Some Clique Enumerations in Database Management

Benny Kimelfeld
Technion Data & Knowledge Lab
tdk.net.technion.ac.il
Enumerating Graph Cliques

- Many apps of (max) clique enumeration
  - Genome-data analysis [Harley+ 01]
  - Protein-data analysis [Mohseni-Zadeh+ 04]
  - Frequent pattern mining [Koch 01]
  - Sensor-network management [Biswas+ 13]
  - Financial analysis [Boginski+ 05]
  - Social network analysis [Wasserman, Faust 94] [Palla+ 05] [Yan, Gregory 09]

- Long continuum of research on algorithms
  - [Bron, Kerbosch 73] [Johnson+ 88] [Makino, Uno 04] [Tomita+ 06] [Conte+ 16/17] …

- MCs enumerable w/ poly delay, linear space
Sometimes *almost* Cliques

- Maximal cliques often overly restrictive
  - Not all pairs are friends, missing links, ...

- Relaxations proposed; e.g., \( k \)-plex [Seidman,Foster 78]
  - Def: clique, but each \( v \) may miss \( k \) edges
  - Studied in social-network analysis
    [Pattillo 11,13] [Balasundaram+ 11]
  - Poly delay for every fixed \( k \) [Berlowitz,Cohen,K 15]
  - Incremental FPT & “Intractable” if \( k \) is input; reduce from *hypergraph-transversals* (long-standing open)
    [Eiter,Gottlob 95,03,13] [Khachiyan+ 06]
  - Development in scalable algorithms [ Conte+ 17,18]
This dense under-layer of prior trusted relationships made the hijacker network both stealth and resilient. Although we don’t know all of the internal ties of the hijackers’ network it appears that many of the ties were concentrated around the pilots. This is a risky move for a covert network. Concentrating both unique skills and connectivity in the same nodes makes the network easier to disrupt – once it is discovered. Peter Klerks (Klerks 2001) makes an excellent argument for targeting those nodes in the network that have unique skills. By removing those necessary skills from the project, we can inflict maximum damage to the project mission and goals. It is possible that those with unique skills would also have unique ties within the network. Because of their unique human capital and their high social capital the pilots were the richest targets for removal from the network. Unfortunately they were not discovered in time.

Mohamad Atta participates in 36/51 2-plexes of size > 3 (more than anyone else)

Mohamad Atta participates in all 18 3-plexes of size > 5 (2nd best: 14)

Mohamad Atta in 343/621 3-plexes (2nd best: 199)

Krebs, V.: Mapping networks of terrorist cells
Connections 24, 45–52 (2002)
This Talk

2 apps of clique enumeration & counting in database management:

- Reasoning about inconsistency
- Query planning
Outline

- Introduction
  - Cliques in Inconsistent Databases
  - Cliques in Query Planning
Inconsistency in the DBpedia KB

Marion Jones
- dbo:height
  - 1.524
  - 1.778

Cullen Douglas
- dbo:birthPlace
  - dbr:California
  - dbr:Florida

Irene Tedrow
- dbo:deathPlace
  - dbr:California
  - dbr:Hollywood,_Los_Angeles
  - dbr:New_York_City

David Saxe
- dbo:birthYear
  - 1969

Melinda Saxe
- dbo:birthYear
  - 1965

Melinda Saxe
- dbo:parent
  - David Saxe

David Saxe
- dbo:parent
  - Melinda Saxe
Sources of Inconsistent Data

- Imprecise **data sources**
  - Crowd, Web pages, social encyclopedias, sensors, ...

- Imprecise **data generation**
  - ETL, natural-language processing, sensor/signal processing, image recognition, ...

- **Conflicts in data integration**
  - Crowd + enterprise data + KB + Web + ...

- **Data staleness**
  - Entities change address, status, ...

- **And so on ...**
Principled Declarative Approaches

• Several principled approaches proposed for reasoning about inconsistent data

• Concepts in declarative approaches
  – Integrity constraints
    ▪ Or dependencies
  – Inconsistent database
    ▪ Violates the constraints
  – Edit operations
    ▪ Delete/insert tuple, update an attribute
  – Repairs
    ▪ Consistent DB following a legitimate edit

• Theoretical formulation [Arenas, Bertossi, Chomicki 99]
Examples of Integrity Constraints

- Key constraints
  - \texttt{Person(ssn,name,birthCity,birthState)}

- Functional Dependencies (FDs)
  - \texttt{birthCity} $\rightarrow$ \texttt{birthState}

- Conditional FDs
  - \texttt{birthCity} $\rightarrow$ \texttt{birthState} \texttt{whenever} \texttt{country} = “USA”

- Denial constraints
  - not\texttt{[ Parent(x,y) & Parent(y,x) ]}

- Referential (foreign-key) constraints
  - \texttt{Parent(x,y) \rightarrow Person(x) \& Person(y)}
Example: Inconsistent Database

person → birthCity
birthCity → birthState

<table>
<thead>
<tr>
<th>person</th>
<th>birthCity</th>
<th>birthState</th>
</tr>
</thead>
<tbody>
<tr>
<td>Douglas</td>
<td>LA</td>
<td>CA</td>
</tr>
<tr>
<td>Douglas</td>
<td>Miami</td>
<td>FL</td>
</tr>
<tr>
<td>Tedrow</td>
<td>LA</td>
<td>CA</td>
</tr>
<tr>
<td>Tedrow</td>
<td>LA</td>
<td>NYC</td>
</tr>
<tr>
<td>Jones</td>
<td>LA</td>
<td>CA</td>
</tr>
<tr>
<td>person</td>
<td>birthCity</td>
<td>birthState</td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
<td>------------</td>
</tr>
<tr>
<td>Douglas</td>
<td>Miami</td>
<td>FL</td>
</tr>
<tr>
<td>Tedrow</td>
<td>LA</td>
<td>CA</td>
</tr>
<tr>
<td>Jones</td>
<td>LA</td>
<td>NYC</td>
</tr>
</tbody>
</table>

Set-minimal (for deletion)

person → birthCity

birthCity → birthState

Cardinality-minimal (for deletion)

<table>
<thead>
<tr>
<th>person</th>
<th>birthCity</th>
<th>birthState</th>
</tr>
</thead>
<tbody>
<tr>
<td>Douglas</td>
<td>LA</td>
<td>CA</td>
</tr>
<tr>
<td>Douglas</td>
<td>Miami</td>
<td>FL</td>
</tr>
<tr>
<td>Tedrow</td>
<td>LA</td>
<td>CA</td>
</tr>
<tr>
<td>Tedrow</td>
<td>LA</td>
<td>NYC</td>
</tr>
<tr>
<td>Jones</td>
<td>LA</td>
<td>CA</td>
</tr>
</tbody>
</table>

Set-minimal (for attribute update)

<table>
<thead>
<tr>
<th>person</th>
<th>birthCity</th>
<th>birthState</th>
</tr>
</thead>
<tbody>
<tr>
<td>Douglas</td>
<td>Miami</td>
<td>CA</td>
</tr>
<tr>
<td>Douglas</td>
<td>Miami</td>
<td>CA</td>
</tr>
<tr>
<td>Tedrow</td>
<td>LA</td>
<td>NYC</td>
</tr>
<tr>
<td>Tedrow</td>
<td>LA</td>
<td>NYC</td>
</tr>
<tr>
<td>Jones</td>
<td>LA</td>
<td>NYC</td>
</tr>
</tbody>
</table>

Cardinality-minimal (for attribute update)

<table>
<thead>
<tr>
<th>person</th>
<th>birthCity</th>
<th>birthState</th>
</tr>
</thead>
<tbody>
<tr>
<td>Douglas</td>
<td>Miami</td>
<td>FL</td>
</tr>
<tr>
<td>Douglas</td>
<td>Miami</td>
<td>FL</td>
</tr>
<tr>
<td>Tedrow</td>
<td>LA</td>
<td>CA</td>
</tr>
<tr>
<td>Tedrow</td>
<td>LA</td>
<td>CA</td>
</tr>
<tr>
<td>Jones</td>
<td>LA</td>
<td>CA</td>
</tr>
</tbody>
</table>

Update Repairs
Reasoning about Database Inconsistency

- Repairing / Cleaning
  - Compute a (good/best) repair
  - [Bertossi+ 08] [Kolahi,Lakshmanan 09] [Livshits,K,Roy 18]

- Consistent Query Answering (CQA)
  - *Which query answers are not affected by inconsistency?*
  - Formally, find the tuples that belong to $Q(J)$ for all repairs $J$
  - [Arenas+ 99] [Fuxman,Miller 05] [Koutris,Wijsen 17]

- Repair checking
  - Given $I$ and $J$, is $J$ a repair of $I$? ; typically a complexity tool
  - [Afrati,Kolaitis 09] [Chomicki,Marcinkowski 05]

- Repair counting (+enumeration)
  - Measure *consistency* of query answers [Maslowski,Wijsen 14]
  - Measure *inconsistency* [Livshits,K 17] ; also studied in the KR community [DeBona,Grant,Hunter,Konieczny AAAI18]
<table>
<thead>
<tr>
<th>Constraints</th>
<th>Key</th>
<th>Functional</th>
<th>Referential</th>
<th>Conditional Functional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations</td>
<td>Tuple delete</td>
<td>Tuple insert</td>
<td>Attribute update</td>
<td></td>
</tr>
<tr>
<td>Repairs</td>
<td>Set minimal</td>
<td>Cardinality minimal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem</td>
<td>Cleaning</td>
<td>Repair Checking</td>
<td>CQA</td>
<td>Repair Counting</td>
</tr>
</tbody>
</table>
**Constraints**
- Key
- Functional
- Referential
- Conditional Functional

**Operations**
- Tuple delete
- Tuple insert
- Attribute update

**Repairs**
- Set minimal
- Cardinality minimal

**Problem**
- Cleaning
- Repair Checking
- CQA
- Repair Counting

**Max Clique**
Key | Functional | Referential | Conditional Functional
---|---|---|---
Tuple delete | Tuple insert | Attribute update
Set minimal | Cardinality minimal
Cleaning | Repair Checking | CQA | Repair Counting
Reasoning about max cliques
<table>
<thead>
<tr>
<th>Constraints</th>
<th>Key</th>
<th>Functional</th>
<th>Referential</th>
<th>Conditional Functional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations</td>
<td>Tuple delete</td>
<td>Tuple insert</td>
<td>Attribute update</td>
<td></td>
</tr>
<tr>
<td>Repairs</td>
<td>Set minimal</td>
<td>Cardinality minimal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem</td>
<td>Cleaning</td>
<td>Repair Checking</td>
<td>CQA</td>
<td>Repair Counting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#max cliques</td>
</tr>
</tbody>
</table>
Counting Set-Minimal Repairs

- Counting the maximal cliques of a graph is \#P-complete [Provan, Ball 83], inapproximable [Håstad 96]
- Special tractable cases, e.g., P₄-free graphs
  - P₄-free graph (a.k.a. cograph): no induced path of length 4
- What about the consistency graphs?

**Theorem [Livshits, K PODS’17]**

Equivalent for every fixed set of FDs:

1. Repairs can be counted in poly time
2. Every consistency graph is P₄-free

Moreover, testable in poly time (given FDs)

* Assuming P ≠ \#P
Hard (to approx)  Poly time

- ssn → city
- city → state
- ssn → name
- ssn country → license#
- faculty → dean
- faculty building → address
- faculty → dean
- faculty professor → room#
- ssn → uID
- uID → email
- email → ssn
Outline

- Introduction
- Cliques in Inconsistent Databases
- Cliques in Query Planning
Join Query

Friends($x_1, y_1$), School($x_1, s$), School($y_1, s$)  
**Friend from the same high school**

Colleagues($x_2, y_2$), Univ($x_2, u$), Univ($y_2, u$)  
**Colleague from the same university**

Married($x_3, y_3$), Parent($x_3, c$), Parent($y_3, c$)  
**Spouse with a common child**

Same $x$:  
- Artist($x$),  
- Friends($x_1, y_1$), School($x_1, s$), School($y_1, s$),  
- Colleagues($x_2, y_2$), Univ($x_2, u$), Univ($y_2, u$),  
- Married($x_3, y_3$), Parent($x_3, c$), Parent($y_3, c$),  
- Same($x, x_1$), Same($x, x_2$), Same($x, x_3$)
Outline

- Introduction
- Cliques in Inconsistent Databases
- Cliques in Query Planning
  - Caching in Trie Join
  - Enumerating Tree Decompositions
  - Ranked Enumeration
Join Algorithms

- Classic algorithms select a join ordering with “easier” intermediate joins [Selinger+ 79]

- Yannakakis [1981] for acyclic queries
  - And cyclic queries with low hypertree width

- New breed of joiners: worst-case optimal
  - [Ngo,Porat,Ré,Rudra 12]
  - Meet the Atserias-Grohe-Marx [2008] bound
    - Example: \( R(X,Y) \bowtie S(X,Z) \bowtie T(Y,Z) \) — \( n^2 \) vs \( n^{1.5} \)
  - In-memory, scan all relations simultaneously
  - NPRR [2012], Leapfrog Trie Join [Veldhuizen 14], Minesweeper [Ngo+ 14], DunceCap [Tu,Ré 15], ...
Join Processing for Graph Patterns: An Old Dog with New Tricks [Nguyen+ 15]
Leapfrog Trie Join (LFTJ) [Veldhuizen 14]

- Variant of variable elimination
- Relations in trie structures
  - Level = attribute / variable
  - Tuple = root-to-leaf path
- Multiple trie pointers aligned using a leap-frog (jump competition) scan; backtracking

No memory used beyond tries

<table>
<thead>
<tr>
<th>Married(x,y)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Married(x,y)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Univ(y,u)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>S</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
 Married(x,y)       | Univ(y,u)           
 A   B             | A   U              
 B   C             | B   U              
 C   D             | C   S              
 C   E             |                   
```
Caching in LFTJ  [Kalinsky,Etsion,K 17]

Artist(x),
Friends(x_1,y_1) , School(x_1,s) , School(y_1,s),
Colleagues(x_2,y_2) , Univ(x_2,u) , Univ(y_2,u),
Married(x_3,y_3) , Parent(x_3,c) , Parent(y_3,c),
Same(x,x_1) , Same(x,x_2) , Same(x,x_3)

Tree decomposition
In [Kalinsky,Etsion,K 17]:
- Caching policies
- TD selection
- Extension to count(jion)
Experimental Evaluation

Join Evaluation over ca-GrQc

- 9x faster than 2nd best

Join Evaluation over Wiki-Vote

- 14x faster than LFTJ

Join Count Aggregation over ca-GrQc

- 5x faster than 2nd best

Join Count Aggregation over Wiki-Vote

- 500,000x faster than LFTJ

Runtime (ms)
TD Selection Matters!

Movie: $m_1$, $m_2$, $m_3$
Person: $p_1$, $p_2$, $p_3$

4000 sec

40 sec

27000 sec

600 sec
**Definition:** Tree Decomposition (TD) of a Graph G

(t, β), t a tree, β a mapping \( \text{nodes}(t) \rightarrow 2^{\text{nodes}(G)} \) where:

1. For all \( e \in \text{edges}(G) \) there is \( u \in \text{nodes}(t) \) s.t. \( e \subseteq \beta(u) \)
2. For all \( v \in \text{nodes}(G) \), the set \( \{ u \in \text{nodes}(t) \mid v \in \beta(u) \} \)
   induces a connected subtree of t
Standard Goodness Measures

**TD width:** \( \max(|\text{bag}|)-1 \)

**TD fill-in:** \#new edges needed to connect bag neighbors

**Definition:** Tree Decomposition (TD) of a Graph \( G \)

\( (t, \beta) \), \( t \) a tree, \( \beta \) a mapping \( \text{nodes}(t) \rightarrow 2^{\text{nodes}(G)} \) where:

1. For all \( e \in \text{edges}(G) \) there is \( u \in \text{nodes}(t) \) s.t. \( e \subseteq \beta(u) \)
2. For all \( v \in \text{nodes}(G) \), the set \( \{u \in \text{nodes}(t) \mid v \in \beta(u)\} \) induces a connected subtree of \( t \)
Running Example: Width/Fill-in

Artist($x$),
Friends($x_1,y_1$), School($x_1,s$), School($y_1,s$),
Colleagues($x_2,y_2$), Univ($x_2,u$), Univ($y_2,u$),
Married($x_3,y_3$), Parent($x_3,c$), Parent($y_3,c$),
Same($x,x_1$), Same($x,x_2$), Same($x,x_3$)

width = 2
fill-in = 0

chordal / triangulated
Goodness Criteria in Cached LFTJ

[Kalinsky,Etsion,K 17]

- **Cardinality of adhesions** (intersections)
  - This is the dimension of our caches
  - Smaller = better

- **Width, #bags** (#caches)
  - Smaller width = better; higher #bags = better

- **Skew**
  - How effective are the caches?
  - Note: Data (not just query) property
  - Known effectiveness estimators for variable orderings
    [Chu,Balazinska,Suciu 15]
How to find a TD with min estimated cost?
NP-hard to minimize width / fill-in
Heuristic recipe:
1. Generate a large pool of "good" TDs
2. Compute the cost of each
3. Choose the one with the best cost

Need an algorithm to enumerate TDs!
Outline

- Introduction
- Cliques in Inconsistent Databases
- Cliques in Query Planning
  - Caching in Trie Join
  - Enumerating Tree Decompositions
  - Ranked Enumeration
Not Just for Database Queries!

• TD apps can benefit from specialized measures
  – Games (computation of Nash equilibria [Gottlob+ 05])
  – Bioinformatics (prediction of RNA structures [Zhao+ 06])
  – Weighted model counting [Li+ 08]
  – Constraint-satisfaction problems [Kolaitis, Vardi 00]
  – Probabilistic graphical models [Lauritzen, Spiegelhalter 88] and knowledge compilation
    ▪ Otero-Mediero & Dechter [2017] select AND-OR trees for BN:
      – TDs → “pseudo trees” → AND/OR trees
      – Score: $F(\text{td-width, pseudo-tree-height})$
      – Used the algorithm presented next
  – ...

• ML applied to learn TD scores (over TD features) from problem instances [Abseher+ 15]
Solutions?

- Generator of [Abseher+ 15] (ML)
  - Generate a handful (10)
  - Best-effort randomness, no guarantees

- Duncecap [Tu,Ré 15]: candidate generator of generalized hypertree decompositions
  - Goal: join optimization
  - No efficiency guarantees, designed for small query graphs

![Graph showing #time (sec) vs #nodes for Random graphs (avg over 100) results.](image)
Task: enumerate all TDs of a graph

• Complexity guarantees
• Effective practical performance
Which TDs to Generate?

Proper tree decomposition: cannot be improved by removing or splitting bags
Task: enumerate all proper TDs of a graph

• Complexity guarantees
• Effective practical solution
**THEOREM** [Carmeli, Kenig, K PODS’17]

Can enumerate in incremental poly. time:

1. All proper TDs
2. All *minimal triangulations*

Wait — *related to this talk?*
**PROPOSITION**: efficient translation between classes of bag-equivalent proper TDs $\iff$ minimal triangulations

Efficient: $\leq n$ max cliques [Gavril 74]; reduce to max spanning trees over max cliques [Jordan 02]; enum max spanning trees [Yamada+ 10]
A bijection [Parra, Scheffler 97]:

min triangles ⇔ max sets of non-crossing min separators

Non-crossing: not separating nodes of the other
(symmetric [Kloks, Kratsch 97])
**Definition: Minimal Separator of a Graph G**

A set $S$ of nodes is a:

- $(u,v)$-separator if $u$ is not reachable from $v$ in $G-S$
- minimal $(u,v)$-sep. if no subset of $S$ is a $(u,v)$-sep.
- A **minimal separator** if it is a min $(u,v)$-sep. for some $(u,v)$
Given a graph $G$:

1. Build the graph $F$: min-seps as nodes; edge = non-cross
2. Enumerate the **max cliques** of $F$ w/ poly. delay

**Problem:** $F$ can be exp. larger than $G$!

**Challenge:** Enumerate the max cliques of $F$ … without generating $F$!
Enumerates the max cliques over a *Succinct Graph Representation* (SGR)

SGR accessed indirectly (via algs), assuming:
1. Nodes can be enumerated with poly. delay
2. Edges can be verified in poly. time
3. Cliques maximized in poly. time

Redesign of our algorithm for hereditary graph properties [Cohen,K,Sagiv 08]
SGR Assumptions in Our Case

1. Nodes can be enumerated with poly. delay
   [Berry+ 99]: Generating all min seps; we show how to make it poly. delay

2. Edges verified in poly. time
   Straightforward (edge = crossing min seps)

3. Cliques maximized in poly. time
   Using [Heggernes 06], via any triangulation algorithm
## Quality on PIC2011 (30 min)

<table>
<thead>
<tr>
<th>alg.</th>
<th>measure</th>
<th>avg #results</th>
<th>avg #≤first</th>
<th>avg min</th>
<th>avg %improv</th>
<th>max %improv</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS-M</td>
<td>width</td>
<td>33635.0</td>
<td>12733.4</td>
<td>20.2</td>
<td>2.6%</td>
<td>26.3%</td>
</tr>
<tr>
<td></td>
<td>fill-in</td>
<td>12724.9</td>
<td>2043.8</td>
<td>14.4%</td>
<td>55.8%</td>
<td></td>
</tr>
<tr>
<td>LB-T</td>
<td>width</td>
<td>11998.3</td>
<td>4744.1</td>
<td>18.5</td>
<td>3.4%</td>
<td>20.7%</td>
</tr>
<tr>
<td></td>
<td>fill-in</td>
<td>1013.6</td>
<td>965.8</td>
<td>2.2%</td>
<td>27.6%</td>
<td></td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Cliques in Inconsistent Databases
- Cliques in Query Planning
  - Caching in Trie Join
  - Enumerating Tree Decompositions
  - Ranked Enumeration
The Case of Poly \#Min-Separators

- General case: inc. poly. time
- If \#min-separators bounded by a polynomial:
  - Min triangls enumerated with poly. delay
  - A min width/fill-in triangulation can be found in poly. time [Bouchitté, Todinca 01]

- Is poly-\#min-seps a realistic assumption?
Hardness Distribution

Terminated in 1 min? 

Terminated 

Not terminated

- Pace2016-100
- promedas
- ObjectDetection
- DBN
- Segmentation
- CSP
- grids
- ProteinFolding
- ImageAlignment
- Pace2016-1000
- ProteinProtein
- DBPedia
- pedigree
- alchemy

Terminated

Not terminated
The Case of Poly $\#\text{Min-Sep}$-Separators

• General case: inc. poly. time
• If $\#\text{min}$-separators bounded by a polynomial:
  – Min triangls enumerated with poly. delay
  – A min width/fill-in triangulation can be found in poly. time [Bouchitté, Todinca 01]

• Is poly-$\#\text{min}$-seps a realistic assumption?
• Can we get ranked enumeration?
THEOREM [Ravid, Medini, K PODS’19]

If \( \#\text{min-separators} < \text{poly}(G) \), then min triangs (and proper TDs) can be enumerated with:

- polynomial delay
- increasing cost

for any “monotonic” cost function (inc. width, fill).

For every fixed \( w \), min triangs (& proper TDs) of width \( < w \) can be enumerated w/ poly. delay and increasing cost.
Monotonic Cost Function
Conclusions

• Clique enumeration is an important, cross-field tool for computing, particularly data analysis
• Lively community, frequent progress (practice & theory)
• Discussed manifestations in DB theory & practice
  – Reasoning about database inconsistency
  – Query planning
• Favorite directions:
  – Highly parallel architectures [Schmidt+ 09] [Jenkins+ 11]
  – Discrimination: Which scoring functions allow for an efficient ranked enumeration of maximal cliques?
Thanks collaborators!

Yoav Etsion  Batya Kenig  Nofar Carmeli  Oren Kalinsky  Ester Livshits

Dori Medini  Noam Ravid