Patience in Proximity: A Simple Early Termination Strategy for HNSW Graph Traversal in Approximate k-Nearest Neighbor Search

Tommaso Teofili[1][0000-0002-4372-0273]* and Jimmy Lin[2][0009-0007-1571-3854]

[1] Elastic
tommaso.teofili@elastic.co
[2] David R. Cheriton School of Computer Science
University of Waterloo
jimmylin@uwaterloo.ca

Abstract. The Hierarchical Navigable Small World (HNSW) graph is widely recognized for its state-of-the-art performance in approximate k-NN search, leveraging a multi-layer proximity graph to efficiently navigate high-dimensional spaces. However HNSW graph traversal can become computationally expensive, especially for large-scale datasets. To address this challenge, we introduce a strategy to early terminate HNSW graph traversal, dubbed "Patience in Proximity". Inspired by techniques from clustering-informed approximate nearest neighbor algorithms, our approach employs a saturation-based threshold to dynamically halt graph exploration, reducing computational overhead without significantly compromising accuracy.
We evaluate the proposed method on diverse datasets from the BEIR benchmark.

Keywords: Dense Retrieval · Approximate Nearest Neighbor · HNSW.

1 Introduction

Approximate k-Nearest Neighbor (k-NN) search has become a cornerstone in various applications, from recommendation systems and natural language processing to computer vision, due to its ability to efficiently search for nearest neighbors with high-dimensional data. The Hierarchical Navigable Small World (HNSW) graph, a state-of-the-art approach for approximate k-NN search, achieves remarkable performance by building and traversing a proximity graph that efficiently prunes the search space. However, even with such optimizations, the computational cost of traversing the HNSW graph can be prohibitive for extremely large datasets or resource-constrained environments, where every millisecond counts. On a different perspective, speeding up HNSW graph search is important for Dense Retrieval [12] and Retrieval Augmented Generation [8].

* Corresponding author

Inspired by the application of the concept of *patience* in visiting clusters for clustering informed ANN algorithms [1], this work proposes a simple, yet effective early termination strategy tailored for HNSW graph traversal. Dubbed "Patience in Proximity", our approach introduces a saturation-based threshold for dynamically halting the exploration of nearest neighbor candidates at specific layers of the HNSW graph. By adapting the principle of patience from the realm of dense retrieval to the navigable structure of HNSW, we aim to significantly reduce the computational overhead without compromising the accuracy of the k-NN search results.

In our work we try to answer the following research questions:

- Can the application of a saturation-based early termination strategy effectively reduce the computational cost of HNSW graph traversal?
- To what extent does one such approach impact the accuracy of approximate k-NN search compared to exhaustive HNSW traversal?
- How does the proposed strategy perform across diverse datasets, in terms of both efficiency and effectiveness?

Contributions This paper contributes a lightweight, easy-to-implement early termination strategy for HNSW [3], empirical evaluations across multiple benchmarks, and insights into the trade-offs between search accuracy and computational efficiency. Our work is poised to benefit applications requiring fast and accurate k-NN search, particularly in scenarios where computational resources are limited.

2 Background

The problem of efficient Approximate k-Nearest Neighbor (k-NN) search has gathered significant attention over the years due to its broad applicability across various domains. Numerous methods have been proposed to enhance the efficiency of k-NN search, particularly for large-scale, high-dimensional datasets, where exact search methods become computationally infeasible.

Nearest neighbor search is the problem of finding the vectors in a given set that are closest to a given query vector. As such sets of vectors become bigger and bigger, and the vector dimensionality grows up until thousands of dimensions, approximate nearest neighbor methods aim to balance search effectiveness with computational efficiency. More formally, let $X = \{x_1, ..., x_n\} \in R^D$ be a set of n vectors in a D-dimensional space and $q \in R^D$ represents a query vector. Given a value $k \leq n$, ANN search finds the k closest vectors in X to q, according to a pairwise distance function $d(q, x)$, such that: $N = \text{k-argmin}_{x \in X}\, d(q, x)$. The resulting set N contains the k closest vectors to the query vector q.

The Hierarchical Navigable Small World (HNSW) graph introduced by Malkov and Yashunin [5] has emerged as one of the most effective methods for approximate k-NN search due to its ability to efficiently prune the search space using a

[3] https://github.com/apache/lucene/pull/14094

multi-layer proximity graph structure. By organizing nodes in layers, with each successive layer containing progressively more nodes, HNSW enables rapid navigation from coarser to finer-grained searches. Search begins at the top-most layer, which has the fewest nodes, and proceeds layer by layer, with each layer increasing in density. As the search descends to lower layers, the candidate set of neighbors is refined, with nearest neighbors being progressively identified through a best-first traversal mechanism. Formally, let the HNSW graph $G = (V, E)$ where V is the set of nodes, each representing a data point (or vector) in the space, and $E \subseteq V \times V$ represents directed edges between nodes, indicating proximity in the vector space. Each data point $v \in V$ exists in one or more layer $l = 0, \ldots, L$ where L is the highest layer of the graph.

In recent years, a variety of techniques have been explored to further enhance the efficiency of HNSW. These strategies can be broadly categorized into two types: improvements in graph construction and optimizations in search heuristics. Several works have focused on optimizing the graph-building process, aiming to reduce memory consumption and improve the connectivity of graph layers [2, 3, 11]. These methods have yielded important advancements, but graph traversal during the search phase remains the most computationally expensive step. A simple early-exit strategy makes use of a distance-based cutoff, which terminates the search when the current nearest neighbor's distance reaches a certain threshold, and further exploration is unlikely to yield better results [4, 6]. This approach reduces the computational load but may sacrifice accuracy if the cutoff threshold is set too aggressively. Another known strategy is the early stopping based on the number of visits per layer [4]. In this method, the traversal is terminated if a predefined number of candidate nodes have been evaluated at each layer, under the assumption that additional visits are unlikely to significantly alter the outcome. This method is effective in limiting the search time, but it can sometimes miss distant but relevant neighbors, particularly in non-uniform data distributions.

A recent paper by Busolin et al. [1] introduces a strategy that applies a patience threshold during search to stop exploration once improvements in candidate quality saturates. This approach was designed for nearest neighbor algorithms that rely on clustering information, such that it avoids visiting some clusters whose centroids are still possibly close to a query vector. While this cannot be applied as is to improve HNSW, its underlying principle can be adapted to the graph traversal problem in HNSW, where search efficiency is critical.

Our work extends this line of research by proposing a saturation-based early termination strategy, "Patience in Proximity", tailored specifically for HNSW graph traversal. Unlike distance-based cutoffs or fixed visit thresholds, our approach dynamically adjusts the termination point based on the diminishing returns observed in candidate quality during the traversal. This way we aim to minimize unnecessary computations while maintaining the high accuracy that HNSW is known for.

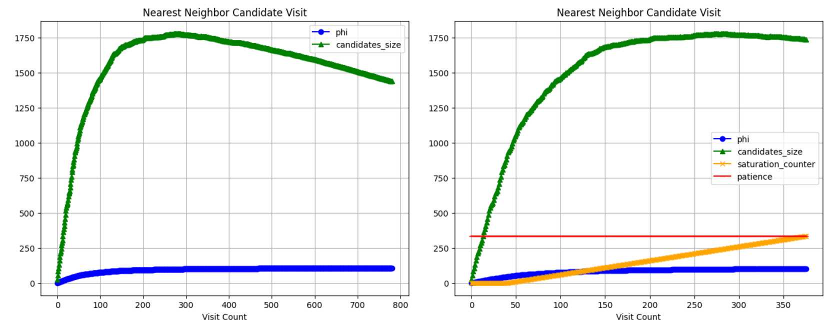
3 Methodology

In standard HNSW traversal, the search process continues until no closer neighbors are found in the bottom-most layer, or a predetermined candidate pool size is reached. While effective, exhaustive search often entails redundant node visits, particularly when the nearest neighbors are already identified.

"Patience in Proximity" strategy dynamically halts HNSW graph traversal once the candidate quality saturates. Instead of imposing a strict cutoff based on visits or distance, this method tracks improvement rates in selected nearest neighbors. Once improvements fall below a saturation threshold, the search is terminated.

Let v be a point in the HNSW graph (at a given layer l) and C a set of candidate neighbors to visit (the nodes linked to v in G at layer l), the HNSW search algorithm visits all $c \in C$ to select k nearest neighbors for a given query vector q, based on a distance function d. During the visit of all $c \in C$, new nearest neighbors are eventually identified and added to the set of nearest neighbors $N(q)$.

Let $N_{h,l}(q)$ be the set of nearest neighbors collected after visiting h candidates at layer l and $\phi_{h,l}(q)$ be the size of the intersection between $N_{h,l}(q)$ and the previous nearest neighbors at iteration $N_{h-1,l}(q)$ divided by k: $\phi_{h,l}(q) = 100 \cdot |N_{h-1,l}(q) \cap N_{h,l}(q)|/k$.



(a) HNSW (b) HNSW$_P$

Fig. 1. Number of candidates visited with plain HNSW (a) and HNSW with "patience" (b) on a sample query vector from HotpotQA.

Figure 1(a) plots $\phi_{h,0}(q)$, for a sample query vector from the HotpotQA dataset. After around 300 visits, the value of ϕ saturates reaching its plateau. This observation, allows HNSW to make an adaptive decision on when to stop visiting further candidates, because actual collection of additional nearest neighbors becomes extremely rare.

While such a saturation might happen also during earlier stages, the early termination only kicks in when the set of collected neighbors doesn't change much for a number Δ (our *patience*) of *consecutive* iterations, e.g., when $\phi_h \geq \gamma \in [90, 100]$ for Δ consecutive iterations. In summary, we stop visiting candidate neighbors for a query vector q if visiting the next candidate keeps at least γ percent of the k collected neighbors unchanged, for Δ consecutive iterations. Note also that the number of candidates to be visited (green line in Figure 1(a)) rapidly increases as the new candidates to be evaluated are way more than the candidates that get *promoted* to nearest neighbors, in the beginning. Again nearly around 300 visits, where very few candidates start being turned into nearest neighbors, the number of candidates to be evaluated starts decreasing, suggesting that HNSW is running out of good candidate options for the given query vector, confirming the saturation phenomenon.

Figure 1(b) plots $\phi_{h,0}(q)$ as well as the *patience* threshold Δ and a *saturation counter* to account for the number of times $\phi_{h,0}(q) \geq \gamma$, with $\gamma = 95$. The patience threshold is met after about 350 iterations and the visit ends, resulting in about 450 less candidate visits with respect to the plain HNSW execution (about 800 visits in Figure 1(a)).

Since HNSW leverage the nearest neighbor search *primitive* also when building the graph, our early termination strategy can be either used at search time only or during both indexing and search. The latter potentially leading to the construction of a more compact HNSW graph.

4 Experiments

We evaluate approximate k-nearest neighbor retrieval using standard evaluation methodology on datasets from the BEIR benchmarks [7]: we consider Robust04, HotpotQA, DBPedia, Climate-FEVER and TREC-COVID datasets. BGE [9] is selected as a representative dense retrieval model for generating the embeddings, specifically *bge-base-en-v1.5*.

In order to assess the effectiveness / efficiency tradeoffs of our early-termination strategy; we report NDCG@10, Recall@100 and Recall@1000 and query-per-second (QPS). We consider the HNSW implementation of the Apache Lucene library (9.11.1 release) and employ the Anserini toolkit [10] to run our experiments in a reproducible way on an Intel(R) Xeon(R) CPU@2.80GHz with 68GiB System memory. Our experiments report results with our early-termination strategy at search time only ($\mathbf{HNSW}_{P(S)}$) and at index-and-search time ($\mathbf{HNSW}_{P(IS)}$).

Table 1 shows a notable improvement in QPS across all datasets, both when early termination is applied only during search and when applied at both indexing and search stages.

With $\text{HNSW}_{P(S)}$, for Robust04 and TREC-COVID we observe up to 20% QPS speedup, for HotpotQA and Climate-FEVER we observe up to 41-43% QPS speedups while we report the best efficiency improvement for DBPedia with a 64% QPS speedup. $\text{HNSW}_{P(IS)}$ QPS is mostly on par with $\text{HNSW}_{P(S)}$ for Robust04, DBPedia and Trec-COVID, whereas its efficiency improvements are more limited for HotpotQA and Climate-FEVER datasets.

Dataset	**Method**	**NDCG@10**	**R@100**	**R@1000**	**QPS**
Robust04	**HNSW**	0.4460	0.3487	0.5938	73.7
	HNSW$_{P(S)}$	0.4463	0.3481	0.5922	87.1
	HNSW$_{P(IS)}$	0.4444	0.3466	0.5588	89.1
HotpotQA	**HNSW**	0.7069	0.8487	0.9153	70.6
	HNSW$_{P(S)}$	0.7018	0.8422	0.9085	100.1
	HNSW$_{P(IS)}$	0.6936	0.8301	0.8958	95.1
DBPedia	**HNSW**	0.4046	0.5234	0.7654	47.9
	HNSW$_{P(S)}$	0.4069	0.5264	0.7684	77.1
	HNSW$_{P(IS)}$	0.3985	0.5129	0.7518	78.5
Climate-FEVER	**HNSW**	0.3108	0.6354	0.8282	73.4
	HNSW$_{P(S)}$	0.3112	0.6361	0.8304	102.5
	HNSW$_{P(IS)}$	0.3123	0.6364	0.8287	84.3
TREC-COVID	**HNSW**	0.7814	0.1406	0.4769	62.4
	HNSW$_{P(S)}$	0.7814	0.1407	0.4787	73.9
	HNSW$_{P(IS)}$	0.7807	0.1405	0.4787	73.2

Table 1. Comparison of Methods Across Multiple BEIR Datasets

On the other hand, effectiveness remains almost untouched when using HNSW$_{P(S)}$, whereas we notice drops by about 1 point for all of NDCG@10, Recall@100 and Recall@1000 for Robust04, HotpotQA and DBPedia when using HNSW$_{P(IS)}$. This last observation seems in line with the fact that HNSW$_{P(IS)}$ builds an HNSW graph with fewer edges. For the Climate-FEVER dataset, we observe an unexpected improvement both in efficiency and effectiveness, although very limited, with our early-termination strategy. We also notice a small improvement in efficiency and effectiveness for HNSW$_{P(S)}$ on the DBPedia dataset. We plan to investigate further in the future on the rationale behind these unexpected (although beneficial) effectiveness improvements.

We also conducted a small experiment for accounting the number of candidates visits with and without our early-termination strategy. We found out that using early-termination leads to 39% to 42% less candidate visits for HotpotQA and Robust04, respectively.

5 Conclusions

In this work, we introduced "Patience in Proximity", a novel saturation-based early termination strategy for optimizing Hierarchical Navigable Small World graph traversal in approximate k-Nearest Neighbor search. By dynamically terminating search once candidate quality improvements plateau, our method effectively reduces computational costs while maintaining accuracy. Our experiments across multiple datasets demonstrate that the patience-based approach provides efficiency improvements over BEIR benchmark datasets. The adaptive nature of our early-termination strategy enables it to cater to varying dataset densities, making it suitable for a wide range of real-world k-NN scenarios. Future work involves evaluation with more datasets; we may also explore extensions of this strategy to inspect the candidate quality saturation patterns more broadly, including layer-specific experiments. We also plan to investigate some unexpected effectiveness improvements observed in our experimental results.

References

1. Busolin, F., Lucchese, C., Nardini, F.M., Orlando, S., Perego, R., Trani, S.: Early exit strategies for approximate k-nn search in dense retrieval. In: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management. p. 3647–3652. CIKM '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3627673.3679903, https://doi.org/10.1145/3627673.3679903
2. Coleman, B., Segarra, S., Smola, A.J., Shrivastava, A.: Graph reordering for cache-efficient near neighbor search. Advances in Neural Information Processing Systems **35**, 38488–38500 (2022)
3. Fu, C., Xiang, C., Wang, C., Cai, D.: Fast approximate nearest neighbor search with the navigating spreading-out graph. arXiv preprint arXiv:1707.00143 (2017)
4. Lin, P.C., Zhao, W.L.: Graph based nearest neighbor search: Promises and failures. arXiv preprint arXiv:1904.02077 (2019)
5. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans. Pattern Anal. Mach. Intell. **42**(4), 824–836 (2020). https://doi.org/10.1109/TPAMI.2018.2889473, https://doi.org/10.1109/TPAMI.2018.2889473
6. Ren, J., Zhang, M., Li, D.: Hm-ann: Efficient billion-point nearest neighbor search on heterogeneous memory. Advances in Neural Information Processing Systems **33**, 10672–10684 (2020)
7. Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., Gurevych, I.: BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2) (2021), https://openreview.net/forum?id=wCu6T5xFjeJ
8. Wang, Z.J., Chau, D.H.: Mememo: On-device retrieval augmentation for private and personalized text generation. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 2765–2770 (2024)
9. Xiao, S., Liu, Z., Zhang, P., Muennighoff, N., Lian, D., Nie, J.Y.: C-Pack: Packed resources for general chinese embeddings. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 641–649. SIGIR '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3626772.3657878, https://doi.org/10.1145/3626772.3657878
10. Yang, P., Fang, H., Lin, J.: Anserini: Reproducible ranking baselines using Lucene. Journal of Data and Information Quality **10**(4), Article 16 (2018)
11. Yang, S., Xie, J., Liu, Y., Yu, J.X., Gao, X., Wang, Q., Peng, Y., Cui, J.: Revisiting the index construction of proximity graph-based approximate nearest neighbor search. arXiv preprint arXiv:2410.01231 (2024)

12. Zhao, W.X., Liu, J., Ren, R., Wen, J.R.: Dense text retrieval based on pretrained language models: A survey. ACM Transactions on Information Systems **42**(4), 1–60 (2024)