Estudio y Análisis del Algoritmo Hierarchical Navigable Small (HNSW) para Búsqueda Aproximada de Vecinos Más Cercanos

1st Gianfranco Gonzalo Cordero Aguirre
Faculty of Computing
UTEC
Lima, Perú
gianfranco.cordero@utec.edu.pe

2nd Federico Iribar Casanova
Faculty of Computing
UTEC
Lima, Perú
federico.iribar@utec.edu.pe

3rd Jose Eduardo Huamani Ñaupas
Faculty of Computing
UTEC
Lima, Perú
jose.huamani@utec.edu.pe

4th David Alexander Salaverry Cuzcano
Faculty of Computing
UTEC
Lima, Perú
david.salaverry@utec.edu.pe

***Abstract*—Este trabajo presenta un análisis exhaustivo del algoritmo Hierarchical Navigable Small World (HNSW), una estructura gráfica jerárquica destinada a la búsqueda aproximada eficiente de vecinos más cercanos en grandes conjuntos de datos multidimensionales. Se abordan el funcionamiento interno del algoritmo, sus límites teóricos, su comparación con otras estructuras similares, su implementación práctica y aplicaciones, buscando ofrecer un panorama claro y completo para aplicaciones en aprendizaje automático y sistemas de recomendación.**

***Index Terms*—búsqueda aproximada, vecinos más cercanos, HNSW, aprendizaje automático, estructuras de datos, grafos navegables**

I. Introducción

La búsqueda de vecinos más cercanos Nearest Neighbor Search, (NNS) es una operación fundamental en muchas aplicaciones donde el aprendizaje automático, como la clasificación, recomendación y recuperación de información. Los algoritmos tradicionales, como los árboles k-dimensionales (k-d trees), sufren de una disminución de eficiencia en espacios de alta dimensión debido a la maldición de la dimensionalidad. En este contexto, el algoritmo Hierarchical Navigable Small World (HNSW) ha surgido como una solución eficiente para la búsqueda aproximada de vecinos en grandes volúmenes de datos.

HNSW se basa en una estructura de grafo jerárquico, donde cada capa del grafo tiene un subconjunto de los datos. Las capas superiores contienen menos nodos con conexiones de largo alcance, mientras que las capas inferiores detallan las conexiones locales. Este enfoque jerárquico permite realizar búsquedas rápidas, comenzando en los niveles más altos y refinando los resultados conforme se desciende.

A diferencia de otros métodos, HNSW ofrece una combinación única de precisión y eficiencia, que lo convierte en una opción ideal para aplicaciones de alto rendimiento en aprendizaje automático, tales como sistemas de recomendación y análisis de imágenes. Este trabajo proporciona un análisis exhaustivo de HNSW, sus fundamentos, su implementación, y su comparación con otros algoritmos de búsqueda aproximada de vecinos, con el objetivo de destacar su efectividad en la resolución de problemas en espacios de alta dimensión.

II. Marco Teórico

En el campo de la búsqueda de vecinos más cercanos, existen diversas técnicas y algoritmos, cada uno con sus ventajas y limitaciones. Las principales categorías de estos algoritmos incluyen los métodos exactos, como los árboles k-dimensionales (k-d trees), y los algoritmos de búsqueda aproximada, que son más eficientes en espacios de alta dimensión.

Los árboles k-dimensionales (k-d trees) son una estructura de datos basada en la partición recursiva del espacio en hiperesferas o celdas. Este método funciona bien en espacios de baja dimensión, pero su eficiencia disminuye drásticamente cuando se trata de datos de alta dimensión debido a la maldición de la dimensionalidad. En situaciones de alta dimensión, la mayoría de las distancias se vuelven similares, lo que provoca que la búsqueda sea ineficiente.

En este contexto, los algoritmos de búsqueda aproximada de vecinos más cercanos, como HNSW, han demostrado ser eficaces para superar estas limitaciones. Estos algoritmos utilizan estructuras de datos más complejas, como grafos de mundos pequeños navegables (Navigable Small World, NSW), para permitir búsquedas rápidas y precisas en espacios de alta dimensión.

HNSW se basa en la combinación de grafos NSW y listas de salto probabilísticas, lo que permite realizar búsquedas rápidas mediante un enfoque jerárquico y de navegación codiciosa. Este enfoque ha demostrado ser particularmente

eficaz en tareas como sistemas de recomendación, clasificación de imágenes y procesamiento de lenguaje natural.

III. Fundamentos del Algoritmo HNSW

El algoritmo HNSW combina dos conceptos clave: los grafos de mundos pequeños navegables (Navigable Small World, NSW) y las listas de salto probabilísticas (probabilistic skip lists). Los grafos NSW permiten una búsqueda eficiente mediante una estructura de grafo donde cada nodo está conectado a sus vecinos más cercanos. Las listas de salto probabilísticas facilitan la inserción y búsqueda rápidas mediante una estructura de lista enlazada con múltiples niveles.

HNSW extiende los grafos NSW al introducir una jerarquía de capas, donde cada capa representa un subconjunto de los datos. La capa superior contiene menos nodos con conexiones de largo alcance, mientras que la capa inferior incluye todos los nodos con conexiones locales detalladas. Esta estructura jerárquica permite realizar búsquedas rápidas y precisas mediante un enfoque de búsqueda codiciosa (greedy search) que desciende desde la capa superior hacia la inferior.

IV. Construcción y Búsqueda en HNSW

La construcción del índice HNSW se realiza de manera incremental. Al insertar un nuevo elemento, se determina su nivel en la jerarquía y se conecta a sus vecinos más cercanos en cada capa utilizando una heurística que prioriza la diversidad para mejorar la navegabilidad y evitar redundancias. Este enfoque permite optimizar la conectividad en las capas superiores sin perder la precisión en las capas inferiores.

La búsqueda de vecinos más cercanos en HNSW comienza en un nodo de entrada en la capa superior y desciende iterativamente por la jerarquía, siguiendo conexiones que acercan la consulta a los vecinos más próximos. Este método ofrece una gran ventaja sobre los enfoques tradicionales, especialmente en términos de escalabilidad y tiempos de ejecución cuando se manejan grandes cantidades de datos.

Otras estrategias de construcción: Aunque nuestro índice se construye de forma incremental en CPU, vale la pena mencionar que investigaciones recientes han explorado *construcciones por lotes* para acelerar la fase de indexado. Un ejemplo destacado es **CAGRA** [4], que reformula la inserción de HNSW como un proceso paralelo en bloques: (1) genera un grafo k-NN inicial en lote, (2) añade aristas inversas para asegurar simetría, y (3) aplica un *pruning* en dos fases para mantener a lo sumo M vecinos por nodo. El algoritmo logra hasta $30\times$ mayor velocidad de construcción en datasets de varios millones de vectores sin degradar el *recall*. Si bien nuestra implementación no utiliza GPU, CAGRA demuestra que los principios de HNSW pueden escalar aún más cuando se dispone de hardware masivamente paralelo, lo cual abre líneas futuras de optimización.

Variantes de búsqueda. La búsqueda greedy original puede acelerarse o afinarse con técnicas recientes. Por ejemplo, *CAGRA* reutiliza el mismo recorrido codicioso pero lo ejecuta íntegramente en GPU mediante dos kernels: *single-CTA* (consultas por lotes) y *multi-CTA* (query única), alcanzando hasta 77× más QPS o 53× menos latencia que HNSW-CPU a 95 % de *recall* [4]. En el plano algorítmico, *Distance-Adaptive Beam Search* [2] reemplaza el tamaño fijo del haz por un criterio de parada basado en la distancia al mejor candidato, obteniendo mejor *recall* con menos evaluaciones de distancia en varios datasets. Estas alternativas ilustran cómo la fase de búsqueda de HNSW puede seguir optimizándose ya sea con hardware paralelo o con nuevas reglas de terminación.

Ejemplo paso a paso: La Fig. 1 ilustra la construcción y la búsqueda para un conjunto de **6 documentos** bidimensionales con parámetro $M = 2$. Las coordenadas se asignaron de forma que la distancia euclídea refleje la similitud semántica entre textos (Tabla I).

TABLE I
Mini-dataset empleado en la Fig. 1.

ID	Texto del documento	Coord. (x, y)	Nivel L
1	El gato se sentó.	(1,1)	1
2	Los perros ladran fuerte.	(5,1)	1
3	Gatos y perros.	(3,2)	1
4	Los pájaros vuelan alto.	(6,6)	0
5	Los peces nadan.	(2,5)	0
6	El zorro marrón rápido.	(4,4)	0

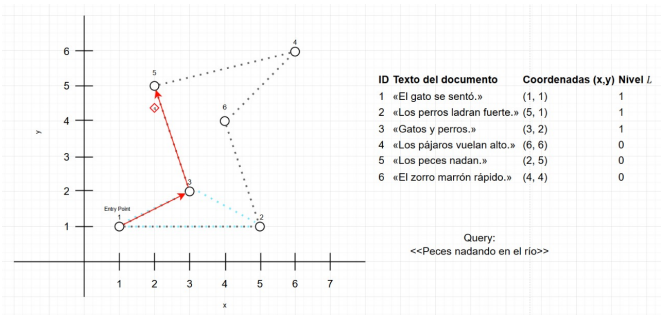

Fig. 1. Construcción (aristas punteadas) y búsqueda codiciosa (flechas rojas) en un índice HNSW con $M = 2$. El rombo rojo marca la consulta $q =$ Peces nadando en el río.

a) Recorrido de la consulta q (Peces nadando en el río):

1) **Nivel 1.** El algoritmo parte del *entry point* (nodo 1). Compara $d(q, 1)$ con las distancias a sus dos aristas largas 1–2 y 1–3; el nodo 3 resulta el más próximo y pasa a ser el nuevo punto de referencia.
2) **Nivel 0.** Desde 3 se exploran sus dos vecinos base (nodos 1 y 5). La distancia $d(q, 5) \approx 0.45$ es la menor encontrada, por lo que el algoritmo devuelve el nodo **5** como vecino más cercano aproximado.

En esta ruta $1 \to 3 \to 5$ el algoritmo evalúa sólo **5 distancias** (una inicial, dos en la capa 1 y dos en la capa 0), ilustrando el ahorro logarítmico de HNSW frente a un barrido lineal sobre los seis documentos.

V. Aplicaciones en Aprendizaje Automático

HNSW ha demostrado ser altamente eficaz en diversas aplicaciones de aprendizaje automático que requieren búsquedas rápidas y precisas en espacios de alta dimensión. Algunas de estas aplicaciones incluyen:

- **Sistemas de recomendación**: HNSW se utiliza para encontrar productos o contenidos similares a los intereses de los usuarios, mejorando la personalización de las recomendaciones.
- **Reconocimiento de imágenes**: En tareas de clasificación y búsqueda de imágenes, HNSW permite identificar imágenes similares en grandes bases de datos visuales.
- **Procesamiento de lenguaje natural**: HNSW facilita la búsqueda de palabras o frases semánticamente similares, mejorando la comprensión y generación de lenguaje natural.
- **Análisis de datos biométricos**: En aplicaciones de reconocimiento facial o de huellas dactilares, HNSW ayuda a identificar patrones similares en datos biométricos.

Estas aplicaciones destacan la versatilidad y eficiencia de HNSW en el manejo de grandes volúmenes de datos multidimensionales.

VI. Comparación con Otros Algoritmos

A continuación, se presenta una comparación entre HNSW y otros algoritmos populares de búsqueda de vecinos más cercanos:

TABLE II
Comparación de Algoritmos de Búsqueda de Vecinos Más Cercanos

Algoritmo	Precisión	Velocidad	Escalabilidad
HNSW	Alta	Rápida	Alta
LSH	Moderada	Muy rápida	Moderada
k-d Tree	Alta	Lenta	Baja
IVF	Alta	Moderada	Alta

Esta tabla muestra que HNSW ofrece un equilibrio entre precisión, velocidad y escalabilidad, lo que lo convierte en una opción preferida para aplicaciones que requieren búsquedas eficientes en grandes conjuntos de datos.

VII. Implementación Práctica

Existen varias bibliotecas que implementan el algoritmo HNSW, facilitando su integración en proyectos de aprendizaje automático. Algunas de estas bibliotecas incluyen:

- **hnswlib**: Una biblioteca ligera y eficiente para la implementación de HNSW en Python.
- **FAISS**: Una biblioteca de Facebook AI Research que proporciona implementaciones de HNSW y otros algoritmos de búsqueda de vecinos más cercanos.
- **NMSLIB**: Una biblioteca que ofrece implementaciones de HNSW y otros algoritmos de búsqueda aproximada de vecinos más cercanos.

Estas bibliotecas proporcionan interfaces sencillas para construir índices HNSW y realizar búsquedas eficientes en grandes conjuntos de datos.

VIII. Resultados Experimentales y Comparación con Otras Estructuras

A. Comparativa de rendimiento de HNSW con otras estructuras

Para evaluar la efectividad del algoritmo Hierarchical Navigable Small World (HNSW), se han realizado múltiples experimentos comparativos con otras estructuras populares en la búsqueda aproximada de vecinos cercanos, tales como NSW, Annoy, VP-tree, FLANN y FALCONN. A continuación se presentan los resultados detallados para diversos conjuntos de datos multidimensionales.

1) Dataset SIFT: La Figura 2 muestra la comparación en el dataset SIFT.

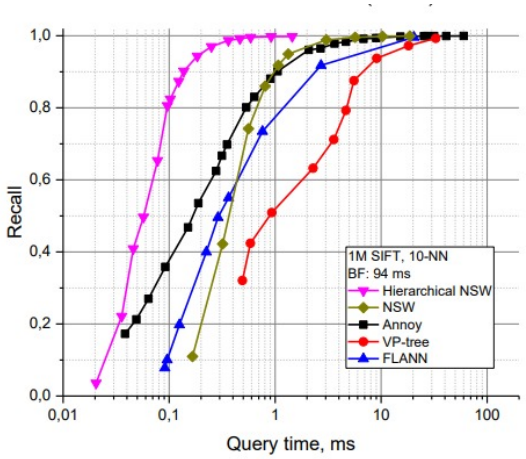

Fig. 2. Comparación del rendimiento en dataset SIFT entre HNSW y otros algoritmos populares.

2) Dataset GloVe: La Figura 3 presenta la comparación en el dataset GloVe.

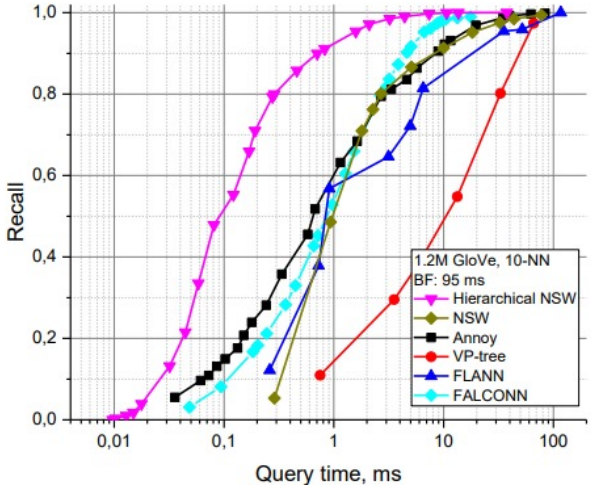

Fig. 3. Comparación del rendimiento en dataset GloVe entre HNSW y otros algoritmos populares.

3) Dataset CoPhIR: La Figura 4 ilustra la comparación en el dataset CoPhIR.

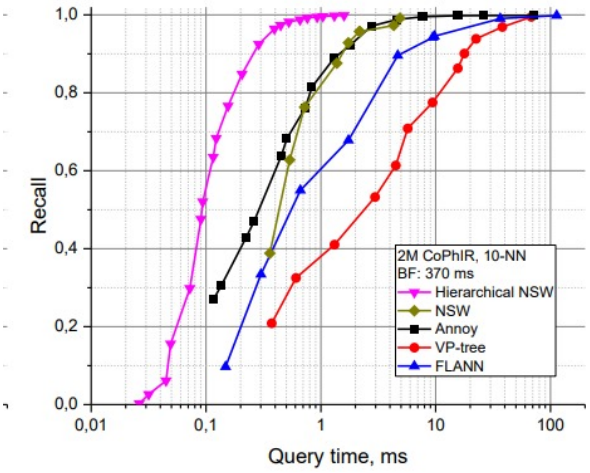

Fig. 4. Comparación del rendimiento en dataset CoPhIR entre HNSW y otros algoritmos populares.

4) Dataset Random (d=4): La Figura 5 muestra la comparación en un dataset aleatorio de dimensión 4.

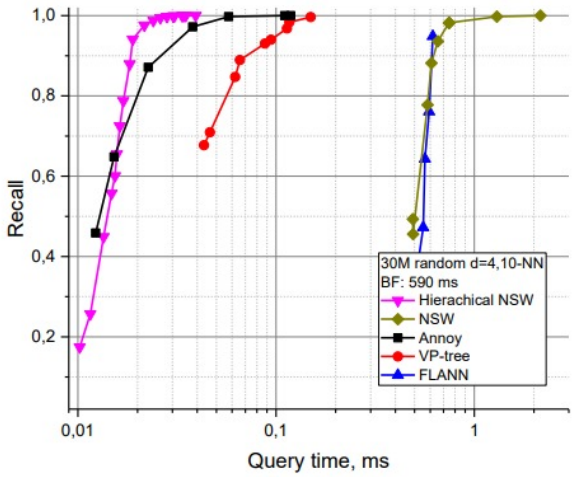

Fig. 5. Comparación del rendimiento en dataset aleatorio (d=4) entre HNSW y otros algoritmos populares.

5) Dataset MNIST: La Figura 6 exhibe la comparación en el dataset MNIST.

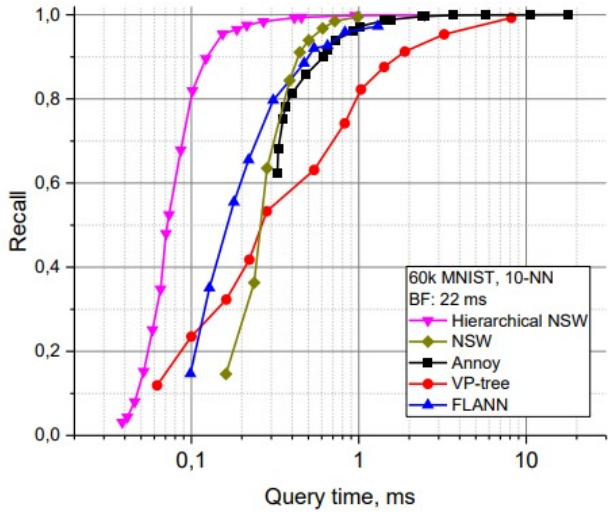

Fig. 6. Comparación del rendimiento en dataset MNIST entre HNSW y otros algoritmos populares.

6) Dataset Deep: Finalmente, la Figura 7 muestra la comparación en el dataset Deep.

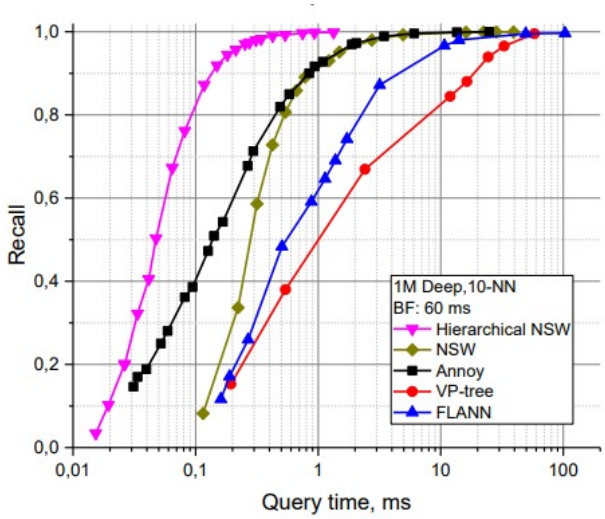

Fig. 7. Comparación del rendimiento en dataset Deep entre HNSW y otros algoritmos populares.

Estos resultados muestran claramente que el algoritmo HNSW alcanza niveles superiores de recall en menor tiempo de consulta en comparación con otras estructuras populares, destacándose especialmente en contextos con grandes volúmenes y alta dimensionalidad.

IX. Visualización del Proceso de Búsqueda en HNSW

El diagrama en la Figura 8 ilustra cómo se lleva a cabo la búsqueda de vecinos más cercanos en el algoritmo HNSW. Este diagrama muestra cómo la búsqueda comienza en el nivel superior del grafo, donde se encuentran conexiones de largo alcance. Luego, a medida que la búsqueda desciende a niveles más bajos, se realiza una exploración más detallada.

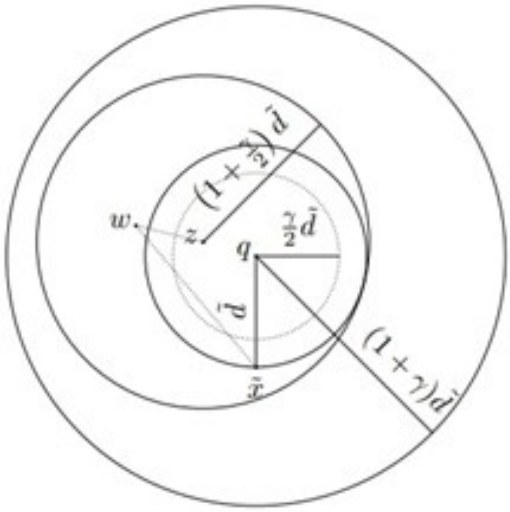

Fig. 8. Visualización del proceso de búsqueda en el algoritmo HNSW.

Como se muestra en la Figura 8, la búsqueda de vecinos en HNSW se representa mediante círculos concéntricos alrededor de los puntos q (nodo de consulta) y $\tilde{x}$ (nodo candidato). Los círculos indican las áreas de influencia dentro de las cuales se realiza la búsqueda. El nodo w es un **nodo no expandido**, lo que significa que no se encuentra en la misma área de influencia que q o $\tilde{x}$, según el teorema demostrado en la figura.

Este proceso jerárquico es crucial para optimizar el rendimiento de la búsqueda, permitiendo que se inicie en un nivel alto con un alcance global y luego se refine a medida que se desciende en la jerarquía para encontrar los vecinos más cercanos con mayor precisión.

X. Conclusiones

El algoritmo Hierarchical Navigable Small World (HNSW) ha demostrado ser una herramienta poderosa y eficiente para la búsqueda aproximada de vecinos más cercanos en grandes conjuntos de datos multidimensionales. Su estructura jerárquica y su enfoque de búsqueda codiciosa permiten realizar búsquedas rápidas y precisas, incluso en espacios de alta dimensión.

Su aplicación en diversas áreas del aprendizaje automático, como sistemas de recomendación, reconocimiento de imágenes y procesamiento de lenguaje natural, resalta su versatilidad y efectividad. Además, su implementación práctica mediante bibliotecas como hnswlib, FAISS y NMSLIB facilita su integración en proyectos reales.

Se espera que futuras investigaciones continúen explorando y optimizando el algoritmo HNSW, ampliando su aplicabilidad y mejorando su rendimiento en escenarios aún más desafiantes.

References

[1] Y. A. Malkov y D. A. Yashunin, "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824-836, 2020.

[2] Y. Al-Jazzazi, H. Diwan, J. Gou, C. Musco, C. Musco y T. Suel, "Distance Adaptive Beam Search for Provably Accurate Graph-Based Nearest Neighbor Search," arXiv preprint arXiv:2505.15636, 2025.

[3] Fogfish, "Hierarchical Navigable Small World (HNSW) algorithm," GitHub repository, 2023. [En línea]. Disponible: https://github.com/fogfish/hnsw.

[4] H. Ootomo, A. Naruse, C. Nolet, R. Wang, T. Feher y Y. Wang, "CAGRA: Highly Parallel Graph Construction and Approximate Nearest Neighbor Search for GPUs," en *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, 2024. arXiv:2308.15136. [En línea]. Disponible: https://arxiv.org/abs/2308.15136.