Answering Why-Questions for Subgraph Queries in Multi-Attributed Graphs

Qi Song[1] Mohammad Hossein Namaki[1] Yinghui Wu[1,2]

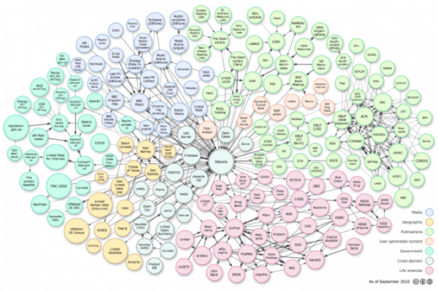
1 Washington State University

2 Pacific Northwest National Laboratory

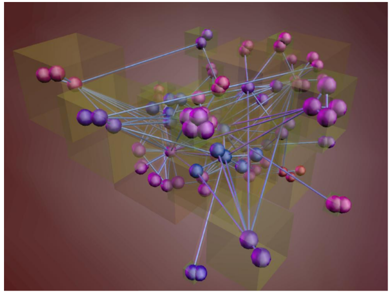
Introduction

- Multi-attributed graphs and entity search based on subgraph queries
  - Subgraph query Q: a (labeled) graph pattern with an output node u_o;
  - Answer of Q refers to entities that are matches of u_o $(Q(u_o, G))$.

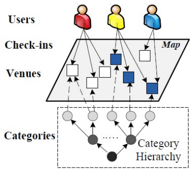
Social Network:
POI Recommendation

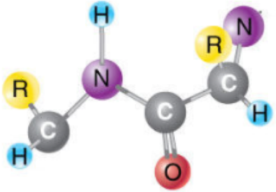


Knowledge Graph:
Knowledge Extraction



Protein Network:
Medical Analysis





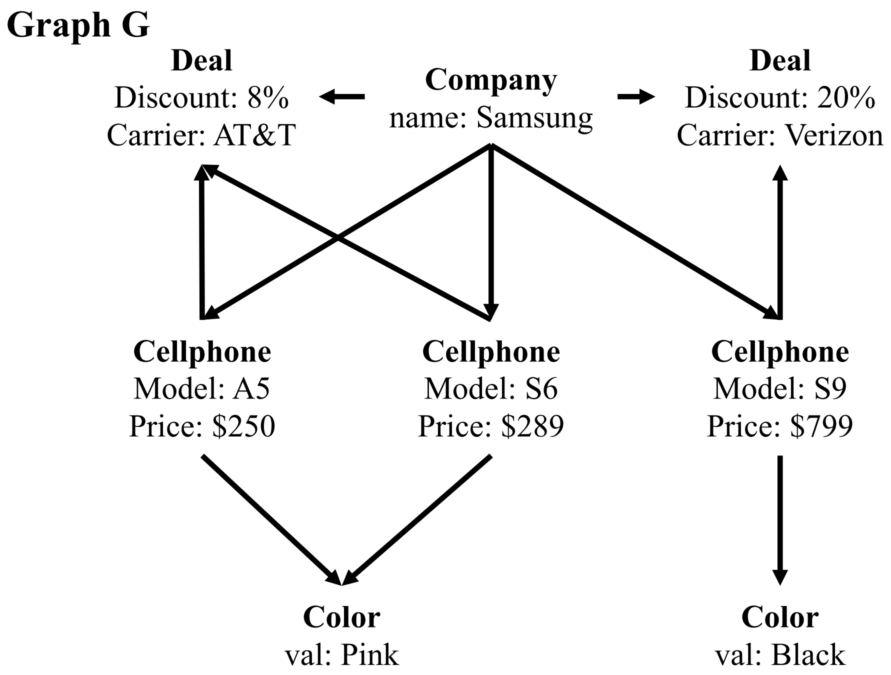
Why-questions

- Writing queries is nevertheless a nontrivial task for end users.
  - The graph is large and heterogeneous;
  - Users often need to revise the queries multiple times to find desirable answers.
  - An explain functionality supported by query rewriting is thus desirable to help users understand the unexpected answers.

- Why-questions.
  - **Why question**: "why some (unexpected) entities are in the query answer?"; and
  - **Why-not question**: "why certain entities are missing from the query result?"

- Answering Why-questions helps users to tune their queries towards desirable answers.

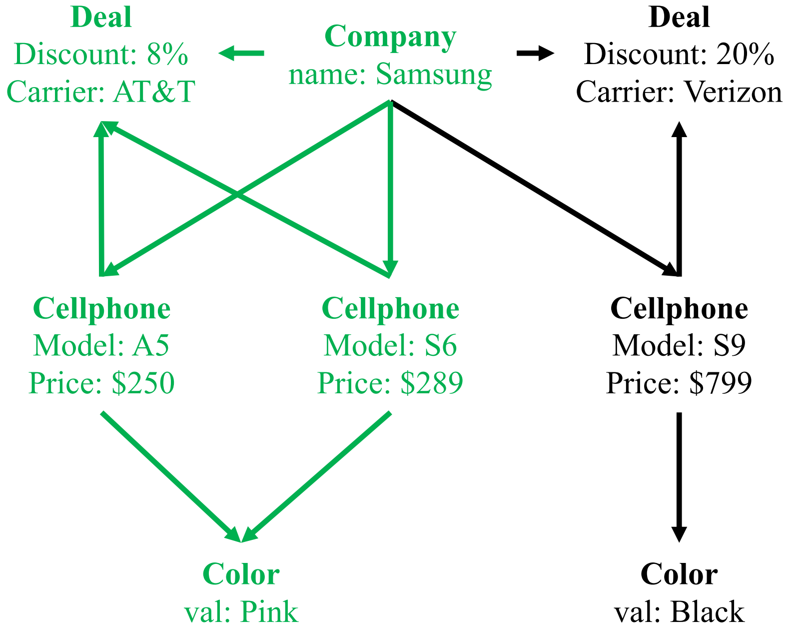
- Example: a knowledge graph G about products of an online store



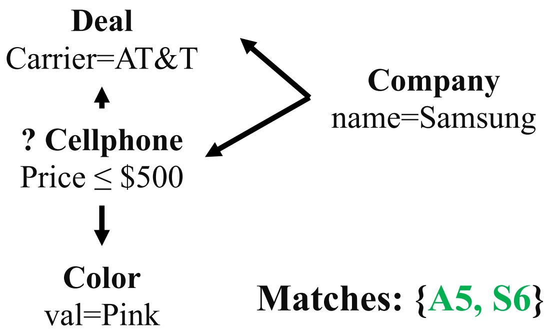
Example

A user wants to search for Samsung cellphones packed with color pink and carrier AT&T , with price less than $500.

Graph G



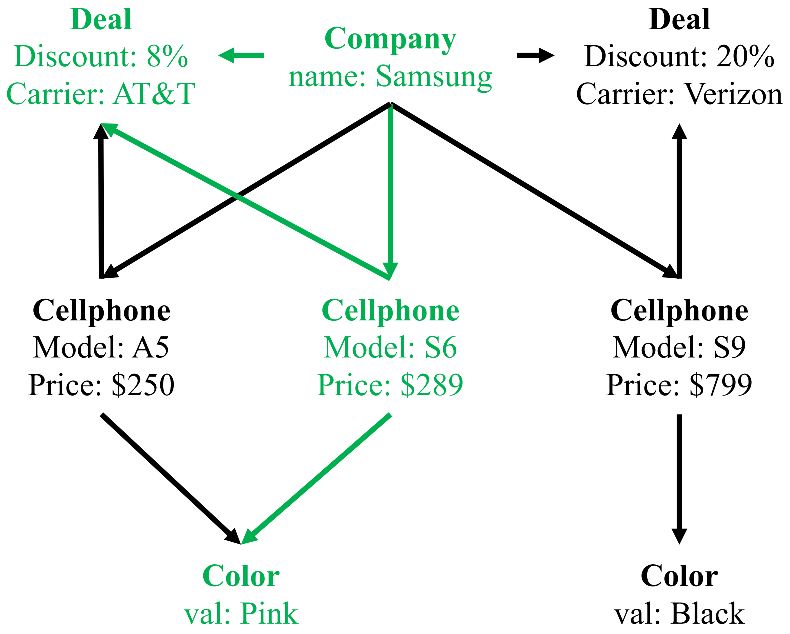
Query Q



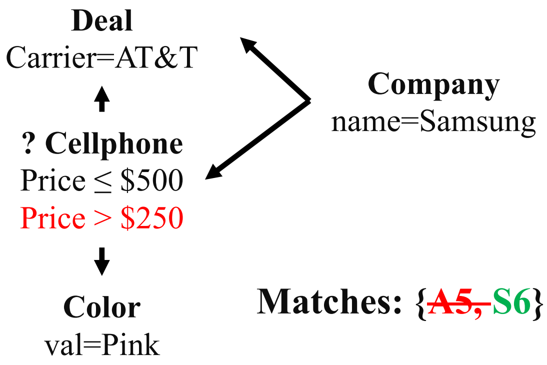
Matches: {A5, S6}

*"why model **A5** is in the query result of Q ?"*

Graph G



Query rewrite Q_1



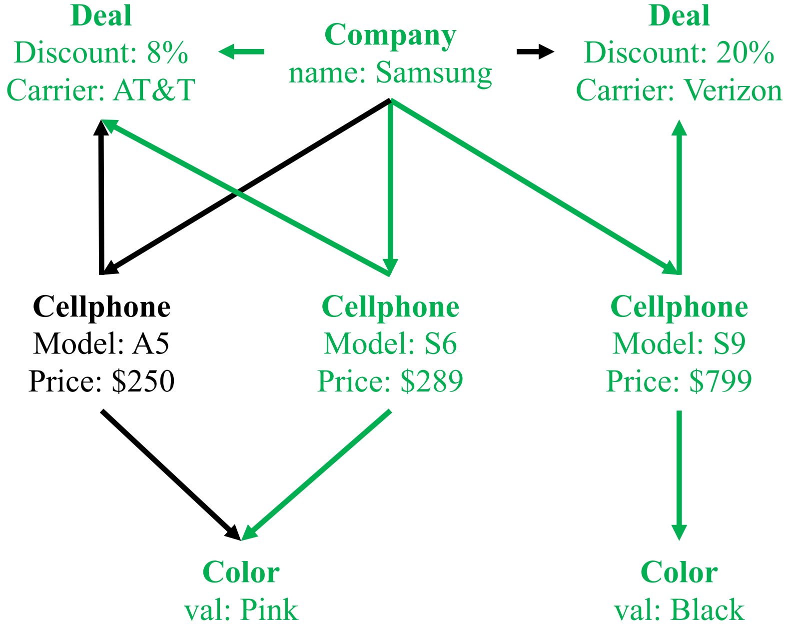
Matches: {~~A5,~~ S6}

The user may not be interested in cheaper versions as a new lower bar of price $250 is added.

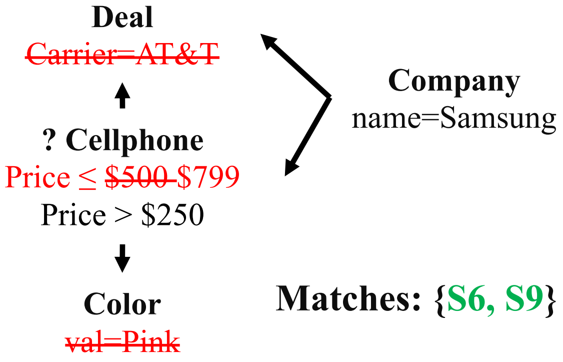
Example

*“why model **S9** are not in the query result ?”*

Graph G



Query rewrite Q_2

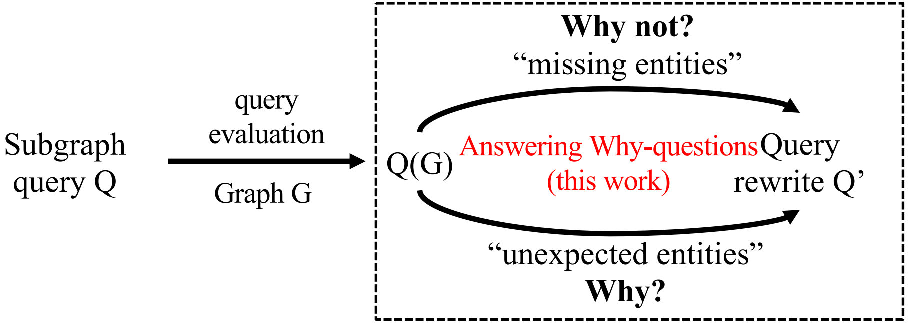


Matches: {S6, S9}

S9 is more expensive than expected (price relaxed to $799), there is no Pink *S9* model, and no *S9* model is supported by AT&T.

Contributions

- The need of answering Why-questions is evident in exploratory graph search.



- Contributions
  - We formalize Why-questions for subgraph queries in terms of graph query rewrites.
  - We formalize the problem of answering Why-questions.
  - We develop both exact and approximation algorithms.
  - We experimentally verify the effectiveness and efficiency of our algorithms.

Problem Formulation

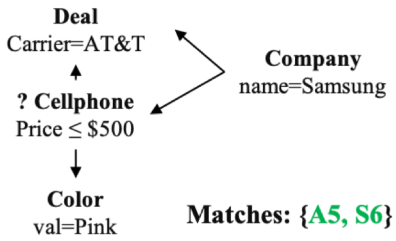
- Categorization of Why-Questions.
  - Why-not: why the nodes in V_{Cu} , with attribute values that satisfy the value constraints in C (if not empty), are not matches of u of Q?

$$\left(u_o, V_{C_u}, C\right)$$

Output node of Q. | A set of ”missing matches” of u_o. | Selection condition

?Cellphone | **{S9}** | {Cellphone.OS $\geq$ 5.0}

Query Q



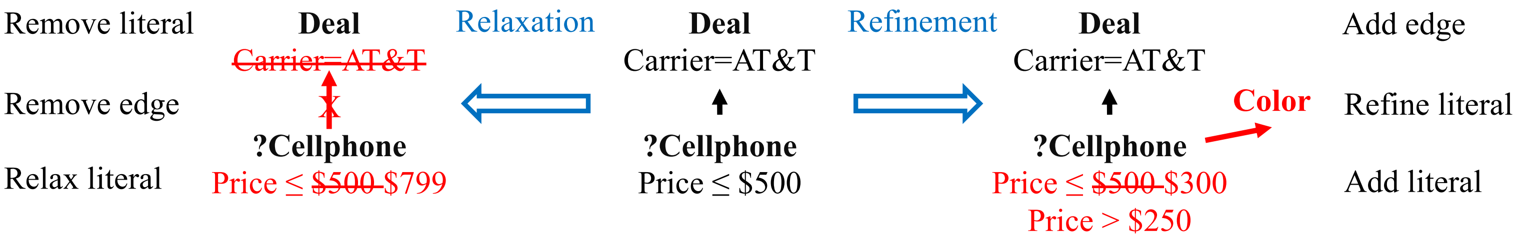
  - Why: why the nodes in V_{Nu} are included as matches for u_o in G?

$$\left(u_o, V_{N_u}\right)$$

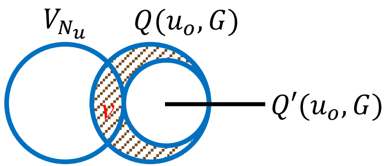
Output node of Q. | A set of ”unexpected matches” of u_o.

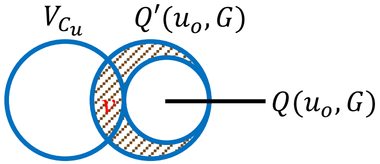
?Cellphone | **{A5}**

- Answers for Why-Questions
  - Query rewrites: six classes of primitive query editing operators



  - Answering Why-Questions: a query rewrite $Q' = Q \oplus O$ is an answer of a
    - Why: Q' excludes at least one unexpected match $v \in V_{N_u}$.
    - Why-not: $Q'(u_o, G)$ contains at least one "missing" match in V_{C_u} that satisfies C.



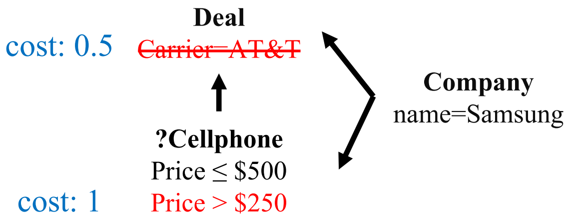


Problem Formulation

- Query editing cost: operators that modify more "important" fraction (closer to u_o) should be more expensive.

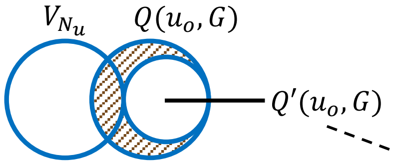
Based on output centrality of node *u'*

$$\mathrm{oc}(u', u_o) = \frac{d_Q}{d(u', u_o) + 1}$$



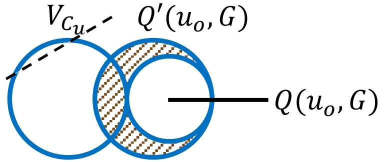
- Answer closeness: between $Q(u_o, G)$ and $Q'(u_o, G)$
  - Why: the fraction of V_{N_u} that are excluded from $Q'(u_o, G)$.

$$\mathrm{cl}(O, V_{N_u}) = \frac{|(Q(u_o, G) \setminus Q'(u_o, G)) \cap V_{N_u}|}{|V_{N_u}|}$$



  - Why-not: the fraction of new matches in V_{C_u} that are introduced in $Q'(u_o, G)$.

$$\mathrm{cl}(O, V_{C_u}) = \frac{|(Q'(u_o, G)) \cap V_{C_u}|}{|V_{C_u}|}$$



Guard condition: avoid over-refinement or over-relaxation

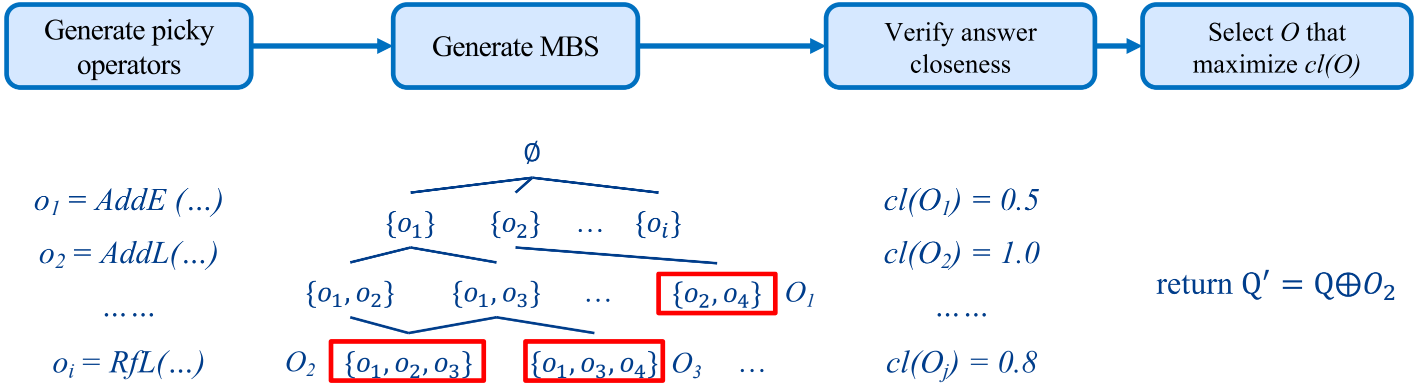
Problem Formulation

- Problem statement
  - Given a query Q, answer $\mathrm{Q}(u_o, G)$, graph G, a Why-question W, editing budget B,
  - Compute a query rewrite $\mathrm{Q}' = \mathrm{Q} \oplus O^*$, such that

$$O^* = \operatorname*{argm\mathit{ax}}_{O:c(O)\leq B} cl(O, V_u)$$

  - The problem of answering Why and Why-not questions are both NP-hard.

- Computing optimal query rewrites
  - Maximum bounded set (MBS): with $c(O) \leq B$ and all of its superset has cost exceeds B.
  - An exact algorithm (ExactWhy):



- Time cost: $O(|Q|\left|N_{d_Q+1}(Q(u_o, G))\right| + |O_s|^{3B}|N_{d_Q+1}(V_{N_u})|^{|Q|})$.

- Approximating optimal query rewrites
  - Given refinement operator set O, the *marginal gain* of an operator o to O: $mg(O, o) = cl(O \oplus \{o\}) - cl(O)$;
  - Function $cl(\cdot)$ is submodular over picky set O_s;
  - An approximation algorithm ApproxWhy:

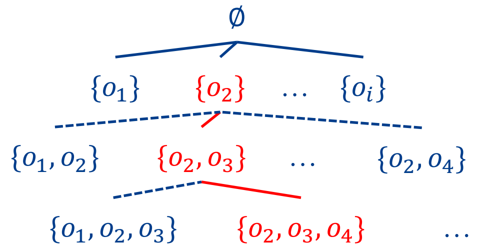
Generate picky operators → Estimate $\widehat{mg(o)}$ ⇄ Iteratively select o with maximize $\frac{\widehat{mg(o)}}{c(o)}$ → Output O that maximize $cl(O)$

$o_1 = AddE\ (\ldots)$

$o_2 = AddL(\ldots)$

……

$o_i = RfL(\ldots)$



return $Q' = Q \oplus O$

- Approximation ratio: $\frac{1}{2} \cdot \left(1 - \frac{1}{e}\right) \cdot cl(O^*, V_{N_u}) - 6B\varepsilon$;
- Time cost: $O(|Q| \left|N_{d_Q+1}(Q(u_o, G))\right| + |O_s| \left|N_{d_Q+1}(V_{N_u})\right|^{|Q|} + |O_s|^2 \left|N_{d_Q+1}(V_{N_u})\right|)$.

- Computing optimal query rewrites (ExactWhyNot):
  - Following the similar manner with ExactWhy but with MBS contains only relaxation operators;
  - Time cost: $O(|Q||O_s|^{2B}|N_{d_Q}(V_{C_u})|^{|Q|})$.

- A faster heuristic
  - Function $cl(\cdot)$ is not submodular for relaxation operators;
  - Following the similar manner with ApproxWhy
  - Time cost: $O(|Q|\left|N_{d_Q}(V_{C_u})\right| + |O_s|^2\left|N_{d_Q}(V_{C_u})\right|)$.

- Why-empty and Why-so-many
  - Why the answer set is empty? Special cases of Why-not without specifying V_{C_u};
  - Why there exist so many answers? Special cases of Why-not without specifying V_{N_u};

Experiment Setting

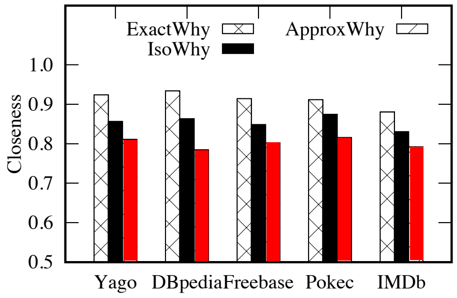
- Dataset

Dataset	Description	# of nodes	# of edges	# of attributes per node
DBPedia	Knowledge Graph	4.86M	15M	9
Yago	Knowledge Graph	1.54M	2.37M	5
Freebase	Knowledge Graph	40.32M	63.2M	8
Pokec	Social Network	1.6M	30.6M	60
IMDb	Movie Network	1.7M	5.2M	6
BSBM	E-commerce	Synthetic		

- Query & Question generation
  - Generate queries controlled by query size and topologies.
  - Randomly select a set of nodes in $Q(u_o, G)$ as V_{N_u}, randomly select V_{C_u} with the same type of u_o.
- Algorithms
  - Why: ExactWhy, ApproxWhy, IsoWhy;
  - Why-not: ExactWhyNot, FastWhyNot, IsoWhyNot.

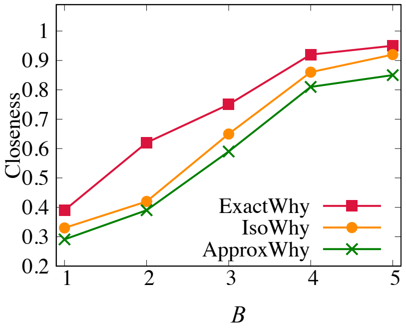
Experiment Result

- Answering Why questions: Effectiveness

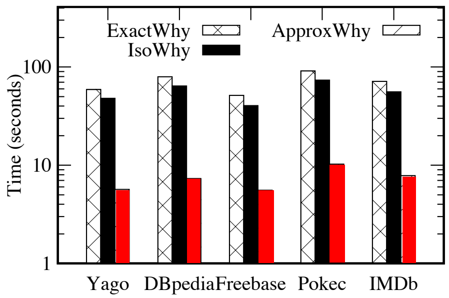


Answer closeness: ApproxWhy achieves at least 85% to their optimal counterpart

Varying cost budget B: it often requires a small B to answer why questions in practice.

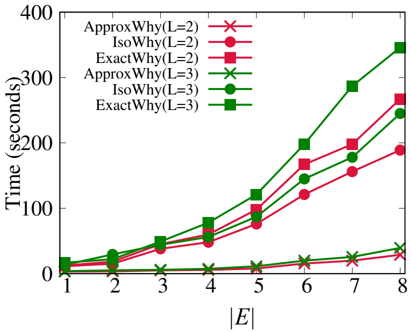


- Answering why questions: Efficiency

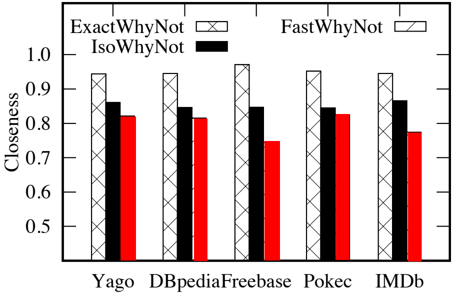


Efficiency: ApproxWhy outperforms ExactWhy and IsoWhy, by 9.7 times and 7.7 times on average

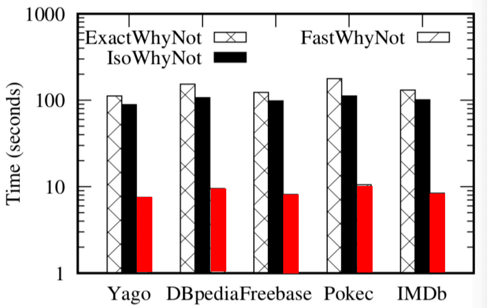
Varying graph size: for practical query with 5 edges, it takes 8.7 seconds to answer a why question.



- Answering Why-not questions:

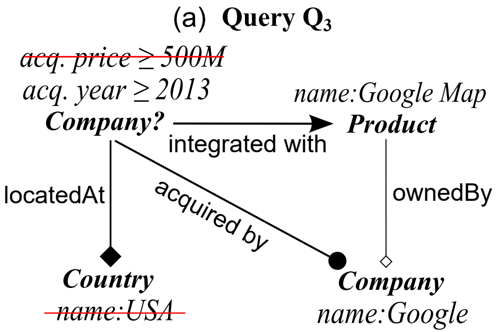


Answer closeness: ApproxWhyNot achieves at least 84% to their optimal counterpart.



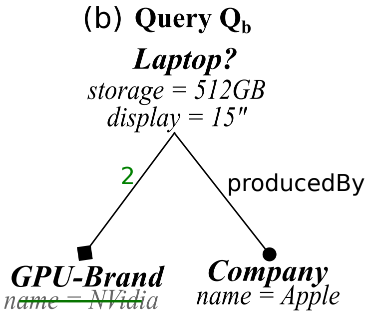
Efficiency: FastWhyNot is feasible, it takes in average 9 seconds to answer a why-not question.

- Case study:



(a) Business inquiry:

- Answer: {*Skybox Imaging*}
- Why-not *Urban-Engines* and *Waze*?
- No price was reported for *Urban-Engines* and *Waze* was founded in Israel.



(b) Product recommendation:

- Answer: {}; (Why-empty)
- MacBook is powered by either Intel or AMD GPU.

Conclusion

- Answering Why-Questions for subgraph queries in multi-attributed graphs
  - We have formalized the problem of answering Why-questions for subgraph queries.
  - We have developed feasible algorithms, from exact and approximation to fast heuristics.
- Following up work (Answering Why-Questions by Exemplars – SIGMOD 2019)
  - Instead of missing/unexpected entities, users input a set of exemplars;
  - Q-Chase, an extension of Chase to characterize graph query rewriting under constraints;
  - Feasible Q-Chase-based algorithms to compute optimal query rewrites (using star views);
  - NAVIGATE: Explainable Visual Graph Exploration by Examples (Demo system).

Sponsored by: NSF USDA HIRP SIEMENS

Thank you!