PODS 2026, Bengaluru, India
June 1st, 2026

Enumeration Theory *through the Lens of* Database Challenges

Benny Kimelfeld
Technion – Israel Institute of Technology
& RelationalAI

Accompanying Paper (Companion)

Enumeration Theory through the Lens of Database Challenges

Florent Capelli
capelli@cril.fr
Artois University
Lens, France

Nofar Carmeli
nofar.carmeli@inria.fr
Inria, LIRMM, University of Montpellier, CNRS
Montpellier, France

Alessio Conte
alessio.conte@unipi.it
University of Pisa
Pisa, Italy

Benny Kimelfeld
bennyk@cs.technion.ac.il
Technion - Israel Institute of Technology & RelationalAI
Haifa, Israel

Reinhard Pichler
reinhard.pichler@tuwien.ac.at
TU Wien
Vienna, Austria

Nikolaos Tziavelis
ntziavel@ucsc.edu
UC Santa Cruz
Santa Cruz, California, USA






The
ENUMERATION
MULTIVERSE
Ways to sit
Subsets
Orderings
All possible choices
Ways to arrange
Partitions
Collections
Ways to describe

Plan

Chapter 1:

Many-Solution Problems in Databases

Chapter 2:

Concepts in Enumeration Theory

Chapter 3:

Reusable Algorithmic Techniques

Chapter 4:

The Angle of Query Answering

Enumeration in Database Challenges

- Many DB challenges involve **listing many solutions**
  - Variety of aspects!
- Often, nontrivial algorithms needed to guarantee an **efficient enumeration**
- Exhaustive search:
  - May take too long if #solutions is manageable
  - Has long latency/throughput of reported solutions
  - Does not support paging of ranked solutions
- Next, a few examples
  - References to pubs on enumeration algorithms & complexity
  - "Efficient enumeration" defined in the next chapter

Enumeration in Query Answering

Most fundamental DB op: **List all query answers**

Authorship

paper	author

Inproc

paper	conf

Location

conf	country

Sponsor

conf	comp	$

Affiliation

author	comp

- Authors and conferences (**join**): Authorship ⋈ Inproc
- Sum of funding per country (**join-group-aggregate**)

```
  SELECT country, sum($)
    FROM Location NATURAL JOIN Sponsor
GROUP BY country
```

- Funders who published in their sponsored conference (**multiway join**):

Authorship ⋈ Inproc ⋈ Location ⋈ Sponsor ⋈ Affiliation

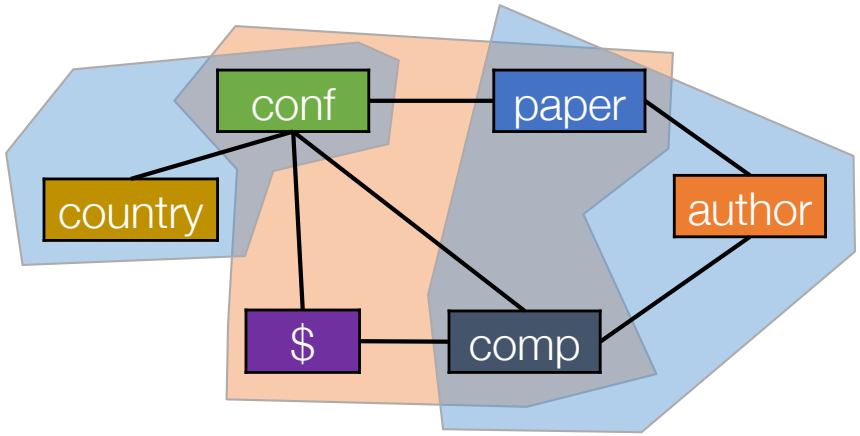
Wait for the last chapter …

(+ PODS tutorials Tue, Wed)

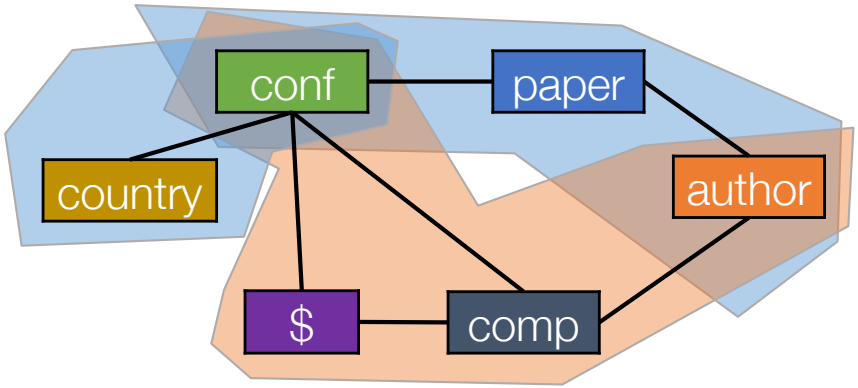
Enumeration in Query Planning

[Carmeli, Kenig, K PODS17] [Ravid, Medini, K PODS19]
[Waas & Galindo-Legaria SIGMOD00] [Brosse, Limouzy, Mary ICALP22] [Lou+ICDT26]

Not an **acyclic** join… need a *hypertree decomposition*



Location ⋈ (Sponsor ⋈ Inproc) ⋈
(Authorship ⋈ Affiliation)



…

Location ⋈ (Sponsor ⋈ Affiliation) ⋈
(Authorship ⋈ Inproc)

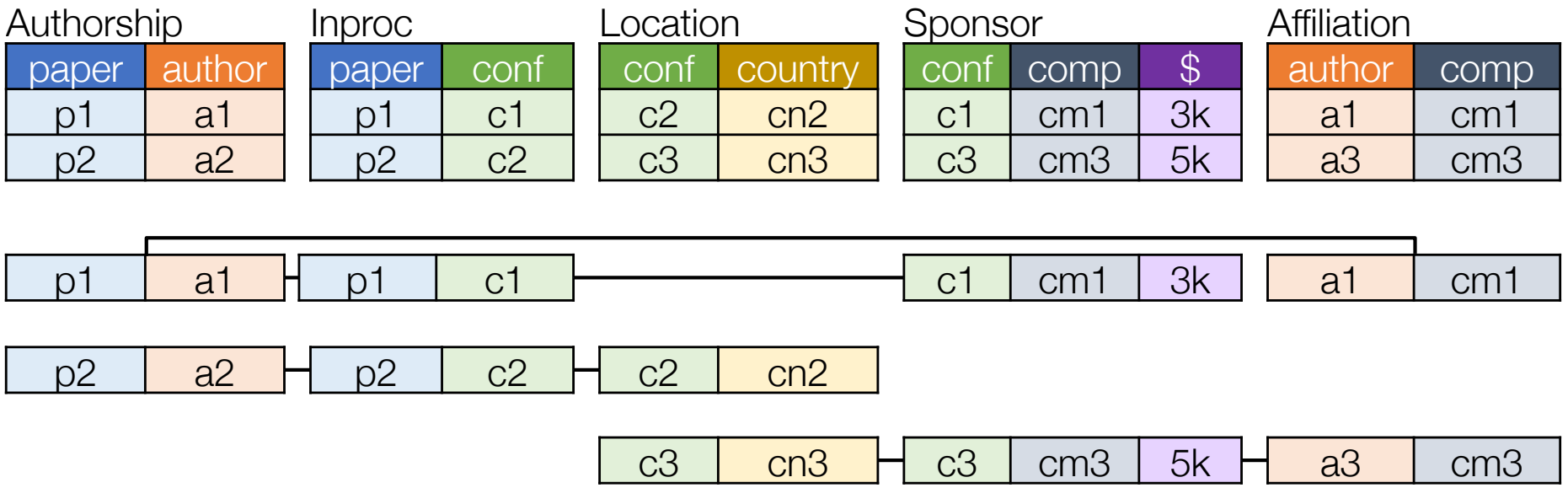
Authorship ⋈ Inproc ⋈ Location ⋈ Sponsor ⋈ Affiliation

Join plans → **Tree decompositions** of a graph → **Minimal triangulations** of a graph

Enumeration in Multiway Outer Joins

[Kanza & Sagiv 2003] [Cohen & Sagiv 2005] [Cohen+2007]

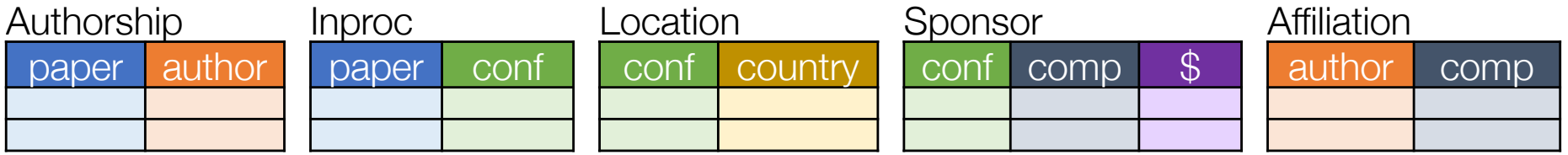
Full disjunction: associative multiway version of the (full) **outer join**

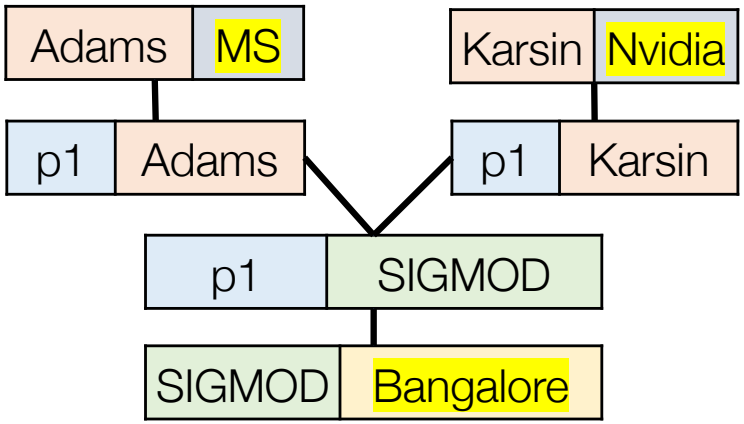


Maximal sets of join-consistent tuples, connected

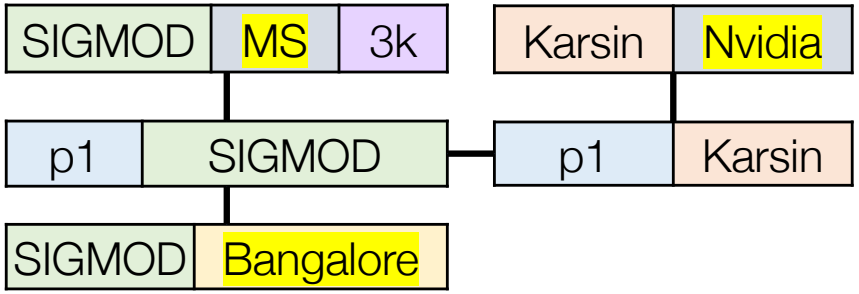
Enumeration in Database Keyword Search

[K & Sagiv PODS06] [Golenberg, K, Sagiv SIGMOD08] [Kargar, An, Yu 13]





Microsoft *&* ***Nvidia*** *have a joint paper at SIGMOD in* ***Bangalore***



Microsoft *sponsors SIGMOD in* ***Bangalore****, where* ***Nvidia*** *has a paper*

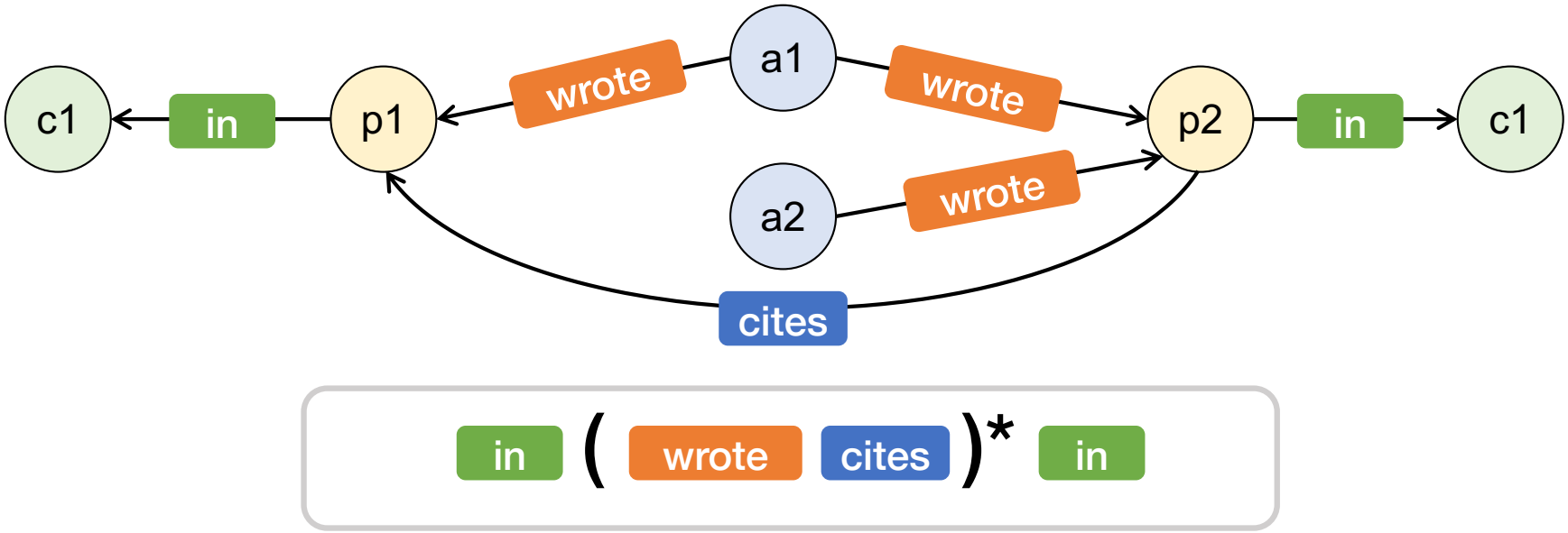
...

Systems: **BANKS**, **DBXplorer**, **Spark**, **EASE**, **BLINKS**, …

Abstraction: Given a graph and set of nodes, find all **Steiner trees** by (approximately) increasing weight

Enumeration in Graph Databases

[Martens & Trautner ICDT18] [Casel & Schmid ICDT21]
[Popp 22] [David, Francis, Marsault PODS24]



Given a graph DB and a regular expression, enumerate all **matching paths** between the source and target nodes

- Several variants: **shortest** paths, **simple** paths, **trails** (no repeated edges)
- Extensions to regular path queries (RPQs) [Abo Khamis+ICDT26]

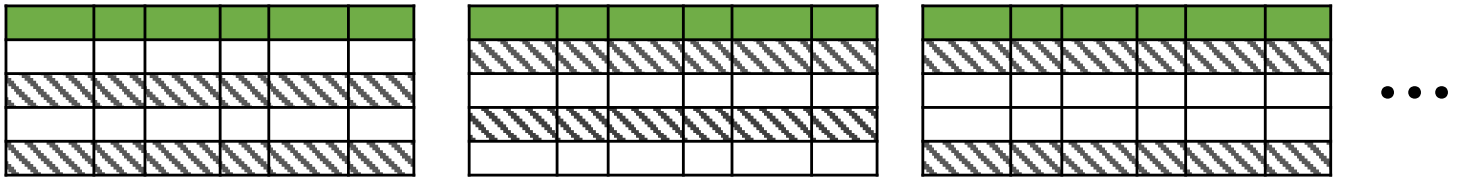
Enumeration in Data Quality Management

Inproceedings

title	lead	venue	year	city	country
DB queries	Anna	SIGMOD	2026	Bangalore	India
DB queries	Bob	SIGMOD	2026	London	UK
DB index	Bob	VLDB	2025	London	UK
DB ML	Cali	VLDB	2025	London	USA

title, lead, year ⟶ venue
venue, year ⟶ city, country
city ⟶ country
title, venue, year ⟶ lead

- Data cleaning: **enumerating database repairs**
  - [K, Livshits, Peterfreund PODS18] [Conte+25]
  - Special case: *hypergraph transversals* (max independent sets of a hypergraph) [Eiter & Gottlob 95] [Eiter, Gottlob, Makino 03] [Kenig & Mizrahi ICDT25]



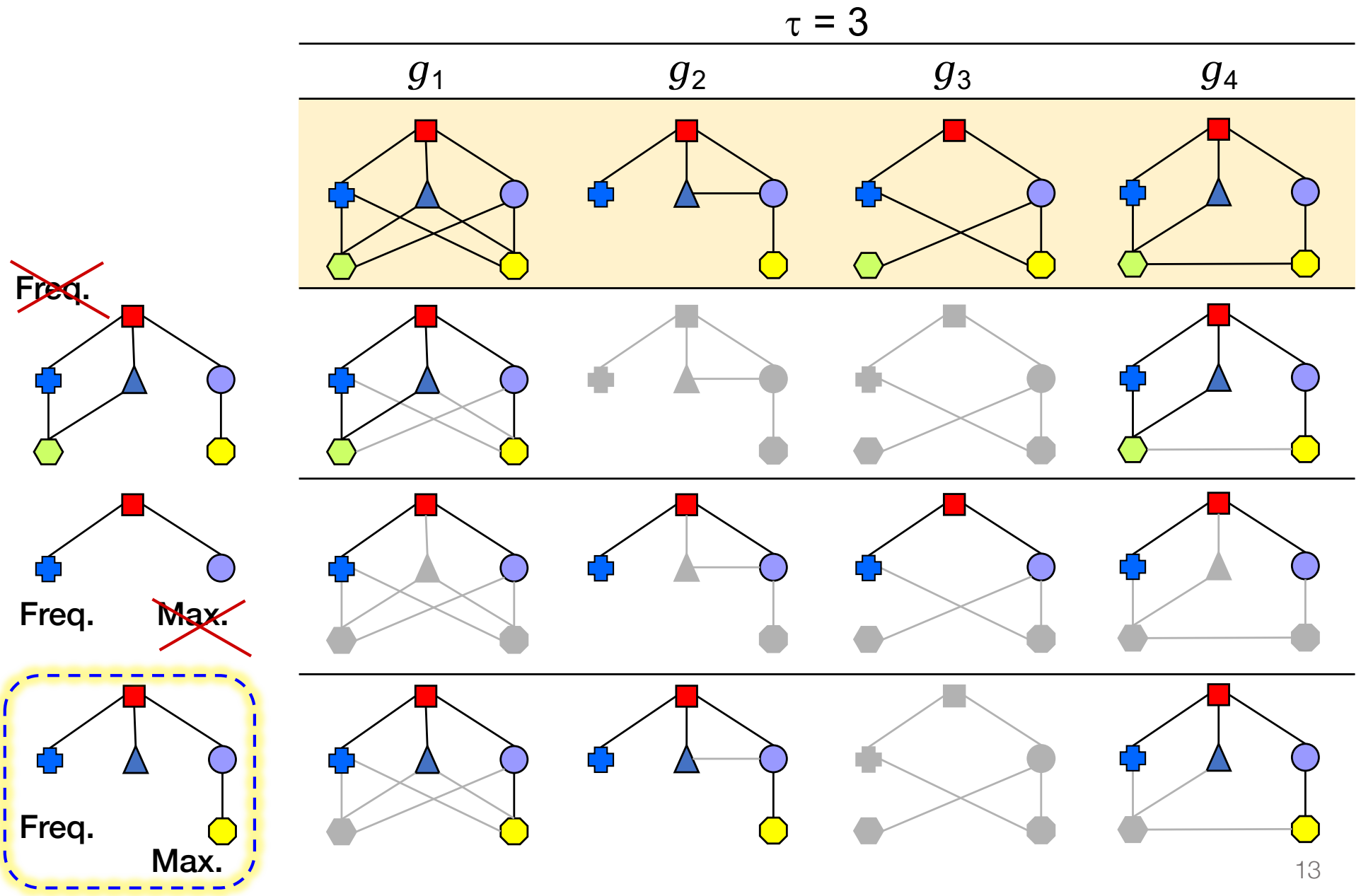
- Data profiling/cleaning: **constraint discovery**
  - Enumerating approximate *denial constraints* [Livshits+VLDB20]
  - Enumerating approximate *multivalued dependencies* [Kenig+SIGMOD20]

- Schema design/normalization: **enumerating candidate keys**
  - {title, lead, year} {title, venue, year} …
  - [Lucchesi & Osborn 78] [Ennaoui & Nourine 25]

Enumeration in Data Mining

- **Frequent itemset** mining
  - Given a collection of transactions (item sets), find *maximal item sets that co-occur together frequently*
    - Above some threshold τ of the transactions include all
    - [Boros+03] [Uno, Kiyomi, Arimura 04]
- **Frequent pattern** mining
  - Given a collection of (labeled) graphs, find *maximal subgraphs that occur frequently*
    - I.e., isomorphic to subgraphs of at least τ of the graphs
    - [K & Kolaitis 13]
    - Systems: **SPIN**, **MARGIN**

Maximal Frequent Patterns



Other Enumeration Problems

- Relational queries over **text** (document spanners)
  - [Florenzano+PODS18] [Amarilli+ICDT19] [Peterfreund+PODS19] [Gawrychowski+PODS26]
- **Community** detection in social data
  - "Almost" graph cliques (k-plexes)
  - [Berlowitz, Cohen, K SIGMOD15] [Yu+SIGMOD22]
- **Matrix** query languages
  - [Muñoz, Riveros, Vansummeren ICDT24]
- Likely answers over **probabilistic databases**
  - [K & Sagiv PODS07]
- Temporal **network analysis**
  - Identifying time-sensitive patterns
  - [Kumar & Calders VLDB17]
- …

Plan

Chapter 1:

Many-Solution Problems in Databases

Chapter 2:

Concepts in Enumeration Theory

Chapter 3:

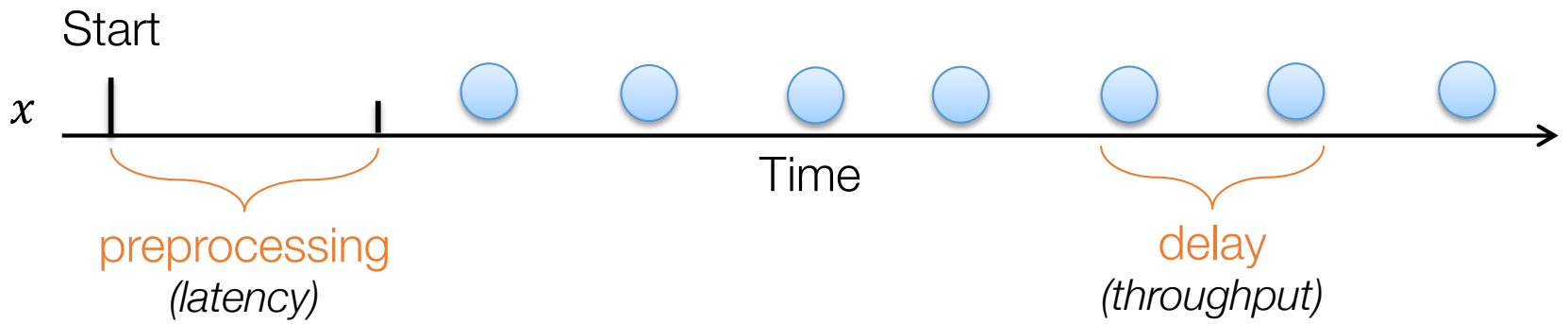
Reusable Algorithmic Techniques

Chapter 4:

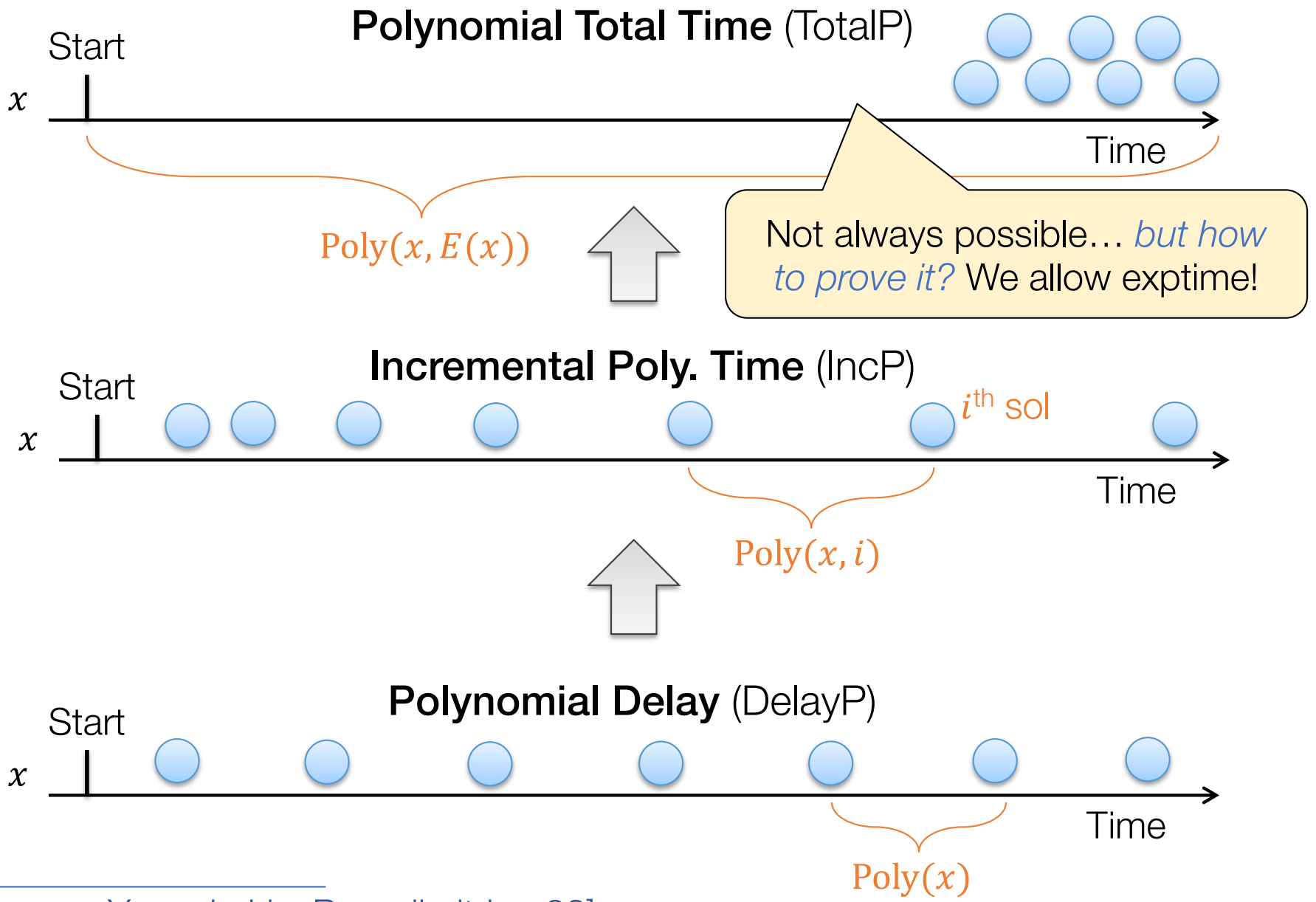
The Angle of Query Answering

Setup

- Enumeration problem E:
  - Input space $\mathcal{X}$; output space $\mathcal{Y}$
  - Every input $x \in \mathcal{X}$ has a finite $E(x) \subseteq \mathcal{Y}$ of solutions
- **Core task:** Given x, produce $E(x)$
  - Solutions reported throughout the execution, no duplicates
  - Typical #answers: exp (coarse-grained) ; large poly (fine-grained)
- **Delay**: time between two consecutive reports (throughput)
- **Preprocessing**: time before the first report (latency)



Notions of Tractability (Coarse Grained)



[Johnson, Yannakakis, Papadimitriou 88]

The One-More (-Solution) Problem

Simple argument: If hard to test whether *any* solution exists, then enumeration is intractable

- Oftentimes not enough – non-emptiness may be easy (even trivial), while enumeration is still intractable
- **Stronger tool** (very common): the OneMore problem –

OneMore Problem for Enum Problem E:

Given an input x and a set $S \subseteq E(x)$ of solutions, is there any solution that is **not** in S? (That is, is $S \neq E(x)$?)

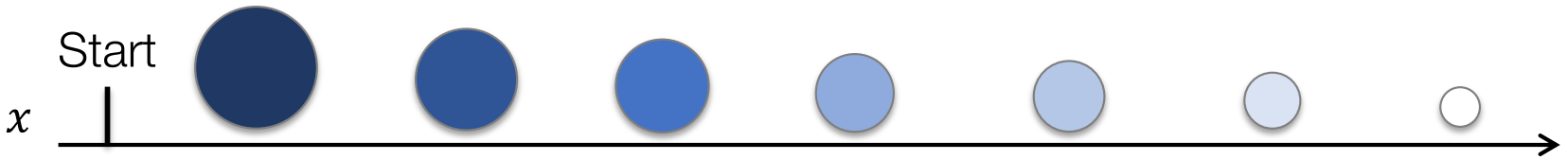
Proposition (folklore)

E is in **TotalP** $\Rightarrow$ OneMore is in **PTime**

Hence, impossibility often proved by showing that OneMore is hard

Ordered Enumeration

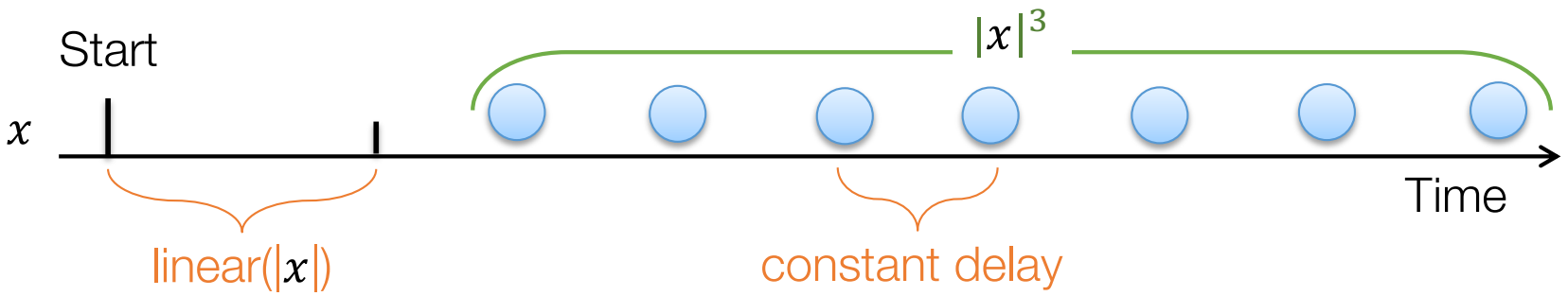
- In an ordered enumeration:
  - Solutions are ordered (ranked)
  - We require enumeration **in order** (first's first)



- Common orders:
  - Lexicographic (tuples, e.g., ORDER BY)
  - Decreasing score / increasing cost
- Also, notions of **approximation**: if S_1 is printed before S_2, then S_2 is at most α times better than S_1 [Fagin, Lotem, Naor 01] [K & Sagiv 06]

Coarse-Grained vs. Fine-Grained

- The previous tractability yardsticks are **coarse-grained**
  - Separating polynomial from exponential time
  - *Preprocessing* is usually not interesting – subsumed by 1st delay
- In **fine-grained** complexity analysis, we care about the actual degree of the polynomials
  - There, preprocessing matters! For example:



Challenge Slide: Tools of Lower Bounds

- Fine-grained complexity: rich ground of reference problems to choose from ; very useful
  - Somewhat risky: new **hardness conjectures** coming frequently
- Coarse-grained complexity: **limited machinery**
  - Exception: ONEMORE problem
  - Too few "hard" reference problems, e.g., minimal-transversal
- Rarely can we separate good **total time** from good **delay**
  - Easy to engineer artificial examples of separating problems

Plan

Chapter 1:

Many-Solution Problems in Databases

Chapter 2:

Concepts in Enumeration Theory

Chapter 3:

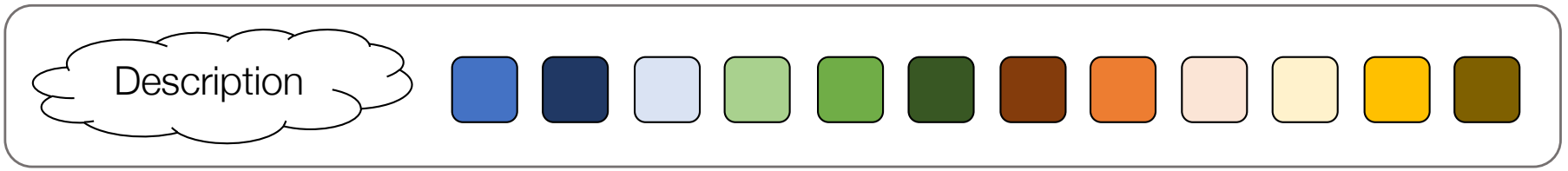
Reusable Algorithmic Techniques

Chapter 4:

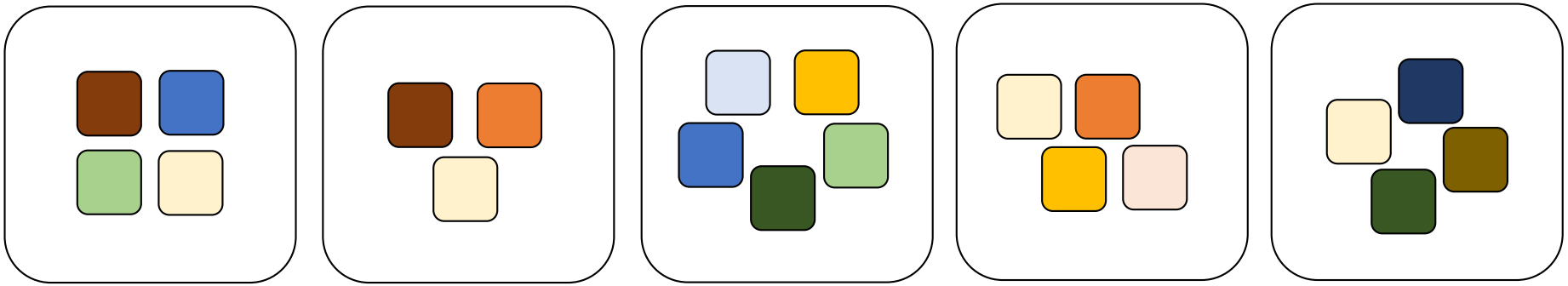
The Angle of Query Answering

Setup Revisited

Input x includes a set of **items**

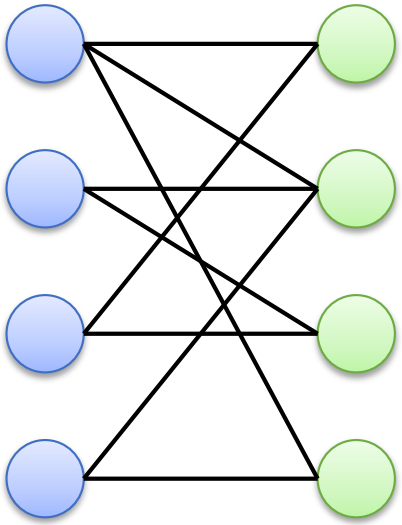


A **solution** is a set of items ($E(x) \subseteq 2^{\text{items}}$)



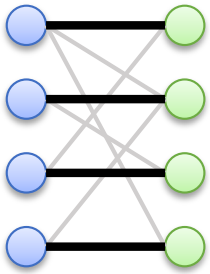
solution solution solution solution solution

Running Example: Enumerating Matchings

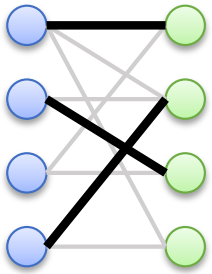


Input x**:** bipartite graph $G = (V, E)$ with $E(x) = \{ e_1, \ldots, e_m \}$

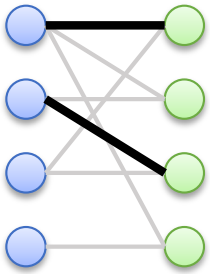
Matching: subgraph where each node has ≤ 1 edges



perfect



maximal



not maximal

Items: Edges $e_1, \ldots, e_m$

Solutions: Edge sets $M \subseteq E$ inducing "good" matchings

Binary Partition (Flashlight Search)

Inclusion item set

Exclusion item set

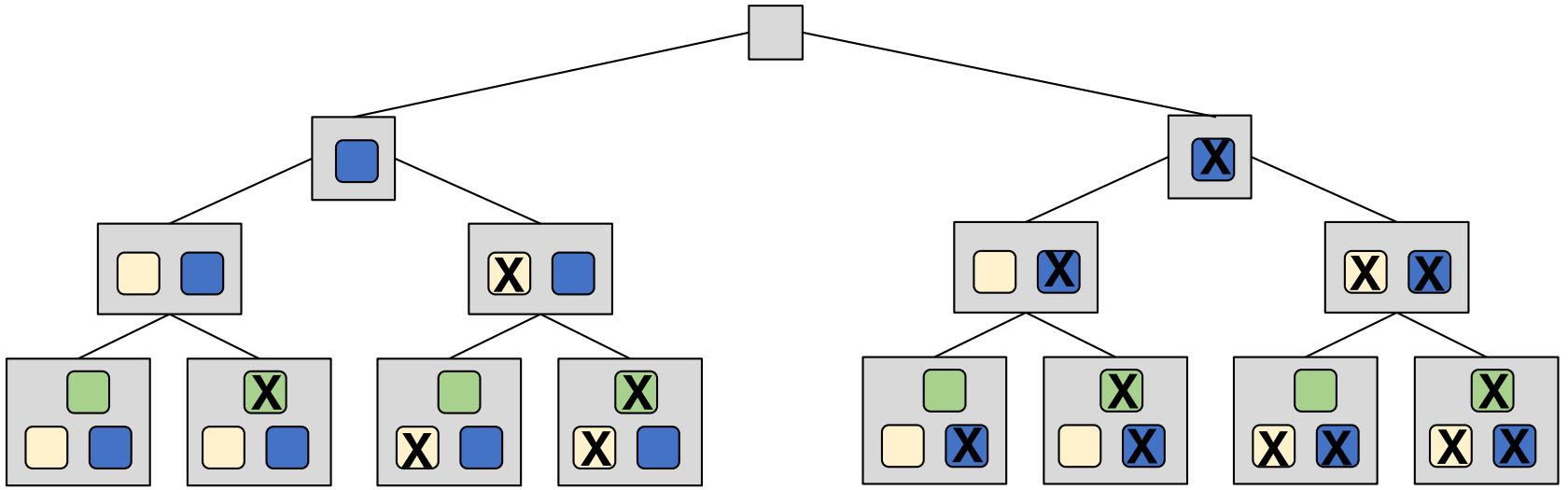
Enum(x, ∅, ∅)

Enum(x, Inc, Exc):

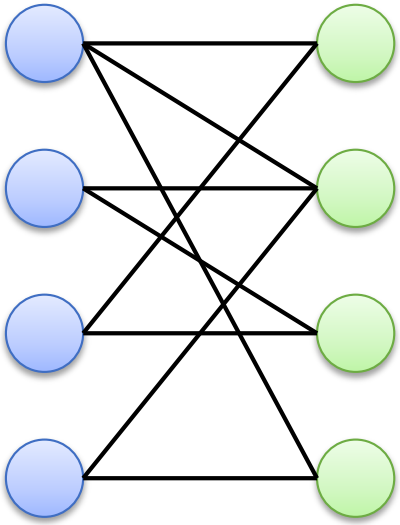
- **If** no solution includes Inc and excludes Exc, **return**
- **If** Inc is a solution, **print**(Inc)
- Choose an item $e \notin$ Inc ∪ Exc
- Enum(x, Inc ∪ {e}, Exc)
- Enum(x, Inc, Exc ∪ {e})

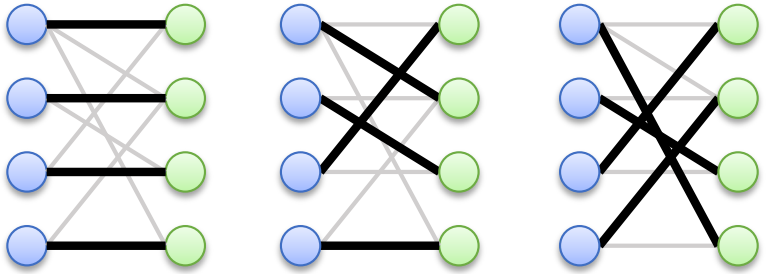
If none exist, return

If PTime emptiness test, then PDelay



Example: Perfect Matchings

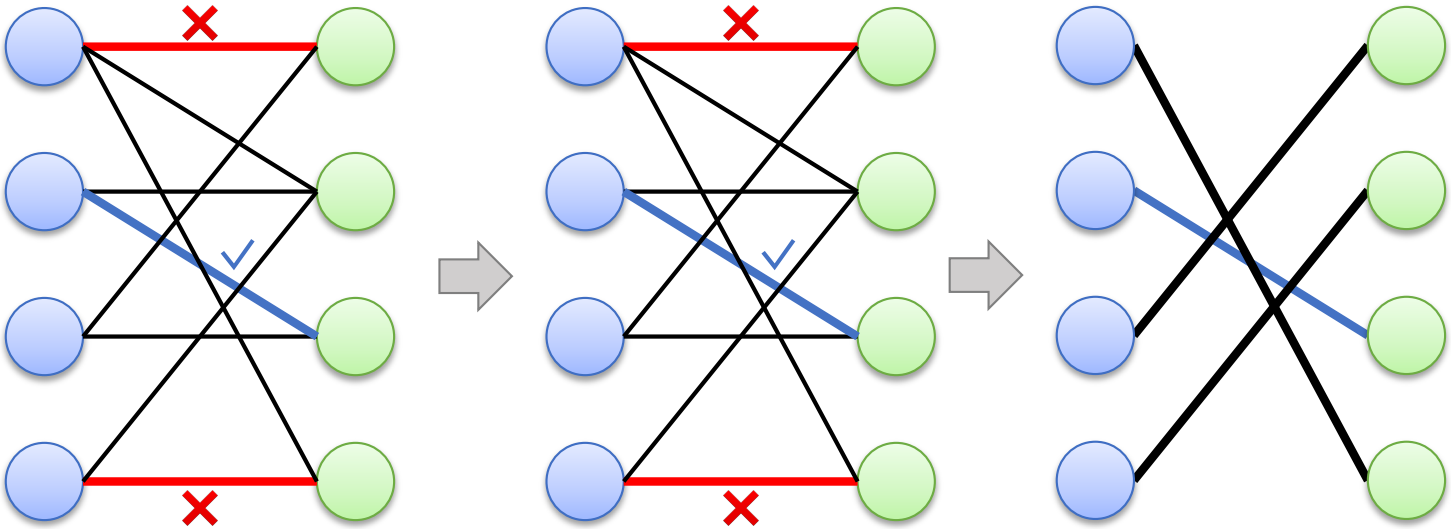




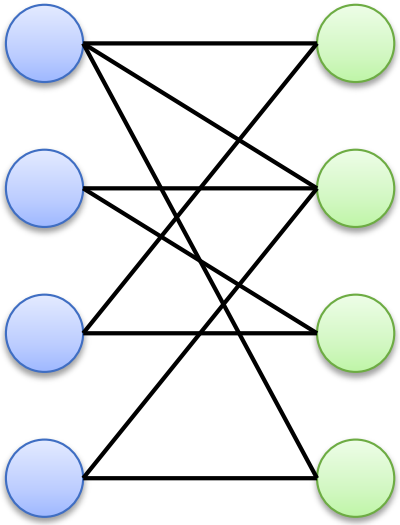
CONCLUSION:

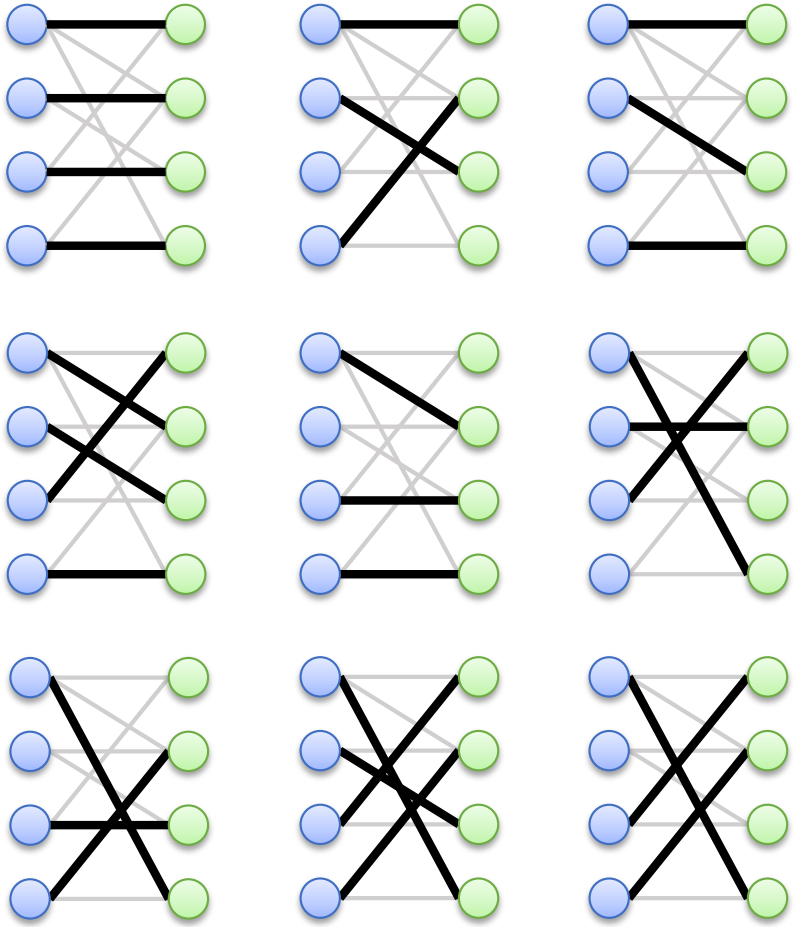
Perfect matchings enumerable w/ **PDelay**

Handling inclusion / exclusion constraints:

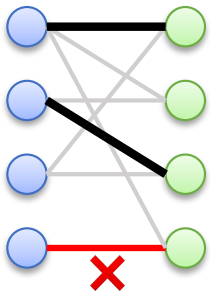


What about **Maximal** Matchings?





Problem: Need to assure that excluded edges are **not used**, and **cannot be used**!

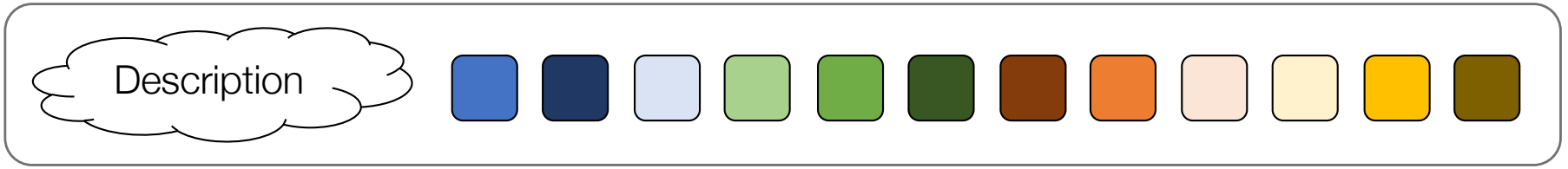


not maximal

We cannot just throw away bad solutions… too many of these

Maximal Subsets with a Hereditary Property

Input x includes a set of **items**

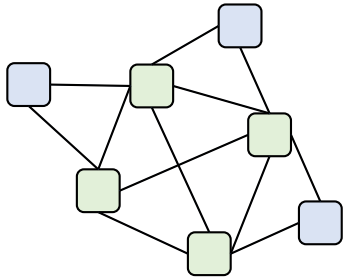


A **hereditary property** of subsets: closed to taking subsets

The **solutions** are the **maximal** subsets with the hereditary property

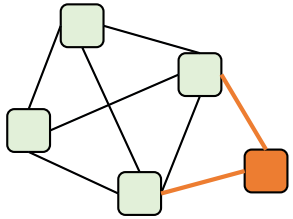
The Input-Restricted Problem

Unrestricted enumeration problem



(Enumerate the maximal subsets with hereditary property $\mathcal{P}$)

Input restricted: $\mathcal{P}$ is satisfied by the set of *all items except for one*

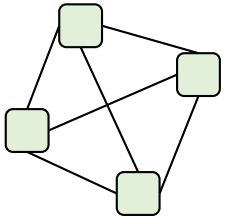


THEOREM:

Similar complexity as the unrestricted problem:

- TotalP ↔ TotalP
- IncP ↔ IncP
- PTime → PDelay

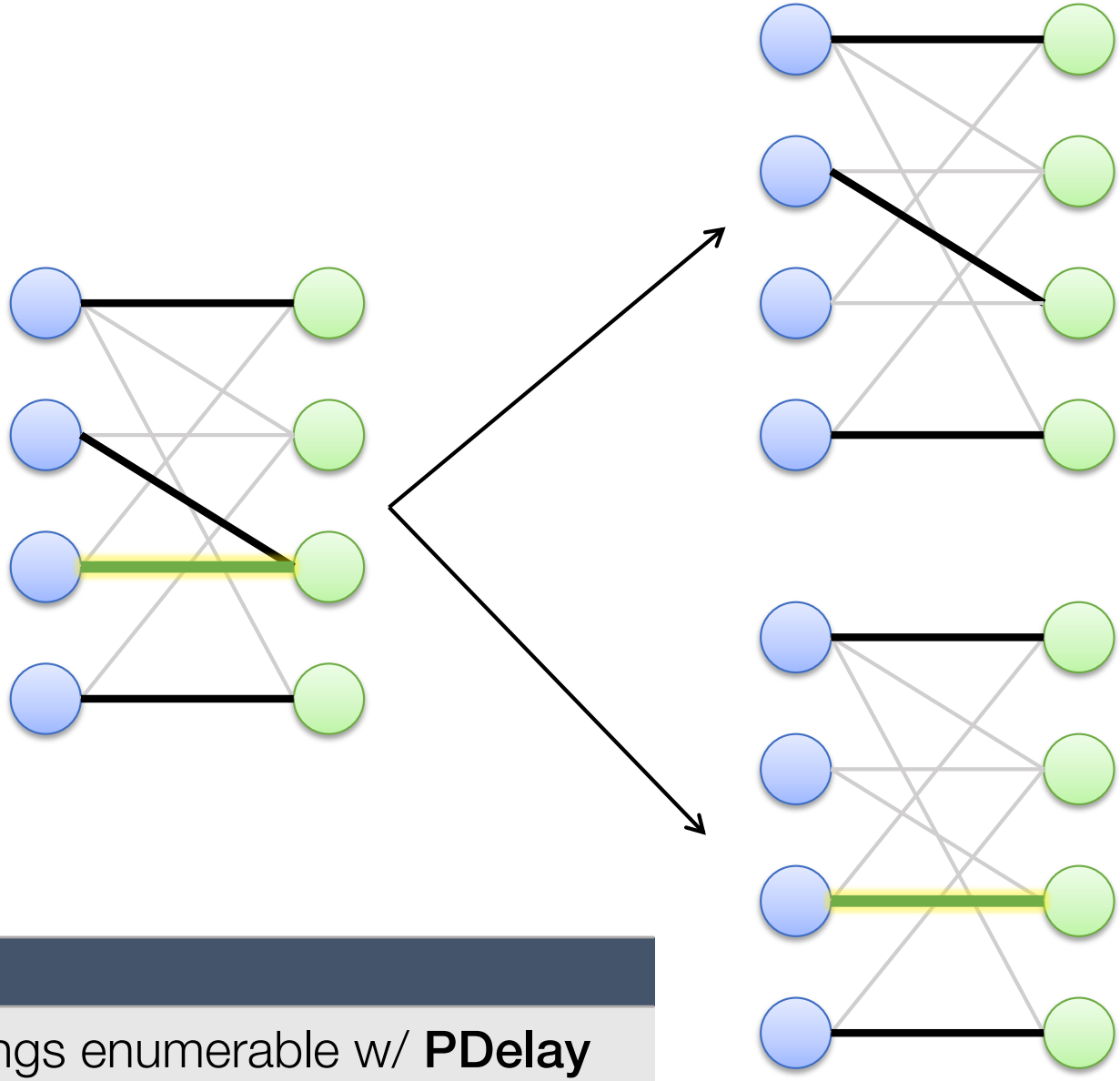
Trivial case: $\mathcal{P}$ is satisfied by the set of all items



(1 solution: all)

Many DBMS applications: join-trees, candidate keys (schema design), MVDs, outer joins, repairs, frequent patterns, communities, prob DBs, …

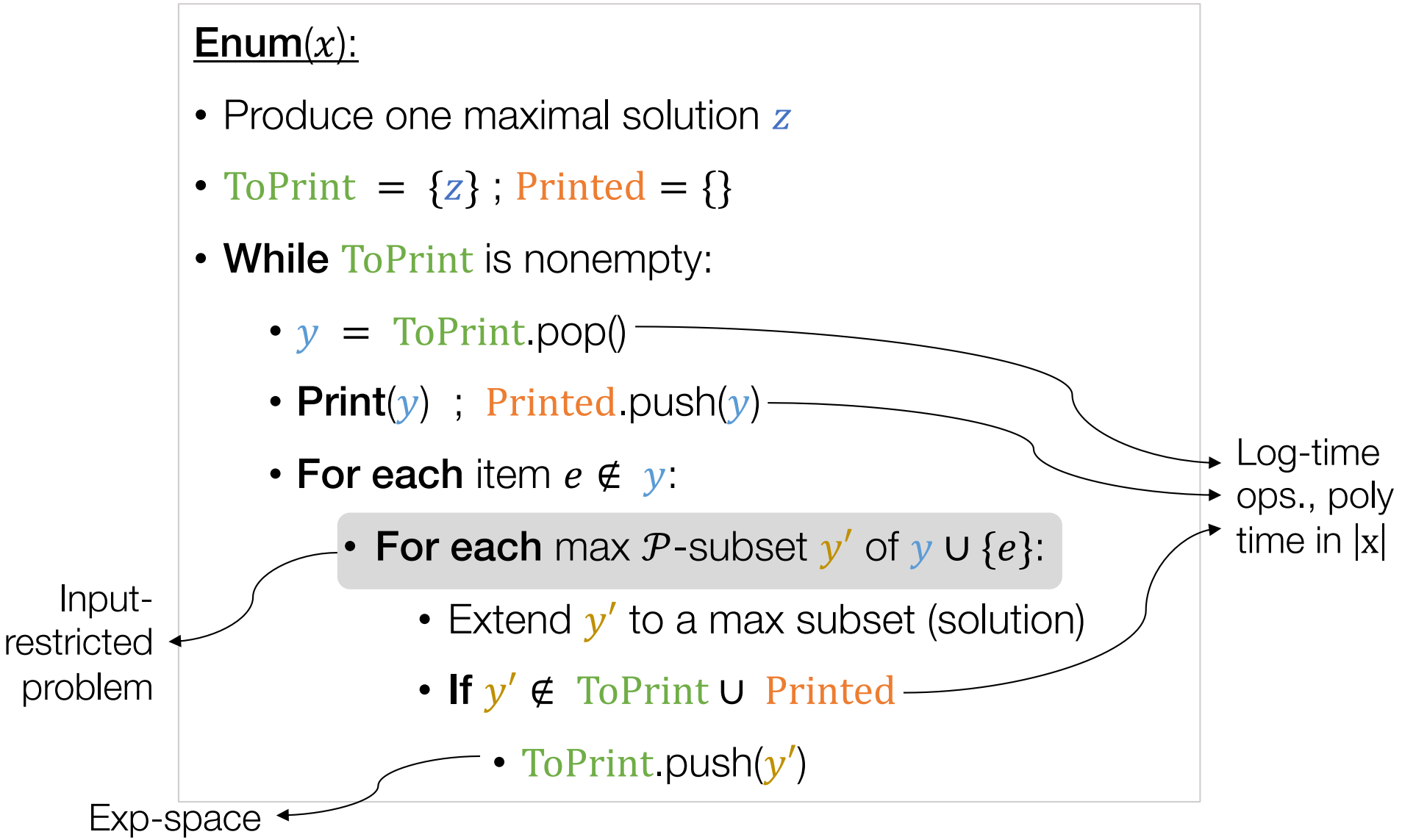
Example: IRP in Maximal Matchings



CONCLUSION:

Maximal matchings enumerable w/ **PDelay**

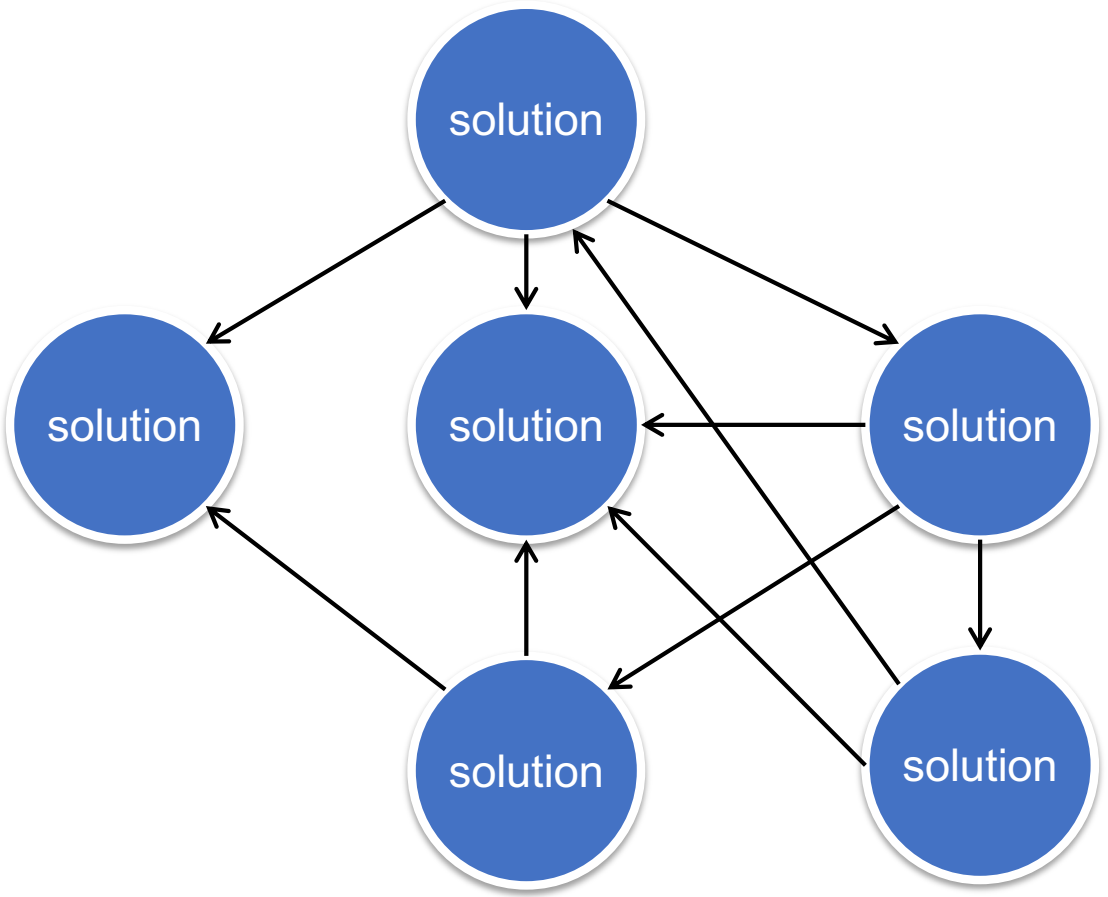
Reduction to the Input-Restricted Problem



[Cohen, K, Sagiv, 2008]

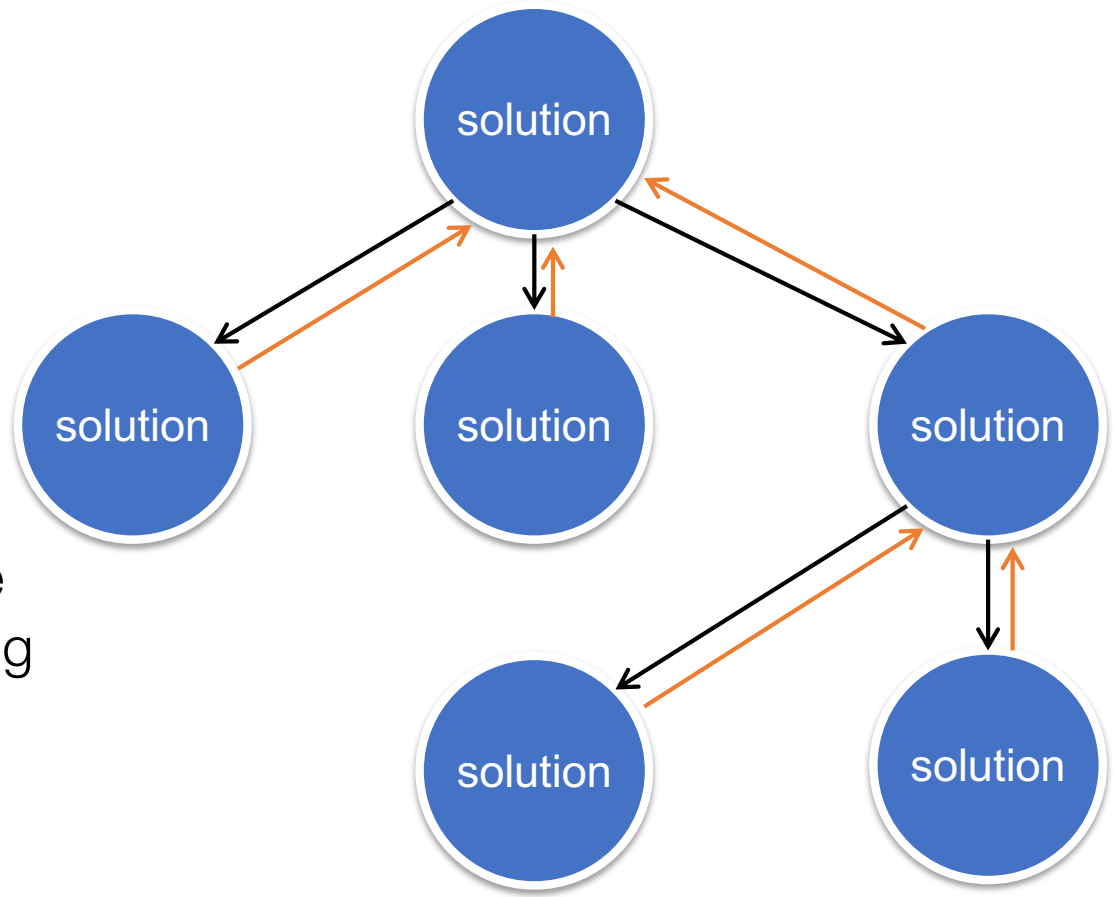
Why Exponential Space?

- Traversal over a graph of all solutions (nodes)
- Nodes & edges produced during enumeration
- Need to avoid reporting **duplicate nodes**
- Need to know when **done**
- Hence, need to **remember all past nodes**



Polynomial Space via Reverse Search

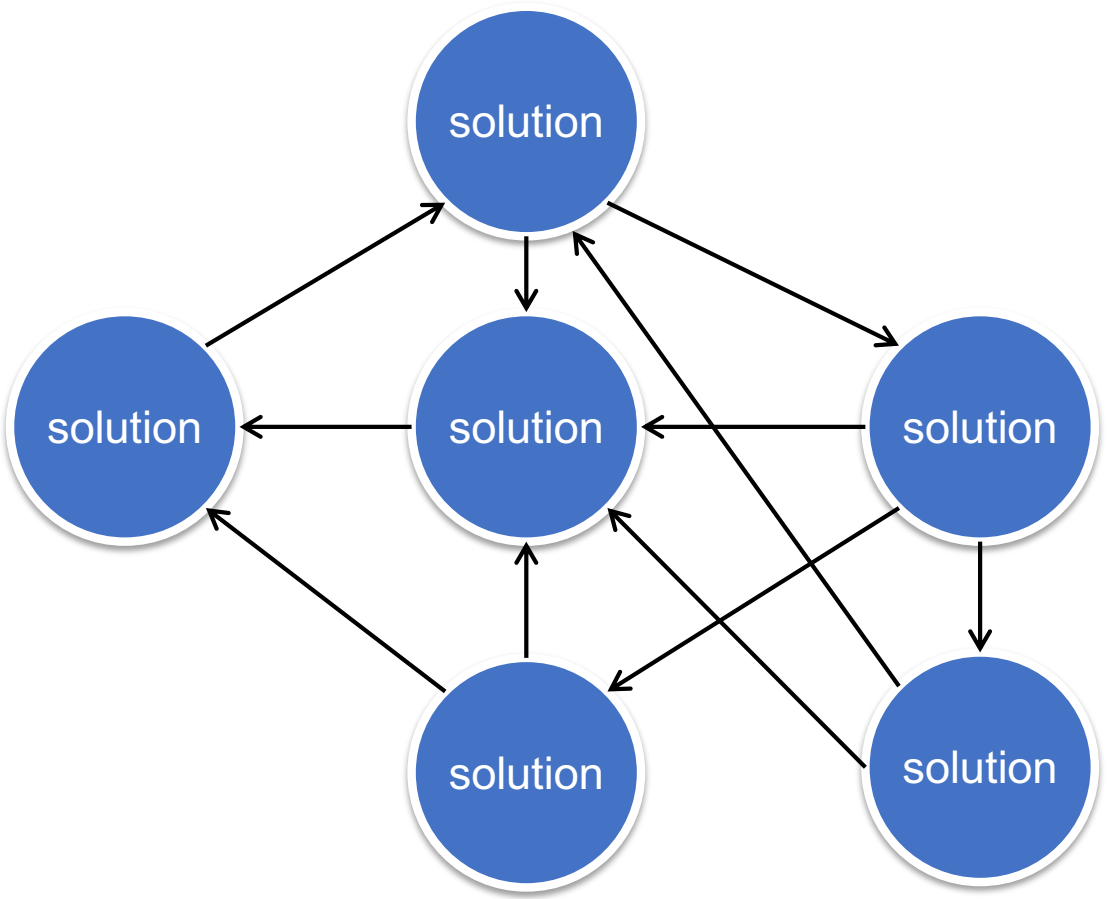
- Solutions organized in an ordered **tree structure**
- Condition: given a solution, efficiently compute:
  - Its sequence of children
  - Its unique parent
- Then – traversal can be done without remembering anything
- Nontrivial condition, uncommon
- Relevant application: generation of candidate keys



[Avis & Fukuda, 1996]

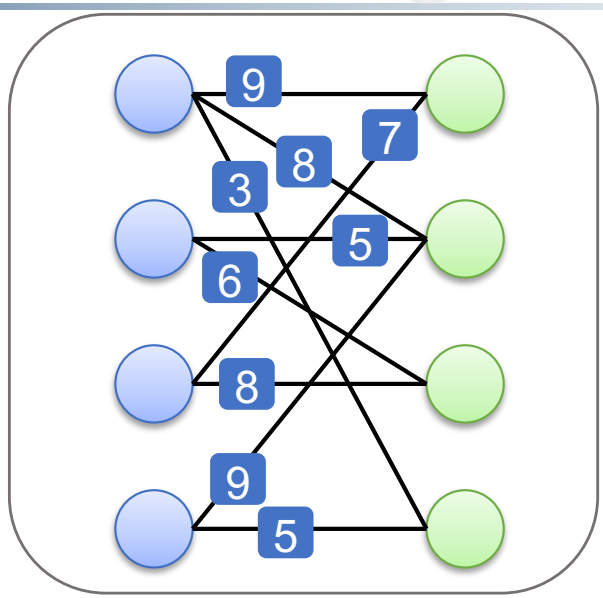
Generalizing IRP: Proximity Search

- General technique for PDelay via a solution graph
- Well beyond hereditary, well beyond IRP
  - Led to **solutions of open problems**
    - Max bipartite graphs
    - Max chordal subgraphs
    - Max induced trees
    - Min triangulations (join plans)
- Idea: edges + **proximity** between solutions
- Condition: If $S_1 \neq S_2$ then S_1 has a neighbor closer to S_2
  - Assures that the solution graph is strongly connected

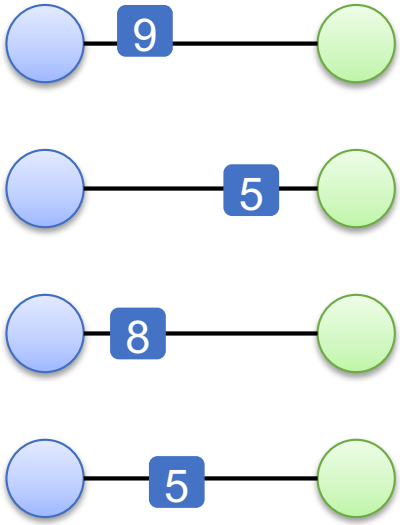


[Conte & Uno, STOC19]

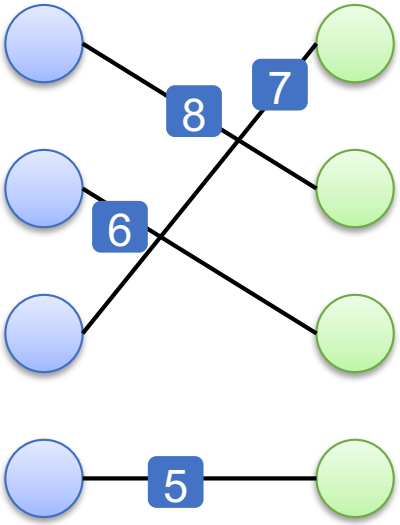
Ranked Perfect Matchings



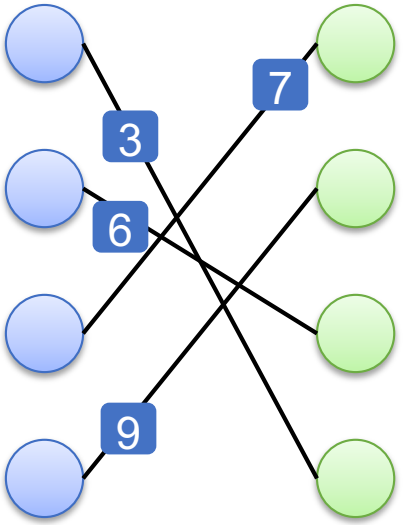
Input



27 Best

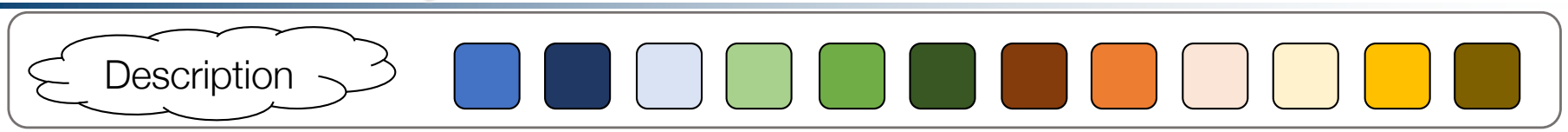


26 2nd best

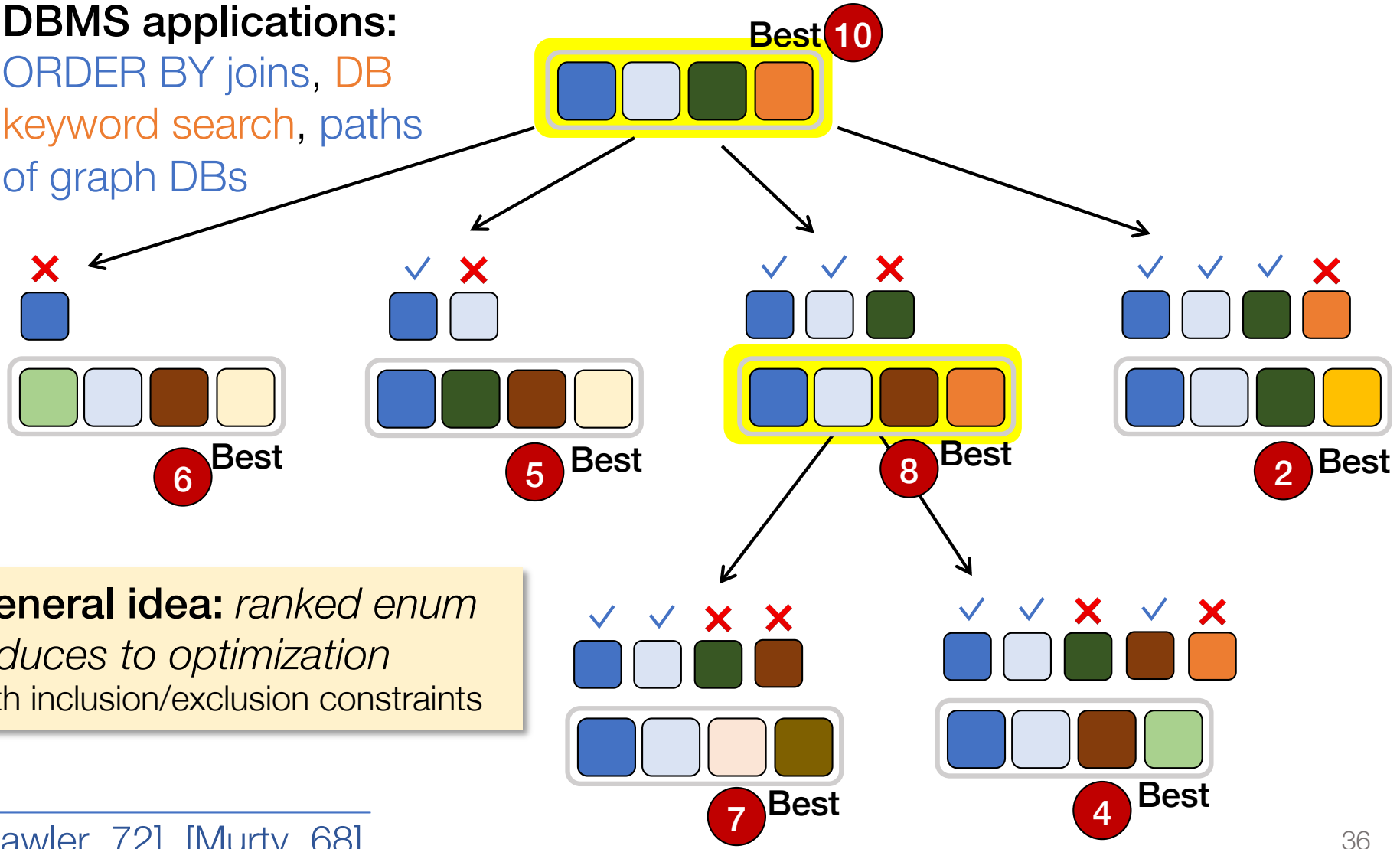


25 3rd best

Lawler-Murty for Ranked Enumeration



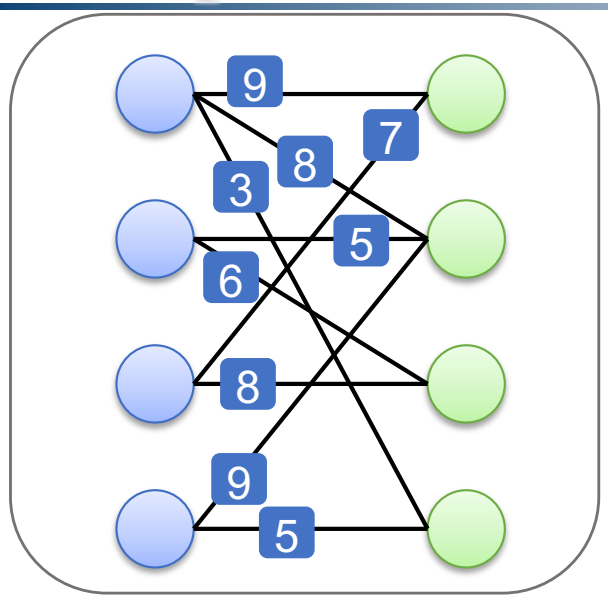
DBMS applications:
ORDER BY joins, DB keyword search, paths of graph DBs



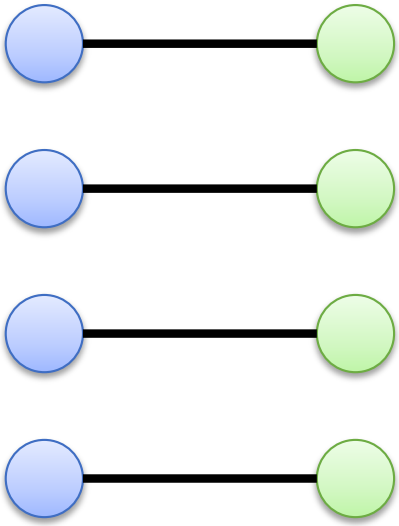
General idea: *ranked enum reduces to optimization* with inclusion/exclusion constraints

[Lawler, 72] [Murty, 68]

Example on Perfect Matchings



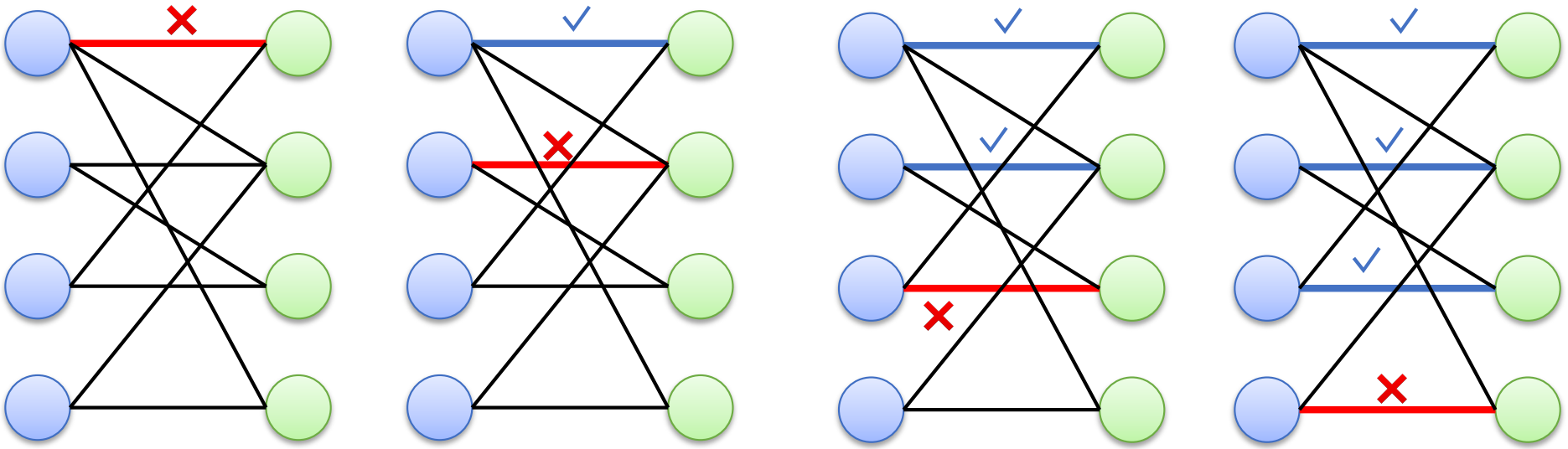
Input



27 Best

CONCLUSION:

Perfect matchings enumerable by dec. weight w/ **PDelay**



Summary of General Enumerators

- **Binary Partition** (a.k.a. Flashlight Search)
  - For simple problems, polynomial delay and space
  - Requires PTime **non-emptiness under inc/exc constraints**
- **Input-Restricted Problem** (IRP)
  - Applicable to max subsets for a hereditary property
  - Requires an algorithm for the **IRP** (PTime for PDelay)
- **Reverse Search**
  - Polynomial delay and **polynomial space**
  - Nontrivial requirement for a solution tree
- **Proximity Search**
  - Allows for a flexible definition of neighbors
  - Led to the resolution of **open problems** via. PDelay algorithms
- **Lawler-Murty**
  - Ranked enumeration with PDelay
  - Reduces to **optimization under inc/exc constraints**

Challenge Slide: Practical Limitations

- Vanilla style, the reusable techniques are often too slow
  - Require heavy engineering and post-optimization
- High practical cost for runtime guarantees
  - Often, naive techniques outperform algorithms with guarantees
    - Albeit the theoretical risk of an exponential gap
- We need a general programming library with effective common optimizations
- We need a better understanding of *when the worst-case scenario might hit*

Plan

Chapter 1:

Many-Solution Problems in Databases

Chapter 2:

Concepts in Enumeration Theory

Chapter 3:

Reusable Algorithmic Techniques

Chapter 4:

The Angle of Query Answering

Ranked Joins in SQL

```
SELECT DISTINCT Rd.Ad , Re.Ae , …
  FROM R1 , R2 , R3 , …
 WHERE Ri.Ai=Rj.Aj AND Rl.Al=Rm.Am AND …
ORDER BY Rf.Af , Rg.Ag , …
```

Complexity Setup for this Part

- Data complexity
  - Fixed query, database given as input
  - Each query defines a separate computational problem
  - (Hence, complexity results are often **dichotomies** – partitioning a space of queries into tractable and intractable classes)
- **Fine-grained complexity**
- Desired complexity: linear+const
  - Linear preproc (DB read); constant delay (tuple write)
  - Polylog(DB) factors allowed: linear*+const*

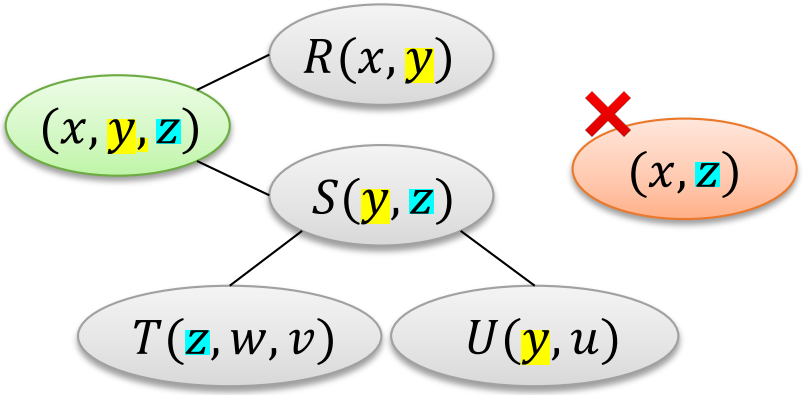
Linear Preprocessing + Constant-Delay

$$R(x,y) \bowtie S(y,z) \bowtie T(z,w,v) \bowtie U(y,u)$$

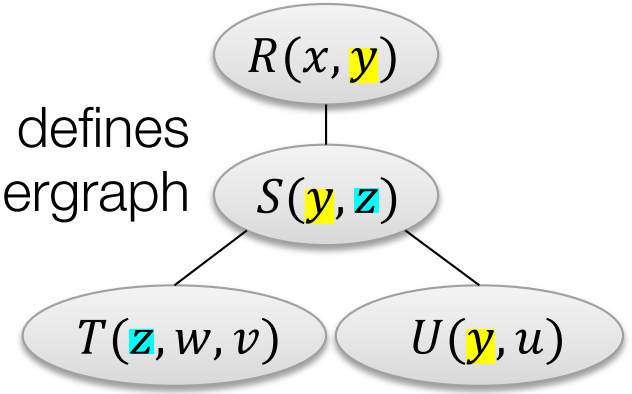
THEOREM [Bagan, Durand, Grandjean 07]

A join admits linear+const iff it is **acyclic**.

(Under assumptions – next slide)



A join is *acyclic* if it defines an acyclic hypergraph



$$\pi_{x,y,z}\left(R(x,y) \bowtie S(y,z) \bowtie T(z,w,v) \bowtie U(y,u)\right)$$

For projection, replace "acyclic" ↔ "**acyclic free-connex**"

– The hypergraph is acyclic even if we add the projection list

Assumptions for the Lower Bounds

- Lower bound assumes conjectures:
  - *Hyperclique detection* for full joins
  - Can be *zero-clique* conjecture [Bringmann, Carmeli, Mengel 25]
  - Boolean matrix multiplication (BMM) for CQs with projection
- Lower bound assumes **no self-joins**
  - Self-joins considerably complicate [Carmeli & Segoufin 23]

Algorithmic Paradigm: Compilation

Same idea as the "factorized reps" and "factorized DBs"

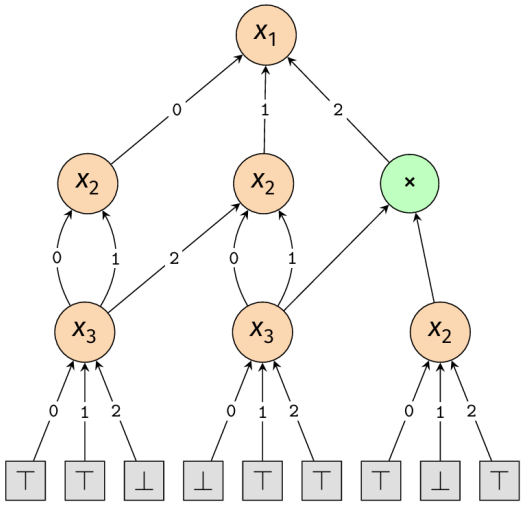
- [Olteanu & Závodný 15]
- [Olteanu& Schleich 16]

Source problem E:
Given D,
compute $Q(D)$

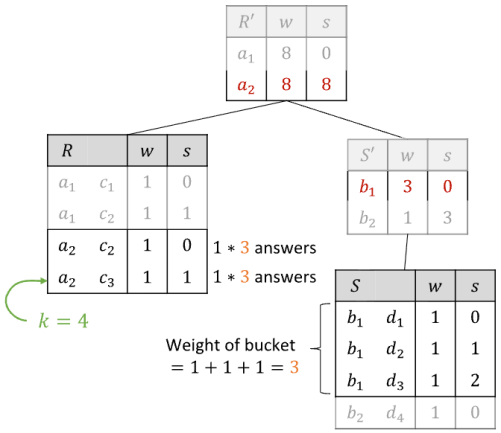
Compilation
in Linear time

Data structure with simple const-delay enumeration, typically tree/circuit

"Happy Structure"



[Capelli & Irwin 24]



[Carmeli+21]

Even Stronger Dichotomy

- When linear+const is infeasible:

  Impossible in linear total time!

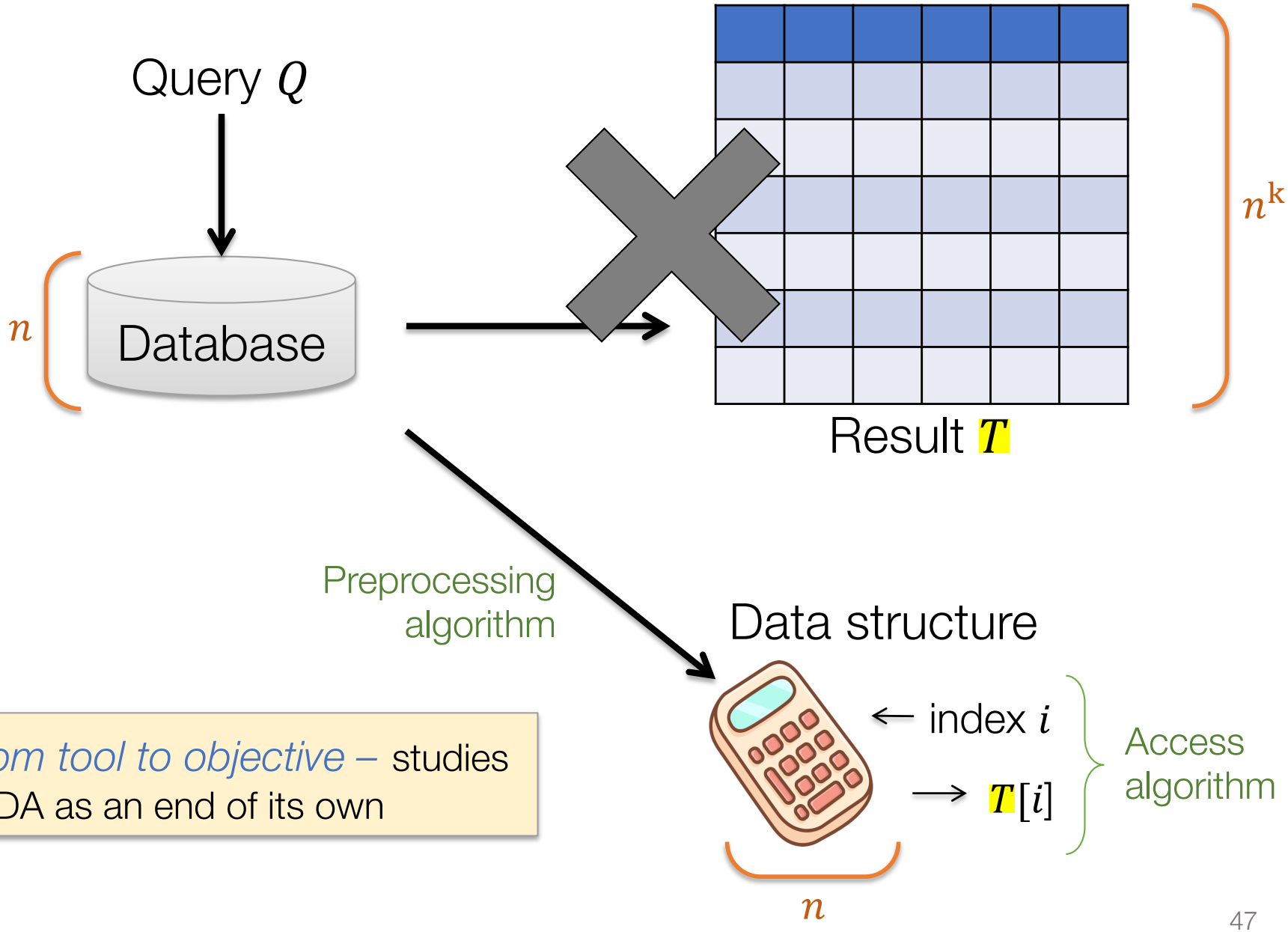
  – This is what the proof actually shows

- When linear+const is possible:

  Even constant-time **Direct Access** is possible!

  – Provided already by the happy structure [Brault-Baron 2013]

Direct Access to Query Answers

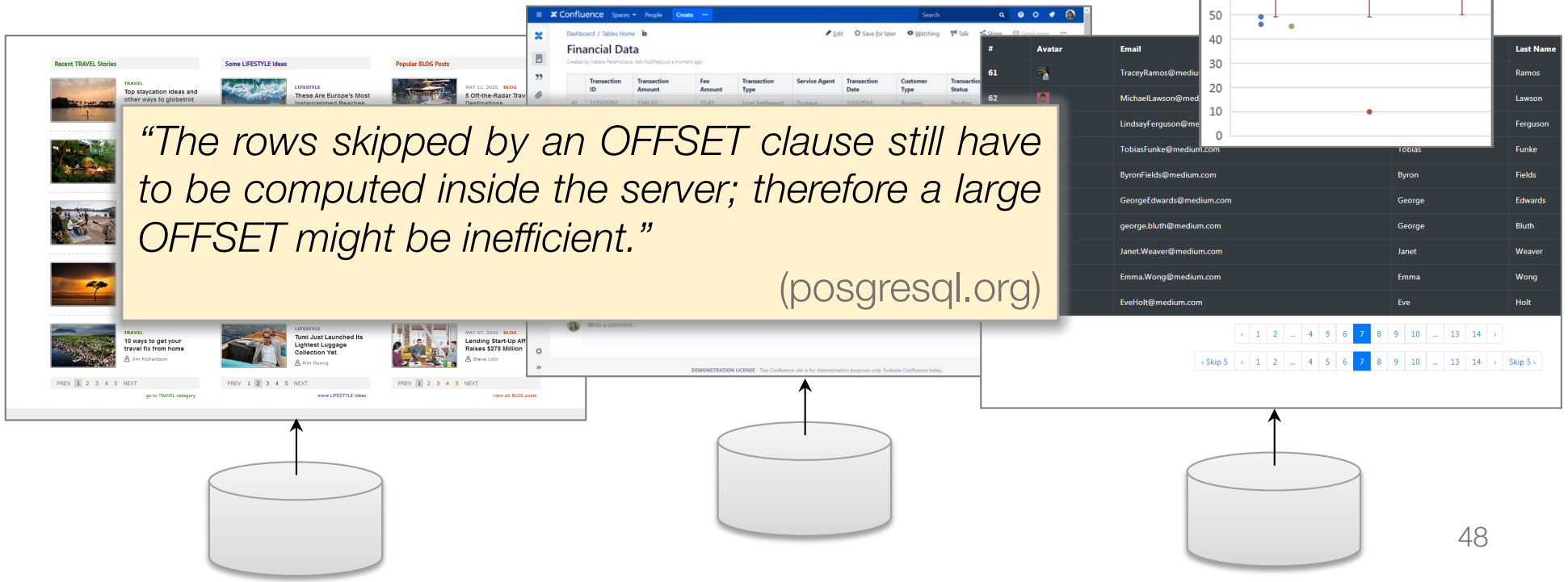


From tool to objective – studies of DA as an end of its own

Example: Pagination Support in SQL

```
SELECT DISTINCT Rd.Ad , Re.Ae , …
  FROM R1 , R2 , R3 , …
 WHERE Ri.Ai=Rj.Aj AND Rl.Al=Rm.Am AND …
ORDER BY Rf.Af , Rg.Ag , …
   LIMIT k  OFFSET m
```

Useful for quantile stats

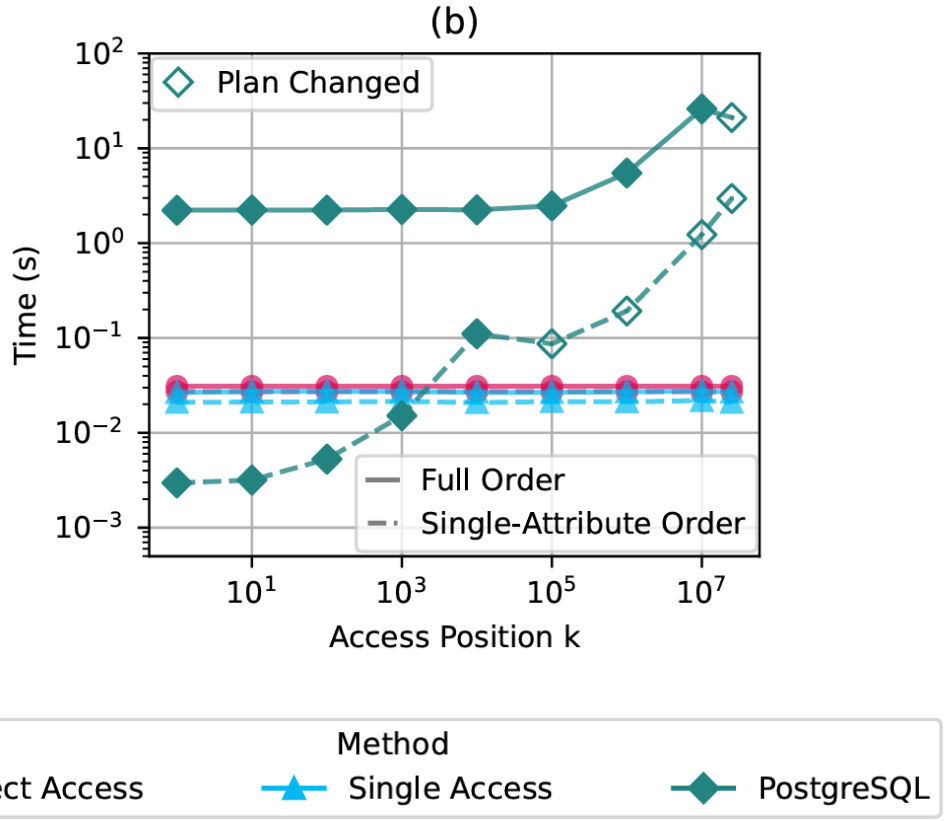


"The rows skipped by an OFFSET clause still have to be computed inside the server; therefore a large OFFSET might be inefficient."

(posgresql.org)

Really? Yes!

3-way join (synthetic data)
[Hu & Tziavelis 26]

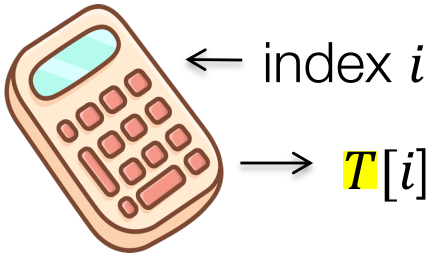


Summary Total Time, Delay, and DA

Linear-time
evaluation

Linear+const
enumeration

Linear+const
DA



Separation Between the Problems

For acyclic free-connex CQs without self joins:

THEOREM [Deep & Koutris 21] [Tziavelis+22]

Answer enumeration in lin*+const* for **every lex order**.

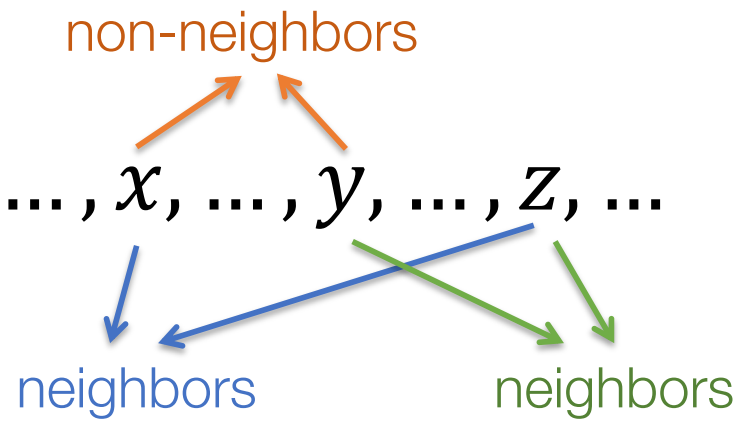
Technique: *Lawler-Murty*

THEOREM [Carmeli+23]

DA in lin*+const* iff the lex order is **reverse elimination**.

Lower bound: *sparse-BMM* and *Hyperclique, no self-joins*

Easy detection:
no disruptive trio



Beyond Linear-Time Preprocessing

- When the CQ is not acyclic, linear time is insufficient for preprocessing
- *So, what is?*
- Characterization for DA + lexicographic orders via hypertree decompositions
  - [Berkholz & Schweikardt 19] [Bringmann, Carmeli, Mengel 25]
- PANDA algorithm [Abo Khamis, Ngo, Suciu 17]
  - Constant-delay with preproc $O(|D|^w)$, w is the *submodular width* of the query
- Ref: SIGMOD Record survey [Tao & Wang 25]

… and much more

- View maintenance (update the happy structure)
  - [Berkholz, Keppeler, Schweikard 17] [Kara+20]
- CQs with self-joins [Carmeli & Segoufin 23]
- CQs with negation [Brault-Baron 12] [Deep & Koutris 25] [Zhao+24] [Capelli & Irwin 24]
- CQs w/ inequalities [Carmeli & Tziavelis 25] [Wang & Yi 26] [Tziavelis, Gatterbauer, Riedewald 21]
- Union of CQs [Carmeli & Kröll 21] [Bringmann & Carmeli 25]
- Aggregation (/ semiring annotation)
  - [Keppeler 20] [Eldar, Carmeli, K 24] [Figueira & Freire 25]

The Double Challenge of Aggregation

$$\pi_{x,y}\left(R(x,y) \bowtie S(y,z) \bowtie T(z,w)\right)$$

$R(x,y)$

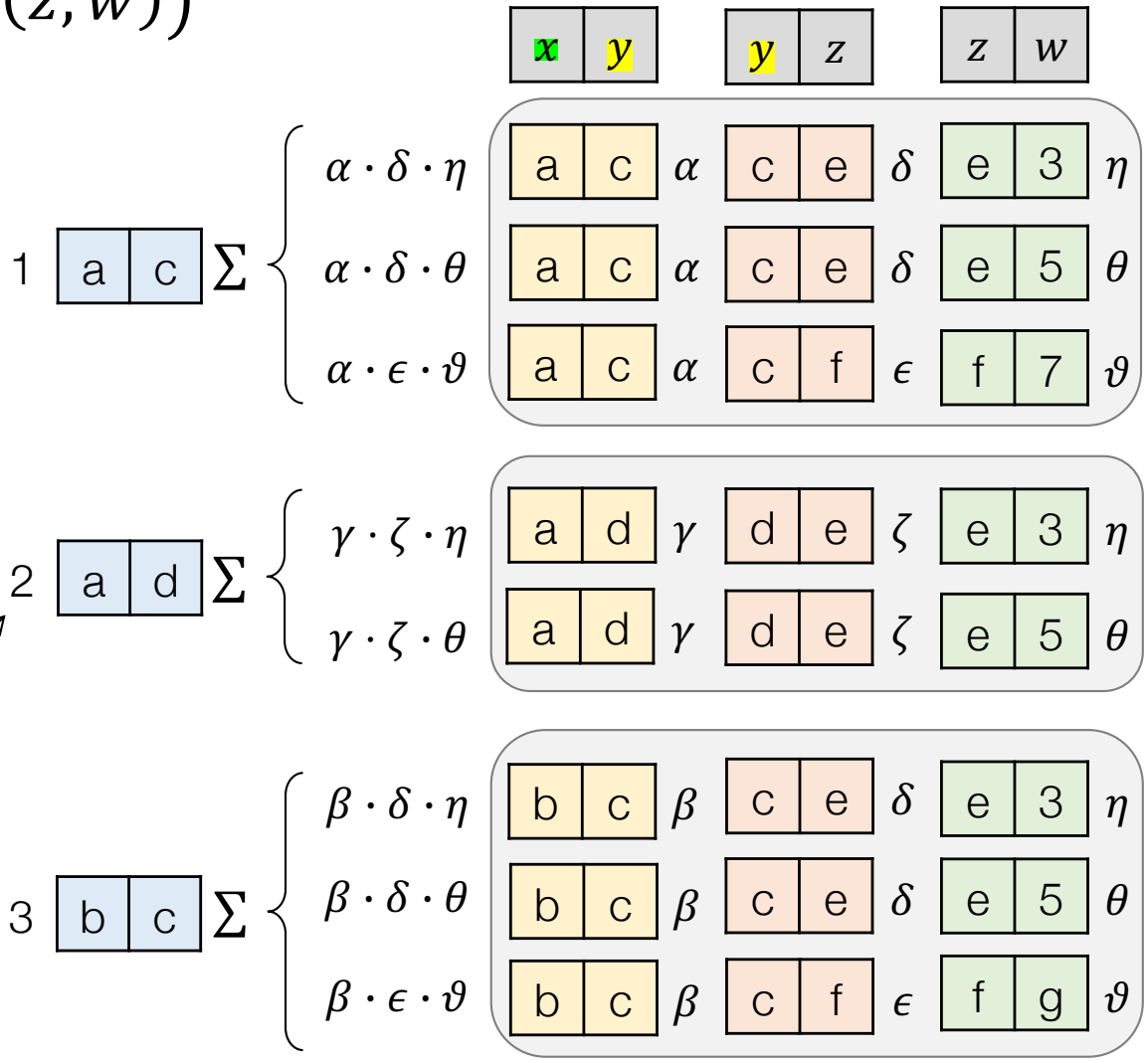
x	y	
a	c	α
b	c	β
a	d	γ

$S(y,z)$

y	z	
c	e	δ
c	f	ϵ
d	e	ζ

$T(z,w)$

z	w	
e	3	η
e	5	θ
f	7	ϑ



Efficient DA: get the ith combination w/o materializing groups and w/o calculating annotations

Challenge Slide: Direct Access in Practice

- Cost preprocessing for DA may outweigh its benefits
  - Albeit the linear-time guarantee, compute-all-and-sort competitive
  - Easy to engineer adversarial cases where theory wins – *can we devise effective query planning to select the tool?*
- Not enough to have DA for limited query fragments – we need a theory of composition (algebra) over DA structures
  - E.g., to compute the intersection of two DA structures, we need to materialize the full sets of answers
  - Insufficient understanding of "selection push down" for DA

Wrap-up

Summary / Takeaways

- Enumeration problems arise throughout DB management
  - Query answering, query optimization, schema design, graph databases, data quality, data mining, search, …
- Well-established complexity theory
  - Preprocessing, delay, ranking, coarse-grained/fine-grained
- Reusable algorithms frequently reused
  - Binary search, input-restricted problem, reverse search, proximity search, Lawler–Murty ranking, …
- Rich landscape of techniques/insights for DB querying
  - Fine-grained dichotomy theorems, structural tractability
  - DA from a means to an aim
- Challenges remain – some pragmatic, some fundamental

Thank you!

& thanks to the Gems-of-PODS Committee

Questions?