A cluster-based iterative search approach over HNSW

Marco Antônio Vieira[1], **Anderson R. Tavares**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

Abstract. *Approximate Nearest Neighbor Search (ANNS) algorithms use various techniques to perform an approximate similarity search on vector databases, exchanging the optimality of exhaustive search for drastic time savings. The HNSW algorithm for ANNS uses a hierarchical Proximity Graph (PG) to reduce the number of hops during search. However, HNSW cannot produce diverse results because it uses a single entry point. In this work, we propose iHNSW, a novel cluster-based approach to enable iterative search of the HNSW graph. Empirical experiments on SIFT1M, GIST, MNIST, and Fashion-MNIST vector datasets show that our approach consistently leads to an increase in recall metrics at the cost of extending search time.*

1. Introduction

Finding the k most similar elements to a query q in a vectorial space induced by a dataset D is a fundamental problem in information retrieval, and can be defined as the k-Nearest Neighbor (k-NN) search problem. Approximate Nearest Neighbor Search (ANNS) algorithms use different strategies to efficiently explore only a subset of elements $D' \subset D$, reducing solution complexity, such that it trims the number of distance calculations. Proximity Graphs (PGs) connect data points in D, creating a connected, navigable structure. Searching begins in an entry point and proceeds greedily by selecting the most promising neighbor until reaching the (approximate) k-NNs of a query with a good performance on time and recall [Malkov et al. 2014, Malkov and Yashunin 2018, Jayaram Subramanya et al. 2019].

The state-of-the-art PG-based Hierarchical Navigable Small World (HNSW) algorithm [Malkov and Yashunin 2018] constructs a hierarchical graph during indexing, where the number of nodes decreases exponentially from a layer to its superior. Search begins in a single point at the top layer, descending until the (approximate) k-NNs of the query. Search time is argued to be logarithmic. However, the algorithm has a fixed entry point, excluding the possibility of restarting a search to find a different result. To tackle this drawback, we present iHNSW, an iterative search approach that uses data clusters to bring diversity to the search results. While searching, iHNSW ignores data belonging to clusters from previously-found k-NNs. Empirical experiments on SIFT1M, GIST, MNIST, and Fashion-MNIST vector datasets show that our approach presents an interesting recall vs time tradeoff, consistently improving recall metrics at the expense of search time.

2. Related Work

2.1. ANNS on Proximity Graphs

Graphs that meet the efficiency requirements of ANNS have a very high indexing complexity (at least $O(n^2)$). ANNS algorithms typically use approximations of this type of

ideal graph, making their construction feasible for real applications while maintaining the same exploration strategy through greedy search [Fu et al. 2019]. Small World Networks, for example, are graphs that can be explored through a variation of greedy search with logarithmic (or polylogarithmic) time and serve as an approximation of the Delaunay graph [Malkov et al. 2014]. This type of graph is used as a reference for the Navigable Small World (NSW) [Malkov et al. 2014] and HNSW [Malkov and Yashunin 2018] ANNS algorithms. The NSG (Navigating Spreading-out Graph) [Fu et al. 2019] and Vamana [Jayaram Subramanya et al. 2019] algorithms are also state-of-the-art ANNS algorithms that use approximations of ideal ANNS graphs.

2.2. Clustering for ANNS

Clustering and vector quantization algorithms are widely used in ANNS in different scenarios. Inverted index-based (IVF) algorithms [Jegou et al. 2010] use quantization to divide the dataset into k' clusters, to reduce computation by searching linearly only the closest w clusters. In systems that support billions of data sets, it is common to use clustering to distribute the data across different machines to parallelize the process [Jayaram Subramanya et al. 2019, Chen et al. 2021]. Algorithms such as NSG and Vamana also use a clustering algorithm to calculate the centroid of all the data and define the entry point as the approximate medoid [Fu et al. 2019, Jayaram Subramanya et al. 2019]. The HCNNG (Hierarchical-Clustering-based Nearest Neighbor Graph) algorithm [Munoz et al. 2019] uses hierarchical clustering to partition the dataset and uses the elements contained in each cluster to construct a sparse graph, a process that is repeated multiple times with overlapping clusters.

3. Proposed Approach

To overcome the infeasibility of search restarts in HNSW graphs and foster diverse results, we propose a simple iterative heuristic named iterative HNSW (iHNSW). In this approach (Figure 1), a clustering algorithm divides the dataset into k' clusters, and we adopt a search heuristic that prunes elements belonging to the same clusters as the current k-NNs found.

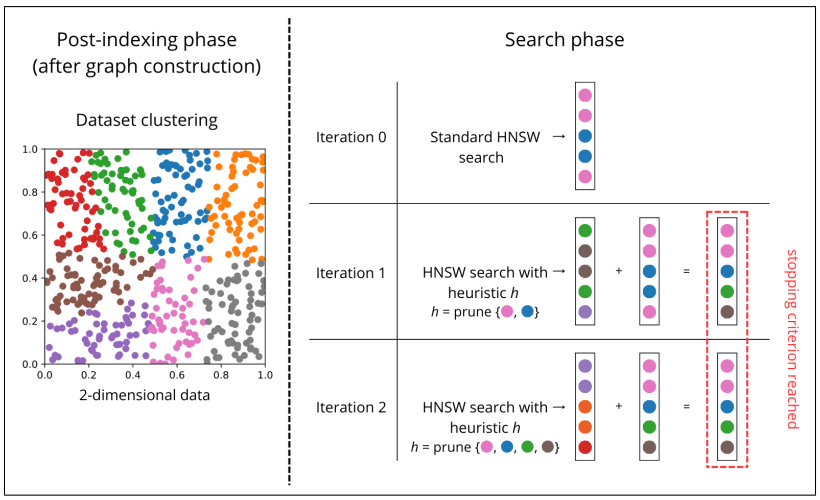

Figure 1. Example of the iHNSW searching pipeline using a toy dataset. First, clustering the dataset (left), then searching iteratively through the HNSW graph until reaching the convergence criteria.

On the first iteration, a standard HNSW search is performed, returning a list of k-NNs. On the following iterations, the pruning heuristic is used in addition to the greedy strategy: points belonging to the clusters of the previously-found k-NNs are avoided. Thus, exploring greedily the remaining nodes could lead to different results, even with the same entry point. The new k-NNs are merged with the previous ones. The stopping criterion for this approach uses the convergence of the k-NNs found over the last two iterations. Additional stopping criteria, such as a maximum number of iterations or a timeout, can be defined, allowing an anytime behavior of the algorithm.

This change aims to increase the diversity of graph exploration and enable results refinement and recall adjustment without requiring graph reconstruction. In fact, since each iteration merges the resulting k-NNs with the previous set of k-NNs, it becomes impossible for iHNSW to achieve a lower recall than single-search HNSW. On the other hand, since iHNSW performs at least two search iterations, it is reasonable to expect that it will not be less time-consuming than HNSW.

4. Experiments and Results

Our implementation of algorithms (our and competitors') and testing routines are available online for reproducibility[1]. We performed experiments on four well-known image-sourced vector datasets: SIFT1M, GIST, MNIST and Fashion-MNIST, from [Aumüller et al. 2018]. We averaged 5 runs over different $seed$ values for random number generation in our tests. Also, iHNSW uses the same graph built for HNSW, in addition to a clustering built by K-Means. Table 1 presents the average query time, distance calculations, and recall@100 of the algorithms over all queries on the test set of the evaluation datasets. The difference between all metrics is statistically significant (p-value < 0.01).

Table 1. Average performance of algorithms over the tested datasets. Bold indicates the best values for each dataset. Arrows indicate whether lower or higher is better.

Dataset	Algorithm	Query time (s) ↓	Distance calculations ↓	Recall@100 ($\pm SD$) ↑
SIFT1M	IVF	0.0604	9,544.81	0.7733±0.169
	HNSW	**0.0334**	**1,908.93**	0.8938±0.116
	iHNSW	0.0984	4,755.65	**0.9077±0.103**
GIST	IVF	0.0989	11,167.17	0.4850±0.227
	HNSW	**0.0448**	**2,485.93**	0.7160±0.163
	iHNSW	0.1585	7,341.11	**0.7453±0.146**
MNIST	IVF	0.0487	1,573.81	0.7966±0.141
	HNSW	**0.0158**	**972.65**	0.9890±0.038
	iHNSW	0.0483	2,384.62	**0.9950±0.014**
Fashion-MNIST	IVF	0.0496	1,611.91	0.8447±0.122
	HNSW	**0.0138**	**857.98**	0.9931±0.020
	iHNSW	0.0383	1,934.56	**0.9952±0.013**

[1] available at: `https://github.com/marcoantonioaav/iHCNSW`

We observed that our iHNSW achieves better recall than IVF, even eventually managing to cost less time (MNIST and Fashion-MNIST) or make fewer distance calculations (SIFT1M and GIST). Also, the iterative version (iHNSW) achieved a higher recall than the single-search counterpart (HNSW), at the cost of a higher average query time.

5. Conclusion and Future Work

We propose a cluster-based iterative search approach over HNSW (iHNSW), which enables the possibility of improving the search results with search restarts. It diversifies an otherwise fixed HNSW search, as it enforces exploration of different regions of the graph by discarding data on clusters belonging to the current list of nearest neighbors. Also, iHNSW does not change the indexing process of HNSW, so it can be applied to already constructed HNSW graphs. Our experiments show that iHNSW converges even with no time limit, consistently improving recall, albeit spending more time searching. This could be mitigated with time-based stopping criteria. Future work could analyze the impact of changing the clustering algorithm and parameters, as well as HNSW hyperparameters.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

References

Aumüller, M., Bernhardsson, E., and Faithfull, A. (2018). ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms. https://arxiv.org/abs/1807.05614.

Chen, Q., Zhao, B., Wang, H., Li, M., Liu, C., Li, Z., Yang, M., and Wang, J. (2021). Spann: Highly-efficient billion-scale approximate nearest neighborhood search. *NeurIPS*, 34:5199–5212.

Fu, C., Xiang, C., Wang, C., and Cai, D. (2019). Fast approximate nearest neighbor search with the navigating spreading-out graph. *Proceedings of the VLDB Endowment*, 12(5):461–474.

Jayaram Subramanya, S., Devvrit, F., Simhadri, H. V., Krishnawamy, R., and Kadekodi, R. (2019). DiskANN: Fast accurate billion-point nearest neighbor search on a single node. *NeurIPS*, 32.

Jegou, H., Douze, M., and Schmid, C. (2010). Product quantization for nearest neighbor search. *IEEE TPAMI*, 33(1):117–128.

Malkov, Y., Ponomarenko, A., Logvinov, A., and Krylov, V. (2014). Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68.

Malkov, Y. A. and Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE TPAMI*, 42(4):824–836.

Munoz, J. V., Gonçalves, M. A., Dias, Z., and Torres, R. d. S. (2019). Hierarchical clustering-based graphs for large scale approximate nearest neighbor search. *Pattern Recognition*, 96:106970.