Query Optimization for Data Analysis

Immanuel Trummer

Cornell University
Popular Data Analysis Skills

Which skills should data scientists have in 2016?
Analyzed ca. 3500 LinkedIn job openings for data scientists.

http://www.crowdflower.com/blog/what-skills-should-data-scientists-have-in-2016
Which of the following tools do you use?

Please select only those tools which you use in your current role on a regular basis (i.e., about every week).
Classical Query Optimization

- Tuning Knobs: Operation order, Operators
- Problem Model
- Query Optimization
- Optimal Plan(s)
- Cost Metric: Execution time
Classical Query Optimization

Problem Model

Tuning Knobs
- Operation order
- Operators

Cost Metric
- Execution time

NP-Hard to Solve

NP-Hard to Approximate

Query Optimization

Optimal Plan(s)
Classical Query Optimization

Fixed processing platform, few operators available

Optimization Context

Context in 1979

Problem Model

Tuning Knobs
Operation order
Operators

NP-Hard to Approximate

Query Optimization

Query

Cost Metric
Execution time

NP-Hard to Solve

Optimal Plan(s)
Modern Query Optimization

Cloud Computing

Optimization Context

Tuning knobs
Operation order
Operators

NP-Hard to Approximate

Query

NP-Hard to Solve

Optimal Plan(s)

Problem Model

Cost Metric
Execution time

Fees

Time

Confidence

Fees

Time

Precision

Time

Approximation

Query Optimization

Crowdsourcing

Modern Query Optimization
Modern Query Optimization

- **Cloud Computing**: Cloud platform, fees, time.
- **Crowdsourcing**: Crowd workers, sample sizes, confidence, fees, execution time.
- **Problem Model**: Query, tuning knobs, operators, cloud platform.
- **Cost Metrics**: Execution time, monetary fees, result precision, energy consumption.
Modern Query Optimization

Cloud Computing

Tuning knobs
- Operation order
- Operators
- Cloud platform
- Crowd workers
- Sample sizes

Large Search Space

Problem Model

Optimization Context

Cost Metrics
- Execution time
- Monetary fees
- Result precision
- Confidence
- Energy consumption

Multiple Cost Metrics
Example: Google

Query

Types ⨝ Properties ⨝ Entities ⨝ Opinions ⨝ Models …
Example: Google

Query

Types Properties Entities Opinions Models ...

Query Plan

Execution Platform
Example: Google

Query

Types ⨝ Properties ⨝ Entities ⨝ Opinions ⨝ Models …

Query Plan

Execution Platform
Example: Google

Types ▷ Properties ▷ Entities ▷ Opinions ▷ Models ...

Query

Query Plan

Execution Platform
Example: Google

Types ∙ Properties ∙ Entities ∙ Opinions ∙ Models …

Query Plan

Query

Execution Platform
Example: Google

**Query**

- Types
- Properties
- Entities
- Opinions
- Models

**Query Plan**

- Query
- Execution Platform

- O
- P
- T
- E
- M
Example: Google

**Query**

**Types** ⨝ **Properties** ⨝ **Entities** ⨝ **Opinions** ⨝ **Models** …

**Query Plan**

**Cost**

*Time: 1 day*
Example: Google

Query

Types ⊗ Properties ⊗ Entities ⊗ Opinions ⊗ Models …

Query Plan

Cost

Time: 4 hours

Execution Platform
Example: Google

Query

Types Properties Entities Opinions Models …

Query Plan

Left-Deep Plan

Execution Platform

Cost

Time: 4 hours
Example: Google

Types • Properties • Entities • Opinions • Models …

Query Plan

M•P•T•E

Query

Cost

Time: 3 hours

Execution Platform
Example: Google

- Types
- Properties
- Entities
- Opinions
- Models

Query Plan

Bushy Plan

Execution Platform

Cost

Time: 3 hours
Example: Google

Types ⨝ Properties ⨝ Entities ⨝ Opinions ⨝ Models …

Query Plan

MnPxE

MP

TxE

Execution Platform

10 % Sample

Cost

Time:
3 hours
Example: Google

- **Types**: M, P, T
- **Properties**: MxP, TxE
- **Entities**: O
- **Opinions**: 10% Sample
- **Models**: Execution Platform

**Query Plan**

**Cost**

- **Time**: 1 hour
Example: Google

Query

Types ⊗ Properties ⊗ Entities ⊗ Opinions ⊗ Models …

Query Plan

Quality

Time: 1 hour

Completeness: 10 %

Execution Platform

M ⊗ P ⊗ T ⊗ E ⊗ O

10 % Sample
Example: Google

Types × Properties × Entities × Opinions × Models …

Query Plan

M × P × T × O

10 % Sample

Execution Platform

Quality

Time: 30 minutes

Completeness: 10 %
Example: Google

**Query**
- Types
- Properties
- Entities
- Opinions
- Models

**Query Plan**
- M x P
- T x O
- M x P
- T x O

**Execution Platform**

**Quality**
- **Time:** 30 minutes
- **Completeness:** 10%
- **Resources:** 500 CHrs

**10% Sample**
Example: Google

Query

Types × Properties × Entities × Opinions × Models ...

Query Plan

M × P × T × O

10 % Sample

M × P

T × O

Query

Time:
34 minutes

Completeness:
10 %

Quality

Execution Platform

Resources:
283 CHrs
Example Summary

Types ★ Properties ★ Entities ★ Opinions ★ Models …

Join Order

Decisions

Platform vs.

Sample Sizes

vs.

Metrics

Execution Time

Resource Consumption

Result Completeness
Example Summary

Query

Types ✗ Properties ✗ Entities ✗ Opinions ✗ Models …

Decisions

Join Order vs. Platform vs. Sample Sizes

Classical Query Optimization

Execution Time

Metrics

Resource Consumption vs. Result Completeness
Example: Google

- Types
- Entities
- Properties
- Opinions
- Models

Query Optimizer

Cost Tradeoffs

Time → Resources
Example: Google

Cost Tradeoffs

Types \ Properties \ Entities

Query Optimizer

Resources

Time
Example: Google

- Types
- Properties
- Entities
- Opinions
- Models

Query Optimizer

Optimal Tradeoffs

Resources vs. Time
Example: Google

Prior Work: Takes Hours!!

Types
Entities
Properties
Models

Query Optimizer

Optimal Tradeoffs
Dissertation Overview
Dissertation Overview

Pre-Computation

Run Time

Query

Optimizer

SLOW

Optimal Plans
Dissertation Overview

Pre-Computation

Before

Run Time

Query Template

Optimizer

SLOW

Potentially Optimal Plans

Query

Optimizer

SLOW

Optimal Plans
Dissertation Overview

Pre-Computation

Optimality Guarantees

Exhaustive

Before

Query Template

Potentially Optimal Plans

Run Time

Query

SLOW

Optimal Plans

Optimizer

SLOW

Optimizer

Optimal Plans

Optimality Guarantees
Dissertation Overview

Before Optimizer

Run Time

Pre-Computation

Optimality Guarantees

Approximate

Optimal Plans

Near-Optimal Plans

Optimal Plans

Potential Optimal Plans

Setup

Query Template

Query

SLOW

FASTER

Exhaustive

Optimality Guarantees

Before Optimizer

Run Time

Pre-Computation

Approximate

Optimal Plans

Near-Optimal Plans

Optimal Plans

Potential Optimal Plans

Setup

Query Template

Query

SLOW

FASTER

Exhaustive

Optimality Guarantees
Dissertation Overview

Before

Pre-Computation

Run Time

Exhaustive

Approximate

Incremental

Optimizer

Potential Optimal Plans

Query Template

SLOW

Optimizer

Optimal Plans

Query

FASTER

Optimizer

Near-Optimal Plans

Query

FASTER

Optimizer

Acceptable Plans ...
Near-Optimal Plans ...
Optimal Plans

Query

FASTER

Optimizer

Near-Optimal Plans

Query

Near-Optimal Plans

Query

Optimal Plans

Optimality Guarantees
Dissertation Overview

Pre-Computation

Optimality Guarantees

Before

Optimal Plans

Potentially

Optimal Plans

SLOW

Exhaustive

Run Time

Approximate

Incremental

Random

Query Template

Optimizer

FASTER

Optimizer

FASTER

Optimizer

FAST

Query

Query

Query

Query

Acceptable Plans ...

Near-Optimal Plans ...

Optimal Plans

SLOW

Near-Optimal Plans

FAST

Best Effort

Plans

FAST
Dissertation Overview

Before Optimizer Query Template

SLOW

Potentially
Optimal Plans

Optimization Platform

After Optimizer Query

SLOW

Near-Optimal Plans

FASTER

Acceptable Plans ...
Near-Optimal Plans ...
Optimal Plans

FAST

Best Effort Plans

Pre-Computation

Run Time

Optimality Guarantees

Exhaustive

Approximate

Incremental

Random
Dissertation Overview

- Pre-Computation
  - Optimizer
  - Query Template
  - Optimally Optimal Plans

- Optimization Platform
  - Solver
  - Query
  - Near-Optimal Plans

- Run Time
  - Optimizer
  - Query
  - Optimal Plans
  - Near-Optimal Plans

- Exhaustive
- Approximate
- Incremental
- Random

- Optimality Guarantees
- Before
- After
Dissertation Overview

Pre-Computation

Run Time

- Optimizer
  - SLOW
  - Optimizer
    - Potentially Optimal Plans
  - Optimizer
    - Near-Optimal Plans
  - Solver
    - Near-Optimal Plans

- Optimization Platform
  - Parallel
    - *Plans

- Solver
  - Near-Optimal Plans
  - Acceptable Plans ...
  - Optimal Plans

- Optimizer
  - FAST
  - Best Effort Plans

- Optimizer
  - FAST
  - Acceptable Plans ...
  - Near-Optimal Plans ...
  - Optimal Plans

- Optimizer
  - FAST
  - Plans

- Optimizer
  - FAST
  - Template

- Optimizer
  - SLOW
Dissertation Overview

Pre-Computation

Run Time

Before

After

Optimizer

SLOW

FASTER

Solver

Near-Optimal Plans

Optimal Plans

Approximate

Optimality Guarantees

Exhaustive

Random

Quantum Optimization Platform

Before Optimizer

Template

Query

Best Effort Plans

Quantum

FASTER

Parallel

* Plans

Acceptable Plans ...
Near-Optimal Plans ...
Optimal Plans

FAST

Near-Optimal Plans

Optimal Plans

Best Effort Plans

FAST

OPTIMIZER

OPTIMIZER

OPTIMIZER

OPTIMIZER

OPTIMIZER

OPTIMIZER
Dissertation Overview

Pre-Computation

- Before
  - VLDB'15
  - VLDB'16

Run Time

- SLOW
  - Optimizer
  - Template

- FASTER
  - Optimizer
  - FASTER

Solver

- Near-Optimal Plans
  - FASTER

- Optimal Plans

- Near-Optimal Plans

- Acceptable Plans ...
  - Near-Optimal Plans ...
  - Optimal Plans

- Best Effort Plans

- Parallel

- Quantum Optimization Platform

Optimality Guarantees

- Exhaustive
- Approximate
- Incremental
- Random
Dissertation Overview

Pre-Computation

**Run Time**

- **Exhaustive Optimizer**
  - Optimal Plans
  - Before

- **Approximate Optimizer**
  - Near-Optimal Plans

- **Incremental Optimizer**
  - Acceptable Plans...

- **Random Optimizer**
  - Near-Optimal Plans...

**Optimality Guarantees**

- **VLDB’15**
  - Best of VLDB
  - Potentially Optimal Plans

- **VLDBJ’16**
  - Query Template

- **SIGMOD’14**
  - Near-Optimal Plans
  - FASTER

- **ArXiv**
  - Solver
  - FASTER

- **SIGMOD’15**
  - Near-Optimal Plans
  - Faster

- **VLDB’16**
  - Near-Optimal Plans
  - FASTER

- **VLDB’16**
  - Best Effort Plans
  - FAST

Optimization Platform

- **Parallel Platform**

**Quantum Optimization Platform**

**VLDB’16**

**SIGMOD’14**

**VLDB’15**

**VLDBJ’16**

**SIGMOD’15**

**ArXiv**

**VLDB’16**

**VLDB’15**

**VLDBJ’16**
Classical Query Optimization

Query: R⨝S⨝T

One Cost Metric
Classical Query Optimization

Query: $R \bowtie S \bowtie T$

One Cost Metric

Optimal Plan

Sub-Optimal Plans
Classical Query Optimization

Query: $R \bowtie S \bowtie T$

One Cost Metric
Classical Query Optimization

Query: R⨝S⨝T

One Cost Metric
Classical Query Optimization

Query: $R \otimes S \otimes T$

One Cost Metric
Classical Query Optimization

Query: $R \land S \land T$

One Cost Metric

$R \land S$  $S \land T$  $R \land T$

$R$  $S$  $T$
Classical Query Optimization

Query: $R \bowtie S \bowtie T$

One Cost Metric
Classical Query Optimization

Query: $R \bowtie S \bowtie T$

One Cost Metric
Classical Query Optimization

Query: $R \bowtie S \bowtie T$

One Cost Metric
Classical Query Optimization

Query: R \bowtie S \bowtie T

One Cost Metric
Multi-Objective Query Optimization

Query: $R \bowtie S \bowtie T$

Two Cost Metrics

Optimal Plans
Parametric Query Optimization

Query: R⨝S⨝T

One Cost Metric
One Unknown
## Parallelization Challenges

### Sub-Queries

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S,U</td>
<td>S,V</td>
<td>T,U</td>
<td>T,V</td>
<td>U,V</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>
Parallelization Challenges

Sub-Queries

Dependencies
Parallelization Challenges

Sub-Queries

Dependencies

Non-Serial Polyadic Dynamic Programming
Prior Art

Shared Memory

Worker 1
Optimize

Worker 2
Optimize

Worker N-1
Optimize

Worker N
Optimize

Query

Master
Assign Sub-Queries

Sub-Queries

T,U,V
R,T,V
S,T,U,V
R,S,V
R,V
R,T
S,T
S,V
R

Partial Plans

Shared Memory
Prior Art

Query: R • S • T • U • V

Optimize: Worker 1, Worker 2, ..., Worker N

Exponential Communication

Exponential Time
Experimental Results

Experimental Setup
Randomly generated queries; two execution cost metrics; approximated optimization
Spark 1.5 on Yarn 2.7; 100 nodes with 2x Intel Xeon 2.6 GHz, 128 GB memory

Number of Workers vs. Optimization Time (Seconds)

Scalability of previous parallel query optimization algorithms
New Approach

Master
Assign Partitions

Query + Partition ID

Worker 1
Optimize

Worker N
Optimize

Query + Partition ID

Best Plan
New Approach

Worker 1
Optimize

Worker N
Optimize

Query + Partition ID, 1

Query + Partition ID

Polynomial Complexity

Polynomial Communication

No Inter-Worker Communication

Master
Assign Partitions

Query

Best Plan
Left-Deep Query Plans

Left-Deep Plan ~
Table Order
Left-Deep Query Plans

Left-Deep Plan ~

Table Order

Plan Space

Partitioning via Unary Constraints

First: **R**

First: **S**

First: **T**

First: **U**

First: **V**
Left-Deep Query Plans

Left-Deep Plan ~
Table Order

Partitioning via Unary Constraints

First: R
First: S
First: T
First: U
First: V
Left-Deep Query Plans

Left-Deep Plan ~
Table Order

Partitioning via Unary Constraints

First: \( R \)  
First: \( S \)  
First: \( T \)  
First: \( U \)  
First: \( V \)

Sub-Queries

<table>
<thead>
<tr>
<th>Sub-Queries</th>
<th>( RS )</th>
<th>( RS )</th>
<th>( RS )</th>
<th>( RS )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( TUV )</td>
<td>( T,U,V )</td>
<td>( R,T,U,V )</td>
<td>( S,T,U,V )</td>
<td>( R,S,T,U,V )</td>
</tr>
<tr>
<td>( TUV )</td>
<td>( T,V )</td>
<td>( R,T,V )</td>
<td>( S,T,V )</td>
<td>( R,S,T,V )</td>
</tr>
<tr>
<td>( TUV )</td>
<td>( U,V )</td>
<td>( R,U,V )</td>
<td>( S,U,V )</td>
<td>( R,S,U,V )</td>
</tr>
<tr>
<td>( TUV )</td>
<td>( T,U )</td>
<td>( R,T,U )</td>
<td>( S,T,U )</td>
<td>( R,S,T,U )</td>
</tr>
<tr>
<td>( TUV )</td>
<td>( R,V )</td>
<td>( S,V )</td>
<td>( R,S,V )</td>
<td></td>
</tr>
<tr>
<td>( TUV )</td>
<td>( R,U )</td>
<td>( S,U )</td>
<td>( R,S,U )</td>
<td></td>
</tr>
<tr>
<td>( TUV )</td>
<td>( R,T )</td>
<td>( S,T )</td>
<td>( R,S,T )</td>
<td></td>
</tr>
<tr>
<td>( TUV )</td>
<td></td>
<td></td>
<td>( R,S )</td>
<td></td>
</tr>
</tbody>
</table>
Left-Deep Query Plans

Left-Deep Plan ~ Table Order

Partitioning via Unary Constraints

Partitioning

First: **R**

First: **S**

First: **T**

First: **U**

First: **V**

Sub-Queries

<table>
<thead>
<tr>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>T,U,V</strong></td>
<td><strong>T,V</strong></td>
<td><strong>U,V</strong></td>
<td><strong>T,U</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S,T,U,V</strong></td>
<td><strong>S,T,V</strong></td>
<td><strong>S,U,V</strong></td>
<td><strong>S,T,U</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R,V</strong></td>
<td><strong>S,V</strong></td>
<td><strong>R,U</strong></td>
<td><strong>R,S,V</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R,U</strong></td>
<td><strong>S,U</strong></td>
<td><strong>R,T</strong></td>
<td><strong>R,S,U</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
<th><strong>TUV</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S,T</strong></td>
<td><strong>R,S,T</strong></td>
<td><strong>R,S</strong></td>
<td><strong>R,S</strong></td>
</tr>
</tbody>
</table>
Left-Deep Query Plans

Plan Space

Left-Deep Plan ~
Table Order

Partitioning via Unary Constraints

First: R
First: S
First: T
First: U
First: V

Sub-Queries

Partitioning

RS

TUV
TUV
TUV
TUV
TUV
TUV
TUV
TUV
TUV

RS

T,U,V
T,U,V
T,U,V
T,U,V
T,U,V
T,U,V
T,U,V
T,U,V
T,U,V

RS

R,T,U,V
R,T,V
R,U,V
R,T,U
R,T,V
R,U,V
R,T,U
R,T,V
R,U,V

RS

S,T,U,V
S,T,V
S,U,V
S,T,U
S,T,V
S,U,V
S,T,U
S,T,V
S,U,V

RS

R,S,T,U,V
R,S,T,V
R,S,U,V
R,S,T,U
R,S,T,V
R,S,U,V
R,S,T,U
R,S,T,V
R,S,U,V

RS

R,T,U,V
R,U
R,U
R,T
R,T

Left-Deep Query Plans

Plan Space

Sub-Queries

R, S, T, U, V

R, T, U, V

S, T, U, V

R, T, V

S, T, V

R, U, V

S, U, V

R, S, T, U

R, S, V

R, S, U

R, S, T

R, S

Parallelism ~ Nr. Tables

Time / 2

Partitioning via Unary Constraints
Left-Deep Query Plans

Left-Deep Plan ~ Table Order
Left-Deep Query Plans

Left-Deep Plan ~ Table Order

Partitioning via Binary Constraints

R Before S

S Before R
Left-Deep Query Plans

Left-Deep Plan ~

Table Order

Partitioning via Binary Constraints

R Before S

S Before R
Left-Deep Query Plans

Plan Space

Left-Deep Plan ~ Table Order

Partitioning via Binary Constraints

R Before S

S Before R

Sub-Queries

TUV

TUV

TUV

TUV

TUV

TUV

TUV

TUV

TUV

TUV
Left-Deep Query Plans

Left-Deep Plan ~ Table Order

Partitioning via Binary Constraints

Plan Space

Sub-Queries

Partitioning

TUV

T,U,V

R,T,U,V

S,T,U,V

R,S,T,U,V

TUV

T,V

R,T,V

S,T,V

R,S,T,V

TUV

U,V

R,U,V

S,U,V

R,S,U,V

TUV

T,U

R,T,U

S,T,U

R,S,T,U

TUV

R,U

S,U

R,S,U

TUV

R,T

S,T

R,S,T

TUV

R,S

S,S

R,R

S,S

TUV

TUV

T,U

R,T

S,T

R,S
Left-Deep Query Plans

Left-Deep Plan ~ Table Order

Partitioning via Binary Constraints

Partitioning

R Before S
- T Before U
- U Before T

S Before R
- T Before U
- U Before T

Sub-Queries

TUV
- T,U,V

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V

TUV
- T,U

TUV
- T,V

TUV
- U,V
Left-Deep Query Plans

**Plan Space**

- **Left-Deep Plan ~**
  - Table Order

**Partitioning**

**Partitioning via Binary Constraints**

- **R Before S**
  - T Before U
  - U Before T

- **S Before R**
  - T Before U
  - U Before T

**Sub-Queries**

- **TUV**
  - RS: T,U,V
  - T,U

- **TUV**
  - T,V

- **TUV**
  - U,V

- **TUV**
  - R,T,U,V

- **TUV**
  - R,V

- **TUV**
  - R,U

- **TUV**
  - R,T
Left-Deep Query Plans

Left-Deep Plan ~ Table Order

Partitioning via Binary Constraints

<table>
<thead>
<tr>
<th>Sub-Queries</th>
<th>RS</th>
<th>RS</th>
<th>RS</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUV</td>
<td>T,U</td>
<td>R,U</td>
<td>S,U</td>
<td>R,S</td>
</tr>
<tr>
<td>TUV</td>
<td>T,U</td>
<td>R,U</td>
<td>S,U</td>
<td>R,S</td>
</tr>
</tbody>
</table>

Partitioning:
- R Before S
  - T Before U
  - U Before T
- S Before R
  - T Before U
  - U Before T
Left-Deep Query Plans

Left-Deep Plan ~ Table Order

Partitioning via Binary Constraints

Parallelism \( \times 2^{nrPartitionLevels} \)

Time \(*(3/4)^{nrPartitionLevels} \)
Bushy Query Plans

Bushy Plan ~ Binary Tree

Plan Space
Bushy Query Plans

Bushy Plan ~
Binary Tree
Bushy Query Plans

Bushy Plan ~ Binary Tree

Partitioning via Ternary Constraints

Path \( R \rightarrow \text{Root}: T \) after \( U \)

Path \( R \rightarrow \text{Root}: T \) not after \( U \)
Bushy Query Plans

Bushy Plan ~ Binary Tree

Partitioning via Ternary Constraints

Path $\text{R} \rightarrow \text{Root}: \text{T after U}$

Path $\text{R} \rightarrow \text{Root}: \text{T not after U}$
Bushy Query Plans

Bushy Plan ~ Binary Tree

Partitioning via Ternary Constraints

Path $R \rightarrow \text{Root}: T \text{ after } U$

Path $R \rightarrow \text{Root}: T \text{ not after } U$

Sub-Queries

<table>
<thead>
<tr>
<th>TUV</th>
<th>RS</th>
<th>TUV</th>
<th>RS</th>
<th>TUV</th>
<th>RS</th>
<th>TUV</th>
<th>RS</th>
<th>TUV</th>
<th>RS</th>
</tr>
</thead>
</table>
Bushy Query Plans

Bushy Plan ~ Binary Tree

Partitioning via Ternary Constraints

Path $R \rightarrow \text{Root}: T$ after $U$

Path $R \rightarrow \text{Root}: T$ not after $U$

Sub-Queries

<table>
<thead>
<tr>
<th>TUV</th>
<th>Sub-Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUV</td>
<td>T,U,V</td>
</tr>
<tr>
<td>TUV</td>
<td>T,V</td>
</tr>
<tr>
<td>TUV</td>
<td>U,V</td>
</tr>
<tr>
<td>TUV</td>
<td>T,U</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TUV</th>
<th>Sub-Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUV</td>
<td>R,T,U,V</td>
</tr>
<tr>
<td>TUV</td>
<td>R,T,V</td>
</tr>
<tr>
<td>TUV</td>
<td>R,U,V</td>
</tr>
<tr>
<td>TUV</td>
<td>R,T,U</td>
</tr>
<tr>
<td>TUV</td>
<td>R,V</td>
</tr>
<tr>
<td>TUV</td>
<td>R,U</td>
</tr>
<tr>
<td>TUV</td>
<td>R,T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TUV</th>
<th>Sub-Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUV</td>
<td>S,T,U,V</td>
</tr>
<tr>
<td>TUV</td>
<td>S,T,V</td>
</tr>
<tr>
<td>TUV</td>
<td>S,U,V</td>
</tr>
<tr>
<td>TUV</td>
<td>S,T,U</td>
</tr>
<tr>
<td>TUV</td>
<td>S,V</td>
</tr>
<tr>
<td>TUV</td>
<td>S,U</td>
</tr>
<tr>
<td>TUV</td>
<td>S,T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TUV</th>
<th>Sub-Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUV</td>
<td>R,S,T,U,V</td>
</tr>
<tr>
<td>TUV</td>
<td>R,S,T,V</td>
</tr>
<tr>
<td>TUV</td>
<td>R,S,U,V</td>
</tr>
<tr>
<td>TUV</td>
<td>R,S,T,U</td>
</tr>
<tr>
<td>TUV</td>
<td>R,S,V</td>
</tr>
<tr>
<td>TUV</td>
<td>R,S,U</td>
</tr>
<tr>
<td>TUV</td>
<td>R,S,T</td>
</tr>
<tr>
<td>TUV</td>
<td>R,S</td>
</tr>
</tbody>
</table>
Bushy Query Plans

Bushy Plan ~ Binary Tree

Partitioning via Ternary Constraints

Path $R \rightarrow \text{Root}: \ T \text{ after } U$
Path $R \rightarrow \text{Root}: \ T \text{ not after } U$

Sub-Queries

<table>
<thead>
<tr>
<th>Sub-Queries</th>
<th>RS</th>
<th>RS</th>
<th>RS</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUV</td>
<td>T,V</td>
<td>R,T,V</td>
<td>S,T,V</td>
<td>R,S,T,V</td>
</tr>
<tr>
<td>TUV</td>
<td>R,V</td>
<td>S,V</td>
<td>R,S,V</td>
<td></td>
</tr>
<tr>
<td>TUV</td>
<td>R,U</td>
<td>S,U</td>
<td>R,S,U</td>
<td></td>
</tr>
<tr>
<td>TUV</td>
<td>R,T</td>
<td>S,T</td>
<td>R,S,T</td>
<td></td>
</tr>
<tr>
<td>TUV</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Plan Space

Bushy Plan ~ Binary Tree
Bushy Query Plans

Bushy Plan ~
Binary Tree

Partitioning via Ternary Constraints

Path $R \rightarrow \text{Root}: T$ after $U$
Path $R \rightarrow \text{Root}: T$ not after $U$
Bushy Query Plans

Parallelism *2
Sub-Queries *(7/8)
Time *(21/27)

Space

Path R→Root: T
Path R→Root: not T

Bushy Plan ~

Partitioning via Ternary Constraints

nrPartitionLevels

Parallelism

Sub-Queries

Time

nrPartitionLevels

nrPartitionLevels

nrPartitionLevels

nrPartitionLevels

nrPartitionLevels

nrPartitionLevels

nrPartitionLevels
### Bushy Query Plans

#### Space

**Bushy Plan ~**

#### Parallelism

*2

#### Sub-Queries

*(7/8)*

#### Time

*(21/27)*

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T,U</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T,U</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T,V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T,U</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T,V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T,V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T,V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>nrPartitionLevels</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>nrPartitionLevels</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Time ?!?!</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Parallelism</th>
<th>Sub-Queries</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>2</em></td>
<td><em>(7/8)</em></td>
<td><em>(21/27)</em></td>
</tr>
</tbody>
</table>

**Partitioning via Ternary Constraints**

**Plan Space**

**Binary Tree**

** blogger Plan ~**
Complexity for Bushy Plans
Complexity for Bushy Plans

Nr. Sub-Queries $\sim 2^{\text{nrTables}}$

Nr. Splits $\sim 3^{\text{nrTables}}$
Complexity for Bushy Plans

Nr. Sub-Queries $\sim 2^{nrTables}$

Nr. Splits $\sim 3^{nrTables}$ Dominant
Calculating Bushy Speedup

Path R→Root: T after U
Calculating Bushy Speedup

Path $R \rightarrow \text{Root: } T$ after $U$
Calculating Bushy Speedup

Path $R \rightarrow \text{Root}$; $T$ after $U$

No $T$ without $U$!
Calculating Bushy Speedup

No T without U!

Path R→Root, T after U
Calculating Bushy Speedup

Path \( R \rightarrow \text{Root} \rightarrow T \) after \( U \)
Calculating Bushy Speedup

Path $R \rightarrow \text{Root}$ $T$ after $U$

No $T$ without $U$!
Calculating Bushy Speedup

27 Cases
6 Excluded
Splits * 21/27

Path $R \rightarrow \text{Root} \rightarrow T$ after $U$

No $T$ without $U$!
## Complexity Results

Results for one cost metric:

<table>
<thead>
<tr>
<th>Metric \ Plan Space</th>
<th>Left-Deep</th>
<th>Bushy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes Sent</td>
<td>$nrWorkers \times (querySize + planSize)$</td>
<td></td>
</tr>
<tr>
<td>Time - Master</td>
<td>$nrWorkers^2$</td>
<td></td>
</tr>
<tr>
<td>Memory - Worker</td>
<td>$2querySize \times (3/4)nrWorkers$</td>
<td>$2querySize \times (7/8)nrWorkers$</td>
</tr>
<tr>
<td>Time - Worker</td>
<td>$2querySize \times (3/4)nrWorkers \times querySize$</td>
<td>$3querySize \times (21/27)nrWorkers$</td>
</tr>
</tbody>
</table>

(Considering multiple metrics multiplies by polynomial factor)
Complexity Results

Results for one cost metric:

<table>
<thead>
<tr>
<th>Metric \ Plan Space</th>
<th>Left-Deep</th>
<th>Bushy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes Sent</td>
<td>( nr\text{Workers} \times (querySize + planSize) )</td>
<td></td>
</tr>
<tr>
<td>Time - Master</td>
<td>( nr\text{Workers}^2 )</td>
<td></td>
</tr>
<tr>
<td>Memory - Worker</td>
<td>( 2\times querySize \times \frac{3}{4} \times nr\text{Workers} )</td>
<td>( 2\times querySize \times \frac{7}{8} \times nr\text{Workers} )</td>
</tr>
<tr>
<td>Time - Worker</td>
<td>( 2\times querySize \times \frac{3}{4} \times nr\text{Workers} \times querySize )</td>
<td>( 3\times querySize \times \frac{21}{27} \times nr\text{Workers} )</td>
</tr>
</tbody>
</table>

(Considering multiple metrics multiplies by polynomial factor)
Complexity Results

Results for one cost metric:

<table>
<thead>
<tr>
<th>Metric \ Plan Space</th>
<th>Left-Deep</th>
<th>Bushy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes Sent</td>
<td>Polynomial</td>
<td>nrWorkers * (querySize + planSize)</td>
</tr>
<tr>
<td>Time - Master</td>
<td>Polynomial</td>
<td>nrWorkers^2</td>
</tr>
<tr>
<td>Memory - Worker</td>
<td>Exponential</td>
<td>2querySize * (3/4) * nrWorkers</td>
</tr>
<tr>
<td>Time - Worker</td>
<td>Exponential</td>
<td>2querySize * (7/8) * nrWorkers</td>
</tr>
</tbody>
</table>

(Considering multiple metrics multiplies by polynomial factor)
## Complexity Results

Results for one cost metric:

<table>
<thead>
<tr>
<th>Metric \ Plan Space</th>
<th>Left-Deep</th>
<th>Bushy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomials</td>
<td>2<em>querySize</em>(3/4)*nrWorkers</td>
<td>nrWorkers^2</td>
</tr>
<tr>
<td>Polynomials</td>
<td>2<em>querySize</em>(3/4)*nrWorkers</td>
<td>2<em>querySize</em>(7/8)*nrWorkers</td>
</tr>
<tr>
<td>Exponentials</td>
<td>2<em>querySize</em>(3/4)<em>nrWorkers</em>querySize</td>
<td>3<em>querySize</em>(21/27)*nrWorkers</td>
</tr>
<tr>
<td>Exponentials</td>
<td>2<em>querySize</em>(3/4)*nrWorkers</td>
<td>2<em>querySize</em>(7/8)*nrWorkers</td>
</tr>
</tbody>
</table>

(nrWorkers * (querySize + planSize))

(Considering multiple metrics multiplies by polynomial factor)
Experimental Results

Experimental Setup
Randomly generated queries; one execution cost metric; approximated optimization
Spark 1.5 on Yarn 2.7; 100 nodes with 2x Intel Xeon 2.6 GHz, 128 GB memory

Scalability of new parallel query optimization algorithm
Experimental Results

**Experimental Setup**
Randomly generated queries; two execution cost metrics; approximated optimization
Spark 1.5 on Yarn 2.7; 100 nodes with 2x Intel Xeon 2.6 GHz, 128 GB memory

Scalability of new parallel query optimization algorithm
Dissertation Overview

Pre-Computation

Before

Query Template

Optimizer

SLOW

Potential Optimal Plans

SIGMOD Research Highlight

VLDB 15

VLDBJ 16

Best of VLDB

Query

Optimizer

Near-Optimal Plans

VLDB 15

VLDBJ 16

Near-Optimal Plans

Optimizer

Parallel

VLDBJ 16

Best Effort Plans

VLDB 16

Near-Optimal Plans

Query

FASTER

Optimizer

VLDB 15

VLDBJ 16

Acceptable Plans ...
Near-Optimal Plans ...

Optimality Guarantees

Quantum Optimization Platform

Before

Run Time

Exhaustive

Approximate

Incremental

Random

Fast
## Quantum Computing

<table>
<thead>
<tr>
<th>Classical Computer</th>
<th>Quantum Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Qubits</td>
</tr>
<tr>
<td>1 or 0</td>
<td>1 and 0</td>
</tr>
</tbody>
</table>
D-Wave Quantum Annealer
D-Wave Quantum Annealer

Qubit

Coupling
D-Wave Quantum Annealer

Temperature: 15 mK

Qubit

Coupling
As far as I’m concerned, this completely nails down the case for computationally-relevant collective quantum tunneling in the D-Wave machine, at least within the 8-qubit clusters. 

Experiments on the D-Wave devices over the past years have indicated that the devices use quantum effects.

The general consensus now is that the D-Wave computers do exhibit some quantum behavior.

The sweet spot for quantum annealers is thus quickly finding approximate solutions.

D-Wave has performed a careful study showing that their latest device is 15x faster than the best optimized classical codes on a single core of an Intel CPU (and thus still comparable to a high-end multi-core CPU).

10,000,000 times faster than simulated annealing running on a single core.

Another challenge is the embedding of problems into the native hardware architecture.

It remains uncertain whether D-Wave’s approach will lead to speedups over the best known classical algorithms.

If you wanted to solve a practical optimization problem, you’d first need to encode it into the Chimera graph, and that reduction entails a loss.

More efficient classical optimization algorithms exist for these problems, which outperform the current D-Wave 2X device for most problem instances.
Comments on D-Wave

Experiments on the D-Wave devices over the past years have indicated that the devices use quantum effects. 12.2015

The general consensus now is that the D-Wave computers do exhibit some quantum behavior. 5.2014

The sweet spot for quantum annealers is thus quickly finding approximate solutions. 12.2015

D-Wave has performed a careful study showing that their latest device is 15x faster than the best optimized classical codes on a single core of an Intel CPU (and thus still comparable to a high-end multi-core CPU) 12.2015

As far as I’m concerned, this completely nails down the case for computationally-relevant collective quantum tunneling in the D-Wave machine, at least within the 8-qubit clusters 12.2015

10,000,000 times faster than simulated annealing running on a single core. 12.2015

it remains uncertain whether D-Wave’s approach will lead to speedups over the best known classical algorithms. 12.2015

Another challenge is the embedding of problems into the native hardware architecture 12.2015

more efficient classical optimization algorithms exist for these problems, which outperform the current D-Wave 2X device for most problem instances. 12.2015

if you wanted to solve a practical optimization problem, you’d first need to encode it into the Chimera graph, and that reduction entails a loss 12.2015
Comments on D-Wave

As far as I’m concerned, this completely nails down the case for computationally-relevant collective quantum effects in the D-Wave machine, at least within the 8-qubit clusters.

Experiments on the D-Wave devices over the past years have indicated that the devices use quantum effects.

The general consensus now is that the D-Wave computers do exhibit some quantum behavior.

The sweet spot for quantum annealers is thus quickly finding approximate solutions.

D-Wave has performed a careful study showing that their latest device is 15x faster than the best optimized classical codes on a single core of an Intel CPU (and thus still comparable to a high-end multi-core CPU).

if you wanted to solve a practical optimization problem, you’d first need to encode it into the Chimera graph, and that reduction entails a loss.

it remains uncertain whether D-Wave’s approach will lead to speedups over the best known classical algorithms.

Another challenge is the embedding of problems into the native hardware architecture.

more efficient classical optimization algorithms exist for these problems, which outperform the current D-Wave 2X device for most problem instances.

10,000,000 times faster than simulated annealing running on a single core.

Quantum Effects

John Martinis
Google/UCSB

Scott Aaronson
MIT

Matthias Troyer
ETH
Comments on D-Wave

The general consensus now is that the D-Wave computers do exhibit some quantum behavior.

As far as I’m concerned, this completely nails down the case for computationally-relevant collective quantum tunneling in the D-Wave machine, at least within the 8-qubit clusters.

The sweet spot for quantum annealers is thus quickly finding approximate solutions.

Another challenge is the embedding of problems into the native hardware architecture.

If you wanted to solve a practical optimization problem, you’d first need to encode it into the Chimera graph, and that reduction entails a loss.

D-Wave has performed a careful study showing that their latest device is 15x faster than the best optimized classical code on a single core of an Intel CPU (and thus still comparable to a high-end multi-core CPU).

The general consensus now is that the D-Wave computers do exhibit some quantum behavior.

10,000,000 times faster than simulated annealing running on a single core.

Quantum Effects Often Beaten at Finding Optimum Faster than Single-Core at Approximation
Comments on D-Wave

As far as I'm concerned, this completely nails down the case for computationally-relevant collective quantum tunneling in the D-Wave machine, at least within the 8-qubit clusters.

The general consensus now is that the D-Wave computers do exhibit some quantum behavior.

The sweet spot for quantum annealers is thus quickly finding approximate solutions.

D-Wave has performed a careful study showing that their latest device is 15x faster than the best optimized classical codes on a single core of an Intel CPU (and thus still comparable to a high-end multi-core CPU).

Another challenge is the embedding of problems into the native hardware architecture.

10,000,000 times faster than simulated annealing running on a single core.

Quantum Effects Often Beaten at Finding Optimum

it remains uncertain whether D-Wave's approach will lead to improvements over the best known classical algorithms.

if you wanted to solve a practical optimization problem, you’d first need to encode it into the Chimera graph, and that reduction entails a loss.

10,000,000 times faster than simulated annealing running on a single core.

How About Relevant Problems??

more efficient classical optimization algorithms exist for these problems, which outperform the current D-Wave 2X device for most problem instances.

12.2015

5.2014
Mapping Approach

Find Optimal Plan Combination
Mapping Approach

Find Optimal Plan Combination

Transform → Quadratic Formula

Binary Variables:
- selectPlan1
- selectPlan2
- ...
Mapping Approach

Find Optimal Plan Combination

1 Transform → Quadratic Formula

Binary Variables:
- selectPlan1
- selectPlan2
- ... 

Transform → Weights on Qubits

Cost Formula:
- costTerms
- savingsTerms
- constraintTerms

selectPlan1

selectPlan2
Experimental Results

Experimental Setup
Compared D-Wave 2X vs. Integer Programming, Genetic Algorithms, Heuristics
Compared approximation time on randomly generated problem instances

Experimental Results

Quantum Speedup

Qubits / Variable

537 Queries, 2 Plans per Query
253 Queries, 3 Plans per Query
140 Queries, 4 Plans per Query
108 Queries, 5 Plans per Query
Multiple Query Optimization on the D-Wave 2X Adiabatic Quantum Computer

Immanuel Trummer and Christoph Koch
{firstname}.{lastname}@epfl.ch
École Polytechnique Fédérale de Lausanne

ABSTRACT

The D-Wave adiabatic quantum annealer solves hard combinatorial optimization problems leveraging quantum physics. The newest version features over 1000 qubits and was released in August 2015. We were given access to such a machine, currently hosted at NASA Ames Research Center in California, to explore the potential for hard optimization problems that arise in the context of databases.

In this paper, we tackle the problem of multiple query optimization (MQO). We show how an MQO problem instance can be transformed into a mathematical formula that in California. This device is claimed to exploit the laws of quantum physics [6] in the hope to solve NP-hard optimization problems faster than traditional approaches. The machine supports a very restrictive class of optimization problems while it is for instance not capable of running Shor’s algorithm [40] for factoring large numbers\(^1\). We will show how instances of the multiple query optimization problem can be brought into a representation that is suitable as input to the quantum annealer. We also report results of an experimental evaluation that compares the time it takes to solve MQO problems on the quantum annealer to the time taken by algorithms that run on traditional computers.
Example

Optimal Tradeoffs

Types

Properties

Entities

Opinions

Models

Query Optimizer

Optimization Time:

Higher Resolution

Faster Optimization

Seconds

Optimization Time:

Resources

Time
Mining Subjective Associations
Mining Subjective Associations
Jogging is so boring...

I find kittens cute

Palo Alto is a small city

I don’t find jogging boring

1

Extract Opinion Statements

The Web

Kittens-Cute

Palo Alto-Big

Jogging-Boring
Jogging is so boring…

Mining Subjective Associations

1. Extract Opinion Statements
   - **Cute Animals**
     - Majority Opinion: Pr(…)=0.9
     - User Opinion: Pr(…)=0.8
     - User Post: I find kittens cute
   - **Big Cities**
     - Majority Opinion: Pr(…)=0.85
     - User Opinion: Pr(…)=0.98
     - User Post: Palo Alto is not big
   - **Boring Sports**
     - Majority Opinion: Pr(…)=0.8
     - User Opinion: Pr(…)=0.95
     - User Post: I don’t find jogging boring

2. Learn User Behavior
   - **Kittens-Cute**
     - User Post: kittens are cute
   - **Palo Alto-Big**
     - User Post: Palo Alto is a small city
   - **Jogging-Boring**
     - User Post: Jogging is so boring…
Mining Subjective Associations

1. Extract Opinion Statements
   - Jogging is so boring...
   - I find kittens cute
   - Palo Alto is a small city
   - Palo Alto is not big
   - Jogging is so boring...

2. Learn User Behavior
   - I find kittens cute
   - Palo Alto is not big
   - I don't find jogging boring

3. Infer Majority Opinion
   - Majority Opinion: Pr(...)=0.9
   - User Opinion: Pr(...)=0.8
   - Majority Opinion: Pr(...)=0.85
   - User Opinion: Pr(...)=0.98
   - Majority Opinion: Pr(...)=0.8
   - User Opinion: Pr(...)=0.95
Statistics and Results

> 4 billion associations

2 hours

5000 nodes

~ 50 terabytes

Text, Entities

Mining
Statistics and Results

- Associations
- Mining
- Text, Entities

Precision

Prior: 54%
This: 77%

Recall

Prior: 48%
This: 97%

> 4 billion
2 hours
5000 nodes
~ 50 terabytes
More Details

SIGMOD '15

Mining Subjective Properties on the Web

Immanuel Trummer*
EPFL
Lausanne, Switzerland
immanuel.trummer@epfl.ch

Alon Halevy
Google, Inc.
Mountain View, USA
halevy@google.com

Hongrae Lee
Google, Inc.
Mountain View, USA
hrlee@google.com

Sunita Sarawagi
Google, Inc. and IIT Bombay
Mountain View, USA/Bombay, India
sarawagi@google.com

Rahul Gupta
Google, Inc.
Mountain View, USA
grahul@google.com

ABSTRACT

Even with the recent developments in Web search of answering queries from structured data, search engines are still limited to queries with an objective answer, such as EUROPEAN CAPITALS or WOODY ALLEN MOVIES. However, many queries are subjective, such as SAFE CITIES, or CUTE ANIMALS. The underlying knowledge bases of search engines do not contain answers to these queries because they do not have a ground truth. We describe the SURVEYOR system that mines the dominant opinion held by authors of Web content about whether a subjective property applies to a given entity. The evidence on which SURVEYOR relies are statements extracted from Web text that either support the property or claim its absence.

Categories and Subject Descriptors
H.4 [Information Systems Applications]: Miscellaneous

Keywords
Text mining; subjective properties; user behavior model

1. INTRODUCTION

In recent years, Web search engines have invested heavily in answering queries with structured data. For example, queries such as WOODY ALLEN MOVIES or AMERICAN PRESIDENTS will yield a display of the appropriate entities. These queries are enabled by large knowledge bases about important held, such as Wikipedia.
Summary

Pre-Computation

Before VLDB'15

- VLDB'16

- Potential Optimal Plans

- Best of VLDB

Optimality Guarantees

- Optimal Plans

- Near-Optimal Plans

- Acceptable Plans

- Best Effort Plans

- Exhaustive

- Approximate

- Incremental

- Random

- SLOW

- FASTER

- Quantum Optimization Platform

- Parallel

- Query

- Template

- Solver

- Optimizer

- Query

- Template

- Solver

- Optimizer

- Pre-Computation

- Run Time

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER

- Query

- Template

- Solver

- Optimizer

- FASTER