Indexing Shortest Paths Between Vertices and Vertex Groups in Weighted Knowledge Graphs — Supplemental Materials

This supplement is available online [*]. The road map of this supplement is as follows.

- In Section S1, we discuss the feasibility of extending the proposed techniques to directed graphs.
- In Section S2, we conduct additional experiments with different thread pool size and d.

S1. EXTENDING THE PROPOSED TECHNIQUES TO DIRECTED GRAPHS

S2. ADDITIONAL EXPERIMENT RESULTS

In the main experiments, we set the size of thread pool to 80 (the computer has 96 threads in total), and set d to 20. Here, we conduct additional experiments where these two parameters are set to different values.

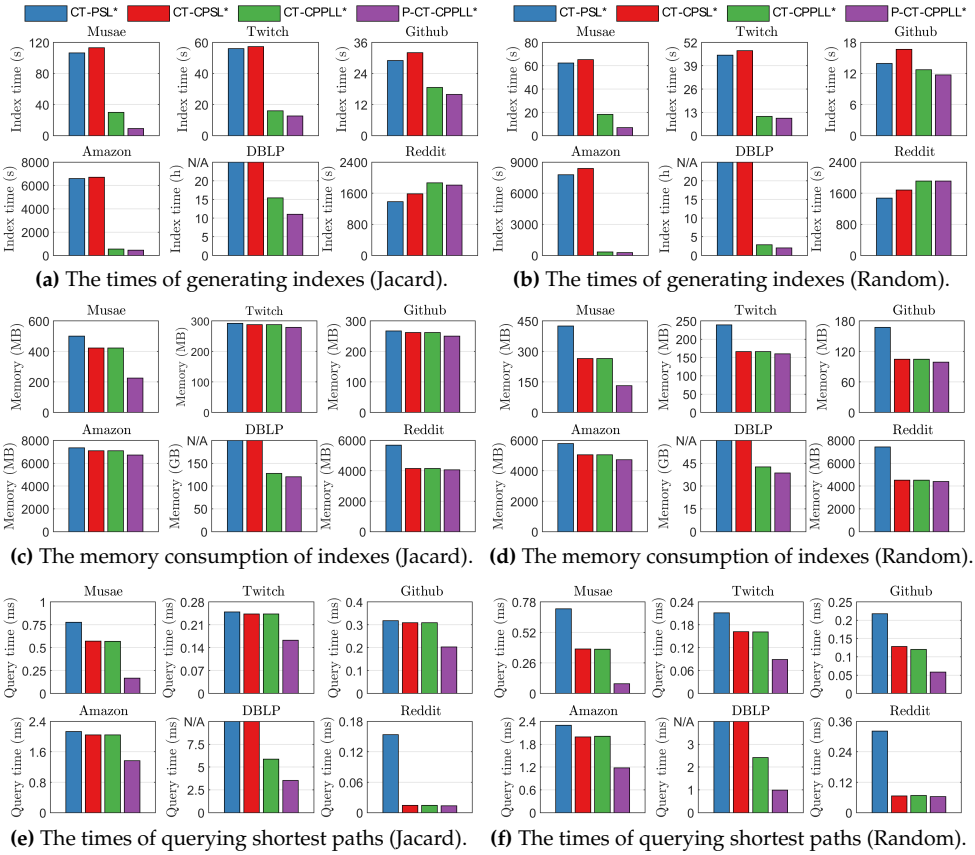
First, we decrease the size of thread pool to 50 (while keeping d to 20), and present the additional experiment results in Figure S1. We observe that, in comparison with the main experiment results, the times of generating indexes are slightly larger in Figure S1. For example, CT-CPSL* takes less than 60s to index the Musae graph with random edge weights in the main experiments, while CT-CPSL* takes more than 60s to do so in Figure S1.

Second, we increase d to 50 (while keeping the size of thread pool to 80), and present the additional experiment results in Figure S2. We observe that, in comparison with the main experiment results, the times of generating indexes are slightly larger in Figure S2. For example, the four algorithms take less than 18s to index the sf Github graph with random edge weights in the main experiments, while these algorithms take more than 30s to do so in Figure S2. Moreover, in comparison with the main experiment results, the times of querying shortest paths are significantly larger in Figure S2. For example, the four algorithms take less than 2.4ms to query shortest paths in the sf Amazon graph with Jacard edge weights in the main experiments, while these algorithms take more than 5ms to do so in Figure S2. The reason is that a larger d induces a large part of tree indexes in Core-Tree indexes, and it is generally slow to query shortest distances or paths using tree indexes [1].

Nevertheless, the key experiment observations in the main experiments also hold in the above additional experiments. First, CT-CPPLL* and P-CT-CPPLL* can be an order of magnitude faster than the extended state-of-the-art algorithm CT-PSL* and its canonical improvement CT-CPSL* for generating indexes. Second, the canonical repair process could removes a significantly amount of redundant non-canonical 2-hop labels, and induces that, when comparing with CT-PSL*, the indexes generated by CT-CPSL* and CT-CPPLL* consume a smaller amount of memory, and the query speeds using the indexes generated by CT-CPSL* and CT-CPPLL* are also higher. Third, when comparing with CT-CPPLL*, the partial indexing approach helps P-CT-CPPLL* to further save memory consumption spaces and increase the index and query speeds.

REFERENCES FOR THE SUPPLEMENT

1. W. Li, M. Qiao, L. Qin, Y. Zhang, L. Chang, and X. Lin, "Scaling up distance labeling on graphs with core-periphery properties," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, (2020), pp. 1367–1381.

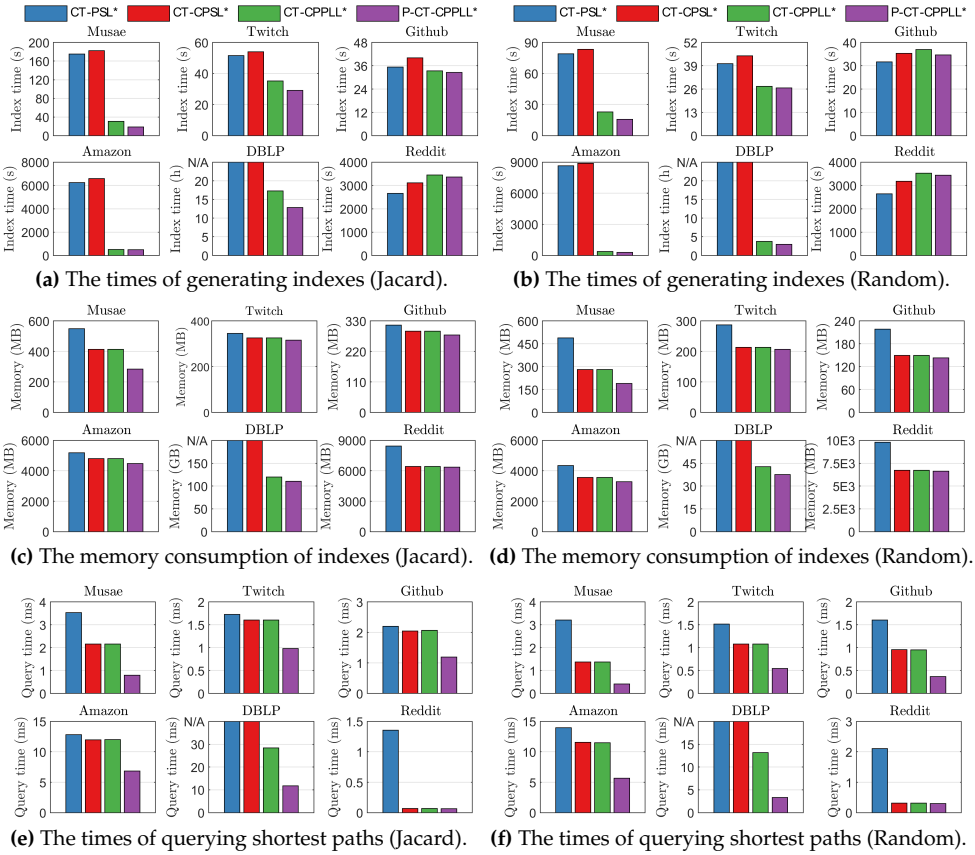


(a) The times of generating indexes (Jacard). **(b)** The times of generating indexes (Random).

(c) The memory consumption of indexes (Jacard). **(d)** The memory consumption of indexes (Random).

(e) The times of querying shortest paths (Jacard). **(f)** The times of querying shortest paths (Random).

Fig. S1. Additional experiment results of 50 threads.



(a) The times of generating indexes (Jacard). **(b)** The times of generating indexes (Random).

(c) The memory consumption of indexes (Jacard). **(d)** The memory consumption of indexes (Random).

(e) The times of querying shortest paths (Jacard). **(f)** The times of querying shortest paths (Random).

Fig. S2. Additional experiment results of $d = 50$.