Path Association Rule Mining

Anonymous Author(s)

ACM Reference Format:
Anonymous Author(s). 2022. Path Association Rule Mining. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

A PROOF

We show proofs of our theorems and lemmas in the main body.

THEOREM 3.1. *If path pattern $p \subseteq p'$, $\mathcal{V}(p) \supseteq \mathcal{V}(p')$.*

Proof. When p is dominated by p', p is not longer than p', all edge labels on p and p' are the same order, and attribute sets on p are subset or equal to those on p'. All vertices that match p' are matched with path patterns whose attributes are subset of p'. Therefore, if path pattern $p \subseteq p'$, $\mathcal{V}(p) \supseteq \mathcal{V}(p')$. □

LEMMA 3.2. *If prefix of path patterns is not frequent, the whole path patterns are not frequent.*

Proof. If v matches $p = (A_0, \ell_1, \cdots, \ell_n, A_m)$, v matches $(A_0, \ell_1, \cdots, \ell_n, A_m)$ where $m < n$. Similarly, if v does note match $(A_0, \ell_1, \cdots, \ell_m, A_m)$, v does not matches $(A_0, \ell_1, \cdots, \ell_n, A_n)$. Therefore, if prefix of path patterns is not frequent, the whole path patterns are not frequent. □

LEMMA 3.3. *If path pattern p is not frequent, p' such that $\forall A'_i \supseteq A_i$ and $\ell_i = \ell'_i$ is not frequent.*

Proof. If v matches p', v matches p because p' is more complex than p. Similarly, if v does note match p, v does not matches p'. Therefore, if p is not frequent, p' is not frequent. □

LEMMA 3.5. *Given length n and (ℓ_n, A_n), the maximum number of matched vertices for path patterns of $n-1$ length concatenated with (ℓ_n, A_n) is $|\mathcal{E}(A_n, \ell_n)| \cdot (d_m)^{n-1})$ where $n > 0$ and d_m indicates the maximum in-degrees among vertices.*

Proof. $|\mathcal{E}(A_n, \ell_n)|$ indicates the number of edges with ℓ_n connecting to vertices whose attribute set includes A_n. Given $\langle A, \ell_n, A_n\rangle$ as path pattern, the maximum number of vertices that match the path pattern is $|\mathcal{E}(A_n, \ell_n)|$. When the length of path patterns, the number of paths whose targets' attributes include A_n exponentially increase by d_m. Therefore, the maximum number of matched vertices for path patterns of $n-1$ length concatenated with (ℓ_n, A_n) is $|\mathcal{E}(A_n, \ell_n)| \cdot (d_m)^{n-1})$. □

LEMMA 3.6. *Given A and ℓ, the maximum number of matched vertices for $\langle A, \ell, \ldots\rangle$ is $|\mathcal{V}(A, \ell)|$.*

Proof. This is obvious because the maximum number of vertices that can be sources of paths that match $\langle A, \ell, \ldots\rangle$ is the number of vertices that have edges with ℓ. □

THEOREM 4.1. *Given frequent path pattern p, the missing probability is $P\left(\theta \geq \frac{|\mathcal{V}(p)|(d_m)^{\psi(n-1)}}{p_m(d_p)^{n-1}}\right)$, where p_m is a probability that vertex matches with p among paths with the suffix and d_p is the average degree of actual paths.*

Proof. We remove the suffix if $|\mathcal{E}(A, \ell)|(d_m)^{\psi(n-1)}$ is not larger than θ. We define $|P|$ as the number of paths from any sources to the target of suffix, and $|P| = \mathcal{V}(p) \cdot p_m$. Furthermore, $|P| = |\mathcal{E}(A, \ell)|(d_p)^{n-1}$. We can derive the following probability:

$$P\left(\theta \geq |\mathcal{E}(A, \ell)|(d_m)^{\psi(n-1)}\right) \tag{1}$$

$$= \quad P\left(\theta \geq \frac{|P|}{(d_p)^{n-1}}(d_m)^{\psi(n-1)}\right) \tag{2}$$

$$= \quad P\left(\theta \geq \frac{|\mathcal{V}(p)|(d_m)^{\psi(n-1)}}{p_m(d_p)^{n-1}}\right) \tag{3}$$

□

B SPACE COMPLEXITY

The space complexity of our algorithm is $O(|\mathcal{V}| + |\mathcal{E}| + |\mathcal{A}| + |C^A| + |C^S| + |C^*| + |C^R| + \sum_i^k |\mathcal{P}_i| + |\mathcal{P}^*| + |\mathcal{R}|)$, where $|C^A|$, $|C^S|$ $|C^*|$, and $|C^R|$ are the maximum sizes of $|C_i^A|$, $|C_i^S|$ $|C_i^*|$, and $|C_i^R|$ for all i, respectively. Generally, the memory space does not become an issue empirically compared with the running time.

C DATASET DETAILS

We describe graphs that we used in our experimental studies in detail.

- Nell [6] is a knowledge graph crawled from the web. Its attributes are, for example, ceo, musician, company, and university, and edge labels are, for example, comapanyceo, competeswith, and workers.
- DBpedia [3] is a knowledge graph extracted from wikipedia data. Its attributes are, for example, actor, award, person, and place, and edge labels are, for example, child, spouse, and deathPlace.
- Pokec is a social network service in which most users are Slovenian. Its attributes are, for example, age, gender, city, and music, edge labels are, for example, follows, likes, and locatedIn. The number of edge label follows is very large, so we divide them according to out-degrees, such as small follows and large follows.

Table 1 shows data statistics.

D ADDITIONAL EXPERIMENT

We show the impact of length k of paths and approximation factors.

Figure 1 shows the run time of each method varying the length k. We set 1,800, 140,000, and 50,000 as θ. We do not show the

Conference'17, July 2017, Washington, DC, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1: Data statistics.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{A}\rvert$	Avg. Attr.
Nell	46,682	231,634	821	266	1.5
DBpedia	1,477,796	2,920,168	504	239	2.7
Pokec	1,666,426	34,817,514	9	36302	1.1

Table 2: Impact of approximation factors to accuracy in Nell

	CR		SA		CR+SA	
	Recall	Precision	Recall	Precision	Recall	Precision
0.2	0.0461	1	0.9999	0.495	0.046	0.500
0.4	0.233	1	0.9999	0.999	0.233	0.998
0.6	0.532	1	0.9999	0.9999	0.532	0.9999
0.8	0.797	1	0.9999	0.9999	0.797	0.9999

Table 3: Impact of approximation factors to accuracy in DBpedia

	CR		SA		CR+SA	
	Recall	Precision	Recall	Precision	Recall	Precision
0.2	1.0	1.0	1.0	1.0	1.0	1.0
0.4	1.0	1.0	1.0	1.0	1.0	1.0
0.6	1.0	1.0	1.0	1.0	1.0	1.0
0.8	1.0	1.0	1.0	1.0	1.0	1.0

performance of the baseline because it does not finish within 24 hours when $k = 3$ in all datasets. Generally, as length increases, the run time increases because the numbers of rules and path patterns increase. In Nell, the number of rules drastically increase when $k = 4$, our algorithm did not finish within 24 hours. In DBpedia, the number path patterns does not increase when k is larger than 2, so the run time does not increase. From these results, we can confirm our approximation methods work well when k is large.

Figure 2 shows the run time varying approximation factor. The approximation factor indicates candidate reduction ψ factor and sampling rate ρ for CR and SA, respectively. From this result, we can see that when the approximation factors are small, the run time is small. However, in Nell, the run time of Our w/ SA is large when $\rho = 0.2$. This is because the number of false negatives is large, so the frequent rule discovery step takes a large time.

Tables 2–4 show the recall and precision varying with approximation factors in each dataset. We can see that the accuracy are quite high in DBpedia and pokec, while in Nell, CR and SA do not work well when approximation factors are small. This is because the size of Nell is small compared with DBpedia and pokec, so the candidate reduction and sampling do not work well in some cases.

Table 5 shows the run time of our methods that we do not use some techniques. "Ours w/o suffix pruning" indicates our method that does not use suffix pruning. "Ours w/o cost-based parallel" indicates our method that randomly assigns the same numbers of vertices to threads without considering estimated costs. From these results, Ours is drastically fast compared with Baseline. The effectiveness of each technique depends on dataset. For example, suffix pruning works well in Pokec, while does not in Nell. When the number of attributes is large, the suffix pruning works well. In parallelization, in Nell and DBpedia, cost-based sampling slightly increase the efficiency, while in Pokec, slightly decreases. In Pokec, the costs of vertices are very similar, so the costs of threads become similar even if we do not estimate their costs.

Table 4: Impact of approximation factors to accuracy in Pokec

	CR		SA		CR+SA	
	Recall	Precision	Recall	Precision	Recall	Precision
0.2	1.0	1.0	0.934	0.983	0.934	0.983
0.4	1.0	1.0	0.967	0.983	0.967	0.983
0.6	1.0	1.0	0.984	0.986	0.984	0.984
0.8	1.0	1.0	0.984	0.984	0.984	0.984

Table 5: Ablation study

	Nell	DBpedia	Pokec
Ours	341.7	15028.3	4038.5
Baseline	525.4	32907	–
Ours w/o suffix pruning	304.4	15466.4	6171.3
Ours w/o cost-based parallel	360.4	15446.8	4006.9

E ADDITIONAL RELATED WORK

We review studies related to the path association rule mining in four categories; (1) support measure on graphs, (2) association rules, (3) graph pattern mining and matching, and (4) path query.

Graph association rule mining. We show the semantics of each graph association rule mining in Table 6.

Graph pattern mining and matching. A number of algorithms have been developed for graph pattern matching [7, 13]. Graph pattern mining algorithms possibly accelerate discovering graph association rules. Similarly, algorithms for isomorphic subgraph matching, e.g., [8, 12], are developed for efficiently discovering matched patterns in a single large graph. These methods are not suitable for discovering frequent path patterns because they require to find subgraphs from scratch repeatedly. Our algorithm maintains the targets of paths to efficiently extend the path patterns and uses anti-monotonic properties to reduce the number of candidates. In addition, graph pattern mining and matching techniques do not handle reachability path patterns.

Path query. The path association rule aims to discover regularities between paths instead of subgraphs. Querying path patterns are actively studied due to their usefulness [1, 2]. In recent real-life query logs for Wikidata [14], more than 90 % of queries are path patterns [4, 5]. From this fact, graph analysis often utilizes path patterns instead of graph patterns to understand real-world relationships. Therefore, path association rule mining can be useful in real-world analysis on many applications.

REFERENCES

[1] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoc. 2017. Foundations of modern query languages for graph databases. *ACM Comput. Surv.* 50, 5 (2017), 68:1–68:40.

[2] Renzo Angles, Juan L. Reutter, and Hannes Voigt. 2019. Graph query languages. In *Encyclopedia of Big Data Technologies*.

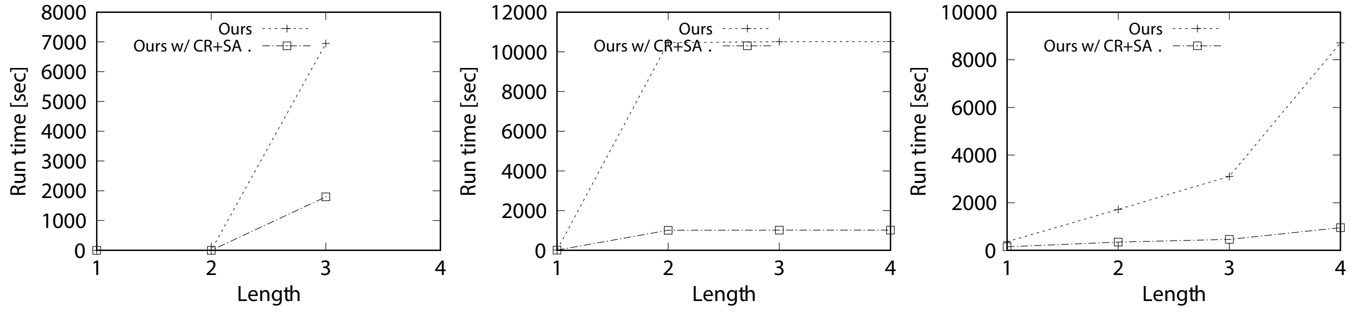
[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*. 722–735.

[4] Angela Bonifati, Wim Martens, and Thomas Timm. 2019. Navigating the Maze of Wikidata Query Logs. In *WWW*. 127–138.

[5] Angela Bonifati, Wim Martens, and Thomas Timm. 2020. An analytical study of large SPARQL query logs. *The VLDB Journal* (2020), 655–679.

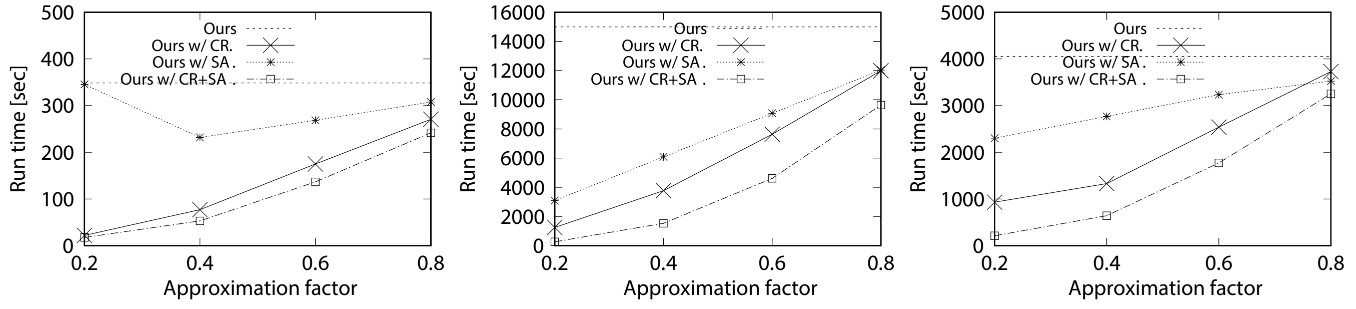
[6] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*. 1306–1313.

[7] Mohammed Elseidy, Ehab Abdelhamid, Spiros Skiadopoulos, and Panos Kalnis. 2014. Grami: Frequent subgraph and pattern mining in a single large graph. *PVLDB* 7, 7 (2014), 517–528.



(a) Nell (b) DBpedia (c) Pokec

Figure 1: Impact of length to run time



(a) Nell (b) DBpedia (c) Pokec

Figure 2: Impact of approximation factors to run time

Table 6: Comparison of association rule mining on a single large graph. $\mathcal{L}_v$ indicates a set of vertex labels.

	$\mathcal{G}$	G_X	G_Y	G_X and G_Y	Algorithm
GPAR [10]	$(\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{L}_v)$	Vertex-centric subgraph	Single edge (or empty)	$V.G_Y \subseteq V.G_X$	top-k diverse patterns
Extending GPAR [15]	$(\mathcal{V}, \mathcal{E}, \phi, \mathcal{A})$	Subgraph (at least one edge)	Subgraph (at least one edge)	$V.G_Y \subseteq V.G_X$ and no common edges	Frequent patterns
QGAR [11]	$(\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{L}_v)$	Quantified subgraph	Quantified subgraph	$V.G_X \subseteq V.G_Y$	Application-specific top-m pattern
GAR [9]	$(\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{A})$	Subgraph	single edge or attributes	$V.G_X \subseteq V.G_Y$	None
Ours	$(\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{A})$	Path	Path	No restriction	Frequent patterns

[8] Grace Fan, Wenfei Fan, Yuanhao Li, Ping Lu, Chao Tian, and Jingren Zhou. 2020. Extending Graph Patterns with Conditions. In *SIGMOD*. 715–729.
[9] Wenfei Fan, Wenzhi Fu, Ruochun Jin, Ping Lu, and Chao Tian. 2022. Discovering association rules from big graphs. *PVLDB* 15, 7 (2022), 1479–1492.
[10] Wenfei Fan, Xin Wang, Yinghui Wu, and Jingbo Xu. 2015. Association rules with graph patterns. *PVLDB* 8, 12 (2015), 1502–1513.
[11] Wenfei Fan, Yinghui Wu, and Jingbo Xu. 2016. Adding counting quantifiers to graph patterns. In *SIGMOD*. 1215–1230.
[12] Myoungji Han, Hyunjoon Kim, Geonmo Gu, Kunsoo Park, and Wook-Shin Han. 2019. Efficient subgraph matching: Harmonizing dynamic programming, adaptive matching order, and failing set together. In *SIGMOD*. 1429–1446.
[13] Prakash Shelokar, Arnaud Quirin, and Óscar Cordón. 2014. Three-objective subgraph mining using multiobjective evolutionary programming. *J. Comput. System Sci.* 80, 1 (2014), 16–26.
[14] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.
[15] Xin Wang, Yang Xu, and Huayi Zhan. 2020. Extending association rules with graph patterns. *Expert Systems with Applications* 141 (2020), 112897.