Selected Research Topics

Mikaël Monet

Simons Institute's *Meet the Fellows* day

Berkeley, Friday September 8th, 2023

Academic career

2012–2015: Engineering school

2014–2015: Parisian master of research in computer science

2015–2018: PhD at Télécom Paris with **Pierre Senellart** and **Antoine Amarilli**, on "Combined Complexity of Probabilistic Query Evaluation"

2019–2020: Postdoc at Millennium Institute for Foundational Research on Data (Santiago, Chili) with **Pablo Barceló**

Octobre 2020–: Research position at Inria Lille

Uncertain data, provenance and knowledge compilation

Uncertain data

- Real-world data can be uncertain
    - missing values
    - inconsistent data sources
    - information extraction from the Web
    - machine learning techniques (NLP, etc.)
    - imprecise sensors in experimental sciences
    - ...

→ We need methods to manage this uncertainty

- Main models for relational data: probabilistic or incomplete databases

What’s the (computational) difficulty?

- Example: **probabilistic databases**

What's the (computational) difficulty?

- Example: **probabilistic databases**
  - → Simplest formalism: tuple-independent probabilistic databases

$D =$

Applies		π
Alice	Inria	0.9
Alice	CNRS	0.5
Bob	Inria	0.2
John	Inria	0.7

What's the (computational) difficulty?

- Example: **probabilistic databases**
  - → Simplest formalism: tuple-independent probabilistic databases

$D' =$

Applies		π
Alice	Inria	0.9
Alice	CNRS	0.5
Bob	Inria	0.2
John	Inria	0.7

What's the (computational) difficulty?

- Example: **probabilistic databases**
  - → Simplest formalism: tuple-independent probabilistic databases

$D' =$

Applies		π
Alice	Inria	0.9
Alice	CNRS	0.5
Bob	Inria	0.2
John	Inria	0.7

$\Pr(D') = (1 - 0.9) \times 0.5 \times (1 - 0.2) \times 0.7$

What's the (computational) difficulty?

- Example: **probabilistic databases**
  - → Simplest formalism: tuple-independent probabilistic databases

D =

Applies		π
Alice	Inria	0.9
Alice	CNRS	0.5
Bob	Inria	0.2
John	Inria	0.7

q = "there are two people applying to the same institution"

What's the (computational) difficulty?

- Example: **probabilistic databases**
  - → Simplest formalism: tuple-independent probabilistic databases

D =

Applies		π
Alice	Inria	0.9
Alice	CNRS	0.5
Bob	Inria	0.2
John	Inria	0.7

q = "there are two people applying to the same institution"

$$\Pr((D,\pi) \models q) = \sum_{\substack{D' \subseteq D \\ D' \models q}} \Pr(D')$$

What's the (computational) difficulty?

- Example: **probabilistic databases**
  - → Simplest formalism: tuple-independent probabilistic databases

$D =$

Applies		π
Alice	Inria	0.9
Alice	CNRS	0.5
Bob	Inria	0.2
John	Inria	0.7

q = "there are two people applying to the same institution"

$\Pr((D,\pi) \models q) = \sum_{\substack{D' \subseteq D \\ D' \models q}} \Pr(D')$ **exhaustive computation is too costly!**

Provenance

- Provenance of a query on a database D: Boolean function whose variables are the tuples of D that intuitively represent "Which combinations of tuples make the query become true?"

Provenance

- Provenance of a query on a database D: Boolean function whose variables are the tuples of D that intuitively represent "Which combinations of tuples make the query become true?"

$D =$

Applies		π
Alice	Inria	0.9
Alice	CNRS	0.5
Bob	Inria	0.2
John	Inria	0.7

$q =$ "there are two people applying to the same institution"

Provenance

- Provenance of a query on a database D: Boolean function whose variables are the tuples of D that intuitively represent "Which combinations of tuples make the query become true?"

$D =$

Applies		π
Alice	Inria	0.9
Alice	CNRS	0.5
Bob	Inria	0.2
John	Inria	0.7

$$\mathrm{Prov}(q, D) = [C(A, I) \land C(B, I)] \lor [C(A, I) \land C(J, I)] \lor [C(B, I) \land C(J, I)]$$

q = "there are two people applying to the same institution"

Provenance

- Provenance of a query on a database D: Boolean function whose variables are the tuples of D that intuitively represent "Which combinations of tuples make the query become true?"

$D =$

Applies		π
Alice	Inria	0.9
Alice	CNRS	0.5
Bob	Inria	0.2
John	Inria	0.7

$$\mathrm{Prov}(q, D) = [C(A,I) \wedge C(B,I)] \vee [C(A,I) \wedge C(J,I)] \vee [C(B,I) \wedge C(J,I)]$$

explain, keep trace of the computation

q = "there are two people applying to the same institution"

Provenance

- Provenance of a query on a database D: Boolean function whose variables are the tuples of D that intuitively represent "Which combinations of tuples make the query become true?"

$D =$

Applies		π
Alice	Inria	0.9
Alice	CNRS	0.5
Bob	Inria	0.2
John	Inria	0.7

$$\mathrm{Prov}(q, D) = [C(A, I) \wedge C(B, I)] \vee [C(A, I) \wedge C(J, I)] \vee [C(B, I) \wedge C(J, I)]$$

explain, keep trace of the computation

q = "there are two people applying to the same institution"

we can use it for probabilistic computation!

Provenance and knowledge compilation

- Use of provenance in probabilistic databases: compute the provenance φ of a query on a probabilistic database, then compute the probability that φ evaluates to true. Problem: This is generally intractable! ($\#\mathrm{P}$-hard)

Provenance and knowledge compilation

- Use of provenance in probabilistic databases: compute the provenance φ of a query on a probabilistic database, then compute the probability that φ evaluates to true. Problem: This is generally intractable! ($\#\mathrm{P}$-hard)
- Need a tractable representation

Provenance and knowledge compilation

- Use of provenance in probabilistic databases: compute the provenance φ of a query on a probabilistic database, then compute the probability that φ evaluates to true. Problem: This is generally intractable! ($\#\mathrm{P}$-hard)

- Need a tractable representation

→ Knowledge compilation: studies Boolean function representations with "good properties"
  - → propositional formulas (DNF, CNF)
  - → Binary Decision Diagrams (OBDDs, FBDDs)
  - → restricted classes of Boolean circuits (NNF, d-DNNF, dec-DNNF, SDDs, d-D, d-SDNNFs etc.)

Relevance score of tuples for query answering

Provenance can also be used to compute so-called Shapley values

Definition: problem $\mathsf{Shapley}(q)$

Input: A database D and a tuple $f \in D$

Output: The value $\mathsf{Shapley}(q, D, f)$

Intuitively: $\mathsf{Shapley}(q, D, f)$ is the "importance" of f in D for the query q

Relevance score of tuples for query answering

Provenance can also be used to compute so-called Shapley values

Definition: problem Shapley(q)

Input: A database D and a tuple $f \in D$

Output: The value Shapley(q, D, f)

Intuitively: Shapley(q, D, f) is the "importance" of f in D for the query q

Proposition [With Daniel Deutch, Nave Frost and Benny Kimelfeld]

Given as input a deterministic and decomposable circuit C representing the provenance, we can compute in time $O(|C| \cdot |D|^2)$ the value SHAP(q, D, f).

Relevance score of tuples for query answering

Provenance can also be used to compute so-called Shapley values

Definition: problem Shapley(q)

Input: A database D and a tuple $f \in D$

Output: The value Shapley(q, D, f)

Intuitively: Shapley(q, D, f) is the "importance" of f in D for the query q

Proposition [With Daniel Deutch, Nave Frost and Benny Kimelfeld]

Given as input a deterministic and decomposable circuit C representing the provenance, we can compute in time $O(|C| \cdot |D|^2)$ the value SHAP(q, D, f).

Similar results for the **SHAP-score** from ML (With Marcelo Arenas, Pablo Barceló and Leopoldo Bertossi).

A hardness result on counting weighted matchings for unbounded-treewidth graph families

Counting weighted matchings and treewidth

Let $\mathcal{G}$ be a family of (undirected) graphs.

Definition: problem ProbMatch($\mathcal{G}$)

Input: A graph $G \in \mathcal{G}$ and probability values p_e for every edge e of G

Output: The probability of obtaining a matching of G when we pick every edge e of G independently with probability p_e

Counting weighted matchings and treewidth

Let $\mathcal{G}$ be a family of (undirected) graphs.

Definition: problem ProbMatch($\mathcal{G}$)

Input: A graph $G \in \mathcal{G}$ and probability values p_e for every edge e of G

Output: The probability of obtaining a matching of G when we pick every edge e of G independently with probability p_e

If $\mathcal{G}$ has bounded treewidth, then ProbMatch($\mathcal{G}$) is in PTIME.

Counting weighted matchings and treewidth

Let $\mathcal{G}$ be a family of (undirected) graphs.

Definition: problem ProbMatch($\mathcal{G}$)

Input: A graph $G \in \mathcal{G}$ and probability values p_e for every edge e of G

Output: The probability of obtaining a matching of G when we pick every edge e of G independently with probability p_e

If $\mathcal{G}$ has bounded treewidth, then ProbMatch($\mathcal{G}$) is in PTIME.

Theorem [With Antoine Amarilli]

Let $\mathcal{G}$ be an arbitrary family of graphs having unbounded treewidth which is *treewidth constructible*. Then ProbMatch($\mathcal{G}$) is intractable.

Enumerating regular languages with bounded delay

Enumerating regular languages with bounded delay

Fix an alphabet Σ, and consider the edit distance $\delta : \Sigma^* \times \Sigma^* \to \mathbb{N}$.

Enumerating regular languages with bounded delay

Fix an alphabet Σ, and consider the edit distance $\delta : \Sigma^* \times \Sigma^* \to \mathbb{N}$.

Definition: constant-distance enumerable

Call a language $L \subseteq \Sigma^*$ constant-distance enumerable if there exists $d \in \mathbb{N}$ and an ordering $w_1, w_2, \ldots$ of the words of L such that $\delta(w_i, w_{i+1}) \leq d$ for all i.

Fix an alphabet Σ, and consider the edit distance $\delta : \Sigma^* \times \Sigma^* \to \mathbb{N}$.

Definition: constant-distance enumerable

Call a language $L \subseteq \Sigma^*$ constant-distance enumerable if there exists $d \in \mathbb{N}$ and an ordering $w_1, w_2, \ldots$ of the words of L such that $\delta(w_i, w_{i+1}) \leq d$ for all i.

Examples: $L_1 = a*$, $L_2 = (a|b)^*$ YES. $L_3 = a^*|b^*$ NO.

Enumerating regular languages with bounded delay

Fix an alphabet Σ, and consider the edit distance $\delta : \Sigma^* \times \Sigma^* \to \mathbb{N}$.

Definition: constant-distance enumerable

Call a language $L \subseteq \Sigma^*$ constant-distance enumerable if there exists $d \in \mathbb{N}$ and an ordering $w_1, w_2, \ldots$ of the words of L such that $\delta(w_i, w_{i+1}) \leq d$ for all i.

Examples: $L_1 = a*$, $L_2 = (a|b)^*$ YES. $L_3 = a^*|b^*$ NO.

Result [With Antoine Amarilli]

We characterize exactly what are the regular languages that are enumerable. When it is the case we provide an algorithm that enumerates the words with a constant delay (the delay depends on the language but not on the length of the current word).

An open problem about perfect matchings in the Boolean lattice

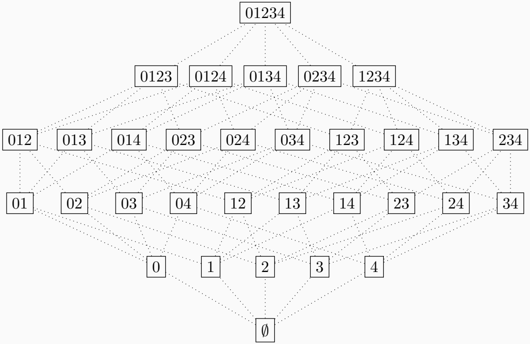
An open problem (1/3)

- A matching of an undirected graph $G = (V, E)$ is a subset $M \subseteq E$ of edges such that $e \cap e'$ for all $e, e' \in M$.
- A matching M is perfect if it touches all vertices of G

An open problem (1/3)

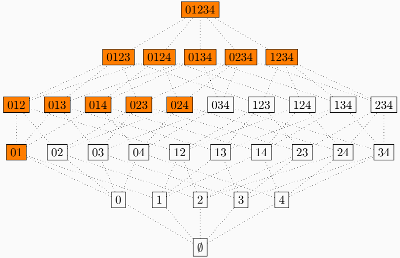
- A matching of an undirected graph $G = (V, E)$ is a subset $M \subseteq E$ of edges such that $e \cap e'$ for all $e, e' \in M$.
- A matching M is perfect if it touches all vertices of G

Let's consider the Boolean lattice over k elements. Example for $k = 5$:



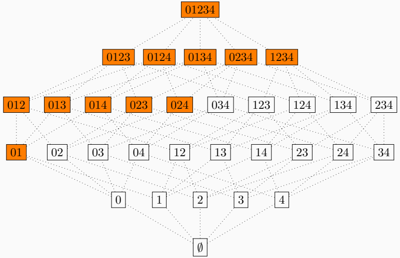
An open problem (2/3)

Let O be a set of nodes that is upward-closed and such that O has as many nodes of even size as nodes of odd size. Example: $O =$ the orange nodes



An open problem (2/3)

Let O be a set of nodes that is upward-closed and such that O has as many nodes of even size as nodes of odd size. Example: $O =$ the orange nodes

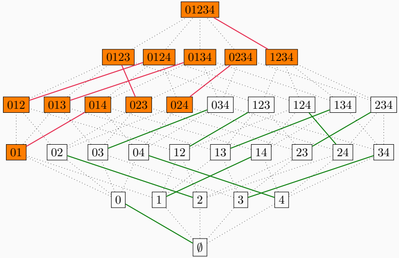


Is this true?

For any k and O satisfying this property, then: either the graph induced by O has a perfect matching, or the complement graph has a perfect matching.

An open problem (2/3)

Let O be a set of nodes that is upward-closed and such that O has as many nodes of even size as nodes of odd size. Example: $O =$ the orange nodes

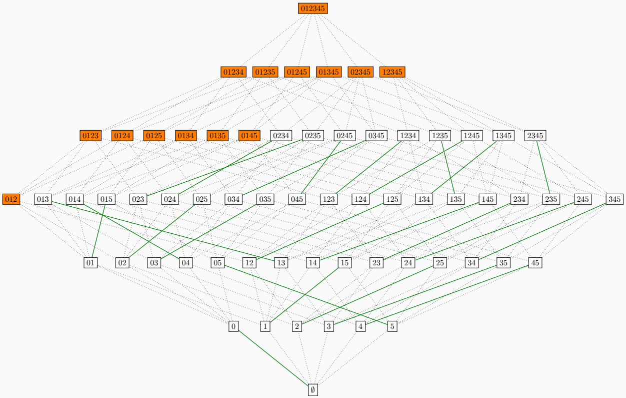


Is this true?

For any k and O satisfying this property, then: either the graph induced by O has a perfect matching, or the complement graph has a perfect matching.

An open problem (3/3)

In some cases, one the top or the bottom graph (but not both) has a perfect matching. Example:



Computer search for counterexample: none so far.

Thanks for your attention!