Martin Aumüller
Tobias Christiani
Rasmus Pagh
Michael Vesterli

PUFFINN

Parameterless and Universally Fast FInding of Nearest Neighbors

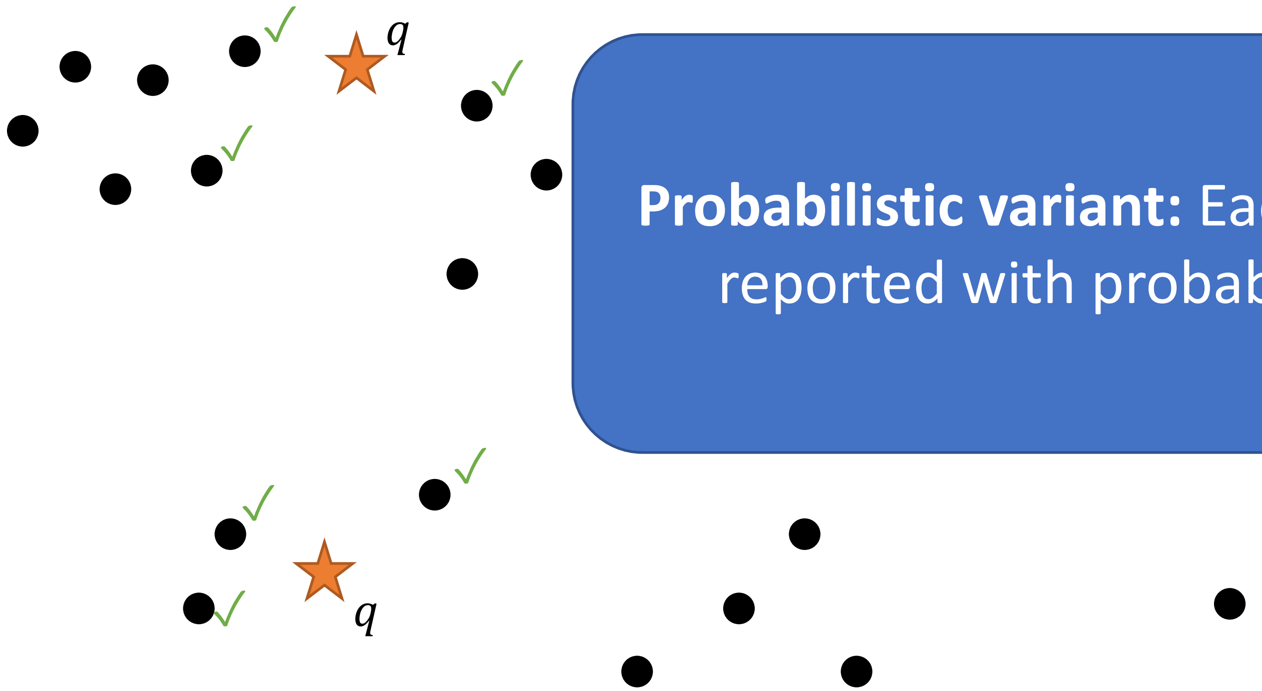
European Research Council
Established by the European Commission
Supporting top researchers from anywhere in the world

IT UNIVERSITY OF COPENHAGEN

BASIC ALGORITHMS RESEARCH COPENHAGEN
BARC!

k-Nearest Neighbor Problem

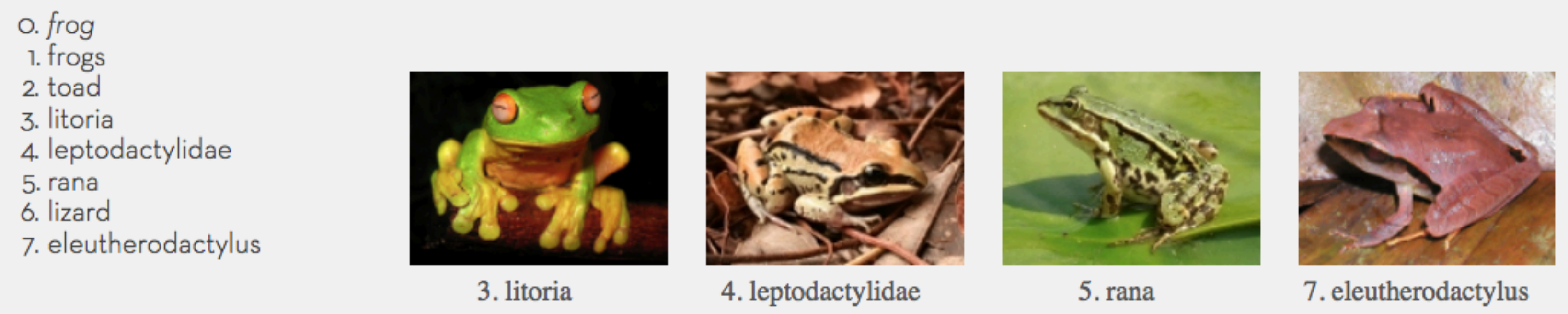
- **Preprocessing**: Build DS for set $S \subseteq \mathbb{R}^d$ of n data points
- **Task**: Given query point $q \in \mathbb{R}$, return k closest points to q in S



Probabilistic variant: Each nearest neighbor is reported with probability at least $1 - \delta$.

Nearest neighbor search on words

- GloVe: learning algorithm to find vector representations for words
- *GloVe.twitter* dataset: **1.2M words**, vectors trained from **2B tweets**, **100 dimensions**
- Semantically similar words: nearest neighbor search on vectors



https://nlp.stanford.edu/projects/glove/

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

GloVe Examples (100d, 1.2M vectors)

“munich”	“germany”	“algorithm”
• “bayern”	• “austria”	• “algorithms”
• “cologne”	• “switzerland”	• “optimization”
• “stuttgart”	• “german”	• “approximation”
• “berlin”	• “europe”	• “iterative”
• “hamburg”	• “poland”	• “computation”

Our Results

Theory

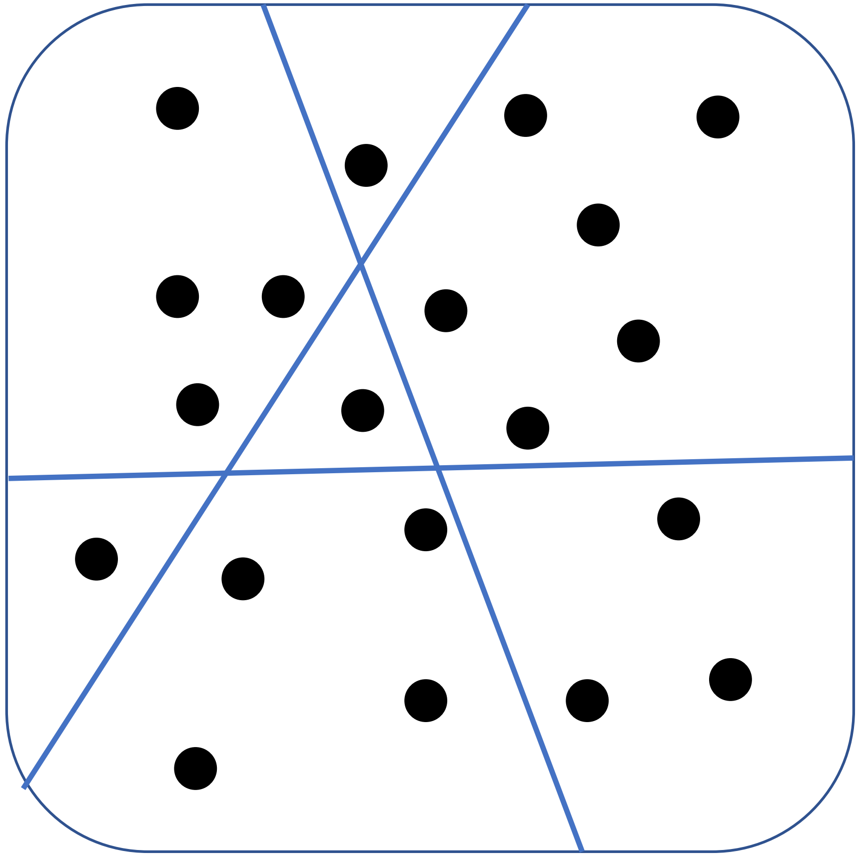
- A novel Locality-Sensitive Hashing (LSH)-based algorithm for probabilistic k-NN
- Avoids standard reduction approach by [Har-Peled et al., 2012]

Practice

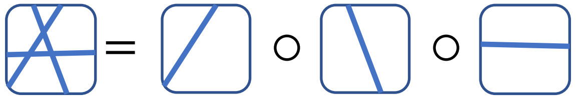
- Theory + algorithm engineering gives a fast implementation with provable guarantees
- in our experiments:
  - competitive with other state-of-the-art approaches (w/o guarantees)
  - faster than state-of-the-art LSH (w/o guarantees) [FALCONN]

How does it work?

Locality-Sensitive Hashing (LSH) [Indyk-Motwani, 1998]

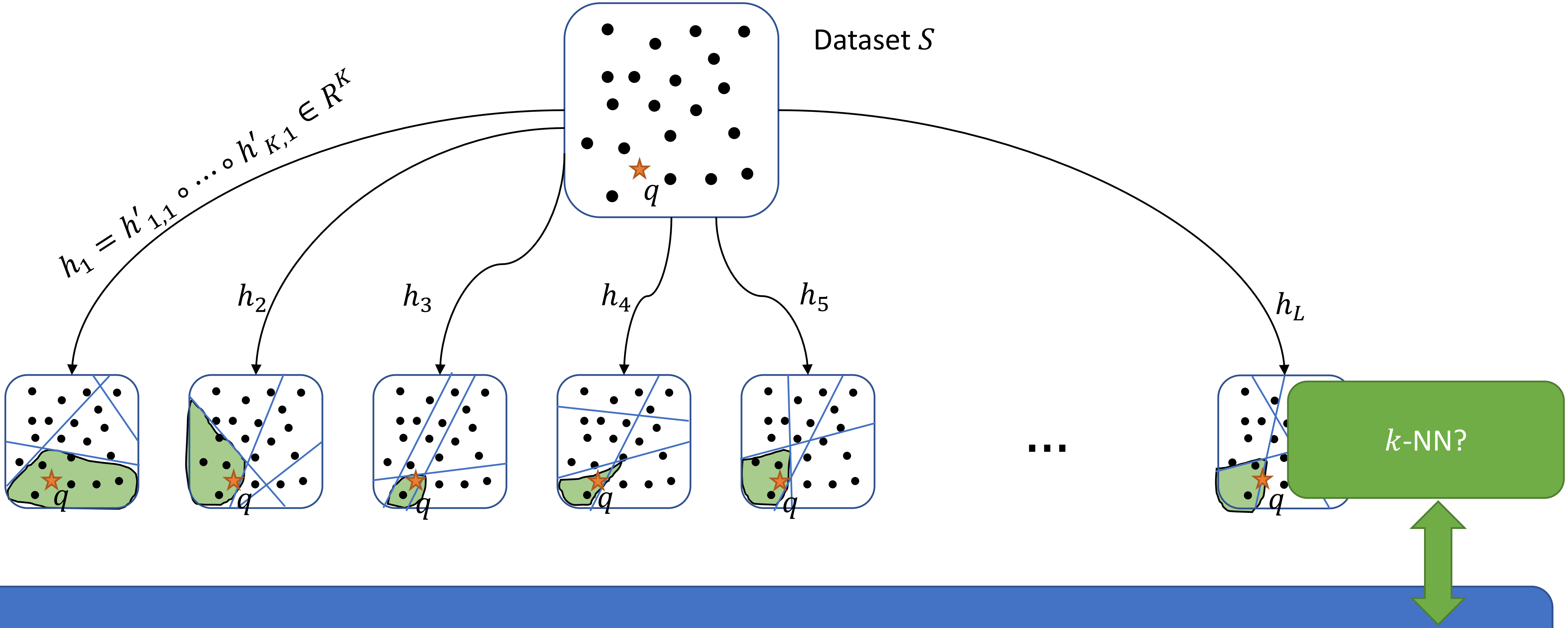


$$h(p) = h_1(p) \circ h_2(p) \circ h_3(p) \in \{0,1\}^3$$



A family $\mathcal{H}$ of hash functions is **locality-sensitive**, if the collision probability of two points is decreasing with their distance to each other.

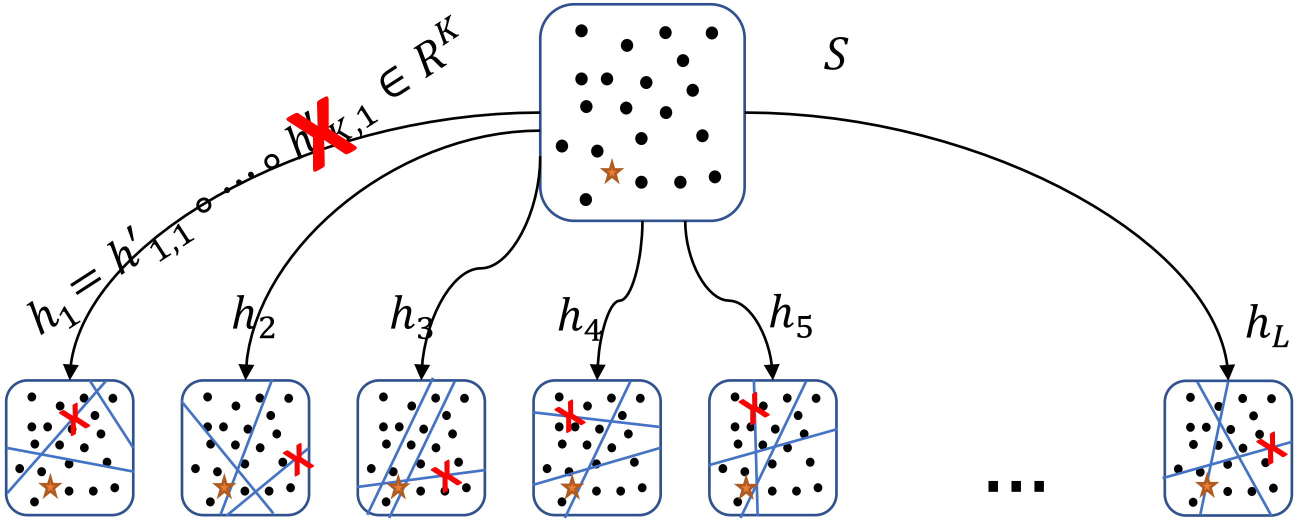
Standard LSH for Reporting Points at Distance $\leq r$



Query Algorithm: Collect all points that collide with q under $h_1, \dots, h_L$. Return all points at distance $\leq r$.

Our Approach: Solving k-NN using LSH

- Check buckets $j \in \{1, \dots, L\}$, one-by-one
- keep track of **current** k closest points
- **Goal**:
  Report with prob. $\geq 1 - \delta$



Termination: If $(1-p)^j \leq \delta$, report **current** top-k.

probability of the **current** k-th nearest neighbor to collide.

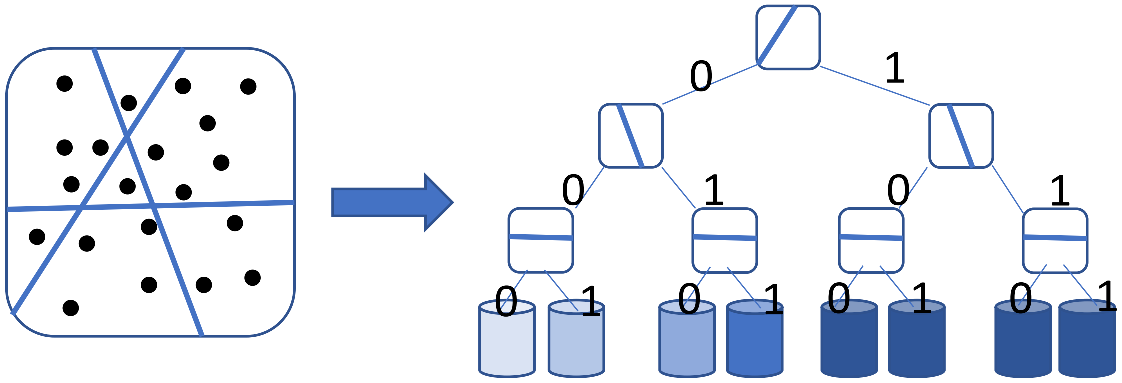
- What if there is no such j?
  - Try again with smaller K

Why does that work? **Monotonicity** of the LSH collision prob.

The Data Structure

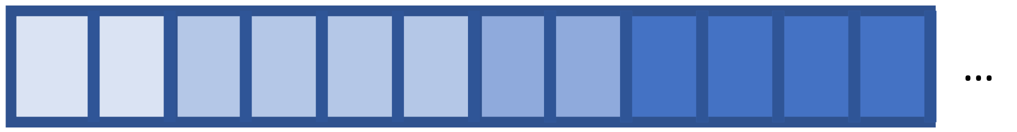
Theoretical

- **LSH Forest:** Each repetition is a Trie build from LSH hash values [Bawa et al., 2005]



Practical

- Store indices of data set points sorted by hash code
- ”Traversing the Trie” by binary search
- use lookup table for first levels



Works with any kind of LSH

PUFFINN

Parameterless and Universally Fast FInding of Nearest Neighbors

“space parameter” + “quality parameter”

no **internal** parameters

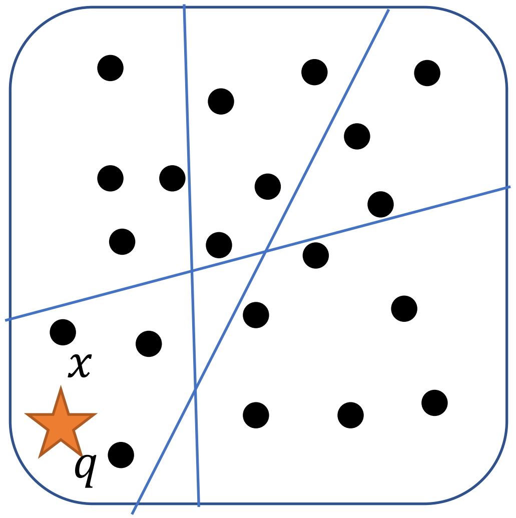
Implicit Tries + Recycling LSH values + ???

[Christiani, 2019]

Sketching to avoid distance computations

SimHash [Charikar, 2002]
1-BitMinHash [König-Li, 2010]

- Have to carry out (expensive) distance computations on candidates
- Can be reduced by storing compact sketch representations



Sketch representation

q 1011100101

x 0101110101

Easy to analyze:
Sum of Bernoulli trials of
$\Pr(X = 1) = f(\text{dist}(q, x))$

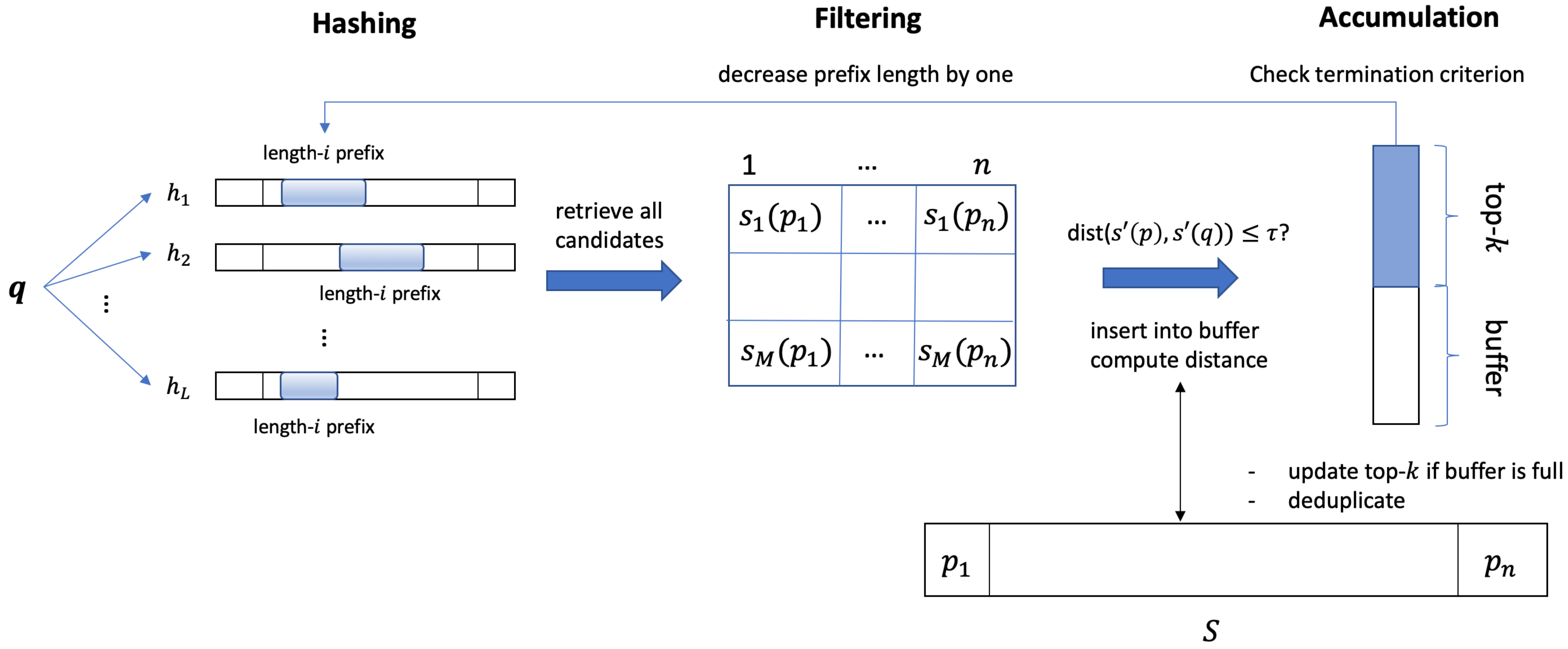
At least τ collisions?

Yes → compute $\text{dist}(q, x)$

No → skip

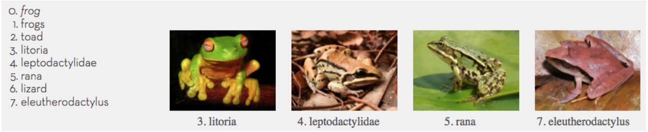
Set τ such that with probability at least $1 - \varepsilon$ we don't disregard point that could be among NN.

Overall System Design

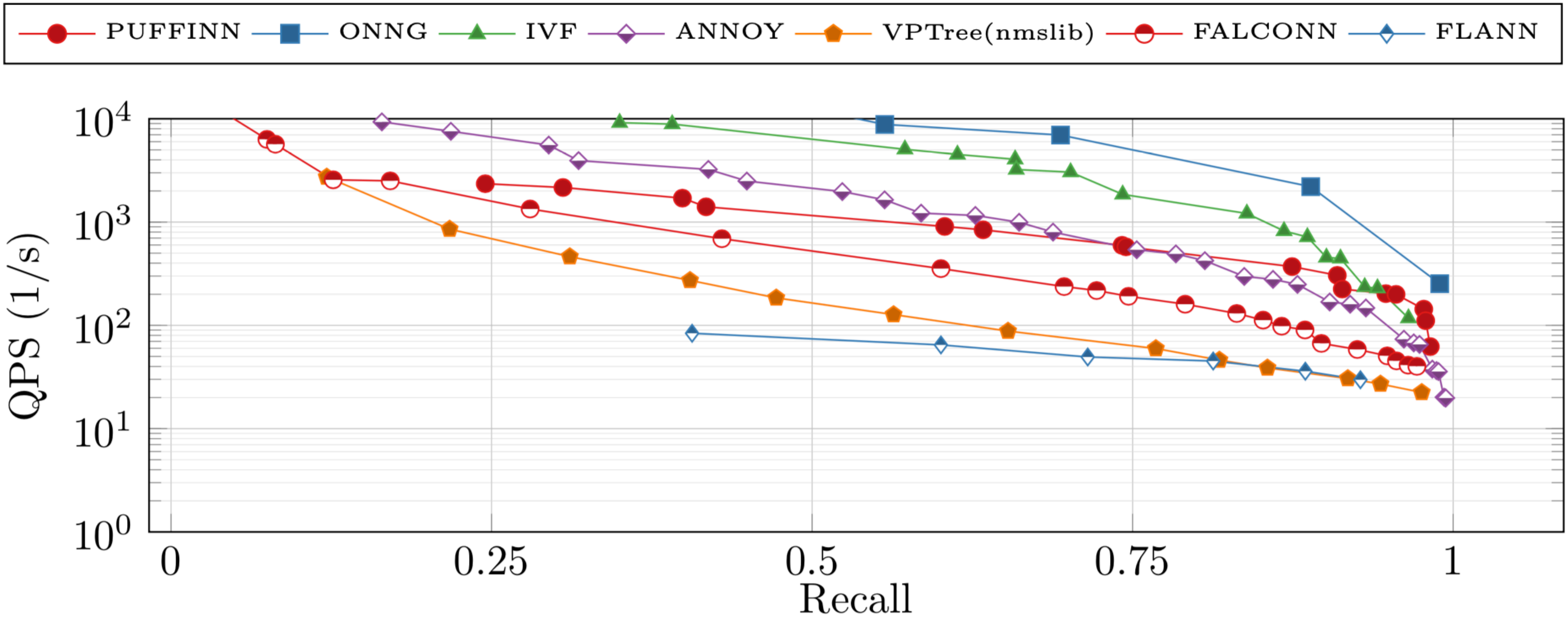


Experimental Evaluation

- Design choices in the implementation
  - Which LSH family to use? Cross-Polytope LSH [Andoni et al., 2015]
  - Which evaluation strategy to use? Pooling
  - Use sketches? Yes
  - Influence of parameters? More space helps, but saturates quickly
- Comparison to other existing k-NN implementations



Running time (Glove 100d, 1.2M, 10-NN)



A difficult (?) data set in $\mathbb{R}^{3d}$

n data points

$$x_1 = (0^d, y_1, z_1)$$

$$\vdots$$

$$x_{n-1} = (0^d, y_{n-1}, z_{n-1})$$
$$x_n = (v, w, 0^d)$$

m query points

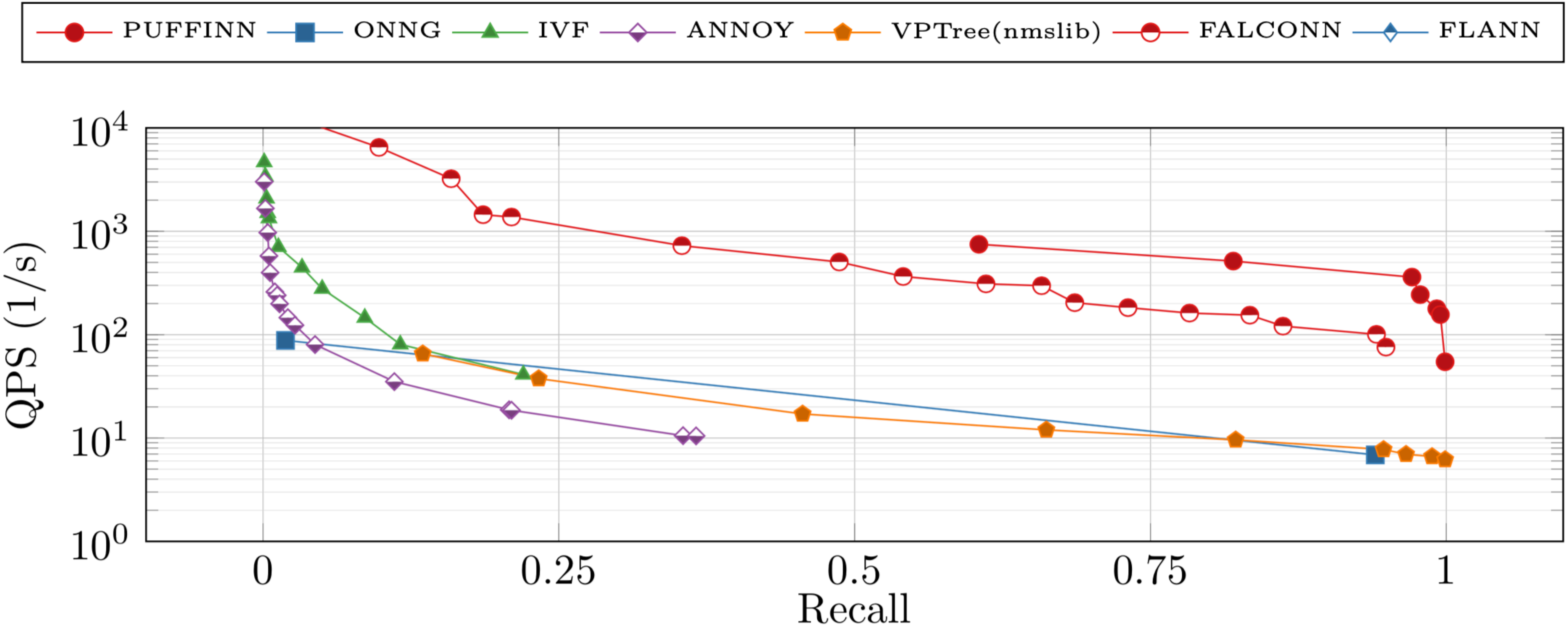
$$q_1 = (v, 0^d, r_1)$$

$$\vdots$$

$$q_m = (v, 0^d, r_m)$$

$$y_i, z_i, v, w, r_i \sim \mathcal{N}^d\left(0, \frac{1}{2d}\right)$$

Running time (“Difficult”, 1M, 10-NN)



Summary

- Using LSH to solve exact k-NN (with probabilistic guarantees)
- Adaptive query algorithm
- Engineering tricks to make it fast (more in the paper!)

- Can ideas be applied to other settings?
  - Similarity Joins

https://github.com/puffinn/puffinn
https://github.com/puffinn/esa-paper

A Bound on the Expected Running Time

- **knows for each query $\boldsymbol{q}$**
  best stopping point in data structure

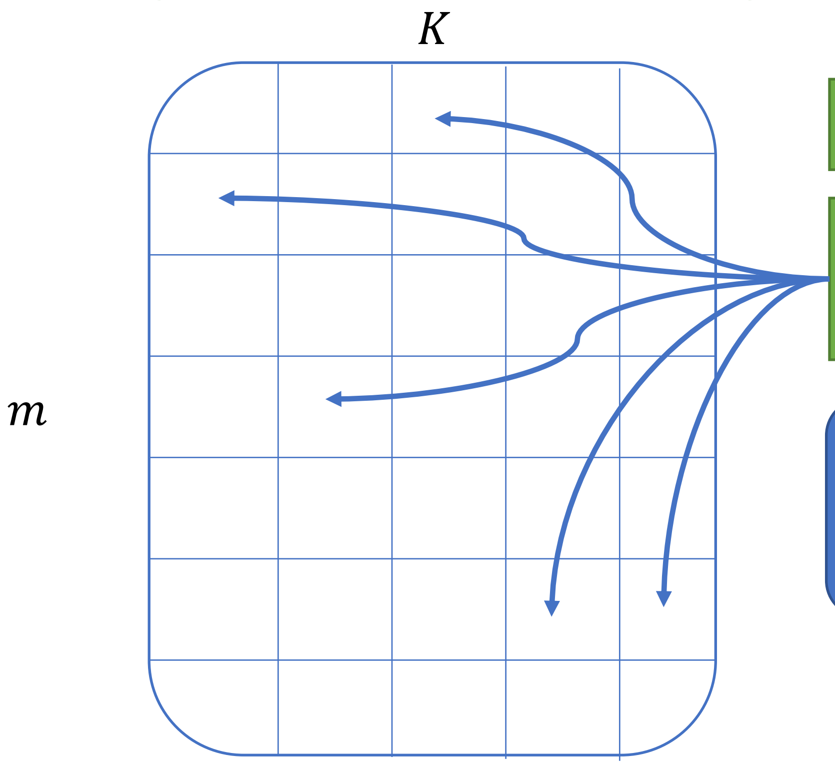
$$OPT(L,K,k,\delta) = \min\left\{ \frac{\ln(1/\delta)}{p(q,x_k)^i}\left(i + \sum_{x\in P} p(q,x)^i\right) \mid 0 \le i \le K, \frac{\ln(1/\delta)}{p(q,x_k)^i} \ge L \right\}$$

- **Lemma**: In expectation, proposed algorithm takes time

$$O(OPT(L,K,k,\delta/k) + L(K+k))$$

Fast Hash Function Evaluation

- **Main Bottleneck**: Computation of Hash Values
- Adapt the "pooling" technique of [Dahlgaard et al., 2017] and [Christiani, 2019]



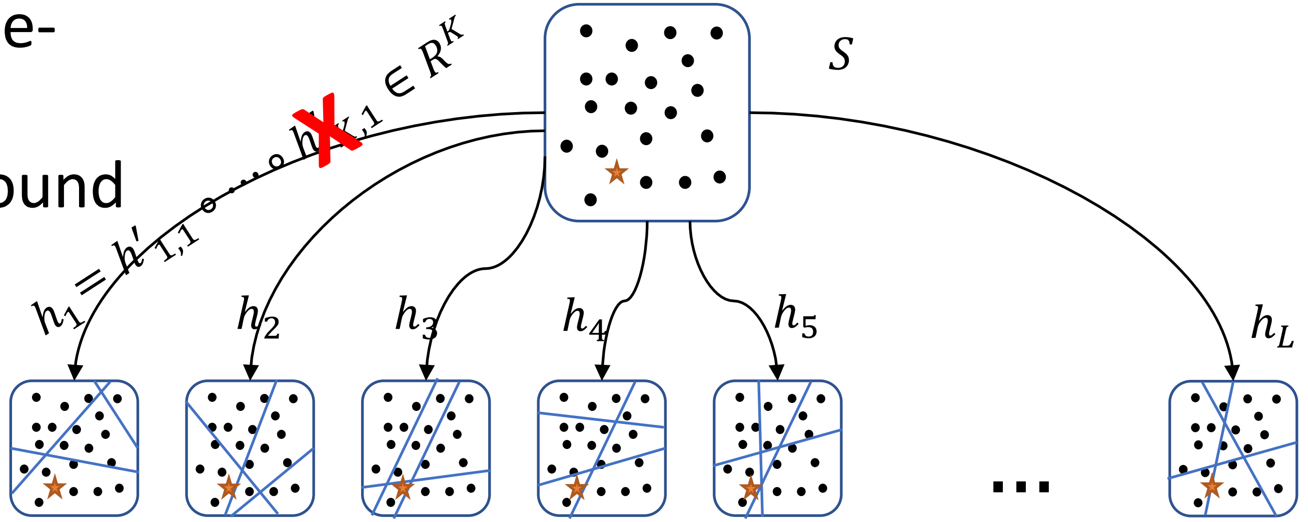
$K \cdot m$ independent hash functions from LSH family, $m \ll L$.

Pick K hash functions in repetition j using universal hash functions in each column.

Analysis using Cantelli's inequality →
Requires different stopping criteria (factor 2 slowdown)

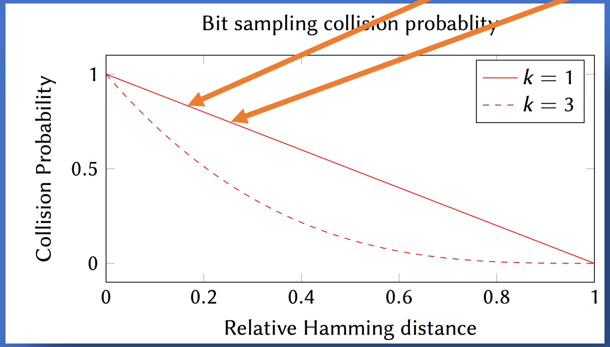
Our Approach: Solving k-NN using LSH

- Check buckets $j \in \{1, \dots, L\}$, one-by-one
- keep track of closest k points found so far



Want: $(1-p^*)^j \leq \delta$,
(p^* collision prob. of true k-NN)

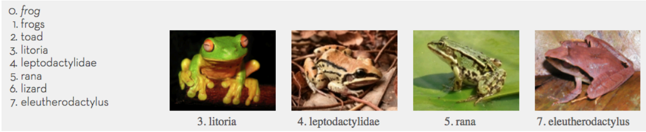
Montonocity of LSH: $p^* \geq p$.



Termination: If $(1-p)^j \leq \delta$, report top-k.

probability of the current k-th nearest neighbor to collide.

Why does that work? Monotonicity of the LSH collision prob.



Influence of Index Size (Glove 100d, 1.2M, 10-NN)

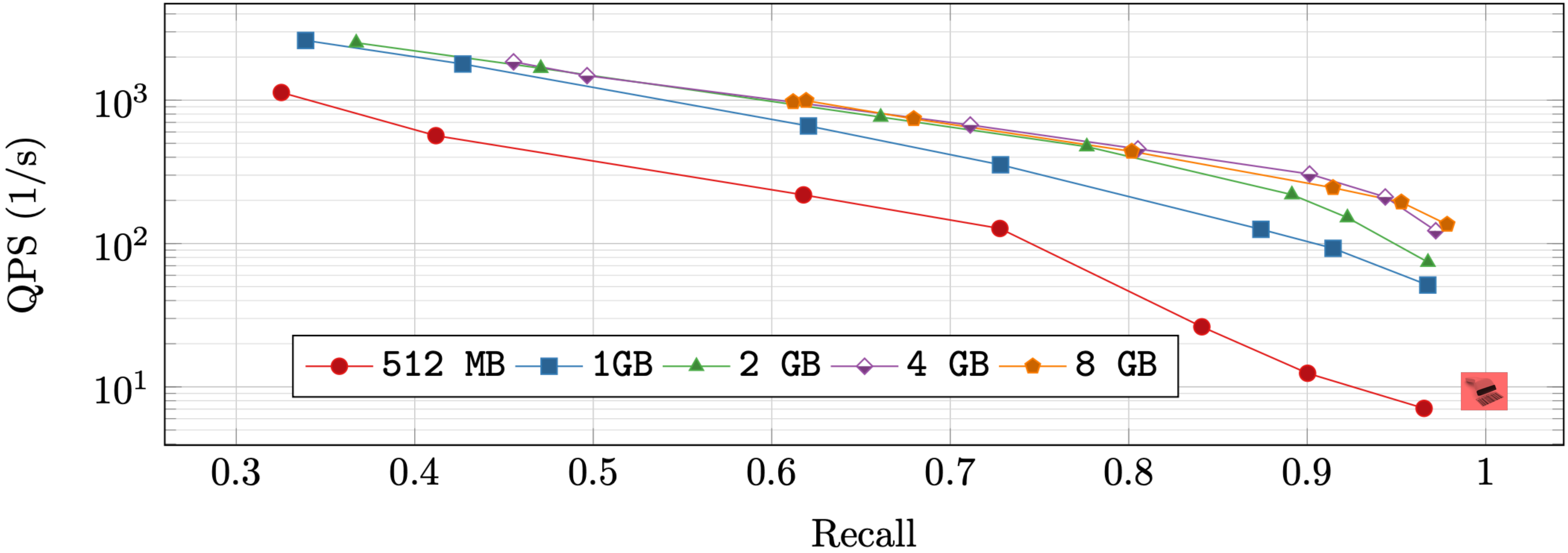


Figure 3 Influence of index size to quality-performance trade-off.