Spring25 CS598YP

11.2 ACORN

Yongjoo Park

University of Illinois at Urbana-Champaign

Recap: HNSW and DiskANN

HNSW: Greedy Search

- Start from the top layer
- For each layer
  - find the node closest to **the query**

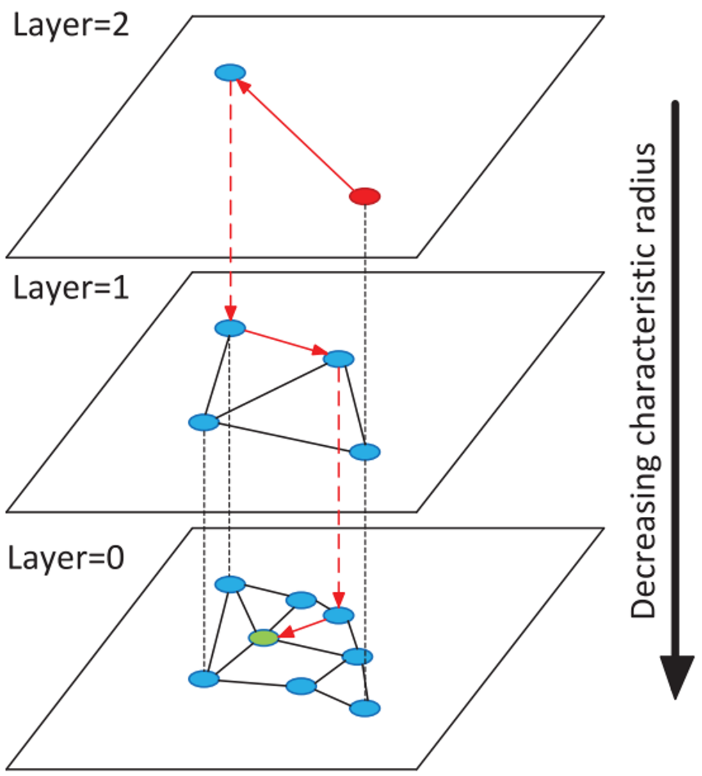
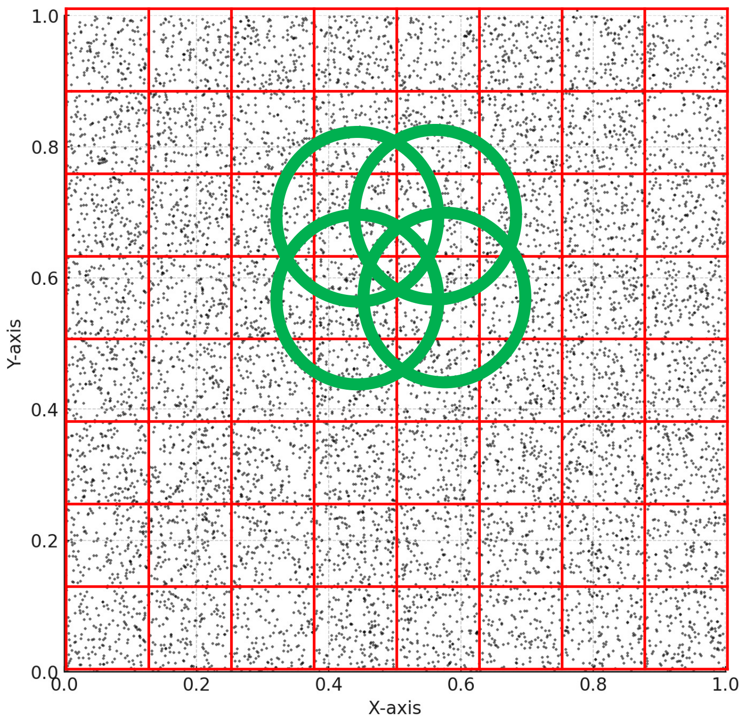


Fig. 1. Illustration of the Hierarchical NSW idea. The search starts from an element from the top layer (shown red). Red arrows show direction of the greedy algorithm from the entry point to the query (shown green).

DiskANN: Overlapping partitioning strategy



A data point belongs to *l* centroids

Each cluster is now extended

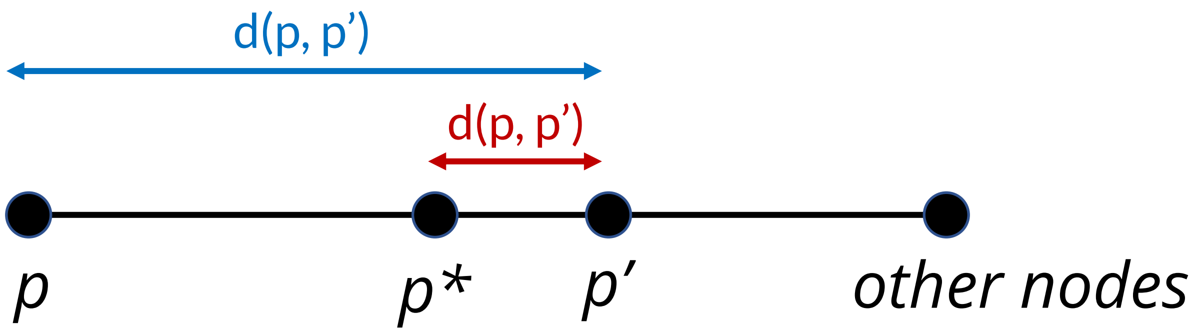
Clusters overlap

Expected size of a cluster?

(*N* points, *k* clusters, *l* overlaps)

Index each cluster separately; then union them

Vamana: RobustPrune



- We want to connect to *p’*, only if it is not too close to *p**, relative to *p*
- d(p*, p’) should *not* be too small compared to d(p, p’)
- d(p*, p’) is too small if d(p*, p’) <= (1/**\alpha)** * d(p, p’) (e.g., \alpha = 2)
- Then, p’ is pruned

ACORN

Filter while searching

slower construction

good retrieval (for not-so-slow selectivity prediactes)

Problem: Filtered Vector Search

- Find (the most) similar items that satisfy a *predicate*
  - Query: “World-wide Caves”
  - ***(hard)*** Predicate: “kid-safe”

NeurIPS'23 Competition Track: Big-ANN

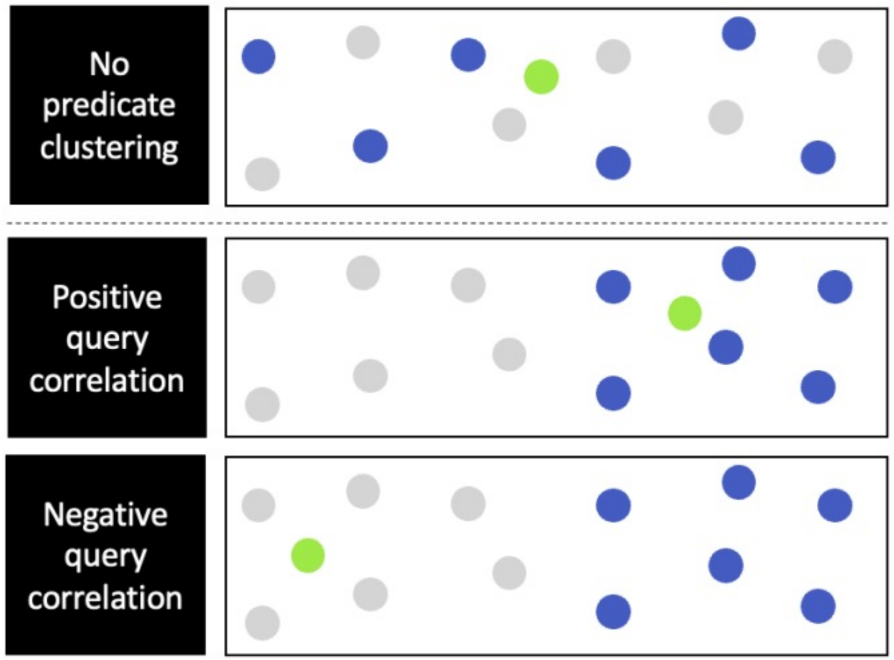
Supported by Microsoft Pinecone aws zilliz

New: the latest ongoing leaderboard has been released (March 1st, 2024).
Top entries:

Filter track			OOD track			Sparse track		
Rank	Algorithm	QPS@90% recall	Rank	Algorithm	QPS@90% recall	Rank	Algorithm	QPS@90% recall
1	Pinecone-filter	85,491	1	Pinecone-ood	38,088	1	Zilliz	10,749
2	Zilliz	84,596	2	Zilliz	33,241	2	Pinecone_smips	10,440
3	ParlayANN IVF^2	37,902	3	RoarANN	22,555	3	PyANNS	8,732
4	Puck	19,193	4	PyANNS	22,296	4	shnsw	7,137
...	...	...	...	...	...	...	...	...
Baseline	FAISS	3,032	Baseline	Diskann	4,133	Baseline	Linscan	93

Filtered Search: Naïve Approaches

- Pre-filter: Brute-force search is too slow
- Post-filter: Can be very slow for “Negative query correlation”



Filtered Search: NHQ (NeurIPS'23)

- Combine encoding vector and attributes
- Encoding vector = `[2.12, 0.12, 3.21, -0.22, … ]`
- Attribute vector = `{kids-safe, K-drama, music, …}`

Final vector:

`[2.12, 0.12, 3.21, -0.22, …, 1, 0, 0, 0, 1, …]`

An Efficient and Robust Framework for Approximate Nearest Neighbor Search with Attribute Constraint

Mengzhao Wang
Hangzhou Dianzi University
wmzssy@yeah.net

Lingwei Lv
Hangzhou Dianzi University
llw@hdu.edu.cn

Xiaoliang Xu*
Hangzhou Dianzi University
xxl@hdu.edu.cn

Yuxiang Wang
Hangzhou Dianzi University
lsswyx@hdu.edu.cn

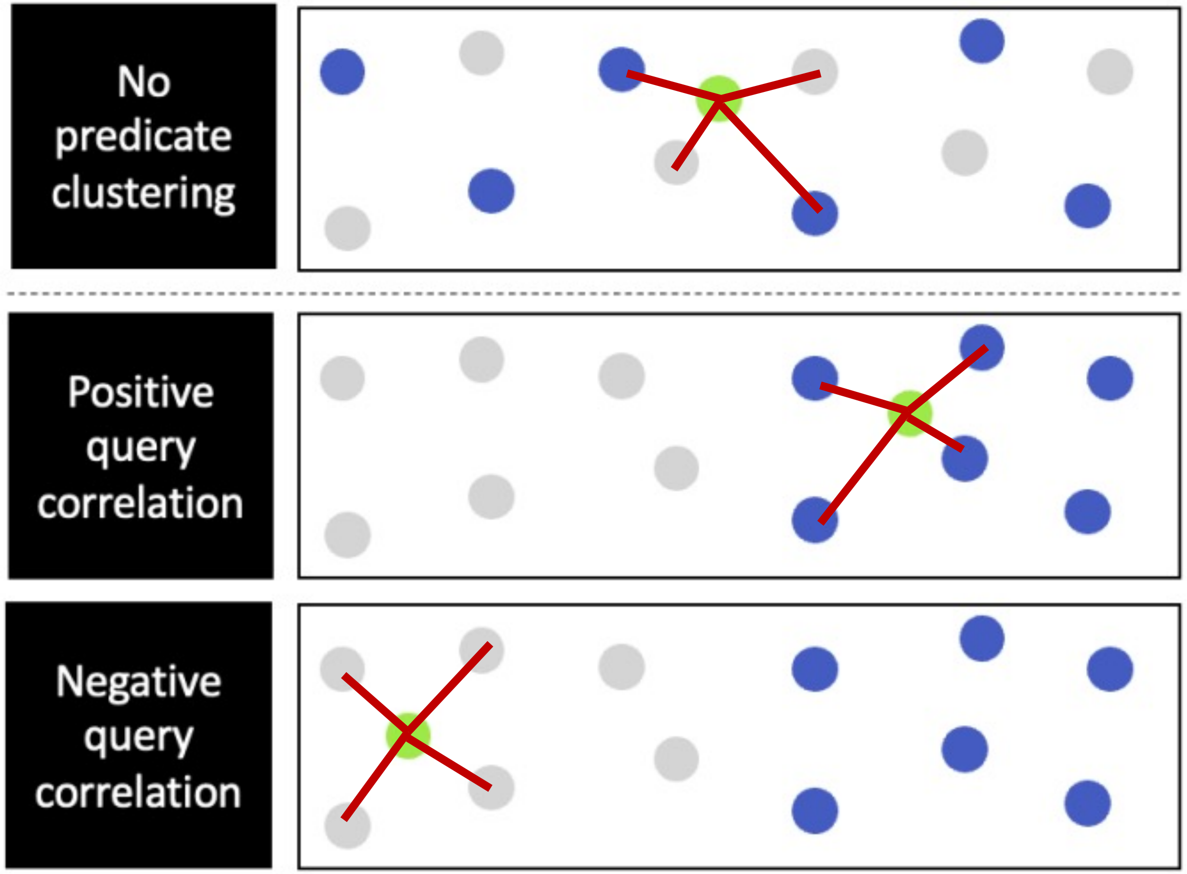
Qiang Yue
Hangzhou Dianzi University
yq@hdu.edu.cn

Jiongkang Ni
Hangzhou Dianzi University
hananoyuuki@hdu.edu.cn

Abstract

This paper introduces an efficient and robust framework for hybrid query (HQ) processing, which combines approximate nearest neighbor search (ANNS) with attribute constraint. HQ aims to find objects that are similar to a feature vector and match some structured attributes. Existing methods handle ANNS and attribute filtering separately, leading to inefficiency and inaccuracy. Our framework, called native hybrid query (NHQ), builds a composite index based on proximity graph (PG) and applies joint pruning for HQ. We can easily adapt existing PGs to this framework for efficient HQ processing. We also propose two new navigable PGs (NPGs) with optimized edge selection and routing, which improve the overall ANNS performance. We implement five HQ methods based on the proposed NPGs and existing PGs in NHQ, and show that they outperform the state-of-the-art methods on 10 real-world datasets (up to 315× faster with the same accuracy).

Basic Idea: Evaluate while traversing



- Potential issue: Some neighbors will be filtered out
- Solution: Increase neighbor size (?!!)

Additional Parameters

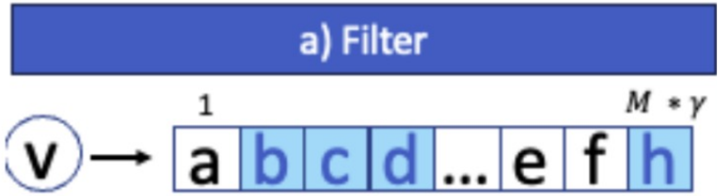
- γ: neighbor list **expansion** factor

Reminder

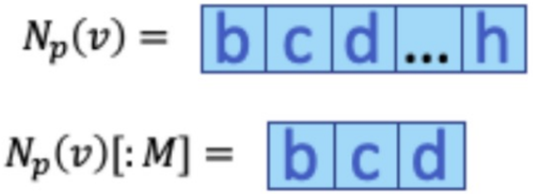
- *efc* ($ef_{construction}$): HNSW's construction parameter
- *M*: HNSW's construction parameter

ACORN Search (version 1)

Hybrid Search Neighbor Selection Strategies



Construction time increases significantly



Search $M \cdot \gamma$ neighbors
Many neighbors -> Expensive

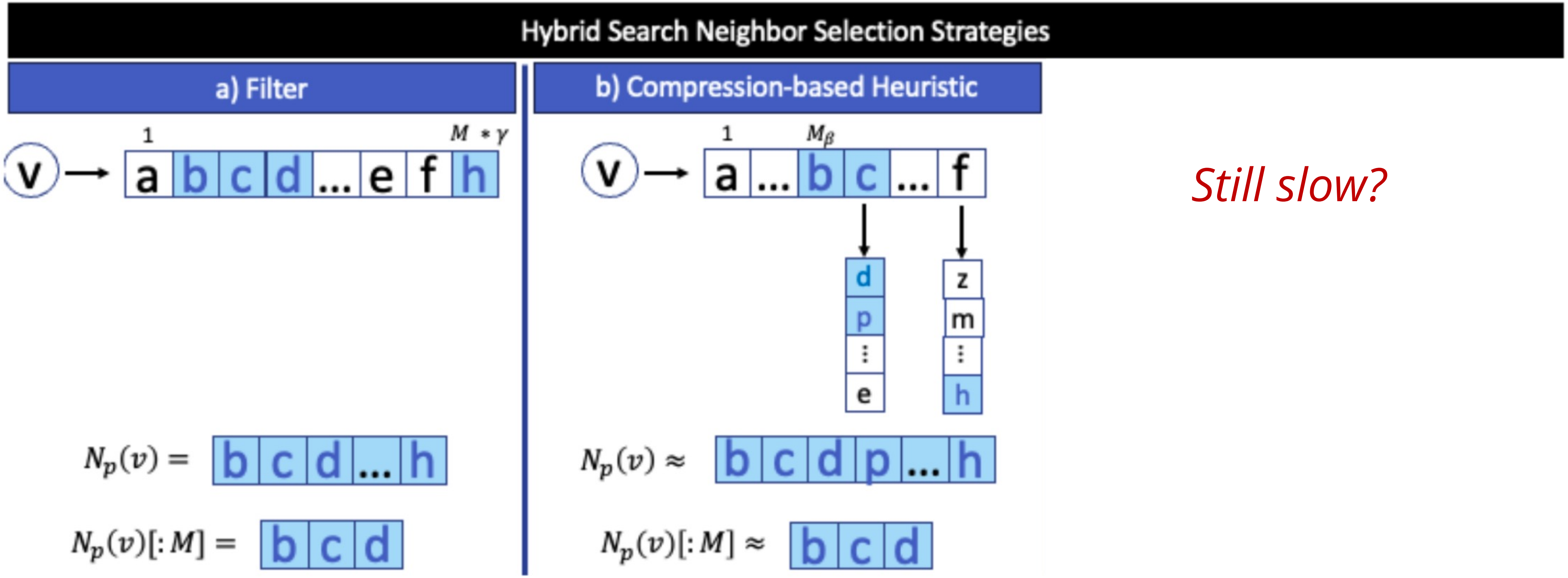
Additional Parameters

- γ: neighbor list **expansion** factor
- M_{β}: compression factor

Reminder

- *efc* ($ef_{construction}$): HNSW’s construction parameter
- M: HNSW’s construction parameter

ACORN Search (version 2)

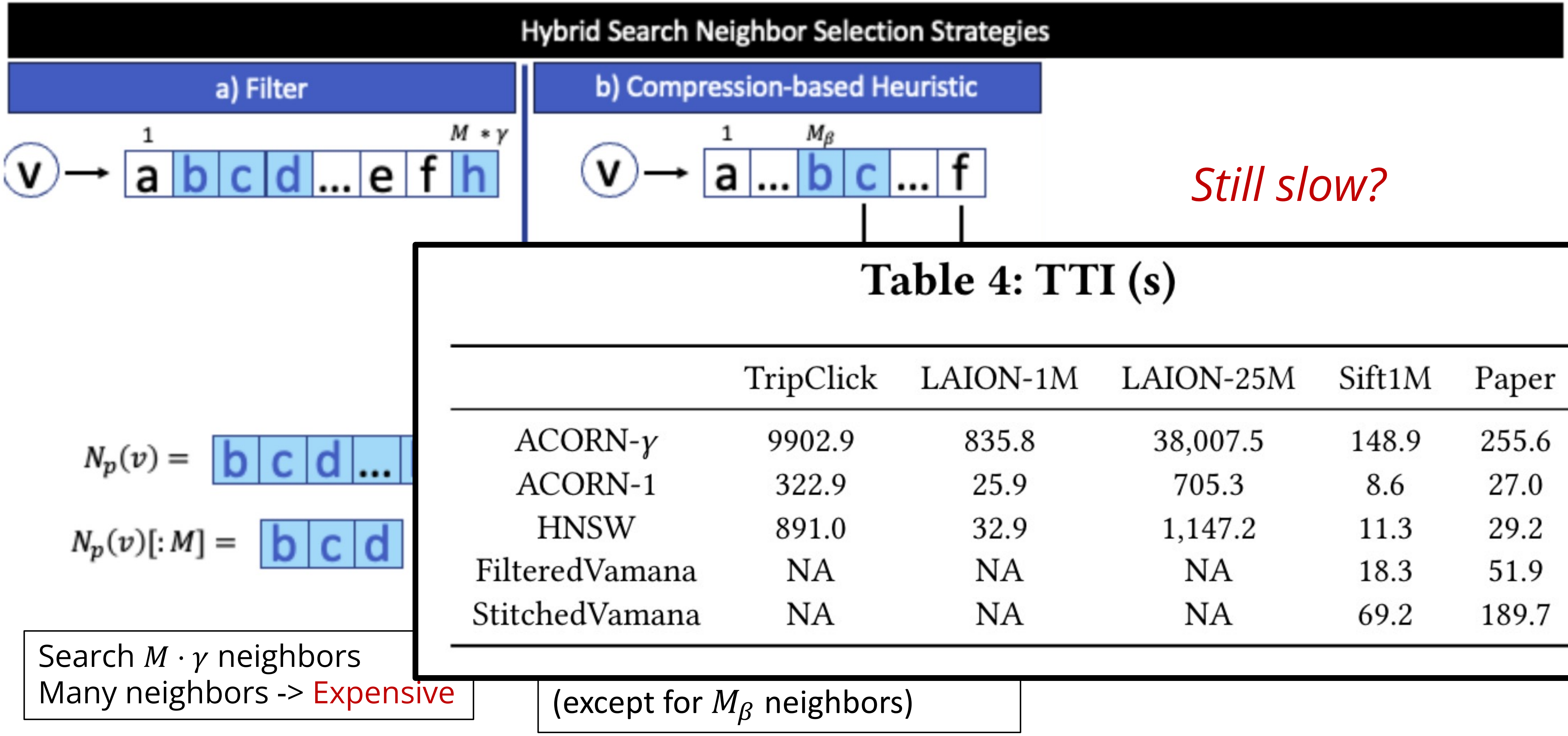


Still slow?

Search $M \cdot \gamma$ neighbors
Many neighbors -> Expensive

Resort to query-time expansion (except for M_β neighbors)

ACORN Search (version 2)

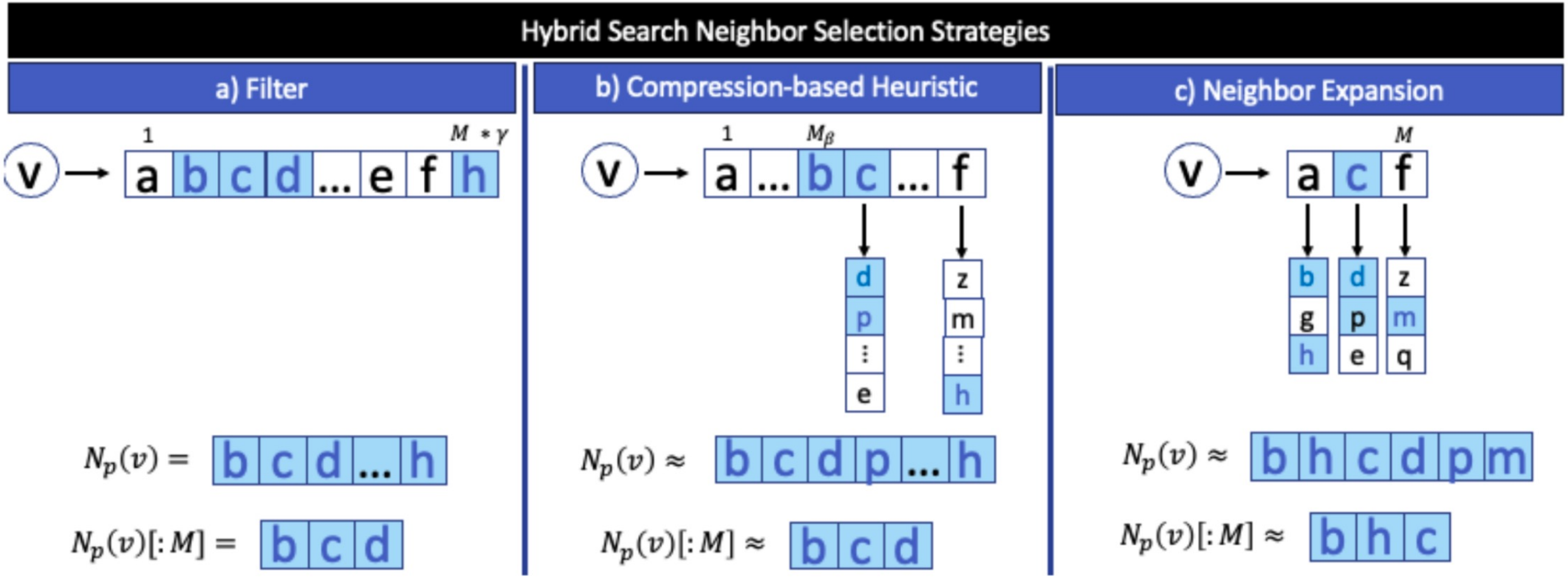


Still slow?

Table 4: TTI (s)

	TripClick	LAION-1M	LAION-25M	Sift1M	Paper
ACORN-γ	9902.9	835.8	38,007.5	148.9	255.6
ACORN-1	322.9	25.9	705.3	8.6	27.0
HNSW	891.0	32.9	1,147.2	11.3	29.2
FilteredVamana	NA	NA	NA	18.3	51.9
StitchedVamana	NA	NA	NA	69.2	189.7

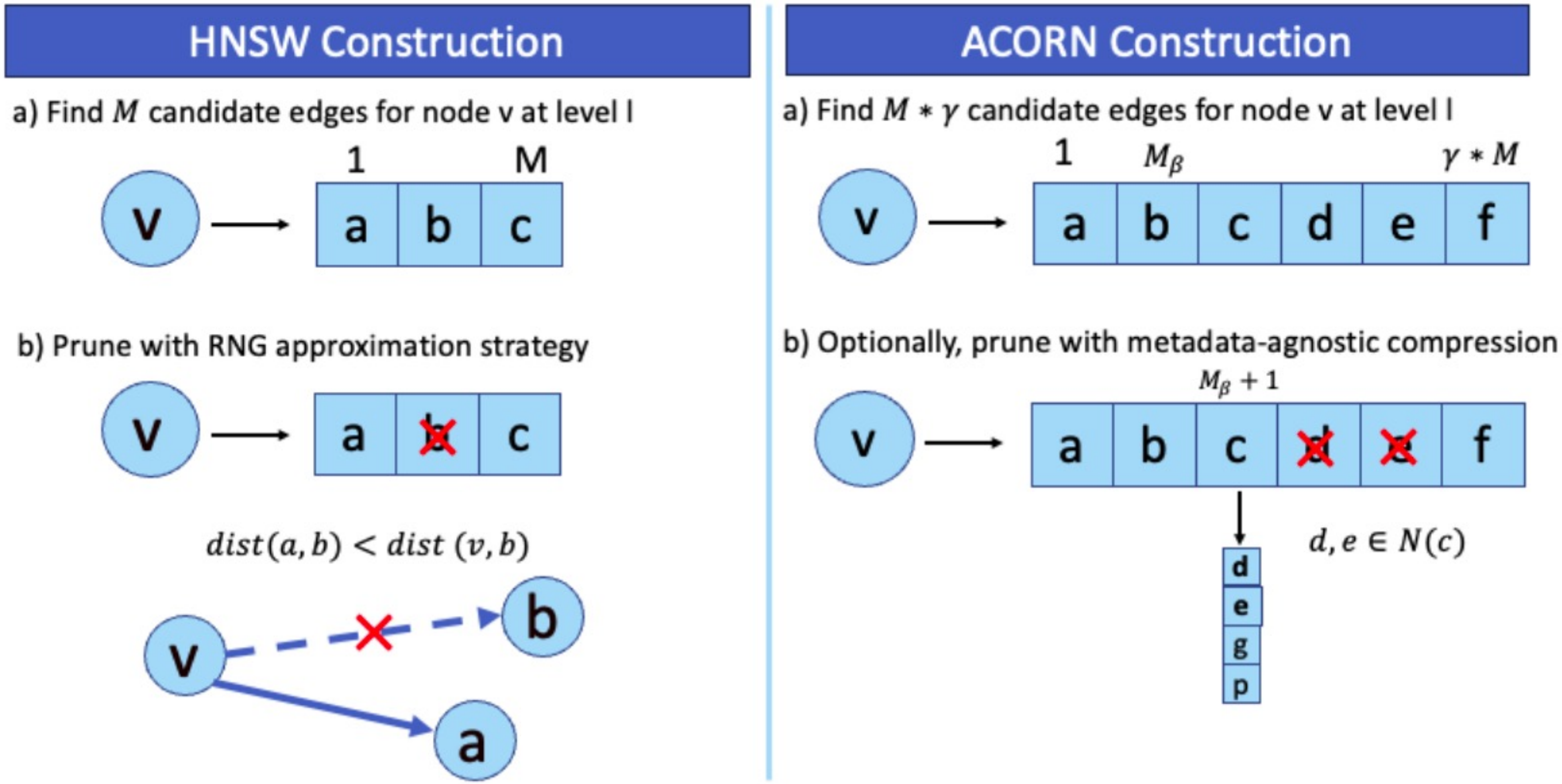
ACORN Search (version 3)



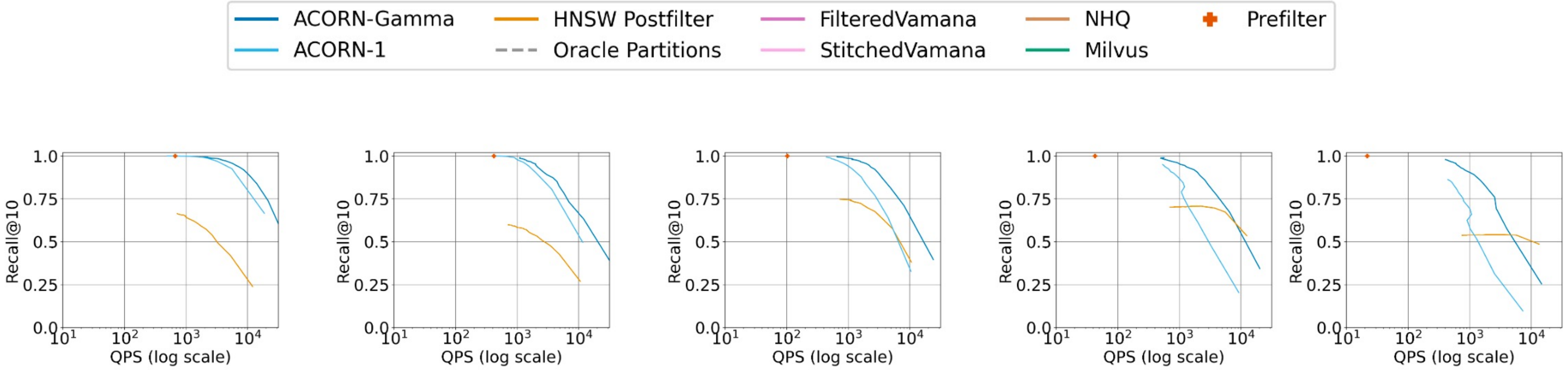
Search $M \cdot \gamma$ neighbors
Many neighbors -> Expensive

Resort to query-time expansion
(except for M_β neighbors)

ACORN Construction



Empirical results



(a) 1p Sel (s=0.0127) (b) 25p Sel (s=0.0485) (c) 50p Sel (s=0.1215) (d) 75p Sel (s=0.2529) (e) 99p Sel (s=0.6164)

Figure 9: Recall@10 vs QPS for Varied Selectivity Query Filters on TripClick

Summary

- ACORN performs filtering while traversing
- Technically, retrieval on an *induced* graph
  - some nodes are pruned dynamically
  - neighbor is expanded
- The induced graph becomes ***too sparse*** for low selectivity filters
- Resort to *brute-force* if likely to be too sparse

Questions?