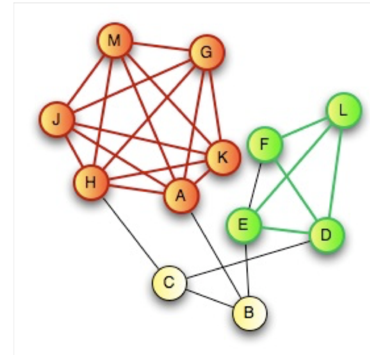
Some Clique Enumerations in Database Management

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

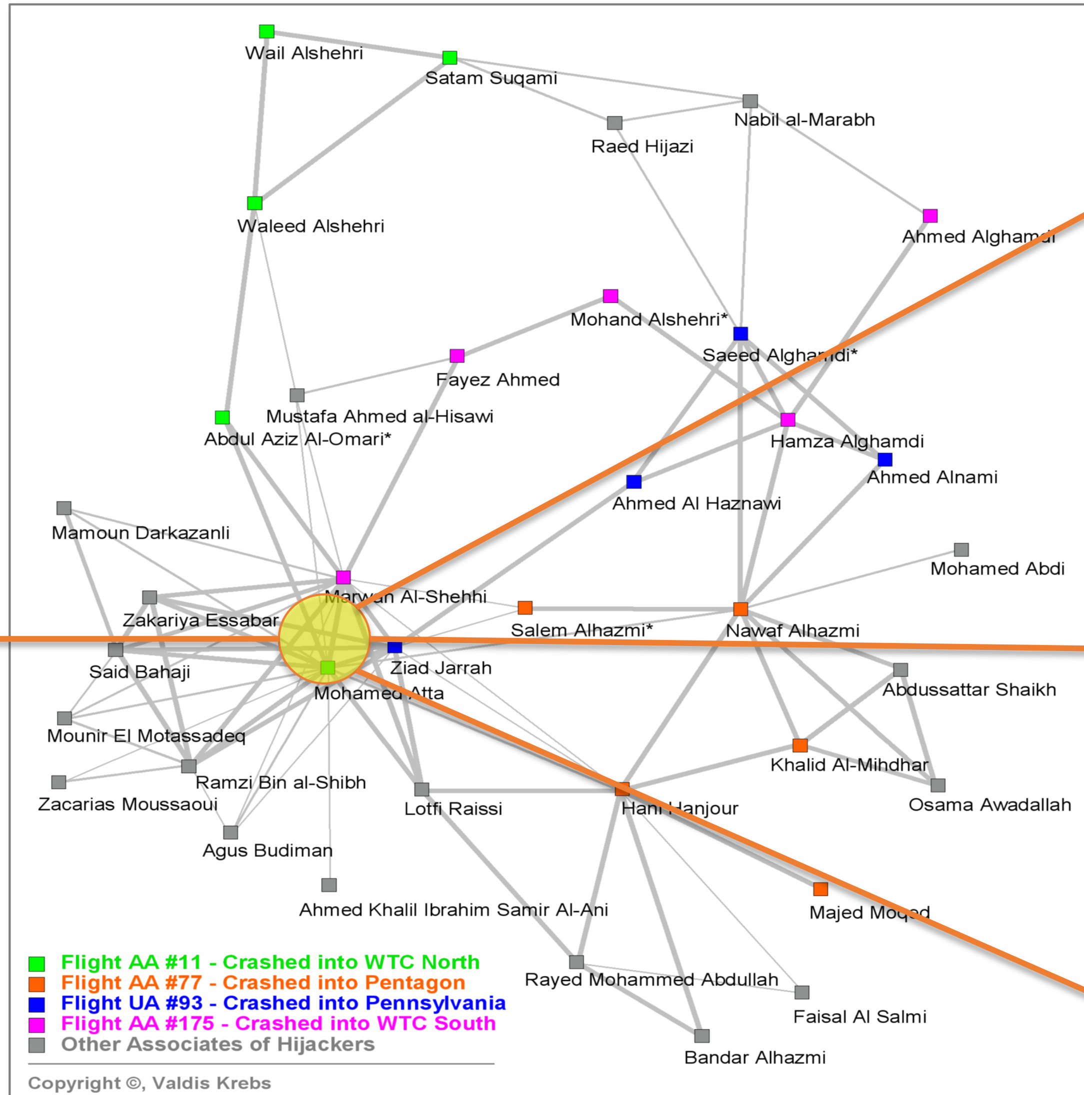
Sometimes *almost* Cliques

- 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]

Illustration on 9/11 Network

k-plex: each v may miss k edges

Mohamad Atta participates in a single clique of size > 3



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 $\xrightarrow{\text{dbo:parent}}$ Melinda Saxe

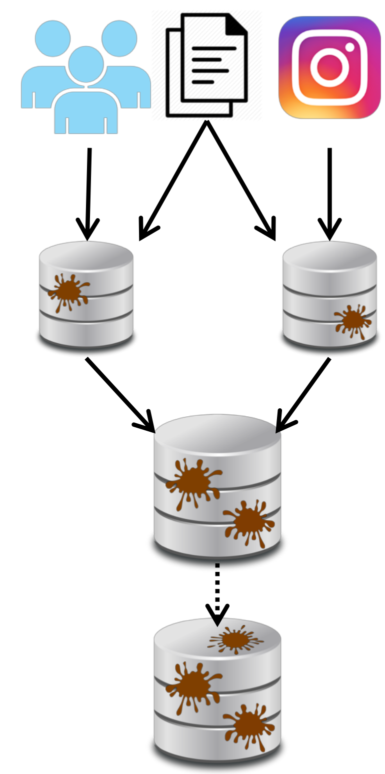
1969 $\xleftarrow{\text{dbo:birthYear}}$ David Saxe

Melinda Saxe $\xrightarrow{\text{dbo:birthYear}}$ 1965

Melinda Saxe $\xrightarrow{\text{dbo:parent}}$ David 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
  - Person(ssn,name,birthCity,birthState)
- Functional Dependencies (FDs)
  - birthCity ⟶ birthState
- Conditional FDs
  - birthCity ⟶ birthState whenever country="USA"
- Denial constraints
  - not[Parent(x,y) & Parent(y,x)]
- Referential (foreign-key) constraints
  - Parent(x,y) ⟶ Person(x) & Person(y)
- …

Example: Inconsistent Database

$$\text{person} \longrightarrow \text{birthCity}$$

$$\text{birthCity} \longrightarrow \text{birthState}$$

person	birthCity	birthState
Douglas	LA	CA
Douglas	Miami	FL
Tedrow	LA	CA
Tedrow	LA	NYC
Jones	LA	CA

Subset Repairs

person	birthCity	birthState
~~Douglas~~	~~LA~~	~~CA~~
Douglas	Miami	FL
~~Tedrow~~	~~LA~~	~~CA~~
Tedrow	LA	NYC
~~Jones~~	~~LA~~	~~CA~~

Set-minimal
(for deletion)

person	birthCity	birthState
~~Douglas~~	~~LA~~	~~CA~~
Douglas	Miami	FL
Tedrow	LA	CA
~~Tedrow~~	~~LA~~	~~NYC~~
Jones	LA	CA

Cardinality-minimal
(for deletion)

$$\mathrm{person} \longrightarrow \mathrm{birthCity}$$

$$\mathrm{birthCity} \longrightarrow \mathrm{birthState}$$

person	birthCity	birthState
Douglas	LA	CA
Douglas	Miami	FL
Tedrow	LA	CA
Tedrow	LA	NYC
Jones	LA	CA

Set-minimal
(for attribute update)

person	birthCity	birthState
Douglas	Miami	CA
Douglas	Miami	CA
Tedrow	LA	NYC
Tedrow	LA	NYC
Jones	LA	NYC

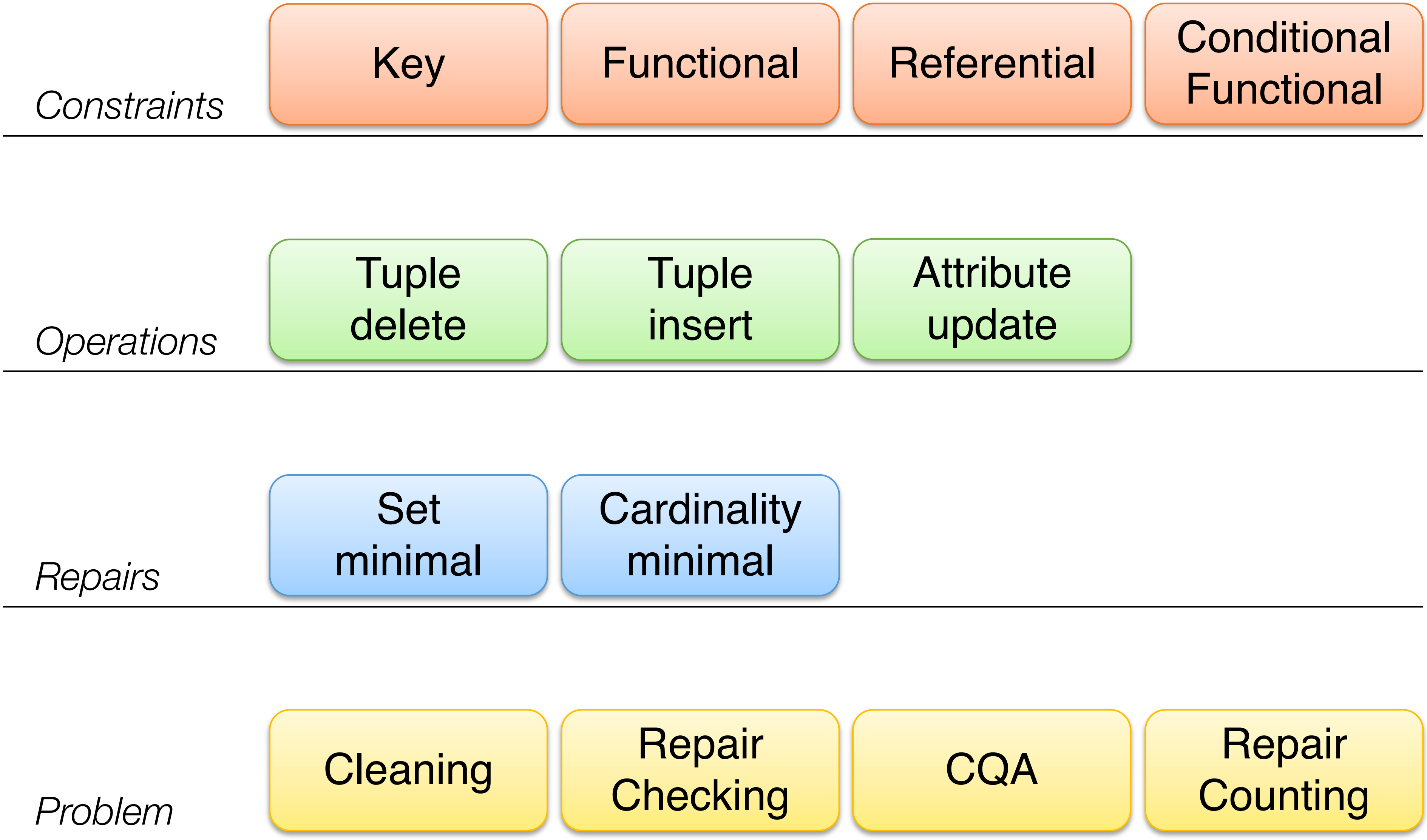
Update Repairs

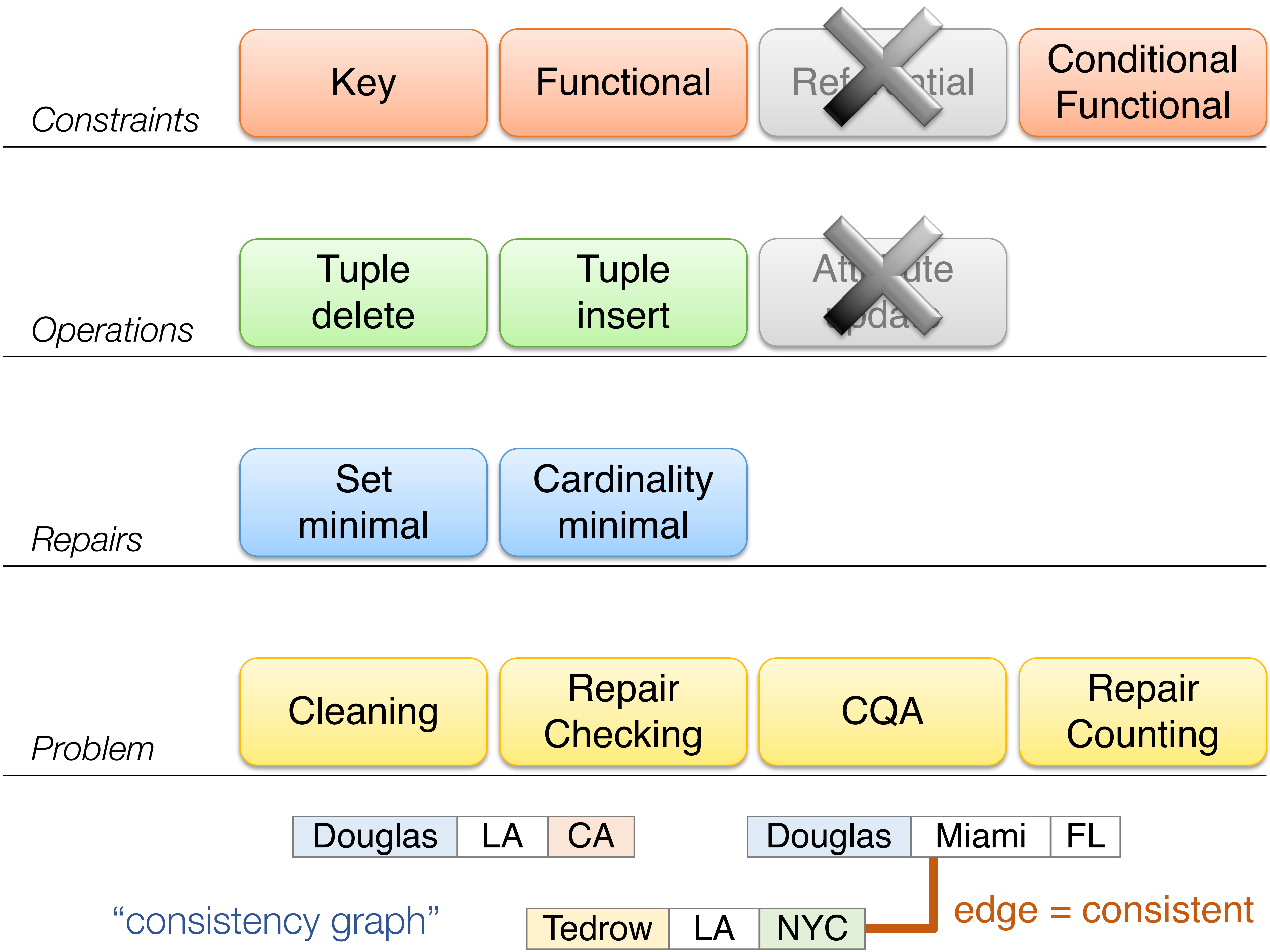
Cardinality-minimal
(for attribute update)

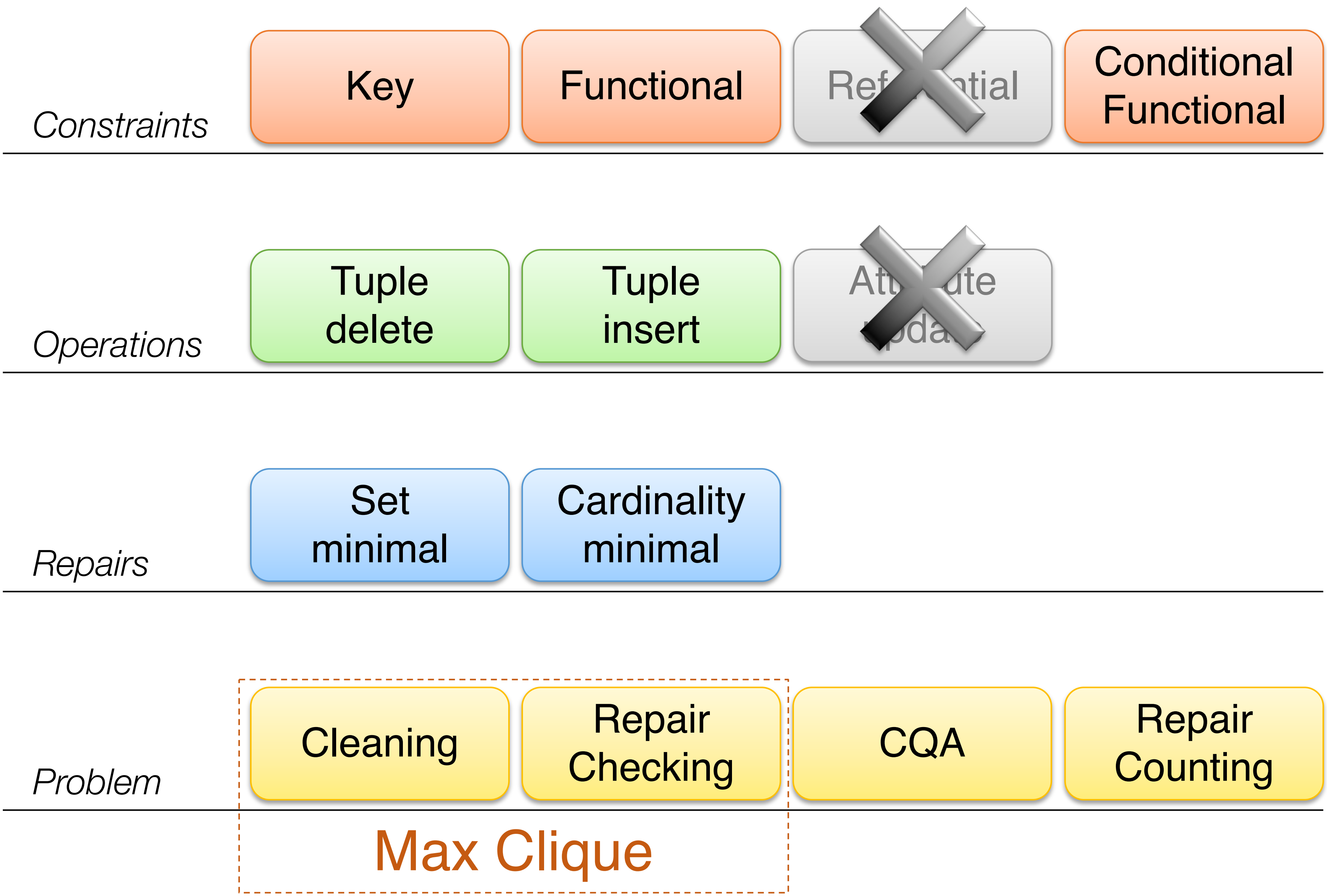
person	birthCity	birthState
Douglas	Miami	FL
Douglas	Miami	FL
Tedrow	LA	CA
Tedrow	LA	CA
Jones	LA	CA

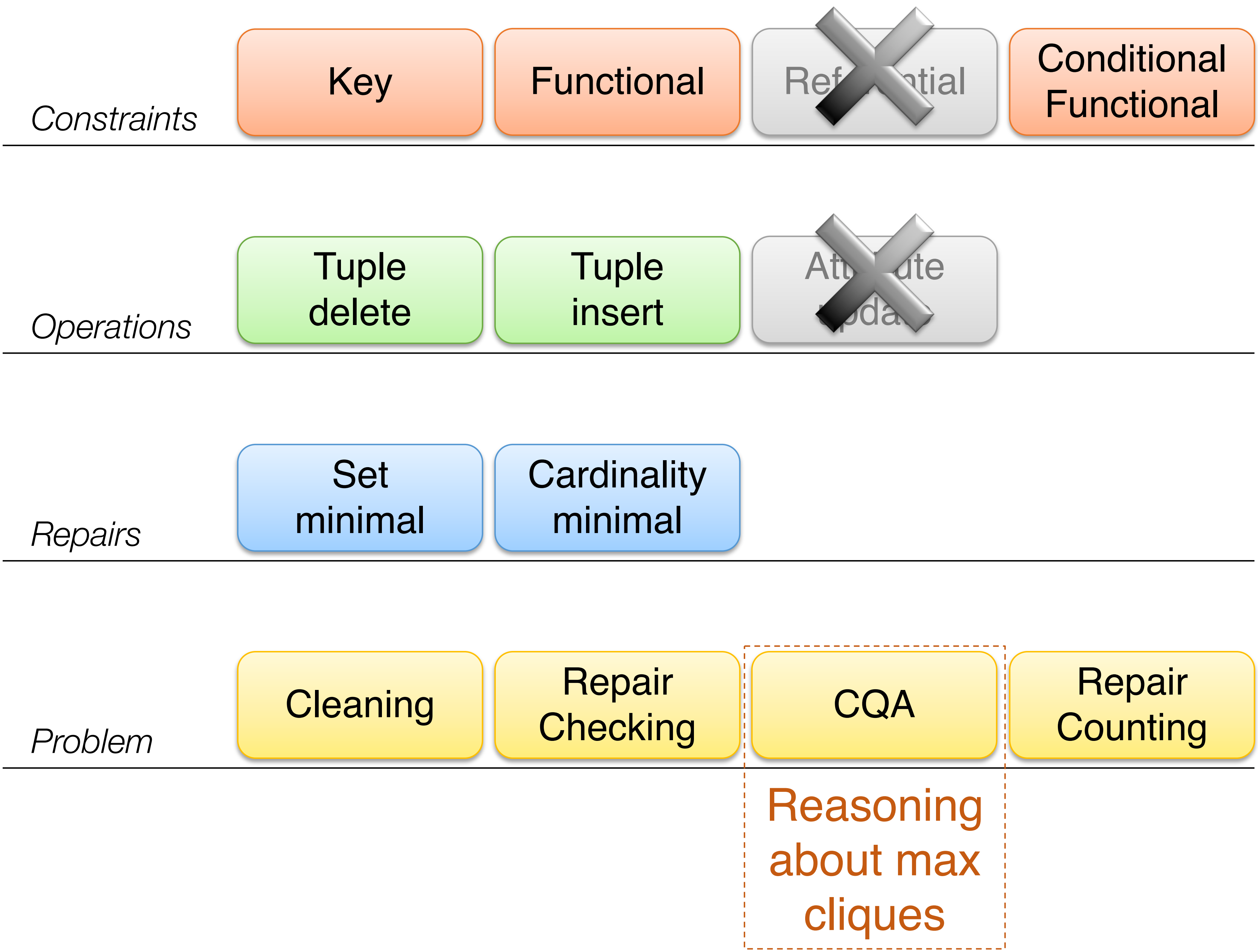
Reasoning about Database Inconsistency

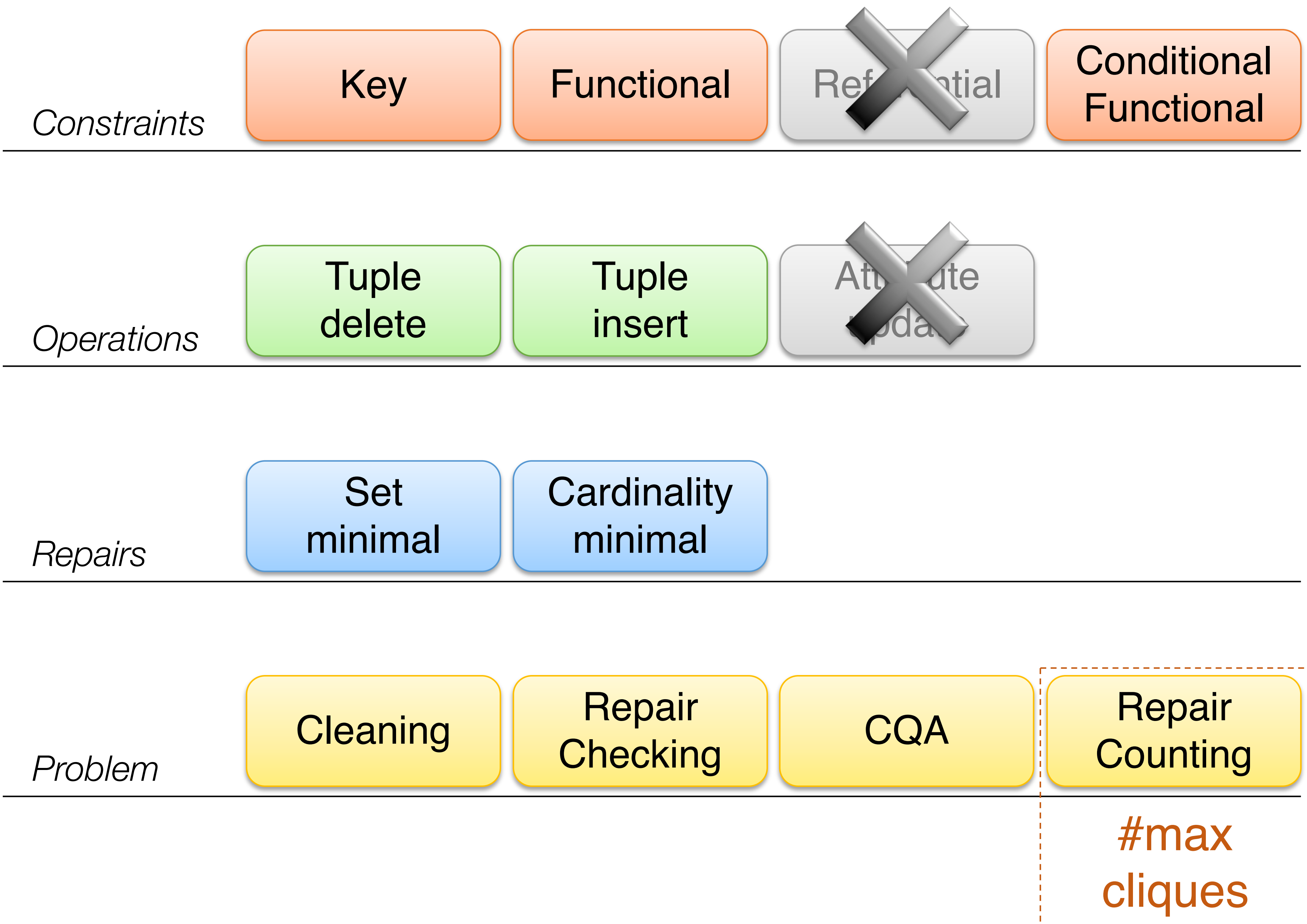
- Repairing / Cleaning
  - Compute a (good/best) repair
  - [Bertossi+ 08] [Kolahi,Lakshmanan 09] [Livshits,K,Roy 18]
- Consistent Query Answering (CQA)
  - *Which query answers are not affected by inconsistency?*
  - Formally, find the tuples that belong to $Q(J)$ for all repairs J
  - [Arenas+ 99] [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]


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


Constraints
Key
Functional
Conditional Functional
Operations
Tuple delete
Tuple insert
Repairs
Set minimal
Cardinality minimal
Problem
Cleaning
Repair Checking
CQA
Repair Counting
Douglas
LA
CA
Douglas
Miami
FL
"consistency graph"
Tedrow
LA
NYC
edge = consistent


Constraints
Key
Functional
Conditional Functional
Operations
Tuple delete
Tuple insert
Repairs
Set minimal
Cardinality minimal
Problem
Cleaning
Repair Checking
CQA
Repair Counting
Max Clique

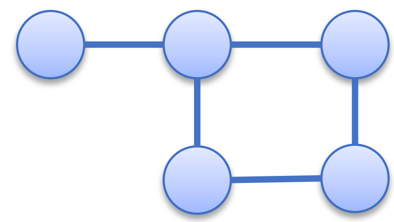

Constraints
Key
Functional
Conditional Functional
Operations
Tuple delete
Tuple insert
Repairs
Set minimal
Cardinality minimal
Problem
Cleaning
Repair Checking
CQA
Repair Counting
Reasoning about max cliques



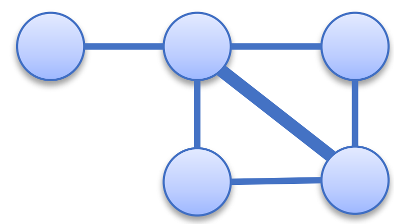
Constraints
Key
Functional
Conditional Functional
Operations
Tuple delete
Tuple insert
Repairs
Set minimal
Cardinality minimal
Problem
Cleaning
Repair Checking
CQA
Repair Counting
#max cliques

Counting Set-Minimal Repairs

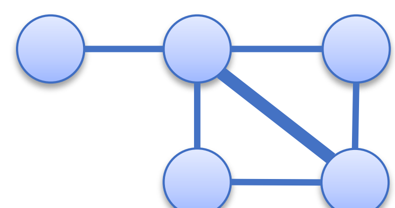
- Counting the maximal cliques of a graph is #P-complete [Provan,Ball 83], inapproximable [Håstad 96]
- Special tractable cases, e.g., P_4-free graphs
  - P_4-free graph (a.k.a. **cograph**): no induced path of length 4
- *What about the consistency graphs?*



Not P_4-free



P_4-free



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 P_4-free

Moreover, testable in poly time (given FDs)

* Assuming $P \neq \#P$

Hard (to approx) | Poly time

ssn $\to$ city
city $\to$ state

ssn $\to$ name
ssn country $\to$ license#

faculty $\to$ dean

faculty $\to$ dean
building $\to$ address

faculty $\to$ dean
faculty professor $\to$ room#

ssn $\to$ uID
uID $\to$ email
email $\to$ ssn

Outline

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

Join Query

Artist(x)

Artist

Friends(x_1,y_1) , School(x_1,s) , School(y_1,s)

Friend from the same high school

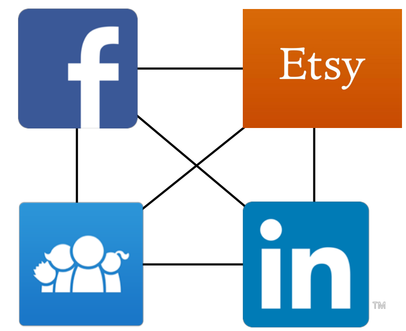
$R(X,Y) \bowtie S(X,Z) \bowtie T(Y,Z)$

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(a,b)

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),

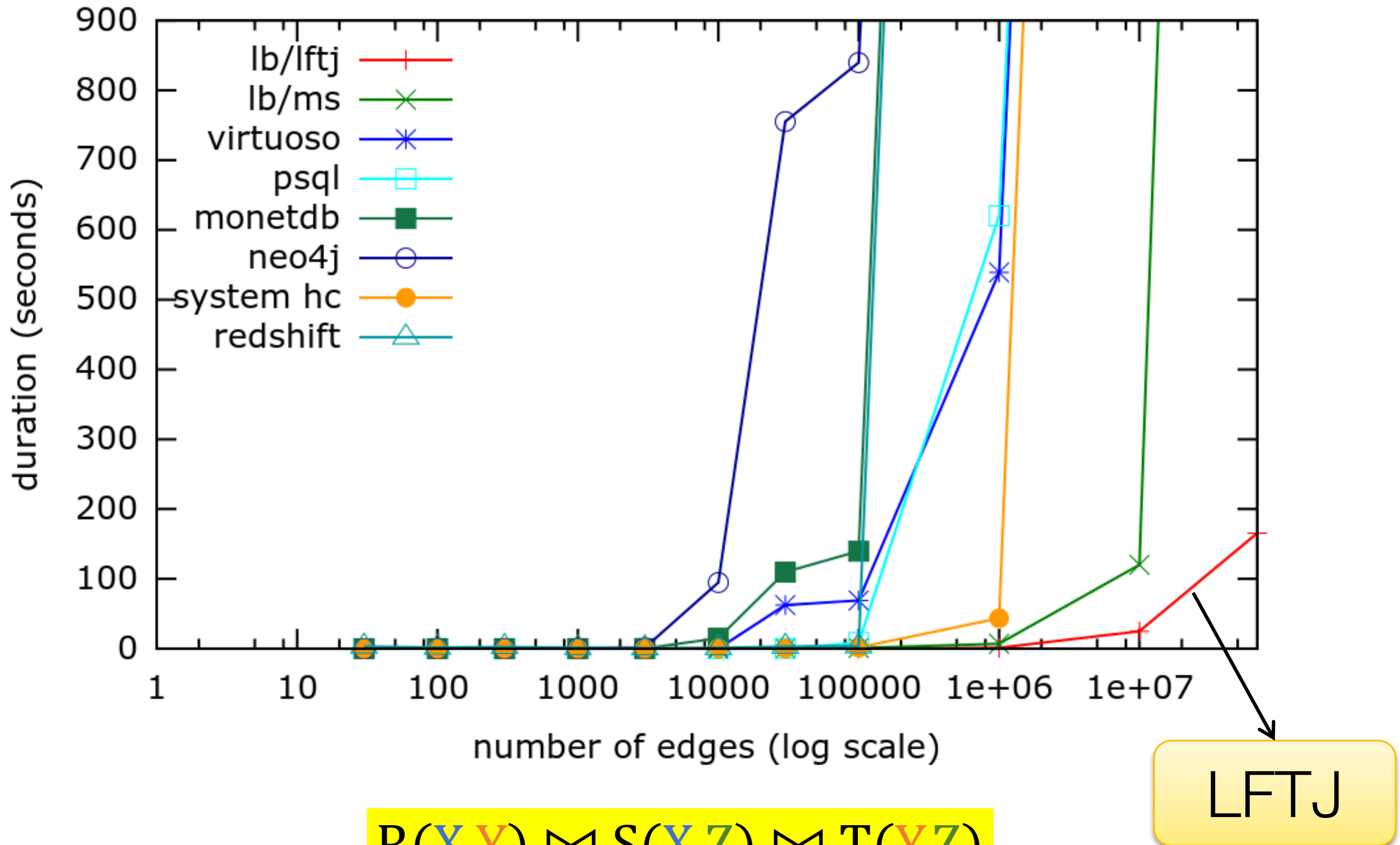
Same(x,x_1) , Same(x,x_2) , Same(x,x_3)

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], ...



$$R(X,Y) \bowtie S(X,Z) \bowtie T(Y,Z)$$

on LiveJournal

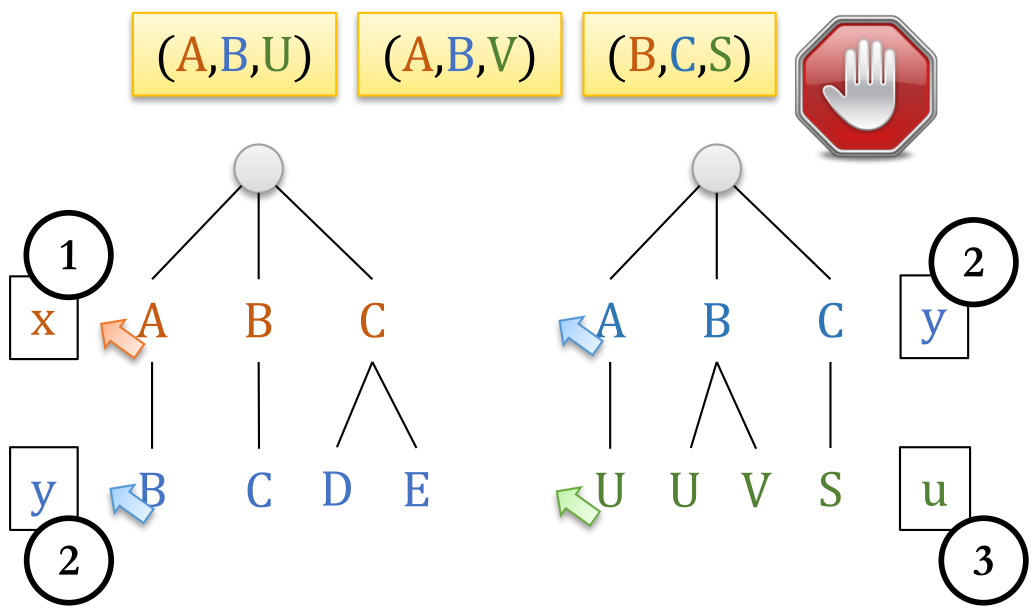
Join Processing for Graph Patterns:
An Old Dog with New Tricks [Nguyen+ 15]

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

Married(x,y)	
A	B
B	C
C	D
C	E



Univ(y,u)	
A	U
B	U
B	V
C	S

Caching in LFTJ [Kalinsky,Etsion,K 17]

Etsy

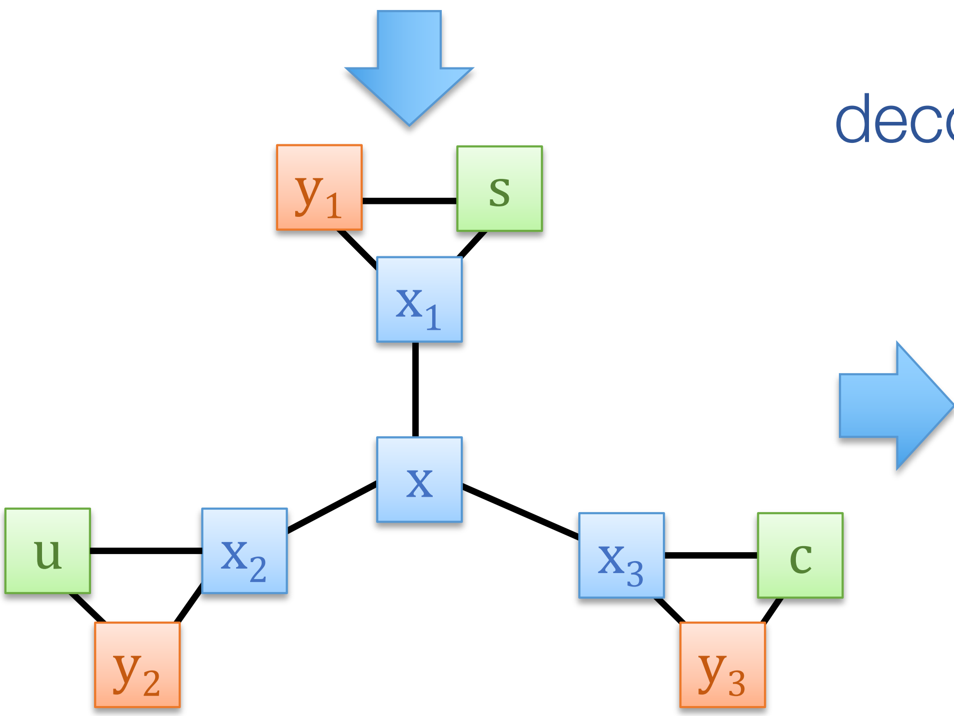
Artist(x),

Friends(x_1,y_1) , School(x_1,s) , School(y_1,s),

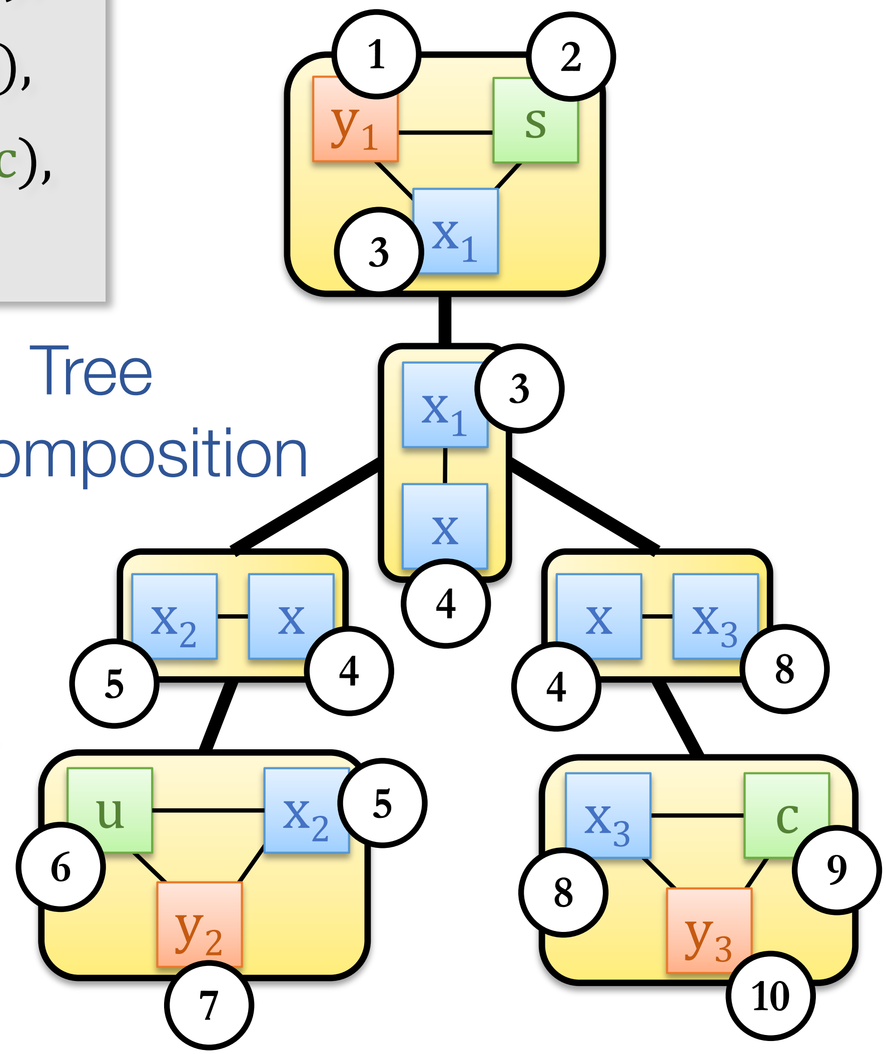
Colleagues(x_2,y_2) , Univ(x_2,u) , Univ(y_2,u),

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)

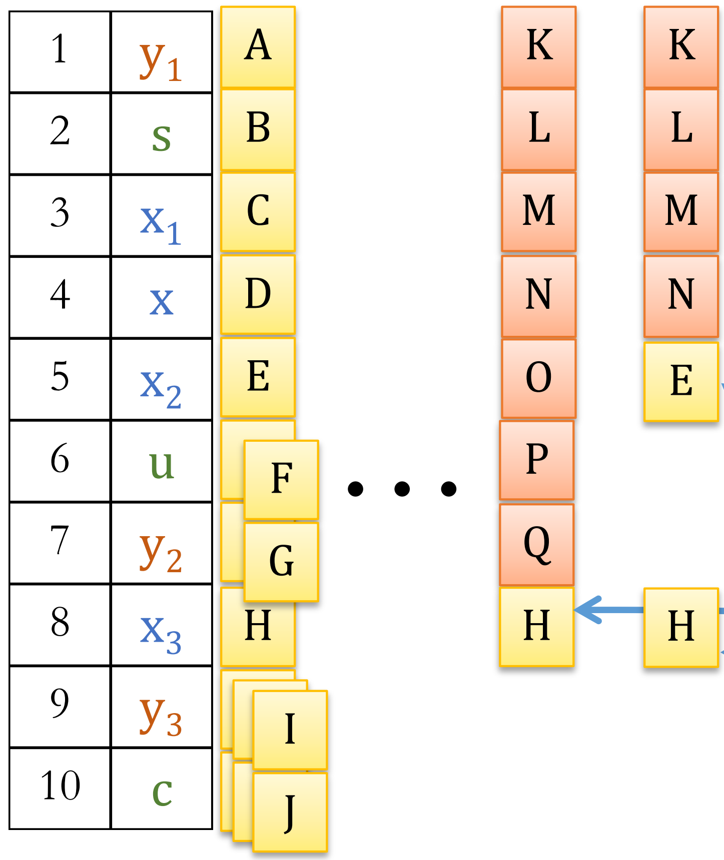


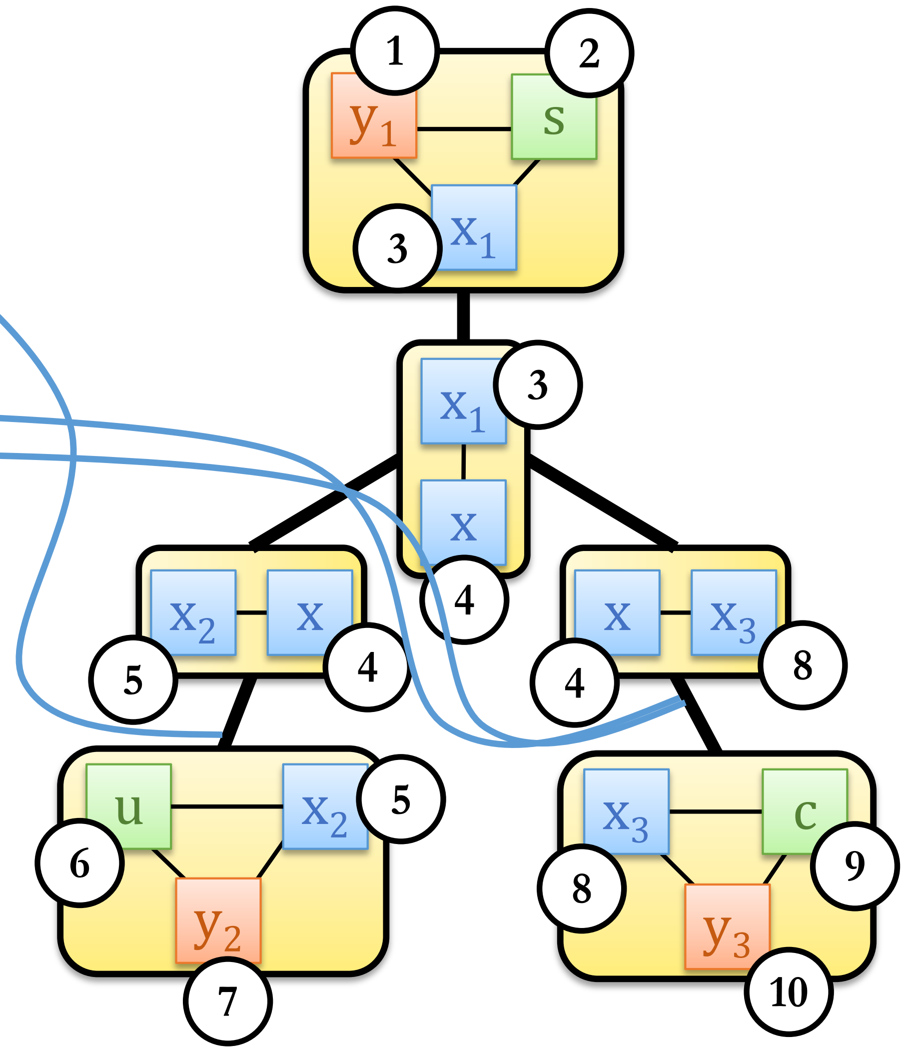
Tree decomposition



Caching in LFTJ [Kalinsky,Etsion,K 17]

- Caching (unlike LFTJ)
- Flexible caching (unlike ord. TD)

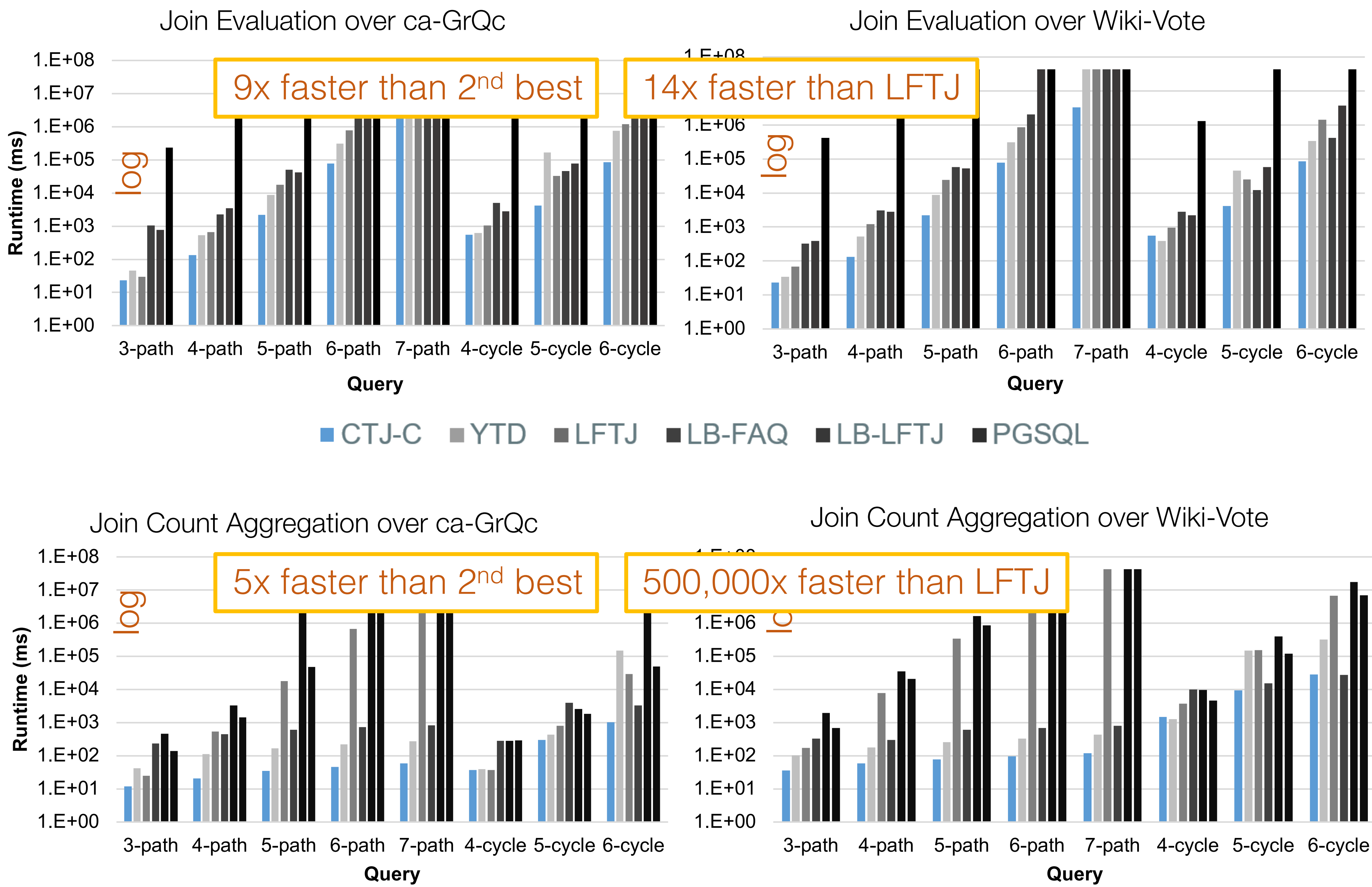




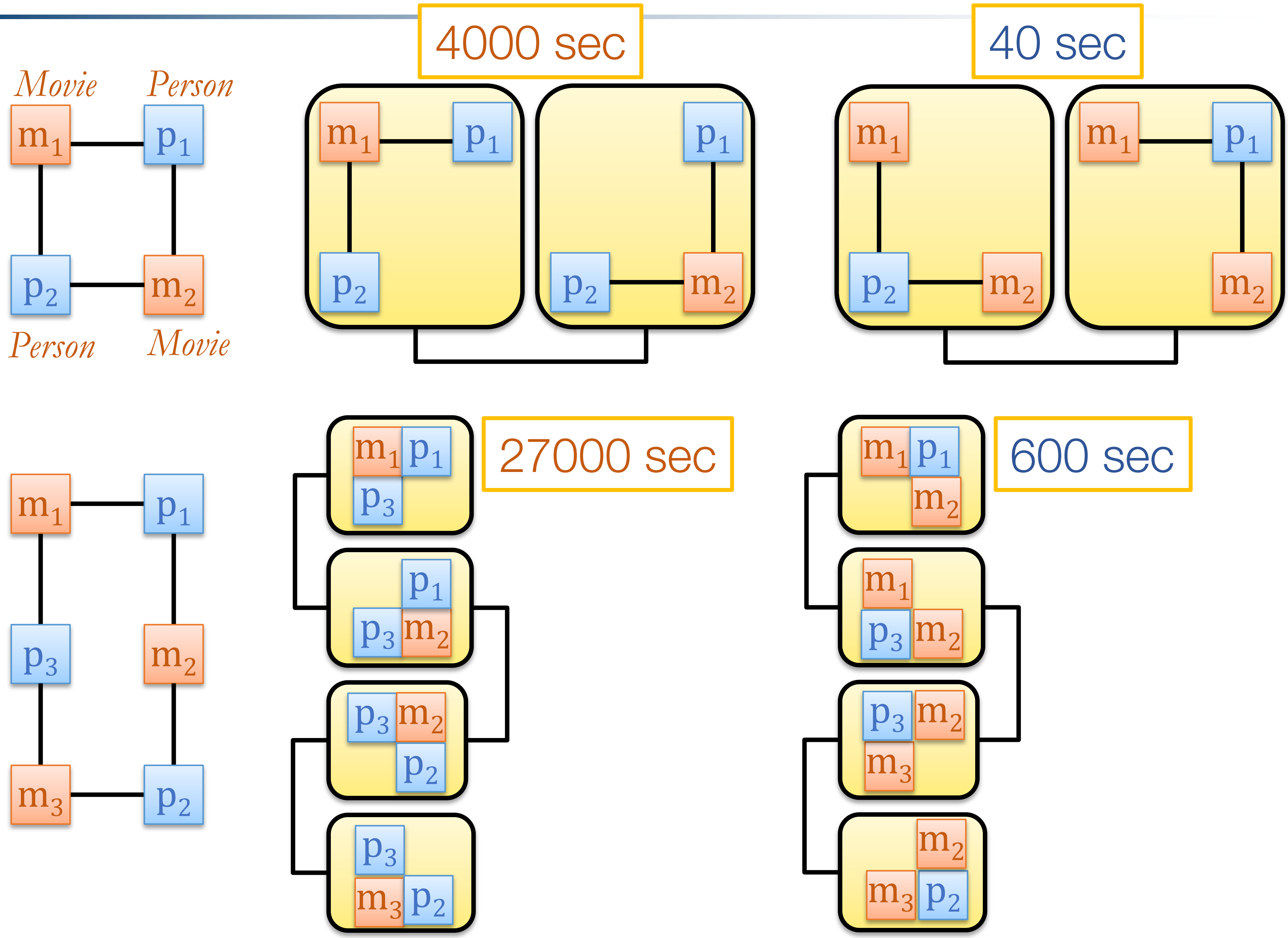
In [Kalinsky,Etsion,K 17]:

- Caching policies
- TD selection
- Extension to count(join)

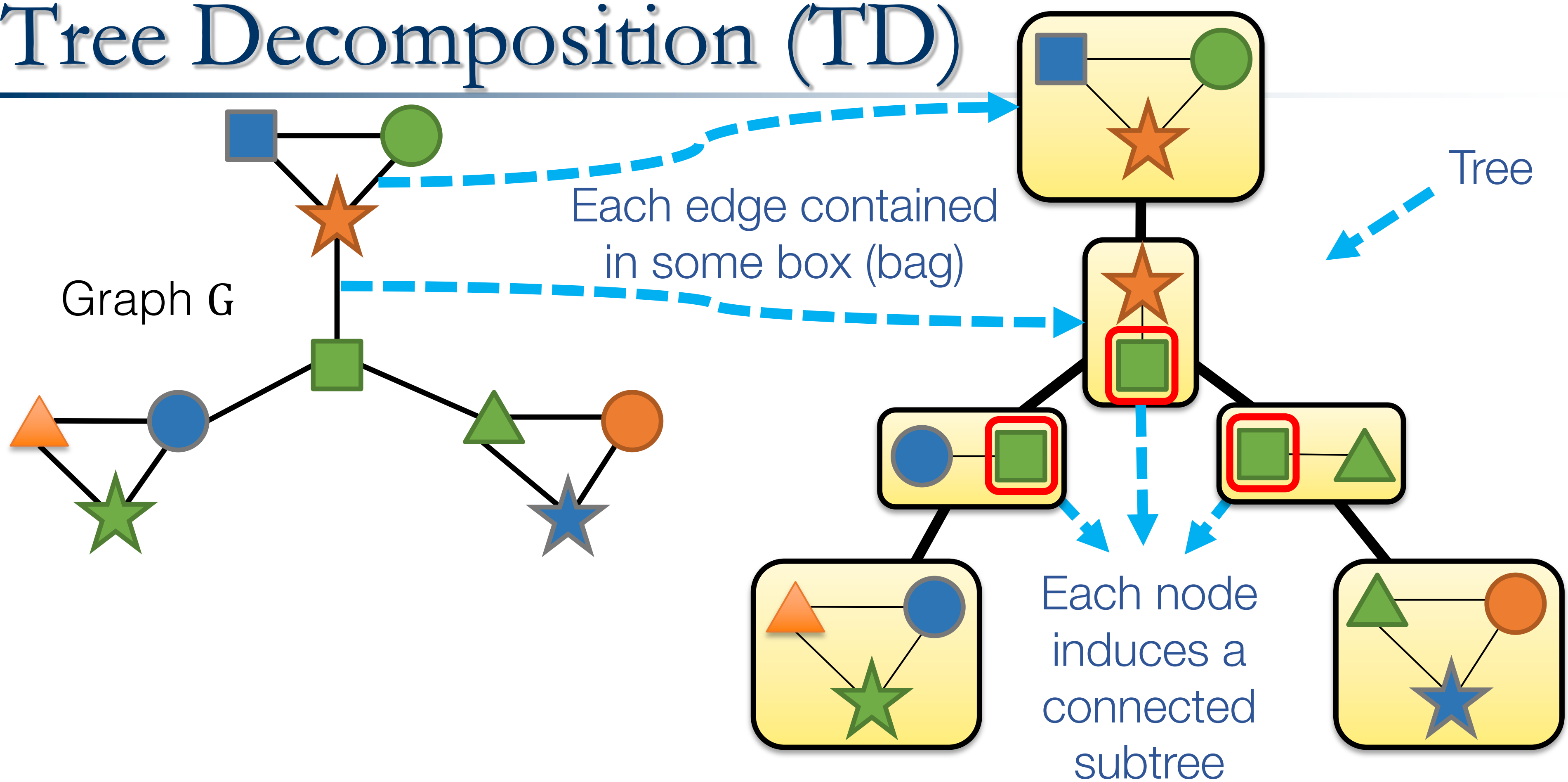
Experimental Evaluation



TD Selection Matters!



Tree Decomposition (TD)

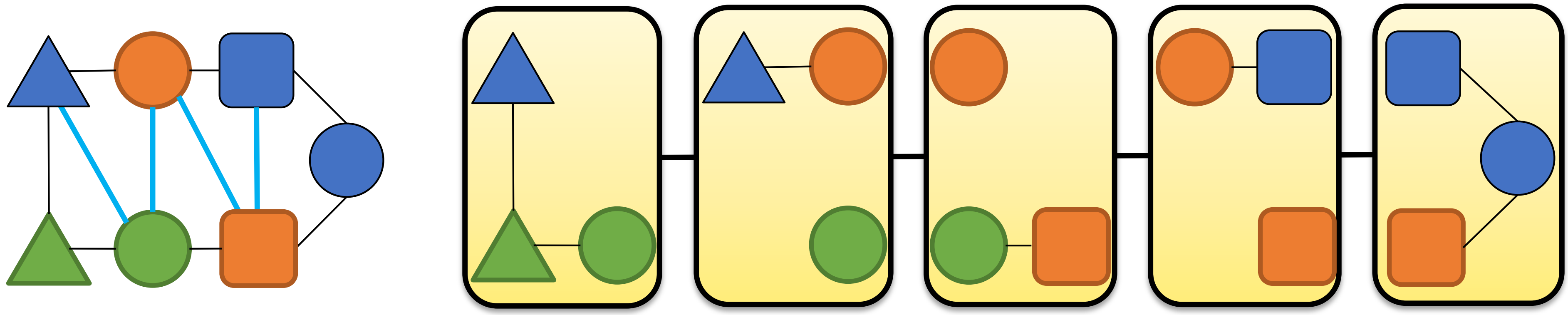


Definition: **Tree Decomposition (TD) of a Graph G**

(t,β), t a tree, β a mapping $\mathrm{nodes}(t) \longrightarrow 2^{\mathrm{nodes}(G)}$ where:

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

Standard Goodness Measures



width=2   fill-in=4

TD width: max(|bag|)-1

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

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

(t,β), t a tree, β a mapping $\mathrm{nodes}(t) \longrightarrow 2^{\mathrm{nodes}(G)}$ where:

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

Running Example: Width/Fill-in

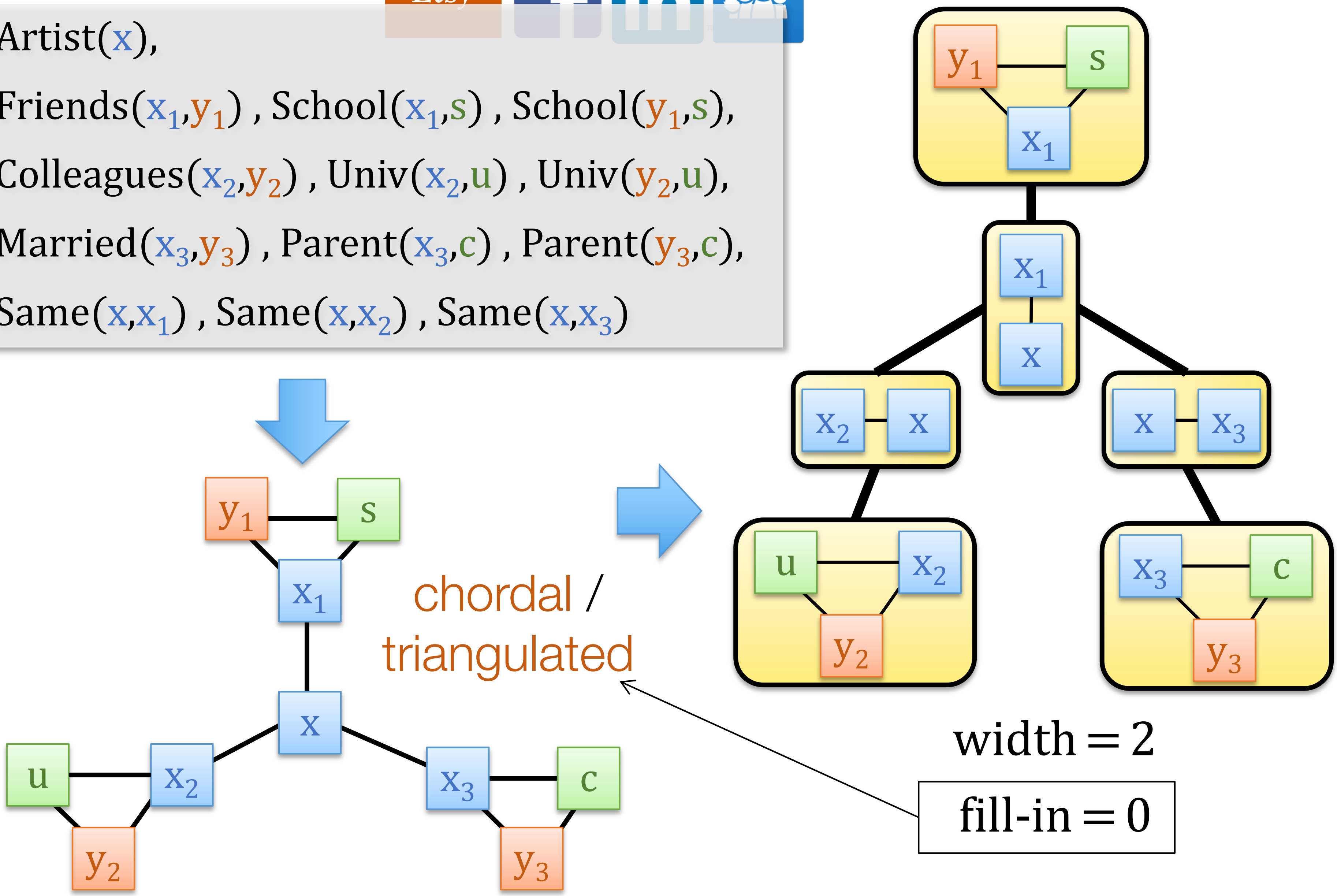
Artist(x),

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)



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]

Finding a Good TD (Query Planning)

- *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

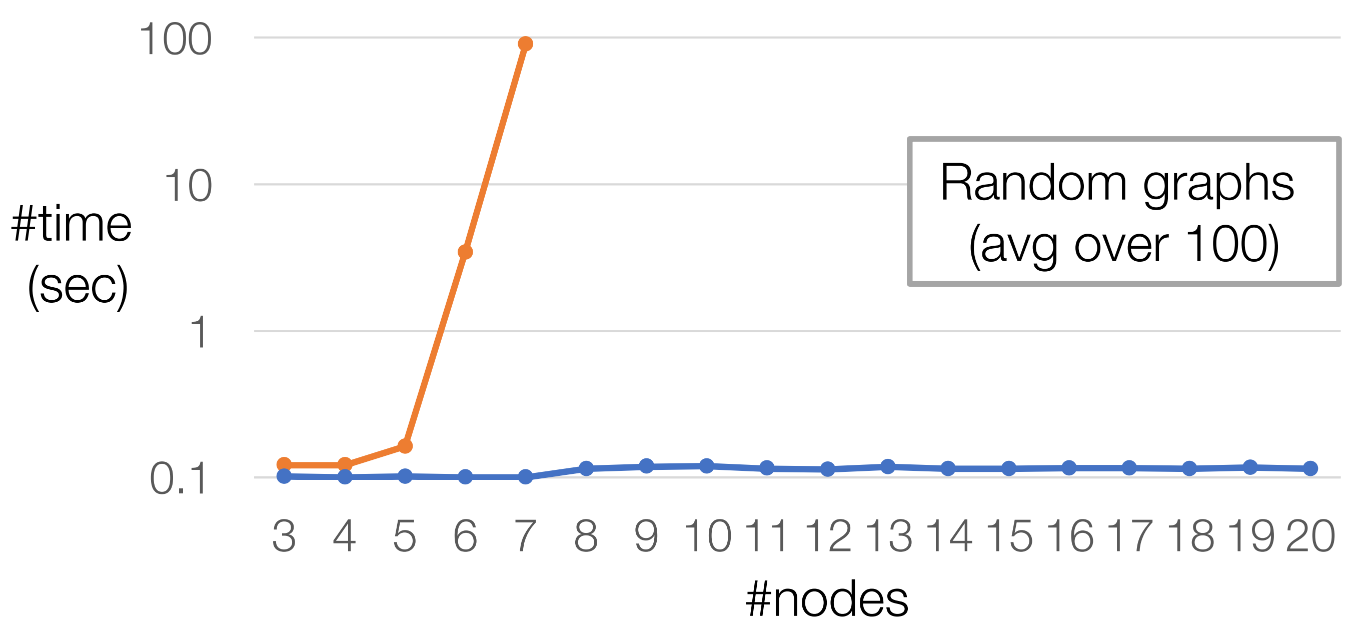
- Introduction
- Cliques in Inconsistent Databases
- Cliques in Query Planning
  - Caching in Trie Join
  - Enumerating Tree Decompositions
  - Ranked Enumeration

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: $\mathrm{F}(\text{td-width}, \text{pseudo-tree-height})$
      - Used the algorithm presented next
  - ...
- ML applied to learn TD scores (over TD features) from problem instances [Abseher+ 15]

Solutions?

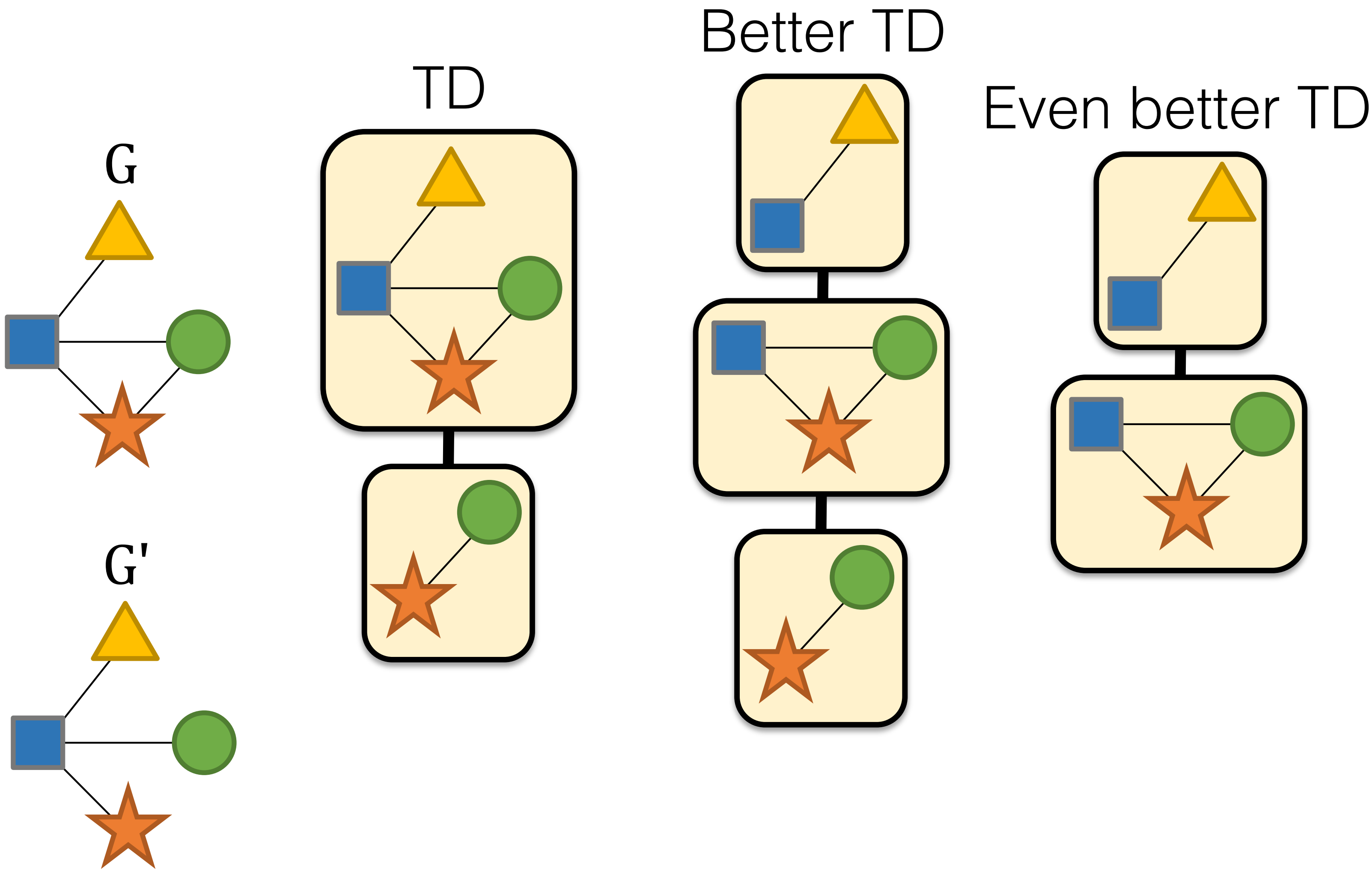
- Generator of [Abseher+ 15] (ML)
  - Generate a handful (10)
  - Best-effort randomness, no guarantees
- Duncecap [Tu,Ré 15]: candidate generator of *generalized hypertree decompositions*
  - Goal: join optimization
  - No efficiency guarantees, designed for small query graphs



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]

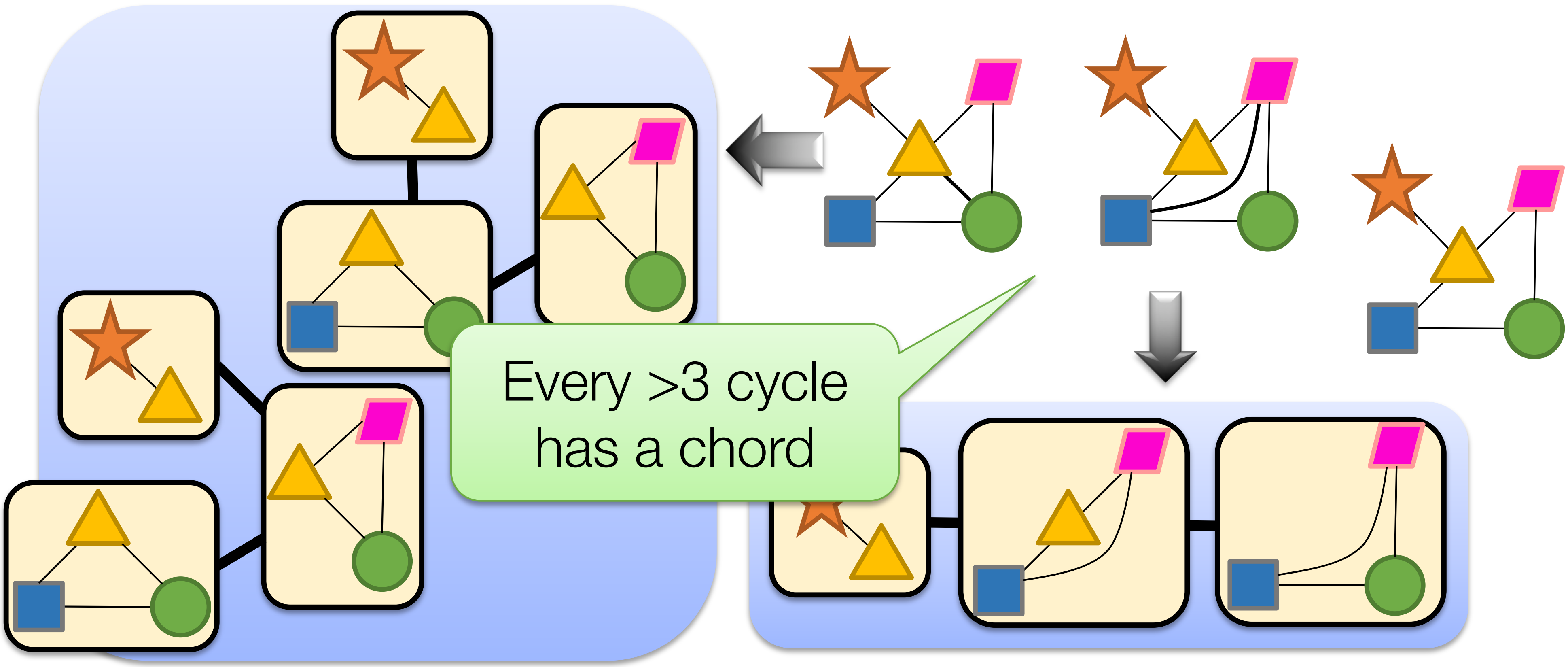
Can enumerate in incremental poly. time:

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

Wait – *related to this talk?*

(1) From Proper TDs to Min Triangs

Proposition: efficient translation between classes of bag-equivalent proper TDs ⇔ minimal triangulations

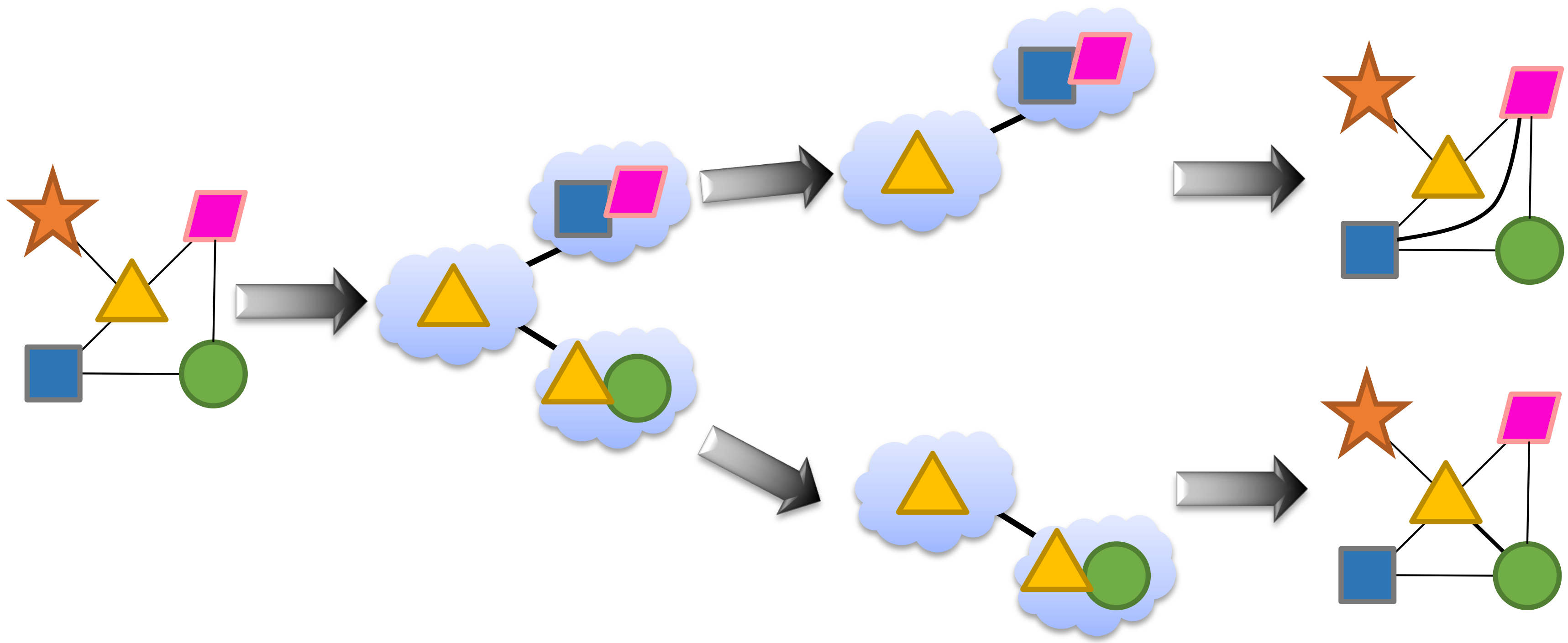


Efficient: $\leqslant n$ max cliques [Gavril 74]; reduce to max spanning trees over max cliques [Jordan 02]; enum max spanning trees [Yamada+ 10]

(2) From Min Triangs to Min Separators

A bijection [Parra,Scheffler 97]:

min triangs ⇔ max sets of non-crossing min separators



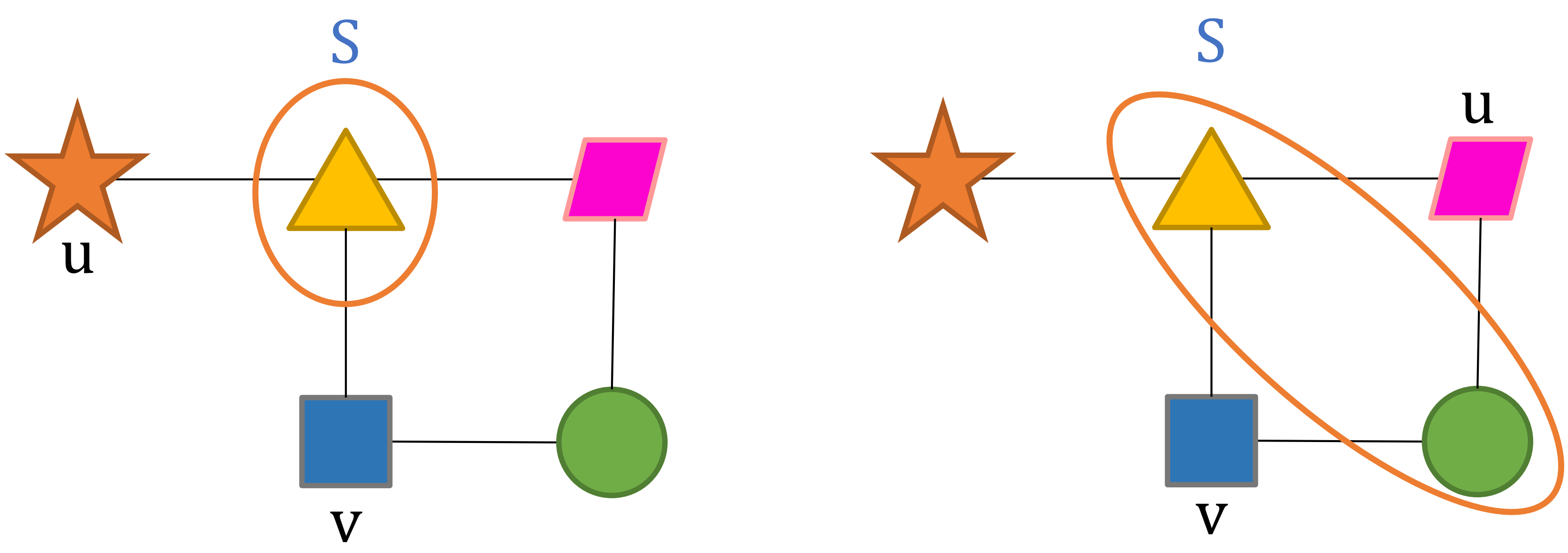
Non-crossing: *not separating nodes of the other*

(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)

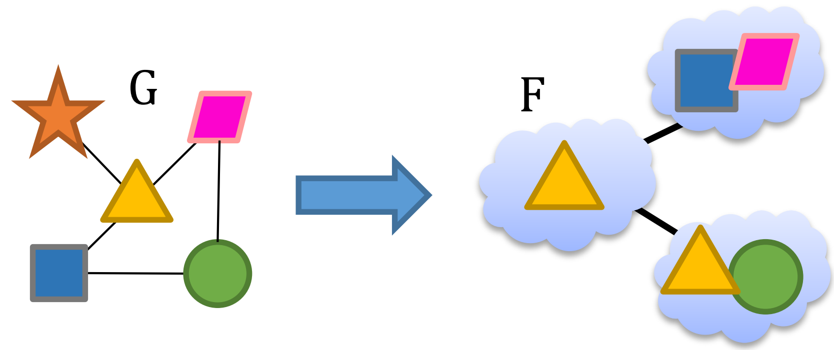


Solution?

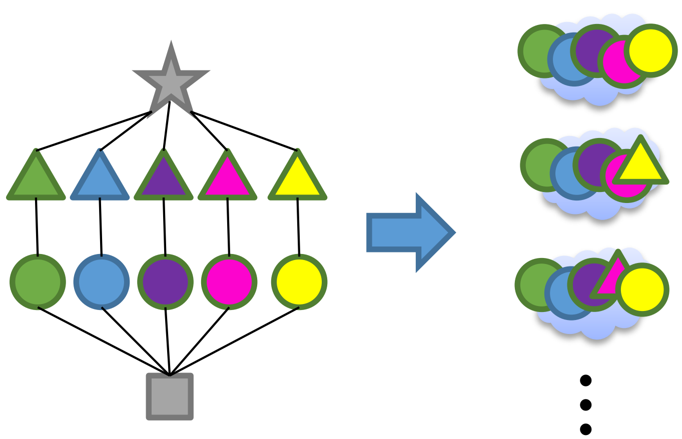
Given a graph G:

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!

Enumeration Algorithm

- 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)

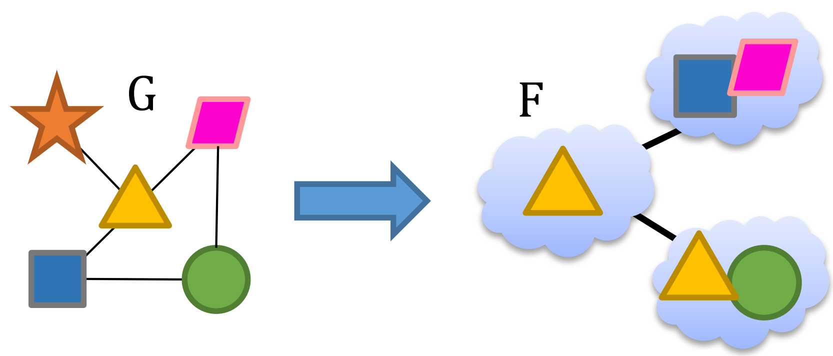
alg.	measure	avg #results	avg #≤first	avg min	avg %improv	max %improv
MCS-M	width	33635.0	12733.4	20.2	2.6%	26.3%
	fill-in		12724.9	2043.8	14.4%	55.8%
LB-T	width	11998.3	4744.1	18.5	3.4%	20.7%
	fill-in		1013.6	965.8	2.2%	27.6%

Outline

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

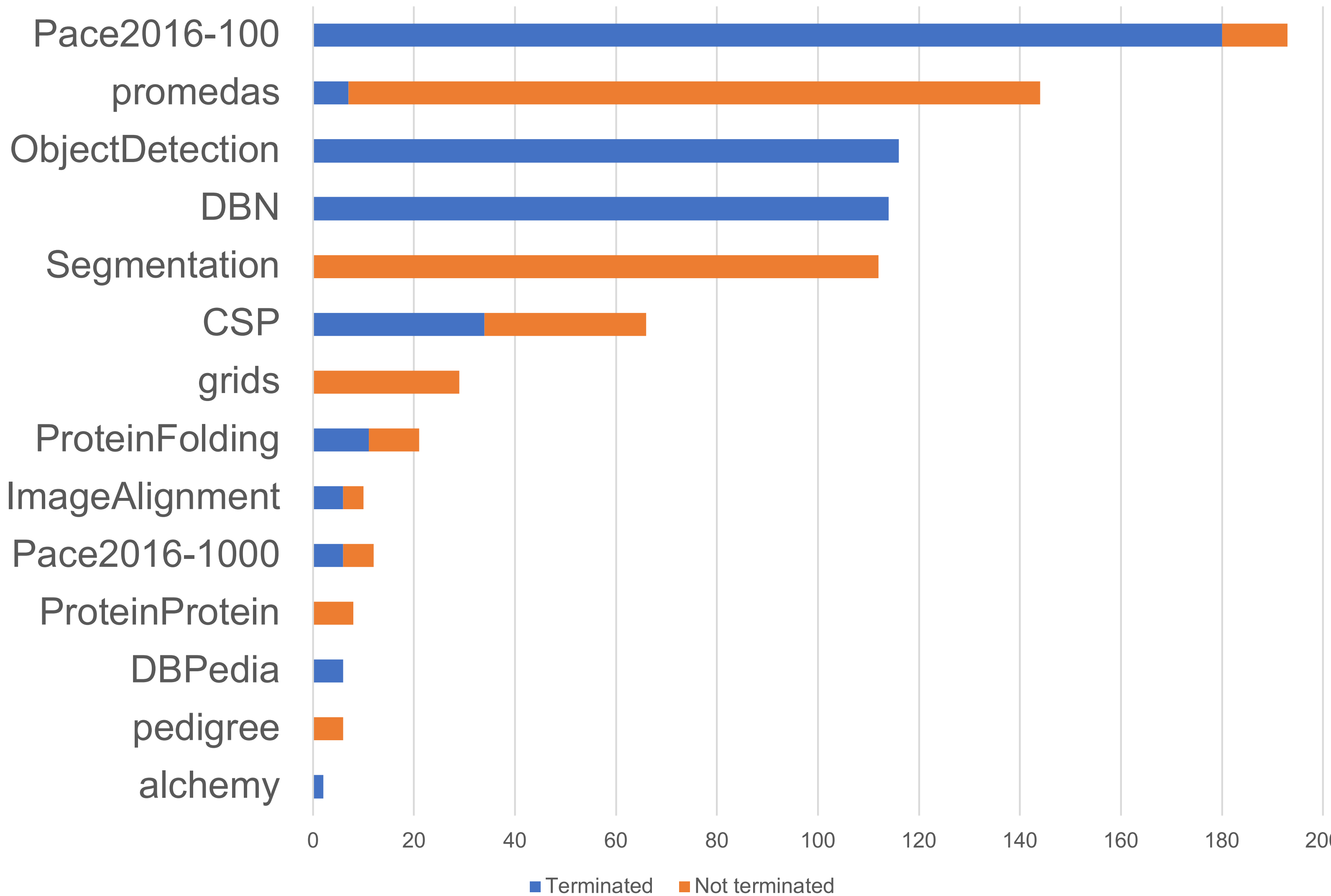
The Case of Poly #Min-Separators

- General case: inc. poly. time
- If #min-separators bounded by a polynomial:
  - Min triangs 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?*

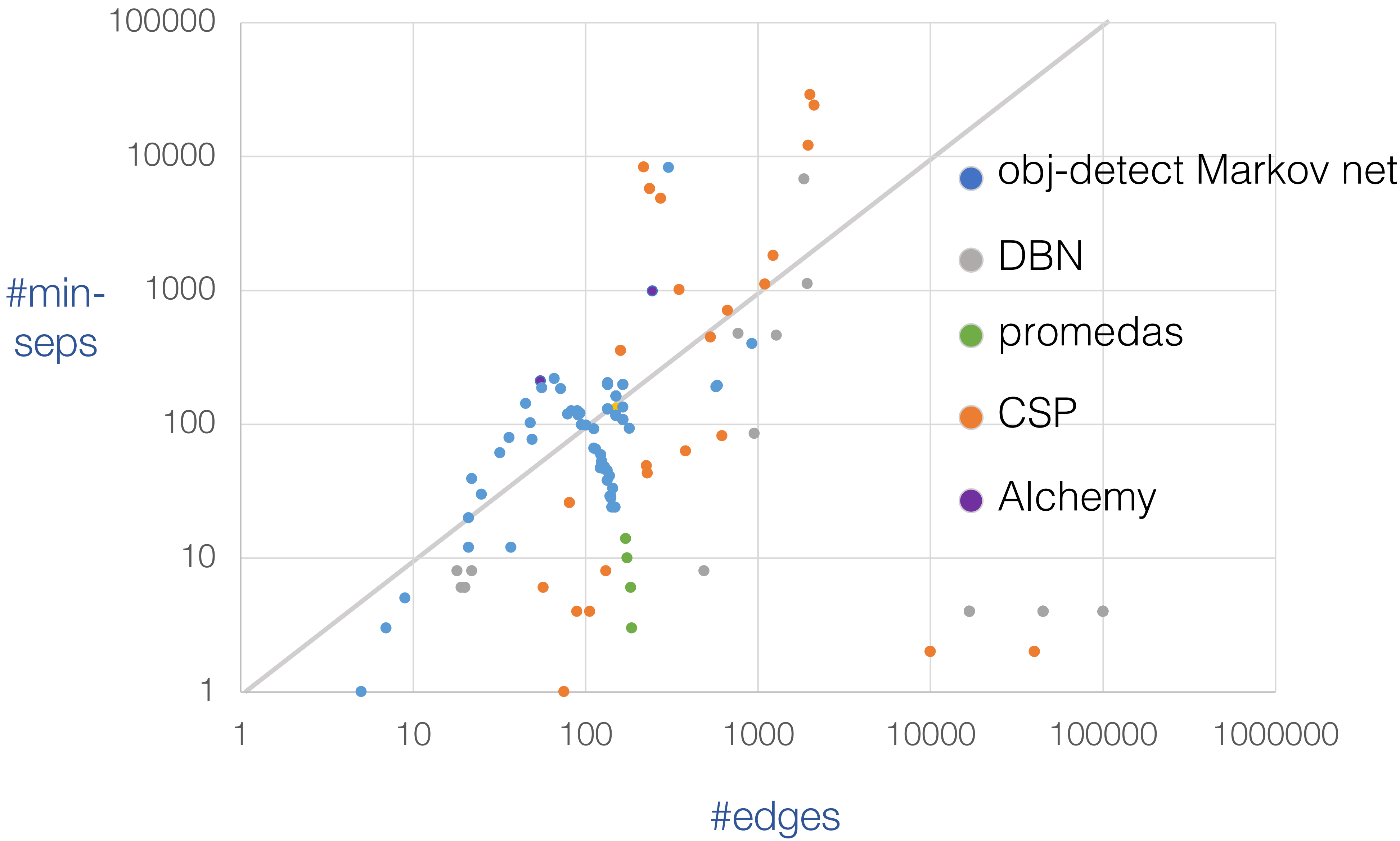


Hardness Distribution

Terminated in 1 min? ● Yes ● No

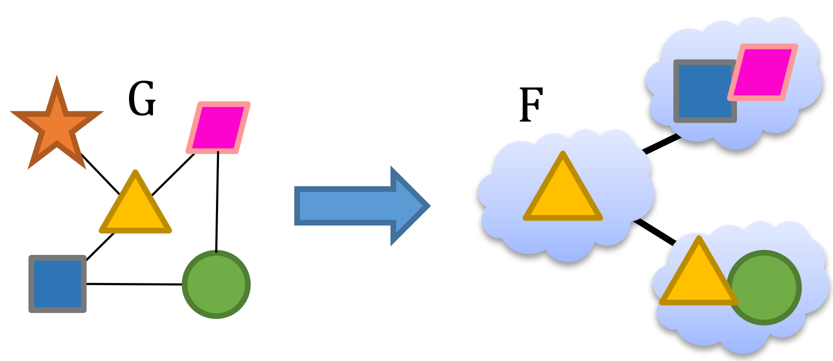


Observed # Minimal Separators



The Case of Poly #Min-Separators

- General case: inc. poly. time
- If #min-separators bounded by a polynomial:
  - Min triangs 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?*
- *Can we get ranked enumeration?*



THEOREM [Ravid,Medini,K PODS'19]

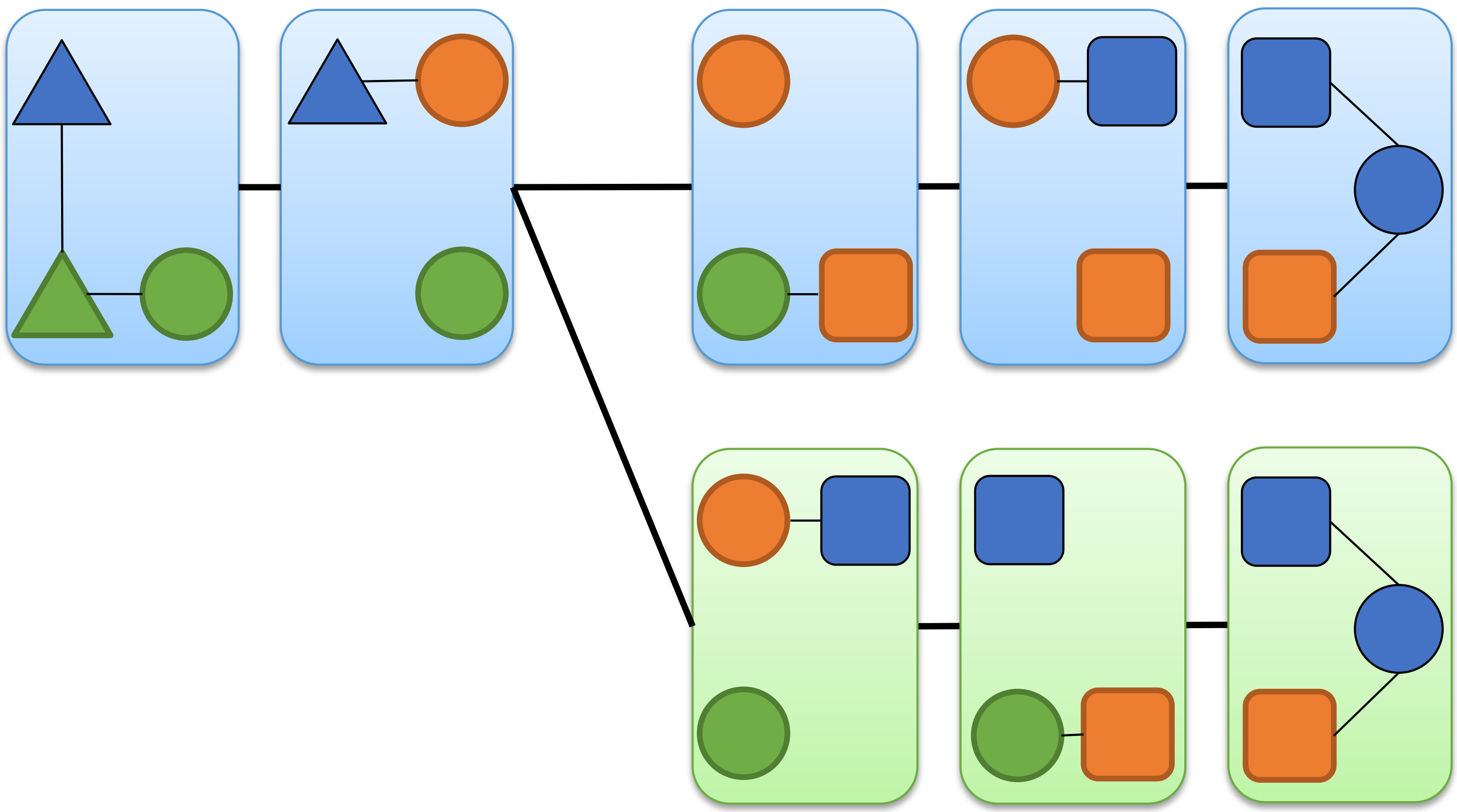
If #min-separators < $\mathbf{poly(G)}$, then min triangs (and proper TDs) can be enumerated with:

- polynomial delay
- increasing cost

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

For every fixed $\mathbf{w}$, min triangs (& proper TDs) of width $<\mathbf{w}$ 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