

# An Information Theoretic Perspective on Database and Graph Matching

Elza Erkip

Department of ECE, New York University

Padovani Lecture  
NASIT, August 19, 2022



**NYU**

TANDON SCHOOL  
OF ENGINEERING



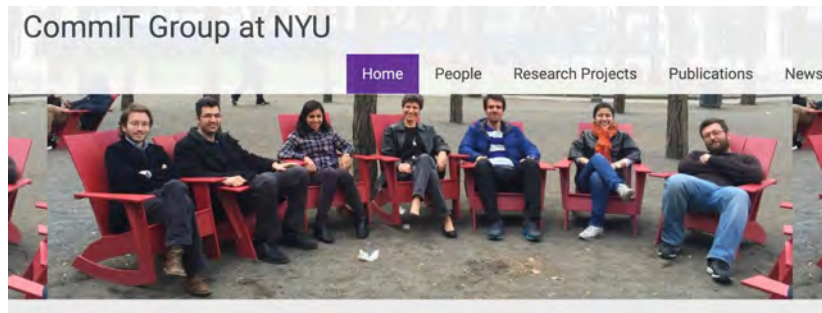
**NYU WIRELESS**

# Background

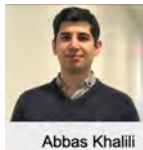
- ECE Department, NYU Tandon School of Engineering.
- Member of NYU WIRELESS.



- Theoretical foundations of networks.
  - Wireless networks.
  - Social networks.

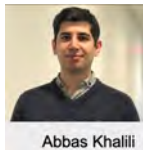


- Power efficiency in next generation wireless networks
  - Theory taking into account practical constraints: Low resolution DAC/ADC, device nonlinearity
  - Capacity, out-of-band emissions, adjacent carrier interference



Abbas Khalili

- Power efficiency in next generation wireless networks
  - Theory taking into account practical constraints: Low resolution DAC/ADC, device nonlinearity
  - Capacity, out-of-band emissions, adjacent carrier interference



- Beam alignment for mmWave and THz
  - A source coding perspective: Delay, error, multiple paths
  - Bounds and algorithms using information theory and theory of group testing



Abbas Khalili



Ozlem Yildiz

- Uncoordinated massive access for IoT
  - Bounds and schemes based on group testing



Jyotish Robin

- Uncoordinated massive access for IoT
  - Bounds and schemes based on group testing



Jyotish Robin

- Machine learning/inference at the wireless edge
  - Compression for hypothesis testing/classification
  - Performance and privacy aspects



Fabrizio Carpi



Kubilay Ulger



Ezgi Ozyilkan

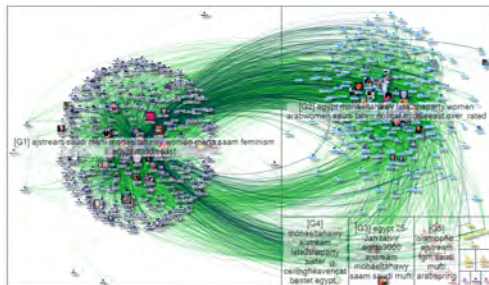
- Foundations of digital privacy
  - Fingerprinting
  - Database matching
  - Graph matching



Serhat Bakirtas



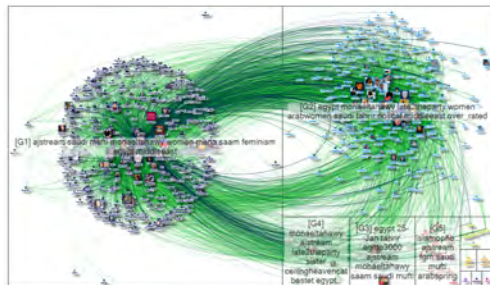
- Social network graph representing user connectivity.



[http://blog.alex-hanna.com/2012/04/  
visualizing-the-polarized-discourse-of-why-do-they-hate-us/](http://blog.alex-hanna.com/2012/04/visualizing-the-polarized-discourse-of-why-do-they-hate-us/)

- Data made available for commercial and research purposes.
  - User identities are removed: *Anonymization*.

- Social network graph representing user connectivity.



<http://blog.alex-hanna.com/2012/04/visualizing-the-polarized-discourse-of-why-do-they-hate-us/>

- Data made available for commercial and research purposes.
  - User identities are removed: *Anonymization*.
- Are anonymized data truly private?

- Social network graph representing user connectivity.
- Data made available for advertisement and research purposes.
  - User identities are removed: *Anonymization*.
- Are anonymized data truly private?

## We Found Joe Biden's Secret Venmo. Here's Why That's A Privacy Nightmare For Everyone.

The peer-to-peer payments app leaves everyone from ordinary people to the most powerful person in the world exposed.



**Ryan Mac**  
BuzzFeed News Reporter



**Katie Notopoulos**  
BuzzFeed News Reporter



**Ryan Brooks**  
BuzzFeed News Reporter



**Logan McDonald**  
BuzzFeed Staff

- Wireless location data can be combined with publicly available information to find user identities.

**The New York Times**

## Your Apps Know Where You Were Last Night, and They're Not Keeping It Secret

Dozens of companies use smartphone locations to help advertisers and even hedge funds. They say it's anonymous, but the data shows how personal it is.

By JENNIFER VALENTINO-DeVRIES, NATASHA SINGER, MICHAEL H. KELLER and AARON KROLIK DEC. 10, 2018

- App developers and data brokers collect and sell sensitive data.

**MOTHERBOARD**  
TECH BY VICE

## Data Broker Is Selling Location Data of People Who Visit Abortion Clinics

It costs just over \$160 to get a week's worth of data on where people who visited Planned Parenthood came from, and where they went afterwards.



By Joseph Cox

May 3, 2022, 12:46pm [Share](#) [Tweet](#) [Send](#)

- Data can be used in targeted attacks.

**MOTHERBOARD**  
TECH BY VICE

## The Inevitable Weaponization of App Data Is Here

A Substack publication used location data from Grindr to out a priest without their consent.



By Joseph Cox

July 21, 2021, 12:10pm  Share  Tweet  Grid

- How do researchers think about de-anonymization?

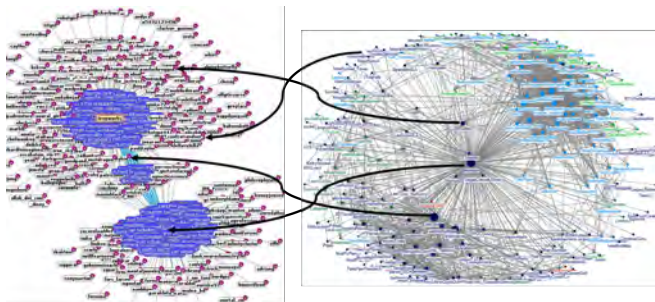
# Social Network De-anonymization

- How do researchers think about de-anonymization?
- Basic set-up: Two (or more) anonymized social networks.
- Objective: Match the users in the two networks.



# Social Network De-anonymization

- How do researchers think about de-anonymization?
- Basic set-up: Two (or more) anonymized social networks.
- Objective: Match the users in the two networks.
- Example: Twitter, Flickr and LiveJournal successfully de-anonymized.  
[Narayanan and Shmatikov, '09]



LiveJournal

[<http://www.intrio.com/ACC2005/>]

Twitter

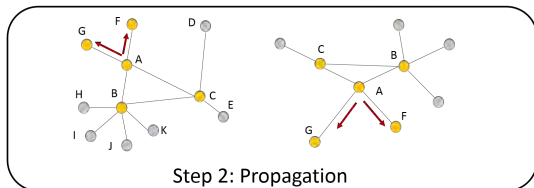
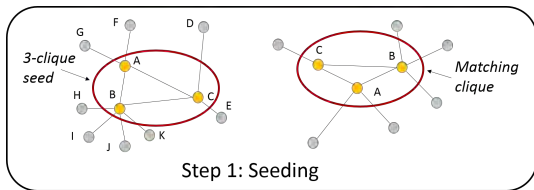
[<http://eppsnets.com/2012/10/visualizing-social-networks/>]

# Algorithm for De-anonymization

- “Seeding” and “propagation” algorithm.  
[Backstrom et al., '07; Narayanan and Shmatikov, '09]

# Algorithm for De-anonymization

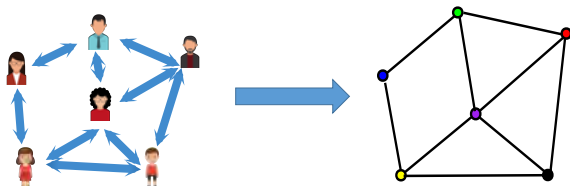
- “Seeding” and “propagation” algorithm.  
[Backstrom et al., '07; Narayanan and Shmatikov, '09]
  - First determine *seed* nodes in both graphs.
    - Identify and match  $k$ -cliques in both graphs.
  - Iteratively propagate seed mapping to new nodes.



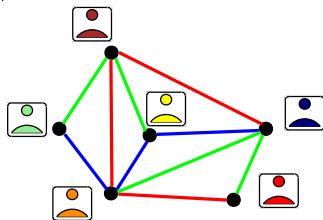
- Social networks and call logs can be modeled using graphs.
- Network graph represents user connectivity.

# Social Network Modeling

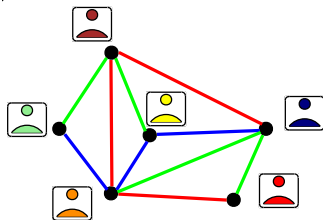
- Social networks and call logs can be modeled using graphs.
- Network graph represents user connectivity.



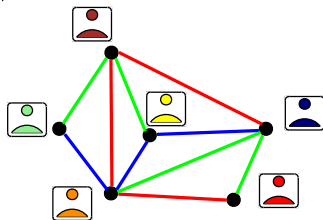
- **Vertices:** Members of the network.
- **Edges:** Friendship relations.
- **Labels (Colors):** Identity of the member.



- Edges may have attributes (non-binary edges).
- Example:
  - 1 Social networks: Close friends, acquaintances, followers, etc..



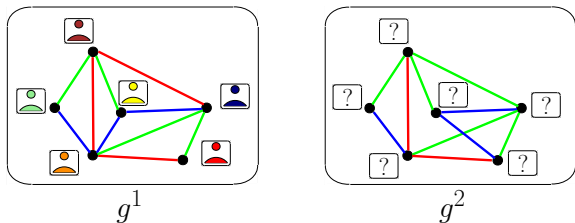
- Edges may have attributes (non-binary edges).
- Example:
  - 1 Social networks: Close friends, acquaintances, followers, etc..
  - 2 Call log networks: Frequent calls, call times, call lengths, etc..



- Edges may have attributes (non-binary edges).
- Example:
  - 1 Social networks: Close friends, acquaintances, followers, etc..
  - 2 Call log networks: Frequent calls, call times, call lengths, etc..
- Attributes modeled by assigning colors (numbers) to edges.

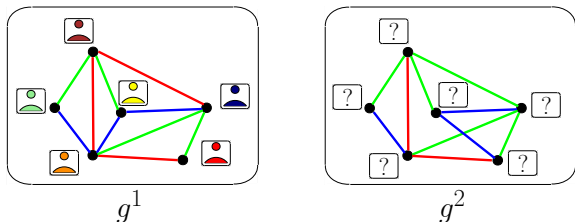


# Network De-anonymization Attacks



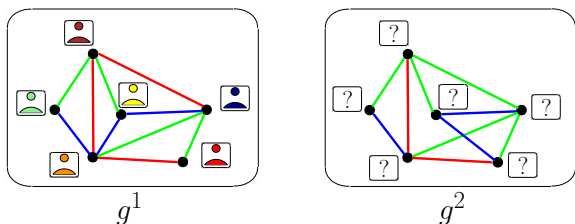
- We are given pair of social networks.

# Network De-anonymization Attacks



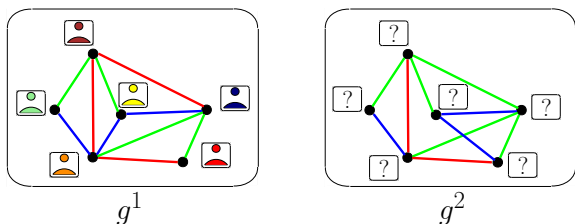
- We are given pair of social networks.
- The first network is de-anonymized (public identities).

# Network De-anonymization Attacks



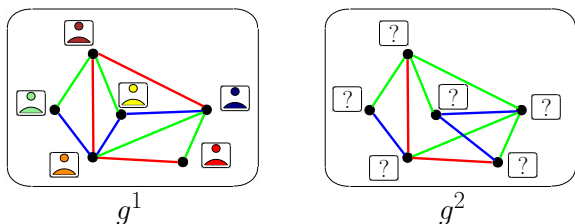
- We are given pair of social networks.
- The first network is de-anonymized (public identities).
- The second network is anonymized (private identities).

# Network De-anonymization Attacks



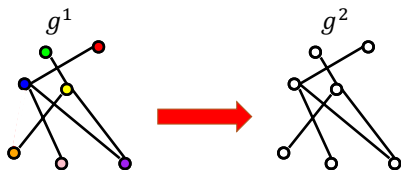
- We are given pair of social networks.
- The first network is de-anonymized (public identities).
- The second network is anonymized (private identities).
- Objective: De-anonymize second network.

# Network De-anonymization Attacks



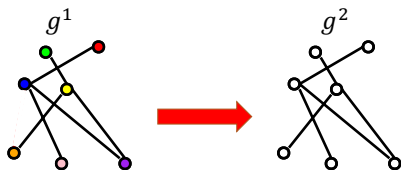
- We are given pair of social networks.
- The first network is de-anonymized (public identities).
- The second network is anonymized (private identities).
- Objective: De-anonymize second network.
- Graph matching.

# Graph Matching



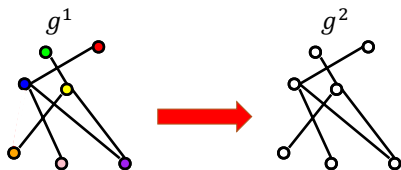
- Two identical graphs  $g^1$  and  $g^2$ .
- $g^1$  has vertex labels,  $g^2$  doesn't.

# Graph Matching



- Two identical graphs  $g^1$  and  $g^2$ .
- $g^1$  has vertex labels,  $g^2$  doesn't.
- How to restore the labels for  $g^2$ ?

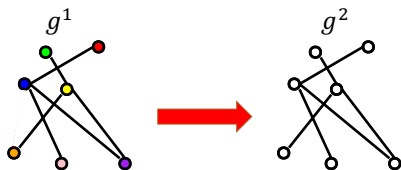
# Graph Matching



- Two identical graphs  $g^1$  and  $g^2$ .
- $g^1$  has vertex labels,  $g^2$  doesn't.
- How to restore the labels for  $g^2$ ?
- What if the graphs are not the same, but **correlated**?

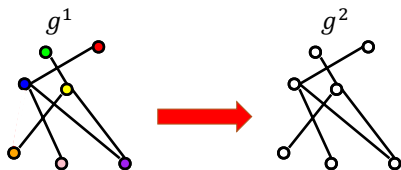


# Graph Matching



- Two identical graphs  $g^1$  and  $g^2$ .
- $g^1$  has vertex labels,  $g^2$  doesn't.
- How to restore the labels for  $g^2$ ?
- What if the graphs are not the same, but **correlated**?
- What if we have **seeds**, vertices in  $g^2$  whose labels are known?

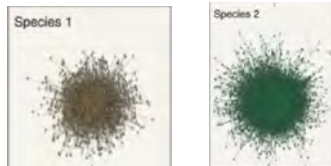
# Graph Matching



- Two identical graphs  $g^1$  and  $g^2$ .
- $g^1$  has vertex labels,  $g^2$  doesn't.
- How to restore the labels for  $g^2$ ?
- What if the graphs are not the same, but **correlated**?
- What if we have **seeds**, vertices in  $g^2$  whose labels are known?
- What if the edges are **multi-valued**?

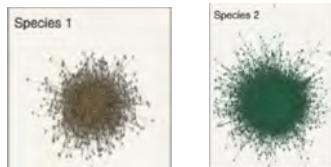
# Other Applications of Graph Matching

- Biology: Match protein interaction networks of species.

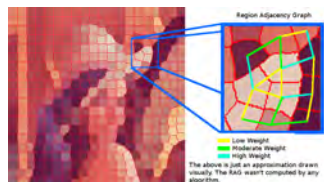


# Other Applications of Graph Matching

- Biology: Match protein interaction networks of species.



- Image classification: Match image segmentation graphs.



# Database Matching Attacks













- A simpler class of matching attacks: [Databases](#).

# Database Matching Attacks

- A simpler class of matching attacks: [Databases](#).
- Databases containing *micro-information* shared and published routinely.
- Examples: Movie preferences, financial transactions data, location data, health records.

# Database Matching Attacks

- A simpler class of matching attacks: **Databases**.
- Databases containing *micro-information* shared and published routinely.
- Examples: Movie preferences, financial transactions data, location data, health records.
- Basic privacy methods: i) anonymization, ii) obfuscation.

Rotten Tomatoes				Netflix				
	Pulp Fiction	Godfather	Fight Club		Pulp Fiction	Godfather	Fight Club	Forrest Gump
	*40	*60	●98		★4.5	★4.5	★5	★4.0
	●88	●80	*66		★4.5	★4.5	★4.5	★4
	●92	●90	●91		★4.0	★4.5	★4	★4.5
	●90	●85	*65		★3.5	★3.5	★4.0	★4
	●87	●83	●79		★4.5	★5	★2.5	★4.5
	*64	●86	*70		★4.5	★4.5	★2.5	★5

# Matching Movie Databases

- Netflix Prize database: Users' movie ratings to develop better recommender systems.
  - Late 2000's, 1 M Prize.



# Matching Movie Databases

- Netflix Prize database: Users' movie ratings to develop better recommender systems.
  - Late 2000's, 1 M Prize.
- User identities anonymized to address privacy concerns.

# Matching Movie Databases

- Netflix Prize database: Users' movie ratings to develop better recommender systems.
  - Late 2000's, 1 M Prize.
- User identities anonymized to address privacy concerns.
- Simply anonymizing users is *not* sufficient.

# Matching Movie Databases

- Netflix Prize database: Users' movie ratings to develop better recommender systems.
  - Late 2000's, 1 M Prize.
- User identities anonymized to address privacy concerns.
- Simply anonymizing users is *not* sufficient.
- De-anonymization using IMDB data  
[Narayanan and Shmatikov 2008], 2019 "Test of Time" Award from the IEEE Symposium on Security and Privacy.

	Movie 1	Movie 2 ....	Movie M
User 1	★★		
User 2			★★★★
User N		★	★★★



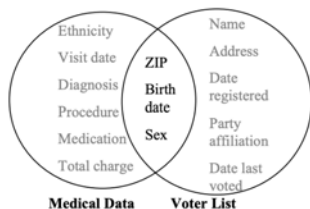
- *Database 1*: Person-specific, field-collected **medical records** given to the Group Insurance Commission.
  - User identities anonymized to address privacy concerns.

# De-anonymizing Medical Databases

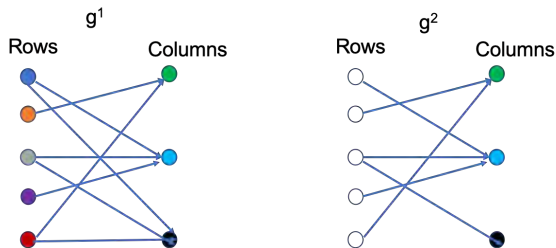
- *Database 1*: Person-specific, field-collected **medical records** given to the Group Insurance Commission.
  - User identities anonymized to address privacy concerns.
- *Database 2*: Vote registration data (includes names and addresses).
  - Database can be purchased for less than \$100.

# De-anonymizing Medical Databases

- *Database 1*: Person-specific, field-collected **medical records** given to the Group Insurance Commission.
  - User identities anonymized to address privacy concerns.
- *Database 2*: Vote registration data (includes names and addresses).
  - Database can be purchased for less than \$100.
- Matching the two databases leads to de-anonymization. [Sweeney 2002]

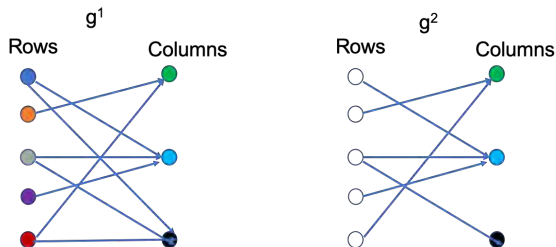


# Database Matching as Graph Matching



- Database matching equivalent to matching rows of multiple matrices.

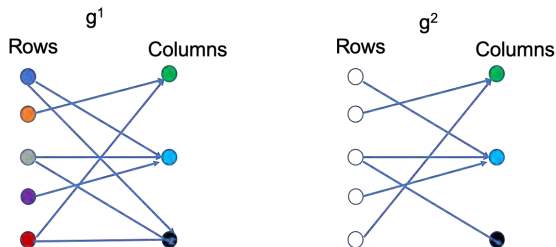
# Database Matching as Graph Matching



- Database matching equivalent to matching rows of multiple matrices.
- Also: Bi-partite graphs with multi-valued edges.
  - Left nodes: Row indices.
  - Right nodes: Column indices.



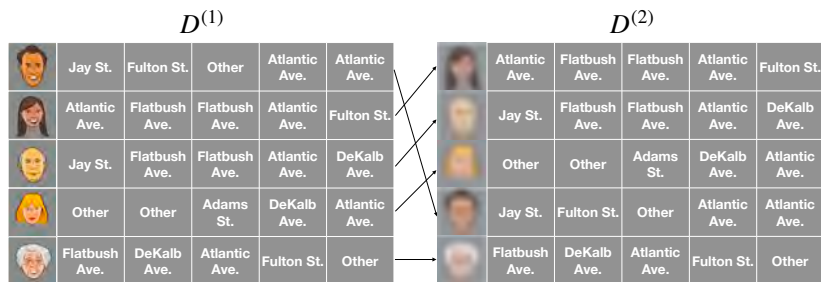
# Database Matching as Graph Matching



- Database matching equivalent to matching rows of multiple matrices.
- Also: Bi-partite graphs with multi-valued edges.
  - Left nodes: Row indices.
  - Right nodes: Column indices.
- Row labels for  $g^2$  missing.

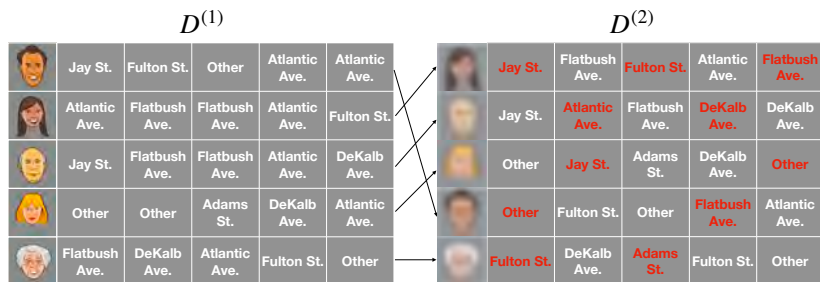
# Database Matching

- Two databases  $D^{(1)}$  and  $D^{(2)}$  (matrices) with equal number of users (rows).
- $D^{(1)}$  has row labels,  $D^{(2)}$  doesn't due to *anonymization*.
- How to restore the labels for  $D^{(2)}$ ?



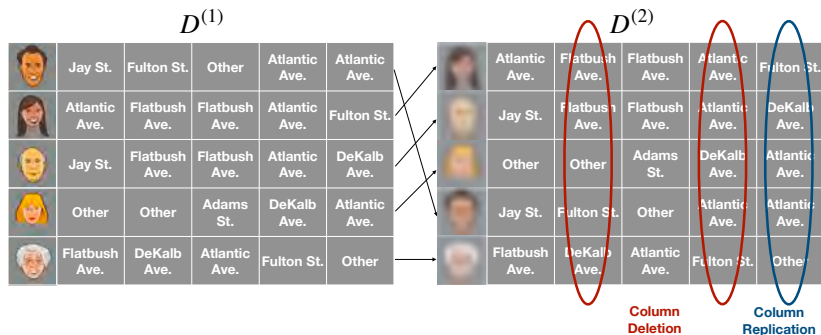
# Database Matching

- What if the databases are not the same, but **correlated** due to *noise* and *obfuscation*?



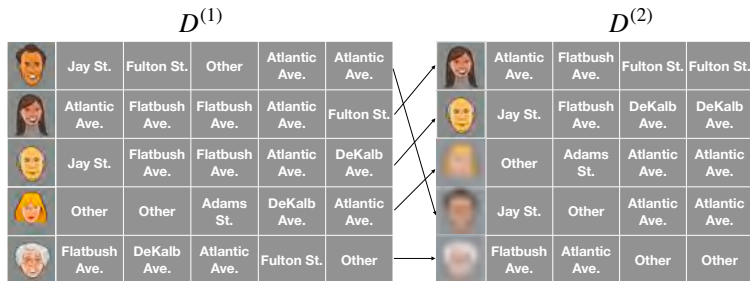
# Database Matching

- What if some attributes (columns) are **deleted** or **repeated** due to synchronization errors?



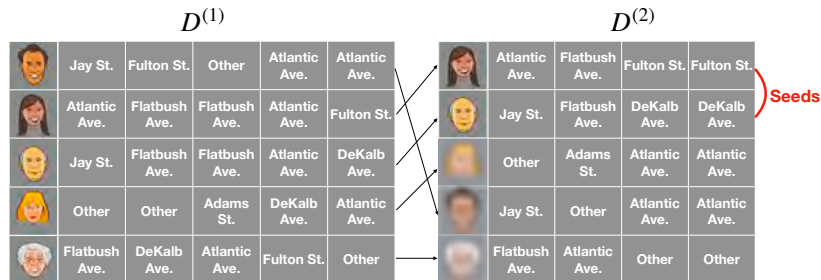
# Database Matching

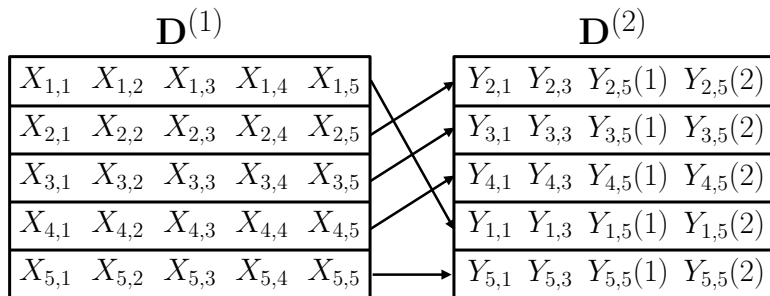
- What if some attributes (columns) are **deleted** or **repeated** due to synchronization errors?



# Database Matching

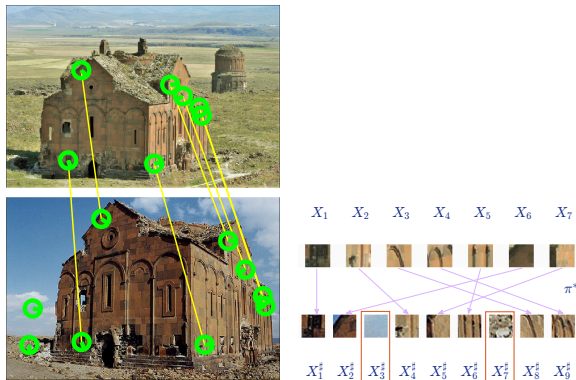
- What if we have **seeds**, rows in  $D^{(2)}$  whose labels are known?





# Other Applications of Database Matching

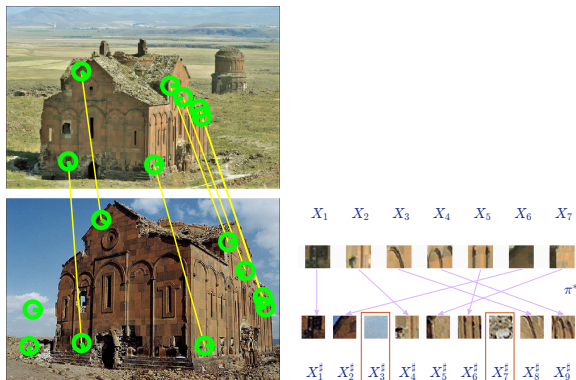
- Computer vision [Galstyan et al., 2021]





# Other Applications of Database Matching

- Computer vision [Galstyan et al., 2021]



- Biological applications
  - DNA Sequencing [Błażewicz et al., 2002]
  - Single-cell data alignment [Chen et al., 2022]

- Popular press on compromising social network privacy.

- Popular press on compromising social network privacy.
- Literature on social network and database de-anonymization attacks.
  - Lacks bounds and guarantees.

- Popular press on compromising social network privacy.
- Literature on social network and database de-anonymization attacks.
  - Lacks bounds and guarantees.
- Abstraction: Graph and database matching.

- Popular press on compromising social network privacy.
- Literature on social network and database de-anonymization attacks.
  - Lacks bounds and guarantees.
- Abstraction: Graph and database matching.
- Applications beyond privacy.

- How can information theory help in database and graph matching?

- How can information theory help in database and graph matching?
- Consider random databases/graphs.
  - Allows using tools from **information theory**, leading to **theoretical guarantees** and **new algorithms**.

- How can information theory help in database and graph matching?
- Consider random databases/graphs.
  - Allows using tools from **information theory**, leading to **theoretical guarantees** and **new algorithms**.
- Database matching.
  - Correlated databases.
  - Time series: Deletions and replicas.
    - Noiseless.
    - Noisy.



- How can information theory help in database and graph matching?
- Consider random databases/graphs.
  - Allows using tools from **information theory**, leading to **theoretical guarantees** and **new algorithms**.
- Database matching.
  - Correlated databases.
  - Time series: Deletions and replicas.
    - Noiseless.
    - Noisy.
- General graph matching.
  - Matching identical graphs.
  - Correlated graphs with seeds.
  - Correlated graphs without seeds.
    - New typicality results on permutations of sequences.

- Different notions of privacy.
  - Privacy-preserving data analytics.
    - Differential privacy.

- Different notions of privacy.
  - Privacy-preserving data analytics.
    - Differential privacy.
  - **Privacy-preserving microdata publishing.**

- Different notions of privacy.
  - Privacy-preserving data analytics.
    - Differential privacy.
    - **Privacy-preserving microdata publishing.**
- Overview of typicality and asymptotic equipartition property (AEP).

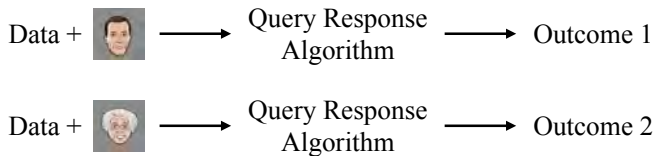
- Different notions of privacy.
  - Privacy-preserving data analysis.
    - Differential privacy.
  - Privacy-preserving microdata publishing.
- Overview of typicality and asymptotic equipartition property (AEP).

## ① Non-Interactive

- Perform statistical task centrally over the dataset and publish the statistics

## ② **Interactive**

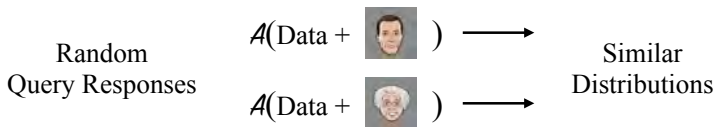
- Adaptive queries applied to database
- Query results may be noisy
- Noise amount is often configurable
- Differential privacy (DP) and its variants



- The two outcomes should be close.

Image: Courtesy of A. Sarwate.

- **Goal:** Any small change individual data causes a small change in the output distribution of algorithm  $\mathcal{A}$ .





[Dwork et al., 2006]

- **Goal:** Any small change in individual data causes a small change in the output distribution of algorithm  $\mathcal{A}$ .

## $\epsilon$ -Differential Privacy

An algorithm  $\mathcal{A}$  achieves  $\epsilon$ -DP if for any user  $u$  in any dataset  $\mathcal{D}$  and any subset  $\mathcal{S} \subseteq \text{Range}(\mathcal{A})$ , we have

$$\Pr(\mathcal{A}(\mathcal{D}) \in \mathcal{S}) \leq e^\epsilon \Pr(\mathcal{A}(\mathcal{D} \setminus u) \in \mathcal{S})$$

- A property of a **randomized** query-response algorithm  $\mathcal{A}$  **at the aggregator/server side.**

- A property of a **randomized** query-response algorithm  $\mathcal{A}$  **at the aggregator/server side.**
- A centralized notion.
- $\epsilon$ -DP is a worst case property. It holds for
  - **any user**  $u$  in **any database**  $\mathcal{D}$ .
  - **any outcome** of  $\mathcal{A}$  no matter how unlikely it might be.

- A property of a **randomized** query-response algorithm  $\mathcal{A}$  **at the aggregator/server side.**
- A centralized notion.
- $\epsilon$ -DP is a worst case property. It holds for
  - **any user**  $u$  in **any database**  $\mathcal{D}$ .
  - **any outcome** of  $\mathcal{A}$  no matter how unlikely it might be.
- **In practice:** Some unlikely outcomes break  $\epsilon$ -DP.
- A common relaxation is  $(\epsilon, \delta)$ -DP.

[Dwork et al., 2006]

- **Goal:** A small change in individual data causes a small change in the output distribution of algorithm  $\mathcal{A}$  **with high probability.**

[Dwork et al., 2006]

- **Goal:** A small change in individual data causes a small change in the output distribution of algorithm  $\mathcal{A}$  **with high probability.**

## $(\epsilon, \delta)$ -Differential Privacy

Algorithm  $\mathcal{A}$  achieves  $(\epsilon, \delta)$ -DP if for any user  $u$  in any dataset  $\mathcal{D}$  and any  $\mathcal{S} \subseteq \text{Range}(\mathcal{A})$ , we have

$$\Pr(\mathcal{A}(\mathcal{D}) \in \mathcal{S}) \leq e^\epsilon \Pr(\mathcal{A}(\mathcal{D} \setminus u) \in \mathcal{S}) + \delta$$

[Kasiviswanathan et al., 2008]

- $\epsilon$ -DP assumes a trustworthy curator/aggregator.
  - May not hold in practice.

[Kasiviswanathan et al., 2008]

- $\epsilon$ -DP assumes a trustworthy curator/aggregator.
  - May not hold in practice.
- A well-established extension is  $\epsilon$ -Local DP (LDP).
- Algorithm  $\mathcal{A}$  is run locally **at the user side**.



# $\epsilon$ -Local Differential Privacy (LDP)

[Kasiviswanathan et al., 2008]

- $\epsilon$ -DP assumes a trustworthy curator/aggregator.
  - May not hold in practice.
- A well-established extension is  $\epsilon$ -Local DP (LDP).
- Algorithm  $\mathcal{A}$  is run locally **at the user side**.

## $\epsilon$ -Local Differential Privacy

Algorithm  $\mathcal{A}$  achieves  $\epsilon$ -LDP if for any input pair  $(x, x')$ , and any outcome  $y \in \text{Range}(\mathcal{A})$ , we have

$$\frac{\Pr(\mathcal{A}(x) = y)}{\Pr(\mathcal{A}(x') = y)} \leq \exp(\epsilon)$$

- Implemented and used by Google and Apple.

# $\epsilon$ -Local Differential Privacy

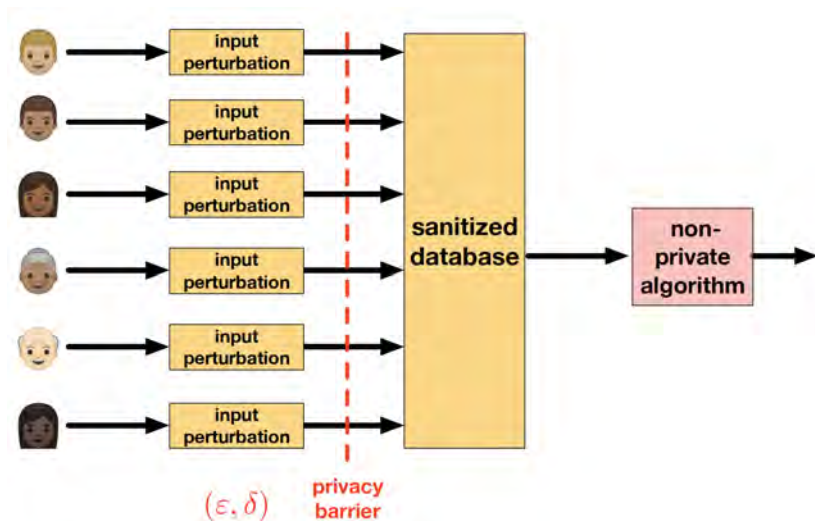


Image: Courtesy of A. Sarwate.

- Different notions of privacy.
  - Privacy-preserving data analysis.
    - Differential privacy.
  - Privacy-preserving microdata publishing.
- Overview of typicality and asymptotic equipartition property (AEP).

- In many settings, *microdata*, raw data at personal level, is available.
- In addition, users have limited control over the data shared.

- In many settings, *microdata*, raw data at personal level, is available.
- In addition, users have limited control over the data shared.
  - Publicly-available government data
    - Voter databases, census data.
  - Data mining or algorithm development contests
    - Netflix Prize.
  - Data collected by apps, traded across companies.
  - Data shared for scientific research.

- Explicit identifiers (EIDs)
  - Name, date of birth, SSN, etc.
  - Compromise user privacy.

- Explicit identifiers (EIDs)
  - Name, date of birth, SSN, etc.
  - Compromise user privacy.
- Quasi identifiers (QIDs)
  - ZIP code, age, race, ethnicity, religion, sex etc.
  - Can also leak information.
  - Assumed to be available publicly.

- Explicit identifiers (EIDs)
  - Name, date of birth, SSN, etc.
  - Compromise user privacy.
- Quasi identifiers (QIDs)
  - ZIP code, age, race, ethnicity, religion, sex etc.
  - Can also leak information.
  - Assumed to be available publicly.
- Sensitive attributes (SAs)
  - Disease, medical tests, salaries etc.
  - Ideally not made public.



- Explicit identifiers (EIDs)
  - Name, date of birth, SSN, etc.
  - Compromise user privacy.
- Quasi identifiers (QIDs)
  - ZIP code, age, race, ethnicity, religion, sex etc.
  - Can also leak information.
  - Assumed to be available publicly.
- Sensitive attributes (SAs)
  - Disease, medical tests, salaries etc.
  - Ideally not made public.
- For privacy, data is usually published/sold after
  - Anonymization/sanitization.
  - Generalization/resolution reduction.

# Anonymization/Sanitization

- Removal of EIDs from data.

User ID	Age (Public)	Registered Voter? (Public)	Borough (Public)	Disease (Private)
	27	No	Manhattan	Diabetes
	30	Yes	Queens	Cancer
	23	No	Manhattan	COVID-19
	24	Yes	Brooklyn	Viral Infection
	28	Yes	Queens	Tuberculosis
	24	Yes	Brooklyn	Heart Disease
	19	No	Brooklyn	COVID-19
	29	No	Manhattan	Heart Disease
	17	No	Brooklyn	Diabetes
	19	No	Brooklyn	Viral Infection


# Anonymization/Sanitization

- Removal of EIDs from data.





User ID	Age (Public)	Registered Voter? (Public)	Borough (Public)	Disease (Private)
	27	No	Manhattan	Diabetes
	30	Yes	Queens	Cancer
	23	No	Manhattan	COVID-19
	24	Yes	Brooklyn	Viral Infection
	28	Yes	Queens	Tuberculosis
	24	Yes	Brooklyn	Heart Disease
	19	No	Brooklyn	COVID-19
	29	No	Manhattan	Heart Disease
	17	No	Brooklyn	Diabetes
	19	No	Brooklyn	Viral Infection

- Some QIDs may be unique to users.
- **Generalization:** Place attributes in broader categories such that the dataset
  - is harder to de-anonymize,
  - still retains its utility.

# Generalization/Resolution Reduction

User ID	Age (Public)	Registered Voter? (Public)	Borough (Public)	Disease (Private)
	27	No	Manhattan	Diabetes
	30	Yes	Queens	Cancer
	23	No	Manhattan	COVID-19
	24	Yes	Brooklyn	Viral Infection
	28	Yes	Queens	Tuberculosis
	24	Yes	Brooklyn	Heart Disease
	19	No	Brooklyn	COVID-19
	29	No	Manhattan	Heart Disease
	17	No	Brooklyn	Diabetes
	19	No	Brooklyn	Viral Infection

# Generalization/Resolution Reduction

User ID	Age (Public)	Registered Voter? (Public)	Borough (Public)	Disease (Private)
	$20 < \text{Age} \leq 30$	No	Manhattan	Diabetes
	$20 < \text{Age} \leq 30$	Yes	Queens	Cancer
	$20 < \text{Age} \leq 30$	No	Manhattan	COVID-19
	$20 < \text{Age} \leq 30$	Yes	Brooklyn	Viral Infection
	$20 < \text{Age} \leq 30$	Yes	Queens	Tuberculosis
	$20 < \text{Age} \leq 30$	Yes	Brooklyn	Heart Disease
	$\text{Age} \leq 20$	No	Brooklyn	COVID-19
	$20 < \text{Age} \leq 30$	No	Manhattan	Heart Disease
	$\text{Age} \leq 20$	No	Brooklyn	Diabetes
	$\text{Age} \leq 20$	No	Brooklyn	Viral Infection

## *k*-Anonymity

An anonymized dataset possesses *k*-anonymity if the QIDs of any user *u* is non-separable from at least  $k - 1$  other users.

- Popular notion of privacy introduced in [Samarati & Sweeney, 1998].
- Non-separable users form a cluster.
- *k*-anonymity is achieved through generalization.

# k-Anonymity: Example

A 2-anonymous database with Disease as a Sensitive Attribute.

User ID	Age (Public)	Registered Voter? (Public)	Borough (Public)	Disease (Private)
	$20 < \text{Age} \leq 30$	No	Manhattan	Diabetes
	$20 < \text{Age} \leq 30$	Yes	Queens	Cancer
	$20 < \text{Age} \leq 30$	No	Manhattan	COVID-19
	$20 < \text{Age} \leq 30$	Yes	Brooklyn	Viral Infection
	$20 < \text{Age} \leq 30$	Yes	Queens	Tuberculosis
	$20 < \text{Age} \leq 30$	Yes	Brooklyn	Heart Disease
	$\text{Age} \leq 20$	No	Brooklyn	COVID-19
	$20 < \text{Age} \leq 30$	No	Manhattan	Heart Disease
	$\text{Age} \leq 20$	No	Brooklyn	Diabetes
	$\text{Age} \leq 20$	No	Brooklyn	Viral Infection



- *k*-anonymity depends on a well-defined QID-SA separation.
  - Not very practical.
  - Different adversaries have access to different attributes.

- $k$ -anonymity depends on a well-defined QID-SA separation.
  - Not very practical.
  - Different adversaries have access to different attributes.
  - *Potential Solution*: Treat all attributes as QIDs.

- $k$ -anonymity depends on a well-defined QID-SA separation.
  - Not very practical.
  - Different adversaries have access to different attributes.
  - *Potential Solution*: Treat all attributes as QIDs.
- Using other public information, data may still be de-anonymized up to a cluster of size  $k$ .

- $k$ -anonymity depends on a well-defined QID-SA separation.
  - Not very practical.
  - Different adversaries have access to different attributes.
  - *Potential Solution*: Treat all attributes as QIDs.
- Using other public information, data may still be de-anonymized up to a cluster of size  $k$ .
- Even with  $k$ -anonymity, it is important to understand *fundamentals* of database matching.

- Different notions of privacy.
  - Privacy-preserving data analytics.
    - Differential privacy.
  - Privacy-preserving microdata publishing.
- Overview of typicality and asymptotic equipartition property (AEP).

# Weak Law of Large Numbers

- Typicality directly follows from law of large numbers.

# Weak Law of Large Numbers

- Typicality directly follows from law of large numbers.
- Weak Law of Large Numbers (WLLN):

## WLLN

Let  $X_1, X_2, \dots$  be a sequence of iid random variables with distribution  $P_X$  and mean  $\mathbb{E}(X)$ . Then,

$$P(|\bar{X}_n - \mathbb{E}(X)| \geq \epsilon) \rightarrow 0, \text{ as } n \rightarrow \infty, \forall \epsilon > 0,$$

where  $\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$ .

- WLLN: empirical average is arbitrarily close to the statistical average for large sequences.



# Weak Law of Large Numbers

- WLLN: empirical average is arbitrarily close to the statistical average for large sequences.
- Ex.: Average fraction of heads when flipping a fair coin  $\approx \frac{1}{2}$ .

- WLLN: empirical average is arbitrarily close to the statistical average for large sequences.
- Ex.: Average fraction of heads when flipping a fair coin  $\approx \frac{1}{2}$ .
- Let  $X^n = (X_1, X_2, \dots, X_n)$  be the outcomes of  $n$  consecutive coin flips.

# Weak Law of Large Numbers

- WLLN: empirical average is arbitrarily close to the statistical average for large sequences.
- Ex.: Average fraction of heads when flipping a fair coin  $\approx \frac{1}{2}$ .
- Let  $X^n = (X_1, X_2, \dots, X_n)$  be the outcomes of  $n$  consecutive coin flips.
- $X^n$  is typical if number of heads  $\approx \frac{n}{2}$ .

# Weak Law of Large Numbers

- WLLN: empirical average is arbitrarily close to the statistical average for large sequences.
- Ex.: Average fraction of heads when flipping a fair coin  $\approx \frac{1}{2}$ .
- Let  $X^n = (X_1, X_2, \dots, X_n)$  be the outcomes of  $n$  consecutive coin flips.
- $X^n$  is typical if number of heads  $\approx \frac{n}{2}$ .
- $X^n$  is atypical otherwise.

- Ex.: Let  $X^n = (X_1, X_2, \dots, X_n)$  be a sequence of iid ternary variables ( $\mathcal{X} = \{1, 2, 3\}$ ).

# Weak Law of Large Numbers

- Ex.: Let  $X^n = (X_1, X_2, \dots, X_n)$  be a sequence of iid ternary variables ( $\mathcal{X} = \{1, 2, 3\}$ ).
- Let  $P_X(1) = P_X(2) = \frac{1}{4}$ ,  $P_X(3) = \frac{1}{2}$ .

- Ex.: Let  $X^n = (X_1, X_2, \dots, X_n)$  be a sequence of iid ternary variables ( $\mathcal{X} = \{1, 2, 3\}$ ).
- Let  $P_X(1) = P_X(2) = \frac{1}{4}, P_X(3) = \frac{1}{2}$ .
- Define  $\bar{N}_i^n, i \in \{1, 2, 3\}$  as the average number of  $i$  outcomes:

$$\bar{N}_i^n = \frac{1}{n} |\{j | X_j = i\}|.$$

- Ex.: Let  $X^n = (X_1, X_2, \dots, X_n)$  be a sequence of iid ternary variables ( $\mathcal{X} = \{1, 2, 3\}$ ).
- Let  $P_X(1) = P_X(2) = \frac{1}{4}, P_X(3) = \frac{1}{2}$ .
- Define  $\bar{N}_i^n, i \in \{1, 2, 3\}$  as the average number of  $i$  outcomes:

$$\bar{N}_i^n = \frac{1}{n} |\{j | X_j = i\}|.$$

- From WLLN:

$$P \left( |\bar{N}_1^n - \frac{1}{4}| > \epsilon \text{ or } |\bar{N}_2^n - \frac{1}{4}| > \epsilon \text{ or } |\bar{N}_3^n - \frac{1}{2}| > \epsilon \right) \rightarrow 0, n \rightarrow \infty.$$



- The set of  $n$ -length,  $\epsilon$ -typical sequences with respect to  $P_X$  is defined as:

$$A_\epsilon^n(X) = \{x^n : |\bar{N}_a^n - P_X(a)| \leq \epsilon, \forall a \in \mathcal{X}\}.$$

- The set of  $n$ -length,  $\epsilon$ -typical sequences with respect to  $P_X$  is defined as:

$$A_\epsilon^n(X) = \{x^n : |\bar{N}_a^n - P_X(a)| \leq \epsilon, \forall a \in \mathcal{X}\}.$$

- Ex.:  
Ternary alphabet  $\mathcal{X} = \{1, 2, 3\}$ ,  
 $P_X(1) = P_X(2) = \frac{1}{4}$ ,  $P_X(3) = \frac{1}{2}$ ,

- The set of  $n$ -length,  $\epsilon$ -typical sequences with respect to  $P_X$  is defined as:

$$A_\epsilon^n(X) = \{x^n : |\bar{N}_a^n - P_X(a)| \leq \epsilon, \forall a \in \mathcal{X}\}.$$

- Ex.:

Ternary alphabet  $\mathcal{X} = \{1, 2, 3\}$ ,  
 $P_X(1) = P_X(2) = \frac{1}{4}$ ,  $P_X(3) = \frac{1}{2}$ ,

$n = 10$ ,  $\epsilon = 0.1$ :

Typical: 3312311233  $\rightarrow \bar{N}_1^n = 0.3$ ,  $\bar{N}_2^n = 0.2$ ,  $\bar{N}_3^n = 0.5$ .

- The set of  $n$ -length,  $\epsilon$ -typical sequences with respect to  $P_X$  is defined as:

$$A_\epsilon^n(X) = \{x^n : |\bar{N}_a^n - P_X(a)| \leq \epsilon, \forall a \in \mathcal{X}\}.$$

- Ex.:

Ternary alphabet  $\mathcal{X} = \{1, 2, 3\}$ ,  
 $P_X(1) = P_X(2) = \frac{1}{4}$ ,  $P_X(3) = \frac{1}{2}$ ,  
 $n = 10$ ,  $\epsilon = 0.1$ :

Typical: 3312311233  $\rightarrow \bar{N}_1^n = 0.3$ ,  $\bar{N}_2^n = 0.2$ ,  $\bar{N}_3^n = 0.5$ .

Atypical: 1111111111  $\rightarrow \bar{N}_1^n = 1$ ,  $\bar{N}_2^n = 0$ ,  $\bar{N}_3^n = 0$ .

- Consider the pair  $(X, Y)$  with joint distribution  $P_{X,Y}$ .

# Joint Typicality

- Consider the pair  $(X, Y)$  with joint distribution  $P_{X,Y}$ .
- The set of  $n$ -length,  $\epsilon$ -typical sequences with respect to  $P_{X,Y}$  is defined as:

$$A_\epsilon^n(X, Y) = \{(x^n, y^n) : |\bar{N}_{a,b}^n - P_{X,Y}(a, b)| \leq \epsilon, \forall (a, b) \in \mathcal{X} \times \mathcal{Y}\}.$$

- Consider the pair  $(X, Y)$  with joint distribution  $P_{X,Y}$ .
- The set of  $n$ -length,  $\epsilon$ -typical sequences with respect to  $P_{X,Y}$  is defined as:

$$A_\epsilon^n(X, Y) = \{(x^n, y^n) : |\bar{N}_{a,b}^n - P_{X,Y}(a, b)| \leq \epsilon, \forall (a, b) \in \mathcal{X} \times \mathcal{Y}\}.$$

- Example:

Binary alphabet  $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ ,

$$P_{X,Y}(0, 0) = P_{X,Y}(1, 1) = \frac{1}{3}, P_{X,Y}(0, 1) = P_{X,Y}(1, 0) = \frac{1}{6},$$

# Joint Typicality

- Consider the pair  $(X, Y)$  with joint distribution  $P_{X,Y}$ .
- The set of  $n$ -length,  $\epsilon$ -typical sequences with respect to  $P_{X,Y}$  is defined as:

$$A_\epsilon^n(X, Y) = \{(x^n, y^n) : |\bar{N}_{a,b}^n - P_{X,Y}(a, b)| \leq \epsilon, \forall (a, b) \in \mathcal{X} \times \mathcal{Y}\}.$$

- Example:

Binary alphabet  $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ ,

$$P_{X,Y}(0, 0) = P_{X,Y}(1, 1) = \frac{1}{3}, P_{X,Y}(0, 1) = P_{X,Y}(1, 0) = \frac{1}{6},$$

$n = 6, \epsilon = 0.1$ :

Typical:

$$\begin{cases} X^n = (001101) \\ Y^n = (011100). \end{cases} \rightarrow \bar{N}_{0,0}^n = \bar{N}_{1,1}^n = \frac{1}{3}, \bar{N}_{0,1}^n = \bar{N}_{1,0}^n = \frac{1}{6}.$$



# Asymptotic Equipartition Property (AEP)

- **Result 1:** Let  $X^n$  be an iid sequence where  $X_i \sim P_X$ , then:

$$P(X^n \in A_\epsilon^n(X)) \rightarrow 1, \text{ as } n \rightarrow \infty.$$

# Asymptotic Equipartition Property (AEP)

- **Result 1:** Let  $X^n$  be an iid sequence where  $X_i \sim P_X$ , then:

$$P(X^n \in A_\epsilon^n(X)) \rightarrow 1, \text{ as } n \rightarrow \infty.$$

- **Result 2:** Let  $(X^n, Y^n)$  be an iid sequence where  $(X_i, Y_i) \sim P_{X,Y}$ , then:

$$P((X^n, Y^n) \in A_\epsilon^n(X, Y)) \rightarrow 1, \text{ as } n \rightarrow \infty.$$

# Asymptotic Equipartition Property (AEP)

- **Result 1:** Let  $X^n$  be an iid sequence where  $X_i \sim P_X$ , then:

$$P(X^n \in A_\epsilon^n(X)) \rightarrow 1, \text{ as } n \rightarrow \infty.$$

- **Result 2:** Let  $(X^n, Y^n)$  be an iid sequence where  $(X_i, Y_i) \sim P_{X,Y}$ , then:

$$P((X^n, Y^n) \in A_\epsilon^n(X, Y)) \rightarrow 1, \text{ as } n \rightarrow \infty.$$

- **Result 3:** Let  $X^n$  and  $Y^n$  be two iid sequences where  $(X_i, Y_i) \sim P_X P_Y$ , then:

$$P((X^n, Y^n) \in A_\epsilon^n(X, Y)) \leq 2^{-n(I(X;Y)-2\epsilon)}.$$

- How can information theory help in database and graph matching?
- Consider random databases/graphs.
  - Allows using tools from **information theory**, leading to **theoretical guarantees** and **new algorithms**.
- Database matching.
  - Correlated databases.
  - Time series: Deletions and replicas.
    - Noiseless.
    - Noisy.
- General graph matching.
  - Matching identical graphs.
  - Correlated graphs with seeds.
  - Correlated graphs without seeds.
    - New typicality results on permutations of sequences.

# Database Matching

$\mathbf{D}^{(1)}$			
User ID	Attribute Vector		
1	$X_{1,1}$	$\cdots$	$X_{1,n}$
$\vdots$	$\vdots$		$\vdots$
$m_n$	$X_{m_n,1}$	$\cdots$	$X_{m_n,n}$

$\mathbf{D}^{(2)}$		
Attribute Vector		
$Y_{\Theta^{-1}(1),1}$	$\cdots$	$Y_{\Theta^{-1}(1),n}$
$\vdots$		$\vdots$
$Y_{\Theta^{-1}(m_n),1}$	$\cdots$	$Y_{\Theta^{-1}(m_n),n}$

- Databases  $\mathbf{D}^{(1)} = (X_{i,j})$ ,  $\mathbf{D}^{(2)} = (Y_{i,j})$ ,  $i \in [m_n]$ ,  $j \in [n]$ .

# Database Matching

$\mathbf{D}^{(1)}$				$\mathbf{D}^{(2)}$		
User ID	Attribute Vector			Attribute Vector		
1	$X_{1,1}$	...	$X_{1,n}$	$Y_{\Theta^{-1}(1),1}$	...	$Y_{\Theta^{-1}(1),n}$
$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
$m_n$	$X_{m_n,1}$	...	$X_{m_n,n}$	$Y_{\Theta^{-1}(m_n),1}$	...	$Y_{\Theta^{-1}(m_n),n}$

- Databases  $\mathbf{D}^{(1)} = (X_{i,j})$ ,  $\mathbf{D}^{(2)} = (Y_{i,j})$ ,  $i \in [m_n]$ ,  $j \in [n]$ .
- Labeling  $\Theta$ : Permutation of  $[m_n]$ .
- Rows with the same ID are called **matching**.  
(i.e.  $i = \Theta^{-1}(p) \Rightarrow i$  and  $p$  matching.)

# Database Matching

$\mathbf{D}^{(1)}$				$\mathbf{D}^{(2)}$		
User ID	Attribute Vector			Attribute Vector		
1	$X_{1,1}$	$\cdots$	$X_{1,n}$	$Y_{\Theta^{-1}(1),1}$	$\cdots$	$Y_{\Theta^{-1}(1),n}$
$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
$m_n$	$X_{m_n,1}$	$\cdots$	$X_{m_n,n}$	$Y_{\Theta^{-1}(m_n),1}$	$\cdots$	$Y_{\Theta^{-1}(m_n),n}$

- Databases  $\mathbf{D}^{(1)} = (X_{i,j})$ ,  $\mathbf{D}^{(2)} = (Y_{i,j})$ ,  $i \in [m_n]$ ,  $j \in [n]$ .
- Labeling  $\Theta$ : Permutation of  $[m_n]$ .
- Rows with the same ID are called **matching**.  
(i.e.  $i = \Theta^{-1}(p) \Rightarrow i$  and  $p$  matching.)
- Entries are generated stochastically.
- Entries with matching IDs correlated:  $f(x, y)$
- Entries with different member IDs independent.

- Objective: Given  $(\mathbf{D}^{(1)}, \mathbf{D}^{(2)})$ , find  $\hat{\Theta}$  s.t.:

$$P(\Theta(I) = \hat{\Theta}(I)) \rightarrow 1 \text{ as } n \rightarrow \infty,$$

where  $I \sim U(1, m_n)$ .

- Almost all entries must be matched correctly.



- Objective: Given  $(\mathbf{D}^{(1)}, \mathbf{D}^{(2)})$ , find  $\hat{\Theta}$  s.t.:

$$P(\Theta(I) = \hat{\Theta}(I)) \rightarrow 1 \text{ as } n \rightarrow \infty,$$

where  $I \sim U(1, m_n)$ .

- Almost all entries must be matched correctly.
  - In [Cullina, Mittal, Kiyavash, 2018]: All entries must be matched correctly.

- Objective: Given  $(\mathbf{D}^{(1)}, \mathbf{D}^{(2)})$ , find  $\hat{\Theta}$  s.t.:

$$P(\Theta(I) = \hat{\Theta}(I)) \rightarrow 1 \text{ as } n \rightarrow \infty,$$

where  $I \sim U(1, m_n)$ .

- Almost all entries must be matched correctly.
  - In [Cullina, Mittal, Kiyavash, 2018]: All entries must be matched correctly.
- This allows us to use information theoretic tools and work with arbitrary distributions.

# Analogy with Channel Coding

$\mathbf{D}^{(1)}$

User ID	Attribute Vector		
1	$X_{1,1}$	$\cdots$	$X_{1,n}$
$\vdots$	$\vdots$		$\vdots$
$m_n$	$X_{m_n,1}$	$\cdots$	$X_{m_n,n}$

Attribute Vector

$Y_{\Theta^{-1}(i),1}$	$\cdots$	$Y_{\Theta^{-1}(i),n}$
------------------------	----------	------------------------

- Pick a row from  $\mathbf{D}^{(2)}$ .
- Goal: Find the matching row in  $\mathbf{D}^{(1)}$ .

# Analogy with Channel Coding

$\mathbf{D}^{(1)}$

User ID	Attribute Vector		
1	$X_{1,1}$	...	$X_{1,n}$
$\vdots$	$\vdots$		$\vdots$
$m_n$	$X_{m_n,1}$	...	$X_{m_n,n}$

Attribute Vector

$Y_{\Theta^{-1}(i),1}$	...	$Y_{\Theta^{-1}(i),n}$
------------------------	-----	------------------------

- Pick a row from  $\mathbf{D}^{(2)}$ .
- Goal: Find the matching row in  $\mathbf{D}^{(1)}$ .
- Similar to [channel decoding](#).

# Analogy with Channel Coding

$\mathbf{D}^{(1)}$

User ID	Attribute Vector		
1	$X_{1,1}$	...	$X_{1,n}$
$\vdots$	$\vdots$		$\vdots$
$m_n$	$X_{m_n,1}$	...	$X_{m_n,n}$

Attribute Vector

$Y_{\Theta^{-1}(i),1}$	...	$Y_{\Theta^{-1}(i),n}$
------------------------	-----	------------------------

- Pick a row from  $\mathbf{D}^{(2)}$ .
- Goal: Find the matching row in  $\mathbf{D}^{(1)}$ .
- Similar to [channel decoding](#).
- Channel coding analogy:
  - Number of members  $m_n \rightarrow$  [number of messages](#).

# Analogy with Channel Coding

$\mathbf{D}^{(1)}$

User ID	Attribute Vector		
1	$X_{1,1}$	...	$X_{1,n}$
$\vdots$	$\vdots$		$\vdots$
$m_n$	$X_{m_n,1}$	...	$X_{m_n,n}$

Attribute Vector

$Y_{\Theta^{-1}(i),1}$	...	$Y_{\Theta^{-1}(i),n}$
------------------------	-----	------------------------

- Pick a row from  $\mathbf{D}^{(2)}$ .
- Goal: Find the matching row in  $\mathbf{D}^{(1)}$ .
- Similar to [channel decoding](#).
- Channel coding analogy:
  - Number of members  $m_n \rightarrow$  [number of messages](#).
  - Attribute length  $n \rightarrow$  [blocklength](#).

# Analogy with Channel Coding

$\mathbf{D}^{(1)}$

User ID	Attribute Vector		
1	$X_{1,1}$	...	$X_{1,n}$
$\vdots$	$\vdots$		$\vdots$
$m_n$	$X_{m_n,1}$	...	$X_{m_n,n}$

Attribute Vector

$Y_{\Theta^{-1}(i),1}$	...	$Y_{\Theta^{-1}(i),n}$
------------------------	-----	------------------------

- Pick a row from  $\mathbf{D}^{(2)}$ .
- Goal: Find the matching row in  $\mathbf{D}^{(1)}$ .
- Similar to [channel decoding](#).
- Channel coding analogy:
  - Number of members  $m_n \rightarrow$  [number of messages](#).
  - Attribute length  $n \rightarrow$  [blocklength](#).
  - Conditional density  $f(y|x)$  on pairs of matching entries  $\rightarrow$  [channel transition probability](#).

# Analogy with Channel Coding

$\mathbf{D}^{(1)}$

User ID	Attribute Vector		
1	$X_{1,1}$	...	$X_{1,n}$
$\vdots$	$\vdots$		$\vdots$
$m_n$	$X_{m_n,1}$	...	$X_{m_n,n}$

Attribute Vector

$Y_{\Theta^{-1}(i),1}$	...	$Y_{\Theta^{-1}(i),n}$
------------------------	-----	------------------------

- Pick a row from  $\mathbf{D}^{(2)}$ .
- Goal: Find the matching row in  $\mathbf{D}^{(1)}$ .
- Similar to [channel decoding](#).
- Channel coding analogy:
  - Number of members  $m_n \rightarrow$  [number of messages](#).
  - Attribute length  $n \rightarrow$  [blocklength](#).
  - Conditional density  $f(y|x)$  on pairs of matching entries  $\rightarrow$  [channel transition probability](#).
  - Database growth rate  $R = \lim_{n \rightarrow \infty} \frac{1}{n} \log m_n \rightarrow$  [codebook rate](#).



## Theorem

Databases with growth rate  $R$  generated according to  $f(x, y)$  can be successfully matched if

$$R < I(X; Y).$$

Furthermore, a necessary condition for the existence of a successful matching scheme is:

$$R \leq I(X; Y).$$

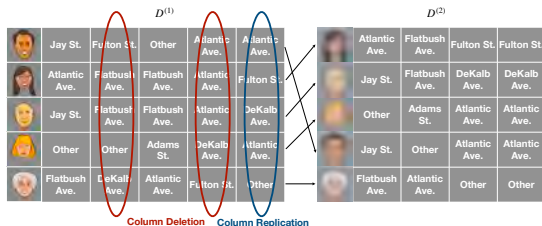
- Shirani, Garg, Erkip, ISIT 2019.

- Proof borrows results from channel coding.
- Similarity with channel coding:
  - Achievability: Typicality matching similar to typicality decoding.
  - Converse: A variation of Fano's inequality.

- Proof borrows results from channel coding.
- Similarity with channel coding:
  - Achievability: Typicality matching similar to typicality decoding.
  - Converse: A variation of Fano's inequality.
- Differences with channel coding:
  - Database distribution not a design parameter.
  - All database entries matched exactly once.

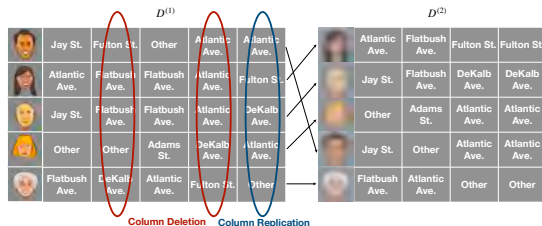
- How can information theory help in database and graph matching?
- Consider random databases/graphs.
  - Allows using tools from **information theory**, leading to **theoretical guarantees** and **new algorithms**.
- Database matching.
  - Correlated databases.
  - Time series: Deletions and replicas.
    - Noiseless.
    - Noisy.
- General graph matching.
  - Matching identical graphs.
  - Correlated graphs with seeds.
  - Correlated graphs without seeds.
    - New typicality results on permutations of sequences.

# Database Matching Under Column Deletions/Replicas



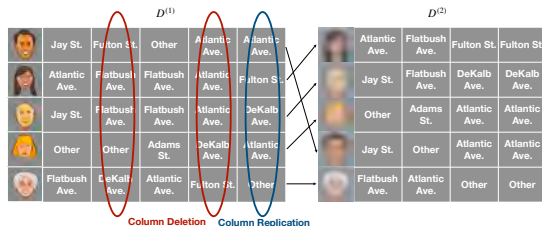
- Synchronization errors in time-indexed databases.
- Indices of the deleted/replicated columns are not known.

# Database Matching Under Column Deletions/Replicas



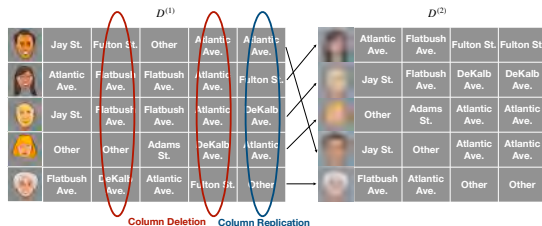
- Synchronization errors in time-indexed databases.
- Indices of the deleted/replicated columns are not known.
- Deletion/replica pattern constant across  $W$  rows.

# Database Matching Under Column Deletions/Replicas



- Synchronization errors in time-indexed databases.
- Indices of the deleted/replicated columns are not known.
- Deletion/replica pattern constant across  $W$  rows.
  - $W = 1$ : Ideas from correlated databases and deletion channel.

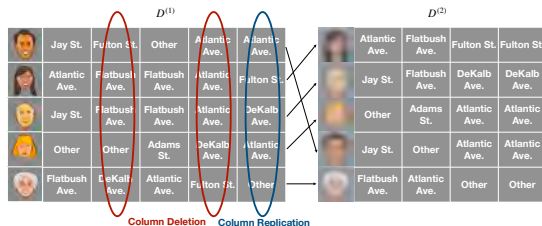
# Database Matching Under Column Deletions/Replicas



- Synchronization errors in time-indexed databases.
- Indices of the deleted/replicated columns are not known.
- Deletion/replica pattern constant across  $W$  rows.
  - $W = 1$ : Ideas from correlated databases and deletion channel.
  - $W = m_n$ : This talk.



# Database Matching Under Column Deletions/Replicas



- Synchronization errors in time-indexed databases.
- Indices of the deleted/replicated columns are not known.
- Deletion/replica pattern constant across  $W$  rows.
  - $W = 1$ : Ideas from correlated databases and deletion channel.
  - $W = m_n$ : This talk.
- Bakirtas, Erkip, ISIT 2021, ITW 2022, Asilomar 2022.

# Noiseless Column Repetition: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .

# Noiseless Column Repetition: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .
- $\Theta$ : Uniform permutation of  $[m_n]$ .

# Noiseless Column Repetition: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .
- $\Theta$ : Uniform permutation of  $[m_n]$ .
- Column repetition pattern: random vector  $S^n = \{S_1, S_2, \dots, S_n\}$  with  $S_j \stackrel{i.i.d.}{\sim} p_S$ .
  - $\text{supp}(p_S) = \{0, \dots, S_{\max}\}$

# Noiseless Column Repetition: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .
- $\Theta$ : Uniform permutation of  $[m_n]$ .
- Column repetition pattern: random vector  $S^n = \{S_1, S_2, \dots, S_n\}$  with  $S_j \stackrel{i.i.d.}{\sim} p_S$ .
  - $\text{supp}(p_S) = \{0, \dots, S_{\max}\}$
  - *Repetition* includes deletions and replicas.

# Noiseless Column Repetition: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .
- $\Theta$ : Uniform permutation of  $[m_n]$ .
- Column repetition pattern: random vector  $S^n = \{S_1, S_2, \dots, S_n\}$  with  $S_j \stackrel{i.i.d.}{\sim} p_S$ .
  - $\text{supp}(p_S) = \{0, \dots, S_{\max}\}$
  - *Repetition* includes deletions and replicas.
- $\mathbf{D}^{(2)}$ : Obtained from  $\mathbf{D}^{(1)}$  by
  - 1 Row shuffling by  $\Theta$ .

# Noiseless Column Repetition: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .
- $\Theta$ : Uniform permutation of  $[m_n]$ .
- Column repetition pattern: random vector  $S^n = \{S_1, S_2, \dots, S_n\}$  with  $S_j \stackrel{i.i.d.}{\sim} p_S$ .
  - $\text{supp}(p_S) = \{0, \dots, S_{\max}\}$
  - *Repetition* includes deletions and replicas.
- $\mathbf{D}^{(2)}$ : Obtained from  $\mathbf{D}^{(1)}$  by
  - 1 Row shuffling by  $\Theta$ .
  - 2 Column repetition by  $S^n$ .
    - Replicate the  $j^{\text{th}}$  column  $S_j$  times if  $S_j > 0$ .
    - Delete the  $j^{\text{th}}$  column if  $S_j = 0$ .

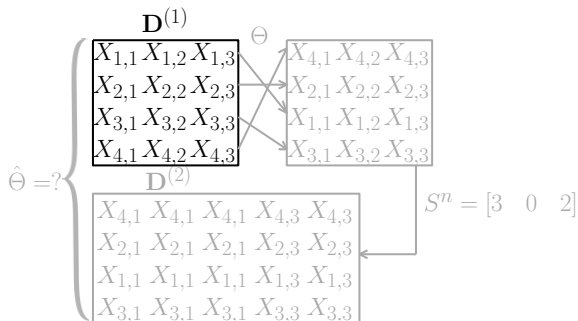
# Noiseless Column Repetition: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .
- $\Theta$ : Uniform permutation of  $[m_n]$ .
- Column repetition pattern: random vector  $S^n = \{S_1, S_2, \dots, S_n\}$  with  $S_j \stackrel{i.i.d.}{\sim} p_S$ .
  - $\text{supp}(p_S) = \{0, \dots, S_{\max}\}$
  - *Repetition* includes deletions and replicas.
- $\mathbf{D}^{(2)}$ : Obtained from  $\mathbf{D}^{(1)}$  by
  - 1 Row shuffling by  $\Theta$ .
  - 2 Column repetition by  $S^n$ .
    - Replicate the  $j^{\text{th}}$  column  $S_j$  times if  $S_j > 0$ .
    - Delete the  $j^{\text{th}}$  column if  $S_j = 0$ .
- No noise on the entries.

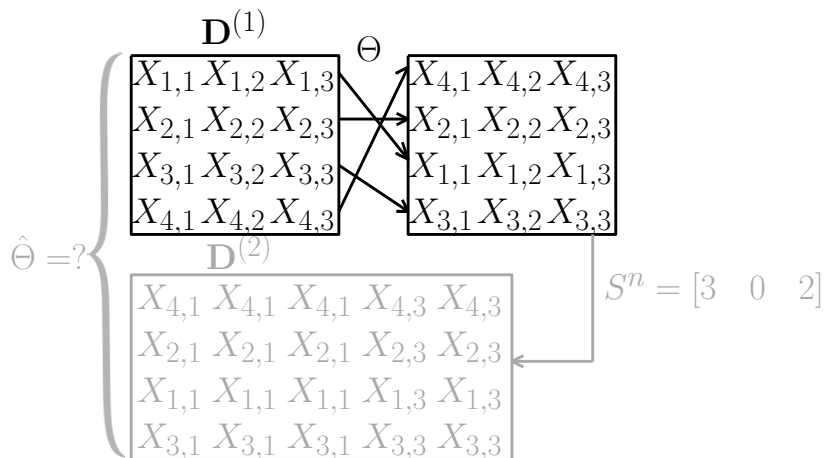


# Example

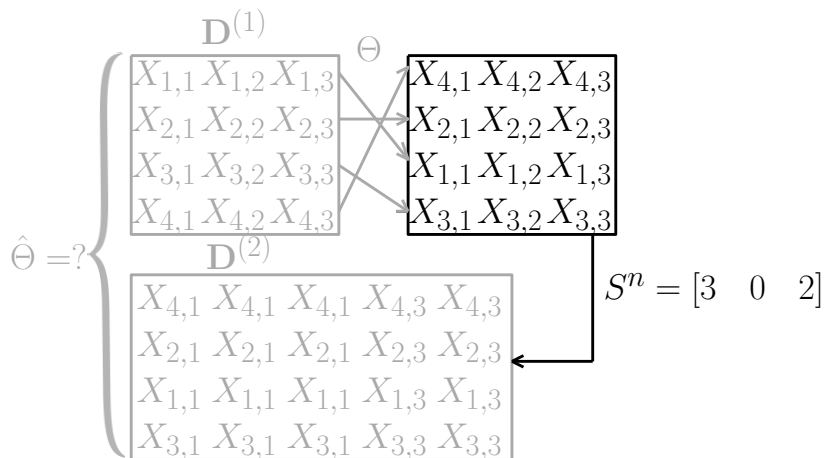
- Two databases the same (i.e. no noise), except for row permutation, column deletions/repetitions.
- $m_n = 4, n = 3, S^n$ : deletion/replica pattern.



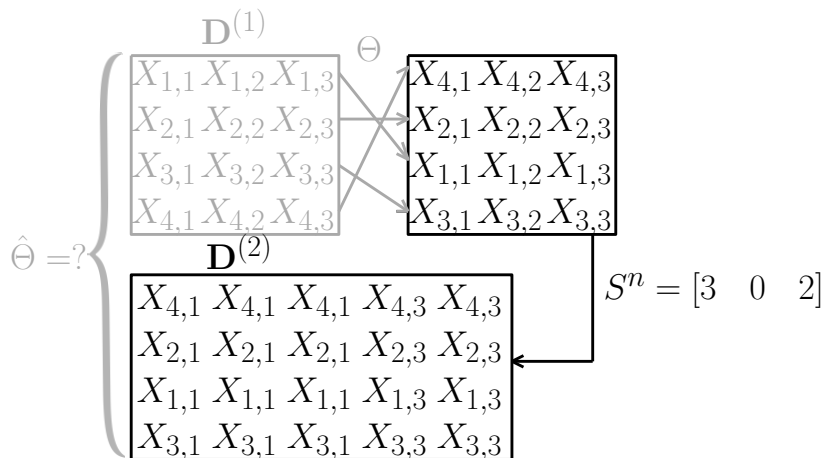
# Example



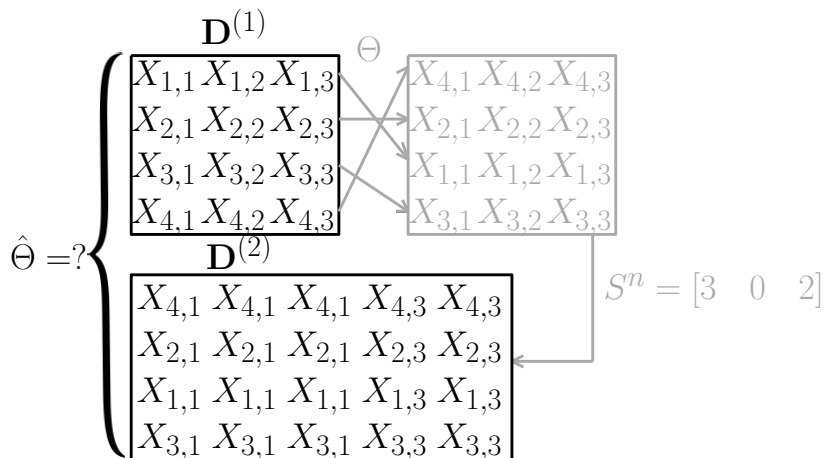
# Example



# Example



# Example



# Proposed Matching Scheme

- No noise, only repetitions.
- Exploit the identical repetition pattern across rows.

# Proposed Matching Scheme

- No noise, only repetitions.
- Exploit the identical repetition pattern across rows.
  - ① Find a permutation-invariant unique feature of the columns.

# Proposed Matching Scheme

- No noise, only repetitions.
- Exploit the identical repetition pattern across rows.
  - ① Find a permutation-invariant unique feature of the columns.
  - ② By matching these features, infer  $S^n$ .



# Proposed Matching Scheme

- No noise, only repetitions.
- Exploit the identical repetition pattern across rows.
  - ① Find a permutation-invariant unique feature of the columns.
  - ② By matching these features, infer  $S^n$ .
  - ③ Discard the deleted columns from  $\mathbf{D}^{(1)}$ .
  - ④ Discard the replicated columns from  $\mathbf{D}^{(2)}$ .

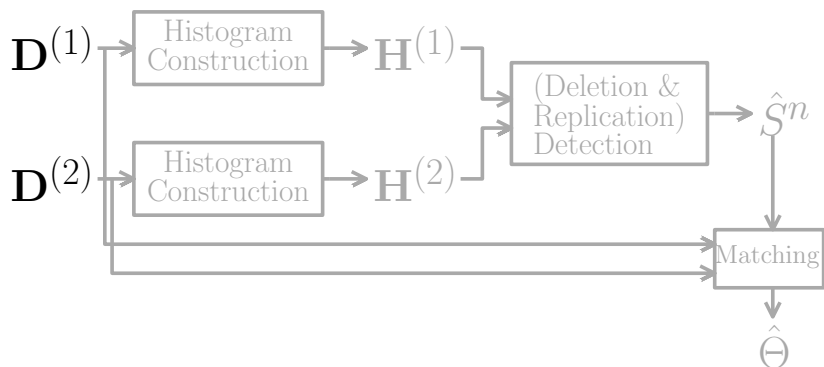
# Proposed Matching Scheme

- No noise, only repetitions.
- Exploit the identical repetition pattern across rows.
  - 1 Find a permutation-invariant unique feature of the columns.
  - 2 By matching these features, infer  $S^n$ .
  - 3 Discard the deleted columns from  $\mathbf{D}^{(1)}$ .
  - 4 Discard the replicated columns from  $\mathbf{D}^{(2)}$ .
  - 5 Perform exact rowwise matching.

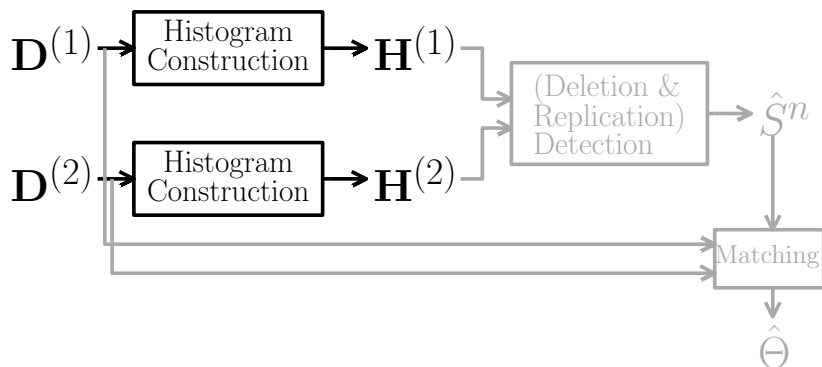
# Proposed Matching Scheme

- No noise, only repetitions.
- Exploit the identical repetition pattern across rows.
  - 1 Find a permutation-invariant unique feature of the columns.
  - 2 By matching these features, infer  $S^n$ .
  - 3 Discard the deleted columns from  $\mathbf{D}^{(1)}$ .
  - 4 Discard the replicated columns from  $\mathbf{D}^{(2)}$ .
  - 5 Perform exact rowwise matching.
- We will use **column histograms** as the permutation-invariant feature.

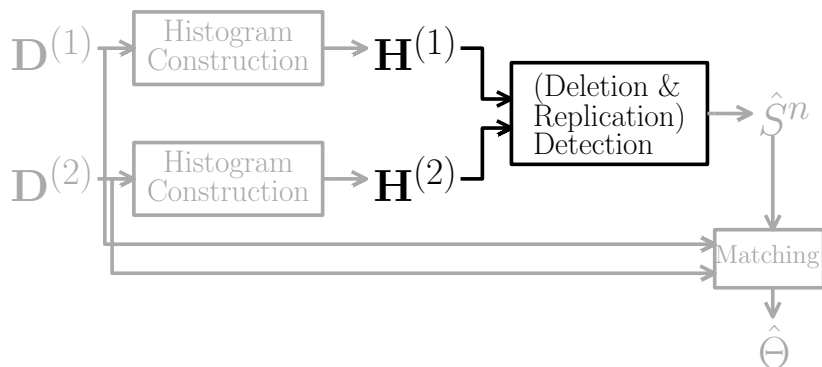
# Matching Scheme



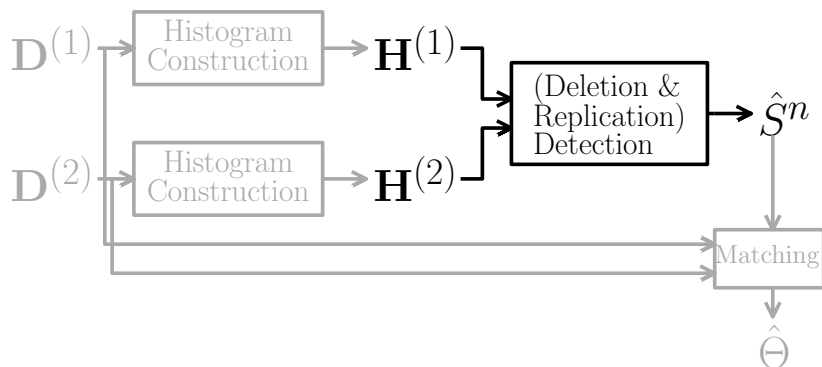
# Matching Scheme



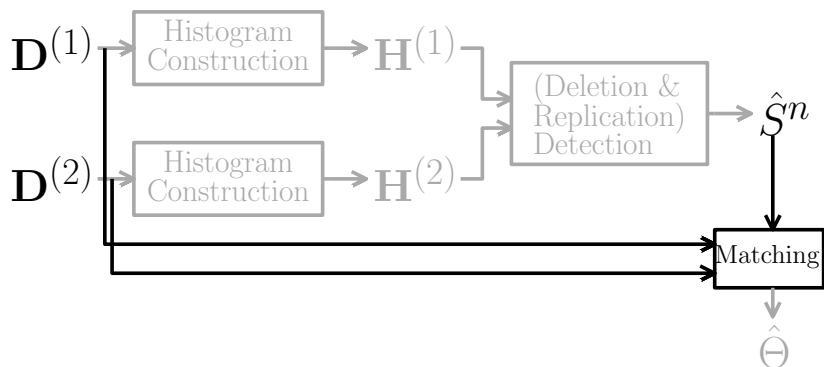
# Matching Scheme



# Matching Scheme

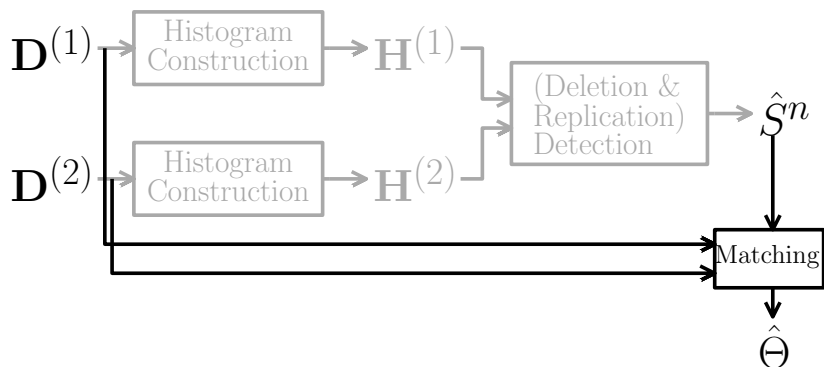


# Matching Scheme





# Matching Scheme



# Matching Scheme: Example

**D**<sup>(1)</sup>

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

**D**<sup>(2)</sup>

<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>a</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>d</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>

# Matching Scheme: Example

**D**<sup>(1)</sup>

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

**H**<sup>(1)</sup>

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

# Matching Scheme: Example

**D**<sup>(1)</sup>

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

**H**<sup>(1)</sup>

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

# Matching Scheme: Example

**D**<sup>(1)</sup>

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

**H**<sup>(1)</sup>

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

# Matching Scheme: Example

**D**<sup>(1)</sup>

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

**H**<sup>(1)</sup>

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

**D**<sup>(1)</sup>

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

**H**<sup>(1)</sup>

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

# Matching Scheme: Example

$\mathbf{D}^{(1)}$

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

$\mathbf{H}^{(1)}$

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1



# Matching Scheme: Example

**D**<sup>(1)</sup>

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

**H**<sup>(1)</sup>

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

# Matching Scheme: Example

$\mathbf{D}^{(1)}$

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

$\mathbf{H}^{(1)}$

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

# Matching Scheme: Example

**D**<sup>(1)</sup>

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

**H**<sup>(1)</sup>

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

$\mathbf{D}^{(1)}$

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

$\mathbf{H}^{(1)}$

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

# Matching Scheme: Example

$D^{(2)}$

<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>a</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>d</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>

$H^{(2)}$

3	3	1	1	1	2	0
1	1	4	4	4	4	4
4	4	3	3	3	2	3
0	0	0	0	0	0	1

$\mathbf{H}^{(1)}$ 

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

 $\mathbf{H}^{(2)}$ 

3	3	1	1	1	2	0
1	1	4	4	4	4	4
4	4	3	3	3	2	3
0	0	0	0	0	0	1

$$\hat{\mathbf{S}} = [2 \ 0 \ 3 \ 1 \ 0 \ 1]$$

$\mathbf{H}^{(1)}$ 

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

 $\mathbf{H}^{(2)}$ 

3	3	1	1	1	2	0
1	1	4	4	4	4	4
4	4	3	3	3	2	3
0	0	0	0	0	0	1

$$\hat{\mathbf{S}} = [2 \ 0 \ 3 \ 1 \ 0 \ 1]$$

$\mathbf{H}^{(1)}$ 

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

 $\mathbf{H}^{(2)}$ 

3	3	1	1	1	2	0
1	1	4	4	4	4	4
4	4	3	3	3	2	3
0	0	0	0	0	0	1

$$\hat{\mathbf{S}} = [2 \ 0 \ 3 \ 1 \ 0 \ 1]$$



$\mathbf{H}^{(1)}$ 

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

 $\mathbf{H}^{(2)}$ 

3	3	1	1	1	2	0
1	1	4	4	4	4	4
4	4	3	3	3	2	3
0	0	0	0	0	0	1

$$\hat{\mathbf{S}} = [2 \ 0 \ 3 \ 1 \ 0 \ 1]$$

$\mathbf{H}^{(1)}$ 

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

 $\mathbf{H}^{(2)}$ 

3	3	1	1	1	2	0
1	1	4	4	4	4	4
4	4	3	3	3	2	3
0	0	0	0	0	0	1

$$\hat{\mathbf{S}} = [2 \ 0 \ 3 \ 1 \ 0 \ 1]$$

$\mathbf{H}^{(1)}$ 

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

 $\mathbf{H}^{(2)}$ 

3	3	1	1	1	2	0
1	1	4	4	4	4	4
4	4	3	3	3	2	3
0	0	0	0	0	0	1

$$\hat{\mathbf{S}} = [2 \ 0 \ 3 \ 1 \ 0 \ 1]$$

$\mathbf{H}^{(1)}$ 

3	1	1	2	0	0
1	3	4	4	3	4
4	4	3	2	5	3
0	0	0	0	0	1

 $\mathbf{H}^{(2)}$ 

3	3	1	1	1	2	0
1	1	4	4	4	4	4
4	4	3	3	3	2	3
0	0	0	0	0	0	1

$$\hat{\mathbf{S}} = [2 \ 0 \ 3 \ 1 \ 0 \ 1]$$

$$\hat{\mathbf{S}} = [2 \ 0 \ 3 \ 1 \ 0 \ 1]$$

**D**<sup>(1)</sup>

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

**D**<sup>(2)</sup>

<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>a</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>d</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>

$$\hat{\mathbf{S}} = [2 \ 0 \ 3 \ 1 \ 0 \ 1]$$

**D**<sup>(1)</sup>

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

**D**<sup>(2)</sup>

<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>a</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>d</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>

# Matching Scheme: Example

$$\hat{\mathbf{S}} = [2 \ 0 \ 3 \ 1 \ 0 \ 1]$$

$\mathbf{D}^{(1)}$

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>c</i>
<i>a</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>d</i>
<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>
<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>c</i>

$\mathbf{D}^{(2)}$

<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>c</i>	<i>b</i>	✓
<i>a</i>	<i>a</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	✓
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>	✗
<i>a</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>d</i>	✓
<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	✗
<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>	✗
<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>b</i>	<i>b</i>	✗
<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>c</i>	✓

## Lemma

Let  $H_i$  denote the  $i^{\text{th}}$  column of the histogram matrix  $\mathbf{H}^{(1)}$ . Then,  $P(\exists i, j \in [n], i \neq j, H_i = H_j) \rightarrow 0$  as  $n \rightarrow \infty$  if  $m_n = \omega\left(n^{\frac{4}{|\mathbb{X}|-1}}\right)$ .



## Lemma

Let  $H_i$  denote the  $i^{\text{th}}$  column of the histogram matrix  $\mathbf{H}^{(1)}$ . Then,  $P(\exists i, j \in [n], i \neq j, H_i = H_j) \rightarrow 0$  as  $n \rightarrow \infty$  if  $m_n = \omega\left(n^{\frac{4}{|\mathcal{X}|-1}}\right)$ .

- For  $R > 0$ ,  $m_n = \omega(n^p) \forall p \in \mathbb{N}$ .
- $\Rightarrow$  Asymptotically, columns of  $\mathbf{H}^{(1)}$  are unique.

## Lemma

Let  $H_i$  denote the  $i^{\text{th}}$  column of the histogram matrix  $\mathbf{H}^{(1)}$ . Then,  $P(\exists i, j \in [n], i \neq j, H_i = H_j) \rightarrow 0$  as  $n \rightarrow \infty$  if  $m_n = \omega\left(n^{\frac{4}{|\mathcal{X}|-1}}\right)$ .

- For  $R > 0$ ,  $m_n = \omega(n^p) \forall p \in \mathbb{N}$ .
- $\Rightarrow$  Asymptotically, columns of  $\mathbf{H}^{(1)}$  are unique.
- When there is no noise, can be matched with  $\mathbf{H}^{(2)}$ .

## Theorem

For the no noise case with column deletion probability  $\delta$ , databases with growth rate  $R$  can be successfully matched if

$$R < (1 - \delta)H(X).$$

Furthermore, a necessary condition for the existence of a successful matching scheme is

$$R \leq (1 - \delta)H(X).$$

## Theorem

For the no noise case with column deletion probability  $\delta$ , databases with growth rate  $R$  can be successfully matched if

$$R < (1 - \delta)H(X).$$

Furthermore, a necessary condition for the existence of a successful matching scheme is

$$R \leq (1 - \delta)H(X).$$

- Any deletion/replica can be converted to **erasure**.
- Deleted/repeated columns do not offer additional information.

## Theorem

For the no noise case with column deletion probability  $\delta$ , databases with growth rate  $R$  can be successfully matched if

$$R < (1 - \delta)H(X).$$

Furthermore, a necessary condition for the existence of a successful matching scheme is

$$R \leq (1 - \delta)H(X).$$

- Any deletion/replica can be converted to **erasure**.
- Deleted/repeated columns do not offer additional information.
- When there is noise:
  - Histograms can no longer be matched.
  - Repeated columns offer additional information.

- How can information theory help in database and graph matching?
- Consider random databases/graphs.
  - Allows using tools from **information theory**, leading to **theoretical guarantees** and **new algorithms**.
- Database matching.
  - Correlated databases.
  - Time series: Deletions and replicas.
    - Noiseless.
    - Noisy.
- General graph matching.
  - Matching identical graphs.
  - Correlated graphs with seeds.
  - Correlated graphs without seeds.
    - New typicality results on permutations of sequences.

# Database Matching Under Noisy Column Repetitions

- Noise & column deletions/replicas.



- Histograms cannot be matched.

# Database Matching Under Noisy Column Repetitions

- Noise & column deletions/replicas.



- Histograms cannot be matched.
  - We cannot replace exact histogram matching with typicality matching.
  - Each column pair chosen from  $\mathbf{H}^{(1)}$  and  $\mathbf{H}^{(2)}$  is jointly-typical with high probability.



# Database Matching Under Noisy Column Repetitions

- Noise & column deletions/replicas.



- Histograms cannot be matched.
  - We cannot replace exact histogram matching with typicality matching.
  - Each column pair chosen from  $\mathbf{H}^{(1)}$  and  $\mathbf{H}^{(2)}$  is jointly-typical with high probability.
- We need new deletion & replica detection algorithms.
  - Noisy replica detection.

# Database Matching Under Noisy Column Repetitions

- Noise & column deletions/replicas.



- Histograms cannot be matched.
  - We cannot replace exact histogram matching with typicality matching.
  - Each column pair chosen from  $\mathbf{H}^{(1)}$  and  $\mathbf{H}^{(2)}$  is jointly-typical with high probability.
- We need new deletion & replica detection algorithms.
  - Noisy replica detection.
  - **Seeded** deletion detection.

## Noisy Column Repetitions: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .

# Noisy Column Repetitions: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .
- $\Theta$ : uniform permutation of  $[m_n]$ .

# Noisy Column Repetitions: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .
- $\Theta$ : uniform permutation of  $[m_n]$ .
- **Column repetition pattern**: random vector  $S^n = \{S_1, S_2, \dots, S_n\}$  with  $S_j \stackrel{i.i.d.}{\sim} p_S$ .
  - $\text{supp}(p_S) = \{0, \dots, s_{\max}\}$

# Noisy Column Repetitions: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .
- $\Theta$ : uniform permutation of  $[m_n]$ .
- **Column repetition pattern**: random vector  $S^n = \{S_1, S_2, \dots, S_n\}$  with  $S_j \stackrel{i.i.d.}{\sim} p_S$ .
  - $\text{supp}(p_S) = \{0, \dots, s_{\max}\}$
- $\mathbf{D}^{(2)}$ : Obtained from  $\mathbf{D}^{(1)}$  by
  - 1 Row shuffling by  $\Theta$ .

# Noisy Column Repetitions: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .
- $\Theta$ : uniform permutation of  $[m_n]$ .
- **Column repetition pattern**: random vector  $S^n = \{S_1, S_2, \dots, S_n\}$  with  $S_j \stackrel{i.i.d.}{\sim} p_S$ .
  - $\text{supp}(p_S) = \{0, \dots, s_{\max}\}$
- $\mathbf{D}^{(2)}$ : Obtained from  $\mathbf{D}^{(1)}$  by
  - 1 Row shuffling by  $\Theta$ .
  - 2 Column deletion/replication by  $S^n$ .
    - Replicate the  $j^{\text{th}}$  column  $S_j$  times if  $S_j > 0$ .
    - Delete the  $j^{\text{th}}$  column if  $S_j = 0$ .

# Noisy Column Repetitions: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .
- $\Theta$ : uniform permutation of  $[m_n]$ .
- **Column repetition pattern**: random vector  $S^n = \{S_1, S_2, \dots, S_n\}$  with  $S_j \stackrel{i.i.d.}{\sim} p_S$ .
  - $\text{supp}(p_S) = \{0, \dots, s_{\max}\}$
- $\mathbf{D}^{(2)}$ : Obtained from  $\mathbf{D}^{(1)}$  by
  - 1 Row shuffling by  $\Theta$ .
  - 2 Column deletion/replication by  $S^n$ .
    - Replicate the  $j^{\text{th}}$  column  $S_j$  times if  $S_j > 0$ .
    - Delete the  $j^{\text{th}}$  column if  $S_j = 0$ .
  - 3 *i.i.d.* noise  $p_{Y|X}$  on the retained entries.



# Noisy Column Repetitions: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .
- $\Theta$ : uniform permutation of  $[m_n]$ .
- **Column repetition pattern**: random vector  $S^n = \{S_1, S_2, \dots, S_n\}$  with  $S_j \stackrel{i.i.d.}{\sim} p_S$ .
  - $\text{supp}(p_S) = \{0, \dots, s_{\max}\}$
- $\mathbf{D}^{(2)}$ : Obtained from  $\mathbf{D}^{(1)}$  by
  - 1 Row shuffling by  $\Theta$ .
  - 2 Column deletion/replication by  $S^n$ .
    - Replicate the  $j^{\text{th}}$  column  $S_j$  times if  $S_j > 0$ .
    - Delete the  $j^{\text{th}}$  column if  $S_j = 0$ .
  - 3 *i.i.d.* noise  $p_{Y|X}$  on the retained entries.
    - $X$  and  $Y$  are not independent:  $p_{Y|X} \neq p_Y$

# Noisy Column Repetitions: System Model

- $\mathbf{D}^{(1)}$ :  $m_n \times n$  random matrix with entries  $X_{i,j} \stackrel{i.i.d.}{\sim} p_X$ .
- $\Theta$ : uniform permutation of  $[m_n]$ .
- **Column repetition pattern**: random vector  $S^n = \{S_1, S_2, \dots, S_n\}$  with  $S_j \stackrel{i.i.d.}{\sim} p_S$ .
  - $\text{supp}(p_S) = \{0, \dots, s_{\max}\}$
- $\mathbf{D}^{(2)}$ : Obtained from  $\mathbf{D}^{(1)}$  by
  - 1 Row shuffling by  $\Theta$ .
  - 2 Column deletion/replication by  $S^n$ .
    - Replicate the  $j^{\text{th}}$  column  $S_j$  times if  $S_j > 0$ .
    - Delete the  $j^{\text{th}}$  column if  $S_j = 0$ .
  - 3 *i.i.d.* noise  $p_{Y|X}$  on the retained entries.
    - $X$  and  $Y$  are not independent:  $p_{Y|X} \neq p_Y$
- **Seeds**: Sub-databases  $(\mathbf{G}^{(1)}, \mathbf{G}^{(2)})$  consisting of  $\Lambda_n$  pairs of correctly-matched rows.

# Proposed Matching Scheme for Noisy Repetitions

- Exploit the identical repetition pattern across rows.

# Proposed Matching Scheme for Noisy Repetitions

- Exploit the identical repetition pattern across rows.
  - ① Find a permutation-invariant unique feature of the columns of  $\mathbf{D}^{(2)}$ .

# Proposed Matching Scheme for Noisy Repetitions

- Exploit the identical repetition pattern across rows.
  - ① Find a permutation-invariant unique feature of the columns of  $\mathbf{D}^{(2)}$ .
  - ② By threshold testing these features, infer the noisy replicas.

# Proposed Matching Scheme for Noisy Repetitions

- Exploit the identical repetition pattern across rows.
  - ① Find a permutation-invariant unique feature of the columns of  $\mathbf{D}^{(2)}$ .
  - ② By threshold testing these features, infer the noisy replicas.
  - ③ Using the seeds  $(\mathbf{G}^{(1)}, \mathbf{G}^{(2)})$ , extract the deletion pattern.

- Exploit the identical repetition pattern across rows.
  - ① Find a permutation-invariant unique feature of the columns of  $\mathbf{D}^{(2)}$ .
  - ② By threshold testing these features, infer the noisy replicas.
  - ③ Using the seeds  $(\mathbf{G}^{(1)}, \mathbf{G}^{(2)})$ , extract the deletion pattern.
  - ④ Group the noisy replica runs by introducing markers between the columns of  $\mathbf{D}^{(2)}$ .
  - ⑤ Replace the deleted columns with erasure symbols in  $\mathbf{D}^{(2)}$ .

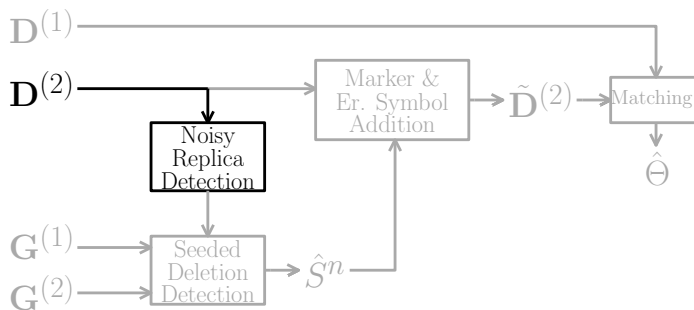
- Exploit the identical repetition pattern across rows.
  - 1 Find a permutation-invariant unique feature of the columns of  $\mathbf{D}^{(2)}$ .
  - 2 By threshold testing these features, infer the noisy replicas.
  - 3 Using the seeds ( $\mathbf{G}^{(1)}, \mathbf{G}^{(2)}$ ), extract the deletion pattern.
  - 4 Group the noisy replica runs by introducing markers between the columns of  $\mathbf{D}^{(2)}$ .
  - 5 Replace the deleted columns with erasure symbols in  $\mathbf{D}^{(2)}$ .
  - 6 Perform a typicality-based rowwise matching.



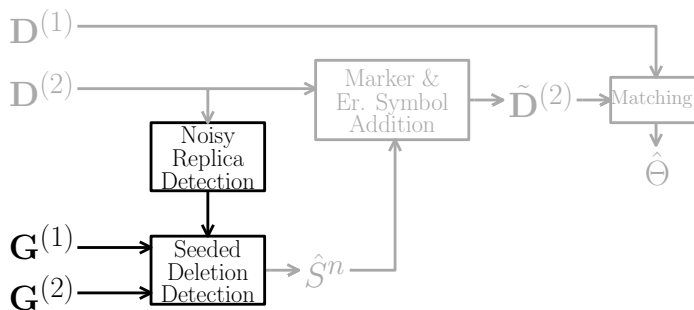
# Proposed Matching Scheme for Noisy Repetitions

- Exploit the identical repetition pattern across rows.
  - ① Find a permutation-invariant unique feature of the columns of  $\mathbf{D}^{(2)}$ .
  - ② By threshold testing these features, infer the noisy replicas.
  - ③ Using the seeds ( $\mathbf{G}^{(1)}, \mathbf{G}^{(2)}$ ), extract the deletion pattern.
  - ④ Group the noisy replica runs by introducing markers between the columns of  $\mathbf{D}^{(2)}$ .
  - ⑤ Replace the deleted columns with erasure symbols in  $\mathbf{D}^{(2)}$ .
  - ⑥ Perform a typicality-based rowwise matching.
- We will use the *Hamming distances between the consecutive columns of  $\mathbf{D}^{(2)}$*  as the permutation-invariant feature.

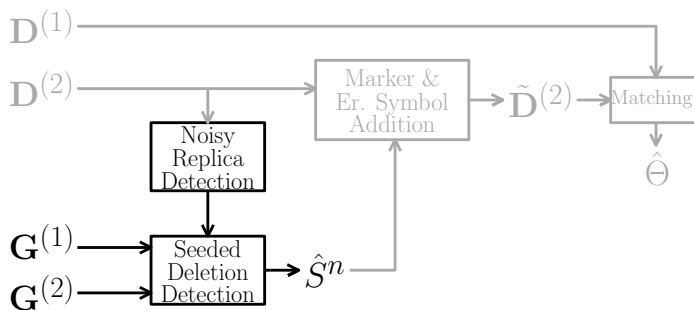
# Proposed Matching Scheme



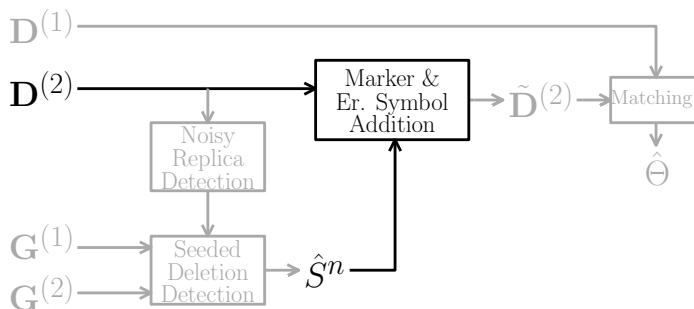
# Proposed Matching Scheme



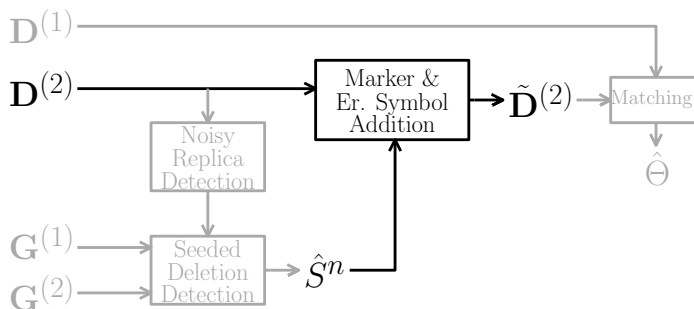
# Proposed Matching Scheme



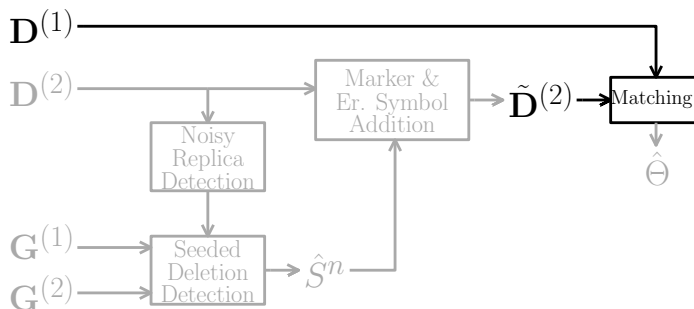
# Proposed Matching Scheme



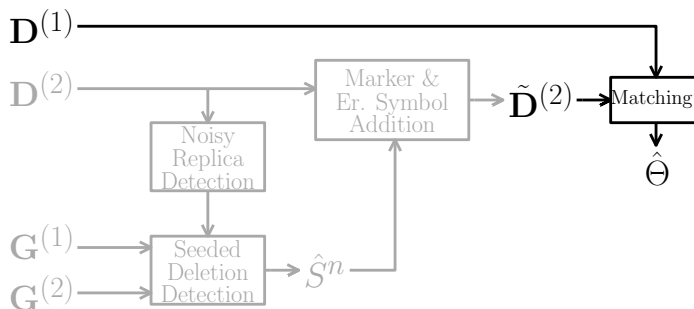
# Proposed Matching Scheme



# Proposed Matching Scheme



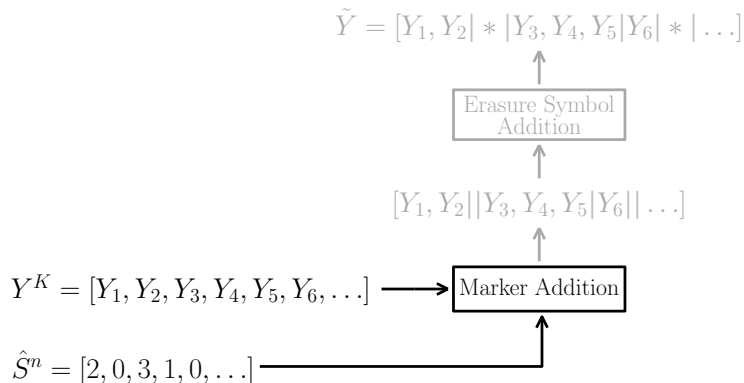
# Proposed Matching Scheme





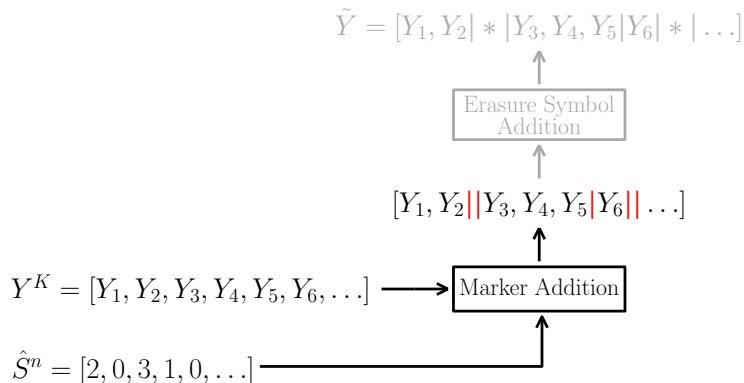
# Marker & Erasure Symbol Addition

$Y^K$ : a row of  $\mathbf{D}^{(2)}$ .



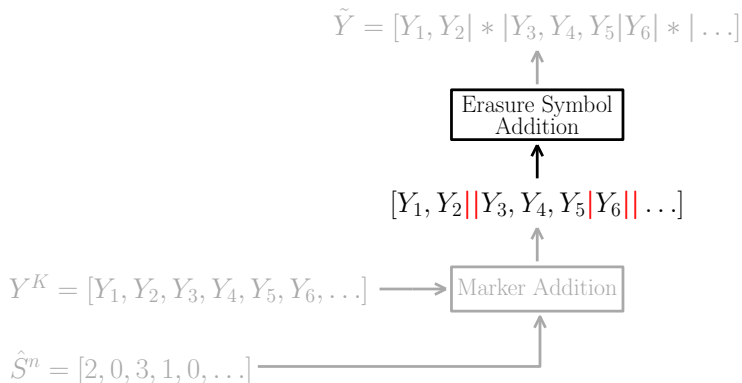
# Marker & Erasure Symbol Addition

$Y^K$ : a row of  $\mathbf{D}^{(2)}$ .



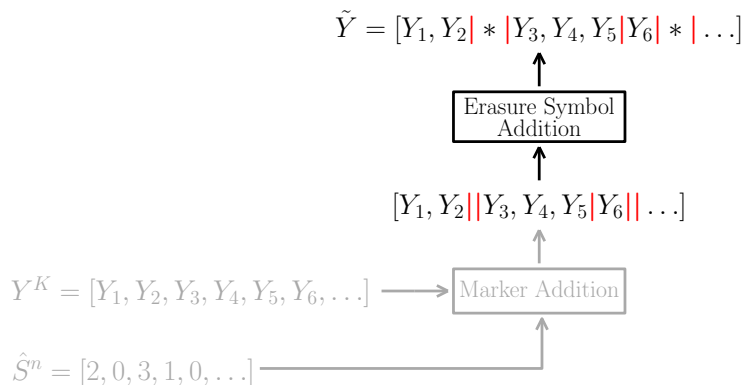
# Marker & Erasure Symbol Addition

$Y^K$ : a row of  $\mathbf{D}^{(2)}$ .



# Marker & Erasure Symbol Addition

$Y^K$ : a row of  $\mathbf{D}^{(2)}$ .



$\tilde{Y}$ : the corresponding row of  $\tilde{\mathbf{D}}^{(2)}$ .

## Theorem

In the noisy column repetition case, for seed size  $\Lambda_n$  linear with  $n$ , databases with growth rate  $R$  can be successfully matched if

$$R < I(X; Y^S, S)$$

where  $S \sim p_S$  and  $Y^S = Y_1, \dots, Y_S$  such that

$$\Pr(Y^S = y_1, \dots, y_S | X = x) = \prod_{i=1}^S p_{Y|X}(y_i|x).$$

Furthermore, a necessary condition for the existence of a successful matching scheme is

$$R \leq I(X; Y^S, S)$$

- Since  $X$  and  $S$  are independent

$$R = I(X; Y^S, S) = I(X; Y^S | S).$$

- We can achieve database growth rates as if we knew the repetition pattern  $S^n$  **a-priori**.

- Since  $X$  and  $S$  are independent

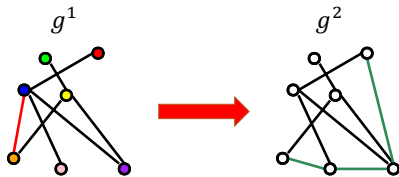
$$R = I(X; Y^S, S) = I(X; Y^S | S).$$

- We can achieve database growth rates as if we knew the repetition pattern  $S^n$  **a-priori**.
- Requires linear seed size  $\Lambda_n$  in  $n$ .
  - $m_n$ : Exponential in  $n$ .

- How can information theory help in database and graph matching?
- Consider random databases/graphs.
  - Allows using tools from **information theory**, leading to **theoretical guarantees** and **new algorithms**.
- Database matching.
  - Correlated databases.
  - Time series: Deletions and replicas.
    - Noiseless.
    - Noisy.
- **General graph matching.**
  - **Matching identical graphs.**
  - Correlated graphs with seeds.
  - Correlated graphs without seeds.
    - New typicality results on permutations of sequences.

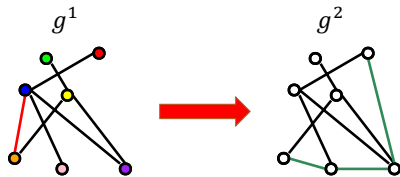


# Random Graph Matching



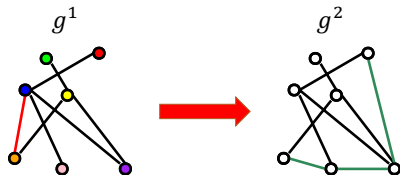
- Graph  $g^i = (\mathcal{V}^i, \mathcal{E}^i), i \in [1, 2]$ :
  - 1  $\mathcal{V}^i$ : Vertex set,  $\{v_1^i, v_2^i, \dots, v_n^i\}$ .
  - 2  $\mathcal{E}^i$ : Edge set, subset of  $\mathcal{V} \times \mathcal{V}$ .

# Random Graph Matching



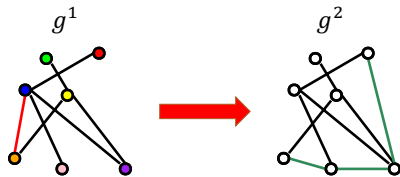
- Graph  $g^i = (\mathcal{V}^i, \mathcal{E}^i), i \in [1, 2]$ :
  - 1  $\mathcal{V}^i$ : Vertex set,  $\{v_1^i, v_2^i, \dots, v_n^i\}$ .
  - 2  $\mathcal{E}^i$ : Edge set, subset of  $\mathcal{V} \times \mathcal{V}$ .
- Labeling  $\sigma^i : \mathcal{V}^i \rightarrow [1, n]$ .

# Random Graph Matching



- Graph  $g^i = (\mathcal{V}^i, \mathcal{E}^i), i \in [1, 2]$ :
  - 1  $\mathcal{V}^i$ : Vertex set,  $\{v_1^i, v_2^i, \dots, v_n^i\}$ .
  - 2  $\mathcal{E}^i$ : Edge set, subset of  $\mathcal{V} \times \mathcal{V}$ .
- Labeling  $\sigma^i : \mathcal{V}^i \rightarrow [1, n]$ .
- Matching edges generated i.i.d. based on  $P_{X_1, X_2}$ .
  - Binary valued edges: Erdős-Rényi graph,  $Er(n, p)$ .

# Random Graph Matching

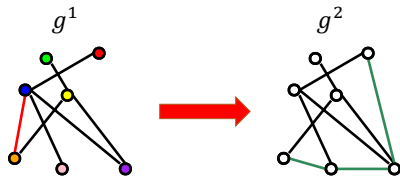


- Graph  $g^i = (\mathcal{V}^i, \mathcal{E}^i), i \in [1, 2]$ :
  - 1  $\mathcal{V}^i$ : Vertex set,  $\{v_1^i, v_2^i, \dots, v_n^i\}$ .
  - 2  $\mathcal{E}^i$ : Edge set, subset of  $\mathcal{V} \times \mathcal{V}$ .
- Labeling  $\sigma^i : \mathcal{V}^i \rightarrow [1, n]$ .
- Matching edges generated i.i.d. based on  $P_{X_1, X_2}$ .
  - Binary valued edges: Erdős-Rényi graph,  $Er(n, p)$ .
- Objective: Given  $(g^1, g^2)$  and  $\sigma^1$ , find  $\hat{\sigma}^2$  s.t.:

$$P(\sigma^2(I) = \hat{\sigma}^2(I)) \rightarrow 1 \text{ as } n \rightarrow \infty,$$

where  $I \sim U(1, n)$ .

# Random Graph Matching



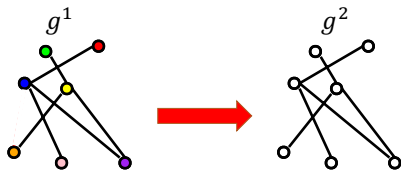
- Graph  $g^i = (\mathcal{V}^i, \mathcal{E}^i), i \in [1, 2]$ :
  - 1  $\mathcal{V}^i$ : Vertex set,  $\{v_1^i, v_2^i, \dots, v_n^i\}$ .
  - 2  $\mathcal{E}^i$ : Edge set, subset of  $\mathcal{V} \times \mathcal{V}$ .
- Labeling  $\sigma^i : \mathcal{V}^i \rightarrow [1, n]$ .
- Matching edges generated i.i.d. based on  $P_{X_1, X_2}$ .
  - Binary valued edges: Erdős-Rényi graph,  $Er(n, p)$ .
- Objective: Given  $(g^1, g^2)$  and  $\sigma^1$ , find  $\hat{\sigma}^2$  s.t.:

$$P(\sigma^2(I) = \hat{\sigma}^2(I)) \rightarrow 1 \text{ as } n \rightarrow \infty,$$

where  $I \sim U(1, n)$ .

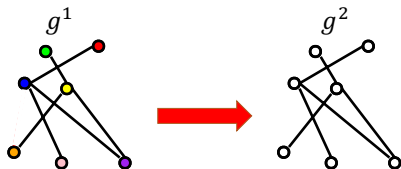
- Almost all indices should be matched correctly.

# Graph Isomorphism: Matching Identical Graphs



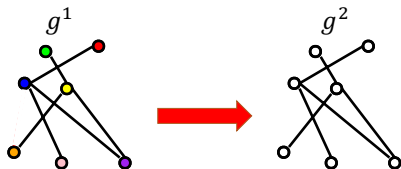
- Two identical graphs  $g^1$  and  $g^2$ .
- Erdős-Rényi graph.
  - 1  $n$  vertices.
  - 2 Edge probability  $p_n$ .
  - 3 Denoted as  $Er(n, p_n)$ .

# Graph Isomorphism: Matching Identical Graphs



- Two identical graphs  $g^1$  and  $g^2$ .
- Erdős-Rényi graph.
  - 1  $n$  vertices.
  - 2 Edge probability  $p_n$ .
  - 3 Denoted as  $Er(n, p_n)$ .
- $g^1$  has vertex labels,  $g^2$  doesn't.
- How to restore the labels for  $g^2$ ?

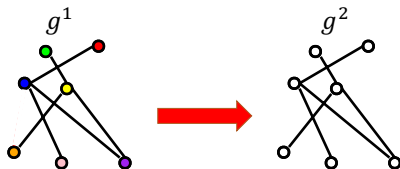
# Graph Isomorphism: Matching Identical Graphs



- Two identical graphs  $g^1$  and  $g^2$ .
- Erdős-Rényi graph.
  - 1  $n$  vertices.
  - 2 Edge probability  $p_n$ .
  - 3 Denoted as  $Er(n, p_n)$ .
- $g^1$  has vertex labels,  $g^2$  doesn't.
- How to restore the labels for  $g^2$ ?
- Solvable iff graph has trivial automorphism group.



# Graph Isomorphism: Matching Identical Graphs



- Two identical graphs  $g^1$  and  $g^2$ .
- Erdős-Rényi graph.
  - 1  $n$  vertices.
  - 2 Edge probability  $p_n$ .
  - 3 Denoted as  $Er(n, p_n)$ .
- $g^1$  has vertex labels,  $g^2$  doesn't.
- How to restore the labels for  $g^2$ ?
- Solvable iff graph has trivial automorphism group.
- Are there polynomial time algorithms for matching?

- Matching can be done iff  $p_n \in [\frac{\log n}{n}, 1 - \frac{\log n}{n}]$  [Erdős and Rényi '61, Wright '73].
  - When  $p_n$  is very small, disconnected vertices.

- Matching can be done iff  $p_n \in \left[\frac{\log n}{n}, 1 - \frac{\log n}{n}\right]$  [Erdős and Rényi '61, Wright '73].
  - When  $p_n$  is very small, disconnected vertices.
- Polynomial time algorithms if  $p_n \in \left[\Theta\left(\frac{\log n}{n}\right), 1 - \Theta\left(\frac{\log n}{n}\right)\right]$ .

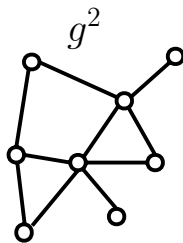
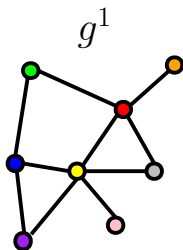
- Matching can be done iff  $p_n \in [\frac{\log n}{n}, 1 - \frac{\log n}{n}]$  [Erdős and Rényi '61, Wright '73].
  - When  $p_n$  is very small, disconnected vertices.
- Polynomial time algorithms if  $p_n \in [\Theta(\frac{\log n}{n}), 1 - \Theta(\frac{\log n}{n})]$ .
  - 1 **Maximum Degree:**  $p_n \in [\omega(n^{-\frac{1}{5}} \log n), \frac{1}{2}]$  [Bollobas '01].
  - 2 **Degree Sequences:**  $p_n \in [\Theta(\frac{\log n}{n}), o(n^{-\frac{11}{12}})]$  [Bollobas '82].
  - 3 **Neighboring Degrees:**  $p_n \in [\omega(\frac{\log^4 n}{n \log \log n}), \frac{1}{2}]$  [Pandurangan '07].

- Matching can be done iff  $p_n \in [\frac{\log n}{n}, 1 - \frac{\log n}{n}]$  [Erdős and Rényi '61, Wright '73].
  - When  $p_n$  is very small, disconnected vertices.
- Polynomial time algorithms if  $p_n \in [\Theta(\frac{\log n}{n}), 1 - \Theta(\frac{\log n}{n})]$ .
  - 1 **Maximum Degree:**  $p_n \in [\omega(n^{-\frac{1}{5}} \log n), \frac{1}{2}]$  [Bollobas '01].
  - 2 **Degree Sequences:**  $p_n \in [\Theta(\frac{\log n}{n}), o(n^{-\frac{11}{12}})]$  [Bollobas '82].
  - 3 **Neighboring Degrees:**  $p_n \in [\omega(\frac{\log^4 n}{n \log \log n}), \frac{1}{2}]$  [Pandurangan '07].
- Focus on **Maximum Degree**:
  - Relation to correlated graphs with seeds and database matching.
  - Alternate proofs using information theoretic tools.

# Maximum Degree Algorithm

- **Step 1:** Order all vertices based on degrees:

$$g^1 : v_{(1)}^1, v_{(2)}^1, \dots, v_{(n)}^1, \quad d(v_{(i)}^1) \leq d(v_{(i+1)}^1)$$
$$g^2 : v_{(1)}^2, v_{(2)}^2, \dots, v_{(n)}^2, \quad d(v_{(i)}^2) \leq d(v_{(i+1)}^2).$$

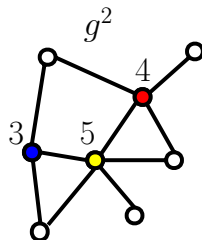
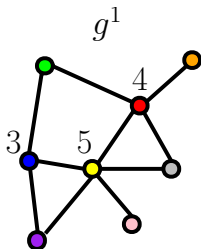


# Maximum Degree Algorithm

- **Step 2:** Match highest degree vertices that are not repeated:

$$\text{If } \forall i \leq j \quad d(v_{(j-1)}^1) \neq d(v_{(j)}^1) \Rightarrow v_{(i)}^1 \sim v_{(i)}^2.$$

Assume that  $m$  vertices are matched at this step.

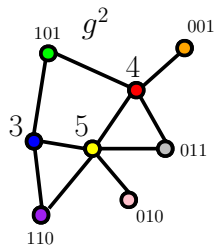
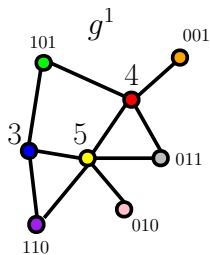


# Maximum Degree Algorithm

- **Step 3:** For each of the remaining vertices, construct the binary signature  $c^m$ :

$$c_i = \begin{cases} 1 & \text{if } v \text{ is connected to } v_{(i+(n-m))} \\ 0 & \text{Otherwise.} \end{cases}$$

If  $c^m$  uniquely identifies  $v$ , match the vertex.





- For Max Degree to give the correct matching
  - Signatures  $c^m$  should be unique.
  - Possible only for large  $m$ .

- For Max Degree to give the correct matching
  - Signatures  $c^m$  should be unique.
  - Possible only for large  $m$ .
- For an Erdős-Rényi graph with edge probability  $p_n$ :
  - 1 Find  $m_1$ , number of vertices with maximum unique degrees.

- For Max Degree to give the correct matching
  - Signatures  $c^m$  should be unique.
  - Possible only for large  $m$ .
- For an Erdős-Rényi graph with edge probability  $p_n$ :
  - 1 Find  $m_1$ , number of vertices with maximum unique degrees.
  - 2 Find smallest  $m_2$ , such that signatures of length  $m_2$  are unique.

- For Max Degree to give the correct matching
  - Signatures  $c^m$  should be unique.
  - Possible only for large  $m$ .
- For an Erdős-Rényi graph with edge probability  $p_n$ :
  - 1 Find  $m_1$ , number of vertices with maximum unique degrees.
  - 2 Find smallest  $m_2$ , such that signatures of length  $m_2$  are unique.
  - 3  $m_2 \leq m_1 \Rightarrow$  **success**.

- Number of vertices with maximum unique degrees is

$$m_1 = O\left(\frac{(p_n(1-p_n)n)^{\frac{1}{4}}}{\log^{\frac{1}{4}} n}\right).$$

- Number of vertices with maximum unique degrees is

$$m_1 = O\left(\frac{(p_n(1-p_n)n)^{\frac{1}{4}}}{\log^{\frac{1}{4}} n}\right).$$

*Proof:* Shown by [Bollobas '01]. We prove using method of types.

- Number of vertices with maximum unique degrees is

$$m_1 = O\left(\frac{(p_n(1-p_n)n)^{\frac{1}{4}}}{\log^{\frac{1}{4}} n}\right).$$

*Proof:* Shown by [Bollobas '01]. We prove using method of types.

- Set of all vertices with degree  $np_n + t$ : Type  $p_n + \frac{t}{n}$  vertices.

- Number of vertices with maximum unique degrees is

$$m_1 = O\left(\frac{(p_n(1-p_n)n)^{\frac{1}{4}}}{\log^{\frac{1}{4}} n}\right).$$

*Proof:* Shown by [Bollobas '01]. We prove using method of types.

- Set of all vertices with degree  $np_n + t$ : Type  $p_n + \frac{t}{n}$  vertices.
- Probability of  $v$  being of type  $p_n + \frac{t}{n}$

$$2^{-nD(p_n + \frac{t}{n} || p_n)}.$$



- Number of vertices with maximum unique degrees is

$$m_1 = O\left(\frac{(p_n(1-p_n)n)^{\frac{1}{4}}}{\log^{\frac{1}{4}} n}\right).$$

*Proof:* Shown by [Bollobas '01]. We prove using method of types.

- Set of all vertices with degree  $np_n + t$ : Type  $p_n + \frac{t}{n}$  vertices.
- Probability of  $v$  being of type  $p_n + \frac{t}{n}$

$$2^{-nD(p_n + \frac{t}{n} || p_n)}.$$

- Approximate  $D(p_n + \frac{t}{n} || p_n)$ , and find  $t$  such that expected number of vertices of type  $p_n + \frac{t}{n}$  is 1.

- Smallest signature length that guarantees unique signatures is

$$m_2 = \frac{\log n}{H_2(p_n)},$$

where  $H_2(\cdot)$  is the Rényi entropy with parameter 2.

- Smallest signature length that guarantees unique signatures is

$$m_2 = \frac{\log n}{H_2(p_n)},$$

where  $H_2(\cdot)$  is the Rényi entropy with parameter 2.

*Proof:* See Shirani, Garg, Erkip, Allerton 2017.

- Smallest signature length that guarantees unique signatures is

$$m_2 = \frac{\log n}{H_2(p_n)},$$

where  $H_2(\cdot)$  is the Rényi entropy with parameter 2.

*Proof:* See Shirani, Garg, Erkip, Allerton 2017.

- Combining: Max Degree finds the correct matching if

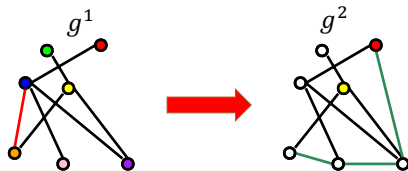
$$\frac{\log n}{H_2(p_n)} < O\left(\frac{(p_n(1-p_n)n)^{\frac{1}{4}}}{\log^{\frac{1}{4}} n}\right),$$

which is equivalent to

$$p_n \in [\omega(n^{-\frac{1}{5}} \log n), 1 - \omega(n^{-\frac{1}{5}} \log n)].$$

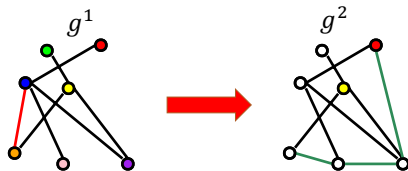
- How can information theory help in database and graph matching?
- Consider random databases/graphs.
  - Allows using tools from **information theory**, leading to **theoretical guarantees** and **new algorithms**.
- Database matching.
  - Correlated databases.
  - Time series: Deletions and replicas.
    - Noiseless.
    - Noisy.
- **General graph matching.**
  - Matching identical graphs.
  - **Correlated graphs with seeds.**
  - Correlated graphs without seeds.
    - New typicality results on permutations of sequences.

# Random Graphs with Seeds



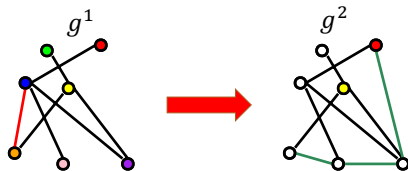
- **Seeds:** Some vertex labels in  $g^2$  are given.

# Random Graphs with Seeds



- **Seeds:** Some vertex labels in  $g^2$  are given.
- **Objective:** Match the remaining vertices in  $g^2$ .

# Random Graphs with Seeds



- **Seeds:** Some vertex labels in  $g^2$  are given.
- Objective: Match the remaining vertices in  $g^2$ .
- **Question:** Conditions on seed size and graph statistics?

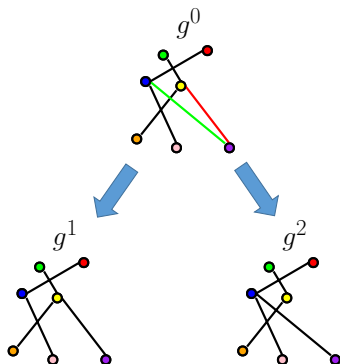


- Rich literature on *network alignment*, matching two realized networks.

## Related Literature

- Rich literature on *network alignment*, matching two realized networks.
- Recent work on matching random graphs:
  - Mostly consider the edge erasure model.

- Rich literature on *network alignment*, matching two realized networks.
- Recent work on matching random graphs:
  - Mostly consider the edge erasure model.
  - Original graph  $g^0 \sim Er(n, p_n)$ .
  - Edge erasure probability is  $1 - s_n$ .



- No seeds:
  - [Pedarsani, Grossglauser, 2011]:  $\frac{p_n s_n^3}{8(2-s_n)} = \frac{\log n}{n} + \omega(n^{-1})$ .
  - [Cullina, Kiyavash, 2016]:  $\frac{p_n s_n^2}{2} = \frac{\log n}{n} + \omega(n^{-1})$ .
  - [Onaran, Garg, Erkip 2016]: If  $s_n = \omega(n^{-1})$ , matching possible if  $p_n s_n (1 - \sqrt{1 - s_n^2}) = 3 \frac{\log n}{n} + \omega(n^{-1})$ .

- No seeds:
  - [Pedarsani, Grossglauser, 2011]:  $\frac{p_n s_n^3}{8(2-s_n)} = \frac{\log n}{n} + \omega(n^{-1})$ .
  - [Cullina, Kiyavash, 2016]:  $\frac{p_n s_n^2}{2} = \frac{\log n}{n} + \omega(n^{-1})$ .
  - [Onaran, Garg, Erkip 2016]: If  $s_n = \omega(n^{-1})$ , matching possible if  $p_n s_n (1 - \sqrt{1 - s_n^2}) = 3 \frac{\log n}{n} + \omega(n^{-1})$ .
  - [Cullina, Kiyavash, 2017]: Extend results to general binary distribution  $P_{X_1, X_2}$ .

- No seeds:
  - [Pedarsani, Grossglauser, 2011]:  $\frac{p_n s_n^3}{8(2-s_n)} = \frac{\log n}{n} + \omega(n^{-1})$ .
  - [Cullina, Kiyavash, 2016]:  $\frac{p_n s_n^2}{2} = \frac{\log n}{n} + \omega(n^{-1})$ .
  - [Onaran, Garg, Erkip 2016]: If  $s_n = \omega(n^{-1})$ , matching possible if  $p_n s_n (1 - \sqrt{1 - s_n^2}) = 3 \frac{\log n}{n} + \omega(n^{-1})$ .
  - [Cullina, Kiyavash, 2017]: Extend results to general binary distribution  $P_{X_1, X_2}$ .
- With seeds, seed size  $|\Lambda|$ :
  - Kazemi, Hassani, Grossglauser '15:  $p \in [\frac{1}{n}, n^{-\frac{5}{6}}]$  and

$$|\Lambda| > \left(1 - \frac{1}{r}\right) \left(\frac{(r-1)!}{n(p s^2)^r}\right)^{\frac{1}{1-r}},$$

where  $r$  is an arbitrary coefficient.

- Beyah et. al.'16:  $|\Lambda| \geq \frac{4(2 \log n + 1)(2 - s - ps)}{ps^3(1-p)^2}$ .

- No seeds:
  - [Pedarsani, Grossglauser, 2011]:  $\frac{p_n s_n^3}{8(2-s_n)} = \frac{\log n}{n} + \omega(n^{-1})$ .
  - [Cullina, Kiyavash, 2016]:  $\frac{p_n s_n^2}{2} = \frac{\log n}{n} + \omega(n^{-1})$ .
  - [Onaran, Garg, Erkip 2016]: If  $s_n = \omega(n^{-1})$ , matching possible if  $p_n s_n (1 - \sqrt{1 - s_n^2}) = 3 \frac{\log n}{n} + \omega(n^{-1})$ .
  - [Cullina, Kiyavash, 2017]: Extend results to general binary distribution  $P_{X_1, X_2}$ .
- With seeds, seed size  $|\Lambda|$ :
  - Kazemi, Hassani, Grossglauser '15:  $p \in [\frac{1}{n}, n^{-\frac{5}{6}}]$  and

$$|\Lambda| > \left(1 - \frac{1}{r}\right) \left(\frac{(r-1)!}{n(p s^2)^r}\right)^{\frac{1}{1-r}},$$

where  $r$  is an arbitrary coefficient.

- Beyah et. al.'16:  $|\Lambda| \geq \frac{4(2 \log n + 1)(2 - s - ps)}{ps^3(1-p)^2}$ .
- These works mostly analyze the performance of Maximum-a-Posteriori (MAP) matching.

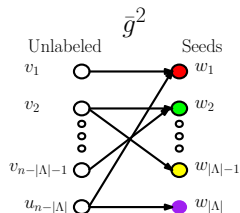
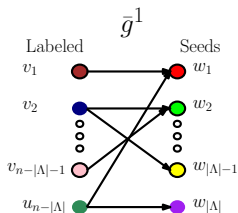
# Our Seeded Graph Matching Algorithm

- Let  $w_1, w_2, \dots, w_{|\Lambda|}$  be the labeled seeds.



# Our Seeded Graph Matching Algorithm

- Let  $w_1, w_2, \dots, w_{|\Lambda|}$  be the labeled seeds.
- **Step 1:** Construct the reduced bipartite graph connecting the unmatched vertices to seeds.



# Our Seeded Graph Matching Algorithm

- **Step 2:** Construct the adjacency matrices of the bipartite graphs corresponding to  $g^1$  and  $g^2$ .

$g^1$

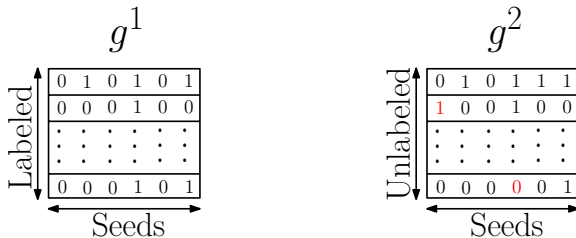
Labeled	0	1	0	1	0	1
	0	0	0	1	0	0
	⋮	⋮	⋮	⋮	⋮	⋮
	⋮	⋮	⋮	⋮	⋮	⋮
	0	0	0	1	0	1
	Seeds					

$g^2$

Unlabeled	0	1	0	1	1	1
	1	0	0	1	0	0
	⋮	⋮	⋮	⋮	⋮	⋮
	⋮	⋮	⋮	⋮	⋮	⋮
	0	0	0	0	0	1
	Seeds					

# Our Seeded Graph Matching Algorithm

- **Step 2:** Construct the adjacency matrices of the bipartite graphs corresponding to  $g^1$  and  $g^2$ .



- Analogy to database matching:

Database Matching	Graph Matching
Rows (User IDs)	Unseeded Vertices
Columns (Attributes)	Seeded Vertices
Entries	Edges

# Our Seeded Graph Matching Algorithm

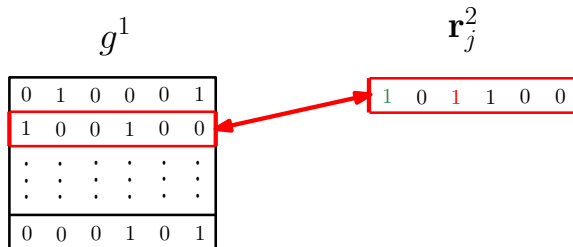
- **Step 3:** Find the unique row  $\mathbf{r}_k^1$  in  $g^1$  which is jointly typical with  $\mathbf{r}_j^2$  with respect to  $p_{X_1, X_2}$ .

$g^1$						$\mathbf{r}_j^2$					
0	1	0	0	0	1	1	0	1	1	0	0
1	0	0	1	0	0	1	0	1	1	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	0	1	0	1	⋮	⋮	⋮	⋮	⋮	⋮

# Our Seeded Graph Matching Algorithm

- **Step 3:** Find the unique row  $\mathbf{r}_k^1$  in  $g^1$  which is jointly typical with  $\mathbf{r}_j^2$  with respect to  $p_{X_1, X_2}$ .

$g^1$	$\mathbf{r}_j^2$
0 1 0 0 0 1	1 0 1 1 0 0
1 0 0 1 0 0	
⋮ ⋮ ⋮ ⋮ ⋮ ⋮	
0 0 0 1 0 1	



- If such a row does not exist or not unique, algorithm fails.
- If successful, add the identified vertex as a seed, and repeat steps 1, 2 until matching is complete.

## Theorem

For a pair of Erdős-Rényi graphs  $(g^1, g^2)$  characterized by  $p_{X_1, X_2}$ , the above algorithm succeeds in matching if  $|\Lambda| > \frac{\log n}{I(X_1; X_2)}$ .

- The proof is similar database matching.

## Theorem

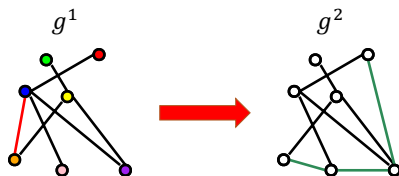
For a pair of Erdős-Rényi graphs  $(g^1, g^2)$  characterized by  $p_{X_1, X_2}$ , the above algorithm succeeds in matching if  $|\Lambda| > \frac{\log n}{I(X_1; X_2)}$ .

- The proof is similar database matching.
- $I(X_1; X_2)$  accounts for reduction in information each seed node provides.
  - Binary valued edges:  $I(X_1; X_2) \leq 1$
- Shirani, Garg, Erkip, JSAC 2021.

- How can information theory help in database and graph matching?
- Consider random databases/graphs.
  - Allows using tools from **information theory**, leading to **theoretical guarantees** and **new algorithms**.
- Database matching.
  - Correlated databases.
  - Time series: Deletions and replicas.
    - Noiseless.
    - Noisy.
- **General graph matching.**
  - Matching identical graphs.
  - Correlated graphs with seeds.
  - **Correlated graphs without seeds.**
    - **New typicality results on permutations of sequences.**

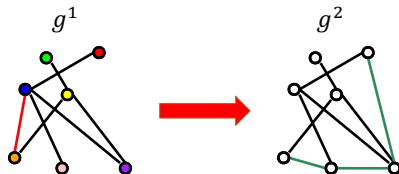


# Graph Matching without Seeds



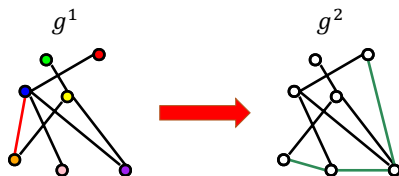
- Edges are pairwise correlated.

# Graph Matching without Seeds



- Edges are pairwise correlated.
- There are no seeds.

# Graph Matching without Seeds

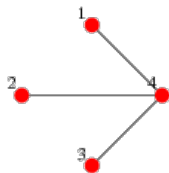


- Edges are pairwise correlated.
- There are no seeds.
- **Question:** Conditions for successful graph matching?

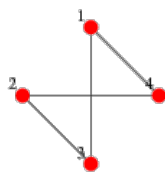
- Consider the adjacency matrix of a graph  $G_\sigma$ .

# Adjacency Matrix

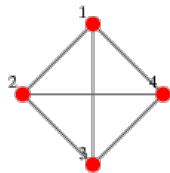
- Consider the adjacency matrix of a graph  $G_\sigma$ .



$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$



$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$



$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

- The adjacency matrix is symmetric.

# Labelings and Adjacency Matrix

$$\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{cccc} 1 & 2 & 3 & 4 \\ \left[ \begin{array}{cccc} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{array} \right] \end{array}$$



The diagram shows an adjacency matrix with red arrows indicating a permutation of rows. The matrix is:

$$\begin{array}{c} 3 \\ 2 \\ 1 \\ 4 \end{array} \begin{array}{cccc} 3 & 2 & 1 & 4 \\ \left[ \begin{array}{cccc} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array} \right] \end{array}$$

Red arrows show the mapping: 3 → 1, 2 → 2, 1 → 3, and 4 → 4.

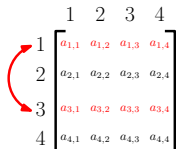
- Each labeling gives a permutation of  $G_\sigma$ .

# Labelings and Adjacency Matrix

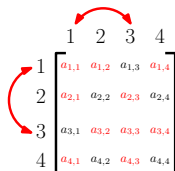
$$\begin{array}{c} 1 \ 2 \ 3 \ 4 \\ \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$


$$\begin{array}{c} 3 \ 2 \ 1 \ 4 \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

- Each labeling gives a permutation of  $G_\sigma$ .
- In databases, mislabeling entries only affects the corresponding rows.
- In graphs, mislabeling nodes permutes the rows and columns of the adjacency matrix.


$$\begin{array}{c} 1 \ 2 \ 3 \ 4 \\ \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} \end{array}$$

Database


$$\begin{array}{c} 1 \ 2 \ 3 \ 4 \\ \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} \end{array}$$

Graph

- Database matching can be done by matching each row of  $\mathcal{C}^{(2)}$  separately.



- Database matching can be done by matching each row of  $\mathcal{C}^{(2)}$  separately.
  - Typicality matching.

- Database matching can be done by matching each row of  $\mathcal{C}^{(2)}$  separately.
  - Typicality matching.
- For graph matching need to match the complete adjacency matrices.

- Database matching can be done by matching each row of  $\mathcal{C}^{(2)}$  separately.
  - Typicality matching.
- For graph matching need to match the complete adjacency matrices.
  - Typicality of adjacency matrices.

- Consider the upper triangle adjacency matrix  $[G_{\sigma,i,j}]_{i<j}$ .

# Typicality of Erdős-Rényi Graphs

- Consider the upper triangle adjacency matrix  $[G_{\sigma,i,j}]_{i<j}$ .
- Can think of  $[G_{\sigma,i,j}]_{i<j}$  as a  $\frac{n(n-1)}{2}$  sequence of bits.

# Typicality of Erdős-Rényi Graphs

- Consider the upper triangle adjacency matrix  $[G_{\sigma,i,j}]_{i<j}$ .
- Can think of  $[G_{\sigma,i,j}]_{i<j}$  as a  $\frac{n(n-1)}{2}$  sequence of bits.
  - If  $g \sim Er(n, p)$ , then  $[G_{\sigma,i,j}]_{i<j}$  is a vector of i.i.d. Bern(p) random variables.

# Typicality of Erdős-Rényi Graphs

- Consider the upper triangle adjacency matrix  $[G_{\sigma,i,j}]_{i<j}$ .
- Can think of  $[G_{\sigma,i,j}]_{i<j}$  as a  $\frac{n(n-1)}{2}$  sequence of bits.
  - If  $g \sim Er(n, p)$ , then  $[G_{\sigma,i,j}]_{i<j}$  is a vector of i.i.d. Bern(p) random variables.
- Recall:  $A_{\epsilon}^n(X)$ :  $\epsilon$ -typical set of length  $n$  sequences w.r.t  $P_X$ .

# Typicality of Erdős-Rényi Graphs

- Consider the upper triangle adjacency matrix  $[G_{\sigma,i,j}]_{i<j}$ .
- Can think of  $[G_{\sigma,i,j}]_{i<j}$  as a  $\frac{n(n-1)}{2}$  sequence of bits.
  - If  $g \sim Er(n, p)$ , then  $[G_{\sigma,i,j}]_{i<j}$  is a vector of i.i.d. Bern(p) random variables.
- Recall:  $A_\epsilon^n(X)$ :  $\epsilon$ -typical set of length  $n$  sequences w.r.t  $P_X$ .
- **Result:**  $[G_{\sigma,i,j}]_{i<j}$  is  $\epsilon$ -typical w.r.t Bern(p) with probability one:

$$P([G_{\sigma,i,j}]_{i<j} \in A_\epsilon^{\frac{n(n-1)}{2}}(X)) \rightarrow 1, n \rightarrow \infty.$$



# Typicality of Erdős-Rényi Graphs

- Consider the upper triangle adjacency matrix  $[G_{\sigma,i,j}]_{i<j}$ .
- Can think of  $[G_{\sigma,i,j}]_{i<j}$  as a  $\frac{n(n-1)}{2}$  sequence of bits.
  - If  $g \sim Er(n, p)$ , then  $[G_{\sigma,i,j}]_{i<j}$  is a vector of i.i.d. Bern(p) random variables.
- Recall:  $A_\epsilon^n(X)$ :  $\epsilon$ -typical set of length  $n$  sequences w.r.t  $P_X$ .
- **Result:**  $[G_{\sigma,i,j}]_{i<j}$  is  $\epsilon$ -typical w.r.t Bern(p) with probability one:

$$P([G_{\sigma,i,j}]_{i<j} \in A_\epsilon^{\frac{n(n-1)}{2}}(X)) \rightarrow 1, n \rightarrow \infty.$$

- All permutations of  $[G_{\sigma,i,j}]_{i<j}$  are also  $\epsilon$ -typical w.r.t Bern(p) with probability one.

# Typicality of Graph Pairs

 $g^1$ 

0	1	0	1	0	1
1	0	0	1	0	0
⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮
1	0	0	1	0	1

 $g^2$ 

0	0	0	1	1	1
0	0	0	1	1	0
⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮
1	0	0	0	0	1

- Recall matching edges generated i.i.d. according to  $P_{X_1, X_2}$ .
- Result:** The pair  $(G_{\sigma_1}^1, G_{\sigma_2}^2)$  is  $\epsilon$ -typical w.r.t  $P_{X_1, X_2}$  with probability one when  $\sigma_1 = \sigma_2$ .

# Typicality of Graph Pairs

 $g^1$ 

0	1	0	1	0	1
1	0	0	1	0	0
⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮
1	0	0	1	0	1

 $g^2$ 

0	0	0	1	1	1
0	0	0	1	1	0
⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮
1	0	0	0	0	1

- Recall matching edges generated i.i.d. according to  $P_{X_1, X_2}$ .
- **Result:** The pair  $(G_{\sigma_1}^1, G_{\sigma_2}^2)$  is  $\epsilon$ -typical w.r.t  $P_{X_1, X_2}$  with probability one when  $\sigma_1 = \sigma_2$ .
- **Idea for graph matching:** Find  $\sigma_2$  (labeling for  $g^2$ ) which results in a jointly typical pair  $(G_{\sigma_1}^1, G_{\sigma_2}^2)$ .

- *Wrong* labeling of nodes leads to a permutation of the adjacency matrix and permutation of the corresponding binary sequence.

- *Wrong* labeling of nodes leads to a permutation of the adjacency matrix and permutation of the corresponding binary sequence.
- To argue success of typicality-based graph matching, we need to investigate joint typicality of permutations of random vectors.

- *Wrong* labeling of nodes leads to a permutation of the adjacency matrix and permutation of the corresponding binary sequence.
- To argue success of typicality-based graph matching, we need to investigate joint typicality of permutations of random vectors.
- Bound as a function of permutation parameters: # of fixed points, and cycle decomposition.

- Example:  $\pi(Y^5) = (Y_2, Y_1, Y_4, Y_3, Y_5)$ .

- Example:  $\pi(Y^5) = (Y_2, Y_1, Y_4, Y_3, Y_5)$ .
- Permutation parameters:
  - 1 **Fixed Point:** Index  $i$  such that  $\pi(i) = i$ .
  - 2 **Cycle:** Orbit of each index  $(i, \pi(i), \pi^2(i), \dots)$ .
  - 3 **Cycle Length:**  $|\{\pi^k(i) | k \in \mathbb{N}\}|$ .



- Example:  $\pi(Y^5) = (Y_2, Y_1, Y_4, Y_3, Y_5)$ .
- Permutation parameters:
  - 1 **Fixed Point:** Index  $i$  such that  $\pi(i) = i$ .
  - 2 **Cycle:** Orbit of each index  $(i, \pi(i), \pi^2(i), \dots)$ .
  - 3 **Cycle Length:**  $|\{\pi^k(i) | k \in \mathbb{N}\}|$ .
- Notation:
  - # of fixed points:**  $m$ ,
  - # of non-trivial cycles:**  $c$ ,
  - Length of cycles:**  $(i_1, i_2, \dots, i_c), i_1 \geq i_2 \geq \dots \geq i_c$ .

- Example:  $\pi(Y^5) = (Y_2, Y_1, Y_4, Y_3, Y_5)$ .
- Permutation parameters:
  - 1 **Fixed Point:** Index  $i$  such that  $\pi(i) = i$ .
  - 2 **Cycle:** Orbit of each index  $(i, \pi(i), \pi^2(i), \dots)$ .
  - 3 **Cycle Length:**  $|\{\pi^k(i) | k \in \mathbb{N}\}|$ .
- Notation:
  - # of fixed points:**  $m$ ,
  - # of non-trivial cycles:**  $c$ ,
  - Length of cycles:**  $(i_1, i_2, \dots, i_c), i_1 \geq i_2 \geq \dots \geq i_c$ .
- Example:  $m = 1, c = 2, i_1 = 2, i_2 = 2$ .  
Notation:  $\pi = (12)(34)(5)$ .

- **Result:** Suppose  $(X^n, Y^n)$  i.i.d.  $\sim P_{X,Y}$ . Then for any permutation  $\pi$ ,

$$P((X^n, \pi(Y^n)) \in A_\epsilon^n(X, Y))$$

is a function of  $m, c$  and  $(i_1, i_2, \dots, i_c)$ .

## Theorem

Let  $(X^n, Y^n)$  be a pair of i.i.d sequences. For any permutation  $\pi$  with  $m$  fixed points, the following holds:

$$\begin{aligned} P((X^n, \pi(Y^n)) \in A_\epsilon^n(X, Y)) \\ \leq 2^{-\frac{n}{4}(D(P_{X,Y} \| ((1-\alpha)P_X P_Y + \alpha P_{X,Y}) - |\mathcal{X}||\mathcal{Y}|\epsilon + O(\frac{\log n}{n}))}, \end{aligned}$$

where  $\alpha = \frac{m}{n}$ .

- Proof idea:
  - Convexity of KL divergence
  - Grouping elements of  $(X^n, \pi(Y^n))$  into independent subsets.

## Theorem

Typicality based graph matching is successful if

$$\forall \alpha \in [0, 1 - \delta] : 8(1 - \alpha) \frac{\log n}{n} \leq D(P_{X,Y}^{(n)} \| (1 - \alpha)P_X^{(n)}P_Y^{(n)} + \alpha P_{X,Y}^{(n)})$$

as  $n \rightarrow \infty$ .

Furthermore, a necessary condition for existence of successful graph matching is

$$\frac{2 \log n}{n} \leq I(X_1; X_2).$$

- Shirani, Garg, Erkip, JSAC 2021.

- Database and graph matching.
  - Privacy implications.
  - Rich literature in system level approach and algorithms for matching given graphs/databases.

- Database and graph matching.
  - Privacy implications.
  - Rich literature in system level approach and algorithms for matching given graphs/databases.
- Stochastic approach allows:
  - Use of tools from probability and information theory.
  - New algorithms.
  - Necessary and sufficient conditions for success.

- Database matching.
  - Analogy with channel decoding.



- Database matching.
  - Analogy with channel decoding.
  - Structure is useful.
  - Repetitions can be inferred when
    - When there is no noise.
    - When there are enough (but small) number of seeds.

- Database matching.
  - Analogy with channel decoding.
  - Structure is useful.
  - Repetitions can be inferred when
    - When there is no noise.
    - When there are enough (but small) number of seeds.
- Graph matching.
  - Exploits richer structure.

- Database matching.
  - Analogy with channel decoding.
  - Structure is useful.
  - Repetitions can be inferred when
    - When there is no noise.
    - When there are enough (but small) number of seeds.
- Graph matching.
  - Exploits richer structure.
  - Seeds are still very important.

- **Database matching:**
  - Obtaining entries (column) for each member (row) may be costly.

- **Database matching:**

- Obtaining entries (column) for each member (row) may be costly.
- How can matching be done by incurring only a small cost?
- Relates to *rateless codes* and *feedback*.
- Also relates to *fingerprinting*.

- **Database matching:**

- Obtaining entries (column) for each member (row) may be costly.
- How can matching be done by incurring only a small cost?
- Relates to *rateless codes* and *feedback*.
- Also relates to *fingerprinting*.
- Efficient algorithms for noisy databases with deletions/replicas.

- **Database matching:**

- Obtaining entries (column) for each member (row) may be costly.
- How can matching be done by incurring only a small cost?
- Relates to *rateless codes* and *feedback*.
- Also relates to *fingerprinting*.
- Efficient algorithms for noisy databases with deletions/replicas.
- Correlated rows/columns.

- **Database matching:**

- Obtaining entries (column) for each member (row) may be costly.
- How can matching be done by incurring only a small cost?
- Relates to *rateless codes* and *feedback*.
- Also relates to *fingerprinting*.
- Efficient algorithms for noisy databases with deletions/replicas.
- Correlated rows/columns.

- **Graph matching:**

- Matching more than two graphs.
- Matching partially labeled graphs.
- Graphs with partial vertex overlap.
- Community structures and preferential attachment.



- [1] E. Onaran, S. Garg and E. Erkip, "Optimal De-anonymization in Random Graphs with Community Structure," *Asilomar 2016*.
- [2] F. Shirani, S. Garg and E. Erkip, "An Information Theoretic Framework for Active De-anonymization in Social Networks Based on Group Memberships," *Allerton 2017*.
- [3] F. Shirani, S. Garg and E. Erkip, "Seeded Graph Matching: Efficient Algorithms and Theoretical Guarantees," *Asilomar 2017*.
- [4] F. Shirani, S. Garg and E. Erkip, "Typicality Matching for Pairs of Correlated Graphs," *ISIT 2018*.
- [5] F. Shirani, S. Garg and E. Erkip, "Optimal Active Social Network De-anonymization using Information Thresholds," *ISIT 2018*.
- [6] F. Shirani, S. Garg and E. Erkip, "Matching Graphs with Community Structure: A Concentration of Measure Approach," *Allerton 2018*.
- [7] F. Shirani, S. Garg and E. Erkip, "A Concentration of Measure Approach to Database De-anonymization," *ISIT 2019*.
- [8] F. Shirani, S. Garg and E. Erkip, "A Concentration of Measure Approach to Correlated Graph Matching," *JSAIT 2021*.
- [9] M. Shariatnasab, F. Shirani, S. Garg and E. Erkip, "On Graph Matching Using Generalized Seed Side-Information," *ISIT 2021*.
- [10] S. Bakirtas and E. Erkip, "Database Matching Under Column Deletions," *ISIT 2021*.
- [11] M. Shariatnasab, F. Shirani, and E. Erkip, "Fundamental Privacy Limits in Bipartite Networks under Active Attacks," *JSAC 2022*.
- [12] S. Bakirtas and E. Erkip, "Seeded Database Matching Under Noisy Column Repetitions," to appear in *ITW 2022*.
- [13] S. Bakirtas and E. Erkip, "Matching of Markov Databases Under Random Column Repetitions," to appear in *Asilomar 2022*.