The Other Side of Big Graphs
Dependency Theory, Practice and Applications

Wenfei Fan

University of Edinburgh

Backgrounds

The veracity of big graphs
- Inconsistencies, entity resolution
- Knowledge extraction, fraud detection
- Missing values, association rules

These are about the *semantics* of big graphs!

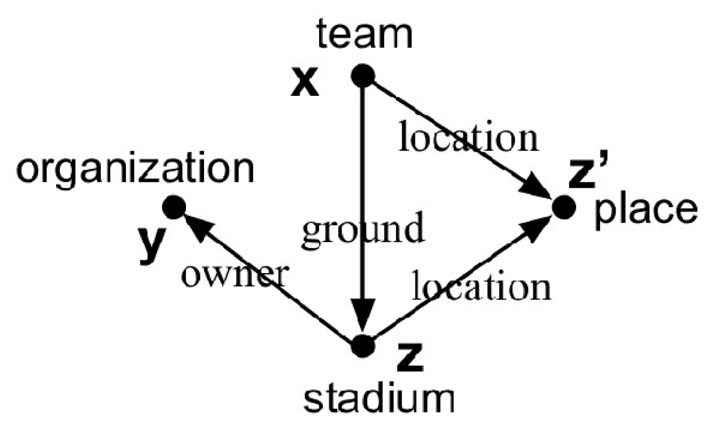
Rule bases approach are practical: 67% tools are rule based.
Challenges for graph rules:
- Capturing topological structures
- Coping with schemaless graphs
- Correctness and non-redundancy
- Catching, detecting and fixing errors
- Making predictions for missing links

The need for a dependency theory for graphs

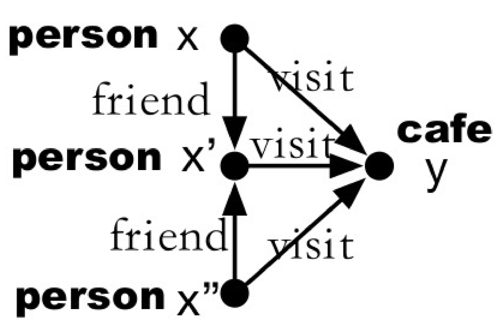
Dependencies for graph

Dependencies $Q[\bar{x}](\mathrm{X} \to \mathrm{Y})$
- Topological + attribute constraints
- Adding ML predictors to literal
- Matching complexity bounds to relational counterparts
- Axiom system: sound, complete and independent

Examples



Q₁



Q₂

Q_1: If a team x uses a stadium z as its ground at the same location z', z is owned by a sport organization y, and ML predicts the association between x and y, then x is a tenant of z

$$Q_1(\mathcal{M}(x, y, \text{association}) \to \text{tenant}(z, x))$$

Q_2: If x' is a friend of both x and x'', and all the three visit the same cafe y and share comment interest, then x and x'' are likely to be friends

$$Q_2(x.\text{interest} = x'.\text{interest} \wedge x'.\text{interest} = x''.\text{interest} \to \mathcal{M}(x, x'', \text{friend}))$$

Error detection and fixing

Error detection and fixing:
- **Input**: a graph G, a set of rules Σ and ground truth Γ
- Error detection: find the set $\mathrm{Vio}(\Sigma, G)$ of violations
  - Incremental error detection: finding $\Delta\mathrm{Vio}(\Sigma, G, \Delta G)$ when given $\mathrm{Vio}(\Sigma, G)$ and batch updates ΔG
- Fixing: find an instance conforming to Σ
  - Fix: combining object identification and data repairing
  - Certain fix: every fix is correct, no errors are newly introduced

Fundamental studies
- Validation and incremental validation: **coNP**-complete
- Consistency with ground truth: **coNP**-complete
- Cleaning: **NP**-complete, **PTIME** data complexity

Algorithms
- Error detection: parallel algorithm
  - Sequential: match and detect
  - Work unit: neighbors around pivot nodes
  - Balance workload and minimizing communication cost
- Incremental detection
  - Divide $\Delta\mathrm{Vio}$ as $\Delta^{+}\mathrm{Vio}$ and $\Delta^{-}\mathrm{Vio}$
  - Update $\Delta^{+}\mathrm{Vio}$ and $\Delta^{-}\mathrm{Vio}$ from edge insertion and deletion respectively
- Certain fix: two modes in practice
  - Online: fix errors pertaining to a small set of user's interest
  - Offline: deduce fixes to the entire graph

Localizable algorithms: affecting small areas surrounding ΔG
Parallel scalable algorithms for all problems: $T(p) = O(\frac{t}{p})$

Reasoning about dependencies and their discovery

Reasoning about a set of rules Σ:
- **Input**: a graph G, a set of rules Σ, one rule φ and a threshold ρ
- Satisfiability: determine if there exists a graph G such that $G \models \Sigma$
- Implication: determine if for any graph G and $G \models \Sigma$, then $G \models \varphi$

Discovering sensible rules from graph
- **Input**: support threshold ρ, graph G
- **Output**: find a cover of non-redundant rules Σ that are ρ-frequent in G

Fundamental studies
- **coNP**-complete and **NP**-complete for satisfiability and implication resp.
- **FPT** for both problems when parameterizing the vertex number in Q

Algorithms
- Reasoning: sequential algorithms from characterizations
- Discovering: vertical and horizontal spawning from generation tree

Parallel scalable algorithms for all problems: $T(p) = O(\frac{t}{p})$
The more processors used, better the performance is

A Uniform Framework

Uniform framework of logic-based and ML-based methods
- Capturing absent links, missing values, and semantic inconsistencies
- All ML methods can be plugged in literals
- Interpret ML predictions using rules

Deductions: deducing $\mathrm{ded}(\Sigma, G)$
- Deductions: extensions of G from literals in Σ, adding missing attributes and edges
- Algorithms by extending chase sequence
- Incremental deduction: update $\mathrm{ded}(\Sigma, G, \Delta G)$ when given graph update ΔG

Algorithms
- Sequential: extending chase sequences
- Parallel: GRAPE based solution
- Incremental: catching invalidness and refining

A first step of unifying rule-based and ML-based methods

Delivered

10 top-tier publications:
- 2021: VLDB [1]
- past: 2020 [2, 3, 4], 2019 [5, 6, 7], and 2018 [8, 9, 10]

Put the package of solutions in action!

Publications

[1] Wenfei Fan et al. "Parallel Discrepancy Detection and Incremental Detection.". In: *PVLDB* 14.8 (2021), pp. 1351–1364.

[2] Wenfei Fan et al. "Discovering Graph Functional Dependencies". In: *TODS* (2020).

[3] Wenfei Fan et al. "Catching Numeric Inconsistencies in Graphs". In: *TODS* (2020).

[4] Wenfei Fan et al. "Capturing associations in graphs". In: *Proceedings of the VLDB Endowment* 13.12 (2020), pp. 1863–1876.

[5] Wenfei Fan and Ping Lu. "Dependencies for graphs". In: *ACM Transactions on Database Systems (TODS)* 44.2 (2019), pp. 1–40.

[6] Wenfei Fan. "Dependencies for Graphs: Challenges and Opportunities". In: *JDIQ* 11.2 (2019), pp. 1–12.

[7] Wenfei Fan et al. "Deducing certain fixes to graphs". In: *PVLDB* 12.7 (2019), pp. 752–765.

[8] Wenfei Fan, Xueli Liu, and Yingjie Cao. "Parallel reasoning of graph functional dependencies". In: *ICDE*. IEEE. 2018, pp. 593–604.

[9] Wenfei Fan et al. "Discovering graph functional dependencies". In: *SIGMOD*. 2018, pp. 427–439.

[10] Wenfei Fan et al. "Catching numeric inconsistencies in graphs". In: *SIGMOD*. 2018, pp. 381–393.