Optimizing HNSW in the Age of Vector Databases

Blaise Munyampirwa
Argmax

Vihan Lakshman
MIT CSAIL

Benjamin Coleman
Google DeepMind

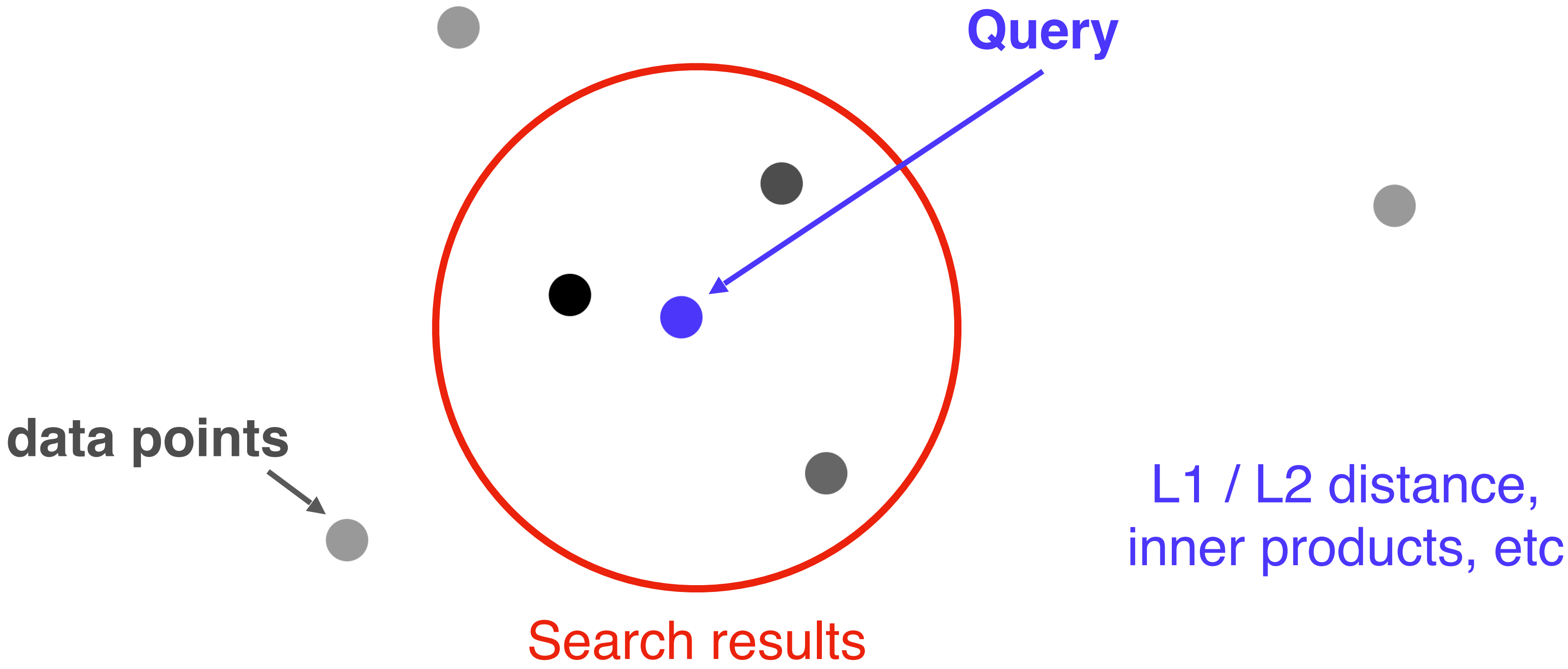
Spoilers

Free lunch: Remove the hierarchy of HNSW for ~30% lower construction memory + systems benefits

Cheap lunch: Reorder the HNSW graph layout to reduce cache misses for >20% faster queries + systems benefits

Doing it in production is easy and can immediately improve performance

Typical search problem



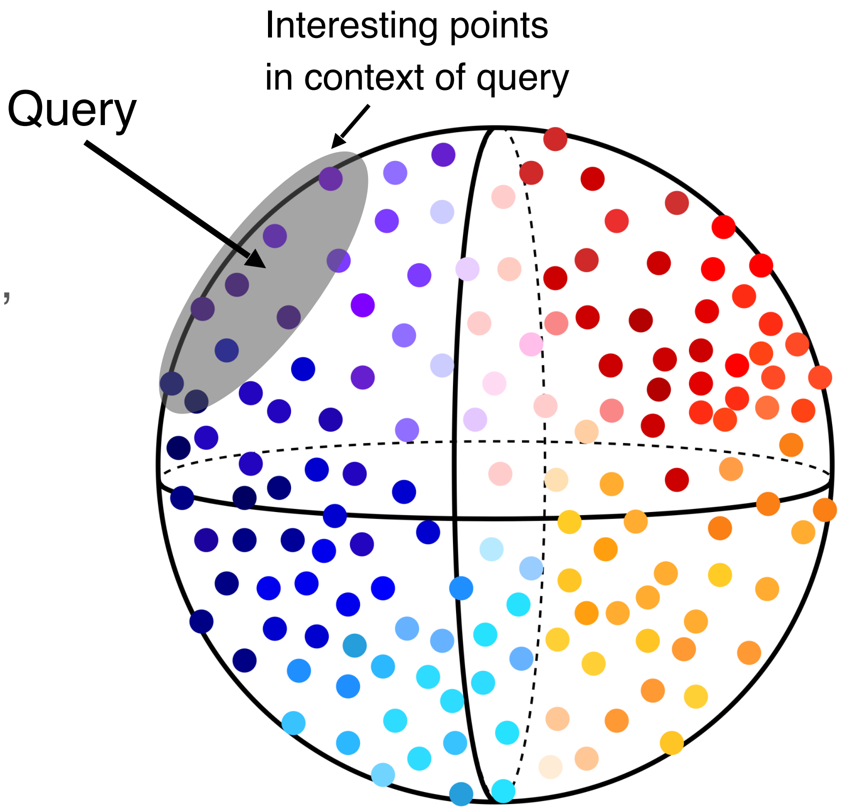
Why near neighbor?

Embedding search

- Retrieval-augmented generation (RAG), recommender systems (retrieval)

Classification

- Popular ML classifier but poor latency / memory

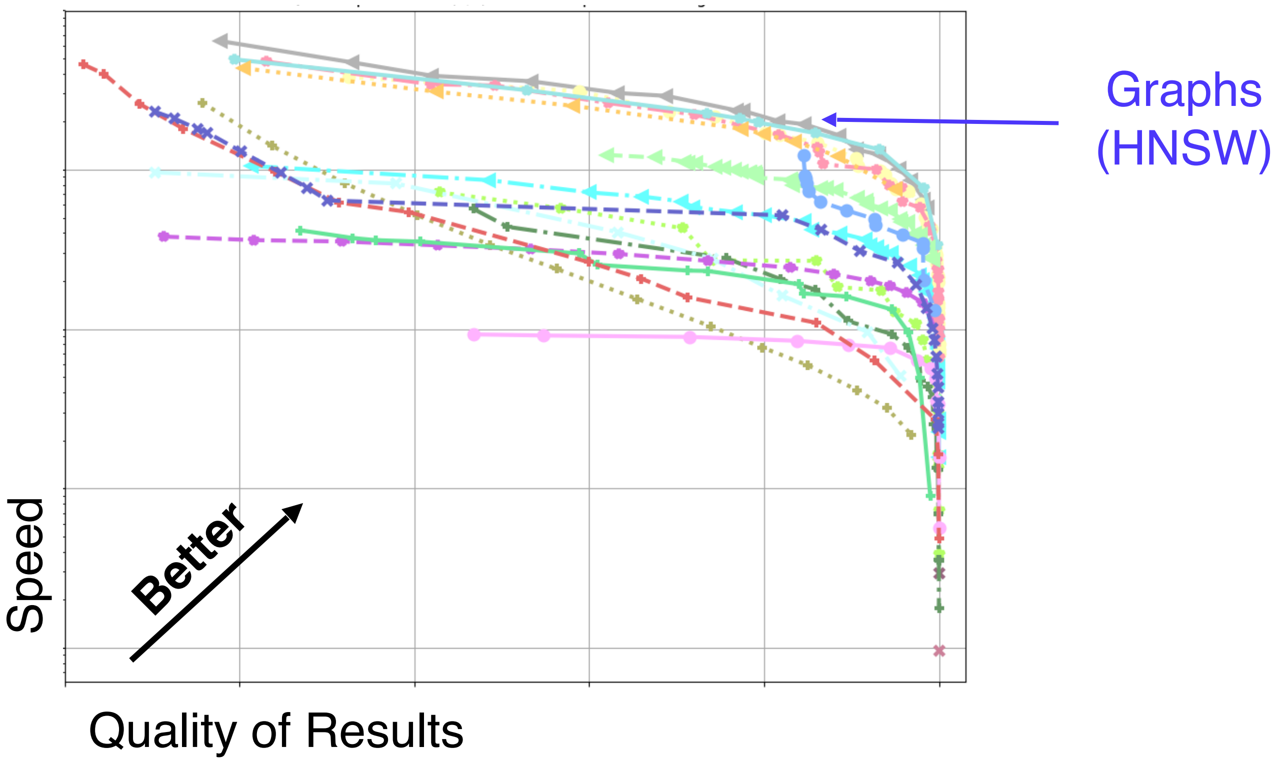


Open source near neighbor projects

clustering **graphs** trees **graphs** hashing

graphs trees **graphs** **graphs**

Graphs are high performance



Data from ANN-Benchmarks (Aumüller, et. al. ICSSA 2017)

In practice, a ton of people just use HNSW...

Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs

YA Malkov, DA Yashunin

☆ Save 99 Cite Cited by 1702 Related articles All 12 versions

nmslib / hnswlib Public

Header-only C++/python library

Apache-2.0 license

☆ 4.5k stars 673 forks

...sometimes without knowing.

HNSW is integrated into many libraries and is often the default choice.

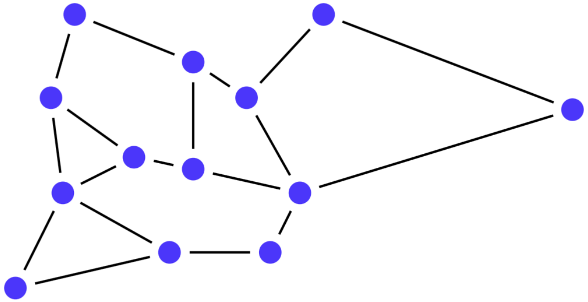
One weird trick to improve HNSW

"*Down with the Hierarchy: The H in HNSW stands for Hubs*"

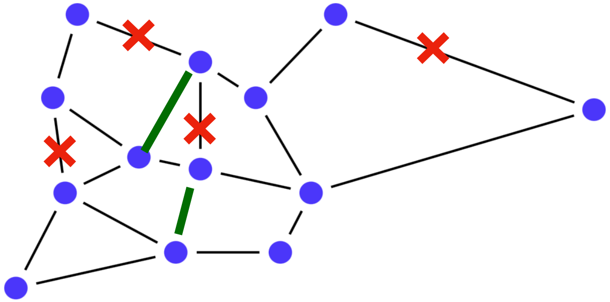
arXiv 2024

KNN with graphs (high-level)

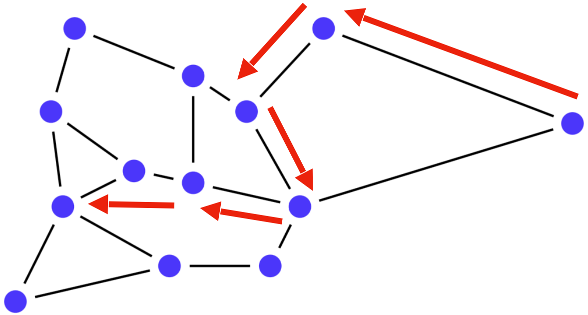
1. Connect nearby points



2. Add / delete edges (based on heuristic)

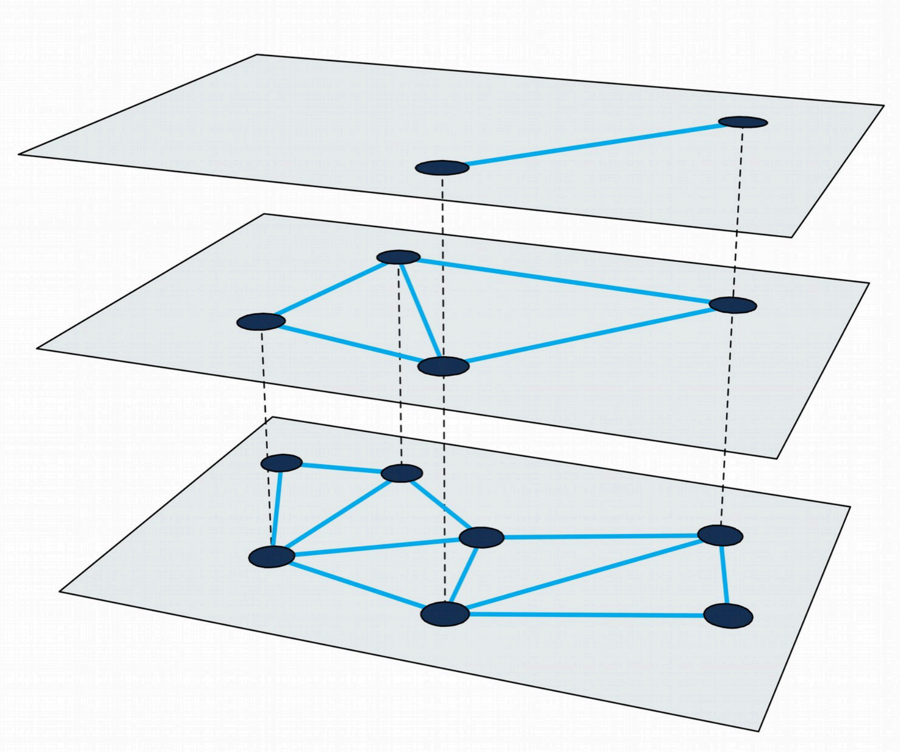


3. Walk to find neighbors

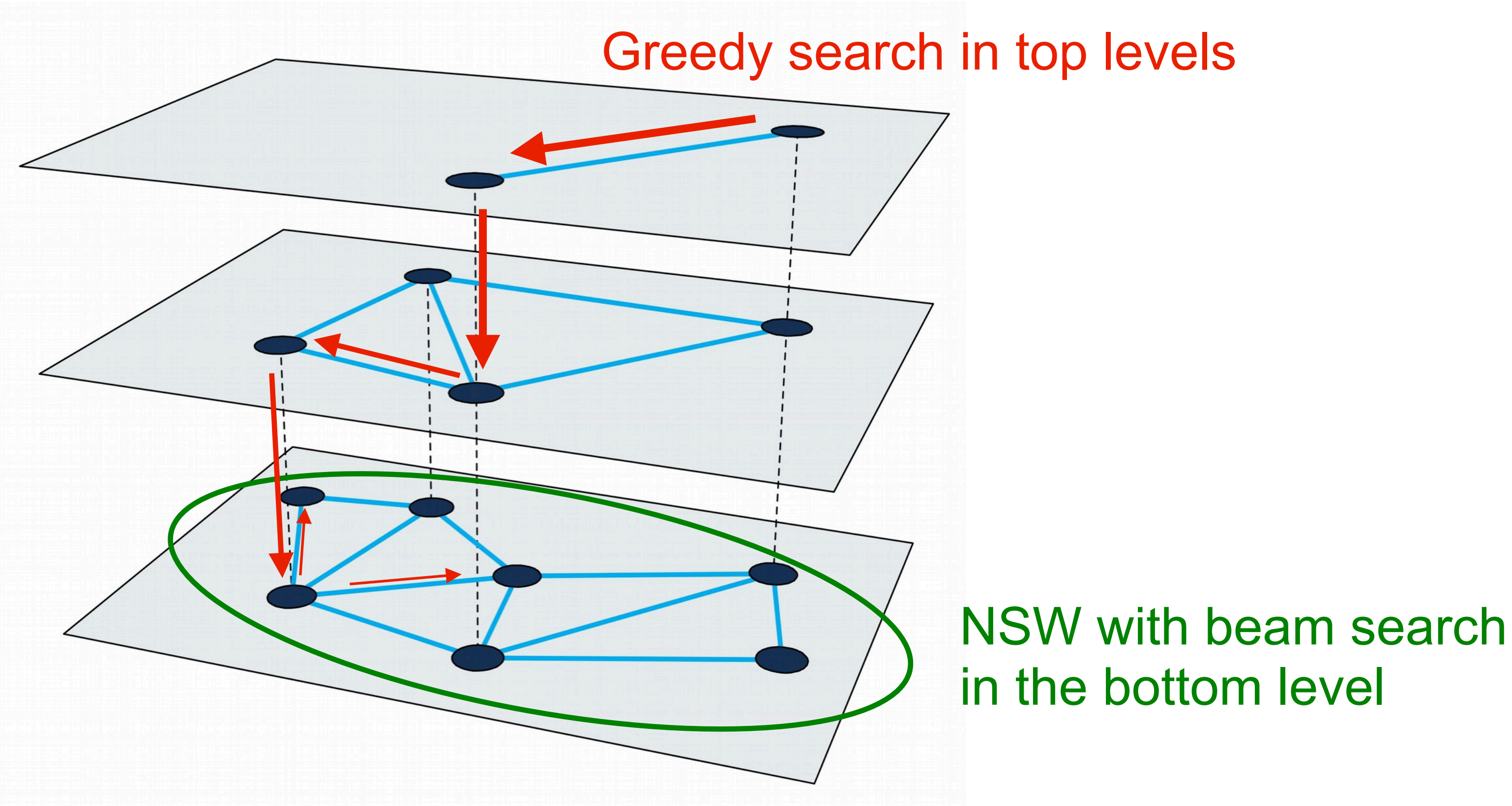


HNSW: Hierarchical Navigable Small-World Graphs

1. Assign points to levels
2. Build a KNN graph in each level
3. Walk along each level, dropping down when you're done
4. Thoroughly explore the bottom

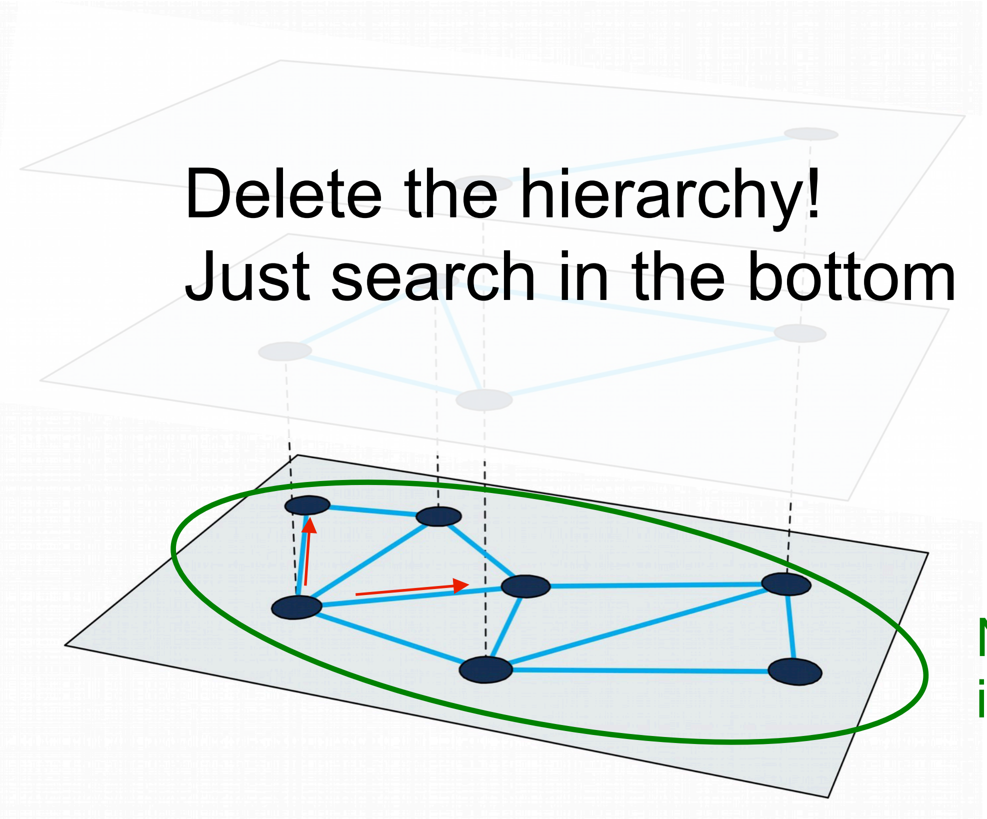


HNSW: Hierarchical Navigable Small-World Graphs



The trick: ~~Hierarchical~~ Navigable Small-World Graphs

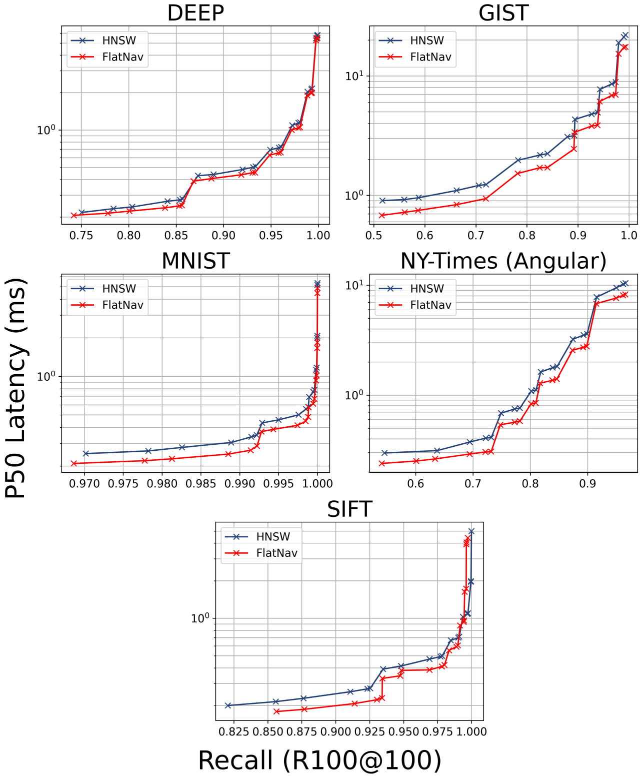
Delete the hierarchy!
Just search in the bottom

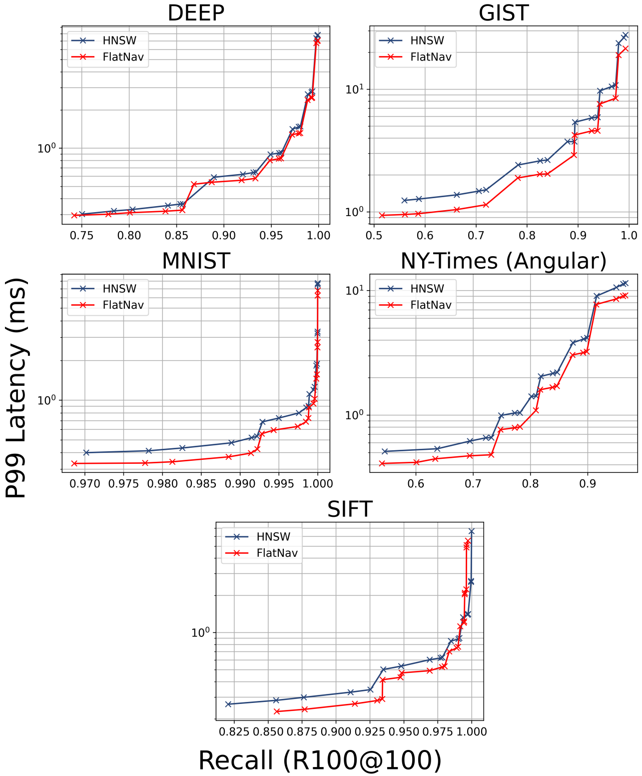


NSW with beam search
in the bottom level

The trick works in practice

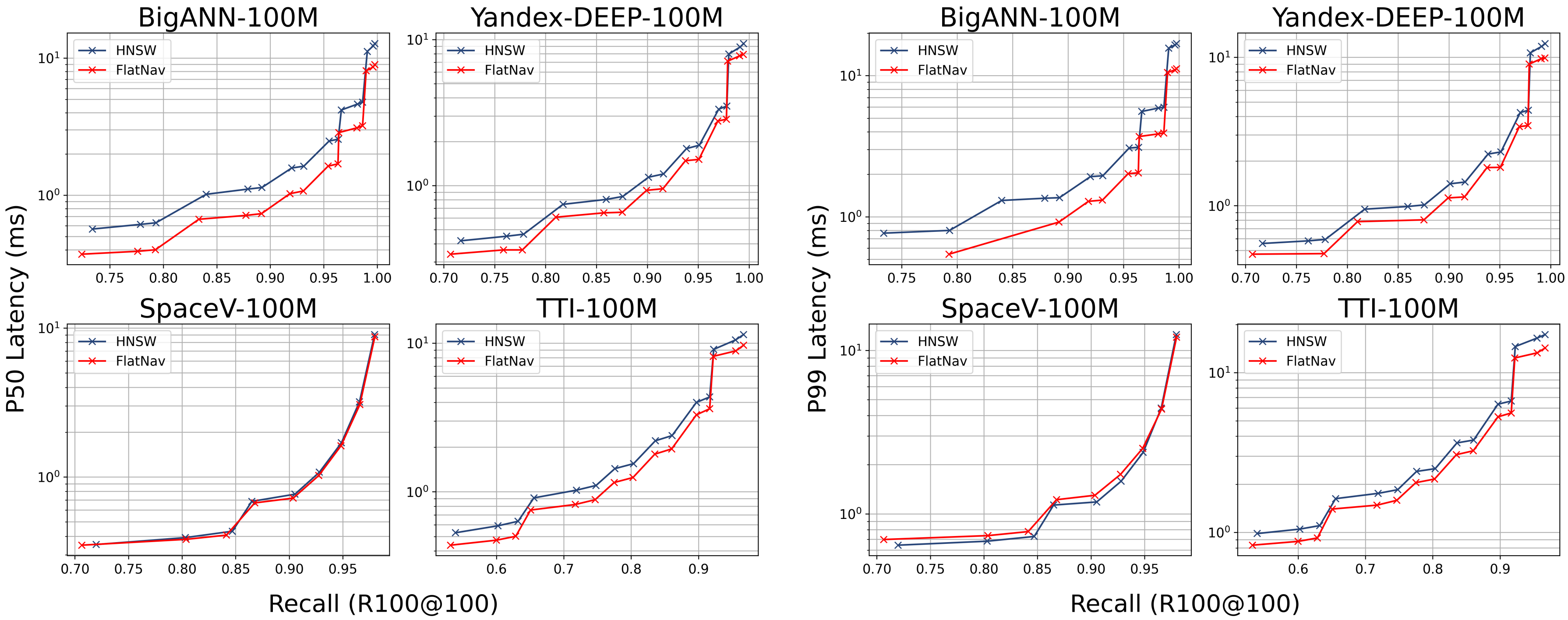
ANN Benchmarks (1M scale): No search Latency Difference





The trick works in practice

Big-ANN Benchmarks (100M scale): No search Latency Difference



How could this *possibly* be okay?

If NSW is all you need...

Why don't we need the hierarchy?

Does this always work?

When does hierarchy still make sense?

How could this *possibly* be okay?

If NSW is all you need...

Why don't we need the hierarchy?

HNSW was a big improvement in 2016. What changed?

Does this <u>always</u> work?

Do we need the hierarchy for "insurance?"

When does hierarchy still make sense?

How could this *possibly* be okay?

If NSW is all you need...

Why don't we need the hierarchy?

We think we (finally) have the answers

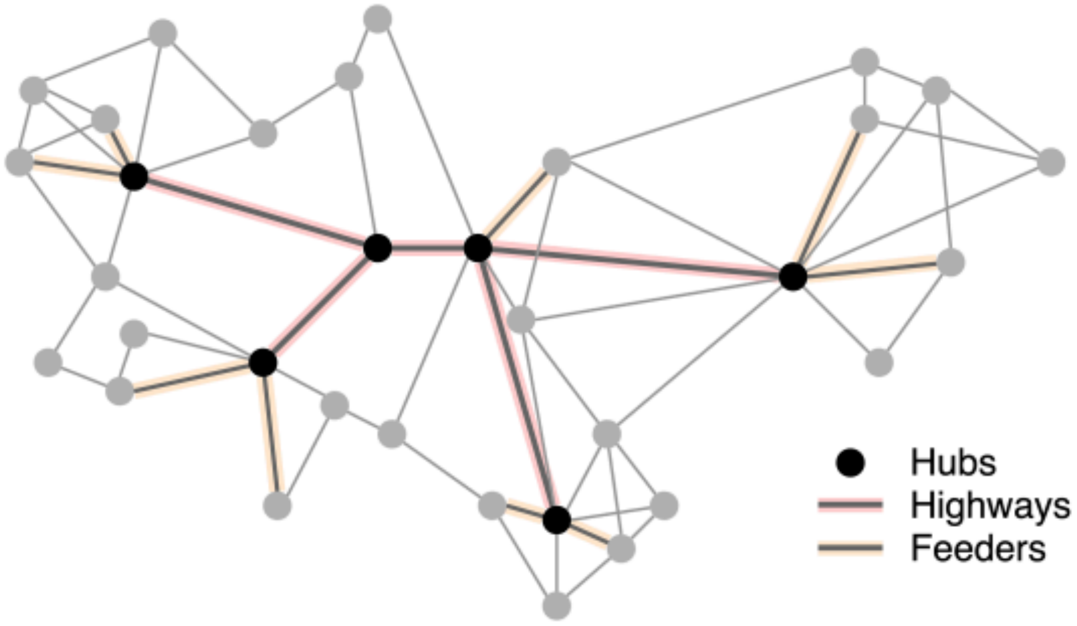
Do we need the hierarchy for "insurance"?

When does hierarchy still make sense?

Hub-Highway Hypothesis

The Hub-Highway Hypothesis

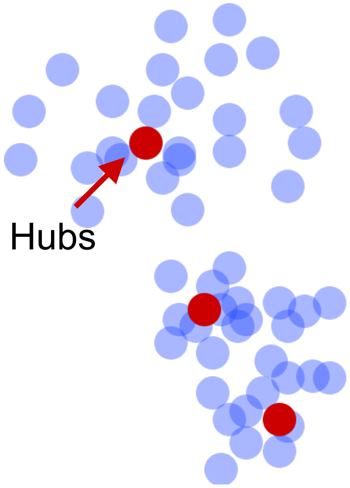
Hypothesis: *In high dimensions, k-NN proximity graphs naturally form a highway routing structure.*



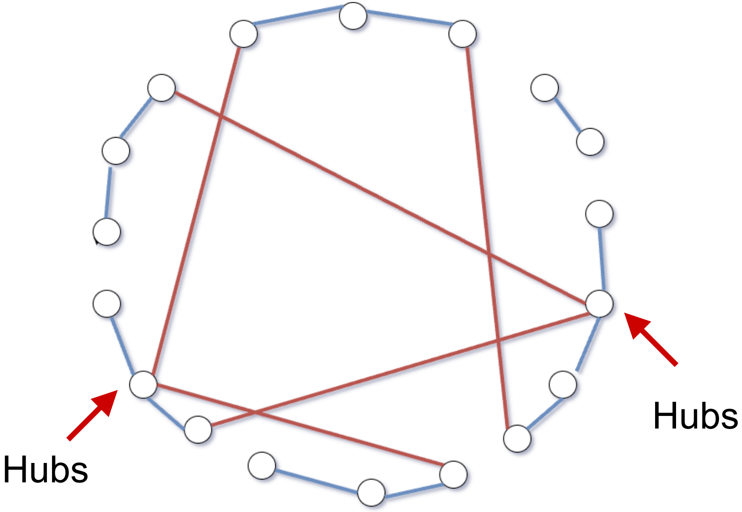
Hubs in space

Hub: *A point that is close to many other points.*

- A mid-point, center of a cluster, center of a KNN graph



A toy dataset

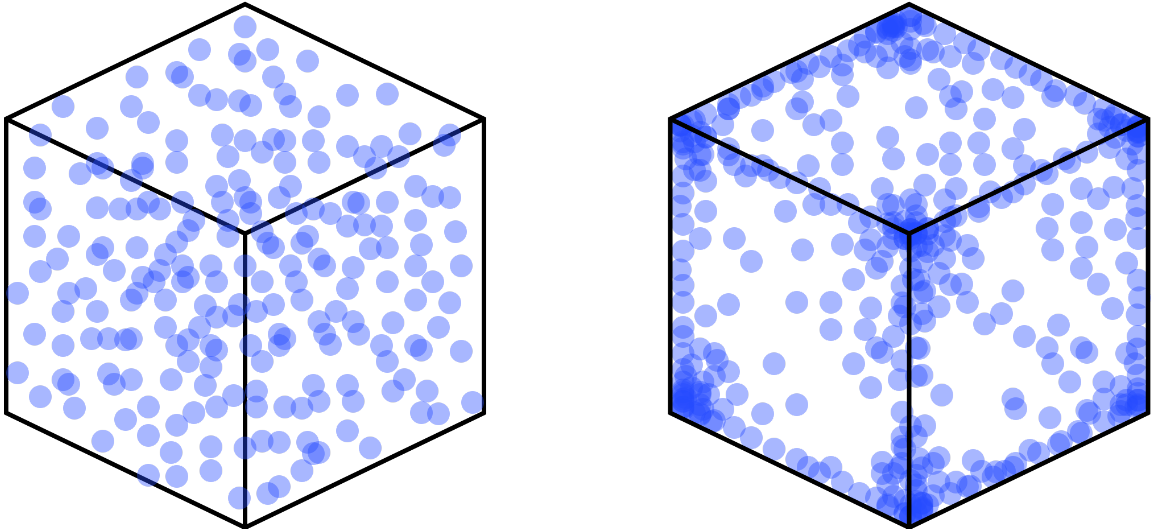


A toy KNN graph

Ideas from "*Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data*" Radovanovic et. al.

As dimension increases, we see more hubs. Why?

Intuition: uniform distribution in a (hyper) cube

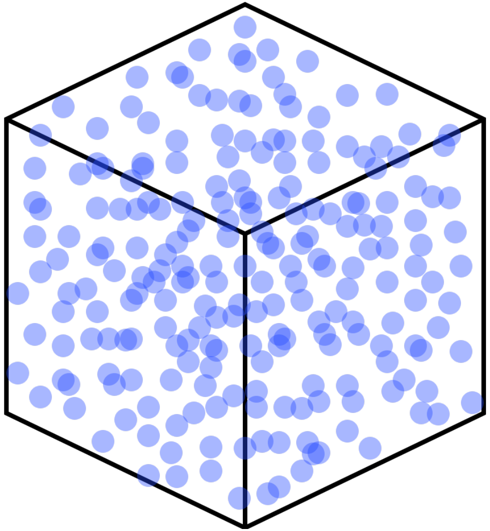


Low dimension High dimension

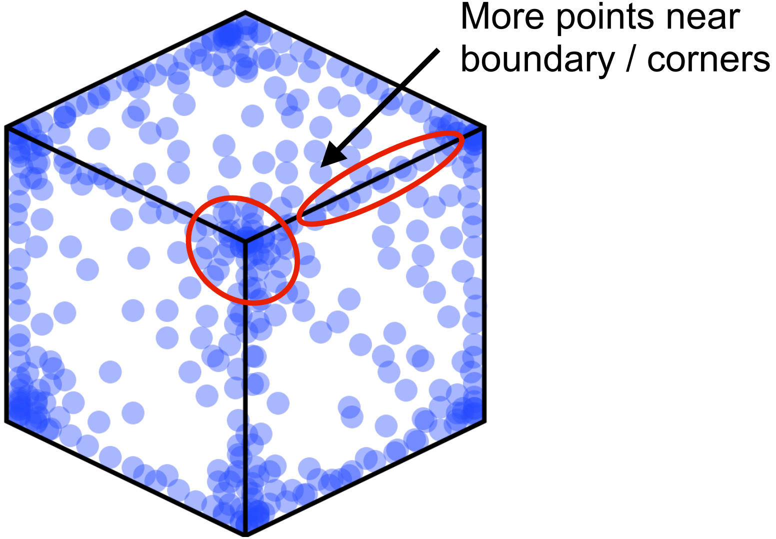
Example from "*The Hubness Phenomenon: Fact or Artifact?*" Borgelt et. al.

As dimension increases, we see more hubs. Why?

Intuition: uniform distribution in a (hyper) cube



Low dimension

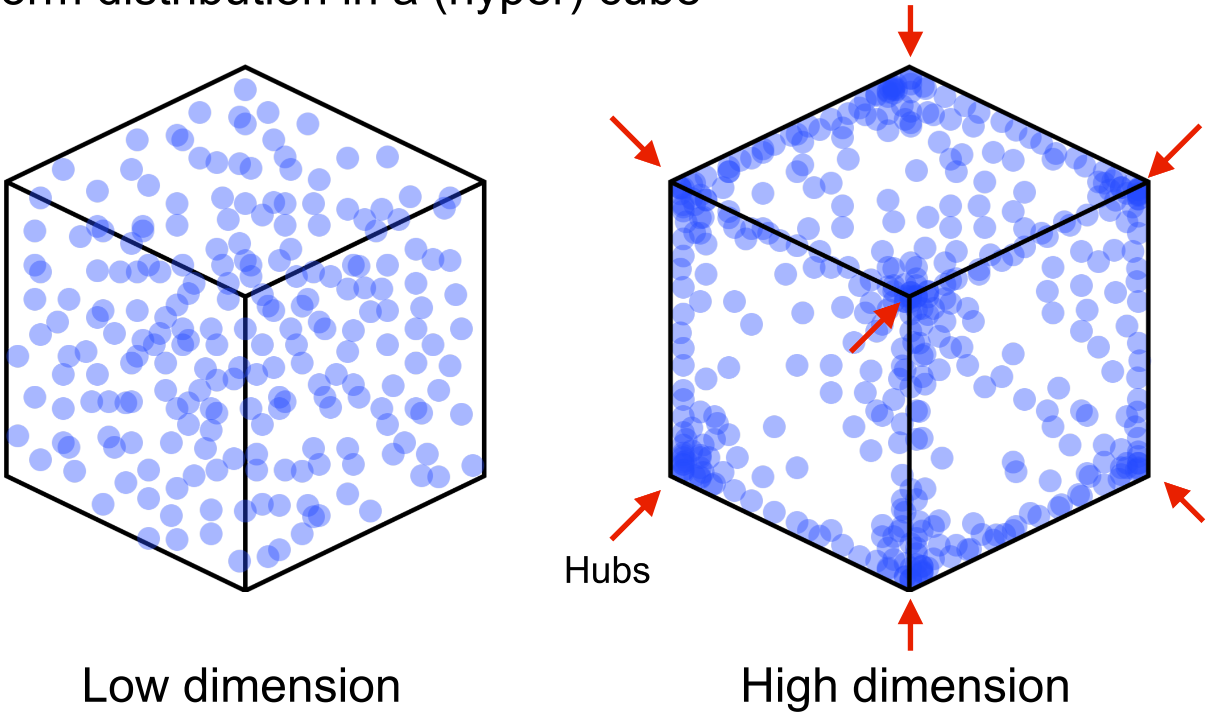


High dimension

Example from "*The Hubness Phenomenon: Fact or Artifact?*" Borgelt et. al.

As dimension increases, we see more hubs. Why?

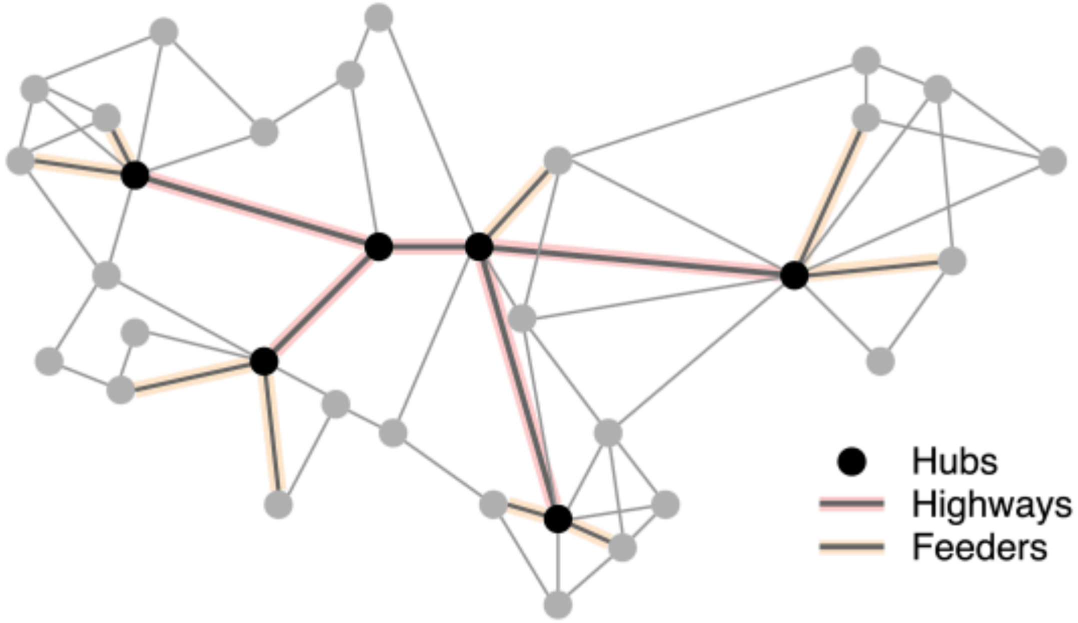
Intuition: uniform distribution in a (hyper) cube



Example from "*The Hubness Phenomenon: Fact or Artifact?*" Borgelt et. al.

The Hub-Highway Hypothesis

Hypothesis: *In high dimensions, k-NN proximity graphs naturally form a highway routing structure.*



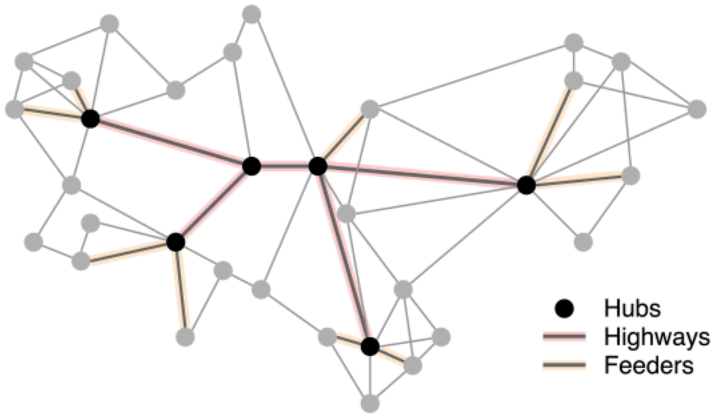
Demonstrating the HHH

Okay cool. **Prove it.**

1. Beam search visits some nodes way more often than others - these are the "hubs."

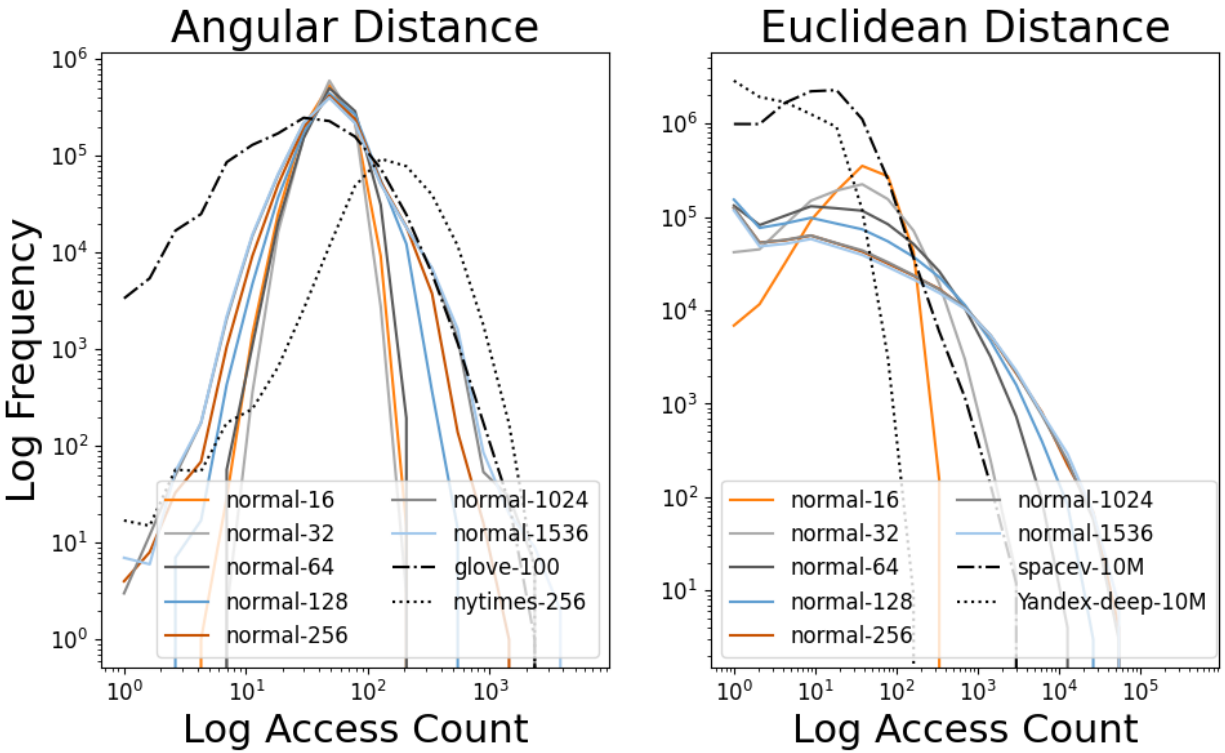
2. The hubs are connected to each other, forming a highway-feeder structure.

3. Queries take the "hub highway" early in search, to arrive at the neighborhood of the



Skewness of the Node Access Distribution

This implies that beam search visits certain nodes (the "hubs") way more often than the rest



Connectivity of the Highway Feeders

Hub nodes are not only frequently accessed, they are also very well connected

"*Down with the Hierarchy: The H in HNSW stands for Hubs*"

arXiv 2024

Table 5: Two-sample t-test and Mann-Whitney U-test p-values. Hub nodes are selected using the p99 threshold for the node access distribution.

Dataset	Dim	Mann-Whitney	Two-Sample t-Test	Effect Size
IID Normal (Angular)	16	0.0006	0.0006	0.1745
IID Normal (L2)	16	$< 10^{-5}$	$< 10^{-5}$	0.6621
IID Normal (Angular)	32	0.0347	0.0347	0.0972
IID Normal (L2)	32	$< 10^{-5}$	$< 10^{-5}$	0.8173
IID Normal (Angular)	64	0.0359	0.0417	0.0927
IID Normal (L2)	64	$< 10^{-5}$	$< 10^{-5}$	0.8725
IID Normal (Angular)	128	0.0093	0.0070	0.1316
IID Normal (L2)	128	$< 10^{-5}$	$< 10^{-5}$	0.8428
IID Normal (Angular)	256	$< 10^{-5}$	$< 10^{-5}$	0.3110
IID Normal (L2)	256	$< 10^{-5}$	$< 10^{-5}$	0.8582
IID Normal (Angular)	1024	0.1472	0.1318	0.0598
IID Normal (L2)	1024	$< 10^{-5}$	$< 10^{-5}$	0.8314
IID Normal (Angular)	1536	$< 10^{-5}$	$< 10^{-5}$	0.2356
IID Normal (L2)	1536	$< 10^{-5}$	$< 10^{-5}$	0.8568
GloVe	100	$< 10^{-5}$	$< 10^{-5}$	0.7642
NYTimes	256	$< 10^{-5}$	$< 10^{-5}$	0.9305
GIST	960	$< 10^{-5}$	$< 10^{-5}$	0.6829
Yandex-DEEP	96	0.0013	0.0013	0.1614
Microsoft-SpaceV	100	0.0011	0.0011	0.1644

Connectivity of the Highway Feeders

Hub nodes are not only frequently accessed, they are also very well connected

"*Down with the Hierarchy: The H in HNSW stands for Hubs*"

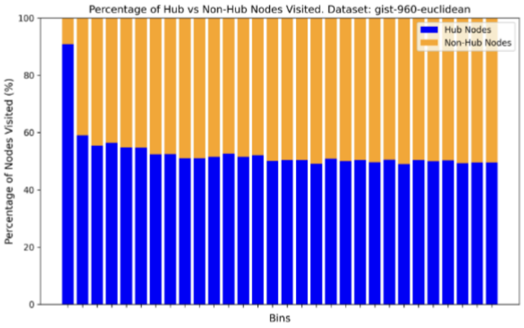
arXiv 2024

Table 5: Two-sample t-test and Mann-Whitney U-test p-values. Hub nodes are selected using the p99 threshold for the node access distribution.

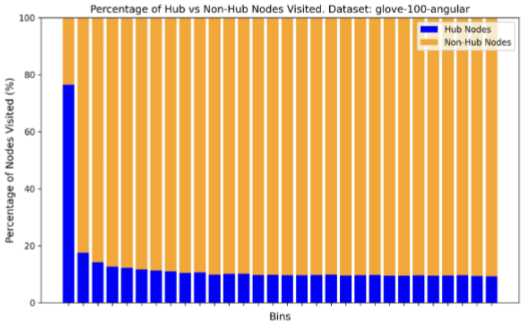
Dataset	Dim	Mann-Whitney	Two-Sample t-Test	Effect Size
IID Normal (Angular)	16	0.0006	0.0006	0.1745
IID Normal (L2)	16	$< 10^{-5}$	$< 10^{-5}$	0.6621
IID Normal (Angular)	32	0.0347	0.0347	0.0972
IID Normal (L2)	32	$< 10^{-5}$	$< 10^{-5}$	0.8173
IID Normal (Angular)	64	0.0359	0.0417	0.0927
IID Normal (L2)	64	$< 10^{-5}$	$< 10^{-5}$	0.8725
IID Normal (Angular)	128	0.0093	0.0070	0.1316
IID Normal (L2)	128	$< 10^{-5}$	$< 10^{-5}$	0.8428
IID Normal (Angular)	256	$< 10^{-5}$	$< 10^{-5}$	0.3110
IID Normal (L2)	256	$< 10^{-5}$	$< 10^{-5}$	0.8582
IID Normal (Angular)	1024	0.1472	0.1318	0.0598
IID Normal (L2)	1024	$< 10^{-5}$	$< 10^{-5}$	0.8314
IID Normal (Angular)	1536	$< 10^{-5}$	$< 10^{-5}$	0.2356
IID Normal (L2)	1536	$< 10^{-5}$	$< 10^{-5}$	0.8568
GloVe	100	$< 10^{-5}$	$< 10^{-5}$	0.7642
NYTimes	256	$< 10^{-5}$	$< 10^{-5}$	0.9305
GIST	960	$< 10^{-5}$	$< 10^{-5}$	0.6829
Yandex-DEEP	96	0.0013	0.0013	0.1614
Microsoft-SpaceV	100	0.0011	0.0011	0.1644

Queries tend to take the “hub highway” early in the Search

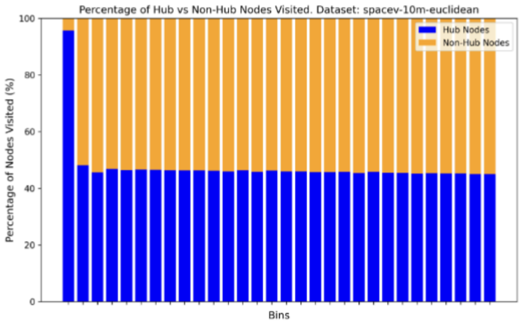
Partitioned bins = proportion of Hub nodes visited at a particular stage during search.



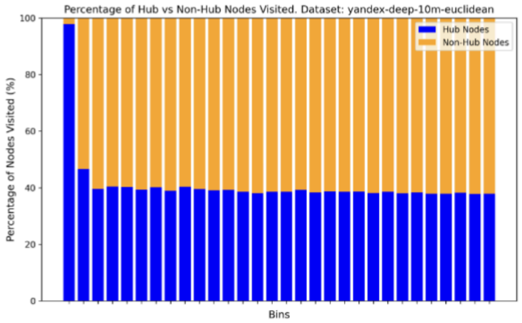
(a) Gist



(b) GloVe



(c) Microsoft SpaceV



(d) Yandex Deep

How could this *possibly* be okay?

If NSW is all you need...

Why don't we need the hierarchy?

Today's datasets are higher-dimensional than in 2016 - enough to form a long-range network naturally.

Does this always work? Yes, provided the data is high-dimensional.

When does the hierarchy still make sense?

It is robust against data that is intrinsically low-dimensional, providing "insurance" [1].

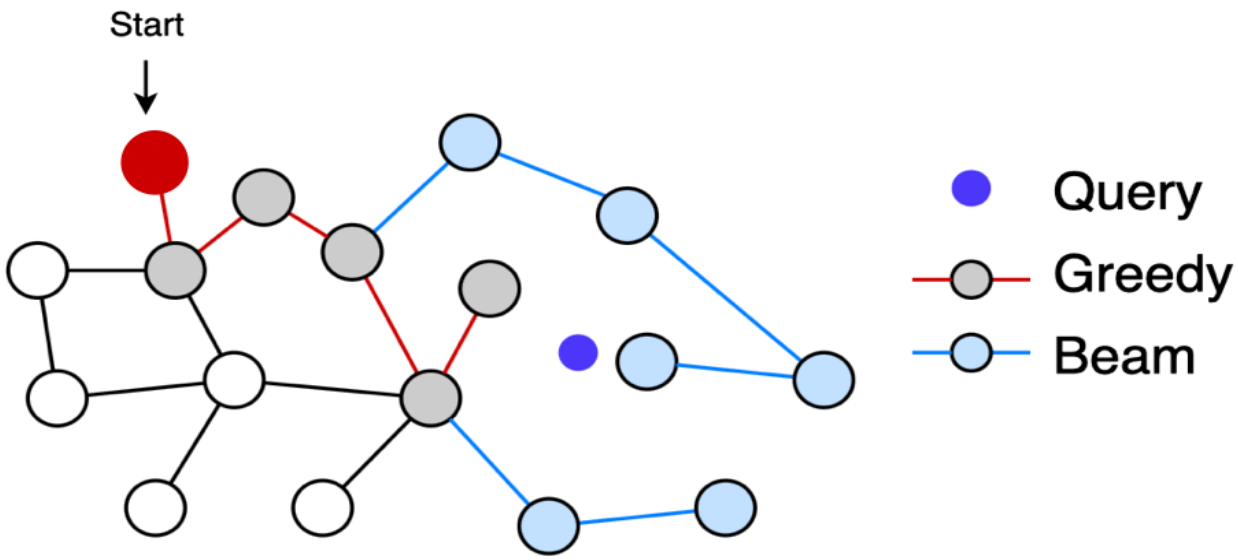
[1] Confirmed via personal correspondence with Yury Malkov

Another weird trick to improve HNSW

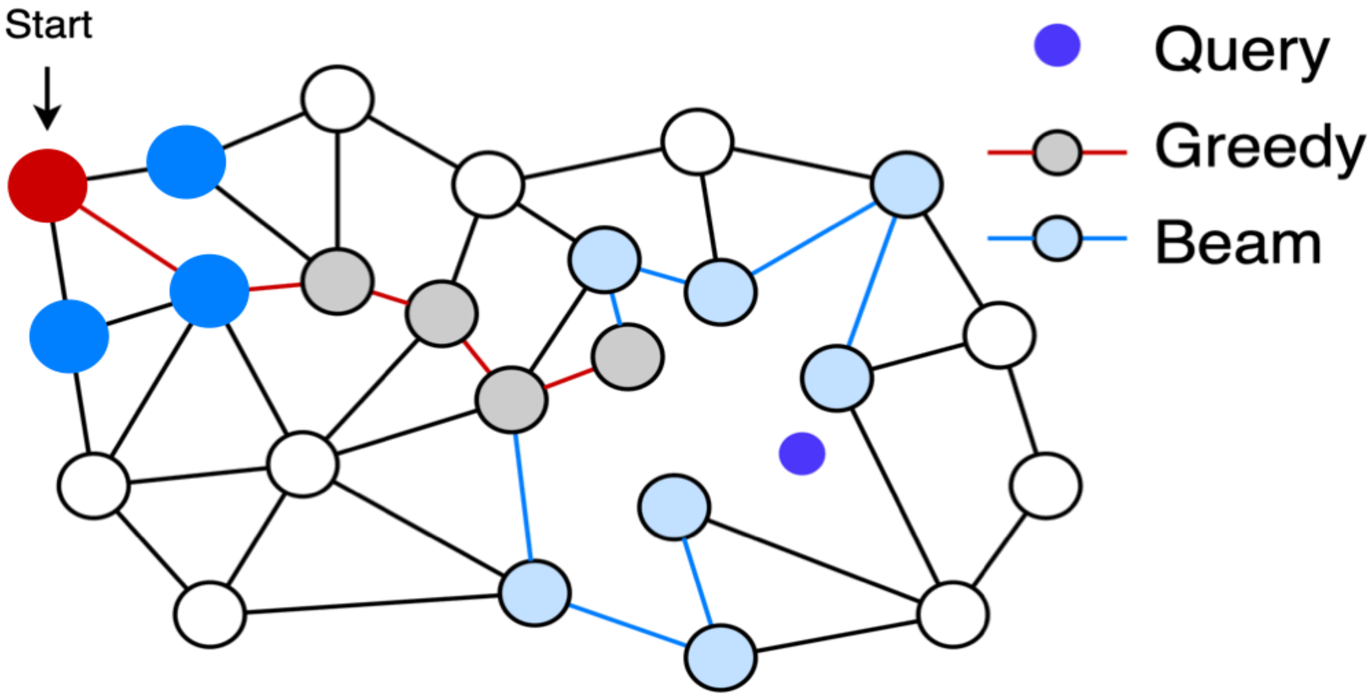
"*Graph Reordering for Cache-Efficient Near Neighbor Search*"

NeurIPS 2022

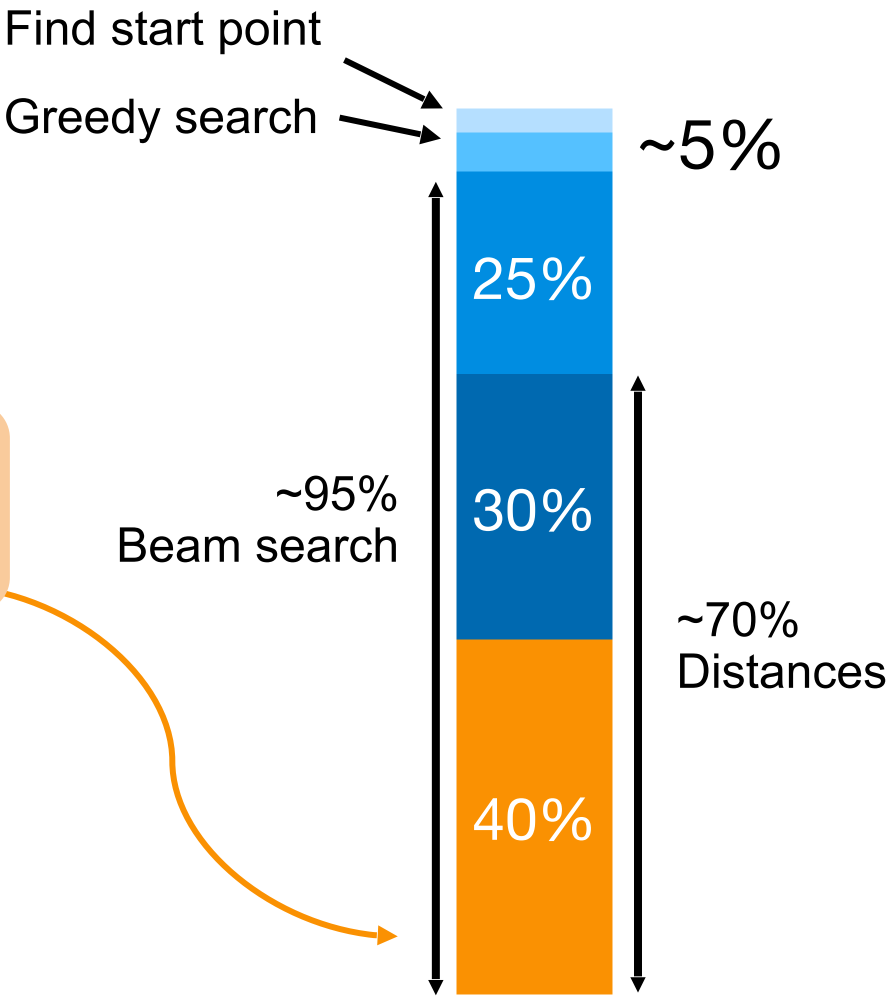
Where does HNSW spend time?



Where does HNSW spend time?



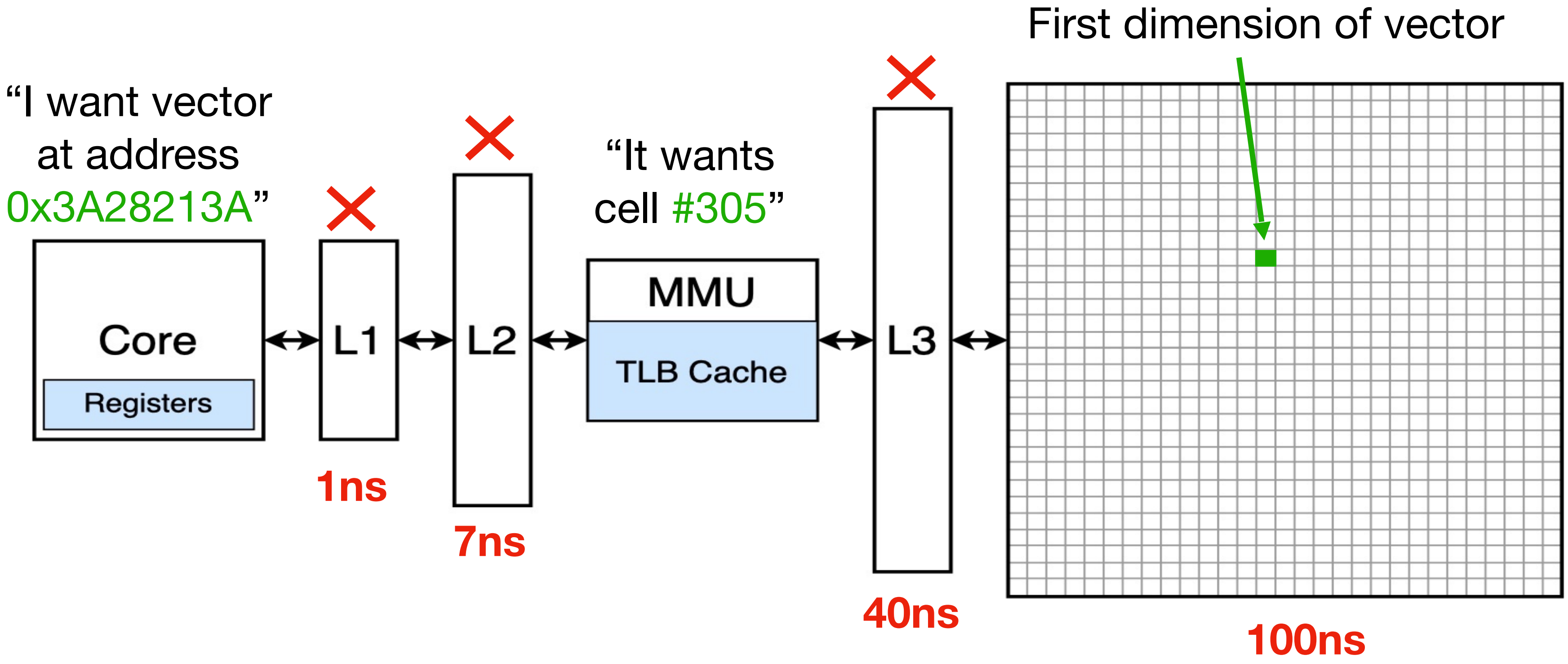
Profiling results



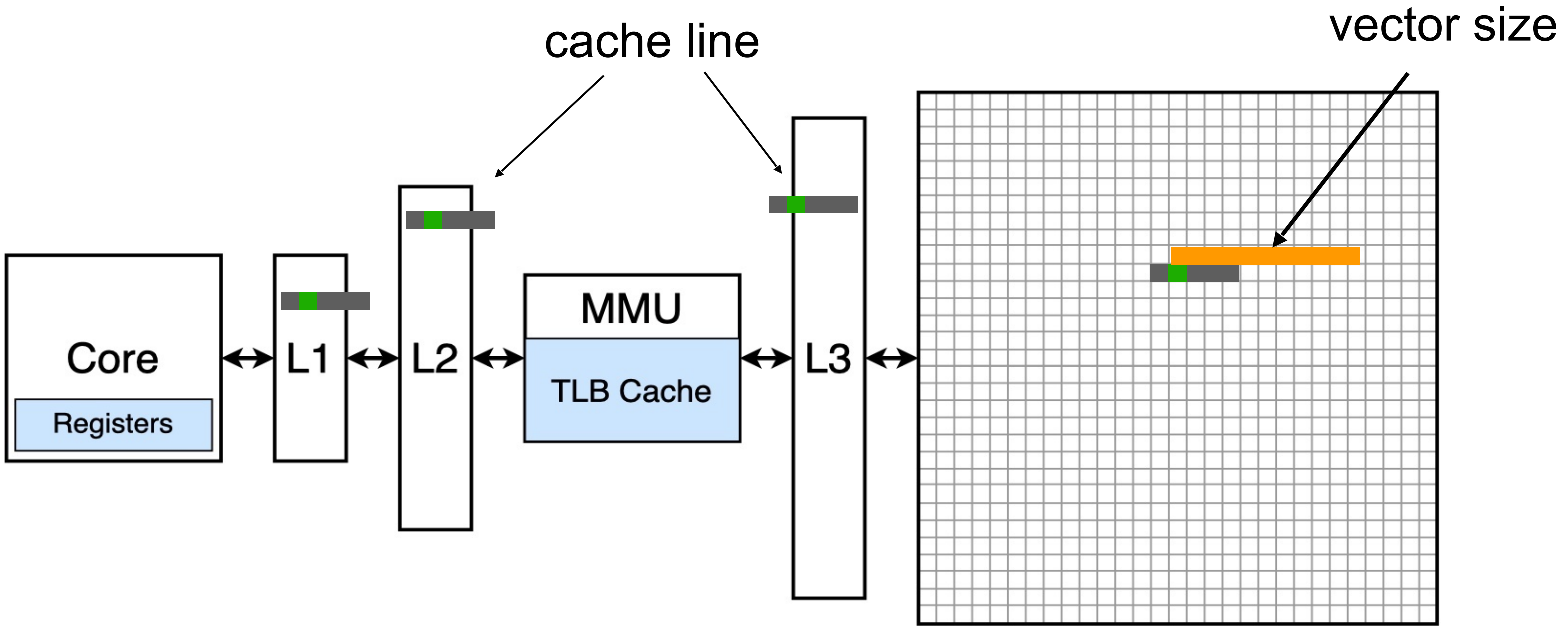
First dimension of the distance computation?!

Details: Ran perf on HNSW and Flatnav for 10K SIFT1B / DEEP1B search queries. Source annotation + cache counters + stack trace samples. Instrumentation with valgrind / cachegrind

Memory access: *the slowest thing on the computer*

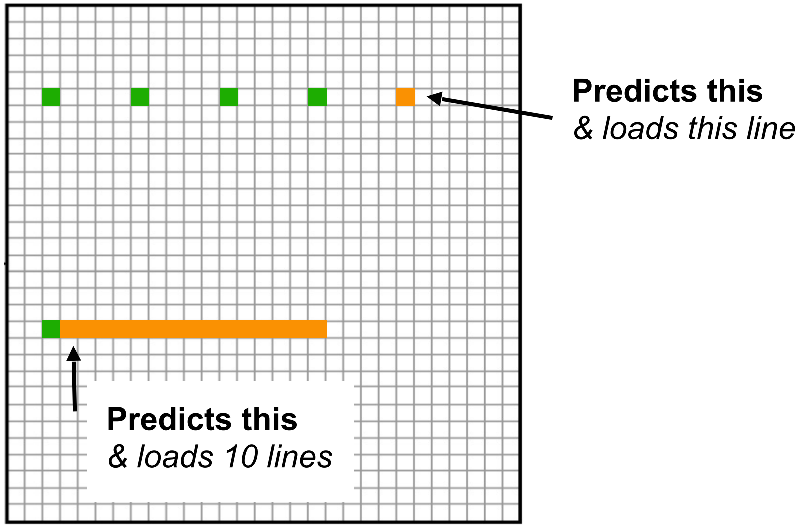


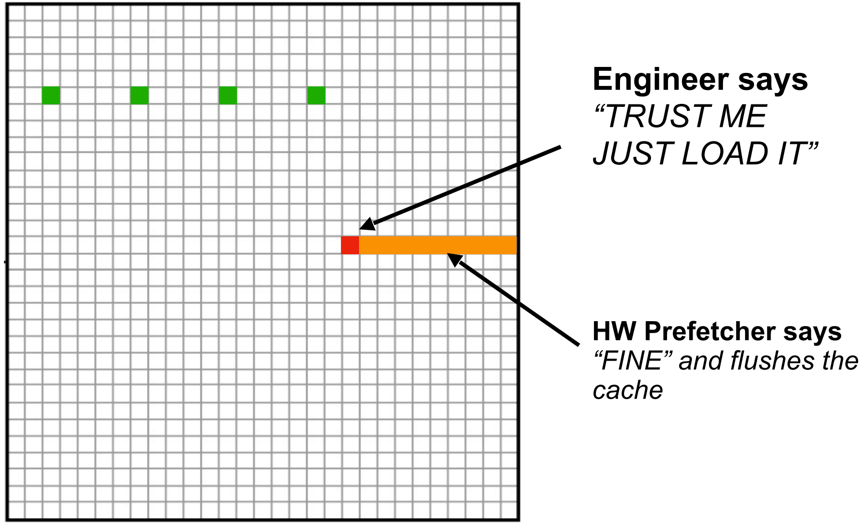
Cache is supposed to fix it, but…



sizeof(Node) > sizeof(cache line), so every node walk is a cache miss!

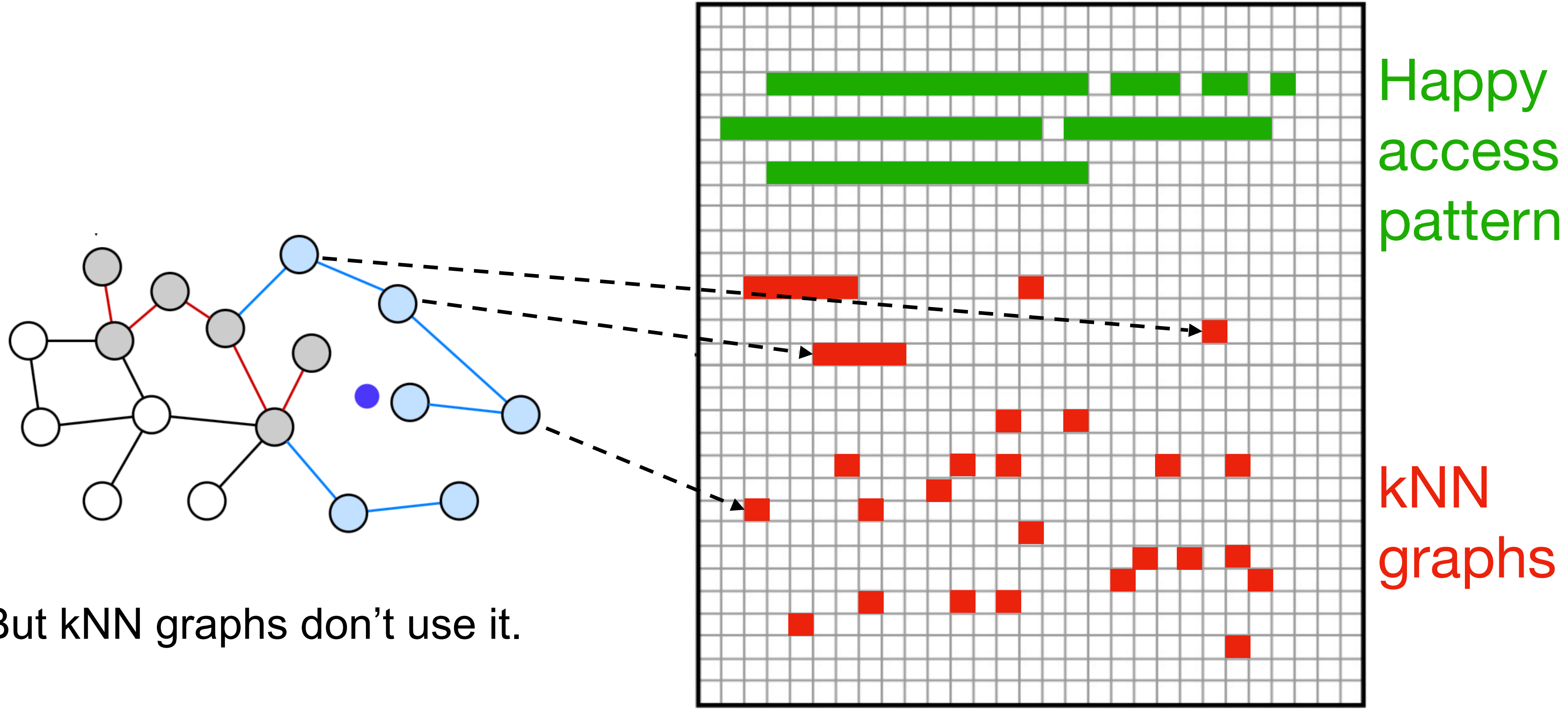
Prefetcher is supposed to fix it, but…





Hardware prefetch is dumb ⟷ **Don't Cooperate** ⟷ Software prefetch is tedious

Spatial locality helps everyone!



But kNN graphs don't use it.

Solution: Put connected nodes next to each other

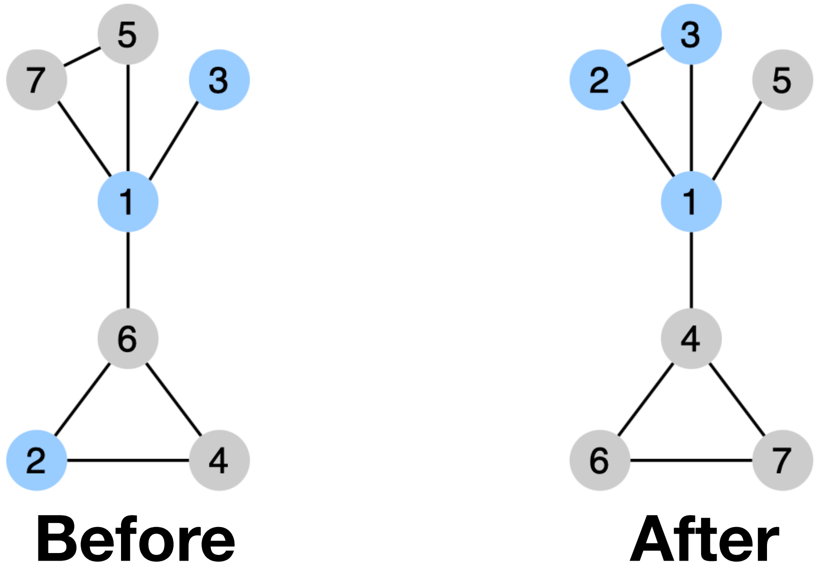
How? Analyze breadth-first search under the ideal cache model

Proxy task for beam search

Cost = # cache misses

Optimize a “node label vector” for cost

- P[node] → memory location



Optimization Objective (NP hard)

"overlap" score

$$\arg\min_{P \in \mathcal{P}} F(P)$$

P[node] = label

All label permutations

[Literature]: 3-4 somewhat-arbitrary choices of F(P)

[Our work]: What is the best objective for KNN search?

Cache efficiency looks *suspiciously* similar to the G-Order objective [1]

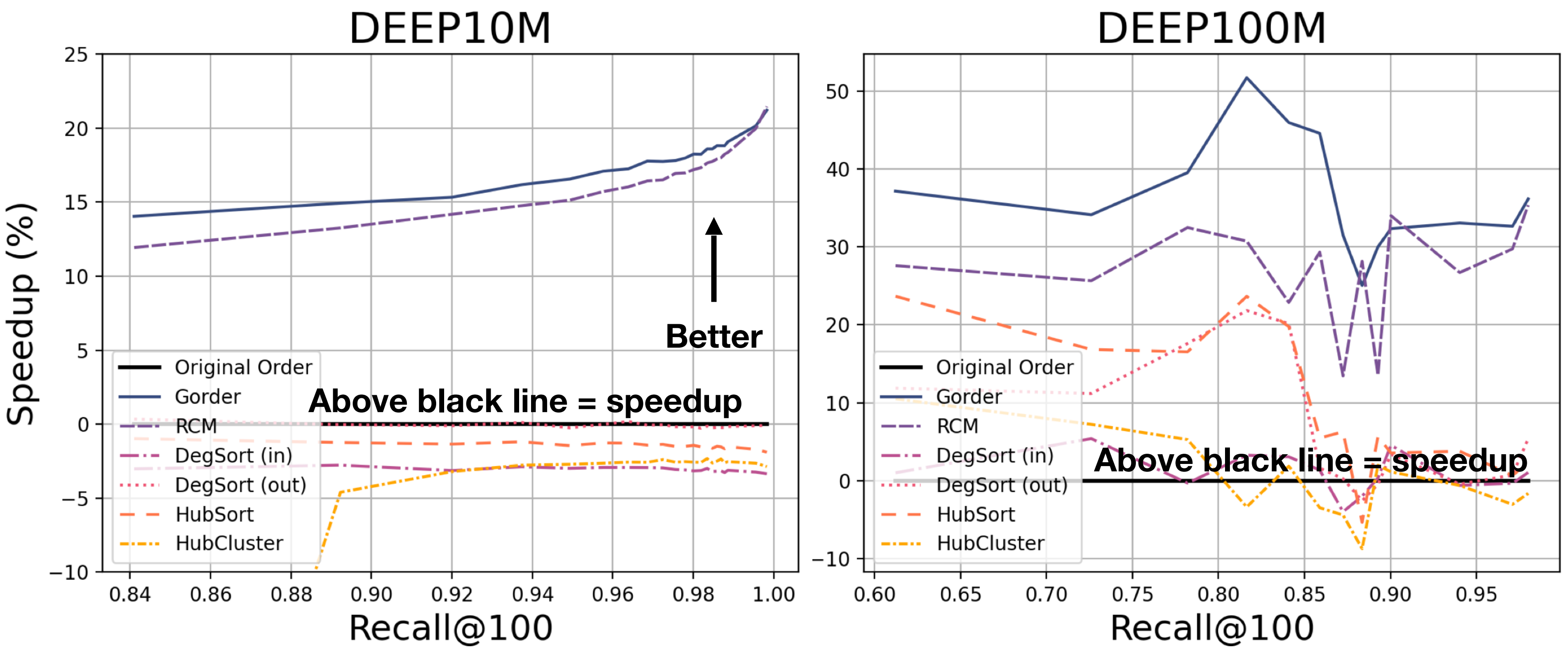
*See our NeurIPS paper for full analysis - there are provable benefits

$$\mathrm{CE} = \sum_{\substack{\text{nodes } u,v \\ u,v \in \text{same line}}} \#(\text{links } u,v) + \#(\text{parents } u,v) - f(u,v)$$

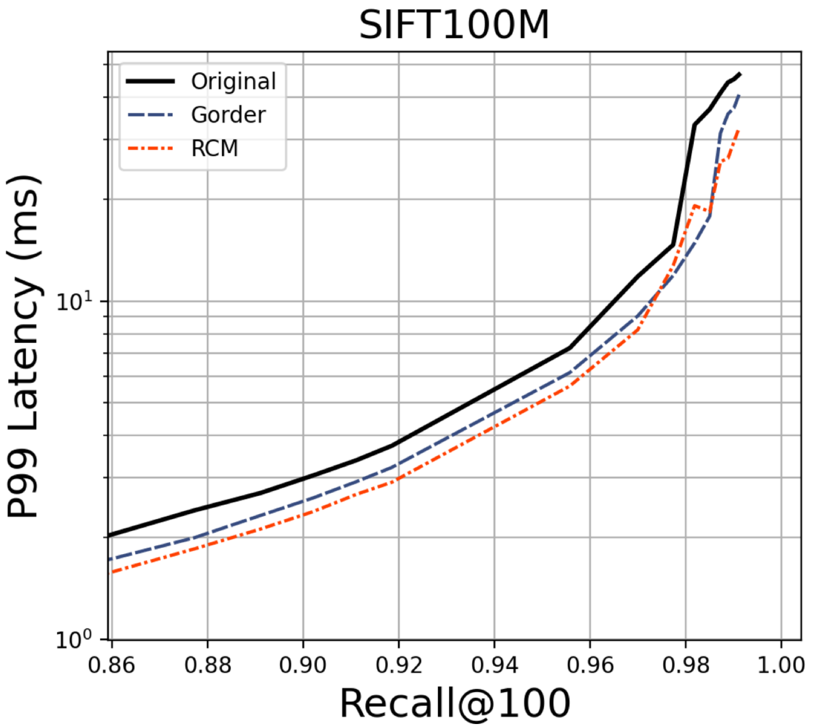
$$\mathrm{GO} = \sum_{\substack{\text{nodes } u,v \\ |P(u) - P(v)| < w}} \#(\text{links } u,v) + \#(\text{parents } u,v)$$

[1] "*Speedup graph processing by graph ordering,*" Wei et. al.

Reduces query time by 10-40%



Reduces 99th percentile latency by 20%



Better

Reduces cache misses by 50%

Cache Misses (lower = better)

Algorithm	L1 (%)	L2 (%)	L3 (%)	TLB (%)
Original	19.53	13.9	6.5	3.85
RCM	17.37	**7.61**	5.1	2.56
Gorder	**14.46**	9.6	**4.0**	**2.14**

We strongly believe in these tricks.

[1] Bioinformatics researchers have already removed the hierarchy from their genome search index and seen -50% query latency and -30% memory.

[2] The DiskANN team saw -25% query latency from our NeurIPS paper.

[3] The Lucene team saw -20% query latency and compression benefits from graph layout (we believe this is independent of our work - great to see!)

[1]: alphaxiv.org/abs/2412.01940
[2]: "*OOD-DiskANN: Efficient and Scalable Graph ANNS for Out-of-Distribution Queries*"
[3]: github.com/apache/lucene/pull/13683

Current FlatNav Integrations

[1] PyTerrier already integrated a vector search retriever based on the `flatnav` python library

https://pyterrier.readthedocs.io/en/latest/ext/pyterrier-dr/indexing-retrieval.html#pyterrier_dr.FlexIndex.flatnav_retriever

```
flatnav_retriever(k=32, *, ef_search=100, num_initializations=100,
    ef_construction=100, threads=16, num_results=1000, cache=True, qbatch=64,
    drop_query_vec=False, verbose=False)
```

Returns a retriever that searchers over a flatnav index.

RETURN TYPE:

`Transformer`

PARAMETERS:

- **k** (*int*) – the maximum number of edges per document in the index
- **ef_search** (*int*) – the size of the list during searches. Higher values are slower but more accurate.
- **num_initializations** (*int*) – the number of random initializations to use during search.
- **ef_construction** (*int*) – the size of the list during graph construction. Higher values are slower but more accurate.
- **threads** (*int*) – the number of threads to use
- **num_results** (*int*) – the number of results to return per query
- **cache** (*bool*) – whether to cache the index to disk
- **qbatch** (*int*) – the number of queries to search at once
- **drop_query_vec** (*bool*) – whether to drop the query_vec column after retrieval
- **verbose** (*bool*) – whether to show progress bars

Added in version 0.4.0.

Changed in version 0.4.1: fixed bug with *num_initializations*

Note

This transformer requires the `flatnav` package to be installed. Instructions are available in the flatnav repository.

Citation

Munyampirwa et al. Down with the Hierarchy: The 'H' in HNSW Stands for "Hubs". arXiv 2024. [link]

Do you want to try this stuff out?

Our reference implementation at flatnav.net has everything:

- Performance parity with HNSW, without hierarchy
- Implementations for the best reordering methods
- Codebase in C++17 (Header-only library)
- Easy-to-use Python bindings