



Graph Data Retrieval Algorithm for Knowledge Fragmentation

Wang Jingbin and Lin Jing^(✉)

College of Mathematics and Computer Science, Fuzhou University,
Fuzhou 350116, China
527564460@qq.com

Abstract. In this era of big data, data are diversified, strongly connected, fragmented, dynamic, and combined with dynamic knowledge fragments to optimize the distributed storage of graphs and enable fast and efficient knowledge graph query problems. Presently, the distributed storage scheme of graph data has a large number of hop accesses between partitions, which leads to a long retrieval response time and is not conducive to fragment knowledge expansion. According to the characteristics of real-time inflow knowledge fragments and the storage structure and principles of graph databases, the Metis+ algorithm is proposed. The label graph is used as the initial initialization segmentation graph, and it is roughened to reduce the cutting of the large-weight edge. The weighted LND algorithm is proposed to run the balancing strategy for storage and assign the similar nodes and closely related nodes to the same partition to the greatest extent, which minimizes jump accesses between the partitions during retrieval.

Keywords: Knowledge graph · Graph database · Label graph · Knowledge fragmentation

1 Introduction

Presently, the relationship between the graph structure of strong connections and the dynamics of data puts an increasingly high load on storage technology [1]. The succession of problems accompanying the processing of massive amounts of data has prompted the development of large-scale knowledge graphs [2], and with their powerful semantic processing capabilities, new solutions to the problems have been made possible [3]. Combined with the structural characteristics of dynamic fragmentation knowledge, this study focuses on the advantages of a graph database in dealing with strong relational data and fuses loose data fragments to establish a knowledge graph network with strong links.

The core technology of distributed data storage is graph partitioning, which divides the graph into several subgraphs and then assigns these subgraphs to different computing nodes [4]. A large number of domestic and foreign experts have researched and improved on the partitioning methods. Combining the characteristics of the resource description framework (RDF) [5] graph data structure in the Semantic Web and parallel computing in a distributed environment, EAGRE [6] compresses the RDF data graph

into an entity graph and uses the METIS algorithm to divide it. However, because the current distributed storage method of graph data is mostly based on static horizontal segmentation of the file, a query on it may require a large number of jump accesses between the cluster partitions when the graph is traversed, affecting the query performance.

This study aims to minimize the jump access between partitions, achieve an optimal storage to enable fast queries, improve the overall retrieval efficiency, and complete the real-time distributed storage of the dynamic graph database. For the demonstration, we constructed an urban safety knowledge graph in this study, taking into consideration the characteristics of the graph data and the load balancing requirements in distributed clusters. We applied the Metis+ algorithm to optimize the distributed storage process and the segmentation and storage of the original graph data. We then performed a balance strategy to store the real-time inflow of knowledge fragments. Finally, the effectiveness of the storage and retrieval algorithm was verified by experiments.

2 Proposed Method

This paper describes the application of the Metis+ algorithm on the constructed urban safety knowledge graph, which is mainly divided into three phases: (1) RDF data graph to graph database mapping; (2) graph data distributed partitioning; and (3) distributed dynamic knowledge fragment storage. The distributed partitioning of the graph data is further divided into two parts: the roughening combined with heavy edge matching (HEM) [7], an edge fusion algorithm, and the segmentation algorithm combined with the weighted leveled nested dissection (LND) algorithm.

2.1 Graph Roughening Combined with HEM Edge Fusion Algorithm

Suppose there are k partitions in the Neo4j [8] distributed cluster. The storage capacity of each partition is M , and thus, the total cluster capacity is kM . $P = \{P(1), P(2), \dots, P(k)\}$ is the sum of all current partition storage states, and $|P(i)|$ indicates the total number of nodes in the partition with subscript i ($1 \leq i \leq k$).

When the graph $G_i = (V_i, E_i)$ is roughened to the next level graph $G_{i+1} = (V_{i+1}, E_{i+1})$, the matching is performed by selecting a larger weight, which can reduce more weights in the roughening graph. The method proceeds to find the maximum matching of edge weights, that is, to find the vertex V in all adjacent unmatched vertices of the vertex U to maximize the weight of the edge e_{uv} , and the algorithm complexity of the method is $O(|E|)$. Figure 1 shows an example where the label graph GL is initialized to the weighted undirected graph GL_0 .

In Fig. 1, the left part shows a partial label graph GL , and the right side shows a weighted undirected graph GL_0 . The total number of instance nodes is the weight $W(V_i)$ of the node V_i in the weighted undirected graph, and the total number of out-degrees and in-degrees of instances between the tags V_i and V_j is the weight $W(e_{ij})$ of the edge e_{ij} in the weighted undirected graph. The steps of the HEM edge fusion algorithm for the graph $GL_0 = (V_0, E_0)$ are as follows: (1) Fusion operation is performed on the vertices: found $\max(W(e_{ij}))$ in all vertices of GL_0 . At this time, the vertex

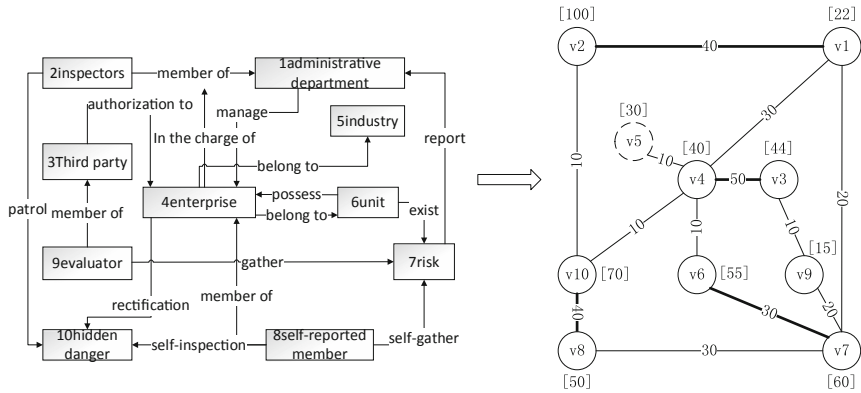


Fig. 1. GL conversion to GL_0

V_i, V_j are merged into a new vertex V_{ij} . (2) When the vertices are merged, the vertex weight is transformed as: $W(V_{ij}) = W(V_i) + W(V_j)$. (3) The edge e_{ij} connecting the vertices V_i, V_j is deleted. (4) Given the threshold θ , the vertex fusion operation is iteratively performed until $|V_m| < \theta$.

2.2 Multi-objective Weighted LND Segmentation Algorithm

After the vertices roughening process, the original graph $GL_0 = (V_0, E_0)$ is coarsened to $GL_m = (V_m, E_m)$ by k steps. Because the hierarchical nesting partitioning LND algorithm does not consider the weight of the vertices, this study hopes that the total weight of the vertices of the graph can be divided into k partitions as evenly as possible when k -way partitioning is performed on the graph GL_m and the sum of the boundary weights in the sub-area is as large as possible.

Definition 3-1. (average vertex weight, Average_W) represents the vertex weight ideally assigned to each partition. It is calculated as follows:

$$Average_W = [\sum_{i=0}^n W(V_{m_i})] / k \tag{1}$$

Definition 3-2. (sum of vertex weights, Sum_W(V_i)) represents the sum of all vertex weights from 0 to label i ($i \geq 0$). It is calculated as follows:

$$Sum_W(V_i) = \sum_{k=0}^i W(V_k) \tag{2}$$

Definition 3-3 (the sum of the maximum edge weights, Sum_BorderW($e_{i,i+1}$)): To obtain the minimum cut edge, when the next hop label from the label i is selected as the vertex of $i + 1$, the accumulation with the largest edge weight should be selected first. It is calculated as follows:

$$\text{Sum_BorderW}(e_{i,i+1}) = \max(\text{Border_W}(e_{i,i+1-j})) \quad (3)$$

Definition 3-4 (multi-objective optimization formula, APP(i, i + 1)): To minimize the difference between the sum of the accumulated vertex weights and the average vertex weights. It is calculated as follows:

$$\text{APP}(i, i + 1) = \min\{\{\text{Sum_W}[\text{Sum_BorderW}(e_{i,i+1})] + \text{Sum_W}(V_i)\} - \text{Average_W}\} \quad (4)$$

2.3 Dynamic Knowledge Fragmentation Storage Strategy

The dynamically influxed knowledge fragment is mapped to the corresponding label set to determine if there is a corresponding label in the k partitions of the distributed cluster. If so, the knowledge fragments are stored in the partition corresponding to the label. Otherwise, the balancing policy for storage is implemented.

Definition 3-5 (Balance strategy): $\min(|P(i)|)$ is chosen. If there are multiple partitions that meet the requirements, one of them is randomly selected. The partition number index is returned according to the following formula.

$$\text{Index} = \text{random}(\{i | \min(|P(i)|), i \in |k|\}) \quad (5)$$

3 Experimental Settings and Result

The experiment used the open domain dataset provided by OpenKG.CN. This study used an urban area for the relevant resource sets. The basic parameters of the dataset are described in Table 1 and the search examples used in the experiment are shown in Table 2. The dataset covers urban air quality, meteorology, related diseases, traffic safety, risk hazards, and many other aspects. The experiment proves that $|V_m| < 100$ is a standard value suitable for ending the roughening process, so the threshold value $\theta = 100$ is selected for the edge fusion algorithm. After the dataset training, the key relationship contribution coefficients are selected as $\alpha = 0.7$, $\beta = 0.3$.

Table 1. Basic parameters of urban datasets

Dataset name	Document size (GB)	Pattern triples	Entity triples (10,000)	Classes	Attributes
Geography	6.20	186	4845	43	51
Meteorological	10.6	186	7596	43	51
Traffic	1.04	487	1253	18	43
Company	3.02	7768	2194	573	2355

Table 2. Search examples

Search	Keyword collection
Q1	世和, 重工业, 注塑机械, 起重机械, 环境污染
Q2	触电, 坍塌, 化学品泄露, 高风险, 中等风险
Q3	化工, 易燃性, 泄露, fire, 爆炸, 高风险, 中等风险
Q4	Aristotle, petrol station, wharf, thermal injury, 稳定性差, 中等风险
Q5	写字楼, 配电箱, 燃气具, 市安全监管局, 市消防监管局, 无防护, 飞溅物, 室内给排水不良, 低风险

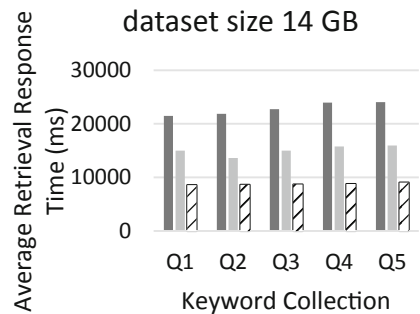
This paper presents a comparative experiment, where the Metis+ algorithm was compared with file horizontal segmentation and the Metis algorithm on the keyword set scale Q1–Q5 with selected data sizes of 3 GB and 14 GB. To eliminate the influence of accidental factors, 10 results were retrieved for each example, and the highest and lowest values were removed to obtain the search average.

It can be seen from Figs. 2 and 3 that when the dataset size and the keyword set size are the same, the average search response time of the Metis+ algorithm is the least. When the data size is 3 GB, the average retrieval response time of the Metis algorithm is 1.12 times that of Metis+, and the file level division is 1.47 times that of Metis+. For a data size of 14 GB, the Metis algorithm is 1.7 times slower than Metis+ and file level division is nearly 2.5 times that of Metis+.



■ Horizontal file division ■ Metis ▨ Metis+

Fig. 2. Comparison of average retrieval response time for 3 GB data scale



■ Horizontal file division ■ Metis □ Metis+

Fig. 3. Comparison of average retrieval response time for 14 GB data scale

When the dataset sizes are the same, the average search response time of the Metis algorithm for Q5 is nearly 1.3 times that of Q1 and the horizontal file division is nearly triple. The Metis+ algorithm used in this study is only doubled. It can be seen from the experiment that Metis+ shows a slower rise in the average retrieval response time as the keyword set size is gradually increased.

4 Conclusion

The algorithm not only realizes the maximum allocation of the same type of nodes and closely related nodes to the same partition, but also minimizes the jump access between partitions during retrieval and improves the retrieval efficiency. In future research, we propose to improve the algorithm by considering the correlation between the semantics in the distributed storage stage of the graph data.

References

1. Pujara, J., Miao, H., Getoor, L., Cohen, W.: Knowledge graph identification. In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8218, pp. 542–557. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41335-3_34
2. Li, J.Z., Hou, L.: Reviews on knowledge graph research. *J. Shanxi Univ. (Nat. Sci. Ed.)* **40**(3), 454–459 (2017). (in Chinese)
3. Cai, D., Hou, D., Qi, Y., Yan, J., Lu, Y.: A distributed rule engine for streaming big data. In: Meng, X., Li, R., Wang, K., Niu, B., Wang, X., Zhao, G. (eds.) WISA 2018. LNCS, vol. 11242, pp. 123–130. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02934-0_12
4. Lasalle, D., Karypis, G.: A parallel hill-climbing refinement algorithm for graph partitioning. In: 45th International Conference on Parallel Processing (ICPP), pp. 236–241 (2016)
5. Leng, Y., Chen, Z., Zhong, F.: BRDPHHC: a balance RDF data partitioning algorithm based on hybrid hierarchical clustering. In: IEEE 12th International Conference on Embedded Software and Systems (ICESS), pp. 1755–1760 (2015)
6. Inokuchi, A., Washio, T., Motoda, H.: Complete mining of frequent patterns from graphs: mining graph data. *Machin. Learn.* **50**(3), 321–354 (2003)
7. Sun, L.Y., Leng, M., Deng, X.C.: Core-sorted heavy-edge matching algorithm based on compressed storage format of graph. *CEA* **47**(10), 41–45 (2011). (in Chinese)
8. Lal, M.: *Neo4j Graph Data Modeling*. Packt Publishing, Birmingham (2015)