OSS in the MSR Database Group

- Work on diverse areas such as streaming, big data, key-value stores, storage, security, scale-out, ML for systems, ...

- Recently, we have open-sourced several research projects
OSS in the MSR Database Group

• Work on diverse areas such as streaming, big data, key-value stores, storage, security, scale-out, ML for systems, ...

• Recently, we have open-sourced several research projects
  • **Trill**: proven *streaming engine* for real-time and offline analytics
    • [https://github.com/Microsoft/Trill](https://github.com/Microsoft/Trill)
OSS in the MSR Database Group

• Work on diverse areas such as streaming, big data, key-value stores, storage, security, scale-out, ML for systems, ...

• Recently, we have open-sourced several research projects
  • Trill: proven streaming engine for real-time and offline analytics
    • https://github.com/Microsoft/Trill
  • FASTER: fast key-value store for resilient state management
    • https://github.com/Microsoft/FASTER
OSS in the MSR Database Group

- Work on diverse areas such as streaming, big data, key-value stores, storage, security, scale-out, ML for systems, ...

- Recently, we have open-sourced several research projects
  - **Trill**: proven streaming engine for real-time and offline analytics
    - [https://github.com/Microsoft/Trill](https://github.com/Microsoft/Trill)
  - **FASTER**: fast key-value store for resilient state management
    - [https://github.com/Microsoft/FASTER](https://github.com/Microsoft/FASTER)
  - **CRA**: powerful distributed runtime for dataflow graphs
    - [https://github.com/Microsoft/CRA](https://github.com/Microsoft/CRA)
OSS in the MSR Database Group

- Work on diverse areas such as streaming, big data, key-value stores, storage, security, scale-out, ML for systems, ...

- Recently, we have open-sourced several research projects
  - **Trill**: proven streaming engine for real-time and offline analytics
    - https://github.com/Microsoft/Trill
  - **FASTER**: fast key-value store for resilient state management
    - https://github.com/Microsoft/FASTER
  - **CRA**: powerful distributed runtime for dataflow graphs
    - https://github.com/Microsoft/CRA
  - **Ambrosia**: author highly robust applications & microservices easily
    - https://github.com/Microsoft/Ambrosia
Trill

Streaming engine for the cloud & edge

Badrish Chandramouli, Jonathan Goldstein, James Terwilliger, Mike Barnett, Yinan Li, Peter Freiling, Zhong Chen, and others
Scenarios for Big Data Analytics
Scenarios for Big Data Analytics

- Real-time
  - Monitor app telemetry (e.g., ad clicks) & raise alerts when problems are detected
Scenarios for Big Data Analytics

- Real-time
  - Monitor app telemetry (e.g., ad clicks) & raise alerts when problems are detected

- Real-time with historical
  - Correlate live data stream with historical activity (e.g., from 1 week back)
Scenarios for Big Data Analytics

- **Real-time**
  - Monitor app telemetry (e.g., ad clicks) & raise alerts when problems are detected

- **Real-time with historical**
  - Correlate live data stream with historical activity (e.g., from 1 week back)

- **Offline**
  - Develop initial monitoring query using logs
  - Back-test monitoring query over historical logs
  - Run interactive queries on data
Scenarios for Big Data Analytics

- **Real-time**
  - Monitor app telemetry (e.g., ad clicks) & raise alerts when problems are detected

- **Real-time with historical**
  - Correlate live data stream with historical activity (e.g., from 1 week back)

- **Offline**
  - Develop initial monitoring query using logs
  - Back-test monitoring query over historical logs
  - Run interactive queries on data
Requirements for “one engine”

- monitor telemetry & raise alerts
- correlate real-time with logs
- develop initial monitoring query
- back-test over historical logs
- offline analysis (BI) with early results
Requirements for “one engine”

- **Performance**
  - High throughput: critical for large offline datasets
  - Low latency & overhead: Important for real time monitoring

- **Scenarios**
  - monitor telemetry & raise alerts
  - correlate real-time with logs
  - develop initial monitoring query
  - back-test over historical logs
  - offline analysis (BI) with early results
Requirements for “one engine”

- **Performance**
  - High throughput: critical for large offline datasets
  - Low latency & overhead: Important for real time monitoring

- **Fabric & language integration**
  - Cloud app/service acts as driver, *uses* engine as library
  - Need rich data-types, integrate custom logic seamlessly

**Scenarios**

- monitor telemetry & raise alerts
- correlate real-time with logs
- develop initial monitoring query
- back-test over historical logs
- offline analysis (BI) with early results
Requirements for “one engine”

- **Performance**
  - High throughput: critical for large offline datasets
  - Low latency & overhead: Important for real time monitoring

- **Fabric & language integration**
  - Cloud app/service acts as driver, *uses* engine as library
  - Need rich data-types, integrate custom logic seamlessly

- **Query model**
  - Need to support real-time and offline data, temporal and relational queries, interactive queries

**Scenarios**

- monitor telemetry & raise alerts
- correlate real-time with logs
- develop initial monitoring query
- back-test over historical logs
- offline analysis (BI) with early results
Trill: Fast Streaming Analytics Library
Trill: Fast Streaming Analytics Library

- **Performance**
  - 2-4 *orders of magnitude* faster than traditional SPEs
  - For relational, comparable to best columnar DBMS
  - User-controlled latency specification
    - explicit latency vs. throughput tradeoff
Trill: Fast Streaming Analytics Library

- **Performance**
  - 2-4 *orders of magnitude* faster than traditional SPEs
  - For relational, comparable to best columnar DBMS
  - User-controlled latency specification
    - explicit latency vs. throughput tradeoff

- **Fabric & language integration**
  - Built as high-level language (HLL) library component
  - Works with arbitrary HLL data-types & libraries
Trill: Fast Streaming Analytics Library

- **Performance**
  - 2-4 *orders of magnitude* faster than traditional SPEs
  - For relational, comparable to best columnar DBMS
  - User-controlled latency specification
    - explicit latency vs. throughput tradeoff

- **Fabric & language integration**
  - Built as high-level language (HLL) library component
  - Works with arbitrary HLL data-types & libraries

- **Query model**
  - Extended LINQ syntax based on temporal + patterns
Used Across Microsoft

- Azure Stream Analytics service
- Bing Ads
- Office, Exchange, Windows
- Halo game monitoring & debugging
- ...
Used Across Microsoft

- Azure Stream Analytics service
- Bing Ads
- Office, Exchange, Windows
- Halo game monitoring & debugging
- ...
Used Across Microsoft

- Azure Stream Analytics service
- Bing Ads
- Office, Exchange, Windows
- Halo game monitoring & debugging
- ...

- Key enabler: performance + fabric & language integration + query model
Current Status – https://aka.ms/Trill

- Use and contribute
  - Open source at https://github.com/Microsoft/Trill
  - Library binaries available on NuGet.org
Current Status – https://aka.ms/Trill

- Use and contribute
  - Open source at https://github.com/Microsoft/Trill
  - Library binaries available on NuGet.org

- Features
  - .NET core → works on edge, cloud, Windows, Linux, ...
  - Pattern detection, signal processing, extensibility endpoints
  - Trill + CRA → Quill for multi-node scan-based analytics
  - Trill + Ambrosia → real-time query pipelines
  - Trill + FASTER → externalize operator state, in progress (covered next)
Current Status – https://aka.ms/Trill

· Use and contribute
  · Open source at https://github.com/Microsoft/Trill
  · Library binaries available on NuGet.org

· Features
  · .NET core → works on edge, cloud, Windows, Linux, ...
  · Pattern detection, signal processing, extensibility endpoints
  · Trill + CRA → Quill for multi-node scan-based analytics
  · Trill + Ambrosia → real-time query pipelines
  · Trill + FASTER → externalize operator state, in progress (covered next)

· Research Papers
  · Trill paper (VLDB 2015), Trill article (IEEE Data Engg. Bulletin 2016), Quill (VLDB 2016), Signal Processing (SIGMOD 2017), Stream Sorting (ICDE 2018), ...
FASTER

Embedded key-value store for state management

Badrish Chandramouli, Donald Kossmann, Guna Prasaad, James Hunter, Justin Levandoski, Mike Barnett, Peter Freiling, James Terwilliger, and others
The State Management Problem

- Tremendous growth in data-intensive applications and services
  - Tracking IoT devices, data center monitoring, streaming, online services, ...
The State Management Problem

• Tremendous growth in data-intensive applications and services
  • Tracking IoT devices, data center monitoring, streaming, online services, ...

• State management is a hard problem
  • State consists of independent objects – *devices, users, ads*
  • State does not fit in memory → **problem for edge & multi-tenant as well**
  • Point ops with lots of updates – *e.g., update per-device average CPU reading*
  • State needs to be recoverable
The State Management Problem

- Tremendous growth in data-intensive applications and services
  - Tracking IoT devices, data center monitoring, streaming, online services, ...

- State management is a hard problem
  - State consists of independent objects – devices, users, ads
  - State does not fit in memory → problem for edge & multi-tenant as well
  - Point ops with lots of updates – e.g., update per-device average CPU reading
  - State needs to be recoverable

Temporal Locality

- Search engine maintains per-user stats over last week
- Billions of users “alive”
- Only millions actively surfing at given instant of time
What is FASTER

- Latch-free concurrent multi-core hash key-value store
- Designed for high performance and scalability across threads (shared memory)
- Supports data larger than main memory + recovery
- Shapes the (changing) hot working set in memory → integrated cache
What is FASTER

- Latch-free concurrent multi-core hash key-value store
  - Designed for high performance and scalability across threads (shared memory)
  - Supports data larger than main memory + recovery
  - Shapes the (changing) hot working set in memory → integrated cache

- Performance: up to 200 million ops/sec for YCSB variants
  - One Intel Xeon machine, two sockets, 72 threads
  - Exceeds throughput of pure in-memory systems when working set fits in memory
What is FASTER

• Latch-free concurrent multi-core hash key-value store
  • Designed for high performance and scalability across threads (shared memory)
  • Supports data larger than main memory + recovery
  • Shapes the (changing) hot working set in memory \(\rightarrow\) integrated cache

• Performance: up to 200 million ops/sec for YCSB variants
  • One Intel Xeon machine, two sockets, 72 threads
  • Exceeds throughput of pure in-memory systems when working set fits in memory

• FASTER Interface
  • Read, Blind Update
  • Atomic read-modify-write (RMW) - for running aggs (like sum), partial field updates
Scalability with # Threads

- When current working set "happens to fit" in memory

100% RMW; 8 byte payloads

100% blind updates; 100 byte payloads
System Architecture

Threads → Hash Index → Hybrid Record Log

... → ..., r₂, r₁ → head → Hybrid Record Log → Disk

... → ... → tail → Hybrid Record Log → Memory (cache)
System Architecture

- Technical Innovations
  - **Indexing**: Concurrent hash index (see paper)
  - **Record Storage**: “Hybrid Log” record allocator
  - **Threading**: Epoch Protection Framework with Trigger Actions (see paper)
Hybrid Log Allocator

- Divide memory into three regions
  - Stable (on disk) \( \rightarrow \) Read-Copy-Update (RCU)
  - Mutable (in memory) \( \rightarrow \) In-Place Update (IPU)
  - Read-only (in memory) \( \rightarrow \) Read-Copy-Update (RCU)
Hybrid Log Allocator

- Basic RMW Algorithm

<table>
<thead>
<tr>
<th>Logical Address</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; Head Offset</td>
<td>Issue async IO request</td>
</tr>
<tr>
<td>&lt; ReadOnly Offset</td>
<td>Copy to tail, CAS-update hash index</td>
</tr>
<tr>
<td>&lt; Infinity</td>
<td>Update in-place</td>
</tr>
<tr>
<td>New Record</td>
<td>Add to tail, update hash table</td>
</tr>
</tbody>
</table>

- Removes append-only log bottleneck
- Elegant design, but hard to maintain multi-threaded correctness
  - See SIGMOD 2018 paper
Status – https://aka.ms/FASTER

- Open sourced August 2018 (github.com/Microsoft/FASTER)
- NuGet package available as well, **C# and C++** versions of code
Status – https://aka.ms/FASTER

- Open sourced August 2018 (github.com/Microsoft/FASTER)
  - NuGet package available as well, C# and C++ versions of code

- Reached front page of Hackernews twice
- Papers: SIGMOD 2018 (core system), VLDB 2018 (demo), SIGMOD 2019 (recovery)
- Integrating FASTER as state store of Trill
Talk Summary

- We have recently open sourced several research projects
  - **Trill**: proven *streaming engine* for real-time and offline analytics
    - https://github.com/Microsoft/Trill
  - **FASTER**: fast key-value store for resilient *state management*
    - https://github.com/Microsoft/FASTER
  - **CRA**: powerful *distributed runtime* for dataflow graphs
    - https://github.com/Microsoft/CRA
  - **Ambrosia**: author highly *robust applications* & microservices easily
    - https://github.com/Microsoft/Ambrosia

- Invite everyone to use, contribute, and perform follow-up research
- Talk to us for more details, go to GitHub for docs & guides
Democratizing Data Preparation for AI

Jiannan Wang

Simon Fraser University
SFU DB/DM Group

History

- Over 30 years of research experience in database and data mining
- Wrote a Data Mining Textbook widely used in the world
- Invented many famous data mining algorithms (e.g., FP-Growth, DBScan, Prefixspan)

Mining frequent patterns without candidate generation
J Han, J Pei, Y Yin
ACM sigmod record 29 (2), 1-12

Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth
J Pei, J Han, B Mortazavi-Asl, H Pinto, Q Chen, U Dayal, MC Hsu
Iccone, 0215

A density-based algorithm for discovering clusters in large spatial databases with noise.
M Ester, HP Kriegel, J Sander, X Xu
Kdd 96 (34), 226-231
SFU DB/DM Group

- **Research Areas:** Machine Learning, Data Science, and Big Data Systems
- **Research Strengths:** Cloud Databases, Crowdsourced Data Management, Data Cleaning and Integration, Data Security and Privacy, Fraud Detection, Interpretable Machine Learning, Precision Medicine, Recommender Systems
- **Ranked 13th** in databases and data mining in North America (source: csrankings.org)

<table>
<thead>
<tr>
<th>Institution</th>
<th>Count Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carnegie Mellon University</td>
<td>17.7</td>
</tr>
<tr>
<td>Univ. of Illinois at Urbana-Champaign</td>
<td>14.0</td>
</tr>
<tr>
<td>Stanford University</td>
<td>13.0</td>
</tr>
<tr>
<td>Georgia Institute of Technology</td>
<td>11.5</td>
</tr>
<tr>
<td>University of Michigan</td>
<td>11.5</td>
</tr>
<tr>
<td>Massachusetts Institute of Technology</td>
<td>10.3</td>
</tr>
<tr>
<td>Cornell University</td>
<td>10.2</td>
</tr>
<tr>
<td>Purdue University</td>
<td>8.6</td>
</tr>
<tr>
<td>Pennsylvania State University</td>
<td>8.7</td>
</tr>
<tr>
<td>University of California - Los Angeles</td>
<td>8.6</td>
</tr>
<tr>
<td>University of Massachusetts Amherst</td>
<td>8.6</td>
</tr>
<tr>
<td>University of Illinois at Chicago</td>
<td>8.2</td>
</tr>
<tr>
<td>Simon Fraser University</td>
<td>8.1</td>
</tr>
<tr>
<td>University of Maryland - College Park</td>
<td>8.1</td>
</tr>
<tr>
<td>University of Waterloo</td>
<td>7.9</td>
</tr>
<tr>
<td>Duke University</td>
<td>7.6</td>
</tr>
<tr>
<td>University of California - Santa Barbara</td>
<td>7.6</td>
</tr>
<tr>
<td>University of California - Santa Cruz</td>
<td>7.6</td>
</tr>
<tr>
<td>University of Wisconsin - Madison</td>
<td>7.4</td>
</tr>
<tr>
<td>Ohio State University</td>
<td>7.2</td>
</tr>
<tr>
<td>University of California - Riverside</td>
<td>7.0</td>
</tr>
<tr>
<td>University of California - San Diego</td>
<td>6.7</td>
</tr>
<tr>
<td>University at Buffalo</td>
<td>6.4</td>
</tr>
</tbody>
</table>

Ke Wang (Joined in 2000)  
Martin Ester (Joined in 2001)  
Jian Pei (Joined in 2004)  
Jiannan Wang (Joined in 2016)  
Tianzheng Wang (Joined Fall 2018)
Democratizing AI
Democratizing AI

Computing

Algorithms

Training Data
Democratizing AI

Computing

Azure

Amazon Web Services

Google Cloud Platform

Algorithms

Training Data
Democratizing AI

Computing

Algorithms

Training Data

Azure

Amazon Web Services

Google Cloud Platform

Microsoft CNTK

PyTorch

TensorFlow
Democratizing AI

Computing

Azure

Amazon Web Services

Google Cloud Platform

Algorithms

Microsoft CNTK

PyTorch

TensorFlow

Training Data

BigGorilla

Trifacta

Snorkel
Democratizing AI

Computing

Algorithms

Training Data

The Bottleneck
What is Data Prep?

Data Lake → Training Data
What is Data Prep?

Data Lake -> Data Preparation -> Training Data
Why is Data Prep so hard?

- Data Discovery
- Data Profiling
- Data Extraction
- Data Normalization
- Data Enrichment
- Data Transformation
- Data Filtering
- Data Provenance
- Data Labeling
- Error Detection
- Schema Matching
- Deduplication
- Outlier Detection
- Imputation
- ...

Inspired by the conversation with Dr. Phil Bernstein at CIDR 2017
New Opportunities for DB Community

Focus on reducing data scientists’ time

- Ease of Use
- Extensibility
- Composability
New Opportunities for DB Community

Focus on reducing data scientists’ time
- Ease of Use
- Extensibility
- Composability

Focus on using advanced ML technologies
- Automated Machine Learning
- Meta Learning (a.k.a. Learning to Learn)
Recent Progress

**Deeper** [SIGMOD 2018 (Demo), SIGMOD 2019]
- Reduce data enrichment time

**AQP++** [SIGMOD 2018]
- Reduce exploratory data analysis time
Recent Progress

**Deeper** [SIGMOD 2018 (Demo), SIGMOD 2019]
- Reduce data enrichment time

**AQP++** [SIGMOD 2018]
- Reduce exploratory data analysis time

**TARS** [VLDB 2019]
- Reduce data labeling time
A Promising Solution

Label Noise vs. Human Cost Trade-off

- Random
- Distance/Weak Supervision
- Crowdsourcing
- Domain Expert
Cleaning Noisy Label

Existing Work*

- No Cleaning
- Machine-based Cleaning

Cleaning Noisy Label

Existing Work*

- No Cleaning
- Machine-based Cleaning

Our Solution

- Oracle-based Cleaning

TARS [named after an intelligent robot in the movie *Interstellar*]

Label Cleaning Advisor for Crowdsourced Noisy Labels

Mohamad Dolatshah  Mathew Teoh  Jiannan Wang  Jian Pei

Dolatshah et al. Cleaning Crowdsourced Labels Using Oracles For Statistical Classification. *PVLDB 2019*
Two Pieces of Advice
Two Pieces of Advice

Advice 1. Model Evaluation

- How accurate is a model?
- (1) Model
- (2) Noisy Test Data

0.8±0.01
Two Pieces of Advice

Advice 1. Model Evaluation

How accurate is a model?

(1) Model
(2) Noisy Test Data

0.8 ± 0.01

Advice 2. Cleaning Strategy

Which label should be cleaned?

(1) Learning Algorithm
(2) Noisy Training Data

<instance_3, label_3>
Take-away Messages

DB community should play an important role in democratizing data preparation for AI
Take-away Messages

DB community should play an important role in democratizing data preparation for AI

We build TARS, a label cleaning advisor to reduce data labeling time for AI
Take-away Messages

DB community should play an important role in democratizing data preparation for AI

We build TARS, a label cleaning advisor to reduce data labeling time for AI

Poster 1: Extracting Highlights from Recorded Live Videos (Changbo)
Poster 2: Explaining ML-embedded SQL Queries (Weiyuan)
INFLUENCE MAXIMIZATION IN MASSIVE GRAPHS

NWDS’2019

Diana Popova
Department of Computer Science
University of Victoria, Canada

February 2019
Overview

- Influence Discovery in Graphs
- Algorithms Scalability
- Influence Maximization
Influence
Discovery
Influence Discovery in Graph
# Graph’s Incidence List

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>48</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>87</td>
<td>7</td>
<td>38</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>53</td>
<td>3</td>
<td>25</td>
<td>5</td>
<td>122</td>
<td>7</td>
<td>65</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>54</td>
<td>3</td>
<td>26</td>
<td>5</td>
<td>156</td>
<td>7</td>
<td>87</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>73</td>
<td>3</td>
<td>67</td>
<td>5</td>
<td>158</td>
<td>7</td>
<td>103</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>88</td>
<td>3</td>
<td>72</td>
<td>5</td>
<td>169</td>
<td>7</td>
<td>129</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>92</td>
<td>3</td>
<td>85</td>
<td>5</td>
<td>180</td>
<td>7</td>
<td>136</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>119</td>
<td>3</td>
<td>122</td>
<td>5</td>
<td>187</td>
<td>7</td>
<td>168</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>126</td>
<td>3</td>
<td>142</td>
<td>5</td>
<td>204</td>
<td>7</td>
<td>213</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>133</td>
<td>3</td>
<td>170</td>
<td>5</td>
<td>213</td>
<td>7</td>
<td>246</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>174</td>
<td>3</td>
<td>188</td>
<td>5</td>
<td>235</td>
<td>7</td>
<td>291</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>236</td>
<td>3</td>
<td>200</td>
<td>5</td>
<td>315</td>
<td>7</td>
<td>304</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>280</td>
<td>3</td>
<td>228</td>
<td>5</td>
<td>316</td>
<td>7</td>
<td>308</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>299</td>
<td>3</td>
<td>274</td>
<td>6</td>
<td>0</td>
<td>7</td>
<td>315</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>315</td>
<td>3</td>
<td>280</td>
<td>6</td>
<td>89</td>
<td>7</td>
<td>322</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>322</td>
<td>3</td>
<td>283</td>
<td>6</td>
<td>95</td>
<td>7</td>
<td>339</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>346</td>
<td>3</td>
<td>323</td>
<td>6</td>
<td>147</td>
<td>7</td>
<td>340</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>20</td>
<td>4</td>
<td>78</td>
<td>6</td>
<td>319</td>
<td>8</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>115</td>
<td>4</td>
<td>152</td>
<td>7</td>
<td>0</td>
<td>8</td>
<td>91</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>116</td>
<td>4</td>
<td>181</td>
<td>7</td>
<td>22</td>
<td>8</td>
<td>110</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>149</td>
<td>4</td>
<td>195</td>
<td>7</td>
<td>31</td>
<td>8</td>
<td>193</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>226</td>
<td>4</td>
<td>218</td>
<td>7</td>
<td>38</td>
<td>8</td>
<td>201</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>312</td>
<td>4</td>
<td>273</td>
<td>7</td>
<td>65</td>
<td>8</td>
<td>245</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>326</td>
<td>4</td>
<td>275</td>
<td>7</td>
<td>87</td>
<td>8</td>
<td>259</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>333</td>
<td>4</td>
<td>306</td>
<td>7</td>
<td>103</td>
<td>8</td>
<td>264</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>343</td>
<td>4</td>
<td>328</td>
<td>7</td>
<td>129</td>
<td>8</td>
<td>252</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>328</td>
<td>7</td>
<td>129</td>
<td>8</td>
<td>252</td>
<td>9</td>
</tr>
</tbody>
</table>
Influence Discovery in Graph
Graph’s Incidence List

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>48</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>87</td>
<td>7</td>
<td>38</td>
<td>9</td>
<td>69</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>53</td>
<td>3</td>
<td>25</td>
<td>5</td>
<td>122</td>
<td>7</td>
<td>65</td>
<td>9</td>
<td>72</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>54</td>
<td>3</td>
<td>26</td>
<td>5</td>
<td>156</td>
<td>7</td>
<td>87</td>
<td>9</td>
<td>75</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>73</td>
<td>3</td>
<td>67</td>
<td>5</td>
<td>158</td>
<td>7</td>
<td>103</td>
<td>9</td>
<td>79</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>88</td>
<td>3</td>
<td>72</td>
<td>5</td>
<td>169</td>
<td>7</td>
<td>129</td>
<td>9</td>
<td>85</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>92</td>
<td>3</td>
<td>85</td>
<td>5</td>
<td>180</td>
<td>7</td>
<td>136</td>
<td>9</td>
<td>105</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>119</td>
<td>3</td>
<td>122</td>
<td>5</td>
<td>187</td>
<td>7</td>
<td>168</td>
<td>9</td>
<td>113</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>126</td>
<td>3</td>
<td>142</td>
<td>5</td>
<td>204</td>
<td>7</td>
<td>213</td>
<td>9</td>
<td>119</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>133</td>
<td>3</td>
<td>170</td>
<td>5</td>
<td>213</td>
<td>7</td>
<td>246</td>
<td>9</td>
<td>122</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>194</td>
<td>3</td>
<td>188</td>
<td>5</td>
<td>235</td>
<td>7</td>
<td>291</td>
<td>9</td>
<td>128</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>236</td>
<td>3</td>
<td>200</td>
<td>5</td>
<td>315</td>
<td>7</td>
<td>304</td>
<td>9</td>
<td>133</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>280</td>
<td>3</td>
<td>228</td>
<td>5</td>
<td>316</td>
<td>7</td>
<td>308</td>
<td>9</td>
<td>134</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>299</td>
<td>3</td>
<td>274</td>
<td>5</td>
<td>0</td>
<td>7</td>
<td>315</td>
<td>9</td>
<td>141</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>315</td>
<td>3</td>
<td>280</td>
<td>6</td>
<td>89</td>
<td>7</td>
<td>322</td>
<td>9</td>
<td>142</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>322</td>
<td>3</td>
<td>283</td>
<td>6</td>
<td>95</td>
<td>7</td>
<td>339</td>
<td>9</td>
<td>156</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>346</td>
<td>3</td>
<td>323</td>
<td>6</td>
<td>147</td>
<td>7</td>
<td>340</td>
<td>9</td>
<td>161</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>219</td>
<td>7</td>
<td>347</td>
<td>9</td>
<td>169</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>2</td>
<td>78</td>
<td>6</td>
<td>319</td>
<td>8</td>
<td>0</td>
<td>9</td>
<td>176</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>115</td>
<td>4</td>
<td>152</td>
<td>6</td>
<td>0</td>
<td>8</td>
<td>91</td>
<td>9</td>
<td>176</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>116</td>
<td>4</td>
<td>181</td>
<td>7</td>
<td>22</td>
<td>8</td>
<td>110</td>
<td>9</td>
<td>185</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>149</td>
<td>4</td>
<td>195</td>
<td>7</td>
<td>38</td>
<td>8</td>
<td>193</td>
<td>9</td>
<td>186</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>226</td>
<td>4</td>
<td>218</td>
<td>7</td>
<td>65</td>
<td>8</td>
<td>201</td>
<td>9</td>
<td>188</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>312</td>
<td>4</td>
<td>273</td>
<td>7</td>
<td>87</td>
<td>8</td>
<td>245</td>
<td>9</td>
<td>199</td>
</tr>
<tr>
<td>24</td>
<td>2</td>
<td>326</td>
<td>4</td>
<td>275</td>
<td>7</td>
<td>103</td>
<td>8</td>
<td>259</td>
<td>9</td>
<td>200</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
<td>333</td>
<td>4</td>
<td>306</td>
<td>7</td>
<td>129</td>
<td>8</td>
<td>264</td>
<td>9</td>
<td>203</td>
</tr>
<tr>
<td>26</td>
<td>2</td>
<td>343</td>
<td>4</td>
<td>328</td>
<td>7</td>
<td>129</td>
<td>8</td>
<td>281</td>
<td>9</td>
<td>231</td>
</tr>
<tr>
<td>27</td>
<td>2</td>
<td>343</td>
<td>4</td>
<td>328</td>
<td>7</td>
<td>129</td>
<td>8</td>
<td>281</td>
<td>9</td>
<td>232</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>343</td>
<td>4</td>
<td>328</td>
<td>7</td>
<td>129</td>
<td>8</td>
<td>281</td>
<td>9</td>
<td>232</td>
</tr>
<tr>
<td>29</td>
<td>2</td>
<td>343</td>
<td>4</td>
<td>328</td>
<td>7</td>
<td>129</td>
<td>8</td>
<td>281</td>
<td>9</td>
<td>232</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>343</td>
<td>4</td>
<td>328</td>
<td>7</td>
<td>129</td>
<td>8</td>
<td>281</td>
<td>9</td>
<td>232</td>
</tr>
</tbody>
</table>
Algorithms Scalability
Scalability

Fair comparison:

• Same graph
Scalability

Fair comparison:

- Same graph
- Max graph size on the same machine
Scalability

Fair comparison:

- Same graph
- Max graph size on the same machine

Tests of eleven different IM algorithms by Arora et al.
Scalability

Fair comparison:

• Same graph
• Max graph size on the same machine

Tests of eleven different IM algorithms by Arora et al.

Scalability

Evaluation:

Figure 11: (a) Summarizing the spectrum of Influence Maximization (IM) techniques based on their strengths. (b) The decision tree for choosing the most appropriate IM algorithm.
Scalability

Evaluation:

Figure 11: (a) Summarizing the spectrum of Influence Maximization (IM) techniques based on their strengths. (b) The decision tree for choosing the most appropriate IM algorithm.
Scalability

Evaluation:

- Quality

Figure 11: (a) Summarizing the spectrum of Influence Maximization (IM) techniques based on their strengths. (b) The decision tree for choosing the most appropriate IM algorithm.
Scalability

Evaluation:

- Quality
- Time and Space

Figure 11: (a) Summarizing the spectrum of Influence Maximization (IM) techniques based on their strengths. (b) The decision tree for choosing the most appropriate IM algorithm.
Influence Maximization
Previous Work

• Kempe, Kleinberg, and Tardos, 2003:
  - Independent Cascade (IC) model of influence propagation.
  - Greedy algorithm for finding the best seed set for a given $k$ (number of seeds).
  - Monte Carlo simulations, randomized selection of edges, and averaging over coverage.

• Borgs, Brautbar, Chayes, and Lucier, 2014:
  - Reverse Influence Sampling: randomized sketching of the transposed graph.
  - Theoretical guarantees: approximation factor of $(1-1/e-\varepsilon)$, for any $\varepsilon > 0$, with 60% confidence.
Influence Maximization (IM)

- **Node Influence** – the number of graph nodes reachable from a given node under a certain model.
- *Information propagation* is a process of spreading information from node to node using edges.
- **IM Problem**: find a given number of *seed* nodes, such that the information would spread far and wide. Class NP. The graph is probabilistic, and the result of influence maximization is an approximation to optimal. Class P.

Our approach:

- Data Structures for small memory footprint
Data Structures for Efficient Computation of Influence Maximization

Reverse Influence Sampling (RIS) idea:
- find the nodes that would influence a randomly selected node;
- do it multiple times;
- if a node appears often as influencer, it is a good candidate for a seed.

Our implementation:
- Webgraph format for the input graph.
- Instead of list of lists, we use flat arrays and boolean arrays (bitset).
- Java 8 parallel streams and lambda expressions.
- Lazy Greedy technique.

Figure 1: Processing time for cnr-2000; k=10, varying $\beta$.  
Comparison to DIM
Webgraph format for storing intermediate results

**Left:** hypergraph as Borgs et al. described in RIS.

**Right:** hypergraph as built by NoSingles.
NoSingles: a Space-Efficient Algorithm for Influence Maximization

Idea: Do not store sketches containing only one node. NS hypergraph and \texttt{node\_count} array are stored on disk.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>min</th>
<th>max</th>
<th>median</th>
<th>1 node sketches</th>
</tr>
</thead>
<tbody>
<tr>
<td>uk100K</td>
<td>1</td>
<td>2925</td>
<td>1</td>
<td>91%</td>
</tr>
<tr>
<td>cnr2000</td>
<td>1</td>
<td>794</td>
<td>1</td>
<td>96%</td>
</tr>
<tr>
<td>eu2005</td>
<td>1</td>
<td>858</td>
<td>1</td>
<td>90%</td>
</tr>
<tr>
<td>ljounal2008</td>
<td>1</td>
<td>78018</td>
<td>1</td>
<td>90%</td>
</tr>
<tr>
<td>arabic2005</td>
<td>1</td>
<td>20708</td>
<td>1</td>
<td>93%</td>
</tr>
</tbody>
</table>

Table 6.4: Sketch Cardinality Statistics ($p = 0.01$).
NoSingles: a Space-Efficient Algorithm for Influence Maximization

CPU=Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz, running OS CentOS, with RAM=1TB; 48 logical cores.

Comparison to two leading IM algorithms, DIM and D-SSA, shows three orders of magnitude savings in required main memory.
NoSingles: a Space-Efficient Algorithm for Influence Maximization

NoSingles can successfully run on a consumer-grade laptop for large graphs.
Borgs’ et al. formula from Theorem 3.1

<table>
<thead>
<tr>
<th>Dataset</th>
<th>( n )</th>
<th>( m )</th>
<th>( \epsilon )</th>
<th>( p )</th>
<th>( k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>arabic2005</td>
<td>22.7 M</td>
<td>0.63 B</td>
<td>0.2</td>
<td>0.001</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 6: Parameters.

<table>
<thead>
<tr>
<th>( R )</th>
<th>sk, total</th>
<th>sk, saved</th>
<th>( H ) size, edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4 T</td>
<td>2.5 B</td>
<td>36.3 M</td>
<td>2.7 B</td>
</tr>
</tbody>
</table>

Table 7: Intermediate results.

<table>
<thead>
<tr>
<th>( H ) space</th>
<th>( H ) time</th>
<th>Seeds time</th>
<th>accuracy</th>
<th>confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 GB</td>
<td>90.5 hrs</td>
<td>136.5 sec</td>
<td>0.43</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Table 8: Results.
CutTheTail: a Space-Efficient Heuristic Algorithm for Influence Maximization

Idea
CutTheTail1: Do not store sketches containing only nodes with low out-degree. 
CutTheTail2: Do not store short sketches.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>n</th>
<th>m</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordAsn</td>
<td>10.6K</td>
<td>72K</td>
<td>association, directed</td>
</tr>
<tr>
<td>Caida</td>
<td>65.5K</td>
<td>106.7K</td>
<td>social, directed</td>
</tr>
<tr>
<td>FB</td>
<td>4K</td>
<td>176K</td>
<td>social, undirected</td>
</tr>
<tr>
<td>EnronD</td>
<td>69K</td>
<td>275K</td>
<td>e-mails, directed</td>
</tr>
<tr>
<td>Enron</td>
<td>36.7K</td>
<td>368K</td>
<td>e-mails, undirected</td>
</tr>
<tr>
<td>Deezer</td>
<td>54.6K</td>
<td>996K</td>
<td>social, undirected</td>
</tr>
<tr>
<td>DBLP2010</td>
<td>326K</td>
<td>1.6 M</td>
<td>collaboration, undirected</td>
</tr>
<tr>
<td>UK100K</td>
<td>100K</td>
<td>3 M</td>
<td>web, directed</td>
</tr>
<tr>
<td>CNR2000</td>
<td>326K</td>
<td>3.2 M</td>
<td>web, directed</td>
</tr>
<tr>
<td>DBLP2011</td>
<td>986K</td>
<td>6.7 M</td>
<td>collaboration, undirected</td>
</tr>
<tr>
<td>Arabic2005</td>
<td>23M</td>
<td>640M</td>
<td>web, directed</td>
</tr>
</tbody>
</table>

Table 1: Test datasets ordered by m.

Confidence test: log(n) runs, for (1 − 1/n) confidence.
Statistics on saved sketches: CTT2 can save only 0.01% sketches.
Monte Carlo simulation of seeds quality: TopDegree varies from 33% of NS spread to 100% of NS spread, but never better than NS.
Conclusion

• Choice of Data Structure proved to be instrumental in raising the scalability of graph analytics.

• Focus on space complexity allowed to design and implement smart algorithms processing large graphs on a consumer-grade laptop.
Integrity Constraints Revisited: From Exact to Approximate Implication

Batya Kenig
Dan Suciu
University of Washington
Problem Statement (Informal)

- Fix a *single* relation instance R.
- Integrity Constraints: FDs and MVDs only
  - Hard: either R $\models \tau$ or R $\not\models \tau$.
  - Soft: R satisfies $\tau$ to some degree.
- Relaxing exact implications:
  - Suppose $\Sigma \models \tau$ holds for hard constraints.
  - If the constraints in $\Sigma$ hold to a large extent, to what extent does $\tau$?
- Lots of applications.
  - Mining of approximate integrity constraints in a DB instance (Chu et al. 2014, Giannella and Robsertson 2004, Kruse and Naumann 2018)
  - Data cleaning (Illyas and Chu 2015)
  - Learning structure of Probabilistic Graphical Models
Outline

- Key Concepts & Ideas
- Main Results
Outline

- Key Concepts & Ideas
- Main Results
Conditional Independence Statements

- We consider discrete probability distributions.
- X is a set of random variables.
- A,B,C,… are subsets of X.
- $A \perp B \mid C \iff P(A,B \mid C)=P(A \mid C)P(B \mid C)$. 
- $A \perp B \mid C$ is saturated if $X=AuBuC$.
- $A \perp B \mid C$ is marginal if $C=\emptyset$.
- $\Sigma$ is a set of CI statements, $\tau$ is a single CI statement.
- An important concept in probabilistic modeling and reasoning.
Definition: Probabilistic CI Implication Problem

Let $\Sigma$ be a set of CI statements and let $\tau$ be a CI statement. We say that $\Sigma$ implies $\tau$, denoted $\Sigma \models \tau$, if every probability distribution that satisfies the CI statements in $\Sigma$ also satisfies the CI statement $\tau$. 
Definition: Probabilistic CI Implication Problem
Let $\Sigma$ be a set of CI statements and let $\tau$ be a CI statement. We say that $\Sigma$ implies $\tau$, denoted $\Sigma \models \tau$, if every probability distribution that satisfies the CI statements in $\Sigma$ also satisfies the CI statement $\tau$.

The semi-graphoid axioms, Pearl 1988

- $A \perp \emptyset | C$ (Triviality)
- $A \perp B | C \implies B \perp A | C$ (Symmetry)
- $A \perp BD | C \implies A \perp D | C$ (Decomposition)
- $A \perp B | CD \land A \perp D | C \implies A \perp BD | C$ (Contraction)
- $A \perp BD | C \implies A \perp B | CD$ (Weak Union)
Definition: Probabilistic CI Implication Problem
Let $\Sigma$ be a set of CI statements and let $\tau$ be a CI statement. We say that $\Sigma$ implies $\tau$, denoted $\Sigma \models \tau$, if every probability distribution that satisfies the CI statements in $\Sigma$ also satisfies the CI statement $\tau$.

The semi-graphoid axioms, Pearl 1988

- $A \perp \emptyset | C$ Triviality
- $A \perp B | C \rightarrow B \perp A | C$ Symmetry
- $A \perp BD | C \rightarrow A \perp D | C$ Decomposition
- $A \perp B | CD \land A \perp D | C \rightarrow A \perp BD | C$ Contraction
- $A \perp BD | C \rightarrow A \perp B | CD$ Weak Union

Theorem (Geiger+Pearl 1993)
Axioms are (1) Sound, and (2) Complete for Saturated and Marginal CIs.
Functional Dependency (FD)

- $R$ satisfies the FD $A \rightarrow B$ if $\forall t_1, t_2 \in R$, $t_1.A = t_2.A \Rightarrow t_1.B = t_2.B$

(Embedded) Multivalued Dependency:

- $R$ satisfies the EMVD $A \rightarrow (B|C)$ if $\Pi_{ABC}(R) = \Pi_{AB}(R) \bowtie \Pi_{AC}(R)$
- MVD: $A \rightarrow B$ is an EMVD $A \rightarrow (B|C)$ where $ABC$ = all attrs

Implication:

- Armstrong’s axioms, Beeri’s algorithm
Review: FD and MVD

Functional Dependency (FD)
• R satisfies the FD $A \rightarrow B$ if $\forall t_1, t_2 \in R$, $t_1.A = t_2.A \Rightarrow t_1.B = t_2.B$

(Embedded) Multivalued Dependency:
• R satisfies the EMVD $A \rightarrow (B|C)$ if $\Pi_{ABC}(R) = \Pi_{AB}(R) \bowtie \Pi_{AC}(R)$
• MVD: $A \rightarrow B$ is an EMVD $A \rightarrow (B|C)$ where ABC=all attrs

Implication:
• Armstrong’s axioms, Beeri’s algorithm
Review: FD and MVD

Functional Dependency (FD)
- $R$ satisfies the FD $A \rightarrow B$ if $\forall t_1, t_2 \in R$, $t_1.A = t_2.A \Rightarrow t_1.B = t_2.B$

(Embedded) Multivalued Dependency:
- $R$ satisfies the EMVD $A \rightarrow (B|C)$ if $\Pi_{ABC}(R) = \Pi_{AB}(R) \times \Pi_{AC}(R)$
- MVD: $A \rightarrow B$ is an EMVD $A \rightarrow (B|C)$ where $ABC =$ all attrs

Implication:
- Armstrong’s axioms, Beeri’s algorithm

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Between Integrity Constraints and CIs

**The Empirical Distribution of relation R**

The probability space of the support of $R$, where each tuple $t \in R$ is sampled with probability $\frac{1}{N}$. 
Fix $R$, and its empirical distribution.

- $A \rightarrow B$ iff $B \perp C|A$ where $ABC=\text{all vars.}$
- Fails for EMVD
  - $\emptyset \rightarrow B|C$, but $\neg(B \perp C)$
  - $p(C = 1) = \frac{2}{5}$
  - $p(C = 1|B = 1) = \frac{1}{2}$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1/5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1/5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1/5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1/5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1/5</td>
</tr>
</tbody>
</table>
Review: Information Theory

- $X$ = r.v. with $n$ outcomes; its entropy is:
  $$H(X) = - \sum_{i=1}^{n} p_i \log p_i$$

- The conditional entropy is:
  $$H(Y|X) = H(XY) - H(X)$$

- The conditional mutual information is:
  $$I(X;Y|Z) = H(XZ) + H(YZ) - H(XYZ) - H(Z)$$
Soft Constraints

- For CIs: $X \perp Y | Z \iff I(X; Y|Z) = 0$.
- We will use $I(X; Y|Z)$ to quantify the degree of independence of $X$, $Y$ given $Z$. 
Soft Constraints

- For CIs: $X \perp Y \mid Z \iff I(X; Y \mid Z) = 0$.
- We will use $I(X; Y \mid Z)$ to quantify the degree of independence of $X$, $Y$ given $Z$.

**Theorem (Lee 1987)**

<table>
<thead>
<tr>
<th>FDs</th>
<th>$X \rightarrow Y$ iff $H(Y \mid X) = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVDs</td>
<td>$X \rightarrow Y \mid Z$ iff $I(Z; Y \mid X) = 0$</td>
</tr>
</tbody>
</table>
Known Impossibility Results

- Implication problem for EMVDs is undecidable (Herrmann 2006)
- Implication problem for conditional independence is not finitely axiomatizable (Studeny 1990)
Outline

• Key Concepts & Ideas

• Main Results
The Relaxation Problem

Fix a set of Cls $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$, and a CI $\tau \notin \Sigma$.

Assume*: $\Sigma \models \tau$

**Problem**: find a bound on $\tau$ in terms of $\Sigma$.

Relaxation: $\tau \leq \sum_i \lambda_i \sigma_i$ where $\lambda_i \geq 0$

Unit relaxation: $\tau \leq \sum_i \sigma_i$

* e.g. using Armstrong's axioms, Beeri's algorithm, or semi-graphoid axioms
FDs Admit Unit Relaxation

Theorem
The following are equivalent:

- \( X_1 \rightarrow Y_1, \ldots, X_m \rightarrow Y_m \implies X \rightarrow Y \)
- \( H(Y|X) \leq H(Y_1|X_1) + \ldots + H(Y_m|X_m) \)

Example: \( AB \rightarrow C, AD \rightarrow E, CE \rightarrow F \implies ABD \rightarrow F \)

Therefore this is a valid information-theoretic inequality:

\[
H(F|ABD) \leq H(C|AB) + H(E|AD) + H(F|CE)
\]
CI’s Do Not Admit Relaxation!

Theorem (Kaced & Romashchenko 2013)

\((C \perp D | A), (C \perp D | B), (A \perp B), (B \perp C | D) \Rightarrow C \perp D\)

However, for any \(\lambda_1, \ldots, \lambda_4 \geq 0\) there exists a distribution s.t.

\[ I(C; D) > \lambda_1 I(C; D | A) + \lambda_2 I(C; D | B) + \lambda_3 I(A; B) + \lambda_4 I(B; C | D). \]

However, we can relax “in the limit”

Theorem

If the exact implication \(\Sigma \models \tau\) holds, then for any \(\varepsilon > 0\) there exist \(\lambda_i \geq 0\) such that:

\[ \tau \leq \sum \lambda_i \sigma_i + \varepsilon H(\text{all-variables}) \]
CI’s Do Not Admit Relaxation!

Theorem (Kaced & Romashchenko 2013)

\[(C \perp D | A), (C \perp D | B), (A \perp B), (B \perp C | D) \implies C \perp D\]

However, for any \(\lambda_1, \ldots, \lambda_4 \geq 0\) there exists a distribution s.t.
\[I(C;D) > \lambda_1 I(C;D | A) + \lambda_2 I(C;D | B) + \lambda_3 I(A;B) + \lambda_4 I(B;C | D).\]

However, we can relax “in the limit”

Theorem

If the exact implication \(\Sigma \models \tau\) holds, then for any \(\varepsilon > 0\) there exist \(\lambda_i \geq 0\) such that:
\[\tau \leq \sum_i \lambda_i \sigma_i + \varepsilon H(\text{all-variables})\]
**Saturated CIs**

**Disjoint CIs**

Two CIs $X \perp Y \mid Z$ and $A \perp B \mid C$ are *disjoint* if at least one of the following is non-empty: (1) $X \cap C$ (2) $Y \cap C$ (3) $Z \cap A$ (4) $Z \cap B$. 
Saturated CIs

Disjoint CIs

Two CIs $X \perp Y \mid Z$ and $A \perp B \mid C$ are disjoint if at least one of the following is non-empty: (1) $X \cap C$ (2) $Y \cap C$ (3) $Z \cap A$ (4) $Z \cap B$.

Note: All semi-graphoid axioms are disjoint.
Saturated CIs

Disjoint CIs
Two CIs $X \perp Y \mid Z$ and $A \perp B \mid C$ are disjoint if at least one of the following is non-empty: (1) $X \cap C$ (2) $Y \cap C$ (3) $Z \cap A$ (4) $Z \cap B$.

Note: All semi-graphoid axioms are disjoint.

Theorem
If $\Sigma$ is a set of disjoint CIs, and $\tau$ is saturated, then the implication $\Sigma \Rightarrow \tau$ (by the Shannon inequalities) admits unit relaxation: $\tau \leq \sum_{i} \sigma_{i}$. 
Saturated CIs

Disjoint CIs
Two CIs $X \perp Y \mid Z$ and $A \perp B \mid C$ are disjoint if at least one of the following is non-empty: (1) $X \cap C$ (2) $Y \cap C$ (3) $Z \cap A$ (4) $Z \cap B$.

Note: All semi-graphoid axioms are disjoint.

Theorem
If $\Sigma$ is a set of disjoint CIs, and $\tau$ is saturated, then the implication $\Sigma \vdash \tau$ (by the Shannon inequalities) admits unit relaxation: $\tau \leq \Sigma_i \sigma_i$.

Example: Contraction Axiom in semi-graphoids:

$X \perp Y \mid Z \ \& \ \ X \perp W \mid YZ \Rightarrow X \perp YW \mid Z$

Relaxes to:

$I(X;YW \mid Z) \leq I(X;Y\mid Z) + I(X;W\mid YZ) \ \// \text{in fact, equality}$
Conclusions

- The connection between constraints and information theory has been known for a long time.
- The *relaxation problem* appears to be new.
- Great practical importance: real data satisfies constraints only approximatively, need to relax.
- Open problems: bound on the coefficients $\lambda_i$ in various settings.
Thank You!
Automating Machine Learning Model Building with Clinical Big Data

Gang Luo
Department of Biomedical Informatics and Medical Education
University of Washington
luogang@uw.edu
Challenges of Using Machine Learning for Clinical Predictive Modeling

- Requires many labor-intensive manual iterations and special computing expertise to select among complex algorithms and hyper-parameter values.
- Most machine learning models give no explanation of prediction results.
  - Explanation is essential for a learning healthcare system.
Challenge 1 – Efficient and Automatic Model Selection

- Automatic selection methods for algorithms and hyper-parameter values have been developed
  - to help individuals with little computing expertise perform machine learning
  - but existing methods cannot efficiently handle clinical big data
  - Search can take several days on a data set with a moderate number of rows and attributes
  - Search time is daunting on large data sets
Challenge 1 – Cont.

- To leverage clinical big data, automated approaches appealing to healthcare researchers are needed for selecting algorithms and hyper-parameter values
  - Completely automatic
  - Efficient
Challenge 2: Explaining Prediction Results

- Explanation is essential for clinicians to
  - Trust prediction results
  - Determine appropriate, tailored interventions
    - E.g., provide transportation for patients who live far from their physicians and have difficulty accessing care
  - Defend their decisions in court if sued for medical negligence
  - Formulate new theories or hypotheses for biomedical research
Challenge 2 – Cont.

- Most machine learning models give no explanation of prediction results
  - Most models are complex
- Prediction accuracy and giving explanation of prediction results are frequently two conflicting goals
- Need to achieve both goals simultaneously
  - Explain prediction results without sacrificing prediction accuracy
Outline

• Our approach to address the challenges
  [HISS’15, HISS’16, HISS’17, JMIR-RP’15, JMIR-RP’17]
  – Efficient and automatic model selection
Current Bayesian Optimization Approach

Test multiple combinations of algorithms and hyper-parameter values;
Build a regression model $R$ to predict a combination’s performance;

While time permits {
    Use $R$ to find a promising combination;
    Evaluate the combination’s performance;
    Update $R$;
}

Return the combination with the best performance;
Integrity Constraints Revisited: From Exact to Approximate Implication

Batya Kenig

Dan Suciu

University of Washington
Main Ideas

- **Major obstacle**: A long time is needed to examine a combination of an algorithm and hyper-parameter values on the entire data set
  - E.g., it takes **two days** on a modern computer to train a champion ensemble model once on 10K patients with 133 independent variables
  - The entire space of algorithms and hyper-parameter values is extremely large
- **Solution**: Perform progressive sampling, filtering, and fine-tuning to quickly narrow the search space
Main Ideas – Cont.

• Use **progressive sampling** to generate a sequence of random samples of the data set, one nested within another
Main Ideas – Cont.

- Conduct inexpensive tests on small samples of the data set to eliminate unpromising algorithms and identify unpromising combinations of hyper-parameter values as early and as much as possible
- Devote more computational resources to fine-tuning promising algorithms and combinations of hyper-parameter values on larger samples of the data set
Main Ideas – Cont.

- The search process is repeated for one or more rounds
- As the sample of the data set expands, the search space shrinks

- In the last round, use (a large part of) the entire data set to find an effective combination of an algorithm and hyper-parameter values
Preliminary Results

• Compared to the state of the art Auto-WEKA automatic selection method on
  – 27 prominent machine learning benchmark data sets
  – A single computer
• On 27 data sets, on average our method
  – Reduces search time by 28 fold
  – Reduces the classification/prediction error rate by 11%
Outline

- Our approach to address the challenges
  - Automatically explain prediction results and suggest tailored interventions
Main Ideas

• A model achieving high accuracy is usually complex and gives no explanation of prediction results

• Challenge: Need to achieve high prediction accuracy as well as explain prediction results

• Key idea: Separate prediction and explanation by using two models concurrently
  – The first model makes predictions and targets maximizing accuracy
  – The second model is rule-based
    • Used to explain the first model’s results rather than make predictions
Main Ideas – Cont.

- The rules used in the second model are mined directly from historical data
- Use one or more rules to explain the prediction result for a patient
- Suggest tailored interventions based on the reasons listed in the rules
Some Results

• Test case: Predicting type 2 diabetes diagnosis within the next year
• Electronic medical record data of 10K patients
• Can explain prediction results for 87% of patients who were correctly predicted by a champion machine learning model to have type 2 diabetes diagnosis within the next year
An Example Rule

- The patient had prescriptions of angiotensin-converting-enzyme (ACE) inhibitor in the past three years **AND** the patient’s maximum body mass index recorded in the past three years is $\geq 35 \rightarrow$ the patient will have type 2 diabetes diagnosis within the next year
  - ACE inhibitor is used mainly for treating hypertension and congestive heart failure
  - Obesity, hypertension, and congestive heart failure are known to correlate with type 2 diabetes
- Example intervention: Enroll the patient in a weight loss program
Thank you 😊
Generating Application-specific In-memory Databases

Cong Yan       Alvin Cheung
University of Washington
Database Application With Object-oriented Programming Interfaces

- Developed using object-oriented languages
  - Java, Python, Ruby, ...

- Object-relational Mapping (ORM) framework
  - Hibernate, Django, Rails

- Example: web applications
Performance Issues
Performance Issues

Profiling result from 12 open-source web apps:

<table>
<thead>
<tr>
<th>Application</th>
<th># github stars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discourse (forum)</td>
<td>22k</td>
</tr>
<tr>
<td>Lobsters (forum)</td>
<td>1.9k</td>
</tr>
<tr>
<td>Gitlab (collaboration)</td>
<td>49k</td>
</tr>
<tr>
<td>Redmine (collaboration)</td>
<td>3k</td>
</tr>
<tr>
<td>Spree (E-commerce)</td>
<td>17k</td>
</tr>
<tr>
<td>ROR Ecommerce</td>
<td>1.7k</td>
</tr>
<tr>
<td>Fulcrum (task mgmt)</td>
<td>697</td>
</tr>
<tr>
<td>Tracks (task mgmt)</td>
<td>3.5k</td>
</tr>
<tr>
<td>Diaspora (social network)</td>
<td>18k</td>
</tr>
<tr>
<td>Onebody (social network)</td>
<td>1.2k</td>
</tr>
<tr>
<td>Openstreetmap (map)</td>
<td>8k</td>
</tr>
<tr>
<td>Fallingfruit (map)</td>
<td>1.1k</td>
</tr>
</tbody>
</table>
Performance Issues

<table>
<thead>
<tr>
<th>Application</th>
<th># github stars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discourse (forum)</td>
<td>22k</td>
</tr>
<tr>
<td>Lobsters (forum)</td>
<td>1.9k</td>
</tr>
<tr>
<td>Gitlab (collaboration)</td>
<td>49k</td>
</tr>
<tr>
<td>Redmine (collaboration)</td>
<td>3k</td>
</tr>
<tr>
<td>Spree (E-commerce)</td>
<td>17k</td>
</tr>
<tr>
<td>ROR Ecommerce</td>
<td>1.7k</td>
</tr>
<tr>
<td>Fulcrum (task mgmt)</td>
<td>697</td>
</tr>
<tr>
<td>Tracks (task mgmt)</td>
<td>3.5k</td>
</tr>
<tr>
<td>Diaspora (social network)</td>
<td>18k</td>
</tr>
<tr>
<td>Onebody (social network)</td>
<td>1.2k</td>
</tr>
<tr>
<td>Openstreetmap (map)</td>
<td>8k</td>
</tr>
<tr>
<td>Fallingfruit (map)</td>
<td>1.1k</td>
</tr>
</tbody>
</table>

Profiling result from 12 open-source web apps:

- 0.1-0.9G of data, 3.3 pages >2sec
- Most slow pages spend >80% on querying data
Why?

- Nested data model
- Predicate involving associated objects
- Program-generated predicate
Chestnut

- Generate app-specific in-memory DB
  - Customize data layout given a workload and a memory budget, minimizing the overall query time

- Specific for database apps using object-oriented programming interface, solves the issues by:
  - Using non-relational storage model
  - Extending index syntax
  - Synthesis-based plan enumeration
1. Nested Data Model

- A mismatch between how the app access data and how data is stored.
  - slow data conversion
  - Example: a chatting app, showing top channels and activities, as well as users for each activity

```ruby
Class Channel:
  has_many: activities => Activity
  ...

Class Activity:
  has_one: user => User
  string type
  ...

Class User:
  ...
```
1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```ruby
Channel.includes(activities, includes(user)).order(id).limit(50)
```
1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```sql
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```sql
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (...);
SELECT * FROM user WHERE id IN (...);
```
1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```ruby
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```sql
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (...);
SELECT * FROM user WHERE id IN (...);
```
1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

Channel.includes(activities, includes(user)).order(id).limit(50)

```
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (...);
SELECT * FROM user WHERE id IN (...);
```
1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```
Channel.includes(activities, includes(user)).order(id).limit(50)
```

SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (...);
SELECT * FROM user WHERE id IN (...);
1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```sql
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```sql
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (...);
SELECT * FROM user WHERE id IN (...);
```
1. Nested Data Model

- A mismatch between how the app access data and how data is stored
- slow data conversion

```sql
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```sql
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (...);
SELECT * FROM user WHERE id IN (...);
```
1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```ruby
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```sql
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (...);
SELECT * FROM user WHERE id IN (...);
```
1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```ruby
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```sql
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (...);
SELECT * FROM user WHERE id IN (...);
```
1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```javascript
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```sql
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (...);
SELECT * FROM user WHERE id IN (...);
```

55 sec

1.7 sec
Chestnut: Using Non-relational Storage Model

- Storing data as array of objects and nested objects, and return objects
Chestnut: Using Non-relational Storage Model

- Storing data as array of objects and nested objects, and return objects

Data conversion: C++ object -> Ruby object
Chestnut: Using Non-relational Storage Model

- Storing data as array of objects and nested objects, and return objects

Data conversion: C++ object -> Ruby object

1.5 sec

2.3 sec
Chestnut: Using Non-relational Storage Model

- Storing data as array of objects and nested objects, and return objects

Data conversion: C++ object -> Ruby object

1.5 sec

2.3 sec

15x speedup!
2. Query Predicate Involving Associated Objects

- Partial index supported by relational databases

Class Channel:
  has_many: activities => Activity
  string status
  ...

Class Activity:
  has_one: user => User
  string type
  ...

Class User:
  ...

Channel.where(status='active').order(id)

index: channel(id, status='active')
2. Query Predicate Involving Associated Objects

- Partial index **not** supported by relational databases

Class Channel:
  has_many: activities => Activity
  string status
...

Class Activity:
  has_one: user => User
  string type
...

Class User:
...

Channel.where(
  exists(activities, type='msg'))
.order(id)

index: ??

C1  C2  C3  C4
Chestnut: Extending Index Syntax

- Such partial index is considered by Chestnut

Class Channel:
  has_many: activities => Activity
  string status
  ...

Class Activity:
  has_one: user => User
  string type
  ...

Class User:
  ...

Channel.where(
  exists(activities, type='msg'))
 .order(id)

index:
channel(id, exists(activities, type='msg'))
Chestnut: Extending Index Syntax

- Allow associated object’s field to appear in keys and predicates

```plaintext
index:
channel(id, exists(activities, type='msg'))
```

```
C1  C2  C3  C4
```

```plaintext
sorted_array: channel(activities.id)
```

```
C2  C1  C2  C4
```
3. Program-generated Query Predicate

- Predicates are generated by chained function calls, often containing overlapping or redundant predicates.
  - E.g., a webpage showing ‘join’ or ‘leave’ (and non-'msg”) activities created or updated recently
3. Program-generated Query Predicate

- Predicates are generated by chained function calls, often containing overlapping or redundant predicates.
  - E.g., a webpage showing ‘join’ or ‘leave’ (and non-’msg’) activities created or updated recently

```
SELECT * FROM activity WHERE type!='msg' AND (type='join' or type='leave') AND (created>? or updated>?)
```
3. Program-generated Query Predicate

- Predicates are generated by chained function calls, often containing overlapping or redundant predicates.
  - E.g., a webpage showing ‘join’ or ‘leave’ (and non-‘msg’) activities created or updated recently

```
SELECT * FROM activity WHERE type!='msg' AND (type='join' OR type='leave') AND (created>?? or updated>??)
```

index1: activity(type, created)
index2: activity(type, updated)
3. Program-generated Query Predicate

SELECT * FROM activity WHERE type!='msg' AND (type='join' or type='leave') AND (created>? or updated>?)

index1: activity(type, created)
index2: activity(type, updated)
3. Program-generated Query Predicate

index1: activity(type, created)
index2: activity(type, updated)

```
SELECT * FROM activity WHERE type!='msg' AND (type='join' or type='leave') AND (created?> or updated?>)
```

Workers Planned: 2

→ Parallel Seq Scan on activities (cost=0.00..479177.81 rows=81168 width=368)
Filter: ((type <> 9) AND ((type = 2) OR (type = 3)) AND ((created > '2018-12-12') OR (updated > '2018-12-12')))
3. Program-generated Query Predicate

```
SELECT * FROM activity WHERE type!='msg' AND (type='join' or type='leave') AND (created>? or updated>?)
```

index1: activity(type, created)
index2: activity(type, updated)

Seq scan: 2.6 sec
3. Program-generated Query Predicate

```sql
SELECT * FROM activity WHERE type!='msg' AND (type='join' or type='leave') AND (created>?? or updated>??)
Seq scan: 2.6 sec
```

```sql
SELECT * FROM activity WHERE type in ('join', 'leave') AND (created>?? or updated>??)
```
3. Program-generated Query Predicate

```sql
SELECT * FROM activity WHERE type!='msg' AND (type='join' or type='leave') AND (created>? or updated>?)
```

-----

```sql
SELECT * FROM activity WHERE type in ('join', 'leave') AND (created>? or updated>?)
```

---

**Workers Planned:** 2

```
--> Parallel Bitmap Heap Scan on activity (cost=5855.50..410948.41 rows=94795 width=368)
 Recheck Cond: ((type = ANY ('[2,3]'::bigint[])) AND (created>'2018-12-12')) OR ((type = ANY ('[2,3]'::bigint[])) AND (updated>'2018-12-12'))
 Filter: (type = ANY ('[2,3]'::bigint[]))
 --> BitmapOr (cost=5855.50..5855.50 rows=227604 width=0)
   --> Bitmap Index Scan on idx_type_created (cost=0.00..23.54 rows=669 width=0)
     Index Cond: ((type = ANY ('[2,3]'::bigint[])) AND (created>'2018-12-12'))
 --> Bitmap Index Scan on idx_type_updated (cost=0.00..5718.20 rows=226935 width=0)
     Index Cond: ((type = ANY ('[2,3]'::bigint[])) AND (updated>'2018-12-12'))
```

**Seq scan: 2.6 sec**
3. Program-generated Query Predicate

index1: activity(type, created)
index2: activity(type, updated)

```
SELECT * FROM activity WHERE type!='msg' AND (type='join' or type='leave') AND (created>? or updated>?)
```

Seq scan: 2.6 sec

```
SELECT * FROM activity WHERE type in ('join', 'leave') AND (created>? or updated>?)
```

Use index: 0.5 sec
Chestnut: Synthesis-based Plan Enumeration

- Rules are not enough to handle all cases!
Chestnut: Synthesis-based Plan Enumeration

- Rules are not enough to handle all cases!

- Enumerate plans
  - From small-size plans to larger-size

```python
r = index1.scan(('join', 2018-01-01), ('msg', ∞))
...
```
```python
r1 = index1.scan(('join', 2018-01-01), ('join', ∞))
r2 = index2.scan(('leave', 2018-01-01), ('leave', ∞))
...
r = distinct(union(r1, r2, r3, r4))
```
Chestnut: Synthesis-based Plan Enumeration

- Rules are not enough to handle all cases!

- Enumerate plans
  - From small-size plans to larger-size

- Verify each plan against query
  - Symbolic execution

```c
r=\text{index1}.\text{scan}((\text{`join',2018-01-01}), (\text{`msg', }, \infty))
```

```c
\ldots
```

```c
r1=\text{index1}.\text{scan}((\text{`join',2018-01-01}), (\text{`join', }, \infty))
```

```c
r2=\text{index2}.\text{scan}((\text{`leave',2018-01-01}), (\text{`leave', }, \infty))
```

```c
\ldots
```

```c
r=\text{distinct}(\text{union}(r1, r2, r3, r4))
```
Chestnut: Synthesis-based Plan Enumeration

- Rules are not enough to handle all cases!

- Enumerate plans
  - From small-size plans to larger-size

- Verify each plan against query
  - Symbolic execution

```
\[ r = \text{index1.scan((‘join’, 2018-01-01), (‘msg’, ∞))} \]
```

```
\[ r = \text{distinct(union(r1, r2, r3, r4))} \]
```

- Slower than existing query optimizer, but sometimes can find better plans
Chestnut: Synthesis-based Plan Enumeration

- Rules are not enough to handle all cases!

- Enumerate plans
  - From small-size plans to larger-size

- Verify each plan against query
  - Symbolic execution

- Slower than existing query optimizer, but sometimes can find better plans
Workflow

Input

Query workload
Mem budget

Chestnut

Q1
storage
plan
Q2
storage
plan

pruning
ILP Solver
Code gen

Output

C++ code for app-specific in-memory DB
Workflow

Input

Query workload
Mem budget

Chestnut

Q1
storage
plan

pruning

ILP Solver

Code gen

Output

C++ code for application-specific in-memory DB

Constraint:
- Each query plan uses some data structures
- The used data structures is within mem budget

Goal:
minimize Σ query time
Workflow

Input

Query workload
Mem budget

Chestnut

Q1

storage
plan

Q2

storage
plan

pruning
ILP Solver
Code gen

Output

C++ code for app-specific in-memory DB
Evaluation

- 3 open-source popular web applications built with Rails
  - kandan: Hipchat-like chatting app
  - redmine: GitHub-like project management
  - lobsters: Hackernews-like forum app

- Compare against:
  - Original setting with MySQL (in-memory)
  - PostgreSQL + automatic indexer (in-memory)
  - Hyper + automatic indexer
  - Chestnut
Evaluation

- 3 open-source popular web applications built with Rails

Chestnut running time:
- kandan: 1min
- redmine: 10min
- lobsters: 54min

(average query time with the same memory)
shaded area: convert relational data into objects
Conclusion

- Chestnut generates in-memory app-specific database
  - Customize data layout given a workload and a memory budget, optimizing the overall query performance
- Uses non-relational storage model, storing data as objects and nested objects
- Extends index syntax, allowing associated object’s field in keys and predicates
- Synthesis-based plan enumeration, enumerate plans and verify each plan
- Achieve significant speedup in real-world web apps
  - >4.8x avg speedup compared to using state-of-the-art in-memory databases
Evaluation

- 3 open-source popular web applications built with Rails

![Graph showing performance improvements](image)

- Chestnut running time:
  - kandan: 1min
  - redmine: 10min
  - lobsters: 54min

*average query time with the same memory*

*shaded area: convert relational data into objects*
Workflow

Input

Query workload
Mem budget

Chestnut

Q1
storage
plan
...
Q2
storage
plan
...
pruning
ILP Solver
Code gen

Output

C++ code for app-specific in-memory DB
3. Program-generated Query Predicate

```sql
SELECT * FROM activity WHERE type!='msg' AND (type='join' or type='leave') AND (created>? or updated>?)
```

Seq scan: 2.6 sec

```sql
SELECT * FROM activity WHERE type in ('join', 'leave') AND (created>? or updated>?)
```

Use index: 0.5 sec

>5x speedup!