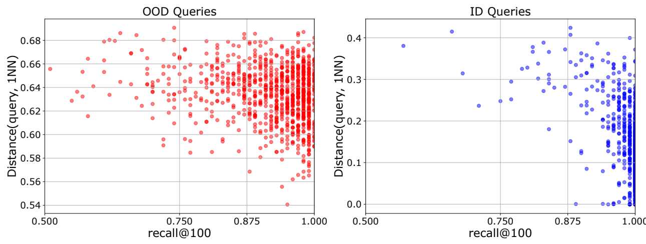


Figure 1: The relationship between recall@100 and the distance from the query to its nearest neighbors in LAION10M. We use HNSW (M = 32, efC = 2000) as the graph index during the search. For OOD queries, the search list size (i.e., efS) is fixed at 600, while for ID queries, efS is fixed at 300.

A DISCUSSION ON THE HARD QUERIES

Recall that in Section 1, we analyzed that the theoretical guarantees of RNG and its variants only hold when the query is close to the base data. The experimental results in Figure 1 support this observation. We use $\delta(q, 1NN)$ to denote the distance between a query and its top-1 nearest neighbor. From the results, we can observe the following: (1) Queries with low accuracy typically have large $\delta(q, 1NN)$ values. This is because the theoretical guarantees of RNG fail to hold when the $\delta(q, 1NN)$ is large. Since these hard queries are far from the base data, it becomes difficult to optimize the graph structure solely from the perspective of the base data. (2) However, a large $\delta(q, 1NN)$ does not necessarily imply low accuracy. This indicates that when building the graph based on the query distribution, we need to distinguish between easy and hard queries, and place greater emphasis on optimizing for the hard ones. (3) Although ID queries share the same vector type with base data (e.g., images), some of them are still located far from the base data in the vector space.

Using historical queries to build the graph helps the index better capture the query distribution. This improves the accuracy of future queries from the same distribution. Another important reason is that, in real-world production environments, hard queries encountered now are likely to reappear in similar forms in the future. On our e-commerce platform, we use the inner product to measure similarity. We define two queries as similar if their distance (1 - inner product) is less than 0.05. Then, for the hard queries within a certain time period, 87% of them will be encountered again in the future with similar queries. Therefore, dynamically fixing the defects in the graph can significantly increase the accuracy of these queries when they are encountered again in the future.

B PROOF

B.1 The Proof of THEOREM 1

THEOREM 1. *For a directed graph index $G = (V, E)$ and a query q. If $N_{i,q}$ can reach $N_{j,q}$ in $NG_{S,q}$ $(1 \le i, j \le S)$. The Algorithm 1 will always visit $N_{j,q}$ when query $= q$, $ep = N_{i,q}$ and $L \ge S$.*

PROOF. *Since $N_{i,q}$ and $N_{j,q}$ are reachable in $NG_{S,q}$, this implies there exists a path $p = \{v_1, v_2, v_3, ..., v_t\}$ such that $v_1 = N_{i,q}, v_t = N_{j,q}$ and $\forall 1 \le x \le t, \delta(v_x, q) \le \delta(N_{S,q}, q)$. Suppose Algorithm 1 has visited v_x but did not add v_{x+1} to the candidate set $\boldsymbol{C}$. According to Algorithm 1, the condition for not adding v_{x+1} to $\boldsymbol{C}$ is that the result set $\boldsymbol{R}$ is full and $\delta(v_{x+1}, q) > max_{x \in \boldsymbol{R}} \delta(x, q)$. However, since $L \ge S$, this means $max_{t \in \boldsymbol{R}} \delta(t, q) \ge \delta(N_{S,q}, q)$, but $\delta(v_{x+1}, q) < \delta(N_{S,q}, q)$, which leads to a contradiction. Since $ep = v_1$, v_1 has already been visited, so the subsequent points $v_2, ..., v_t$ will also be visited.*

B.2 The Proof of THEOREM 2

THEOREM 2. *Given a base data set X and the DG constructed from the points in X. If any edge of DG is removed, there exists a query q such that $NG_{2,q}$ contains only two isolated nodes.*

PROOF. *For each edge in DG, there is a circle passing through the two vertices of the edge but not through any other points. If the edge is removed, we set the query q to the center of this circle, then the $NG_{2,q}$ contains only two isolated points.*

B.3 The Proof of COROLLARY 1

COROLLARY 1. *For a directed graph index $G = (V, E)$ and a query q. the Algorithm 1 will always visit $N_{j,q}$ when query $= q$, $ep = N_{i,q}$ and $L \ge EH(N_{i,q}, N_{j,q}, q, G)$.*

PROOF. *Recall that EH is defined as:*

$$EH(u, v, q, G) = \min_{p \in P(u,v,G)} \max_{x \in p} F_{x,q}$$

where $P(u, v, G)$ denotes the set of all paths from vertex u to vertex v in graph G and $F_{u,q}$ denote the point u is the $F_{u,q}$-th NN of q. This means that there exists a path from u to v, in which the farthest node from the query in the path is the $EH(u, v, q, G)$-th nearest neighbor of the query. Therefore, $N_{i,q}$ can reach $N_{j,q}$ in $NG_{EH(N_{i,q}, N_{j,q}, q, G), q}$. According to Theorem 1, Algorithm 1 will always visit $N_{j,q}$ when query $= q$, $ep = N_{i,q}$ and $L \ge EH(N_{i,q}, N_{j,q}, q, G)$.

B.4 The Proof of THEOREM 3

THEOREM 3. *If points $N_{i,q}$ and $N_{j,q}$ satisfy the condition that $N_{i,q}$ cannot reach $N_{j,q}$ in $NG_{S-1,q}$ but can reach $N_{j,q}$ in $NG_{S,q}$, then $EH(N_{i,q}, N_{j,q}, q, G) = S$ and $\mathbb{H}_{i,j} = S$.*

PROOF. *In $NG_{S-1,q}$, the inability of N_i to reach N_j indicates that $\mathbb{H}_{i,j} > S - 1$. In $NG_{S,q}$, the fact that N_i can reach N_j indicates that $\mathbb{H}_{i,j} \le S$. Therefore, $\mathbb{H}_{i,j} = S$.*

B.5 The Proof of THEOREM 4

THEOREM 4. *Given a query q, Algorithm 3 will add at most $2(N_q - 1)$ extra directed edges.*

PROOF. *Consider the worst-case for the algorithm, where $\forall 1 \le i, j \le N_q, \mathbb{T}_{ij} = 0$ and no pair of points has the same distance. Suppose the algorithm connects an edge (u, v) in the first step, and (v, u) has not yet been connected. Since $\delta(u, v) = \delta(v, u)$, and currently $\mathbb{T}vu = 0$, the second step will certainly connect the edge (v, u). The algorithm will continue in the same manner as steps 1 and 2, essentially connecting undirected edges in ascending order of $\delta(u, v)$. So the algorithm is analogous to Kruskal's algorithm for solving the minimum spanning tree (MST) with edge weight $\delta(u, v)$, and the MST of N_q points will have $N_q - 1$ undirected edges. So the number of directed edges connected in the worst case will be $2(N_q - 1)$.*

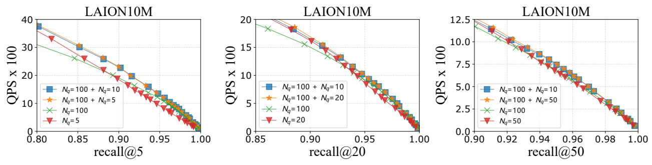


Figure 2: Evaluation of N_q under different k.

B.6 The Proof of THEOREM 5

THEOREM 5. *Let T denote the set of historical queries. Without pruning the edges, our method ensures that when $q \in T$ and $k \le N_q$, the accuracy of GreedySearch(G, q, k, centroid, K_h) is 100% (Algorithm 1).*

PROOF. *Executing RF with parameter N_q guarantees that the Greedy Search can reach at least one of the top-N_q nearest neighbors of the query (i.e., can reach some nodes in $NG_{N_q,q}$).*

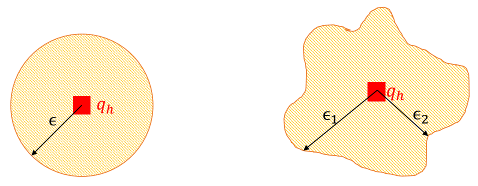
Executing NGFix with parameters N_q and K_h, we can guarantee that starting from any node in $NG_{N_q,q}$, the greedy search can reach all other nodes in $NG_{N_q,q}$ when the search list size $L \ge K_h$.

So, combining RF and NGFix can guarantee the accuracy of GreedySearch(G, q, k, centroid, K_h) is 100% when $q \in T$.

C DISCUSSION ON PARAMETER N_q

According to the strategy described in Section 6.6, we recommend first performing NGFix* (i.e., NGFix + RF) with N_q=100, and then performing NGFix* again with N_q=k based on the number of NNs k to be retrieved. In this section, we mainly focus on explaining the reasons for employing this strategy. Recall that in Theorem 5, we showed that NGFix* is theoretically supported when the search list size $L \ge K_h$. (Since $K_h \ge N_q$, we need $L \ge N_q$.) When aiming to retrieve the top-k NNs, we need to set $L \ge k$ to ensure that the number of retrieved ANNs is no less than k. Therefore, setting N_q=k can better provide theoretical support for the graph index. Furthermore, when N_q is relatively small (e.g., 10), the limited number of NNs considered for each historical query results in insufficient exploitation of the information provided by these queries in the graph index. To address this, we perform an additional NGFix* with N_q=100 to ensure the graph index makes better use of the historical queries.

We evaluated the performance of the graph index by changing N_q across different values of k. The results presented in Figure 2 indicate that performing NGFix* with N_q=k yields slightly better performance. Moreover, in scenarios where k varies frequently, it is not necessary to perform NGFix* for all values of k. Performing NGFix* twice with N_q=10 and N_q=100 is sufficient to produce a high-quality index. The experimental results also provide another insight, which validates the effectiveness of our theorem: Taking k=5 as an example, when the target recall is relatively low (e.g., recall@5=0.8), the graph index performance with N_q=5 is significantly better than with N_q=100, because at this point $L \le 100$ and setting N_q=100 does not provide theoretical support. However, when the target recall is high (e.g., recall@5 = 0.95), the graph index with N_q=100 performs better. This is because at this point $L \ge 100$, giving the graph index some theoretical backing. Meanwhile, setting N_q=5 fails to fully leverage the information from historical queries and thus performs worse than N_q=100 at high recall targets. Performing NGFix twice for each historical query combines the above advantages and avoids their respective drawbacks, resulting in a higher-quality graph index.



(a) $\delta(q_t, q_h) < \epsilon$ **(b)** $\mathcal{N}_{q_t,k} \subseteq \mathcal{N}_{q_h,K}$

Figure 3: The yellow region indicates the set of q_t that satisfy the corresponding condition: (a) $\delta(q_t, q_h) < \epsilon$. (b) $\mathcal{N}_{q_t,k} \subseteq \mathcal{N}_{q_h,K}$.

D $\delta(q_t, q_h) < \epsilon$ VS. $\mathcal{N}_{q_t,k} \subseteq \mathcal{N}_{q_h,K}$

In Section 7, we presented two directions for theoretical extension. In this section, we use experiments to illustrate the relationship between them. Our experiments are primarily conducted on the WebVid2.5M dataset. For simplicity, we use Euclidean distance in this experiment [1]. Figure 3 illustrates the region of q_t that satisfies the corresponding conditions. The region of q_t that satisfies the condition (Figure 3b) is irregular, so we use the following method to compute $\delta(q_t, q_h)$ under this condition: We sample 10k historical queries q_h. For each q_h, we randomly select 10^5 directions and move q_t from q_h along each direction to the boundary of the region satisfying the condition. We then compute $\delta(q_t, q_h)$ at that point (e.g., ϵ_1, ϵ_2 in Figure 3b). The final $\delta(q_t, q_h)$ is obtained by averaging over all directions for each query, and then averaging across all queries. In the WebVid dataset, experimental results show that when the condition $\mathcal{N}_{q_t,10} = \mathcal{N}_{q_h,10}$ is satisfied, the value of $\delta(q_t, q_h)$ is approximately less than 0.03. Moreover, when $\mathcal{N}_{q_t,10} \subseteq \mathcal{N}_{q_h,20}$, the value of $\delta(q_t, q_h)$ is approximately less than 0.114.

[1]Although WebVid adopts cosine similarity, Euclidean distance is equivalent in this case since all vectors in WebVid are normalized.