Distributed Top-k Subgraph Matching in A Big Graph

Jianliang Gao*, Chuqi Lei*, Ling Tian*, Yuan Ling†, Zheng Chen‡, Bo Song‡
*School of Information Science and Engineering, Central South University, Changsha, China
†Amazon Alexa AI, USA
‡College of Information, Drexel University, Philadelphia, USA

***Abstract*—Subgraph matching query is to find out the subgraphs of data graph G which match a given query graph Q. Traditional methods can not deal with big data graphs due to their high computational complex. In this paper, we propose a distributed top-k subgraph search method over big graphs. The proposed method is designed at the level of single vertex and all vertices obtain their matching state separately without requiring global graph information. Therefore, it can be easily deployed in distributed platform like Hadoop. The evaluations of running time, number of messages and supersteps show the efficiency and scalability of the proposed method.**

I. Introduction

Graph matching has been widely used in many real-world applications [1]. Due to the excessive number of matched subgraphs, it is difficult for the users to select the best results in subgraph matching query over big graphs. Top-k subgraph matching is proposed as a typical graph pattern matching to find out top-k matching subgraphs over a data graph [2]–[5]. Given a query graph Q, top-k matching query reports the top-k matches in a data graph G with the k largest matching "scores" [2]. Therefore, the first step is to define the matching score for top-k matching query. Recently, a similarity measurement method is proposed in [6]. Denote data graph $G(V_1, E_1)$ and query graph $Q(V_2, E_2, f)$, where V_1 and V_2 are sets of vertices, E_1 and E_2 are sets of edges, and f is the function to get the weight of a edge $e \in E_2$. For a subgraph $G' \subset G$, the matching score of G' is:

$$Score(G') = \sum_{e \in E'_m} f(e), \tag{1}$$

where E'_m is the set of edges of Q which are matched by the edges of G'. The set of top-k approximate matching subgraphs is denoted as $Top(G, Q, k)$ and formulated as:

$$Top(G, Q, k) = \{G' | G' \in \arg\max \sum_{i=1}^{k} Score(G'(i))\}, \tag{2}$$

where G' is a subgraph of G, and $G'(i) \neq G'(j)$ if $i \neq j$.

Given a query graph Q and a data graph G, top-k subgraph matching query is to find k matching subgraphs which have the top-k highest matching scores. In real world applications, data graph is typically large, even including billions of nodes [4]. The large scale graphs give rise to the serious problems with the subgraph query algorithms such as computing complexity. Therefore, it is urgent to develop distributed method for subgraph matching over big graphs.

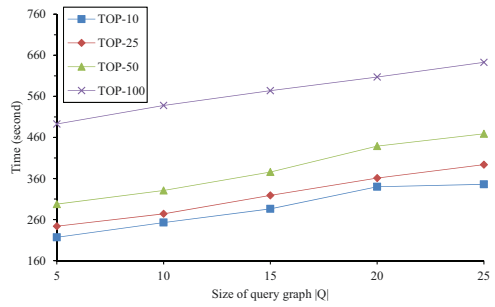
II. Distributed Top-k subgraph matching Algorithm

For query focus based approximate graph simulation, the top-k subgraph search can be simplified as finding the corresponding vertices of query focus in top-k approximate matching subgraphs [5]. To implement top-k subgraph search, the key point is to design algorithm for any vertex which can run simultaneously. Bulk synchronous parallel (BSP) model is adopted for distributed top-k subgraph search. In BSP model, a computation proceeds in a series of global supersteps, which consists of three components: concurrent computation, communication and barrier synchronisation. In a superstep, a vertex not only sends message, but also receives messages. All vertices must stop and wait for a synchronization before next superstep. In the following, the supersteps for top-k subgraph matching query is designed.

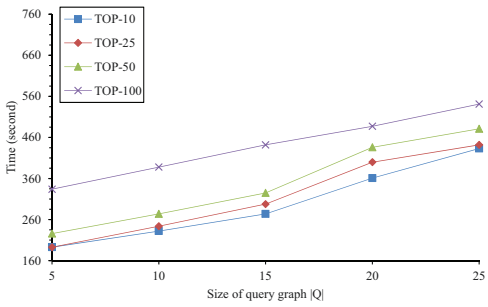
These supersteps consist of initial supersteps, iterative supersteps, and terminated supersteps. At the initial supersteps, each vertex exchanges messages with its parent nodes and child nodes to build the topological view. They also compare their labels with that of the query graph. If there is no matched label in the query graph, the corresponding vertex of data graph is set as inactive and will not attend the further operations. In iterative supersteps, each vertex updates its matching score according to the receiving messages from their parent nodes nodes and child nodes. Due to the limited view of each vertex, it needs multiple supersteps to get the subgraph matching score. With the iterative supersteps, vertex has the overview of subgraph. Therefore, query focus can get the final matching score of subgraph as the proceed of supersteps. In the iterative supersteps, there are mainly two operations. One is to update the matching score and the current state of vertex. If the matching score does not change again, the state of the vertex will become in-active. The other is to aggregative messages by sending their matching score to an aggregative center. In the aggregative center, the current top-k matching scores are kept and updated in each superstep. When all vertices become in-active, it enters terminated supersteps. The list in the aggregative center is the results of top-k matching subgraphs.

In these supersteps, all vertices can run simultaneously including sending messages, receiving messages, and computing according to the received messages. Therefore, the algorithm

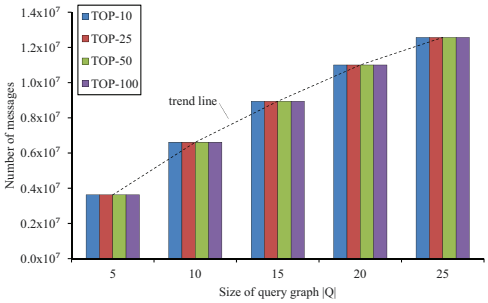
978-1-5386-5035-6/18/$31.00 ©2018 IEEE



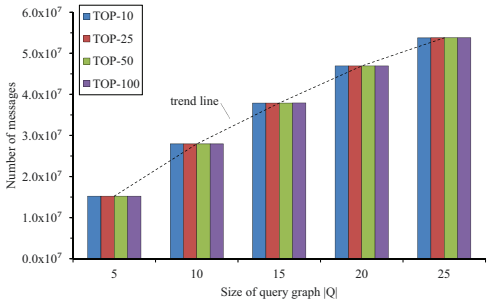
(a) Linkedin-Runtime



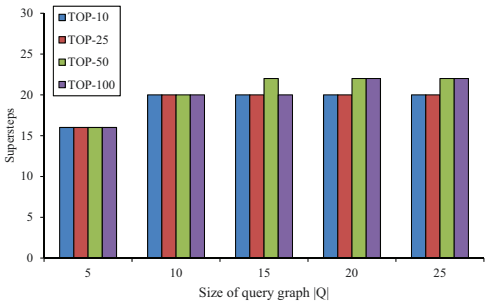
(b) Custom-Runtime



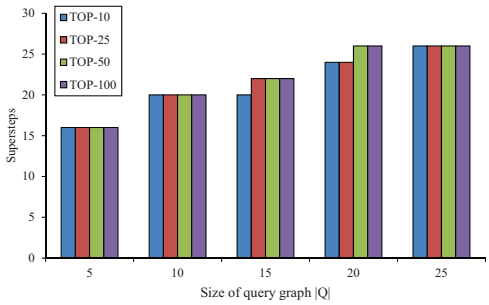
(c) Linkedin-Messages



(d) Custom-Messages



(e) Linkedin-Supersteps



(f) Custom-Supersteps

Fig. 1: The runtime, messages, and supersteps vs. the sizes of query graph $|Q|$

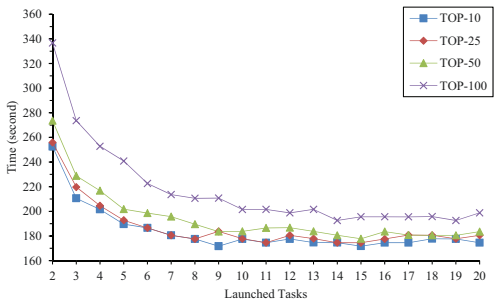
can be deployed in distributed systems and speeded up the subgraph matching query.

III. EXPERIMENTS

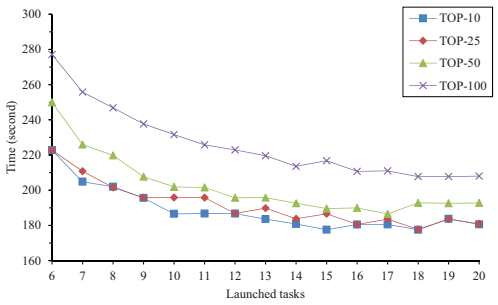
The experiments are conducted on a distributed system, which consists of six nodes. Each node has 128GB RAM, 16 CPUs. Hadoop and HAMA [7] are deployed on the platform. One real world dataset (Named Linkedin in the following) is used in the experiments [8]. The Linkedin data graph is with 2,985,414 vertices and 25,965,384 edges. The other dataset (Named Custom in the following) is generated by Hama with 10,000,000 vertices.

The size of query graph $|Q|$ is an important parameter in the real graph query. Firstly, we evaluate the running time, number of messages, and supersteps with respect to various sizes of query Q, and different k values. In the evaluations, the size of query graph $|Q|$ varies from $5, 10, 15, 20,$ to 25. Fig. 1 shows the results of running time, number of messages, and number of supersteps. The runtime increases as the sizes of $|Q|$ for both datasets. With the same dataset, larger top-k needs more runtime. For number of messages, the sizes of $|Q|$ affect the results greatly. Larger sizes of Q cause more messages because of more comparisons in data graph. An interesting observation is that k value of top-k almost has the same number of messages for the same size of $|Q|$. At last, the supersteps are related to the sizes of $|Q|$. But the supersteps will not increase any more after Q reaches a certain value.

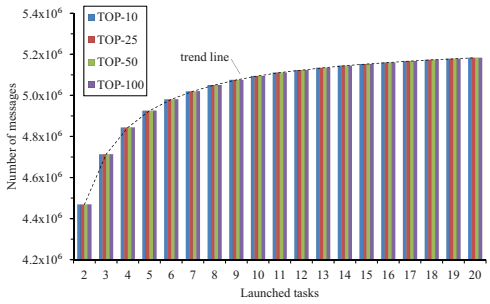
We also evaluate the scalability by setting various tasks on a real distributed computation platform. Fig. 2 shows the results of the scalability by changing launched tasks. For both datasets, the overall trend of runtime is declining as the increase of the task number which can be seen in Fig.



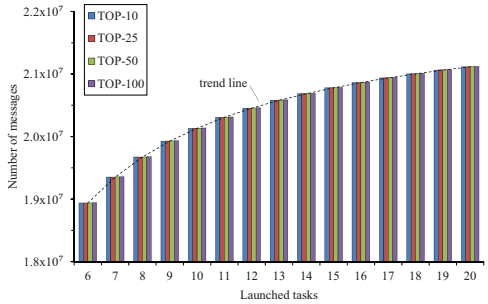
(a) Linkedin-Runtime



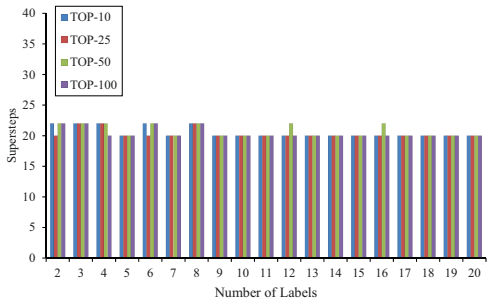
(b) Custom-Runtime



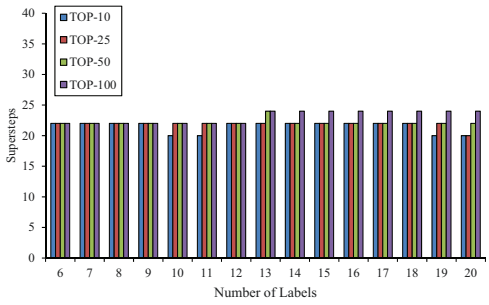
(c) Linkedin-Messages



(d) Custom-Messages



(e) Linkedin-Supersteps



(f) Custom-Supersteps

Fig. 2: The scalability evaluation by changing launched tasks

2(a),(b). When the number of tasks reach a certain degree, the runtime can not be reduced because of the increase cost of communication between nodes as can be seen in Fig. 2(c),(d). The superstep has little relationship with the number of tasks as can be seen in Fig. 2(e),(f). It verifies the concurrent operations at vertex level in a distributed system.

IV. Conclusion

We propose a distributed subgraph matching query algorithm at the level of single vertex. Each vertex can obtain its matching score in an independent manner without requiring global graph information. Therefore, the algorithm can be run concurrently and it is suitable for big graphs due to the strong scalability. The experiment results with real-life and custom datasets show that the proposed algorithm is very efficient with respect to runtime and scalability.

References

[1] S. Hu, L. Zou, J. Yu, H. Wang, D. Zhao. "Answering natural language questions by subgraph matching over knowledge graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 5, pp. 824-37, 2018.

[2] L. Zou, L. Chen, and Y. Lu. "Top-k subgraph matching query in a large graph," *In Proc. of the ACM first Ph. D. workshop in CIKM*, pp. 139-146. ACM, 2007.

[3] M. S. Amin, R. L. Finley, and H. M. Jamil, "Top-k Similar Graph Matching Using TraM in Biological Networks," *IEEE/ACM Trans. on Computational Biology and Bioinformatics*, vol. 9, no. 6, pp. 1790-1804, 2012.

[4] Z. Yang, A. W. Fu, and R. Liu, "Diversified Top-k subgraph searching in a Large Graph," *in Proc. SIGMOD*, pp. 1167-1182, 2016.

[5] W. Fan, X. Wang, Y. Wu, "Diversified Top-k Graph Pattern Matching," *in VLDB*, vol. 6, no. 13, pp. 1510-1521, 2013.

[6] J. Gao, B. Song, P. Liu, W. Ke, J. Wang, X. Hu, "Parallel top-k subgraph query in massive graphs: computing from the perspective of single vertex," *IEEE International Conference on Big Data*, pp. 636 - 645, 2016.

[7] http://hama.apache.org/

[8] https://www.aminer.cn/