Pattern Mining in Linked Data by Edge-Labeling

Xiang Zhang* and Wenyao Cheng

Abstract: Link patterns are consensus practices characterizing how different types of objects are typically interlinked in linked data. Mining link patterns in large-scale linked data has been inefficient due to the computational complexity of mining algorithms and memory limitations. To improve scalability, partitioning strategies for pattern mining have been proposed. But the efficiency and completeness of mining results are still under discussion. In this paper, we propose a novel partitioning strategy for mining link patterns in large-scale linked data, in which linked data is partitioned according to edge-labeling rules: Edges are grouped into a primary multi-partition according to edge labels. A feedback mechanism is proposed to produce a secondary bi-partition according to a quick mining process. Local discovered link patterns in partitions are then merged into global patterns. Experiments show that our partition strategy is feasible and efficient.

Key words: link pattern; labeling; partitioning; scalability evaluation

1 Introduction

With the growth of the semantic web in this decade, linked data has become a popular data representation, providing an open data model for exposing, sharing, and connecting data using URIs and RDF. As indicated by W3C, many linked data sources open their linked data to public access, and many are as large as billions of triples.

Semantic web mining in linked data is attracting more and more attention. Discovering link patterns in linked data has become a very interesting topic, as first described in Ref. [1]. Link patterns are consensus practices characterizing how different types of objects are frequently interlinked. For example, in certain linked data, a Researcher focuses on a ResearchArea, and publishes some Papers in Proceedings of a Conference. In addition, he knows some Researchers in the same ResearchArea. Link patterns are critical in several research topics, such as discovering meaningful semantic associations[2] and characterizing distributed RDF repositories[3].

It is straightforward to discover link patterns with the help of frequent pattern mining algorithms developed for general data mining, such as gSpan[4] or CloseGraph[5]. However, as the volume of linked data grows, the efficiency of pattern mining becomes rather low, due to mining algorithm complexity, discussed in Ref.[1]. Furthermore, existing pattern mining algorithms usually assume that the dataset can fit into main memory, while massive linked data, such as DBpedia, are composed of billions of triples that far exceed current memory limitations. To improve the scalability and efficiency of mining, a partition strategy is needed.

In this paper, we propose an edge-labeling partition strategy for efficient mining of link patterns. Our contributions lie in two areas. First, we describe two different labeling rules for each edge in our graph model. A primary multi-partitioning is generated using edge labels. Second, we propose a quick mining process to provide feedback for a secondary bi-partitioning.
This will further improve the mining efficiency.

2 Link Pattern

Link patterns are frequent and typical styles of how different types of objects are interlinked in linked data, which were first defined in Ref. [1]. Link patterns cannot be directly mined from RDF graphs of linked data. Object types are core elements in link patterns. However, in RDF graphs, object types are implicit, and can only be determined by reasoning according to RDF semantics. In Ref. [1], a Typed Object Graph (TOG) is proposed as the graph model for mining of link patterns, since each TOG is derived from a certain RDF graph, and explicitly embodies object type information. To be brief, we do not repeat the definition of TOG and link patterns in this paper, but offer an example below.

As shown in Fig. 1a, a fragment of a TOG describes a fact: Rudi and Thanh are co-authors of a research paper in the proceedings of the ISWC2011 conference. Each object in this fragment is tagged with a corresponding type. This fragment is from the Semantic Web Dog Food (SWDF) dataset, and this kind of co-author association is very frequent and typical in this dataset. Thus we discover a link pattern, shown in Fig. 1b. From the example, we see that a TOG is derived from an RDF graph in linked data, in which each triple is extended to a link quintuple, additionally containing the types of the subject and the object in the original triple. And a link pattern is a frequent and scheme-level template for a subgraph in TOG.

3 Architecture

As shown in Fig. 2, a TOG derived from linked data is transferred to two alternative labelers: Baseline Labeler and Optimized Labeler. The two labelers assign a number to each edge in the linked data, using different labeling rules. A Multi-partitioner analyzes labeled TOGs, and groups all labeled edges into primary partitions. Then Quick Miner quickly explores each partition to get an estimate of how many link patterns will be discovered in each partition. Feedback will be sent to a bi-partitioner, indicating that some partitions should be further bi-partitioned. Then a Complete Miner is used to discover link patterns in both primary multi-partitions and secondary bi-partitions. As a final step, local link patterns are merged into global patterns in Pattern Merger.

4 Labeling Edges in TOG

There have been discussions on various partition strategies for general-purpose pattern mining. For example, Wang et al. [6] proposed a graph partition strategy for graph mining, in which the graph dataset is divided into smaller and more manageable subgraphs. In their work, two criteria are introduced to minimize the connectivity between the subgraphs, and to isolate frequently updated vertices to a subgraph. Motivated by their work, we propose an edge-labeling partition strategy. The basic criterion for our strategy is making each partition a standalone part of linked data for mining one-edge link patterns. This is done by an edge-labeling approach, which guarantees that edges with different labels belong to different one-edge link patterns. An edge-labeling approach can also guarantee that locally discovered link patterns are also global link patterns.

Definition 1 Edge Label in TOG

Given the vertex set \( V(d) \) and edge set \( E(d) \) of the TOG of linked data \( d \), \( L(V(d) \cup E(d)) \) is a labeling rule, which assigns a number to each vertex and edge. Given an edge \( e = (s, p, o, \text{type}(s, d), \text{type}(o, d)) \in E(d) \), \( L(e) = \max(L(s), L(o)) \) is called the edge label of \( e \).

We propose two labeling rules: a baseline labeling rule and an optimized labeling rule. In a baseline labeling rule, edges are labeled with a number indicating the visiting order of one of its vertexes in

![Fig. 1](a) A fragment of a TOG derived from SWDF; (b) the corresponding link pattern.
Fig. 2 Architecture of mining link patterns based on edge-labeling partition strategy.

a breadth-first traversal; in an optimized labeling rule, edges are labeled with a number indicating one of its vertexes ranking in total degrees. The special design of an optimized labeling rule is to reduce the cases of false failure, and to improve the efficiency of mining in each partition.

**Rule 1 Baseline Labeling Rule**  (1) Randomly select a vertex \( v \) in \( d \) as the root vertex, assign 0 to \( v \) as its label, and record the mapping of \( v \)'s type \( \text{type}(v) \) to its label. (2) Breadth-first traverse the neighboring vertex of \( v \). When visiting vertex \( w \), if there is a recorded mapping from type \( \text{type}(w) \) to a number \( n \), assign \( n \) to \( w \); otherwise, assign a number larger than the current maximum number to \( w \) as its label, and record the mapping of type \( \text{type}(w) \) to its label. (3) For each edge in TOG, use the larger of the labels of its subject and object as its label.

From Rule 1, we see that if two edges in TOG have different labels, they must belong to different one-edge link patterns, because their subjects or objects are different in type. If every pair of partitions has no edges with common labels, it will guarantee that they do not have common one-edge link patterns. Thus each partition will be standalone in mining one-edge link patterns.

Not all linked data can be partitioned with a given labeling rule. If the linked data contains too many edges with identical labels, it is impossible to group edges into partitions that can all fit into memory. To judge the feasibility of a labeling rule with respect to some linked data, we define failure and success cases of the labeling rule.

**Definition 2 Failure and Success Cases of Labeling Rule** Given the TOG of linked data \( d \) and a labeling rule \( L \), we can divide edges into \( j \) edge sets: \( E(d) = \{E_1(d), E_2(d), \ldots, E_j(d)\} \), according to the edge labels. It is required that edge labels in the same edge set are identical, while edge labels in different edge sets are different. The largest edge set is denoted as \( E_{\max}(d) \). Given memory capacity \( N \) (\( N \) indicates the number of edges the memory can hold for mining), if \( |E_{\max}(d)| > N \), \( d \) is a Failure Case of \( L \); otherwise, \( d \) is called a Success Case of \( L \).

**Definition 3 False Failure** If linked data \( d \) is a Failure Case of labeling rule \( L \), and there exists a different labeling rule \( L' \), and \( d \) is a Success Case of \( L' \), we call \( d \) a False Failure of \( L \).

False Failure means a labeling rule is unsuitable for a certain set of linked data, but there exist other labeling rules that can produce properly-sized partitions for loading into memory. In fact, a baseline labeling rule is prone to producing False Failures. We observe that edges containing a popular type of vertex often have identical labels, and since we cannot place two edges with identical labels into different partitions, the linked dataset becomes a failure case. We can reduce this kind of false failure by a heuristic rule: Assign a small number to popular types of vertexes and a large number to unpopular ones, and the label of an edge will be determined by the labels of relatively unpopular types of vertexes. This will reduce the number of edges with identical labels. We call this rule an optimized labeling rule.

**Rule 2 Optimized Labeling Rule**  (1) For each vertex in a TOG of linked data \( d \), first compute the degree of its type in the TOG (both in-degree and out-degree). (2) Rank all the types according to their degrees in descending order. (3) Label each vertex with the ranking of its type.

5 Partitioning TOG

Let \( N \) denote the memory capacity for pattern mining
and $M$ denote the number of edges in linked data $d$. $d$ must be divided into at least $k = \lceil M/N \rceil$ partitions to guarantee that each partition can be loaded into memory.

The purpose of primary multi-partitioning is to make each partition a standalone part of the linked data for mining one-edge link patterns. Our approach will try to evenly group edges into partitions, to balance the mining efficiency in different partitions. As shown in Fig. 3, multi-partitions are generated in three steps: (1) Edges in linked data $d$ are labeled by labeling rule $L$ and then divided into $j$ edge sets, according to common labels. (2) According to the cardinality of each edge set, sort edge sets in descending order to $d = \{RE_1(d), RE_2(d), \ldots , RE_j(d)\}$. (3) If $d$ is a Success Case of $L$, we prepare $k$ empty partitions. Each partition has a maximal volume of $N$ edges. We select the top-$k$-ranked edge sets and put each of them into a partition separately. Then, each remaining ranked edge set is checked in turn, and is put into a suitable partition according to edge labels; which has the highest connectivity linking to it and plenty of volume to hold it. The algorithm is shown as Algorithm 1.

We rank the edges in the edge set according to their cardinality, so that edges can be evenly grouped into partitions. A balanced partitioning will improve the mining efficiency. The notion of connectivity is defined in Definition 4. It indicates the possibility that a partition has connected edges with a given edge set. To maximize the connectivity in each partition and minimize the connectivity between different partitions will improve the efficiency of the merging process.

**Definition 4 Connectivity**

Given a partition $p$ and an edge set $E_j(d), T(p) \subseteq E_j(d)$ is the set of all object types occurring in $p$, and $T(E_j(d))$ is the set of all object types occurring in $E_j(d)$. The connectivity between $p$ and $E_j(d)$ is defined as $C(p, E_j(d)) = |T(p) \cap T(E_j(d))|$, which is the cardinality of the common types.

### Algorithm 1 Multi-partitioning

**Given:** linked data $d, k = \lceil M/N \rceil$, labeling rule $L$

1. if $d$ is a failure case of $L$, return error;
2. prepare $k$ empty partitions;
3. divide edges in $d$ into $j$ disjoint edge sets:
   \( E(d) = \{ E_1(d), E_2(d), \ldots , E_j(d) \} \) according to edge labels;
4. sort $E(d)$ to $RE(d)$ in descending order by cardinality;
5. put each top-$k$ edge set in $RE(d)$ into an empty partition separately;
6. select the most highly ranked remaining edge set in $RE(d)$, denoted as $RE_k(d)$;
7. put $RE_k(d)$ into partition $p_1$ if: 1) $C(p_1, RE_k(d))$ is maximal, and 2) $|p_1| + |RE_k(d)| \leq N$;
8. discard $RE_k(d)$ from $RE(d)$;
9. repeat step 6 to step 9 until $RE(d)$ becomes empty.

**Output:** Partitions: $P_1, P_2, \ldots , P_k$;

![Fig. 3 An illustration of primary multi-partitioning.](image-url)
parameter is max-edge, which limits the maximal edges of discovered link patterns. As shown by the experiments in Ref. [1], with the increase of min-sup and the decrease of max-edge, fewer link patterns will be discovered, and mining time consumption will be greatly reduced, even for very large-scale linked datasets. This indicates that we can use a special setting of min-sup and max-edge, to quickly detect all partitions and predict their differences in terms of the number of potential link patterns. For productive partitions, feedback will be issued, and a further bi-partitioning will be made, as given in Algorithm 2.

7 Merging Link Patterns

Local mining of each partition can discover the complete set of all one-edge link patterns, but some multi-edge link patterns will be missed if we simply combine all local patterns. A special bi-merging process is employed to yield a complete set of global patterns. Our idea is based on the MerJoin operation proposed in Ref. [6]. The main difference between our work and theirs is: When merging 1-edge patterns to get 2-edge patterns, our approach is based on one common vertex label, and our approach can remove patterns whose DFS codes are not minimal.

As shown in Algorithm 3, given a linked dataset \(d\), and supposing \(d\) is divided into partitions \(p_1\) and \(p_2\), and the sets of local patterns are \(lp(p_1)\) and \(lp(p_2)\), then the output is the complete set of link patterns \(lp(d)\). \(lp^i(p_i)\) denotes the set of \(k\)-edge local patterns in \(p_i\). The MF operation represents the removal of patterns whose support is less than min-sup or whose DFS code is not minimal. \(lp^k(p_i)\) represents the result of merging \(lp^k(p_i)\) and \(lp^k(p_j)\) based on \(k - 1\) common edges. Specifically, \(lp^1(p_i)\) represents the result of merging \(lp^1(p_i)\) and \(lp^1(p_j)\), based on one common vertex label. \(M_i\) represents the merged pattern set with \(i\)-edges.

8 Experiments

Our experiments are done on two datasets: SWDF, which is a well-known and widely-used linked dataset for scholars. The other dataset is a subset of DBpedia, which contains data extracted from Wikipedia. The entire DBpedia dataset describes more than 3.64 million objects, with more than 1 billion triples. In our experiments, we randomly extract more than 41 thousand triples from DBpedia.

Table 1 gives an overview of the two datasets. The column "Number of TOGs" represents the number of TOGs derived from linked data, which is equal to the number of RDF documents in the linked data. Our experiments are done on two datasets: SWDF and Sub-DBpedia.

| Dataset       | Number of triples | Number of objects | Number of object links | Number of TOGs |
|---------------|-------------------|-------------------|-----------------------|----------------|
| SWDF          | 166 083           | 16 281            | 54 540                | 148            |
| Sub-DBpedia   | 412 166           | 41 982            | 92 930                | 1              |

8.1 Evaluation of mining time efficiency

Table 2 shows the statistics of mining results on both datasets. min-sup and max-edge are specified as 10 000 and 2 for a quick mining and 1000 and 4 for a complete mining. \(k\) is the number of primary partitions, and \(k = 1\) represents the original gSpan without partitioning.
Table 2 Statistics of mining results.

| Dataset  | $k$ Partition | Number of edges | Number of locally discovered link patterns | Number of link patterns discovered in the merge process |
|----------|----------------|-----------------|---------------------------------------------|--------------------------------------------------------|
| SWDF     | $d$            | 54540           | 1547                                        | 0                                                      |
|          | $p_1$          | 27270           | 481                                         | 1024                                                   |
|          | $p_2$          | 27270           | 42                                          |                                                        |
|          | $d'$           | 92930           | 2987                                        | 0                                                      |
|          | $p_1$          | 46465           | 672                                         | 671                                                    |
|          | $p_2$          | 46465           | 1644                                        |                                                        |
| Sub-DBpedia | $p_1$          | 31310           | 498                                         |                                                        |
|          | $p_2$          | 31310           | 1108                                        | 870                                                    |
|          | $p_3$          | 31310           | 511                                         |                                                        |
|          | $p_4$          | 23198           | 450                                         |                                                        |
|          | $p_2$          | 23267           | 501                                         |                                                        |
|          | $p_3$          | 23233           | 548                                         | 1009                                                   |
|          | $p_4$          | 23232           | 479                                         |                                                        |

The structure of SWDF is special: Edges are tightly connected by some popular objects, and can only divide it into two partitions. Sub-DBpedia has a looser structure than SWDF, and it can be divided into more partitions. It is obvious that our optimized labeling rule can group edges evenly into each partition.

In Figs. 4 and 5, columns in black represent the time consumption for labeling and partitioning, red represents mining, and blue represents merging. Each part of the time consumption is also presented. Time consumption of labeling and partitioning is too small to clearly display. Compared to the original gSpan, the mining time efficiency is greatly improved by using edge-labeling partitioning. For both datasets, mining time is reduced to half or one-third of unpartitioned dataset. In sub-DBpedia, with the increase in the number of partitions, there is an improvement in the mining efficiency in each partition, but more time is needed for merging local patterns into global patterns.

8.2 Evaluation of optimized labeling rule

As a heuristic rule, an optimized labeling rule will theoretically reduce the number of false failure cases, and improve mining efficiency by producing more balanced partitions than the baseline labeling rule. We perform experiments on both datasets to evaluate the optimized labeling rule effectiveness.

Figure 6 shows that in both datasets, and with various partition numbers, the optimized labeling rule shows better performance than the baseline labeling rule in terms of merging time. In sub-DBpedia, although the optimized labeling rule uses more time in mining, it still outperforms the baseline rule in overall efficiency.

8.3 Evaluation of feedback and bi-partitioning

Besides primary multi-partitioning, we use a quick mining process to detect the difference of potential link patterns in different partitions. Where some partitions have many more link patterns than others, feedback is generated from mining to partitioning, and a second bi-partitioning is performed.

Since SWDF can only be divided into two partitions, we evaluate the effectiveness of feedback and bi-partitioning on Sub-DBpedia. In Fig. 7, the $x$-coordinate is the number of primary multi-partitions. When $k$ is 2, a further bi-partitioning will greatly reduce the time consumption of mining; but when $k$ increases, the efficiency difference between mining with feedback and mining without feedback becomes more and more indistinct. This indicates that when the number of
primary partitions increases, the impact of secondary partitioning declines.

9 Related Work

Besides gSpan and Closegraph, there are other widely-used frequent pattern mining algorithms, including AGM[7], Gaston[8], SUBDUE[9], FFSM[10], and SPIN[11]. In Ref. [12], Karypis and Kumar gave an early study of techniques of graph partitioning in databases and developed the software package METIS, which is based on multi-level graph partitioning. In Ref. [13], an algorithm called ADIMIE was designed for mining graphs on large disk-based databases. Wang et al.[6] proposed a partition-based algorithm, PartMiner, to which our merging approach is very similar. But as pointed out in Ref. [13], PartMiner cannot find the correct complete set of frequent patterns in the original dataset. In Ref. [14], Nguyen et al. proposed some optimization techniques in the phase of partitioning and combining.

10 Conclusion and Future Work

In this paper, we propose a novel partition strategy for efficiently mining link patterns, based on edge labeling. We introduce two labeling rules. A primary multi-partitioning aims at producing a set of balanced partitions, and it is guaranteed that each partition is standalone for mining one-edge link patterns. A quick mining process is proposed to provide feedback to multi-partitioning, and a secondary bi-partitioning will further improve mining efficiency. Locally mined patterns are then merged into global patterns. Experiments on two datasets show that our partition strategy is feasible and efficient for mining link patterns.

In our future work, we will explore parallel mining of link patterns to improve mining efficiency. And we will also study the approach of reducing candidate patterns by means of semantic filtering.

Acknowledgment

The work was supported by the National High-Tech Research and Development (863) Program of China (No. 2015AA015406) and the Open Project of Jiangsu Key Laboratory of Data Engineering and Knowledge Service (No. DEKS2014KT002). We would like to thank Xing Li for his valuable suggestions and his work on related experiments.

References

[1] X. Zhang, C. F. Zhao, and P. Wang, Mining link patterns in linked data, presented at the 13th International Conference on Web Age Information Management, Harbin, China, 2012.
[2] A. Sheth, B. Aleman-Meza, B. Arpinar, C. Bertram, Y. S. Warke, and C. Ramakrishnan, Semantic association
identification and knowledge discovery for national security applications, *Journal of Database Management*, vol. 16, no. 1, pp. 33–53, 2005.

[3] A. Basse, F. Gandon, I. Mirbel, M. Lo, and I. Mirbel, DFS-based frequent graph pattern extraction to characterize the content of RDF triple stores, presented at the Web Science Conference 2010: Extending the Frontiers of Society Online, Raleigh, USA, 2010.

[4] X. F. Yan and J. W. Han, Gspan: Graph-based substructure pattern mining, presented at the 2002 IEEE International Conference on Data Mining, Maebashi, Japan, 2002.

[5] X. F. Yan and J. W. Han, CloseGraph: Mining closed frequent graph patterns, presented at the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington DC, USA, 2003.

[6] J. Wang, W. Hsu, M. L. Lee, and C. Sheng, A partition-based approach to graph mining, presented at the 22nd International Conference on Data Engineering, Atlanta, GA, USA, 2006.

[7] A. Inokuchi, T. Washio, and H. Motoda, An apriori-based algorithm for mining frequent substructures from graph data, presented at the 4th European Symposium on the Principle of Data Mining and Knowledge Discovery, Lyon, France, 2000.

[8] S. Nijssen and J. A. Kok, Quickstart in frequent structure mining can make a difference, presented at the 10th ACM SIGKDD International Conference on Knowledge Discovery in Databases (KDD04), Seattle, WA, USA, 2004.

[9] L. B. Holder, D. J. Cook, and S. Djokic, Substructure discovery in the subdue system, in *Proceedings of the AAAI’94 Workshop Knowledge Discovery in Databases*, Seattle, WA, USA, 1994.

[10] J. Huan, W. Wang, and J. Prins, Efficient mining of frequent subgraph in the presence of isomorphism, presented at the 3rd International Conference on Data Mining, Melbourne, FL, USA, 2003.

[11] J. Huan, W. Wang, J. Prins, and J. Yang, Spin: Mining maximal frequent subgraphs from graph databases, presented at the 10th ACM SIGKDD International Conference on Knowledge Discovery in Databases, Seattle, WA, USA, 2004.

[12] G. Karypis and V. Kumar, Multilevel algorithms for multiconstraint graph partitioning, in *Proceedings of the ACM/IEEE Conference on Supercomputing*, 1998, pp. 343–348.

[13] C. Wang, W. Wang, J. Pei, Y. Zhu, and B. Shi, Scalable mining of large disk-based graph databases, presented at the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Orlando, FL, USA, 2004.

[14] S. N. Nguyen, M. E. Orlowska, and X. Li, Graph mining based on a data partitioning approach, presented at the 19th Conference on Australasian Databases, Wollongong, Australia, 2008.

**Xiang Zhang** received the BS degree in computer science from Nanjing University of Aeronautics and Astronautics in 2001, then received the PhD degree in computer software and theory from Southeast University in 2009. Xiang Zhang is now a lecturer in School of Computer Science and Engineering, Southeast University. His major research interests include semantic web, information retrieval, and data mining.

**Wenyao Cheng** received the BS degree in computer science and technology from Anhui University of Science and Technology in 2011, he is currently pursuing master degree in School of Computer Science and Engineering at Southeast University since 2013. His research interests include data mining and cloud computing.