓	N/A	✓	✓	100.00%	100.00%
Jakarta (20)	✓	N/A	✓	✓	100.00%	100.00%

✓ correctly inferred × not inferred N/A not applicable

[illegible]

Performance Improvement

CSP Search Solver

	W/o orientation cons		W/ orientation cons			
	Runtime (sec)	Domain size	Runtime (sec)	Domain size	Runtime reduction	Domain reduction
NSeg125-c	22397.08	1.22	1962.53	1.0	11.41x	1.22x
NSeg125-i	22929.49	6.11	3987.73	4.18	5.75x	1.46x
NSeg206-c	198169.43	1.21	10786.33	1.0	18.37x	1.21x
NSeg206-i	232035.89	7.91	12900.36	4.99	17.99x	1.59x
SSeg131-c	173565.78	1.56	125011.65	1.41	1.39x	1.11x
SSeg131-i	75332.35	12.56	17169.84	3.92	4.39x	3.20x
SSeg178-c	523100.80	1.41	284342.89	1.31	1.84x	1.08x
SSeg178-i	334240.61	8.24	62646.91	3.23	5.34x	2.55x
				Average	**8.31x**	**1.68x**

Large reduction in runtime, significant increase in precision

Outline

- Motivation
  - Building Identification Problem
  - Constraint model refinement
  - Modeling: Constraint Satisfaction Problems (CSPs)
- Constraint-inference framework
  - Outline core components
  - Experimental evaluation
- Contributions and Related Work
- Future Work

Contributions

- Established a new domain for CP research
- A general constraint-inference framework
  - Applied to BID problem and Sudoku puzzles
- Use instance-specific data to specialize a model
  - Eliminates need for model generation a priori
- Handles noise
  - Support levels
- Machine learning approach to dealing with inconsistencies
  - Finding scope using SVMs
- Automated processing reducing user involvement

Related Work

- Constraint Programming
  - Puzzles [Lauriere 1978, Nadal 1990,…]
    - PROVERB [Littman 2002]
    - Sudoku [Simonis 2005]
    - BID problem [Michalowski+ 2005]
  - Uncertain and Probabilistic CSPs [Fargier 1993]
- Constraint Modeling
  - Contextualizing constraints [Graham+ 2006, Cheung+ 1996]
  - Compositional modeling in QR [Falkenhainer+, 1991, …]
  - Specification languages [Frisch+ 2005, Renker+ 2004,…]

Related Work

- Learning Constraints
  - Learning from data [Coletta+ 2003, Bessière+ 2005]
  - Learning to optimize models [Colton+ 2001, Lallouet+ 2005]
- Geospatial
  - Geocoding [Bakshi+ 2004]
  - Computer vision [Agouris+ 1996, Doucette+ 1999]

Outline

- Motivation
  - Building Identification Problem
  - Constraint model refinement
  - Modeling: Constraint Satisfaction Problems (CSPs)
- Constraint-inference framework
  - Outline core components
  - Experimental evaluation
- Contributions and Related Work
- Future Work

Future Work

- Learning inference rules
  - Agglomerative clustering approach
- Enhancing the learning of scopes
  - Non-binary conflicting constraints (multi-class SVM)
- BID problem
  - End-to-end online application for building identification

Thank you!!

Supplemental Material

Application Domain II

Sudoku Puzzles

7		9						
		4	7		8	3		
5			6	3			9	
			3	1	7		8	
			4		6			
	7		2	8	9			
	2		5	4				8
		6	9		1	2		
						5		1

7	3	9	1	2	5	8	4	6
6	1	4	7	9	8	3	2	5
5	8	2	6	3	4	1	9	7
2	6	5	3	1	7	4	8	9
3	9	8	4	5	6	7	1	2
4	7	1	2	8	9	6	5	3
1	2	7	5	4	3	9	6	8
8	5	6	9	7	1	2	3	4
9	4	3	8	6	2	5	7	1

CSP Example: Sudoku

Given:

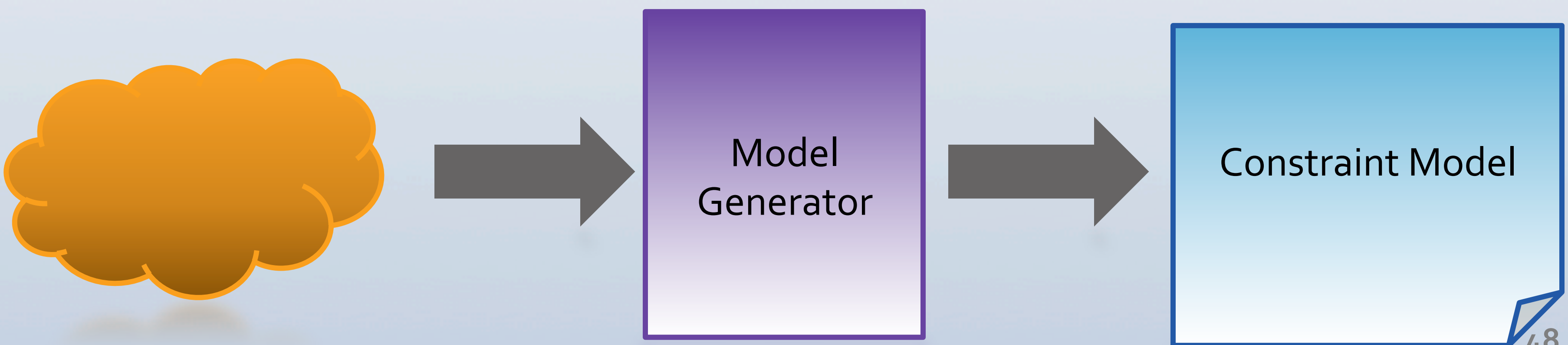
				7	1			
					2	5		
9	1					8		
					7		8	9
		7			8		4	
	5	6	9			7		
1	2	3					6	
4			3					
			7	6				

- One model
  - 81 variables: $C_{1,1} \ldots C_{9,9}$
  - Domains: {1,2,3,…,8,9}
  - Constraints:
    - all-diff constraints, 9-arity
    - One constraint per row
    - One constraint per column
    - One constraint per (3x3) unit

Query: Fill the empty cell such that 1..9 appear in each row, column, and unit w/o repetition

Taking Sudoku One Step Further

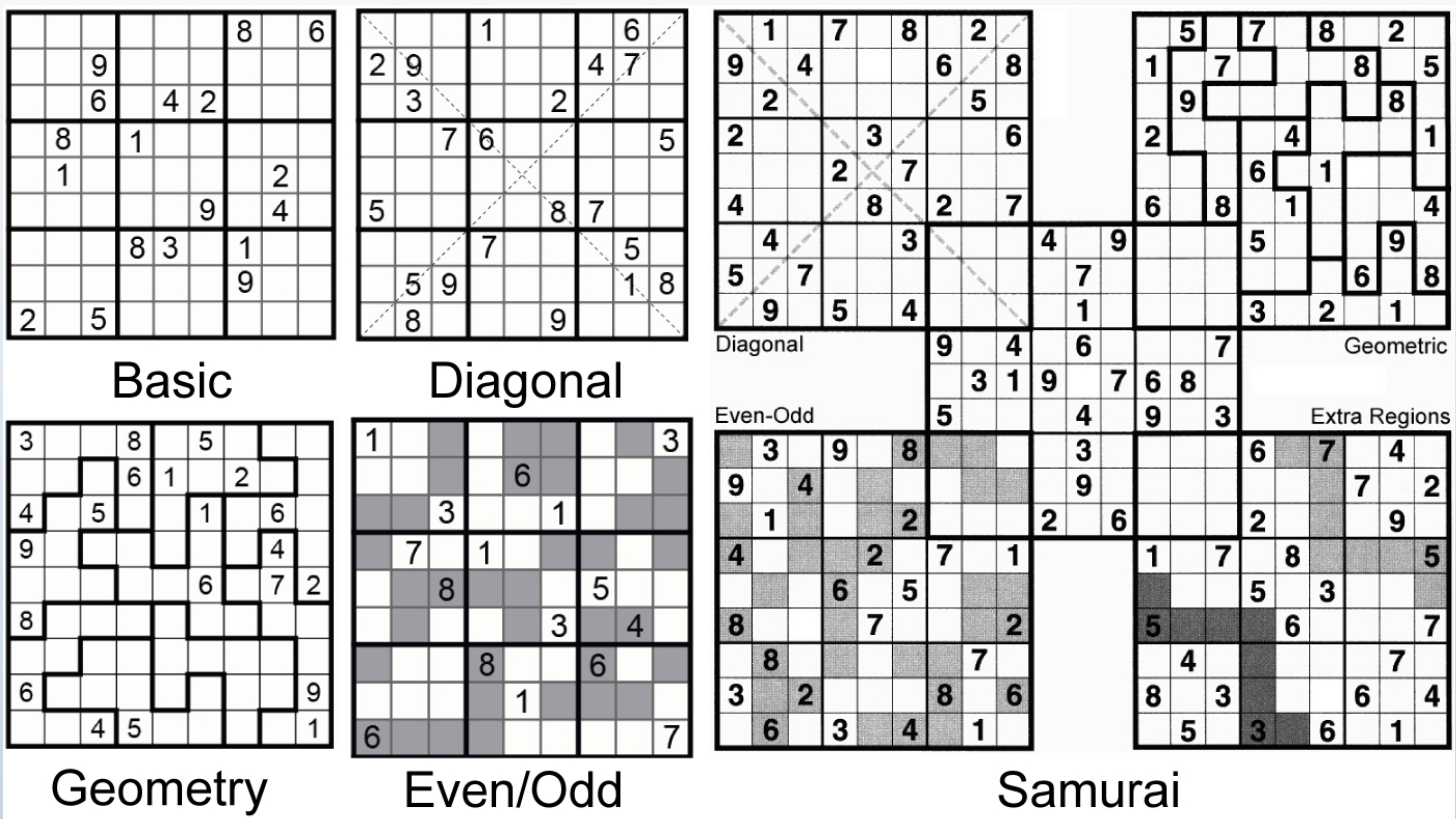
- Variations of Sudoku are played throughout the world
- System that can <u>*easily*</u> solve any variation
  - Can figure out the type of puzzle
  - Easy to add new varieties
  - Leverage techniques in CSP solving
- Can be accomplished using *model refinement*



Some Sudoku Puzzles

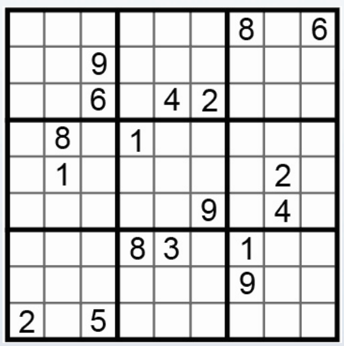
Model Refinement

Sudoku Puzzles

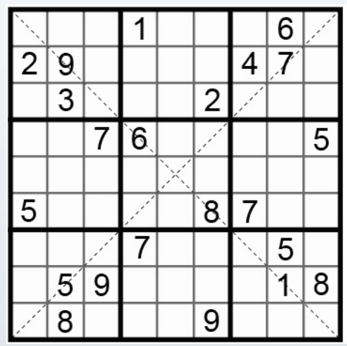


Case Study: Sudoku Puzzles

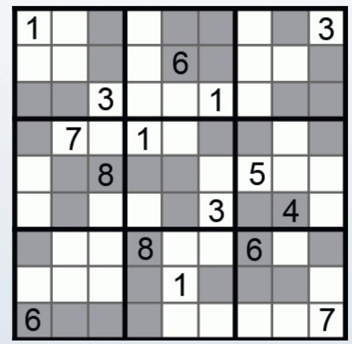
- 100 instances of easy, medium, hard difficulty levels for all puzzle types
- Magic puzzle instances have same difficulty level



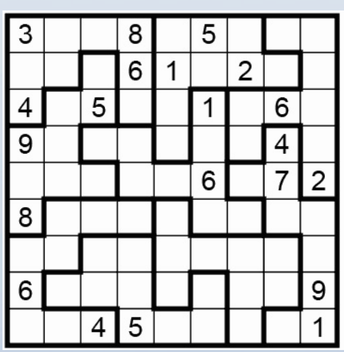
Basic



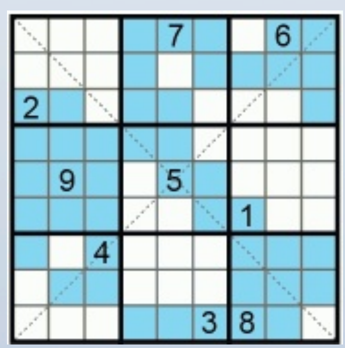
Diagonal



Even/Odd



Geometry



Magic

Selecting Constraints: Evaluation

Sudoku Puzzles

	$\lvert C_{new} \rvert$	Easy Rec.	Easy Prec.	Medium Rec.	Medium Prec.	Hard Rec.	Hard Prec.
Basic	3	1.0	0.88	1.0	0.87	1.0	0.87
Geometry	3	1.0	0.86	1.0	0.88	1.0	0.88
Diagonal	4	0.86	1.0	0.86	1.0	0.85	1.0
Even/Odd	4	1.0	0.93	1.0	0.94	1.0	0.95
Magic	5	(not categorized): Rec.: 0.81, Prec.: 1.0					

Recall: #correctly inferred cons. / total # cons.
Precision: # correctly inferred cons./ total # *inferred* cons.

Constraint Propagation Evaluation

Sudoku Puzzles: New Points

	Easy					Medium					Hard				
	Initial	AC	GAC	SAC	All	Initial	AC	GAC	SAC	All	Initial	AC	GAC	SAC	All
Basic	27	30	64	78	81	27	30	74	76	81	28	32	47	79	80
Geometry	28	32	51	78	81	27	30	71	76	80	27	31	45	79	80
Diagonal	22	22	25	23	25	22	22	25	23	25	22	22	26	23	26
Even/Odd	15	16	16	16	16	15	15	15	15	15	15	15	15	15	15

(a) Categorized puzzles

	Uncategorized				
	Initial	AC	GAC	SAC	All
Magic	9	9	9	9	9

(b) Magic puzzle type

Constraint Propagation Evaluation

Sudoku Puzzles: Inferred Models

	$\lvert C_G \rvert$	$\lvert C_{new} \rvert$	Easy		Medium		Hard	
			Rec.	Prec.	Rec.	Prec.	Rec.	Prec.
Basic	2	3	1.0	0.99	1.0	1.0	1.0	0.99
Geometry	2	3	1.0	1.0	1.0	1.0	1.0	0.99
Diagonal	2	4	0.89	1.0	0.89	1.0	0.88	1.0
Even/Odd	2	4	1.0	0.93	1.0	0.94	1.0	0.94
Magic	2	5	(not categorized): Rec.: 0.81, Prec.: 1.0					

Performance Improvement

Sudoku

	Easy		Medium		Hard	
	% solved	% one sol.	% solved	% one sol.	% solved	% one sol.
Basic	99%	100%	100%	100%	99%	100%
Geometry	100%	100%	100%	100%	99%	100%
Diagonal	100%	57%	100%	56%	100%	53%
Even/Odd	69%	100%	74%	100%	76%	100%
Magic	(¬ categorized): % solved: 100% % one sol.: 10%					

% solved: percentage of instances with a solution(s)
% one sol.: percentage of solved instances with a single solution*

*all puzzle instances are well-formed (a single solution)