



Introducing Relational^{AI}

Probabilistic Graphical Models –and–
Statistical Relational Learning

Molham Aref -- Relational^{AI}

OVERVIEW

Age: 9 months old AI/ML startup

Team:

- 16 people, 10 PhD's, 4 former university professors
- Faculty Network: 9 @ ~20% + summers, additional ~12+ in extended network

Location: distributed team (SV, Atlanta, Seattle, Toronto, NYC, Utrecht, London)
Offices in Berkeley and Utrecht

Financials: Funded through 2019 + revenue from consulting

Industries served: TBD (looking at Financial Services, Business Intelligence)

- If all else fails we're going to pivot to Deep Quantum Crypto Blockchain of Things and hope for the best

HOW WE SEE OURSELVES

Expertise in systems (8 people)

- We built sophisticated compilers & interpreters (OOPSLA, ECOOP, ICFP)
- We built databases that advanced the state of the art (SIGMOD, VLDB)

Expertise in theory (7 people)

- We prove things (PODS, ICDT, TODS, JACM, POPL)
- We designed first-of-a-kind asymptotically efficient algorithms
 - Worst case optimal join algorithms
 - Asymptotically superior query plans

Expertise in ML & AI (7 people)

- We implemented scalable ML & probabilistic systems (NIPS, ICML, AAAI)
- We developed sophisticated statistical and mathematical models

INDUSTRIAL EXPERIENCE

HNC

- Financial services: Fraud detection -- credit card, insurance (23 of top 25 credit card issuing banks)
- Neural networks on proprietary HW accelerators (also computational intelligence, Database mining)
- IPO 1995, acquired by FICO in 2002

Retek

- Retail: Demand forecasting, supply chain optimization, pricing (majority of Retail Global 250)
- Time series with approximate optimization – first hierarchical forecasting solution to scale to Retail volumes
- IPO 1999, acquired by Oracle in 2005

Optimi

- Telecom: Wireless network optimization (AT&T, Cingular, Next, America Movil, Telefonica)
- Monte Carlo simulation, heuristic search (simulated annealing)
- Acquired by Ericsson in 2010

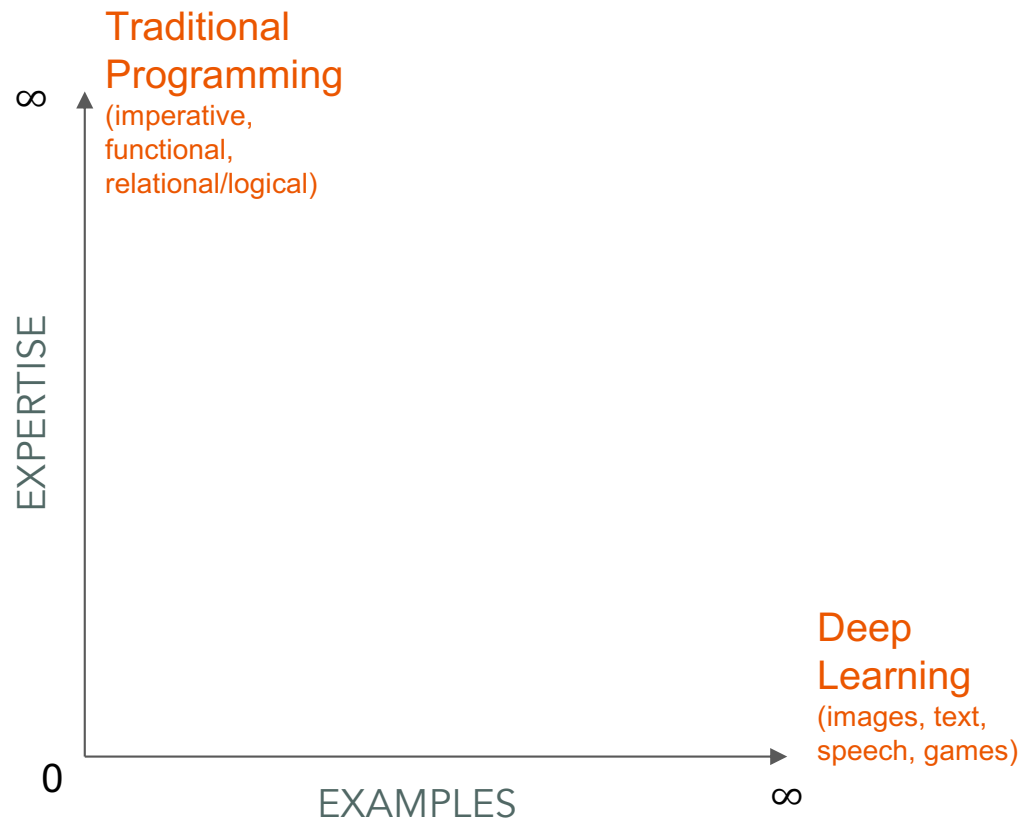
Brickstream

- Retail: in-store video analytics (>30% market share globally)
- Old-school computer vision (pre-deep learning) – first industrial use of stereo cameras
- Acquired by Point-Grey which was acquired by FLIR in 2015

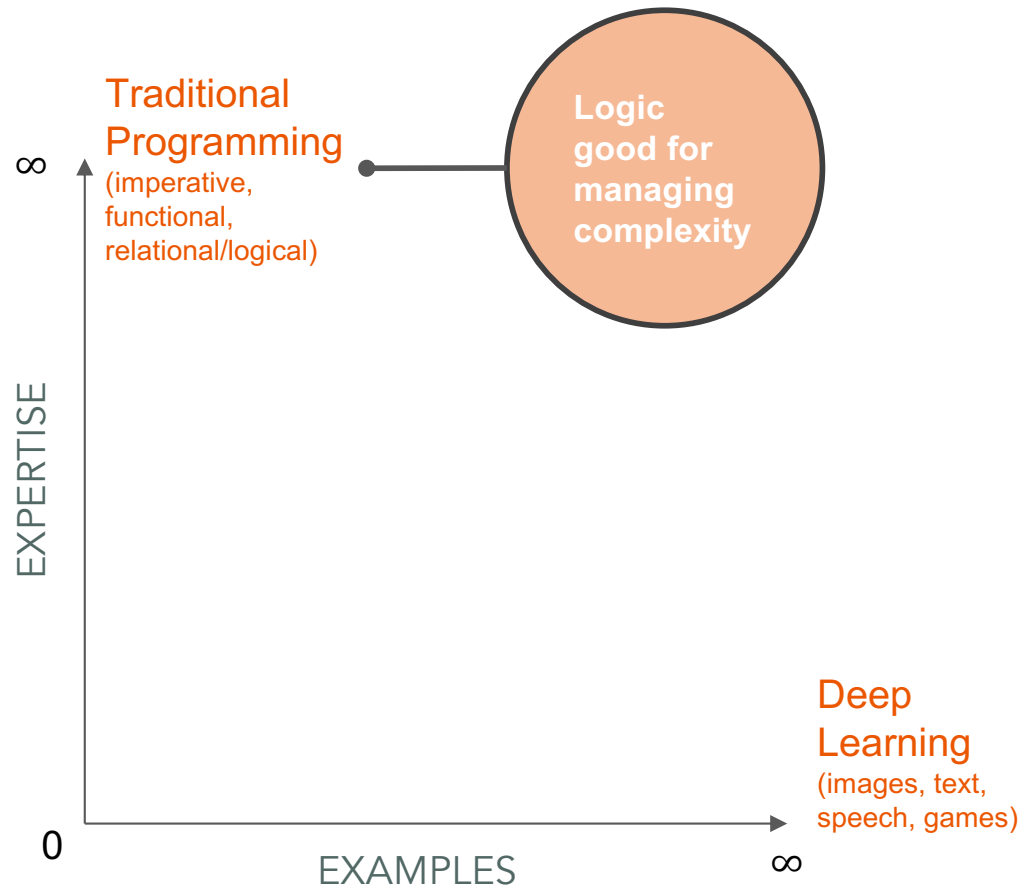
LogicBlox/Predictix

- Retail: Demand forecasting, supply chain optimization, pricing (3 of top 6 US retailers + dozen large global retailers)
- Factorization machines, linear programming, integer programming – first on cloud
- Acquired by Infor in 2016

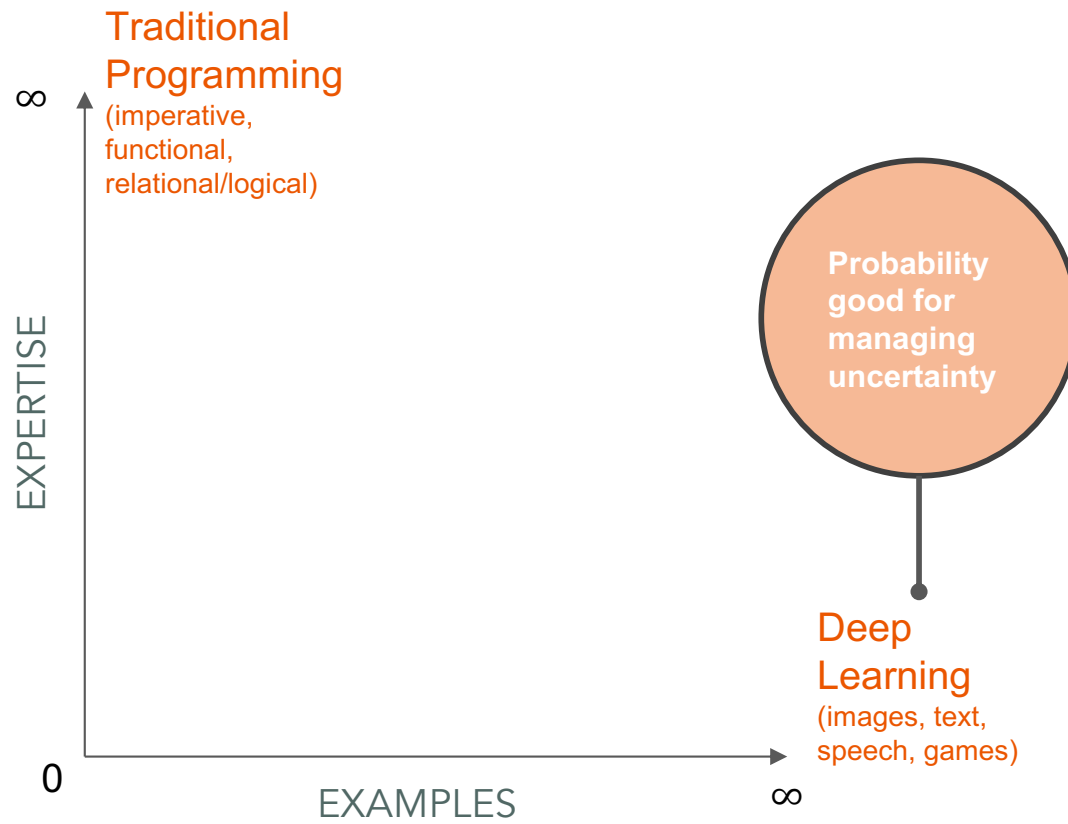
STATE OF THE PRACTICE



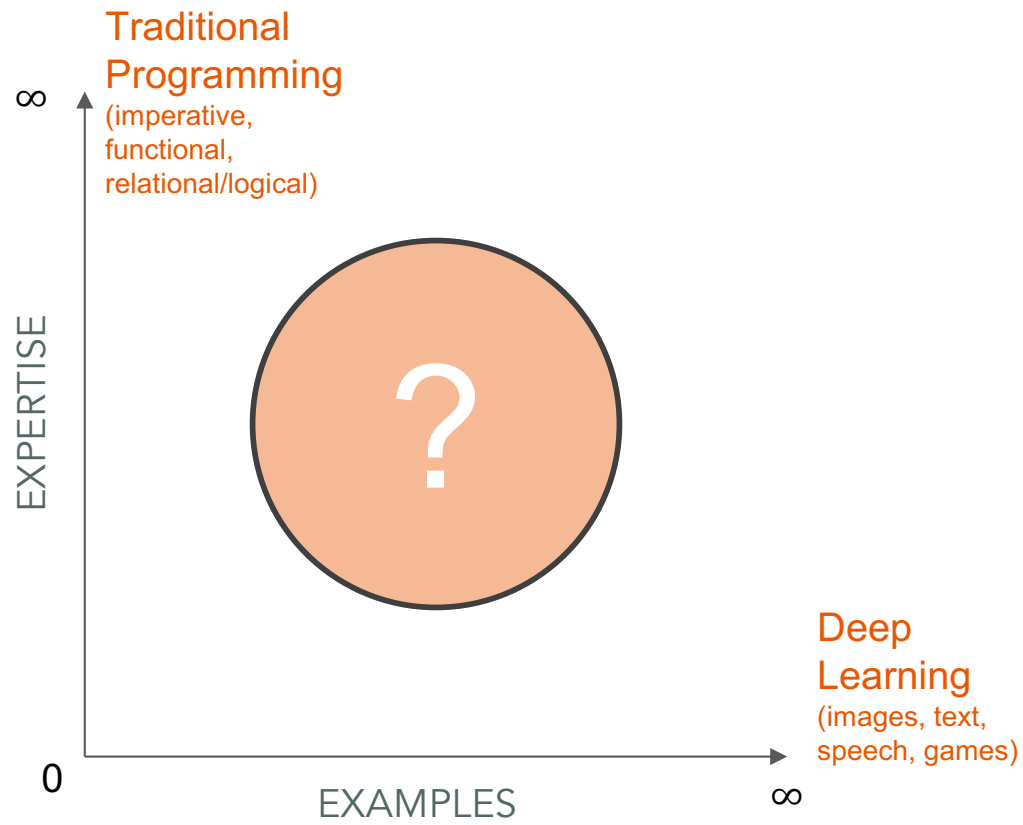
STATE OF THE PRACTICE



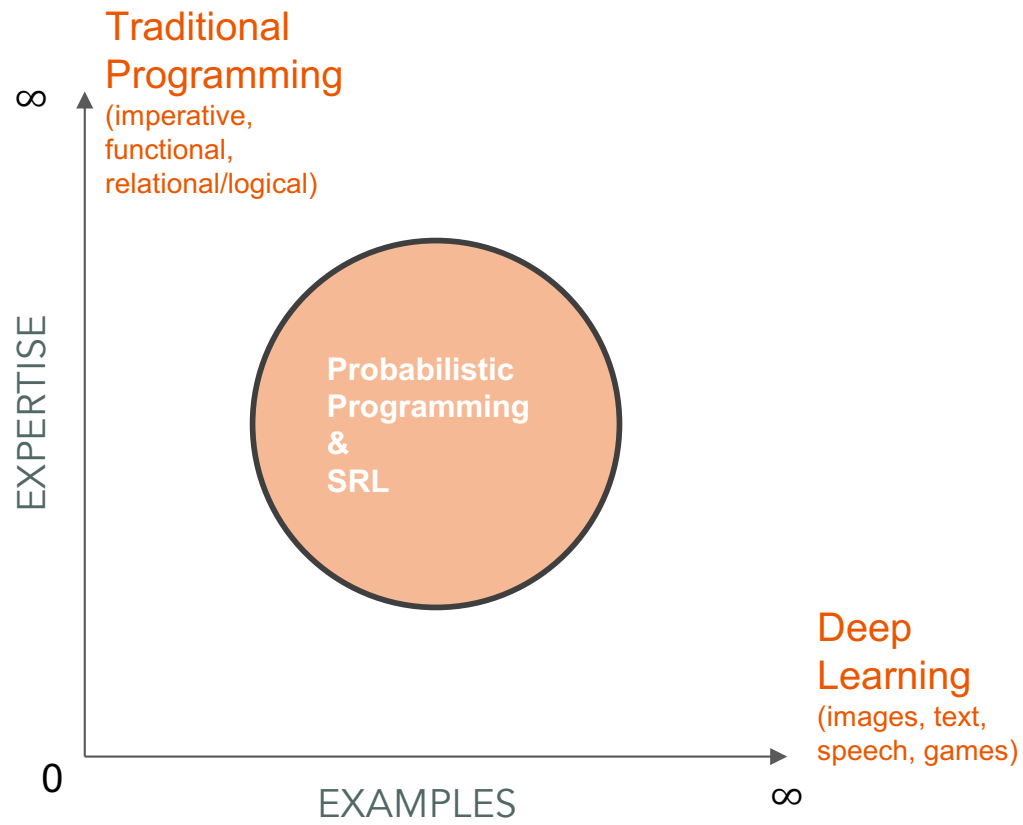
STATE OF THE PRACTICE



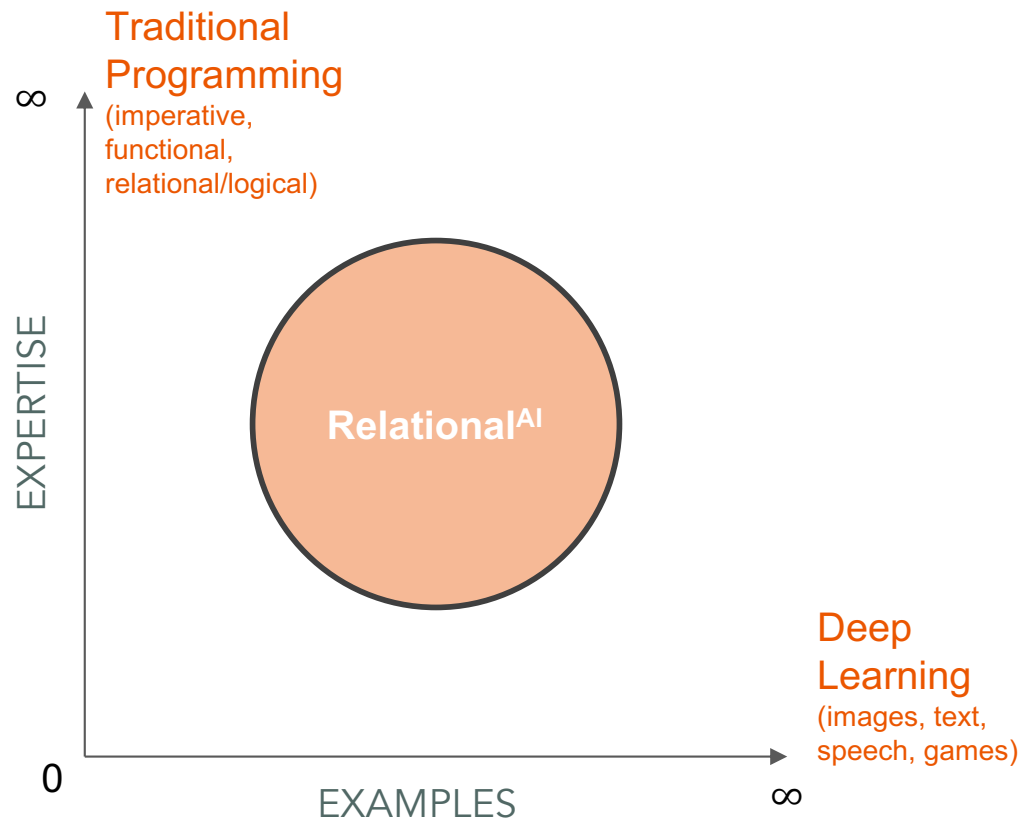
STATE OF THE PRACTICE



STATE OF THE PRACTICE



STATE OF THE PRACTICE



OUR SECRET SAUCE

- We know how to exploit problem structure to make optimization asymptotically faster
 - We know how to perform stochastic gradient descent and batch gradient descent directly on normalized relational data
 - Most ML Methods can be solved well with SGD, some with BGD
- Asymptotically faster optimization means anything that depends on it can go faster
 - Learning/Parameter optimization
 - Hyperparameter optimization
 - Feature engineering
 - Structure learning
 - Inference
 - ...

STRUCTURE FOR TRADITIONAL/DISCRIMINATIVE ML

5 COMPONENTS OF MACHINE LEARNING

- Method (or model class)
 - e.g. FM, decision tree, neural network, ...
- Loss (error) function
 - e.g. Absolute error (L1 norm), Square error (L2 norm), ...
- Generalization mechanism
 - e.g. Regularization (norm * penalty), cross validation
- Evaluation function
 - Takes the model parameters and the input and produces a prediction
- Optimizer
 - E.g. Gradient descent, EM algorithm

5 TYPES OF METHODS or MODEL CLASSES

- Regression: predict a number
 - Linear regression with VIF
 - LASSO regression
 - Multi-time series prediction
 - Nonparametric regression
 - Mixture of experts
 - Factorization machines
 - Polynomial regression
- Classification: predict a category
 - Naïve Bayes classifier
 - Non-parametric Bayes classifier
 - K-nearest neighbor classifier
 - Support vector machine
 - Decision tree
 - Hidden Markov model
- Density Estimation: find likelihood of objects
 - Histograms
 - Kernel density estimation
- Clustering: find natural groups
 - K-means
 - Spectral clustering
 - Mean shift clustering
- Dimension Reduction: combine features
 - Singular value decomposition
 - Maximum variance unfolding
 - Non-negative matrix factorization
 - Kernel principal component analysis
 - (Ensemble) Singular value decomposition
 - GROUSE
 - Random projections
 - Tensor factorization PARFAC/CANDECOMP

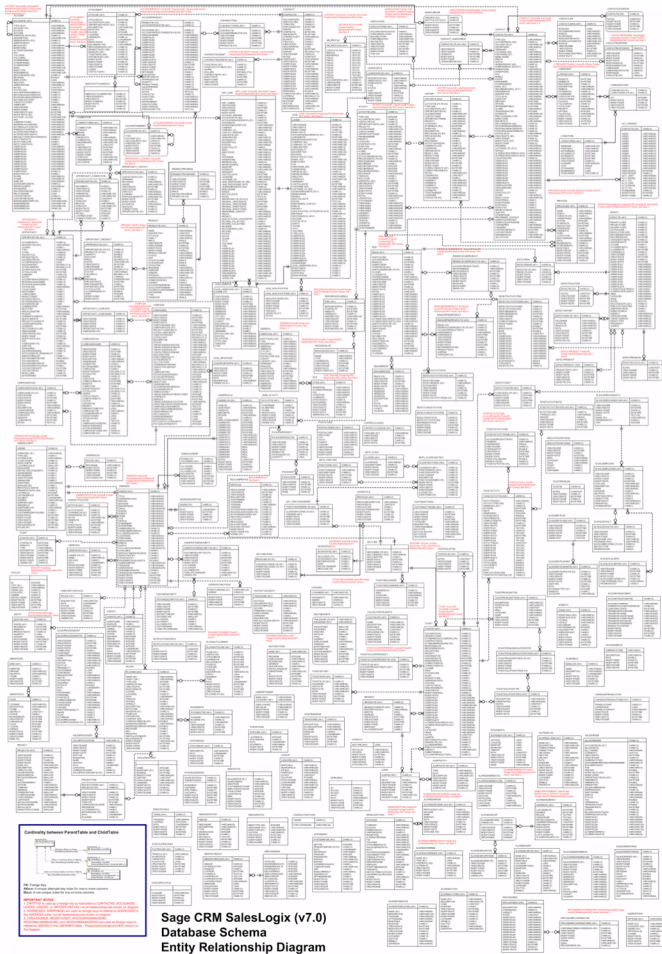
DESIGN MATRIX

Features

Entities

ID	x1	x2	x3	...	y
i_1					
i_2					
...					
i_{1B}					

DESIGN MATRIX IS A VIEW ON STRUCTURED (RELATIONAL) DATA



Entities

Features

ID	x1	x2	x3	...	y

WE CAN EXPLOIT THE RELATIONAL STRUCTURE

We use algebraic structure (e.g. semi-rings) to “push the aggregations through the joins” to implement lifted stochastic and batch gradient descent for efficient learning of a variety of model classes

- Linear regression
- Polynomial regression
- Factorization machines
- Decision trees
- Neural nets
- ...

(many more on the way)

BENCHMARK (1/3)

		v ₁	v ₂	v ₃	v ₄
Join Representation (#values)	Listing	774M	3.614G	3.614G	3.614G
	Factorized	37M	169M	169M	169M
Compression	Fact/List	20.9×	21.4×	21.4×	21.4×
Join Computation (PSQL) for R, TensorFlow, libFM		50.63	216.56	216.56	216.56
Factorized Computation of 43 Counts over Join		8.02	34.15	34.15	34.15
Linear regression					
Features (continuous+categorical)	without FDs	33 + 55	33+55	33+1340	33+2702
	with FDs	same as above, there are no FDs			33+2653
Aggregates (scalar+group-by)	without FDs	595+2,418	595+2,421	595+111,549	595+157,735
	with FDs	same as above, there are no FDs			595+144,589
MADLib (ols)	Learn	1,898.35	8,855.11	> 79,200.00	-
R (QR)	Export/Import	308.83	-	-	-
	Learn	490.13	-	-	-
TensorFlow (FTLR) (1 epoch, batch size 1K)	Export/Import	74.72	372.70	372.70	372.70
	Learn	2762.50	12710.53	12724.94	12708.11
F	Aggregate	93.31	424.81	OOM	OOM
	Converge (runs)	0.01 (359)	0.01 (359)		
AC/DC	Aggregate	25.51	116.64	117.94	895.22
	Converge (runs)	0.02 (343)	0.02 (367)	0.42 (337)	0.66 (365)
AC/DC+FD	Aggregate	same as AC			380.31
	Converge (runs)	there are no FDs			8.82 (366)
Speedup of AC/DC+FD over	MADlib	74.36×	75.91×	> 669.14×	∞
	R	33.28×	∞	∞	∞
	TensorFlow	113.12×	114.01×	112.49×	34.17×
	F	3.65×	3.64×	∞	∞
	AC/DC	same as AC/DC, there are no FDs			2.30×

BENCHMARK (2/3)

		v ₁	v ₂	v ₃	v ₄
Join Representation (#values)	Listing	774M	3.614G	3.614G	3.614G
	Factorized	37M	169M	169M	169M
Compression	Fact/List	20.9×	21.4×	21.4×	21.4×
Join Computation (PSQL) for R, TensorFlow, libFM		50.63	216.56	216.56	216.56
Factorized Computation of 43 Counts over Join		8.02	34.15	34.15	34.15
Polynomial regression degree 2					
Features (continuous+categorical)	without FDs	562+2,363	562+2,366	562+110,209	562+154,033
	with FDs	same as above, there are no FDs			562+140,936
Aggregates (scalar+group-by)	without FDs	158k+742k	158k+746k	158k+65,875k	158k+46,113k
	with FDs	same as above, there are no FDs			158k+36,712k
MADlib (ols)	Learn	> 79,200.00	> 79,200.00	> 79,200.00	-
AC/DC	Aggregate	132.43	517.40	820.57	7,012.84
	Converge (runs)	3.27 (321)	3.62 (365)	349.15 (400)	115.65 (200)
AC/DC+FD	Aggregate	same as AC/DC			1,819.80
	Converge (runs)	there are no FDs			219.51 (180)
Speedup of AC/DC+FD over AC/DC	MADlib	> 583.64×	> 152.01×	> 67.71×	∞
	AC/DC	same as AC/DC , there are no FDs			3.50×

BENCHMARK (3/3)

		v ₁	v ₂	v ₃	v ₄
Join Representation (#values)	Listing	774M	3.614G	3.614G	3.614G
	Factorized	37M	169M	169M	169M
Compression	Fact/List	20.9×	21.4×	21.4×	21.4×
Join Computation (PSQL) for R, TensorFlow, libFM		50.63	216.56	216.56	216.56
Factorized Computation of 43 Counts over Join		8.02	34.15	34.15	34.15
Factorization machine degree 2 rank 8					
Features (continuous+categorical)	without FDs	530+2,363	530+2,366	530+110,209	530+154,033
	with FDs	same as above, there are no FDs			562+140,936
Aggregates (scalar+group-by)	without FDs	140k+740k	140k+744k	140k+65,832k	140k+45,995k
	with FDs	same as above, there are no FDs			140k+36,595k
libFM (MCMC)	Export/Import	412.84	1462.54	3,096.90	3,368.06
	Learn (runs)	19,692.90 (300)	103,225.50 (300)	79,839.13 (300)	87,873.75 (300)
AC/DC	Aggregate	128.97	498.79	772.42	6,869.47
	Converge (runs)	3.03 (300)	3.05 (300)	262.54 (300)	166.60 (300)
AC/DC+FD	Aggregate	same as AC/DC			1,672.83
	Converge (runs)	there are no FDs			144.07 (300)
Speedup of AC/DC+FD over	libFM	152.70×	209.03×	80.34×	50.33×
	AC/DC	same as AC/DC , there are no FDs			3.87×

STATISTICAL RELATIONAL LEARNING

SRL and StarAI

MOTIVATION

- Graphical models are considered by some to be “one of the most exciting advances in machine learning (AI, signal processing, coding, control, ...) in the last decades”
- Graphical models allow us to gain global insight based on local observations
- There are different types of graphical models
 - Directed: eg. Bayesian Networks (aka belief networks)
 - Undirected: e.g. Mark Networks (aka Markov Random Fields), Factor Graphs
 - Mixed: e.g. Chain Graphs - both directed acyclic graphs and undirected graphs are special cases of chain graphs, which can therefore provide a way of unifying and generalizing Bayesian and Markov networks
- Statistical Relational models generalize PGM's in the same way that first order logic generalizes propositional logic – they allow us to quantify over individuals/entities

STATISTICAL RELATIONAL LEARNING (SRL)

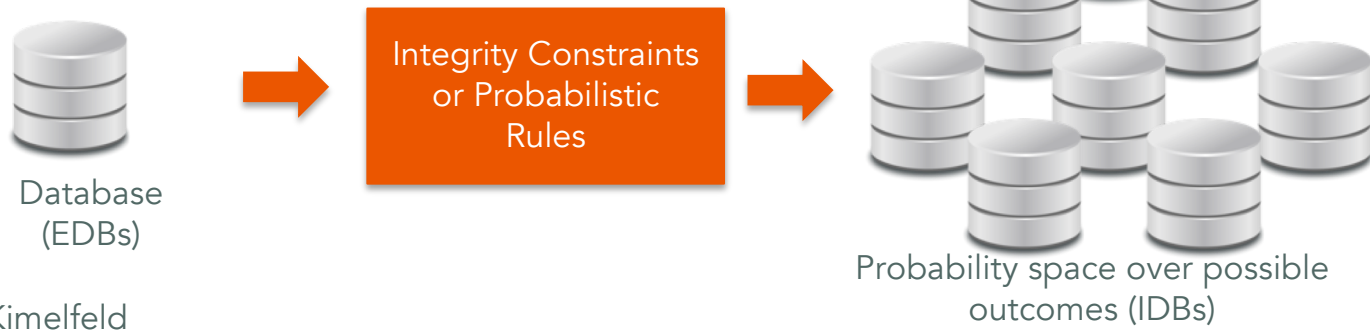
- Knowledge is represented as a distribution over possible worlds
 - Finite and infinite sets of possible worlds are supported
 - Undirected models: via integrity constraints
 - We specify the constraints that determine the legal set of possible worlds & a function to score each of them.
 - We don't have to provide a program to generate each possible world.
 - Directed models: via probabilistic programs
 - We have to provide a program to generate each possible world. Score by normalizing - frequency of a given world relative to all others.
 - Normalize the score of each world by the sum of scores of all the worlds
- Inference
 - Unlike "traditional" methods where prediction is the input applied to the parameters of the model class, inference in SRL requires expensive optimization or (approximate) integration over possible worlds
- Learning
 - Unlike traditional learning algorithms, just one instance to learn from (the relational DB)
 - Structure learning uses inference during each step

SEMANTICS

Ordinary minimal model semantics:



Distribution semantics



Slide thanks to Benny Kimelfeld

UNDIRECTED MODELS

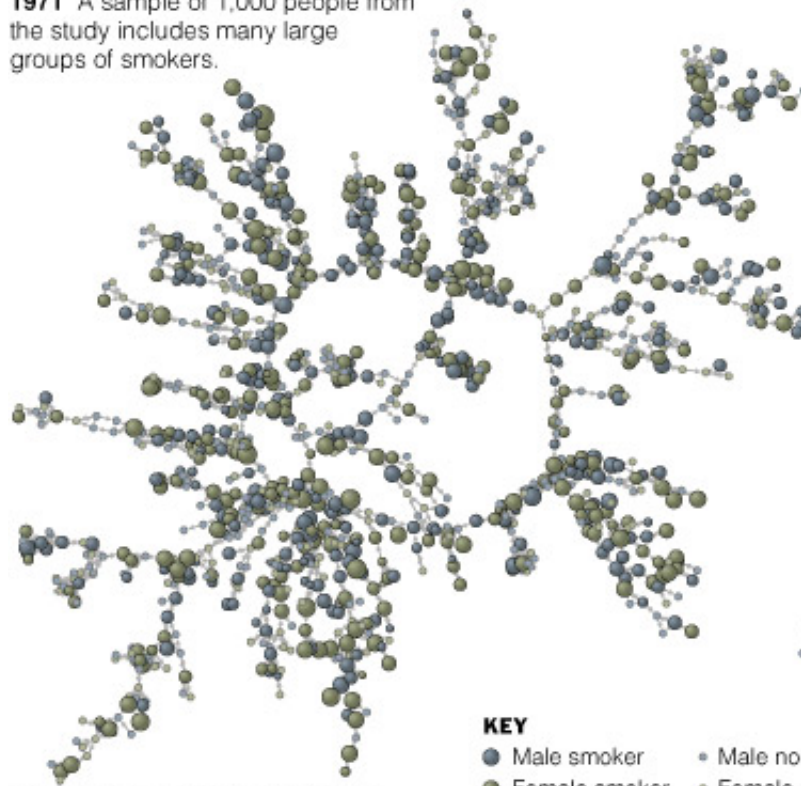
Use Integrity Constraints to specify a set of possible worlds & define a scoring function for each

SMOKERS AND FRIENDS

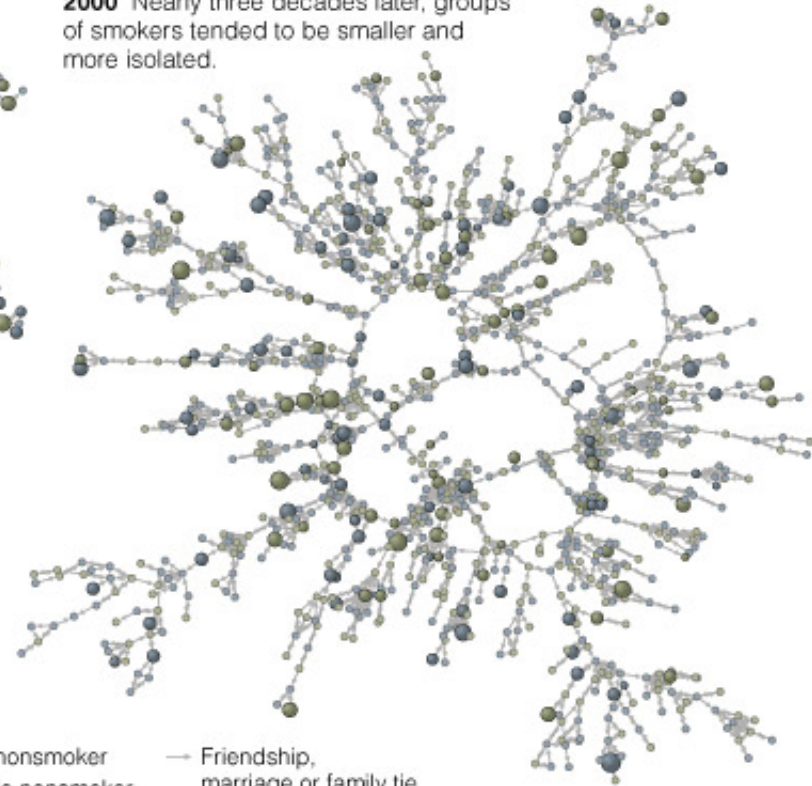
Smoking and Quitting in Groups

Researchers studying a network of 12,067 people found that smokers and nonsmokers tended to cluster in groups of close friends and family members. As more people quit over the decades, remaining groups of smokers were increasingly pushed to the periphery of the social network.

1971 A sample of 1,000 people from the study includes many large groups of smokers.



2000 Nearly three decades later, groups of smokers tended to be smaller and more isolated.



KEY

- Male smoker
- Female smoker
- Male nonsmoker
- Female nonsmoker
- Friendship, marriage or family tie

Circle size is proportional to the number of cigarettes smoked per day.

Sources: *New England Journal of Medicine*; Dr. Nicholas A. Christakis; James H. Fowler

THE NEW YORK TIMES

CERTAIN KNOWLEDGE WITH INTEGRITY CONSTRAINTS

A logical Knowledge Base is a set of Integrity Constraints that define a set of possible worlds:

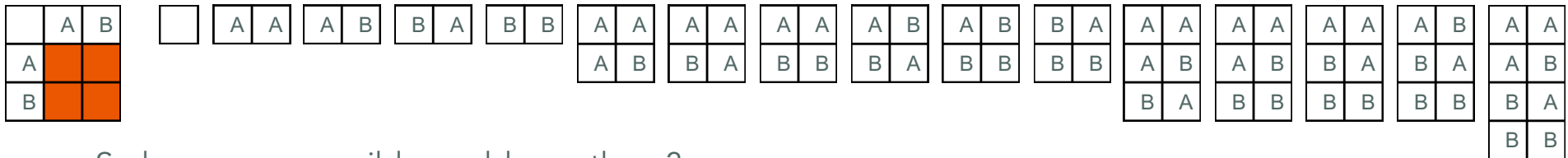
```

person(x)
smokes(x) -> person(x)
cancer(x) -> person(x)
friends(x, y) -> person(x), person(y)
    
```

Assuming persons Alice (A) and Bob (B), then there are 4 possible relations for each of:
 smokes, cancer:



There are 16 possible relations for friends



So how many possible worlds are there?

- 2 bits for for each of smokes cancer and 4 bits for friends:

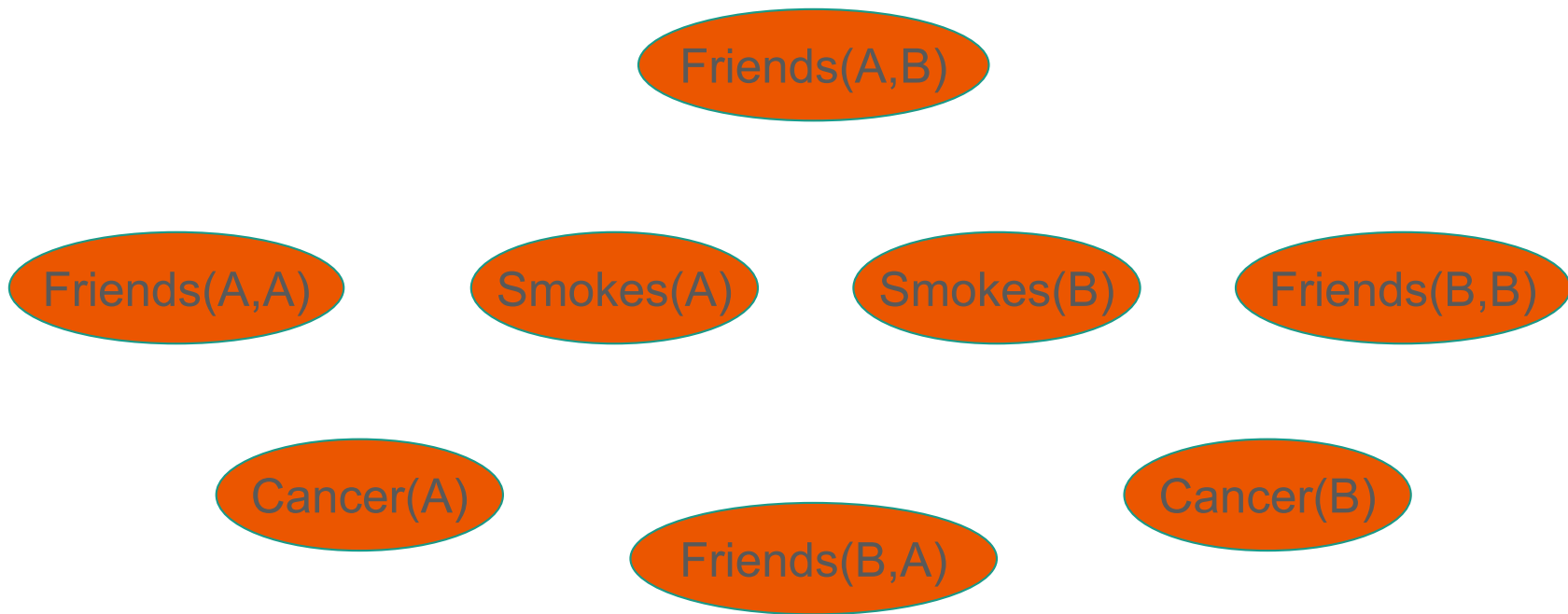
2⁸ or 256 possible worlds

And how many if we add a 3rd person Carla (C)?

- 3 bits * 2 unary relations + 9 bits for binary relation:

2¹⁵ or 32K possible worlds

EACH POSSIBLE TUPLE IS A NODE IN A GRAPHICAL MODEL



PROBABILISTIC KNOWLEDGE WITH WEIGHTED INTEGRITY CONSTRAINTS

Smoking causes cancer
Friends have similar smoking habits

Represent probabilistic knowledge with soft (weighted) Integrity Constraints

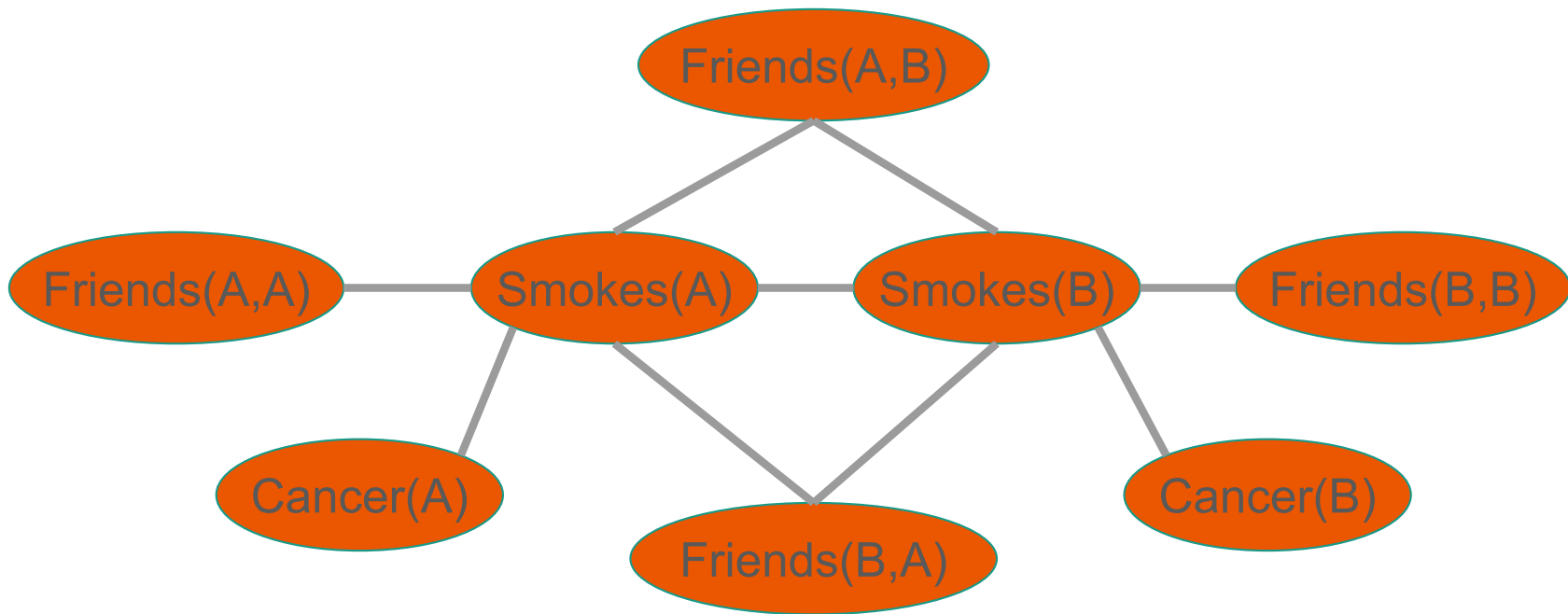
```
person(x)
smokes(x) -> person(x)
cancer(x) -> person(x)
friends(x, y) -> person(x), person(y)
w1 smokes(x) -> cancer(x)
w2 smokes(x), friends(x, y) -> smokes(y)
```

When a world violates a formula, it becomes less probable, not impossible

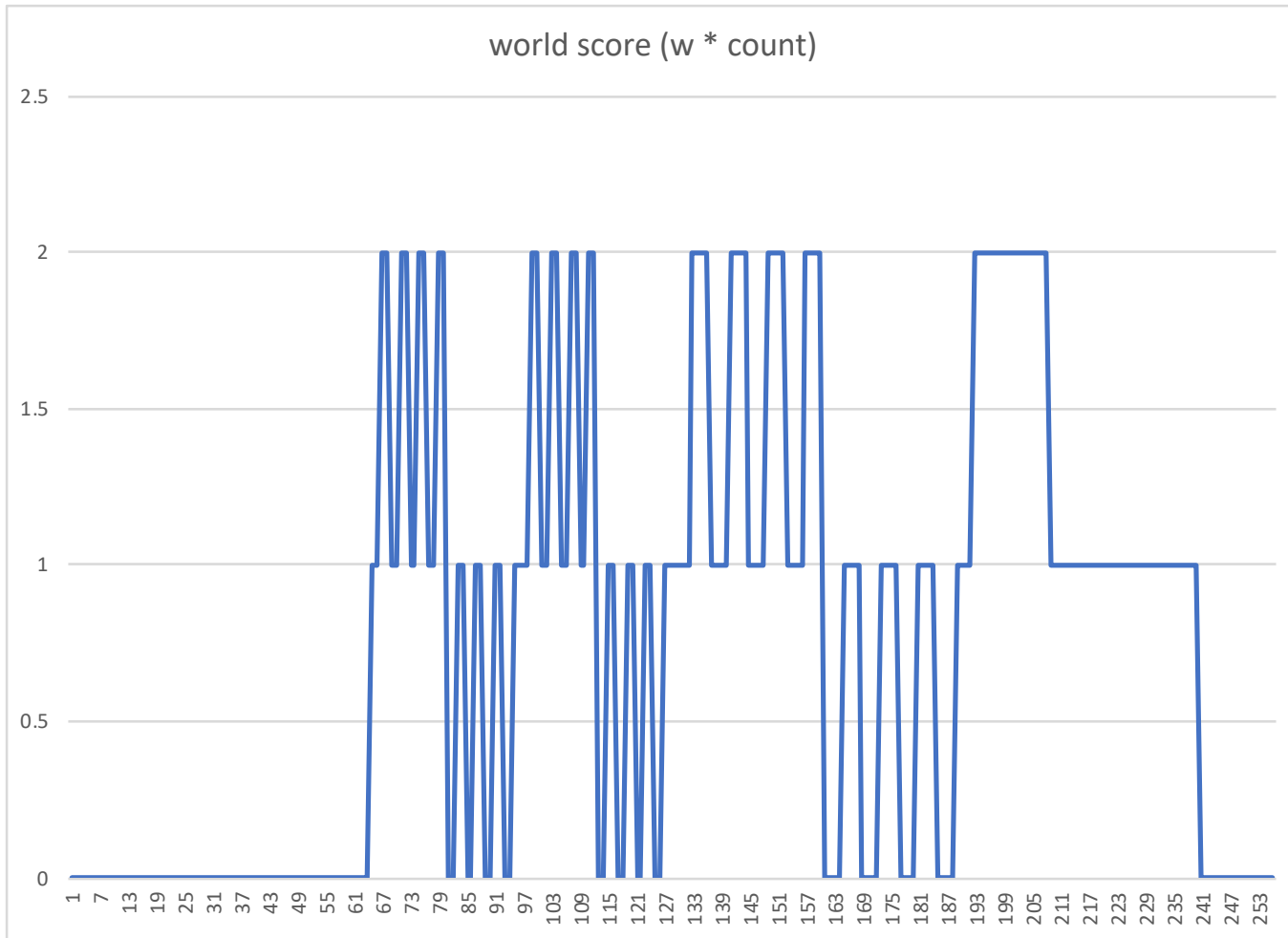
Weights give indication of certainty in domain knowledge or expertise

- 0 -> no confidence -- might as well not have the IC
- Infinity -> absolute certainty – hard constraint
- -Infinity -> absolute certainty in the converse – hard constraint on negated IC

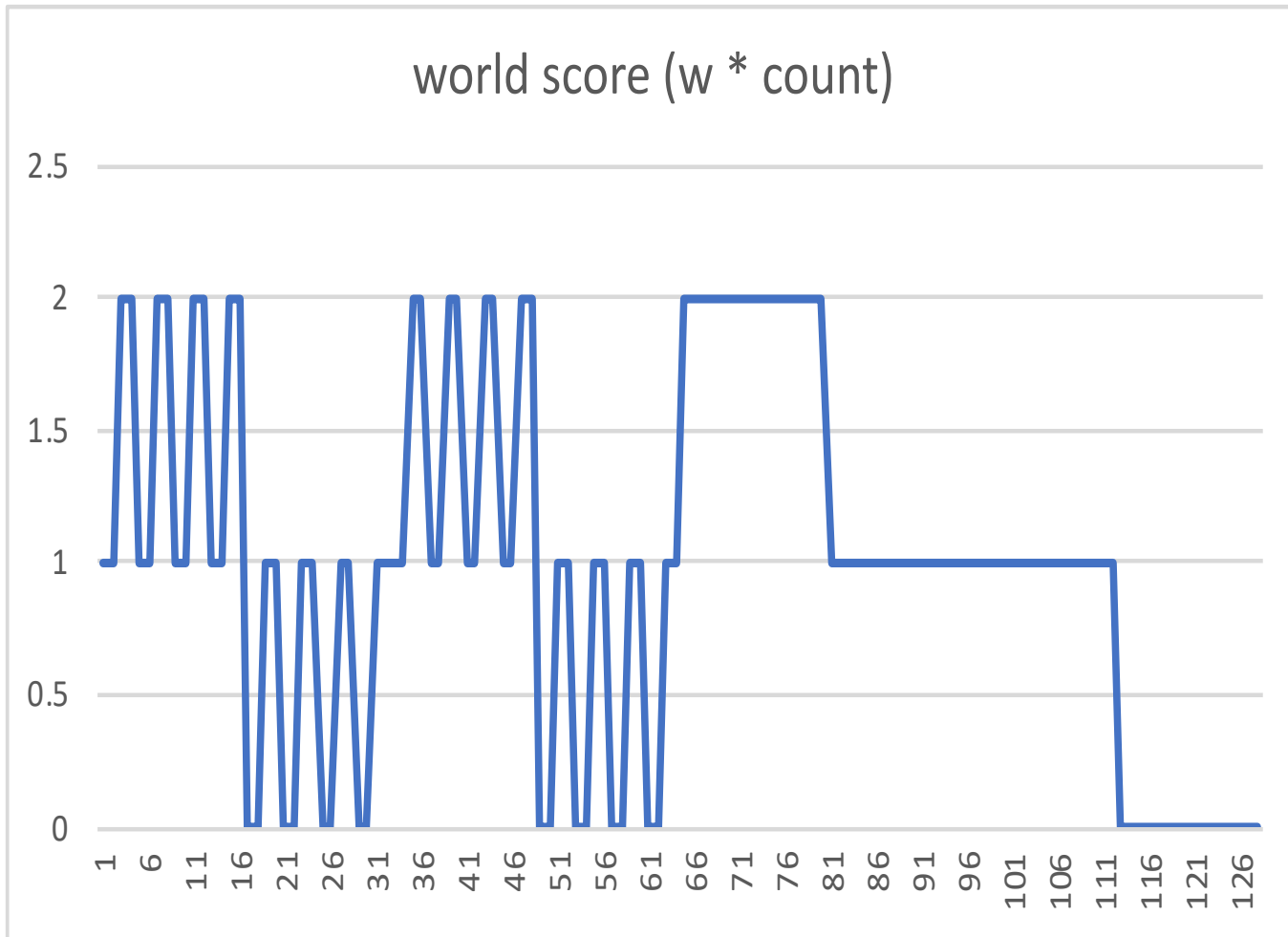
GRAPHICAL MODEL



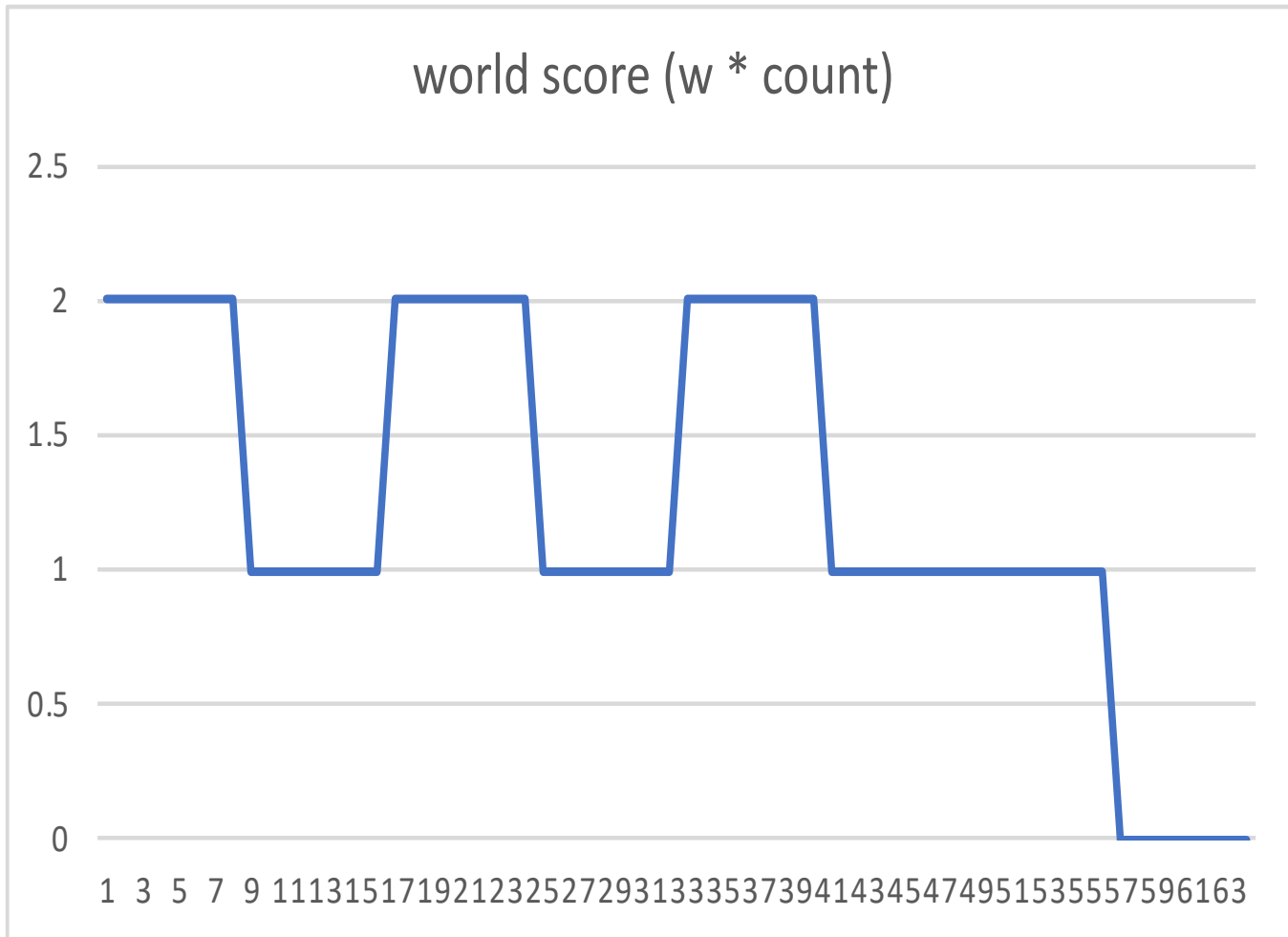
ALL WORLDS



ALL WORLDS where B smokes



ALL WORLDS where B smokes and B is friends with A



QUANTIFYING OVER POSSIBLE WORLDS

Observations/measurements eliminate some of the possible worlds

Finding the mostly likely world can be computed with optimization

Computing the probability of any world requires us to aggregate/integrate over all possible worlds.

.

HOW DO WE MAKE SRL EFFICIENT?

There are 2 important dimensions to consider

- Brawn (i.e. the constant factors)
 - Latency hiding: memory hierarchy and network latencies (e.g. in memory)
 - Specialization: specialize for workload (e.g. JIT compilation), specialize for data
 - Parallelization: SIMD, multi-core, accelerators (e.g. GPU, TPU), in memory computing
- Brain (i.e. the asymptotics)
 - Lifting and Structure exploitation: algebraic (e.g. semi rings, groups), combinatorial, statistical, geometric
 - Approximation (with error bars): e.g. variational methods

both ---> approximate lifted inference

SUMMARY: ADVANTAGES OF (STATISTICAL) RELATIONAL AI

- Performance
 - Exploits the relational structure for asymptotically better performance
- Understandability
 - Declarative relational language can be used to codify knowledge/expertise (**human to machine**) and to return insight (**machine to human**)
- Quality
 - Fewer assumptions regarding independence, identical distributions, # of observations per example, etc. can produce more accurate models
- Versatility
 - Generalized inference: from observations to unknowns in any direction

WORST-CASE OPTIMAL MULTI-WAY JOIN

- Worst-Case Optimal Join Algorithms: Techniques, Results, and Open Problems. Ngo. (**Gems of PODS 2018**)
- Worst-Case Optimal Join Algorithms: Techniques, Results, and Open Problems. Ngo, Porat, Re, Rudra. (**Journal of the ACM 2018**)
- What do Shannon-type inequalities, submodular width, and disjunctive datalog have to do with one another? Abo Khamis, Ngo, Suciu, (PODS 2017 - **Invited to Journal of ACM**)
- Computing Join Queries with Functional Dependencies. Abo Khamis, Ngo, Suciu. (PODS 2017)
- Joins via Geometric Resolutions: Worst-case and Beyond. Abo Khamis, Ngo, Re, Rudra. (PODS 2015, **Invited to TODS 2015**)
- Beyond Worst-Case Analysis for Joins with Minesweeper. Abo Khamis, Ngo, Re, Rudra. (PODS 2014)
- Leapfrog Triejoin: A Simple Worst-Case Optimal Join Algorithm. Veldhuizen (ICDT 2014 - **Best Newcomer**)
- Skew Strikes Back: New Developments in the Theory of Join Algorithms. Ngo, Re, Rudra. (**Invited to SIGMOD Record 2013**)
- Worst Case Optimal Join Algorithms. Ngo, Porat, Re, Rudra. (PODS 2012 – **Best Paper**)

Worst-case Optimal Join Algorithms

Leapfrog Triejoin: A Simple, Worst-Case Optimal Join Algorithm

Worst-case Optimal Algorithms for Conjunctive Queries with Functional Dependencies

What do Shannon-type Inequalities, Submodular Width, and Disjunctive Datalog have to do with one another?

Worst-case Optimal Join Algorithms

HUNG Q. NGO, University at Buffalo, SUNY
ELY PORAT, Bar-Ilan University
CHRISTOPHER RÉ, Stanford University
ATRI RUDRA, University at Buffalo, SUNY

Efficient join processing is one of the most fundamental and well-studied tasks in database research. In this work, we examine algorithms for natural join queries over many relations and describe a new algorithm to process these queries optimally in terms of worst-case data complexity. Our result builds on recent work by Aterias, Grohe, and Marx, who gave bounds on the size of a natural join query in terms of the sizes of the individual relations in the body of the query. These bounds, however, are not constructive; they rely on Shearer's entropy inequality, which is information-theoretic. Thus, the previous results leave open the question of whether there exist algorithms whose runtimes achieve these optimal bounds. An answer to this question may be interesting to database practice, as we show in this article that any project-join style plans, such as ones typically employed in a relational database management system, are asymptotically slower than the optimal for some queries. We present an algorithm whose runtime is worst-case optimal for all natural join queries. Our result may be of independent interest, as our algorithm also yields a constructive proof of the general fractional cover bound by Aterias, Grohe, and Marx without using Shearer's inequality. This bound implies two famous inequalities in geometry: the Loomis-Whitney inequality and its generalization, the Bollobás-Thomason inequality. Hence, our results algorithmically prove these inequalities as well. Finally, we discuss how our algorithm can be used to evaluate full conjunctive queries optimally, to compute a relaxed notion of joins and to optimally (in the worst-case) enumerate all induced copies of a fixed subgraph inside of a given large graph.

CCS Concepts: • Information systems → Relational database model; • Theory of computation → Database query processing and optimization (theory).

Additional Key Words and Phrases: Join Algorithms, fractional cover bound, Loomis-Whitney inequality, Bollobás-Thomason inequality

A preliminary version of this article was presented at PODS12 as Reference [62]. We thank Georg Gottlob for sending us a full version of his work [30]. We thank XuanLong Nguyen for introducing us to the Loomis-Whitney inequality. We thank Dung Nguyen for catching some errors in the earlier statement of our algorithm. We thank the anonymous PODS'12 and JACM referees for many helpful comments that have greatly improved the presentation of the article. HN's work is partly supported by NSF Grants No. CCF-1161196 and No. CCF-1319402. C.R. acknowledges the National Science Foundation (NSF) under CAREER Awards No. IS-1353606 and No. CCF-1356918, the Office of Naval Research (ONR) under Awards No. N000141210041 and No. N000141310129, the Sloan Research Fellowship, the Moore Foundation Data Driven Investigator award, and gifts from American Family Insurance, Google, Lightspeed Ventures, and Toshiba. A.R.'s work on this project is supported by NSF Grants No. CCF-0844796 and No. CCF-1319402.

Authors' addresses: H. Q. Ngo and A. Rudra, 338 Davis Hall, University at Buffalo, Buffalo, NY, 14214, USA; emails: {hungngo, atr}@buffalo.edu, E. Porat, Bar-Ilan University, Ramat-Gan, 5290002 Israel; email: porate@cs.biu.ac.il, C. Ré, Gates Computer Science Building, 353 Serra Mall, Stanford, CA 94305, USA; email: chrisre@cs.stanford.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 ACM 0004-5411/2018/03-ART16-31\$15.00
https://doi.org/10.1145/3180143

IN-DB LEARNING

- In-Database Learning with Sparse Tensors, Abo Khamis, Ngo, Nguyen, Olteanu, Schleich (PODS 2018)
- AC/DC: In-Database Learning Thunderstruck, Abo Khamis, Ngo, Nguyen, Olteanu, Schleich (DEEM 2018)
- Modelling Machine Learning Algorithms on Relational Data with Datalog. Makrynioti, Vasiloglou, Pasalic, Vassalos. (DEEM 2018)
- In-Database Factorized Learning, Ngo, Nguyen, Olteanu, Schleich (AMW 2017)
- Data Science with Linear Programming. Makrynioti, Vasiloglou, Pasalic, Vassalos. (DeLBP 2017)

In-Database Factorized Learning

Hung Q. Ngo¹, XuanLong Nguyen², Dan Olteanu³, and Maximilian Schleich³

In-Database Learning with Sparse Tensors

Mahmoud Abo Khamis¹ Hung Q. Ngo¹ XuanLong Nguyen²
Dan Olteanu³ Maximilian Schleich³

AC/DC: In-Database Learning Thunderstruck

Mahmoud Abo Khamis Hung Q. Ngo XuanLong Nguyen
RelationalAI, Inc RelationalAI, Inc University of Michigan
Dan Olteanu Maximilian Schleich
University of Oxford University of Oxford

ABSTRACT
In-database avoids the cost of moving data from the server to the client. This paper introduces AC/DC, a gradient descent solver for a class of optimization problems over normalized databases. AC/DC decomposes an optimization problem into a set of aggregates over the join of the database relations. It then uses the answers to these aggregates to iteratively improve the solution to the problem until it converges.

The challenges faced by AC/DC are the large database size, the mixture of continuous and categorical features, and the large number of aggregates to compute. AC/DC addresses these challenges by employing a sparse data representation, factorized computation, problem reparameterization under functional dependencies, and a data structure that supports shared computation of aggregates.

To train polynomial regression models and factorization machines of up to 141K features over the join of a real-world dataset of up to 86M tuples, AC/DC needs up to 30 minutes on one core of a commodity machine. This is up to three orders of magnitude faster than its competitors R, MadLib, libFM, and TensorFlow whenever they finish and thus do not exceed memory limitation, 24-hour timeout, or internal design limitations.

1. INTRODUCTION
In this paper we report our on-going work on the design and implementation of AC/DC, a gradient descent solver for a class of optimization problems including ridge linear regression, polynomial regression, and factorization machines. It extends our prior system F for factorized learning of linear regression models [21] to capture non-linear models, categorical features, and model reparameterization under functional dependencies (FDs). Its design is but one fruit of our exploration of the design space for the AI engine.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
DEEM, June 2018, Houston, Texas
© 2018 Association for Computing Machinery.

IN-DB & LIFTED LP/QP/IP SOLVERS

- SolverBlox: Algebraic Modeling in Datalog. Borraz-Sanchez, Klabjan, Pasalic, Aref. (Declarative Logic Programming – Morgan & Claypool 2018)
- The Symbolic Interior Point Method. Mladenov, Belle, Kersting. (AAAI 2017)
- Lifted Inference for Convex Quadratic Programs. Mladenov, Kleinhans, Kersting. (AAAI 2017)
- RELOOP/ A Python-Embedded Declarative Language for Relational Optimization. Mladenov, Heinrich, Kleinhans, Gonsior, Kersting. (AAAI 2016)
- Relational Linear Programs. Kersting, Mladenov, Tokmanov. (2015)
- Lifted Linear Programming. Mladenov, Ahmadi, Kersting. (AISTATS 2012)

Relational Linear Programs

Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Lifted Inference for Convex Quadratic Programs

Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

The Symbolic Interior Point Method

TU Dortmund University
Symmetry recently discovered inference models. The optimization problem is a class of optimization problems. In many AI applications, we lift, i.e., to with the using plane QP exhibits more complex structures. Convex optimization approaches, such as RK1 and learning LASSO and LASSO approaches, include the packages for Chiu, and B between the braic language impossible-discrete, continuous, piece of placing a window in the tation such structure an ever, is likely been demon that has arging with co al. 2016) fe ture of the

Copyright © Intelligence

1

SolverBlox: Algebraic Modeling in Datalog

C. Borraz-Sánchez, Data&Analytics Center, KPMG LLP, Knoxville, TN
cborrazsanchez@kpmg.com
D. Klabjan, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL
d-klabjan@northwestern.edu
E. Pasalic, LogicBlox Inc., Atlanta, GA.
emir.pasalic,molham.aref@logicblox.com
M. Aref, LogicBlox Inc., Atlanta, GA.
{emir.pasalic,molham.aref}@logicblox.com

Datalog is a deductive query language for relational databases. We introduce LogiQL, a language based on Datalog and show how it can be used to specify mixed-integer linear optimization models and solve them. Unlike pure algebraic modeling languages, LogiQL allows the user to both specify models, and manipulate and transform the inputs and outputs of the models. This is an advantage over conventional optimization modeling languages that rely on reading data via plug-in tools or importing data from external sources via files. In this chapter, we give a brief overview of LogiQL and describe two mixed integer programming case studies: a production-transportation model, and a formulation of the traveling salesman problem.

1.1 Introduction

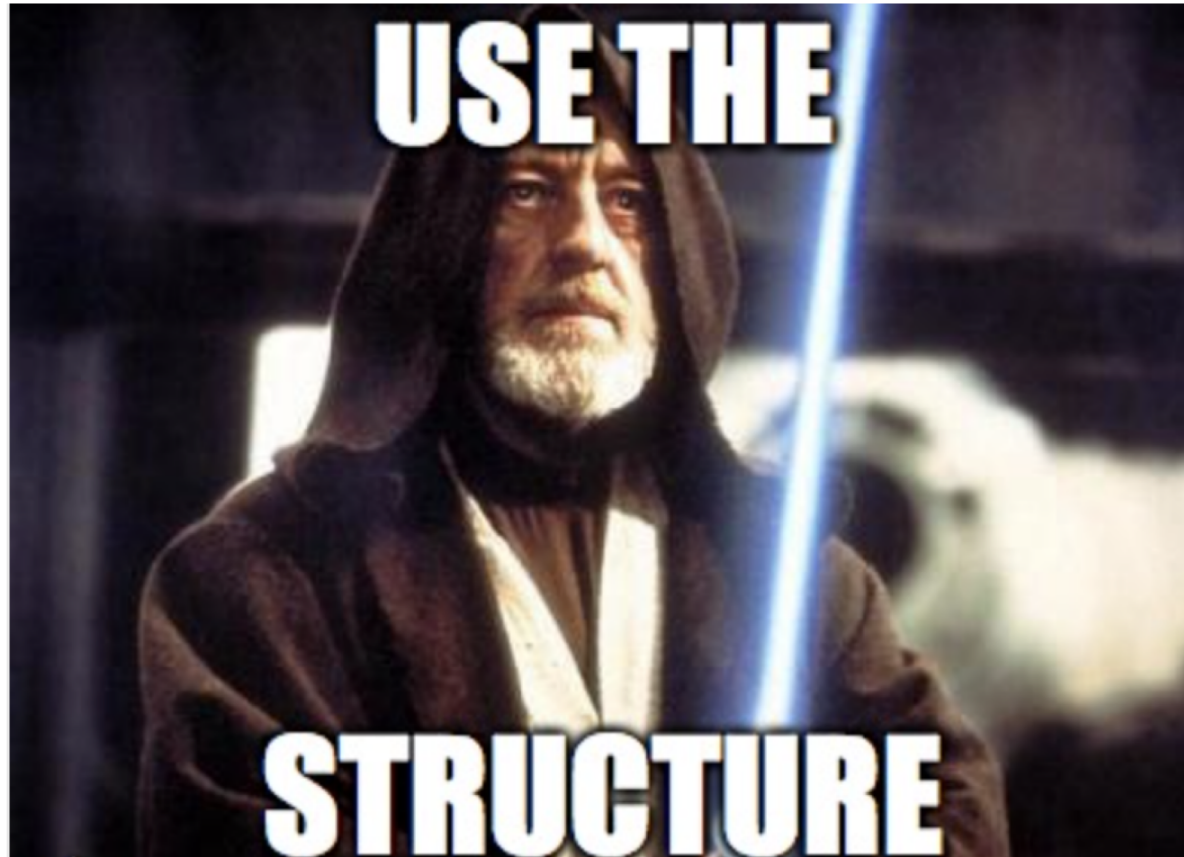
Solutions to many real world problems can be formulated as mathematical programs [Stephen P. Bradley and Magnanti 1977]. Such problems are usually tackled by first building a model, and then converting the model and the corresponding data into a low level problem instance. The instance can then be solved by a solver that returns a solution for further analysis and use.

One of the biggest challenges throughout the entire modeling and solution process is dealing with large amounts and diverse representations of data. One might even argue that mathematical modeling is mainly about data [Hultberg 2007] and should therefore be driven by data. A mathematical programming modeling language could be made more useful if it

1

40

FOR A SMALL GROUP OF REBELS TO BEAT THE EMPIRE, WE HAVE TO...



FIN