10. APPENDIX

Proof of Lemma 1. Given $G'_{\mathcal{E}} = G_{\mathcal{E}} \oplus \mathcal{U}$ and $\mathcal{U}$ contains only merge and insert operators, we first prove that there always exists a graph homorphism h' from $G_{\mathcal{E}}$ to $G'_{\mathcal{E}}$. A mapping h' from $G_{\mathcal{E}}$ to $G'_{\mathcal{E}}$ is constructed as follows. For each node $[v] \in V_{\mathcal{E}}$, $h'[v] \in V'_{\mathcal{E}}$ is an equivalent class that contains node $[v]$, where $[v].\mathsf{id}{=}h'[v].\mathsf{id}$. $(h'[v], h'[v'])$ is an edge in $E'_{\mathcal{E}}$ iff. $([v],[v']) \in E_{\mathcal{E}}$. We show that h' is a homorphism when we consider merging nodes and adding missing links. For each $[v] \in V_{\mathcal{E}}$, $L_{\mathcal{E}}([v]) = L'_{\mathcal{E}}(h'[v])$. Similarly, for each $([v],[v']) \in E_{\mathcal{E}}$, $(h'[v], h'[v']) \in E'_{\mathcal{E}}$ and $L_{\mathcal{E}}([v],[v']) = L'_{\mathcal{E}}(h'[v], h'[v'])$. Thus, (1) any node $[v]$ in $G_{\mathcal{E}}$ has a counterpart $h'[v]$ in $G'_{\mathcal{E}}$; similarly for induced edges; (2) if $[v] \sim [v']$ in $G_{\mathcal{E}}$, then $h'[v] \sim h'[v']$ in $G'_{\mathcal{E}}$, and (3) if $([v],[v']) \in P_{\mathcal{E}}(G_{\mathcal{E}})$, $h'([v],[v']) \in P_{\mathcal{E}}(G'_{\mathcal{E}})$.

Validation Problem for NEs. Following theorem 3, we have the following result for NEs.

Theorem 9: *The validation is* coNP*-complete for* NEs □

Proof: We first show that graph keys (GKeys) [14, 16] is a special case of NE. A GKey for entities of type τ is defined as a graph pattern $Q(x)$, where where x is a designated entity variable, *i.e.,* the conditions specified in $Q(x)$ uniquely identify entities of type τ. NE can be specified to GKeys when $P_{\mathcal{E}}$ contains two isomorphic patterns with u_o and u'_o as two designated nodes. The validation is coNP-complete for GKeys [16]. Given a set of GKeys, (1) there exists a NP algorithm to check they are satisfiable, and (2) it is coNP-hard for GKeys with no constant literals. The lower bound can be proved by a reduction from the complement of the 3-colorability problem. Given that GKey is a special case of NE, the validation for NEs alone is coNP-complete. □

Implication Analysis. Given a set of constraints Σ and a constraint $\varphi \notin \Sigma$, we say Σ *implies* φ, denoted as $\Sigma \models \varphi$, if for any graph G, if $G \models \Sigma$, then $G \models \varphi$. The implication problem is to decide for any given finite set of constraints $\Sigma \cup \{\varphi\}$, whether $\Sigma \models \varphi$.

We first study the implication for NEs and EGs separately, *i.e.,* $\Sigma \models \varphi$ contains only NEs or EGs.

Theorem 10: *The* NE *implication is* NP*-complete.* □

Proof: Given that GKey is a special case of NE and the implication problem of GKeys is NP-complete [16], the NE implication is NP-complete. □

Theorem 11: *The* EG *implication is* NP*-complete.* □

To characterize the implication problem, we introduce the notion of embeddings between two EGs.

<u>*Embedding of* EGs.</u> We say a EG $\varphi = P_{\mathcal{E}} \rightarrow \exists r(u,u')$ is embedded in another EG $\varphi' = P'_{\mathcal{E}} \rightarrow \exists r(u,u')$, denoted as $\varphi \preceq \varphi'$, if (1) $P'_{\mathcal{E}}$ is subgraph isomorphic to $P_{\mathcal{E}}$ via a bijection f and their anchored nodes u and u' have the same node labels, respectively; (2) $X' = f^{-1}(X)$, *i.e.,* for each literal $l \in X$, there exists a literal $l' \in X'$ obtained by renaming $u.A$ in l to $f^{-1}(u).A$ according to l and f. We have the following results.

Lemma 12: *Given a set* Σ *of* EGs *and a* EG $\varphi \notin \Sigma$, $\Sigma \models \varphi$ *iff there exists a* EG φ' *in* Σ*, such that* $\varphi' \preceq \varphi$. □

Proof: (**If**). The **If** condition states that given a EG $\varphi = P_{\mathcal{E}} \rightarrow \exists r(u,u') \notin \Sigma$, if there exists a EG $\varphi' = P'_{\mathcal{E}} \rightarrow \exists r(u,u')$ in Σ, such that $\varphi' \preceq \varphi$, then for any graph G, if $G \models \Sigma$, $G \models \varphi$. We perform a case analysis for $P_{\mathcal{E}}$. (1) If $P_{\mathcal{E}}$ has no match, or for every match h, an edge with r exists between $h(u)$ and $h(u')$, then $G \models \varphi$ trivially. (2) Assume $P_{\mathcal{E}}$ has at least one match, and by contradiction, $G \not\models \varphi$. Then there exists a match h, such that there exist no edge with label r between $h(u)$ and $h(u')$. We construct match h' from h for $P'_{\mathcal{E}}$, where each node u' in $P'_{\mathcal{E}}$, $h'(u') = h(f(u'))$. We can verify that h' is a match satisfying $X' = f^{-1}(X)$. $h'(u) = v$ and $h'(u') = v'$, and $G \models \varphi'$, and therefore there exists an edge with label r between $h(u)$ and $h(u')$. This contradicts to that $G \not\models \varphi$. Thus $G \models \varphi$.

(**Only If**). The **Only If** states that if $\Sigma \models \varphi$, then there exists a EG φ' in Σ, such that $\varphi' \preceq \varphi$ via mapping f. Assume there is no such EG φ'. We construct a graph G such that $G \models \Sigma$ and $G \not\models \varphi$.

For each EG $\varphi' = P'_{\mathcal{E}} \rightarrow \exists r(u,u')$ in Σ, we construct a match by enforcing X' via a mapping h', and moreover, there exists an edge with label r between $h'(u)$ and $h'(u')$. Such match always exists as X' is satisfiable. As there exists no EG that can be embedded in φ, there is no f from $P'_{\mathcal{E}}$ to $P_{\mathcal{E}}$ that preserves (i) label equality, or (ii) literal renaming requirement, *i.e.,* there exists a literal in X that has no renamed literal in X' via f^{-1}. For any of these cases, we can always construct a match $P_{\mathcal{E}}(G)$, such that (a) $h(u) = v, h(u') = v'$, and there exists an edge with label r between v and v'. Set $G = \bigcup P'_{\mathcal{E}}(G) \bigcup P_{\mathcal{E}}(G)$. Clearly, $G \models \Sigma$ and $G \not\models \varphi$.

Putting these together, Lemma 12 follows. □

Theorem 13: *The* EGs *implication is* NP*-complete.* □

Proof: It suffices to show the problem of checking whether the condition of Lemma 12 is NP-complete. (1) We develop a NP algorithm to check whether $\Sigma \models \varphi$. The algorithm guesses a mapping f for each $\varphi' \in \Sigma$, and verifies whether $\varphi' \preceq \varphi$ via f. The verification can be done in polynomial time. (2) The hardness of the problem can be easily verified by a reduction from subgraph isomorphism. Putting these together and given Lemma 12, Theorem 13 follows. □

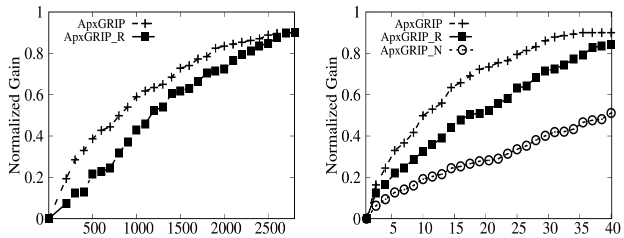
Details of Constraint-level Pruning.

For any operator o that enforces a dynamic constraint φ, procedure Trigger safely prunes all the constraints φ' where $(\varphi, \varphi') \in R_I$ without graph pattern matching. We show the following result.

Lemma 14: *For any pair* $(\varphi, \varphi') \in R_I$*,* φ' *cannot be triggered by any step* $(([v],[v']), \varphi, o)$ *in any* GRIP *sequence.* □

Proof: Any merge or insert enforced by φ revises a node pair $([v],[v'])$ and all the neighbors of the nodes in $[v]$ and $[v']$ from the base graph G. We can verify that for any base graph G and given any condition, the changes introduce no new match for the entity pattern of φ', thus has no trigger in GRIP sequence for φ'. (1) If φ is an EG, then is merges two nodes with label $L(u_o)$. Given that $P'_{\mathcal{E}}$ does not contain any nodes with label $L(u_o)$, φ' can not be triggered. (2) Similarly, if φ is an NE, it adds an link with label r between nodes with label $L(u_o)$ and $L(u'_o)$. Since $P'_{\mathcal{E}}$ does contain such an edge, φ' can not be triggered in any GRIP sequence. □

For instance-level pruning, we have the following result.



(a) B (DBYa) (b) time (s) (DBYa)

Figure 11: Effectiveness

Lemma 15: *Given operator o and any* GRIP *step s $= (([v_o],[v'_o]),\varphi',o')$ triggered by o, (1) $([v_o],[v_o]') \in P'_{\mathcal{E}}(G',o)$ induced by the above cases, and (2) any node pair $([v_o],[v_o]') \notin P'_{\mathcal{E}}(G')$ is not a trigger of φ' for s.* $\square$

Proof: For all four cases, a possible trigger must be a new match of the pattern $P'(u_o,u'_o)$ that fails either entity equivalence or with a missing edge. Upon merge (NE-NE and NE-EG) or insert (EG-NE and EG-EG), such matches can only be identified from the bounded neighborhood of either the newly merged nodes or inserted edges in G'. Specifically for φ' as a NE (NE-NE and EG-NE), new triggers may include one node from the verified matches. Lemma 15 thus follows. $\square$

Proof of Theorem 8. For Why query $\mathsf{why}(o,G,G_{\mathcal{E}})$, given that o is enforced by ρ' and ρ' is a fraction of ρ, any sequence ρ' is a witness.

We next show that given a Why-not query $\mathsf{whyNot}(o,G,G_{\mathcal{E}})$ along with the constraint φ that enforces o, it is NP-hard to verify a GRIP sequence ρ' fails to be a witness, *i.e.,* whether $G'_{\mathcal{E}} \not\models \varphi$. Following the analysis for Theorem 3, the hardness can be shown by a reduction from k-clique problem. Putting these together, the hardness for whyNot is coNP-hard.

Procedure Backward for Why-not questions.

We introduce the Backward procedure for answering whyNot questions. Given a virtual step $s = (([v],[v']),\varphi,o)$, Backward aims to find a set of virtual steps that can *"possibly"* trigger o. (1) It first identifies a set of constraints Σ' that can trigger φ. Σ' is constructed by involving all constraints $\varphi' \in \Sigma$ s.t. $(\varphi',\varphi) \notin R_I$ (see constraint-level pruning in Section 5.2). (2) For each constraint $\varphi' \in \Sigma'$ with pattern $(P'(u_o,u'_o),X')$, Backward finds a set of node pairs within d-hop of $[v]$ and $[v']$, where d is the diameter of the pattern in φ. Each node pair $([v_1],[v_2])$ is a candidate of (u_o,u'_o), *i.e.,* $L([v_1]) = L(u_o)$, $L([v_2]) = L(u'_o)$ and both nodes satisfy the literal constraints in X'. (3) For each node pair, Backward generates an operator $\mathsf{merge}(([v_1],[v_2]),f)$ (resp. $\mathsf{insert}(r([v_1],[v_2])))$ if φ' is a NE (resp. EG). It then constructs a virtual step $s' = (([v_1],[v_2]),\varphi',o')$ and adds it as a independent nodes to $\mathcal{T}$. Finally, it creates an edge from s' to s for each newly constructed virtual step s'.

Variants of provenance. We next introduce how GTrack can be specialized to answer other provenance need. These queries demonstrates how GRIP, a general online framework, takes user feedback as input and generates user-specified results.

- **"What-if?"**. A What-if query $\mathsf{whatIf}(o,G,G_{\mathcal{E}})$ asks "What if o is applied $G_{\mathcal{E}}$", where o is not involved in ρ.
- **"What-if-not?"**. A What-if-not query whatIfNot $(o,G,G_{\mathcal{E}})$ asks "What if o is not applied $G_{\mathcal{E}}$", where o is involved in ρ.

A *witness* for $\mathsf{whatIf}(o,G,G_{\mathcal{E}})$ is a single GRIP sequence ρ', where o is involved in ρ'. Similarly, a GRIP sequence ρ' is a *witness* for whatIfNot $(o,G,G_{\mathcal{E}})$ if it involves o. These two queries are common used in crowdsourcing data integration [40, 30] where the system asks users to select if an operator can be applied or not.

To answering these two types of queries, GTrack leverages ApxGRIP to reconstruct the provenance tree $\mathcal{T}$.

Answering What-if query. To answer a What-if query $\mathsf{whatIf}(o,G,G_{\mathcal{E}})$ where o is not selected by ApxGRIP, GTrack tracks the operator selection process in ApxGRIP and interrupts the process when step s with operator o is dequeued (line 4 in Figure 4). It directly adds s into $\mathcal{U}$ instead of verifying the gain-cost ratio. By invoking Trigger, it adds newly triggered operators into $\mathcal{U}$ and continues the selection process until ApxGRIP terminates. This will generate a new GRIP sequence ρ' and a new provenance tree $\mathcal{T}'$. This sequence ρ' thus leads to result with optimal completeness gain when operator o is enforced.

Answering What-if-not query. Similarly, in order to answer a What-if-not query $\mathsf{whatIf}(o,G,G_{\mathcal{E}})$, GTrack interrupts ApxGRIP by directly discarding o and all operators triggered by it. A new sequence ρ' is thus generated as an answer to this What-if-not query.

Complementary experiment results. *Effectiveness.* Set $B = 2.5K$, we evaluate the effectiveness of ApxGRIP by tracking the change of the normalized gain value. We compare ApxGRIP with it's counterpart ApxGRIP_N and ApxGRIP_R, which does not prioritize the operators (see operator-level pruning). Figure 11 shows the normalized gain with increasing B and time. (1) The quality of graph refinement increases as B and time increase. (2) ApxGRIP converges faster to near-optimal completeness gain due to the optimization techniques. Remarkably, ApxGRIP converges after selecting $2.5K$ operators in less than 40 seconds.

9. REFERENCES

[1] Full version. https://songqi1990.github.io/Full.pdf.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases: the logical level*. 1995.

[3] M. Arenas, L. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, 1999.

[4] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. A knapsack secretary problem with applications. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, pages 16–28. 2007.

[5] I. Bhattacharya and L. Getoor. Entity resolution in graphs. *Mining graph data*, 2006.

[6] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[7] P. Buneman, S. Khanna, and T. Wang-Chiew. Why and where: A characterization of data provenance. In *ICDT*, 2001.

[8] M. Calautti, L. Libkin, and A. Pieris. An operational approach to consistent query answering. In *PODS*, 2018.

[9] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, 2014.

[10] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, and W. Zhang. From data fusion to knowledge fusion. *PVLDB*, 2014.

[11] X. L. Dong and F. Naumann. Data fusion: resolving data conflicts for integration. *PVLDB*, 2(2):1654–1655, 2009.

[12] A. Ebaid, S. Thirumuruganathan, W. G. Aref, A. Elmagarmid, and M. Ouzzani. Explainer: entity resolution explanations. In *ICDE*, 2019.

[13] V. Efthymiou, K. Stefanidis, and V. Christophides. Benchmarking blocking algorithms for web entities. *IEEE Transactions on Big Data*, 2016.

[14] W. Fan, Z. Fan, C. Tian, and X. L. Dong. Keys for graphs. *PVLDB*, 8(12), 2015.

[15] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. Interaction between record matching and data repairing. *SIGMOD*, 2011.

[16] W. Fan and P. Lu. Dependencies for graphs. In *PODS*, 2017.

[17] W. Fan, P. Lu, C. Tian, and J. Zhou. Deducing certain fixes to graphs. *PVLDB*, 2019.

[18] W. Fan, X. Wang, Y. Wu, and J. Xu. Association rules with graph patterns. *PVLDB*, 2015.

[19] D. Firmani, B. Saha, and D. Srivastava. Online entity resolution using an oracle. *PVLDB*, 2016.

[20] L. Galárraga, S. Razniewski, A. Amarilli, and F. M. Suchanek. Predicting completeness in knowledge bases. In *WSDM*, 2017.

[21] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Fast rule mining in ontological knowledge bases with amie++. *VLDBJ*, 2015.

[22] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, 2013.

[23] W. Gatterbauer and D. Suciu. Data conflict resolution using trust mappings. In *SIGMOD*, 2010.

[24] F. Geerts, G. Mecca, P. Papotti, and D. Santoro. The llunatic data-cleaning framework. *PVLDB*, 2013.

[25] S. Guo, X. L. Dong, D. Srivastava, and R. Zajac. Record linkage with uniqueness constraints and erroneous values. *PVLDB*, 2010.

[26] E. N. Hanson and J. Widom. An overview of production rules in database systems. *The Knowledge Engineering Review*, 8(2):121–143, 1993.

[27] C. Kankanamge, S. Sahu, A. Mhedbhi, J. Chen, and S. Salihoglu. Graphflow: An active graph database. In *SIGMOD*, 2017.

[28] P. Lin, Q. Song, J. Shen, and Y. Wu. Discovering graph patterns for fact checking in knowledge graphs. In *DASFAA*, 2018.

[29] H. Ma, M. Alipourlangouri, Y. Wu, F. Chiang, and J. Pi. Ontology-based entity matching in attributed graphs. *PVLDB*, 2019.

[30] R. Meng, H. Xin, L. Chen, and Y. Song. Subjective knowledge acquisition and enrichment powered by crowdsourcing. *arXiv*, 2017.

[31] A. Morteza and C. Fei. Keyminer: Discovering keys for graphs. In *VLDB workshop*, 2018.

[32] W. E. Moustafa, H. Miao, A. Deshpande, and L. Getoor. Grdb: a system for declarative and interactive analysis of noisy information networks. In *SIGMOD*, 2013.

[33] F. Naumann and M. Häussler. Declarative data merging with conflict resolution. 2002.

[34] H. Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.

[35] M. Pershina, M. Yakout, and K. Chakrabarti. Holistic entity matching across knowledge graphs. In *Big Data*, 2015.

[36] P. Rahman, C. Hebert, and A. Nandi. Icarus: minimizing human effort in iterative data completion. *PVLDB*, 2018.

[37] B. Shi and T. Weninger. Discriminative predicate path mining for fact checking in knowledge graphs. *Knowledge-based systems*, 2016.

[38] D. Symeonidou, L. Galárraga, N. Pernelle, F. Saïs, and F. Suchanek. Vickey: mining conditional keys on knowledge bases. In *ISWC*, 2017.

[39] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 2012.

[40] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. Leveraging transitive relations for crowdsourced joins. In *SIGMOD*, 2013.

[41] X. Wang, L. Haas, and A. Meliou. Explaining data integration. *Data Engineering Bulletin*, 2018.

[42] S. E. Whang, D. Marmaros, and H. Garcia-Molina. Pay-as-you-go entity resolution. *TKDE*, 25(5):1111–1124, 2012.

[43] Y. Zhou, D. Chakrabarty, and R. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In *WINE*, 2008.