HOW SHOULD WE EVALUATE DATA DELETION IN GRAPH-BASED ANN INDEXES?

Supplementary Material

A. MATHEMATICAL REPRESENTATION OF EVALUATION METRICS

The 1-Recall@k, which represents search accuracy, is defined as follows. Let n_q be the number of queries, and for a given query $\mathbf{q}_i \in \mathbb{R}^d$, let $g_i \in \{1, 2, \ldots, n\}$ denote the ground-truth nearest neighbor. Additionally, let $\hat{\mathcal{R}}_i \subset \{1, 2, \ldots, n\}$ with $|\hat{\mathcal{R}}_i| = k$ represent the approximate k-nearest neighbors obtained through ANNS. Defining $f(\cdot)$ as a function that returns 1 if the condition is true and 0 otherwise, 1-Recall@k is expressed as shown in Equation 1.

$$\text{1-Recall@}k = \frac{1}{n_q} \sum_{i=1}^{n_q} f\big(g_i \in \hat{\mathcal{R}}_i\big), \tag{1}$$

In this study, we set $k = 10$ and use 1-Recall@10 to evaluate search accuracy. A higher recall indicates better search accuracy.

To evaluate the query processing speed, we use Queries Per Second (QPS). When processing n_q queries in t seconds, QPS is defined as shown in Equation 2:

$$\text{QPS} = \frac{n_q}{t} [1/\text{s}] \tag{2}$$

A higher QPS value indicates faster query processing. We measure QPS-search to evaluate search speed, QPS-add to evaluate data insertion speed, and QPS-delete to evaluate data deletion speed. Additionally, we adopt the QPS-Recall curve as an evaluation metric, where the horizontal axis represents 1-Recall@10, and the vertical axis represents QPS-search. This curve is obtained by varying the search parameters of HNSW. A curve positioned toward the upper right of the graph indicates higher search performance.

B. ALL EXPERIMENTAL RESULTS

The experimental results for SIFT1M [1] are shown in Fig. A, and those for GIST1M [1] are presented in Fig. B. The results for SIFT2M [2] with a batch size of $b = 10^5$ are shown in Fig. C, while those with $b = 10^3$ are given in Fig. D. In the following sections, we discuss the experimental results for each evaluation metric.

B.1. QPS-Recall

From Fig. A1, Fig. B1, Fig. C1, and Fig. D1, it is evident that rebuilding maintains search performance even after repeated insertions and deletions. In contrast, logical deletion significantly degrades search performance across all datasets as insertions and deletions are repeated. The plotted points in each graph indicate that both search accuracy and search speed deteriorate in this case. Furthermore, the search performance of physical deletion is slightly lower than that of rebuilding across all datasets.

B.2. 1-Recall@10

From Fig. A2, Fig. B2, Fig. C2, and Fig. D2, it can be observed that search accuracy in logical deletion decreases as insertions and deletions are repeated. Additionally, the accuracy of rebuilding is the highest, followed by physical deletion, which exhibits lower accuracy than rebuilding. Furthermore, Fig. C2 and Fig. D2 show that in physical deletion, search accuracy stabilizes after a certain number of insertion and deletion steps. This indicates that the structural properties of the graph become stable after a sufficient number of operations. Moreover, a larger batch size results in a higher converged accuracy. This suggests that when insertions and deletions are performed repeatedly, a larger batch size facilitates better recovery of the graph structure during the insertion process.

B.3. Memory Usage

From Fig. A3, Fig. B3, Fig. C3, and Fig. D3, it is evident that memory consumption in logical deletion increases linearly with each step across all datasets. This indicates that in logical deletion, the deleted data remains in memory. In contrast, memory usage remains unchanged for both rebuilding and physical deletion. This confirms that these methods effectively reclaim memory space when data is deleted.

B.4. QPS-add

From Fig. A4, Fig. B4, and Fig. C4, it can be observed that when data is inserted and deleted in batches of $b = 10^5$, the data insertion speed remains unchanged. However, as shown in Fig. D4, when the batch size is reduced to $b = 10^3$, the data insertion speed in logical deletion exhibits significant variations at each step.

Additionally, when data insertion and deletion are performed in batches of $b = 10^5$, physical deletion exhibits the highest data insertion speed. This is likely because repeated physical deletions gradually make the graph sparser, thereby reducing the number of distance calculations required during data insertion.

B.5. QPS-delete

From Fig. A5, Fig. B5, Fig. C5, and Fig. D5, it is evident that across all datasets, logical deletion achieves the highest data deletion speed, on the order of approximately $10^9[1/\mathrm{s}]$. In contrast, both rebuilding and physical deletion operate at a significantly lower speed, at most on the order of $10^3[1/\mathrm{s}]$. When data insertion and deletion are performed in batches of $b = 10^5$, logical deletion can be completed in approximately $10^{-4}[\mathrm{s}]$, whereas physical deletion requires up to $10^2[\mathrm{s}]$.

From Fig. A5 and Fig. B5, it can be observed that the dimensionality of the inserted and deleted vectors affects only the speed of rebuilding. SIFT1M [1] has a dimensionality of 128, whereas GIST1M [1] has a dimensionality of 960. This difference impacts rebuilding because it requires distance calculations during deletion. As the vector dimensionality increases, the time needed for a single-distance calculation also increases, leading to slower deletion speeds.

From Fig. C5 and Fig. D5, it can be observed that the data deletion speed of physical deletion remains almost unchanged regardless of the batch size b. This indicates that physical deletion primarily involves memory operations for the specified deletion queries, leading to a consistent processing speed. Specifically, when the batch size is reduced from $b = 10^5$ to $b = 10^3$, meaning the number of deletions per step is reduced to $1/100$, the deletion speed of physical deletion remains nearly constant. In contrast, the speed of rebuilding decreases by approximately a factor of 100. This is because when the dataset size is relatively small, the processing time required for rebuilding remains almost constant, regardless of the number of deleted data points.

B.6. QPS-search

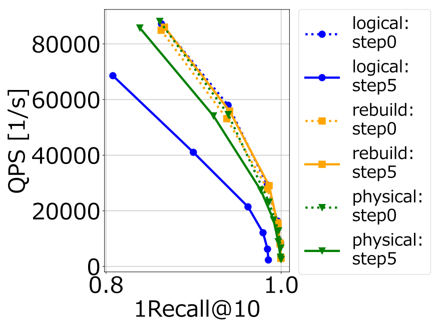
From Fig. A5, Fig. B5, Fig. C5, and Fig. D5, it is evident that across all datasets, search speed is highest when using physical deletion. As discussed in Sec B.4, this is likely because physical deletion gradually makes the graph sparser, reducing the number of distance calculations required during the search.

Similarly, across all datasets, logical deletion results in the slowest search speed. This is likely because, in logical deletion, an additional operation is required after the standard search process: the retrieved results must be filtered by referencing a flag array to exclude deleted data.

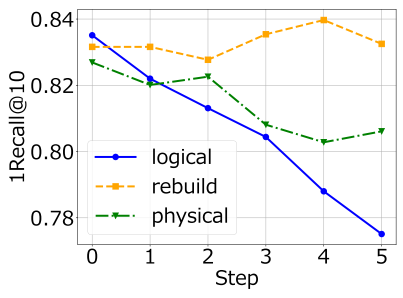
C. REFERENCES

[1] Hervé Jégou, Matthijs Douze, and Cordelia Schmid, “Product quantization for nearest neighbor search,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2010.

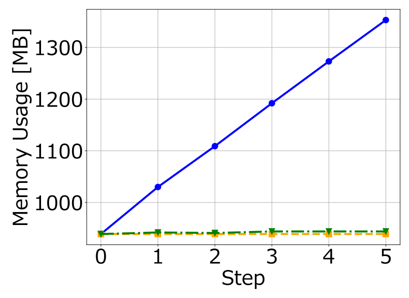
[2] Hervé Jégou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg, “Searching in one billion vectors: re-rank with source coding,” in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 861–864.



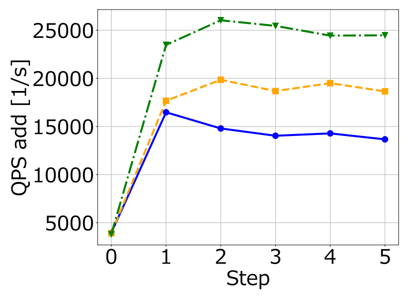
(1) QPS-Recall



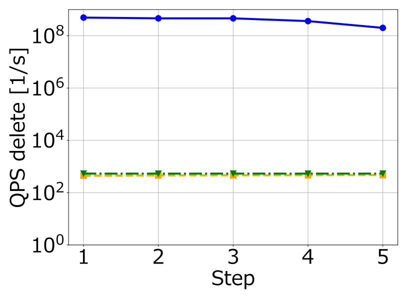
(2) 1-Recall@10



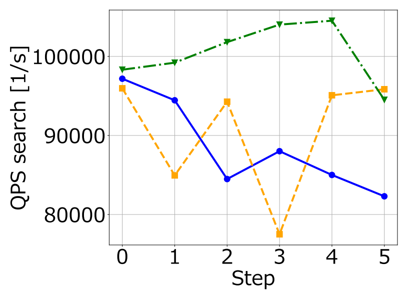
(3) Memory Usage



(4) QPS-add

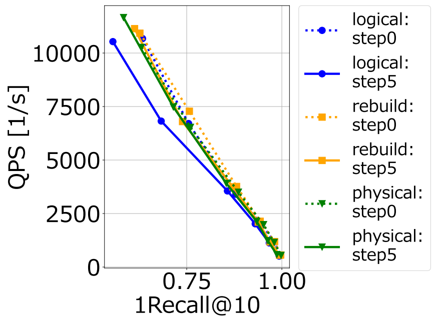


(5) QPS-delete

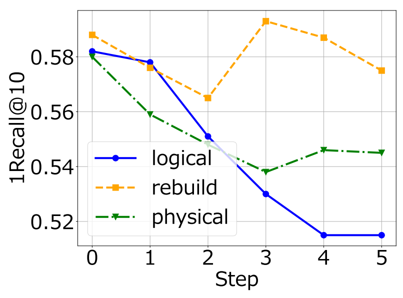


(6) QPS-search

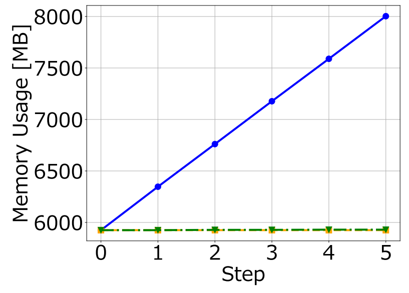
Fig. A: Performance comparison of the three deletion methods at each step on SIFT1M.



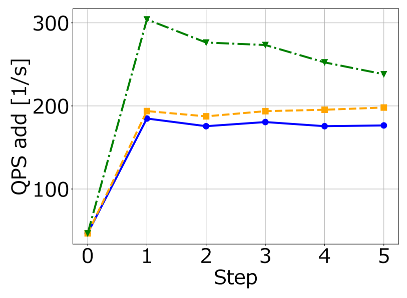
(1) QPS-Recall



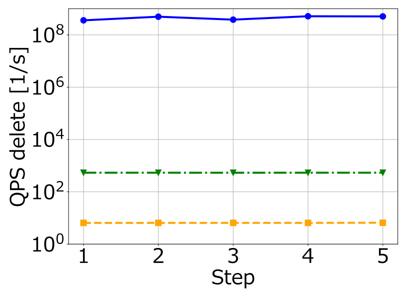
(2) 1-Recall@10



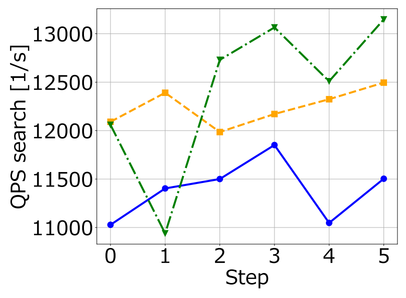
(3) Memory Usage



(4) QPS-add

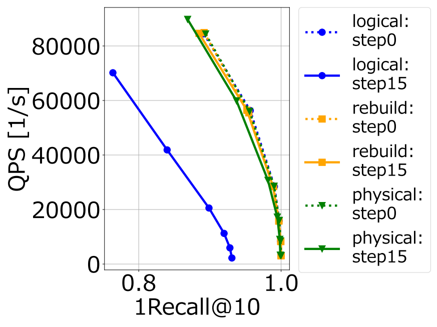


(5) QPS-delete

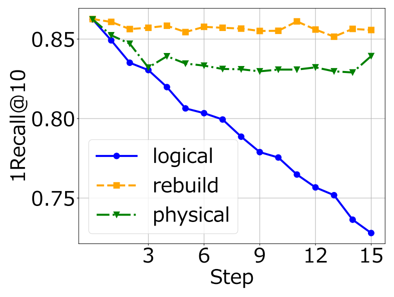


(6) QPS-search

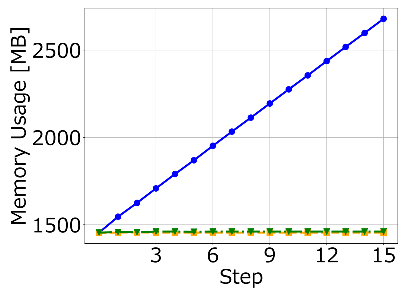
Fig. B: Performance comparison of the three deletion methods at each step on GIST1M.



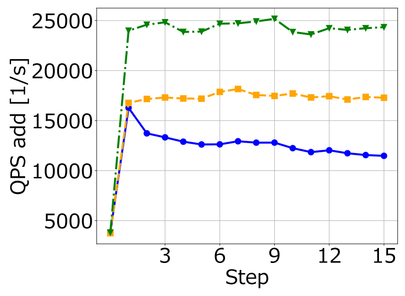
(1) QPS-Recall



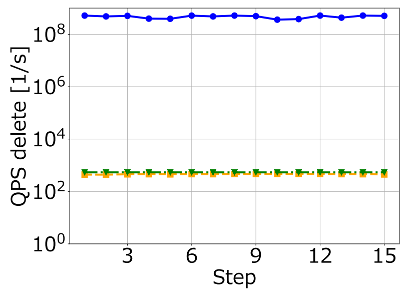
(2) 1-Recall@10



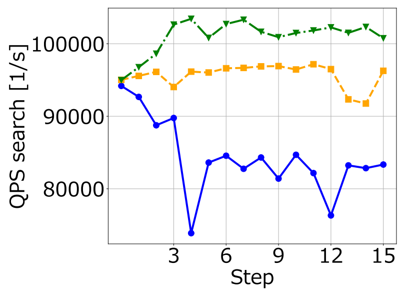
(3) Memory Usage



(4) QPS-add

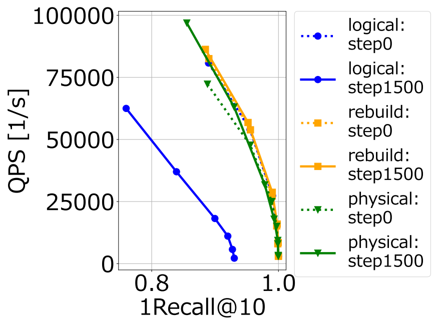


(5) QPS-delete

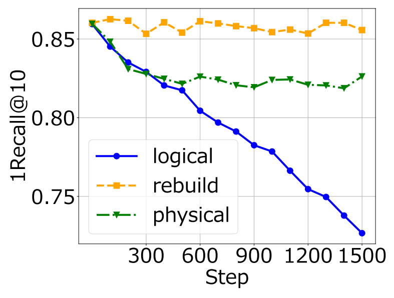


(6) QPS-search

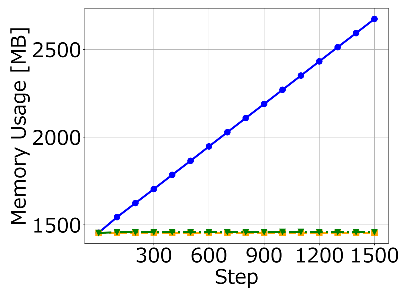
Fig. C: Performance comparison of the three deletion methods at each step on SIFT2M with $b = 10^5$.



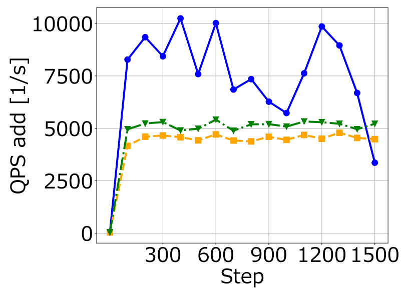
(1) QPS-Recall



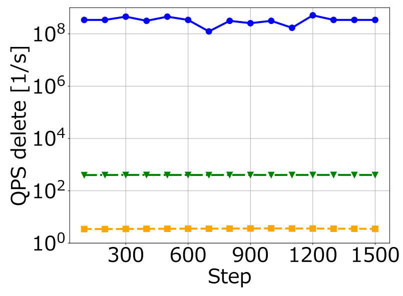
(2) 1-Recall@10



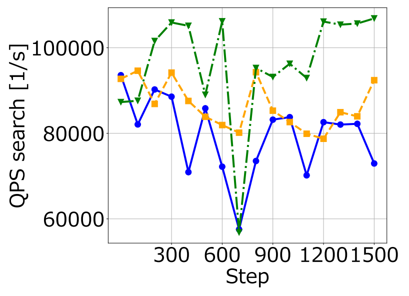
(3) Memory Usage



(4) QPS-add



(5) QPS-delete



(6) QPS-search

Fig. D: Performance comparison of the three deletion methods at each step on SIFT2M with $b = 10^3$.