Falconn++: A locality-sensitive filtering approach for approximate nearest neighbor search

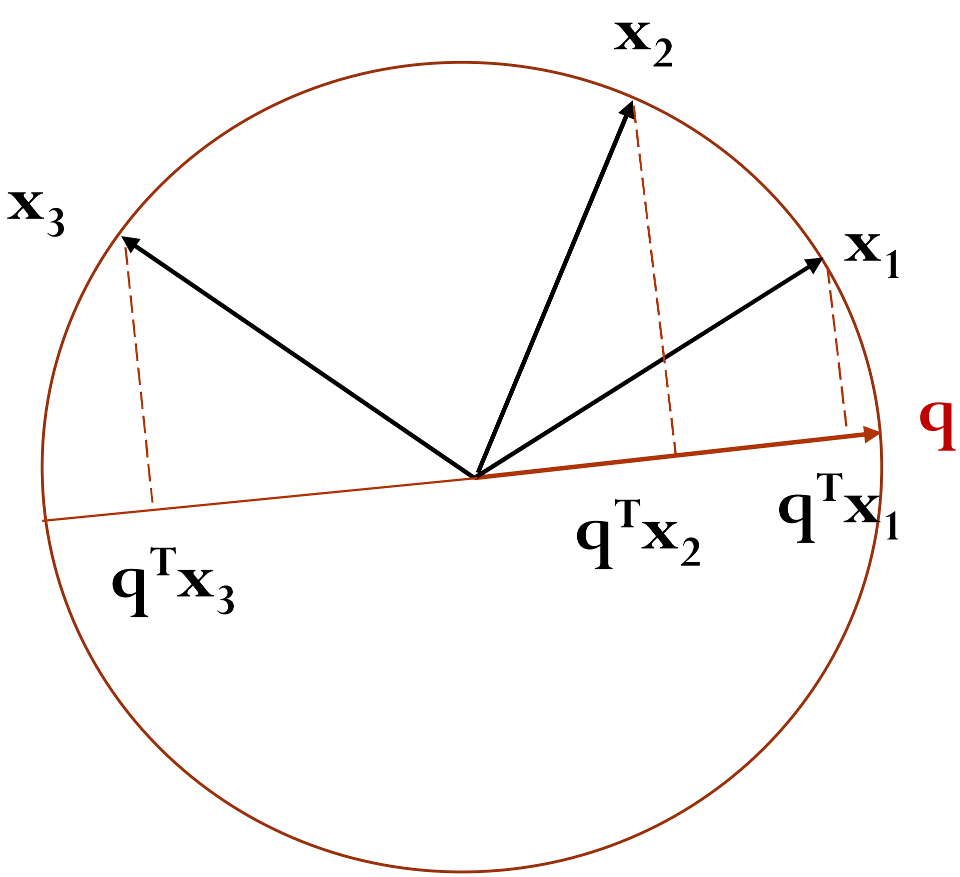
Ninh Pham, Tao Liu

University of Auckland

MIT, Oct 15, 2022

Nearest neighbor search in sphere

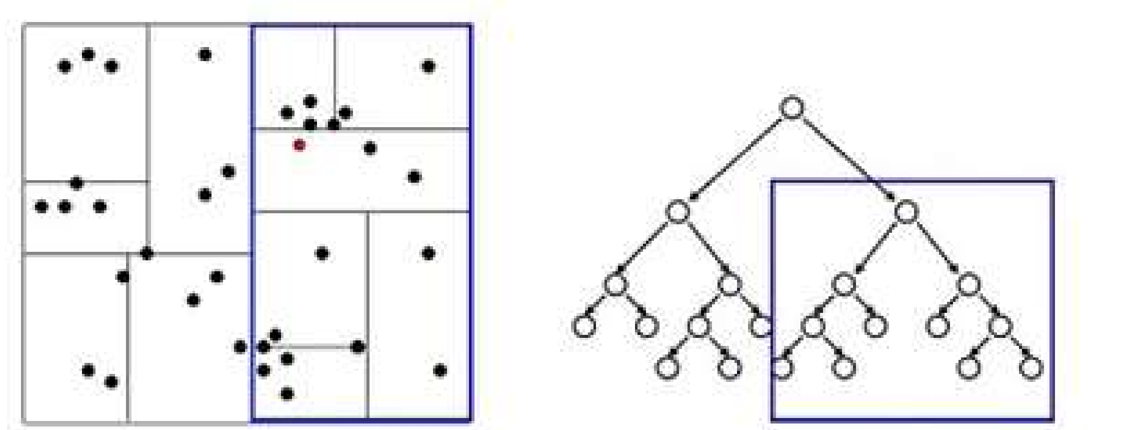
- Nearest neighbor search (NNS) in a unit sphere:
  - Given a data set $\mathbf{X}$ of size $\mathbf{n}$ and a query $\mathbf{q}$ in $\mathbf{d}$ dimensional unit sphere, return the point $\mathbf{x} \in \mathbf{X}$ such that the $\mathbf{dist(q, x)} = \|\mathbf{x} - \mathbf{q}\|$ is minimum.



$\mathbf{x}_1$ is the top-1.

Challenges of NNS

- Curse of dimensionality:
  - Given a polynomial indexing space $\mathbf{n^{O(1)}d^{O(1)}}$, existing a sublinear time algorithm to solve exact NNS refutes SETH [Wil18].
  - Indexing space or query time must be exponential in **d**.
- Classic solutions: KD Tree

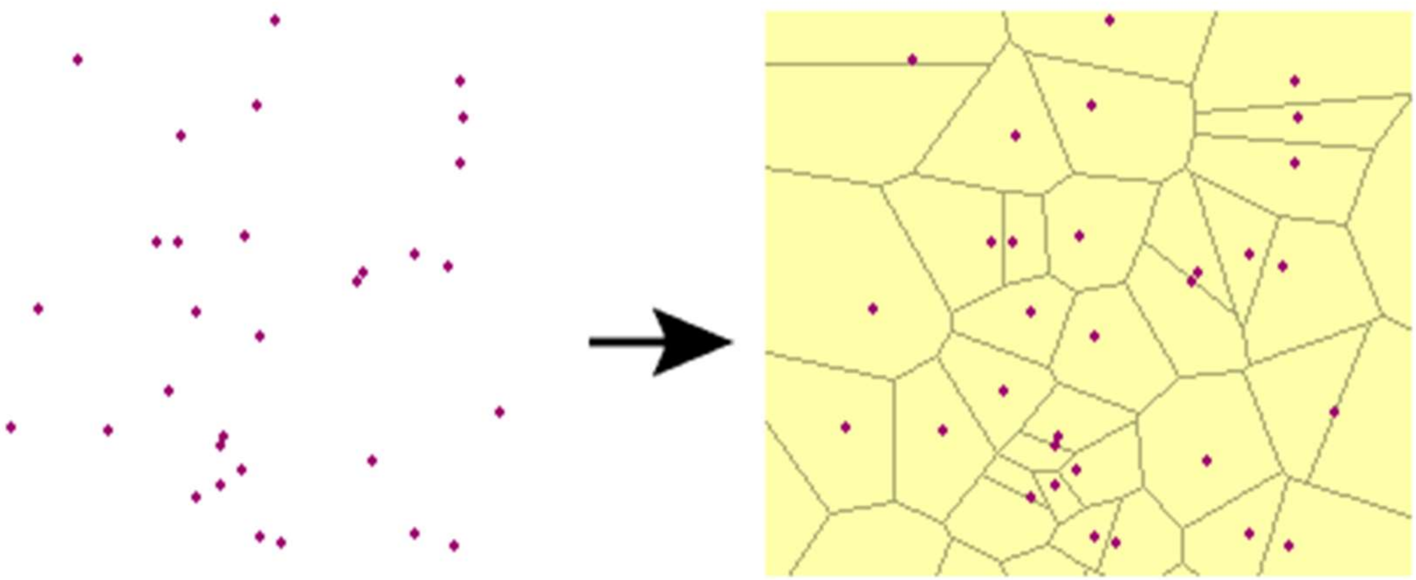


Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

Space: $\mathbf{O(n)}$
Range query: $\mathbf{O(dn^{1-1/d})}$

Challenges of NNS

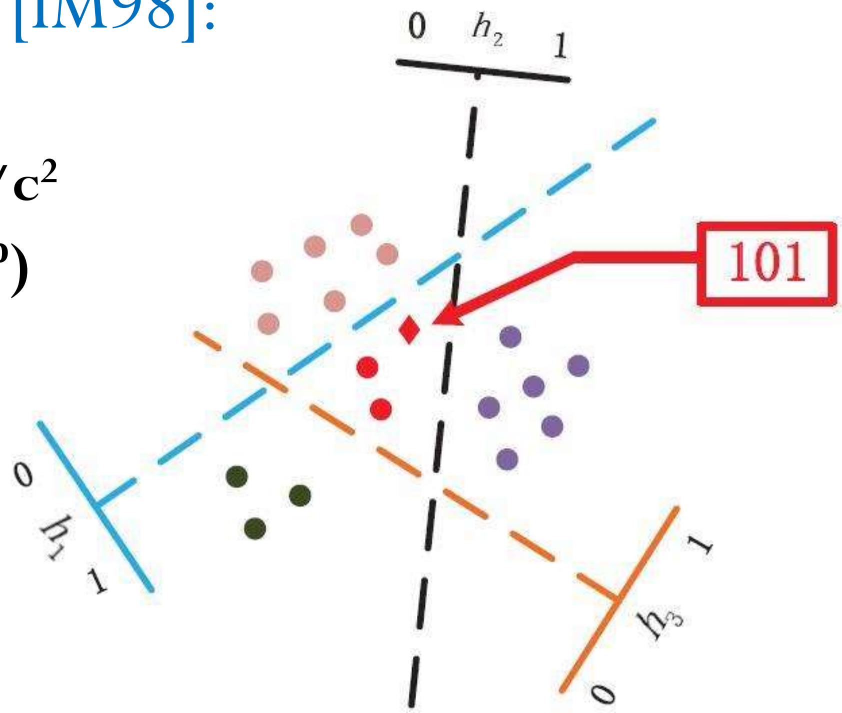
- Curse of dimensionality:
  - Given a polynomial indexing space $\mathbf{n}^{O(1)}\mathbf{d}^{O(1)}$, existing a sublinear time algorithm to solve exact NNS refutes SETH [Wil18].
  - Indexing space or query time must be exponential in $\mathbf{d}$.
- Classic solutions: Voronoi decomposition



Space: $\mathbf{O(n^{O(d)})}$
Time: $\mathbf{O(d^{O(1)}\log n)}$

Approximate NNS

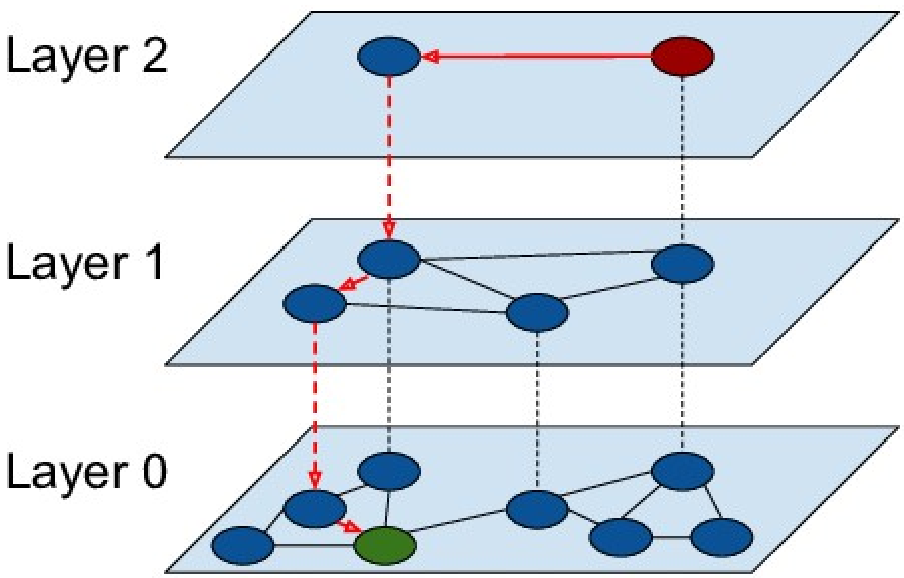
- Approximate NNS (ANNS):
  - Returns **x'** such that **dist(q, x') ≤ c dist(q, x)**
- Locality-sensitive hashing (LSH) [IM98]:
  - Theoretical guarantee to find **x'**
  - Sublinear query time $\mathbf{O(n^{\rho})}$, $\boldsymbol{\rho \approx 1/c^2}$
  - Subquadratic indexing space $\mathbf{O(n^{1+\rho})}$
  - **But**…



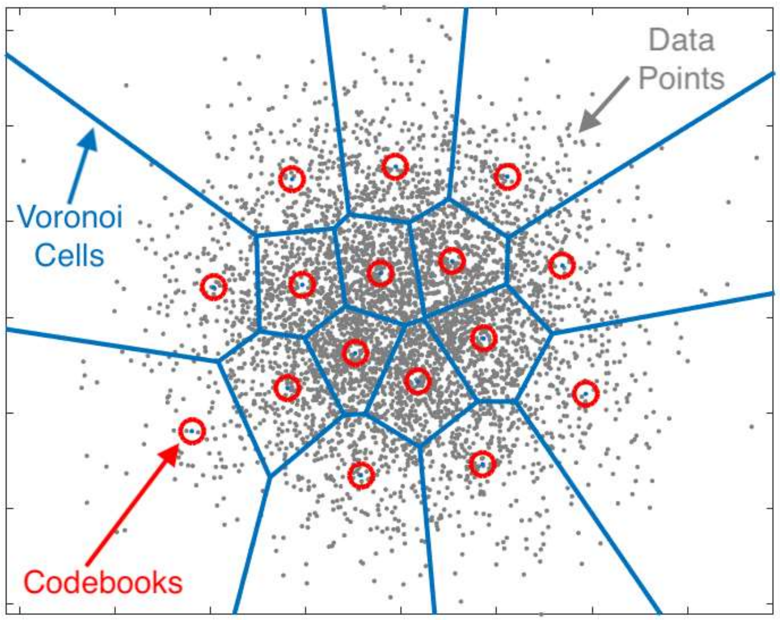
Approximate NNS

- Approximate NNS (ANNS):
  - Seek high empirical search recalls for top-**k** NNS (e.g. tiny **c**)

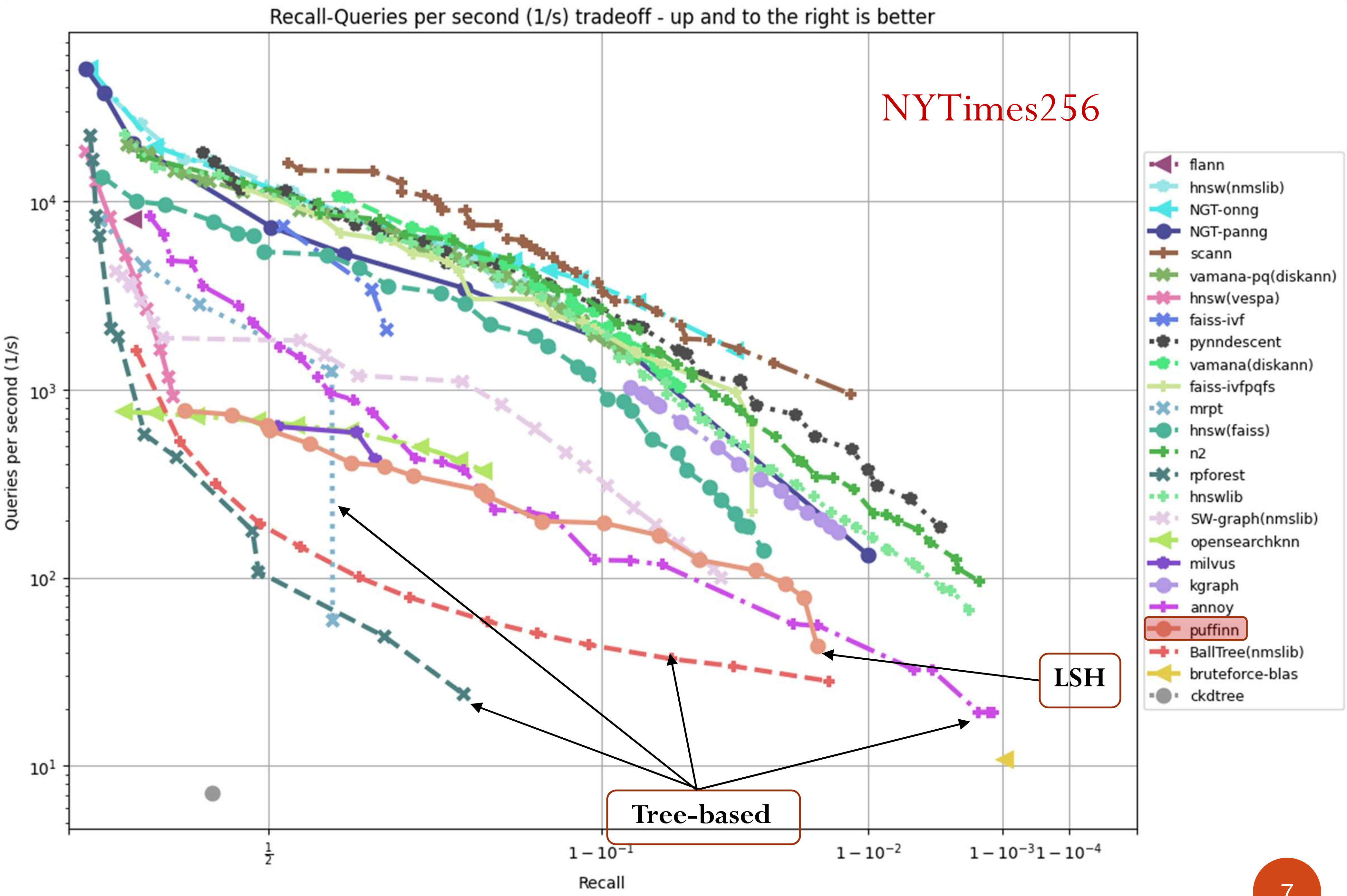
- Data-dependent approaches surpass LSH.



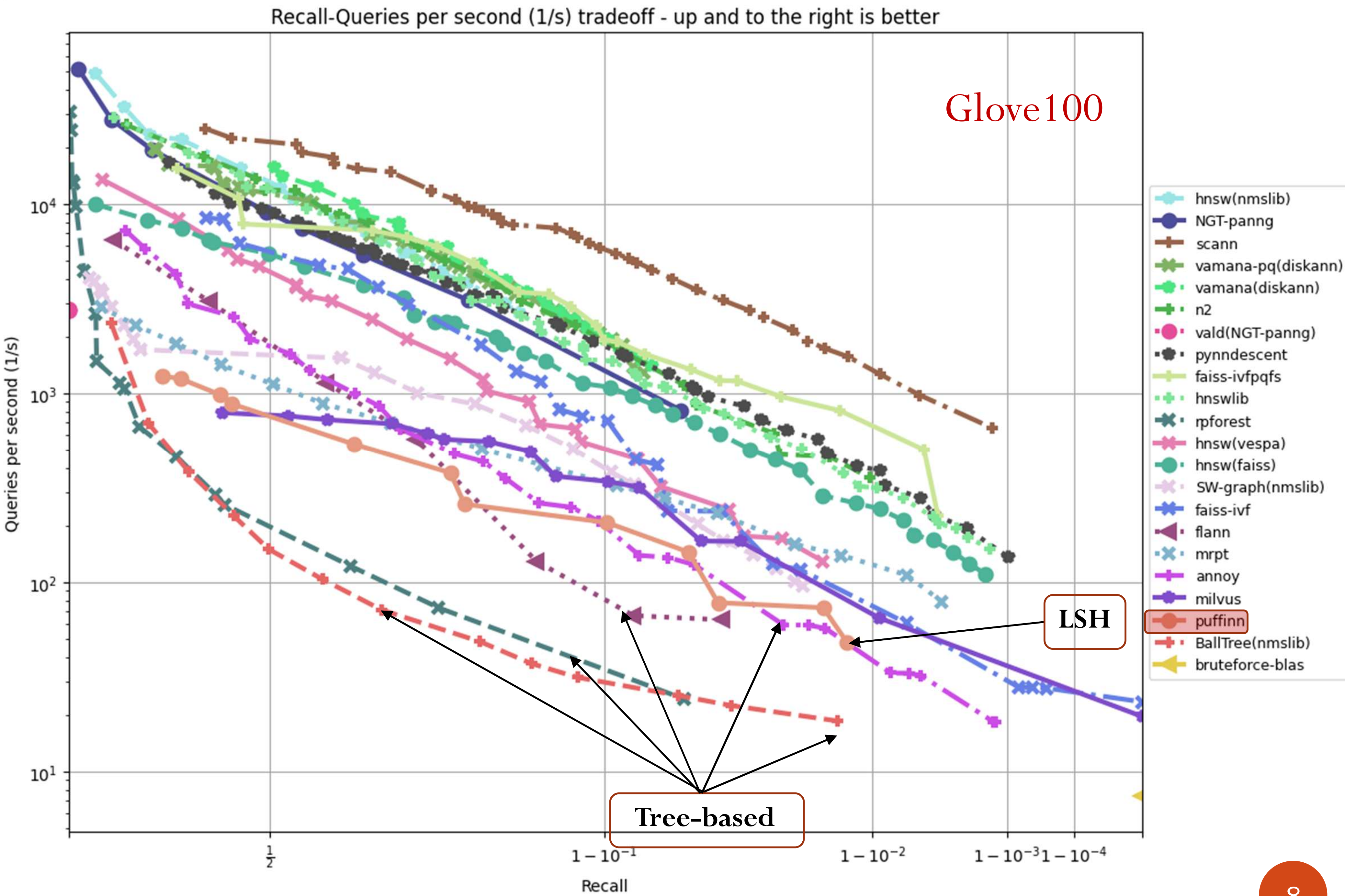
Graph-based index



Vector quantization



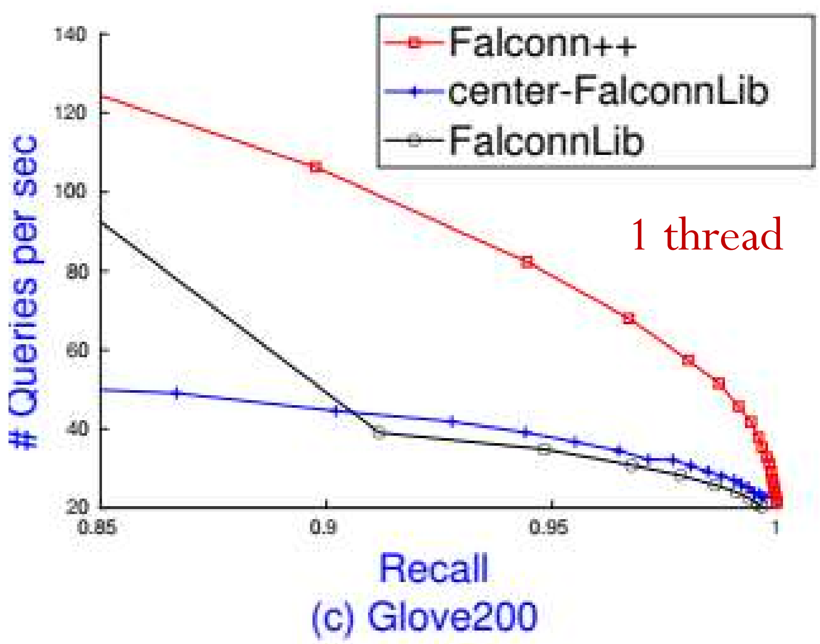
https://github.com/erikbern/ann-benchmarks/

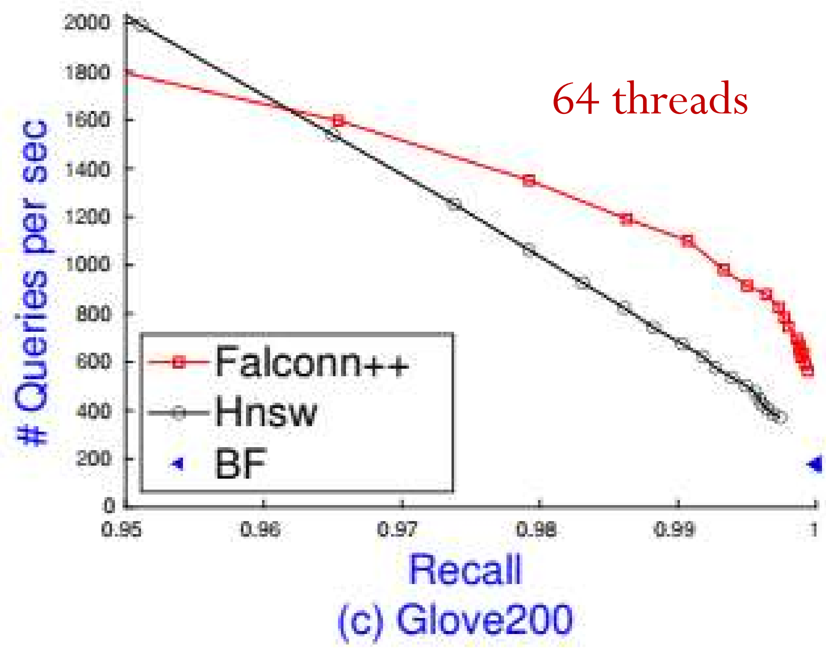


https://github.com/erikbern/ann-benchmarks/

Falconn++

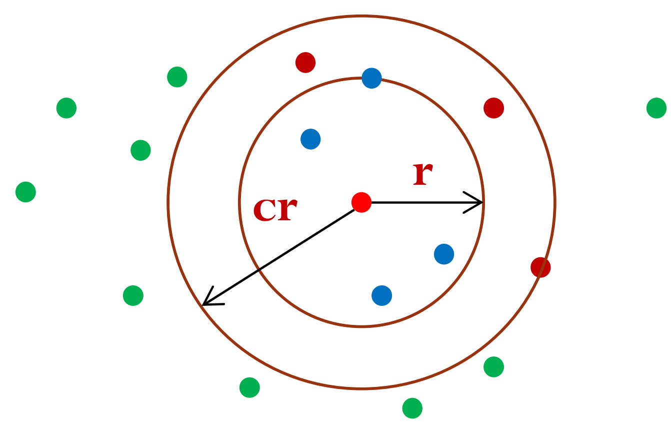
- A practical locality-sensitive filtering approach
  - Lower query time complexity than Falconn, an optimal LSH scheme on angular distance.
  - Empirical higher recall-speed tradeoffs than Falconn
  - Competitive with HNSW on high recall regimes



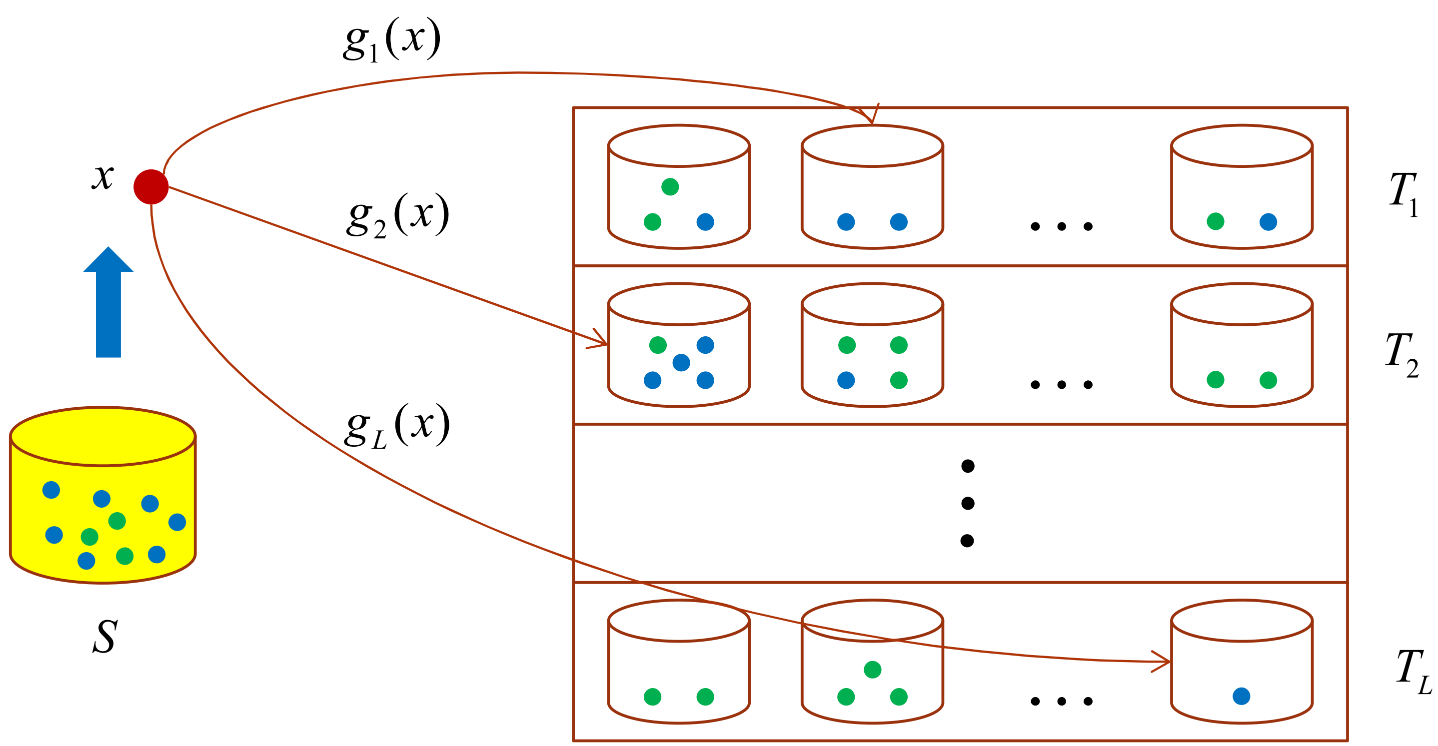


Locality-sensitive hashing (LSH)

- Definition [IM98]:
  - Given a distance function **dist(. , .)** and positive values **r**, **c**, $\mathbf{p_1}$, $\mathbf{p_2}$ where $\mathbf{p_1 > p_2}$, $\mathbf{c > 1}$. A family of functions **H** is called (**r**, **cr**, $\mathbf{p_1}$, $\mathbf{p_2}$)-sensitive if for uniformly chosen $\mathbf{h \in H}$ and all $\mathbf{x, y \in R^d}$:
    - If $\mathbf{dist(x, y) \le r}$ then $\mathbf{Pr\,[h(x) = h(y)] \ge p_1}$; (close points)
    - If $\mathbf{dist(x, y) \ge cr}$ then $\mathbf{Pr\,[h(x) = h(y)] \le p_2}$. (far away points)



Hash tables construction



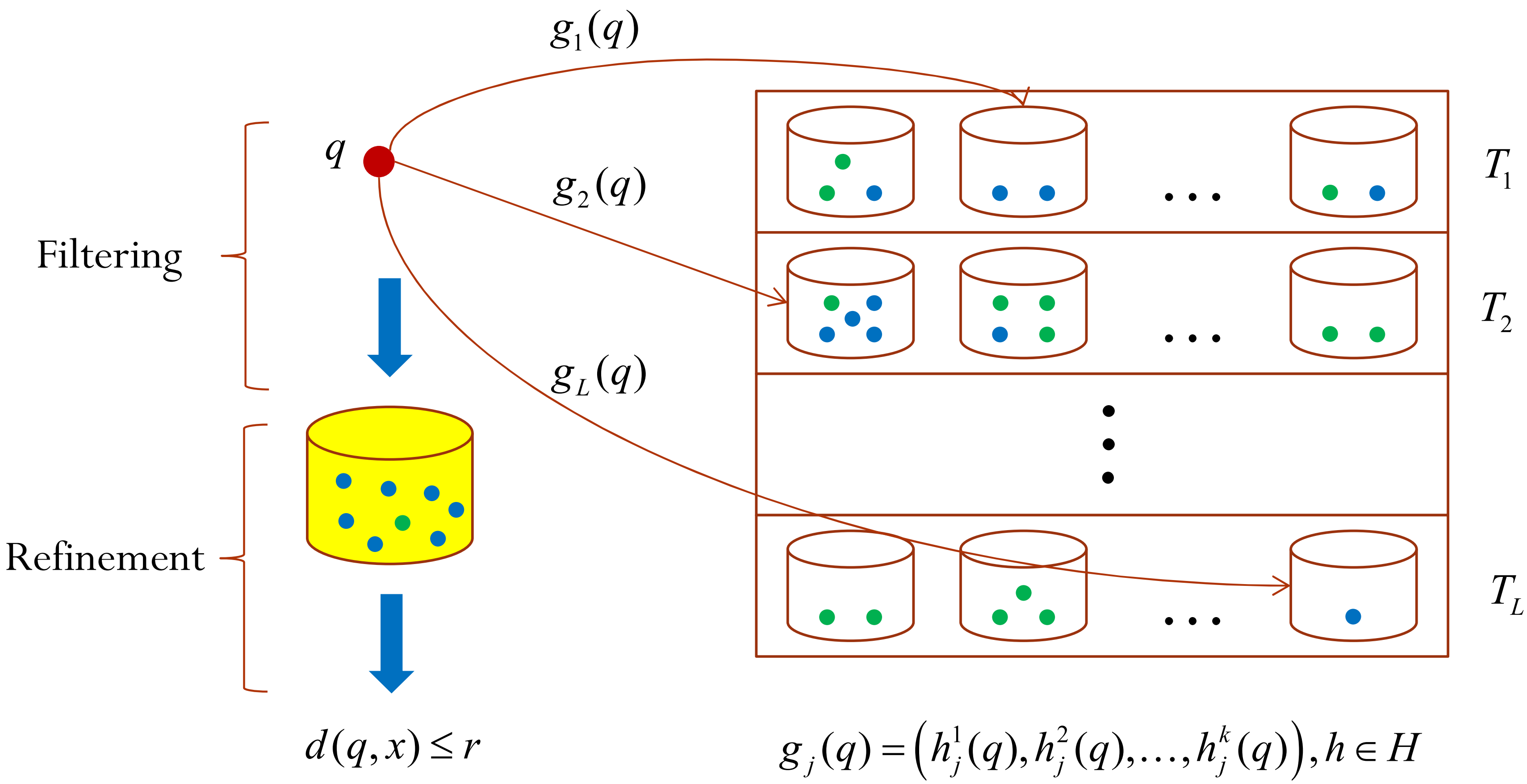
$$g_j(q) = \left(h_j^1(q), h_j^2(q), \ldots, h_j^k(q)\right), h \in H$$

Parameter settings:

$\mathbf{k = \ln(n)/\ln(1/p_2)}$
$\boldsymbol{\rho} \mathbf{= \ln(1/p_1)/\ln(1/p_2)}$
$\mathbf{L = n^{\rho}}$

Space: $\mathbf{O(dn + n^{1+\rho})}$

Hash tables lookup



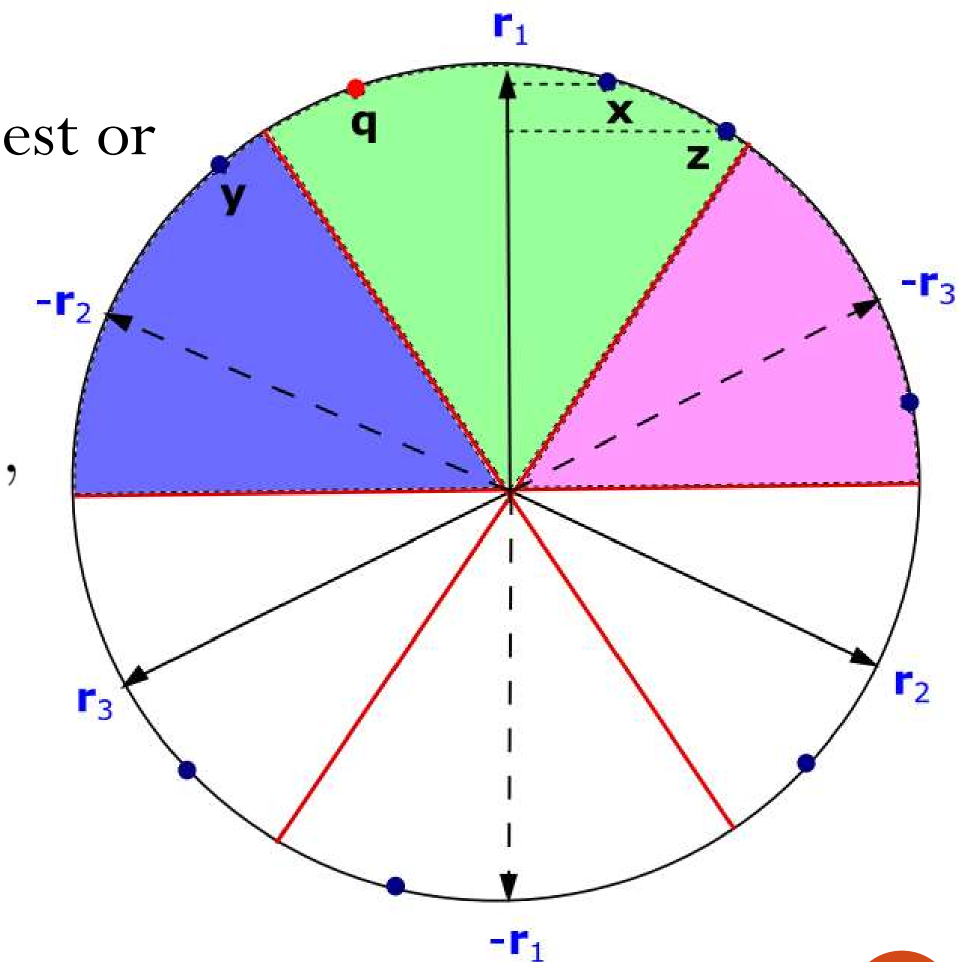
Time: Hashing time + $\mathbf{O(dn^{\rho})}$

Falconn [AIL+15]

- Construct a spherical Voronoi using **D** random vectors $\mathbf{r_i} \sim \mathbf{N^d(0, 1)}$
- **x** and **q** collide if sharing the closest or furthest random vector.
- Assume $\mathbf{r}_1 = \arg\max_{\mathbf{r}_i} |\mathbf{q}^\top \mathbf{r}_i|$,

$$h(\mathbf{q}) = \begin{cases} \mathbf{r}_1 \text{ if } \operatorname{sgn}(\mathbf{q}^\top \mathbf{r}_1) \geq 0\,, \\ -\mathbf{r}_1 \text{ otherwise}\,. \end{cases}$$

- Example:
  - $\mathbf{h(x) = h(z) = h(q) = r_1}$
  - $\mathbf{h(y) = -r_2}$



Falconn's takeaways

- Given **D** random vectors, if $\mathbf{dist(x, q) = r}$, then

$$\boxed{\mathbf{Pr}\left[h(\mathbf{x}) = h(\mathbf{q})\right] \approx D^{-\frac{1}{4/r^2-1}}}$$

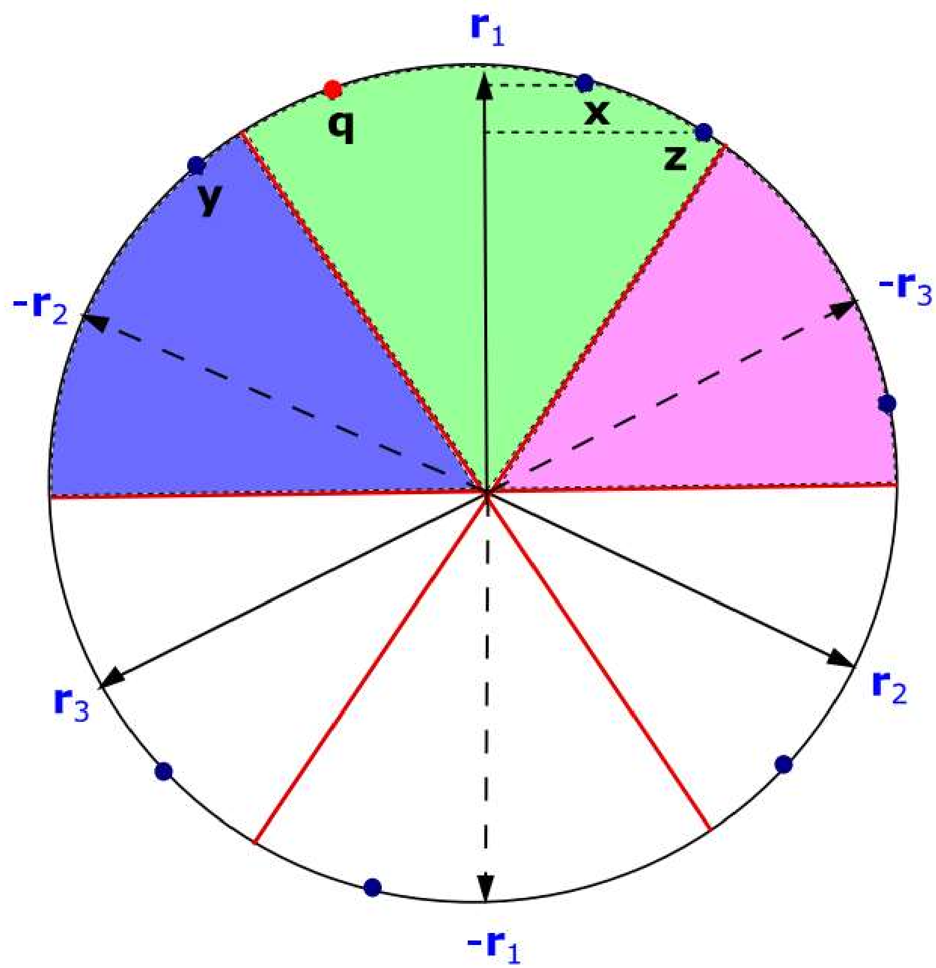
- Given a small **c**, $\mathbf{(r, cr, p_1, p_2)}$-sensitive Falconn has

$$\boxed{\rho \approx \frac{4/c^2r^2-1}{4/r^2-1} \approx 1/c^2}$$

- Lower bound [OZW14]: $\boldsymbol{\rho} \geq \mathbf{1/c^2 - o(1)}$ if $\mathbf{p_2 \geq 1/n}$

Practical multi-probe Falconn

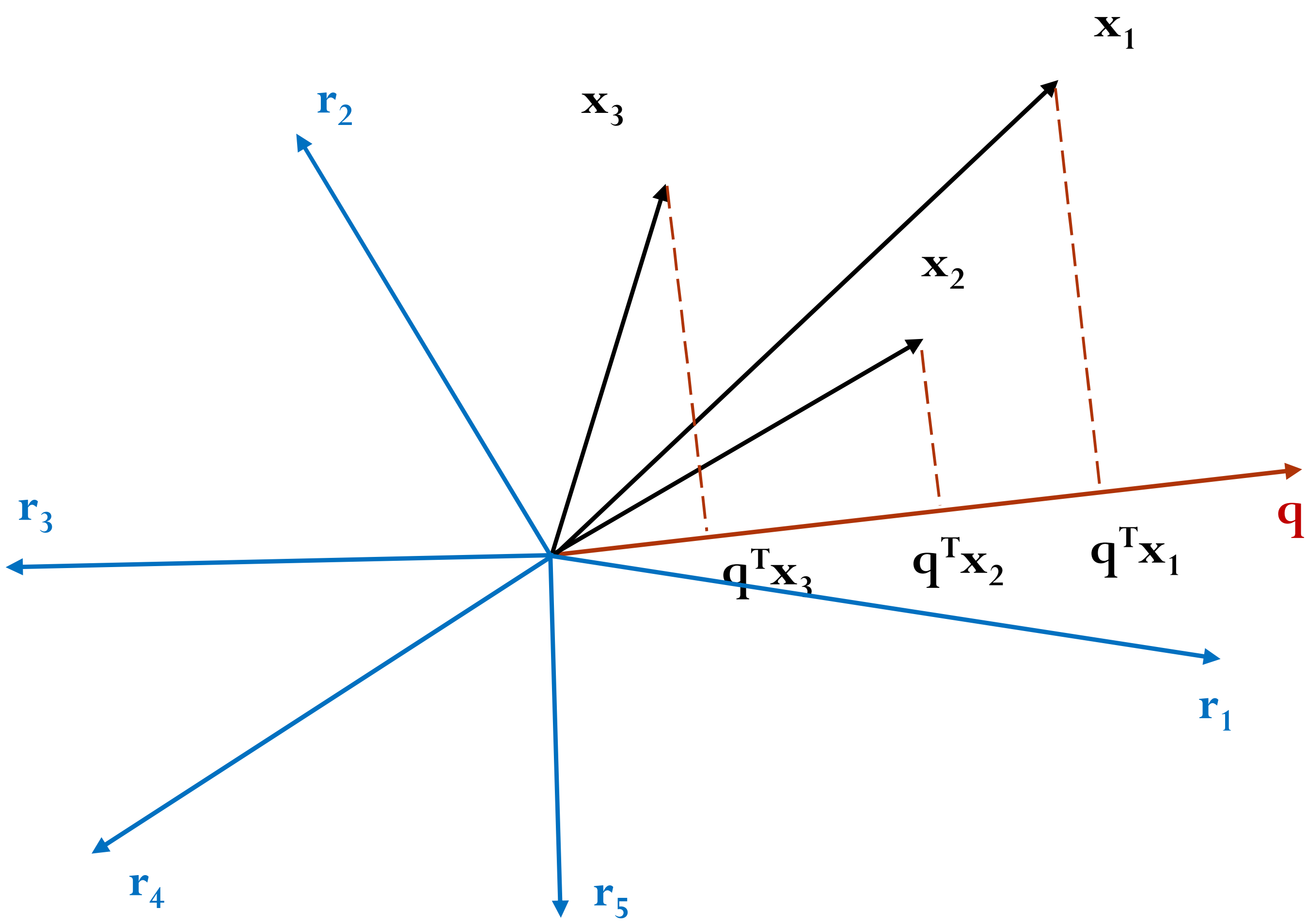
- To reduce **L**, probe the bucket of the next closest or furthest random vectors
  - Require a large **qProbes**
  - Compute up to **0.1n** distances to achieve recall of 90% in Glove300



- Example:
  - **q** is next closest to **–r**$_2$
  - The blue wedge is the next probe

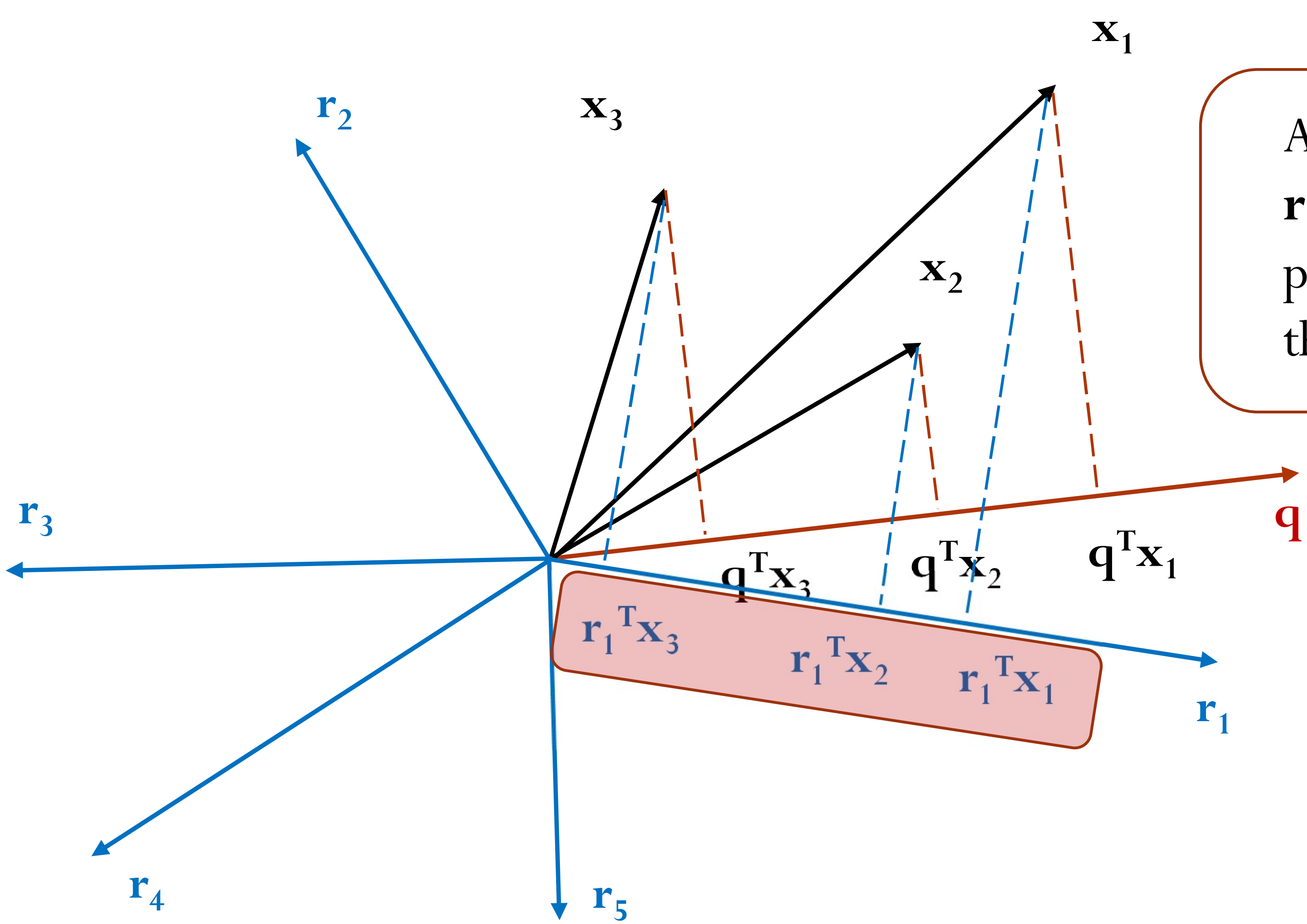
Geometric intuition

- Use random projections to find $\mathbf{x}$ s.t. $\mathbf{x}^T\mathbf{q}$ is maximum.



Geometric intuition

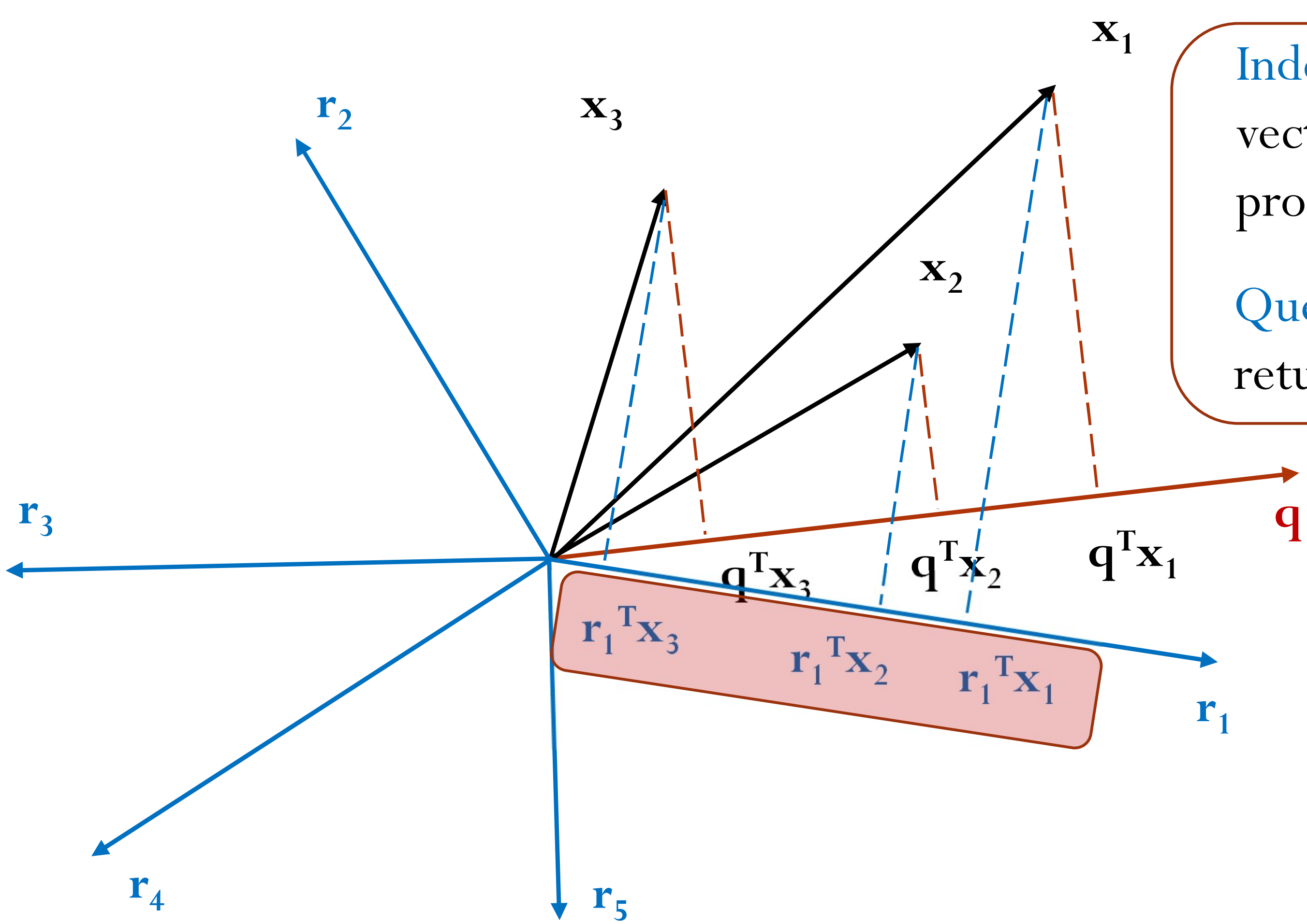
- Use a large number of random projections.



Among **5** random vectors $\mathbf{r_i}$, $\mathbf{r_1}$ is closest to $\mathbf{q}$ and the projections into $\mathbf{r_1}$ preserves the dot product order.

CEOs [Pha21]

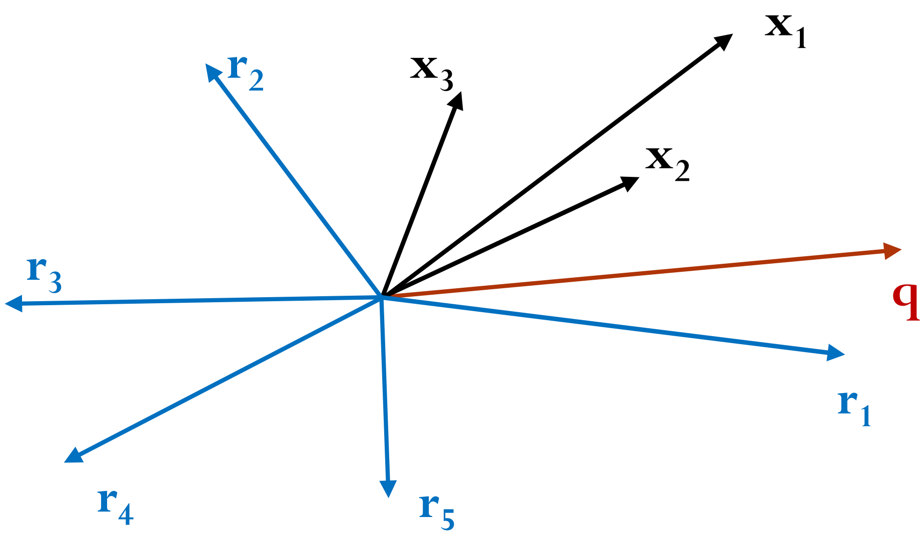
- Use a large number of random projections.



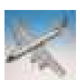
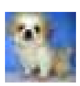
Indexing: Generate many random vectors $\mathbf{r_i}$, precompute the dot product order $\mathbf{L_i}$ for $\mathbf{r_i}$

Querying: Find $\mathbf{r_i}$ closest to $\mathbf{q}$ and return the top-1 NNS from $\mathbf{L_i}$

CEOs: Dimensionality reduction



Among **5** random vectors $\mathbf{r}_i$, we only use $\mathbf{r}_1$ to estimate dot products.

$\mathbf{x}_1$		1.2	0.9
$\mathbf{x}_2$		0.5	0.5
$\mathbf{x}_3$		0.1	0.7

$\mathbf{r}_1$	$\mathbf{r}_2$	$\mathbf{r}_3$	$\mathbf{r}_4$	$\mathbf{r}_5$
4	-2	-4	-3	-3
3	-1	-3	-4	-4
1	-3	-2	-3	-3

$\mathbf{q}$		1.5	0.1

9	4	-8	-3	0

Concomitants of Extreme Order statistics

x 

			$X_i = \mathbf{x}^T\mathbf{r}_i$					
	1.2	0.9	$\Rightarrow$	5	-2	4	-3	-3

q 

			$Q_i = \mathbf{q}^T\mathbf{r}_i$					
	1.5	0.1	$\Rightarrow$	8	4	9	-3	0

Assume $\| \mathbf{q} \| = 1$

Random projections: Using $\mathbf{D}$ random vectors $\mathbf{r}_i \sim \mathbf{N}^d(\mathbf{0}, \mathbf{1})$, we have $\mathbf{D}$ bivariate samples $(\mathbf{Q}_i, \mathbf{X}_i)$ from $\mathbf{N}(0, 0, 1, \| \mathbf{x} \|, \mathbf{x}^T\mathbf{q})$ where $\mathbf{Q}_i = \mathbf{q}^T\mathbf{r}_i$ and $\mathbf{X}_i = \mathbf{x}^T\mathbf{r}_i$

Concomitants of Extreme Order statistics

$\mathbf{x}$  | 1.2 | 0.9 | $\xrightarrow{X_i = \mathbf{x}^T\mathbf{r}_i}$

$X_{[2]}$	$X_{[3]}$	$X_{[1]}$	$X_{[5]}$	$X_{[4]}$
5	-2	4	-3	-3
8	4	9	-3	0
$Q_{(2)}$	$Q_{(3)}$	$Q_{(1)}$	$Q_{(5)}$	$Q_{(4)}$

$\mathbf{q}$  | 1.5 | 0.1 | $\xrightarrow[Q_i = \mathbf{q}^T\mathbf{r}_i]{}$

Assume $\| \mathbf{q} \| = 1$

Random projections: Using $\mathbf{D}$ random vectors $\mathbf{r}_i \sim \mathbf{N}^d(\mathbf{0}, \mathbf{1})$, we have $\mathbf{D}$ bivariate samples $(\mathbf{Q}_i, \mathbf{X}_i)$ from $\mathbf{N}(\mathbf{0}, \mathbf{0}, \mathbf{1}, \| \mathbf{x} \|, \mathbf{x}^T\mathbf{q})$ where $\mathbf{Q}_i = \mathbf{q}^T\mathbf{r}_i$ and $\mathbf{X}_i = \mathbf{x}^T\mathbf{r}_i$

Order statistics: Sort $\mathbf{D}$ pairs $(\mathbf{Q}_i, \mathbf{X}_i)$ by $\mathbf{Q}$-value, we form the order statistics where $\mathbf{Q}_{(1)}$ is the first order statistics and $\mathbf{X}_{[1]}$ is the concomitant of the first order statistics.

Concomitants of Extreme Order statistics

Concomitants of
Extreme Order statistics

$X_{[2]}$ $X_{[3]}$ $X_{[1]}$ $X_{[5]}$ $X_{[4]}$

$\mathbf{x}$ 

				$X_i = x^T r_i$					
x	1.2	0.9	⇒	5	-2	4	-3	-3

$\mathbf{q}$ 

q	1.5	0.1	⇒	8	4	9	-3	0

$Q_i = q^T r_i$

$Q_{(2)}$ $Q_{(3)}$ $Q_{(1)}$ $Q_{(5)}$ $Q_{(4)}$

Assume $\| \mathbf{q} \| = 1$

Extreme Order statistics

Random projections: Using **D** random vectors $\mathbf{r}_i \sim \mathbf{N}^d(0, 1)$, we have **D** bivariate samples $(\mathbf{Q}_i, \mathbf{X}_i)$ from $\mathbf{N}(0, 0, 1, \| \mathbf{x} \|, \mathbf{x}^T\mathbf{q})$ where $\mathbf{Q}_i = \mathbf{q}^T\mathbf{r}_i$ and $\mathbf{X}_i = \mathbf{x}^T\mathbf{r}_i$

Order statistics: Sort **D** pairs $(\mathbf{Q}_i, \mathbf{X}_i)$ by **Q**-value, we form the order statistics where $\mathbf{Q}_{(1)}$ is the first order statistics and $\mathbf{X}_{[1]}$ is the concomitant of the first order statistics.

Extreme order statistics: When **D** is sufficiently large, $\mathbf{Q}_{(1)}$ is the extreme order statistics and $\mathbf{X}_{[1]}$ is the concomitant of the extreme order statistics.

Theory of Concomitants of Extreme Order statistics [DG74]

Concomitants of
Extreme Order statistics

	$X_{[2]}$	$X_{[3]}$	$X_{[1]}$	$X_{[5]}$	$X_{[4]}$
$\mathbf{x}$  1.2, 0.9 $\xrightarrow{X_i = x^T r_i}$	5	-2	4	-3	-3
$\mathbf{q}$  1.5, 0.1 $\xrightarrow{Q_i = q^T r_i}$	8	4	9	-3	0
	$Q_{(2)}$	$Q_{(3)}$	$Q_{(1)}$	$Q_{(5)}$	$Q_{(4)}$

Assume $\| \mathbf{q} \| = 1$

Extreme Order statistics

Extreme order statistics: $\mathbf{Q_{(1)}}$ is the maximum variable among $\mathbf{D}$ random variables $\mathbf{Q_i = q^T r_i \sim N(0, 1)}$.

Extreme order statistics:

$$\mathbf{E[Q_{(1)}] \approx \sqrt{2\ln(D)}, \ Var[Q_{(1)}] \approx 0}$$

Concomitant of extreme order statistics:

$$\mathbf{X_{[1]} \sim N(x^T q \sqrt{2\ln(D)}, \ \| x \|^2 - (x^T q)^2)}$$

Theory of Concomitants of Extreme Order statistics [DG74]

Concomitants of
Extreme Order statistics

$X_{[D-1]}$ $X_{[1]}$ $X_{[2]}$ $X_{[D]}$

x 

1.2	0.9

$X_i = x^T r_i$ $\Rightarrow$

-5	-2	4	• • •	3	-8

q 

1.5	0.1

$Q_i = q^T r_i$ $\Rightarrow$

-8	4	9	• • •	7	-9

$Q_{(D-1)}$ $Q_{(1)}$ $Q_{(2)}$ $Q_{(D)}$

Assume $\| \mathbf{q} \| = 1$

Extreme Order statistics

Top s_0 maximum and minimum order statistics: $Q_{(i)}$ and $Q_{(D-i+1)}$ where $i = 1, \ldots, s_0$

Extreme order statistics:

$$E[Q_{(i)}] \approx \sqrt{2\ln(D)}$$
$$E[Q_{(D-i+1)}] \approx -\sqrt{2\ln(D)}$$

Concomitant of extreme order statistics:

$$X_{[i]} \sim N(x^T q \sqrt{2\ln(D)},\ \| x \|^2 - (x^T q)^2)$$
$$X_{[D-i+1]} \sim N(-x^T q \sqrt{2\ln(D)},\ \| x \|^2 - (x^T q)^2)$$

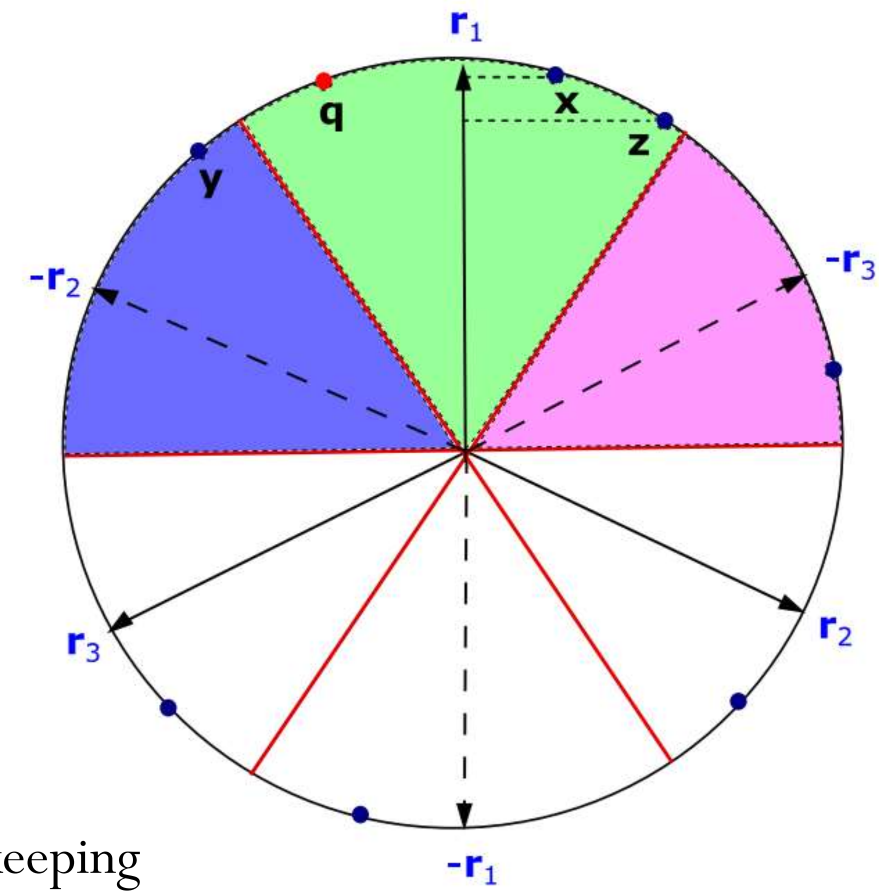
$X_{[i]}$ and $X_{[D-i+1]}$ are independent asymptotically.

Connection to multi-probe Falconn

- Falconn: If $\mathbf{r}_1 = \arg\max_{\mathbf{r}_i} |\mathbf{q}^\top \mathbf{r}_i|$,
  - Use $\mathbf{r}_1$ corresponding to $\mathbf{Q}_{(1)}$ as hash value
  - Use $\mathbf{Q}_{(\mathbf{i})}$ and $\mathbf{Q}_{(\mathbf{D-i})}$ as probing buckets where $\mathbf{i} = 1, \dots, \mathbf{s}_0$
- CEOs: If $\mathbf{r}_1 = \arg\max_{\mathbf{r}_i} |\mathbf{q}^\top \mathbf{r}_i|$,
  - Use $\mathbf{X}_{[1]} = \mathbf{x}^T\mathbf{r}_1$ to estimate $\mathbf{x}^T\mathbf{q}$
  - Use $\mathbf{X}_{[\mathbf{i}]}$ and $\mathbf{X}_{[\mathbf{D-i}]}$ as estimators of $\mathbf{x}^T\mathbf{q}$ where $\mathbf{i} = 1, \dots, \mathbf{s}_0$
- Falconn++ = Falconn + CEOs

Partition **n** points into **2D** buckets

Scale each bucket by keeping $\mathbf{x}$ with largest $\mathbf{x}^T\mathbf{r}_i$



Falconn++: A locality-sensitive filtering

- A locality-sensitive filtering (LSF) mechanism:
  - Given a distance function $\mathbf{dist}(.\,,.)$ and positive values $\mathbf{r}$, $\mathbf{c}$, $\mathbf{q_1}$, $\mathbf{q_2}$ where $\mathbf{q_1 > q_2}$, $\mathbf{c > 1}$. For an ($\mathbf{r}$, $\mathbf{cr}$, $\mathbf{p_1}$, $\mathbf{p_2}$)-sensitive function $\mathbf{h}$ and $\mathbf{x, y}$ in $\mathbf{h(q)}$:
    - If $\mathbf{dist(x, q)} \le \mathbf{r}$ then $\mathbf{Pr}$ [$\mathbf{x}$ is not filtered] $\ge \mathbf{q_1}$
    - If $\mathbf{dist(y, q)} \ge \mathbf{cr}$ then $\mathbf{Pr}$ [$\mathbf{y}$ is not filtered] $\le \mathbf{q_2}$

- Combine LSH and LSF:
  - $\mathbf{Pr}$ [$\mathbf{h(x) = h(q)}$, $\mathbf{x}$ is not filtered] $\ge \mathbf{p_1 q_1}$
  - $\mathbf{Pr}$ [$\mathbf{h(y) = h(q)}$, $\mathbf{y}$ is not filtered] $\le \mathbf{p_2 q_2}$

- We need $\mathbf{\ln(1/q_1) \,/\, \ln(1/q_2) \le \rho \approx 1/c^2}$ to achieve a new exponent $\mathbf{\rho' \le \rho}$

Falconn++

- Asymptotic property of CEOs:
  - $\mathbf{X_{[1]} \sim N(x^T q \sqrt{2 \ln(D)},\ \| x \|^2 - (x^T q)^2)}$
  - $\mathbf{Y_{[1]} \sim N(y^T q \sqrt{2 \ln(D)},\ \| y \|^2 - (y^T q)^2)}$
- Filtering mechanism:
  - Define a threshold $\mathbf{t = (1 - r^2/2)\sqrt{2\ln(D)}}$. For each bucket corresponding to $\mathbf{r_i}$, keep any point $\mathbf{x}$ if $\mathbf{x^T r_i} \geq \mathbf{t}$. Otherwise, discard it.
  - Note: $\mathbf{dist(x, q) = r}$, then $\mathbf{x^T q = 1 - r^2/2}$

Falconn++'s takeaways

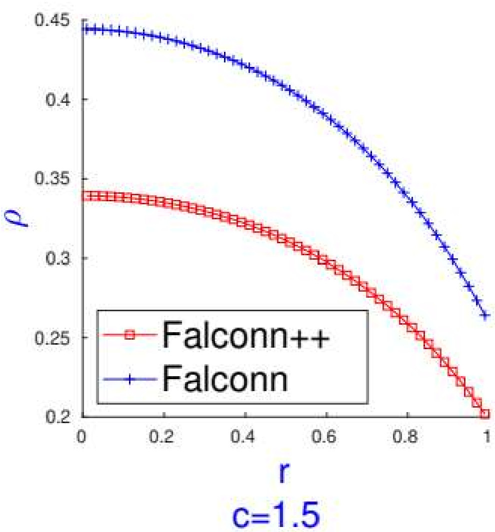
- For sufficiently large **D** random projections, **c > 1**,

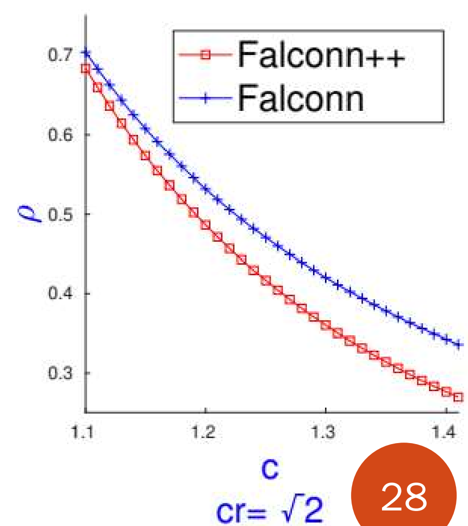
- *If* $\|\mathbf{x} - \mathbf{q}\| \leq r$, *then* $\mathbf{Pr}\left[\mathbf{x} \textit{ is not filtered}\right] \geq q_1 = 1/2$;
- *If* $\|\mathbf{y} - \mathbf{q}\| \geq cr$, *then* $\mathbf{Pr}\left[\mathbf{y} \textit{ is not filtered}\right] \leq q_2 = \frac{1}{\gamma\sqrt{2\pi}} \exp(-\gamma^2/2) < q_1$ *where* $\gamma = \frac{cr(1-1/c^2)}{\sqrt{4-c^2r^2}} \cdot \sqrt{2 \ln D}$.

- New exponent $\boldsymbol{\rho' \approx 1/(2c^2 - 2 + 1/c^2)}$

$$\rho' = \frac{\ln(1/q_1p_1)}{\ln(1/q_2p_2)} \approx \frac{\frac{\ln 2}{\ln D} + \frac{1}{4/r^2-1}}{\frac{(1-1/c^2)^2}{4/c^2r^2-1} + \frac{1}{4/c^2r^2-1}}$$

$$\approx \frac{1}{1 + (1 - 1/c^2)^2} \cdot \frac{4/c^2r^2 - 1}{4/r^2 - 1} \leq \rho\,.$$





Connection to LSF framework [ALRW17]

- Asymmetric LSF frameworks:
  - Apply different filtering conditions on data and query to govern the space-time tradeoff
  - Let $\mathbf{t_u}$ and $\mathbf{t_q}$ be two different thresholds.
  - Collision: $\mathbf{x}$ and $\mathbf{q}$ pass the filter $\mathbf{r_i}$ with $\mathbf{Pr[x^Tr_i \geq t_u , q^Tr_i \geq t_q]}$

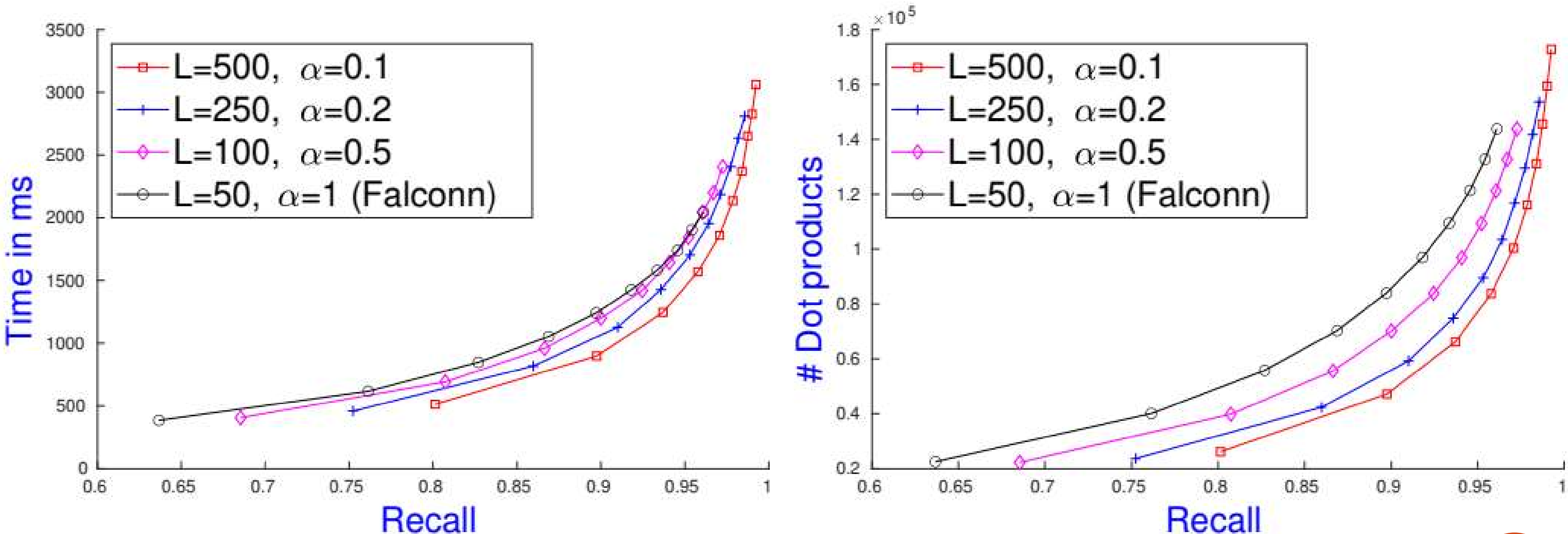
- Falconn++:
  - Use a sufficiently large $\mathbf{D}$ to ensure the asymptotic property of CEOs
  - $\mathbf{t_u = (1 - r^2/2)\sqrt{2\ln(D)}}$, $\mathbf{t_q \approx \sqrt{2\ln(D)}}$
  - For $\mathbf{p_2q_2 = 1/n, D = O(n^{\rho'})}$, Falconn++ yields $\mathbf{O(n^{\rho'})}$ query time where $\boldsymbol{\rho'} \approx 1/(2c^2 - 2 + 1/c^2)$ for $\mathbf{r} \geq \sqrt{\mathbf{2}}$

Practical implementations

- Data-dependent setting:
  - Select a scaling factor $\mathbf{0 < \alpha < 1}$ to scale each bucket of size **B** to $\boldsymbol{\alpha}$ **B**
    - Adapt **t** to various density
    - Easy to govern the memory footprint (i.e. # points in a table)
- Multi-probe indexing:
  - For each point **x**, hash it into **iProbes** buckets corresponding the **iProbes** closest or furthest random vectors
  - Scale each bucket of size **B** to $\boldsymbol{\alpha}$ **B/iProbes**
- Other heuristics:
  - Pseudo-random rotation $\mathbf{HD_3HD_2HD_1}$ to simulate random projections
  - Center the data point **X**
  - Limit scaling: keep **max(k,** $\boldsymbol{\alpha}$ **B/iProbes)** points in a bucket

Falconn++: Scaling bucket

- Experiment on Glove200 with **1.2M** points with **k = 20**:
  - **D = 256**, 2 combined LSH functions, each table has $\mathbf{4D^2}$ buckets
  - Falconn: **qProbes = {1000, … , 20000}**, Falconn++: **qProbes/α**

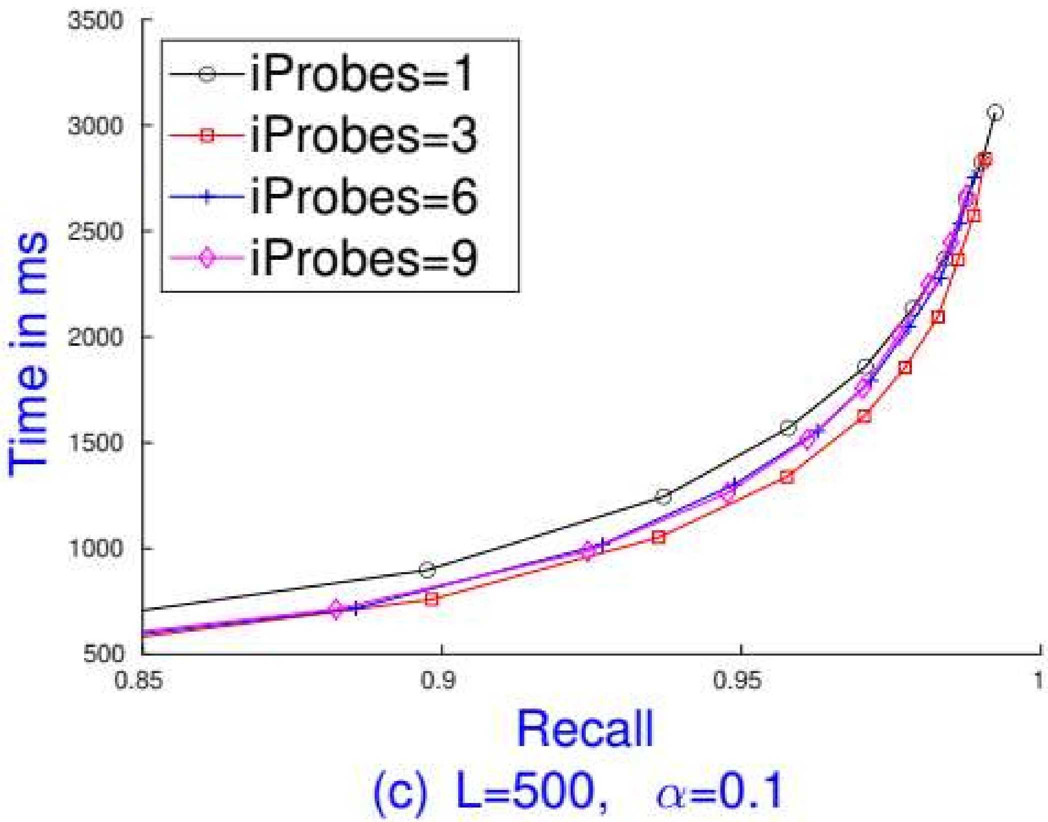


(a) iProbes=1

(b) iProbes=1

Falconn++: Multi-probe indexing

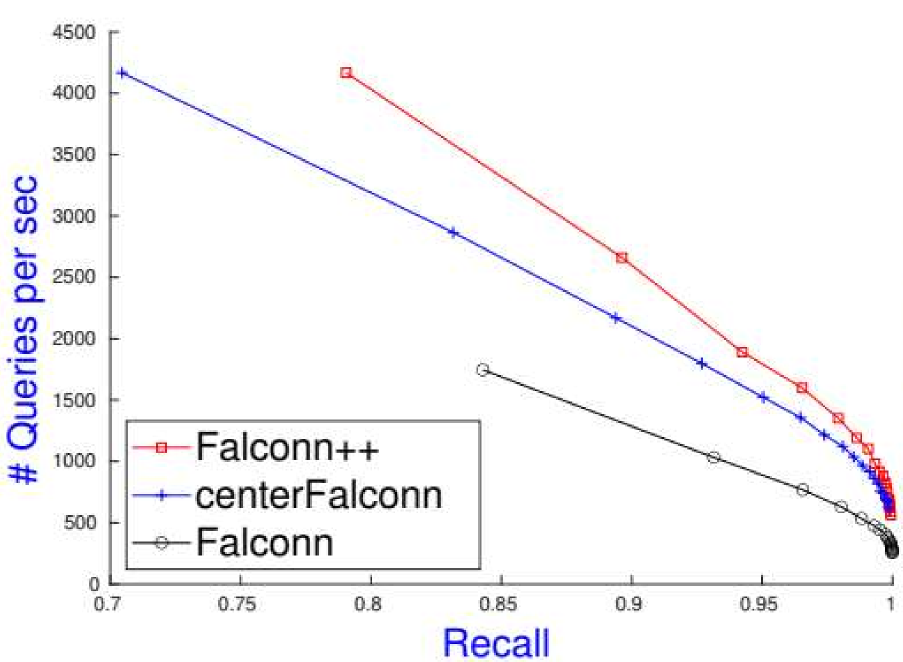
- Observation:
  - Overfitting: Large **iProbes** degrades performance
  - **iProbes = 2D** is as similar as theoretical LSF framework
- Setting iProbes:
  - $\mathbf{k \approx n \cdot iProbes / 4D^2}$
  - Each bucket has roughly **k=20** points, especially sparse buckets
  - $\mathbf{iProbes = 3: 1.2M \cdot 3/2^{18} = 16}$



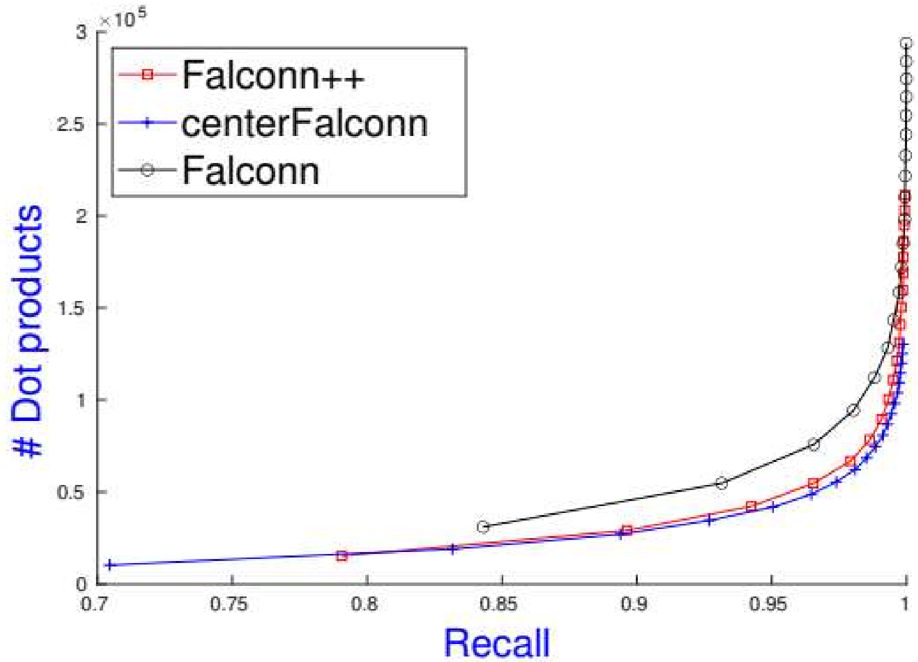
Glove200, **k** = 20

Falconn++: Centering data

- Experiment on Glove200 with **k = 20**:
  - **D = 256**, 2 combined LSH functions, **4D^2** buckets/table, **L = 500**
  - Falconn++: **α = 0.1, iProbes = 3, qProbes/α**



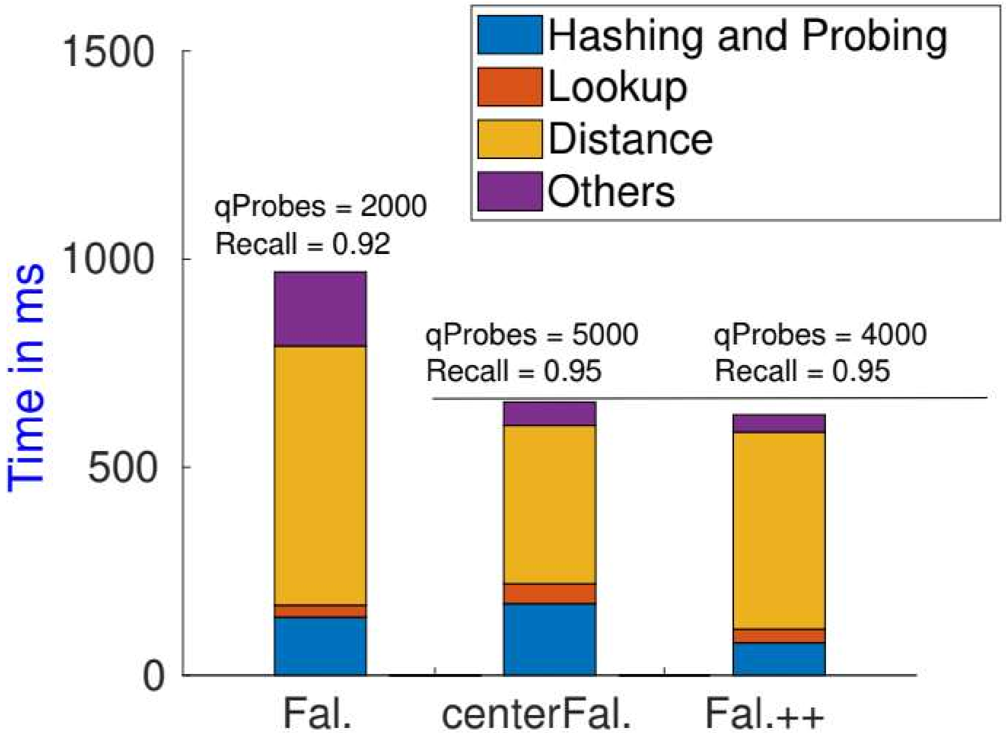
(a) Speed-recall tradeoff



(b) # Dot products-recall tradeoff

Falconn++: Limit scaling & centering

- Observations:
  - After centering, buckets are more balanced.
  - With $\boldsymbol{\alpha = 0.1}$, **iProbes = 3,** and keep $\mathbf{max(k, \alpha B/iProbes)}$ points, # points/table ≈ **2.42 n**
  - Less **qProbes** and hash evaluation time than Falconn
- Future improvement:
  - SimHash signatures to reduce distance computation time

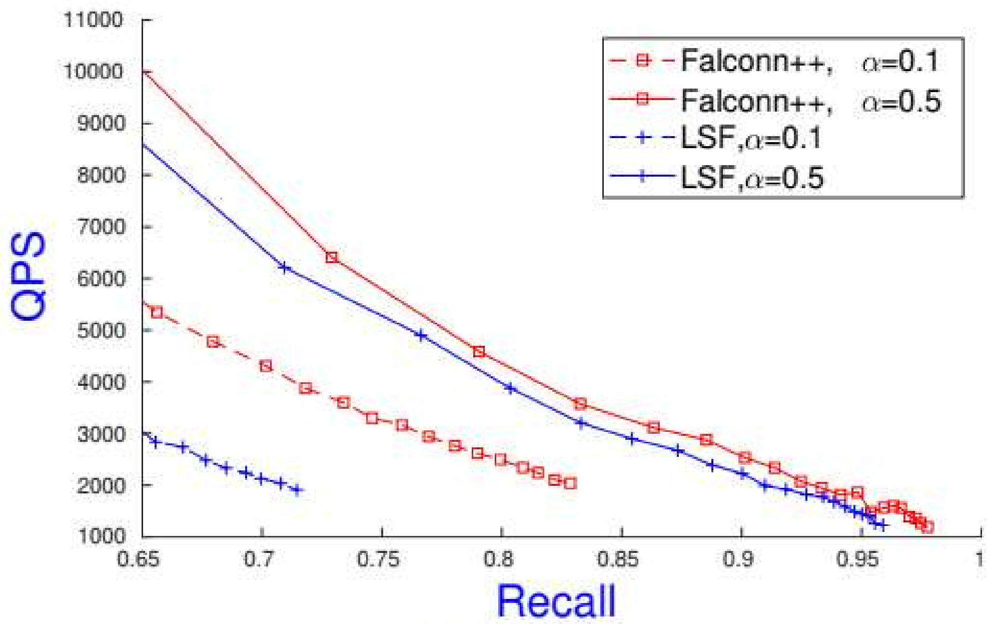


(c) Query time components

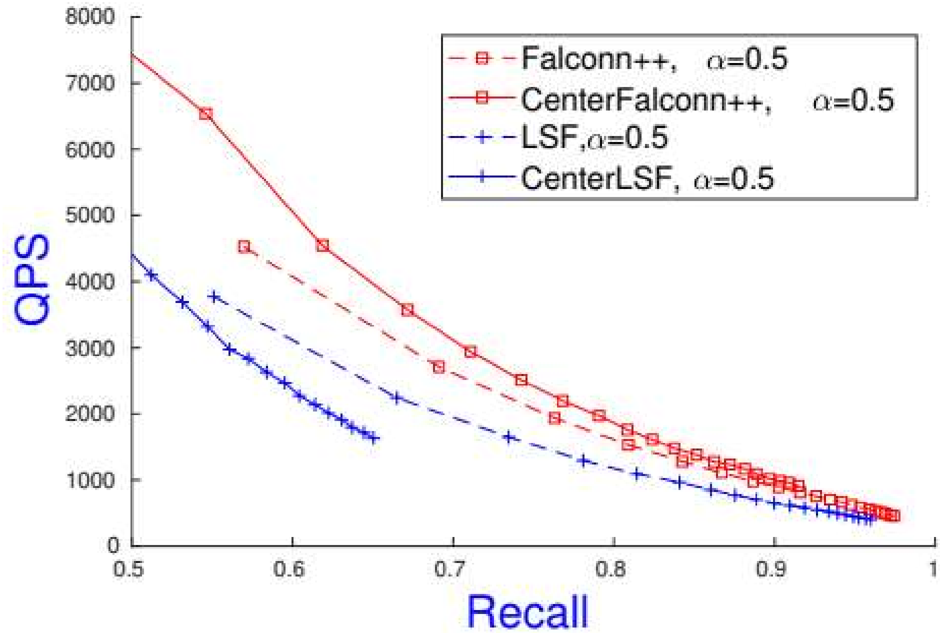
Glove200, **k** = 20

Falconn++ vs. Theoretical LSF

- LSF: Select $\mathbf{t_u}$ s.t. $\mathbf{Pr}\left[\mathbf{x}^\top \mathbf{r}_i \geq t_u\right]^2 = \alpha/4D^2$
  - Falconn++: No limit scaling, only centering
  - **iProbes = 1, D = {128, 256}, L = 100**, 2 combined LSH/LSF functions



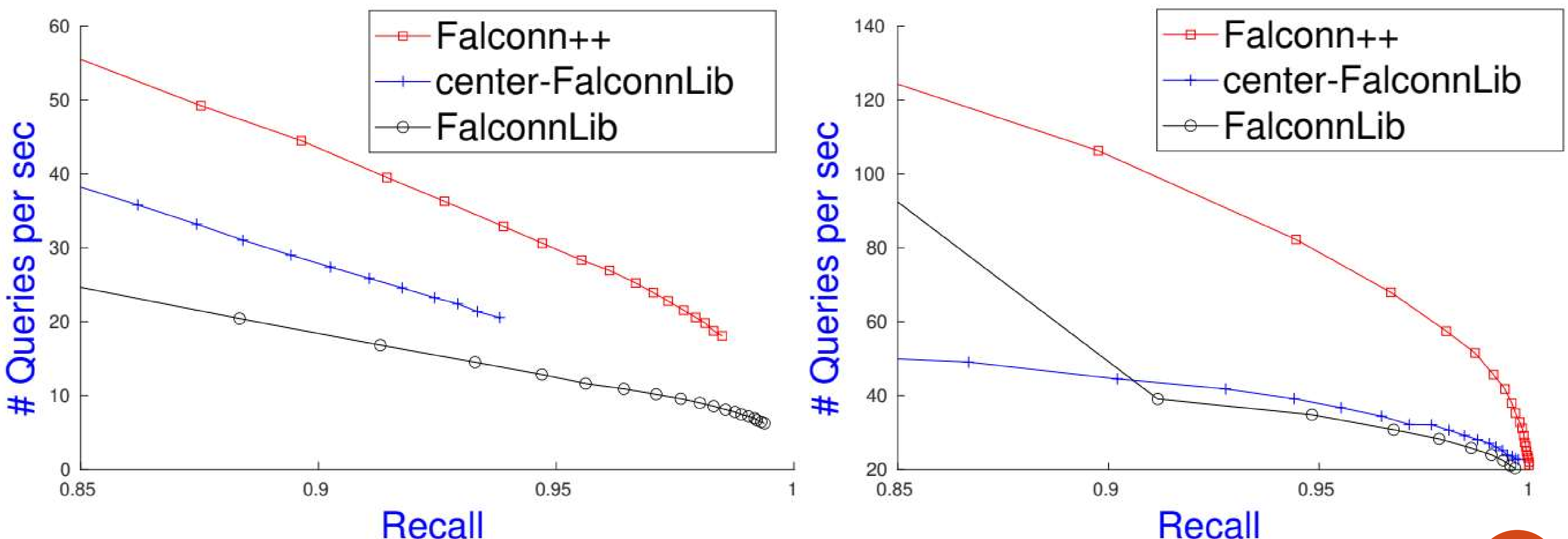
(a) NYTimes



(c) Glove200

Falconn++ vs. FalconnLib

- Glove300 and Glove200 with **k = 20** and **1 thread**:
  - **L = 500, D = 256**, **α = 0.1, iProbes = {1, 3}, 4D²** buckets/table
  - Falconn: **L = 50** (Glove300), **L = 1210** (Glove200)



(b) Glove300

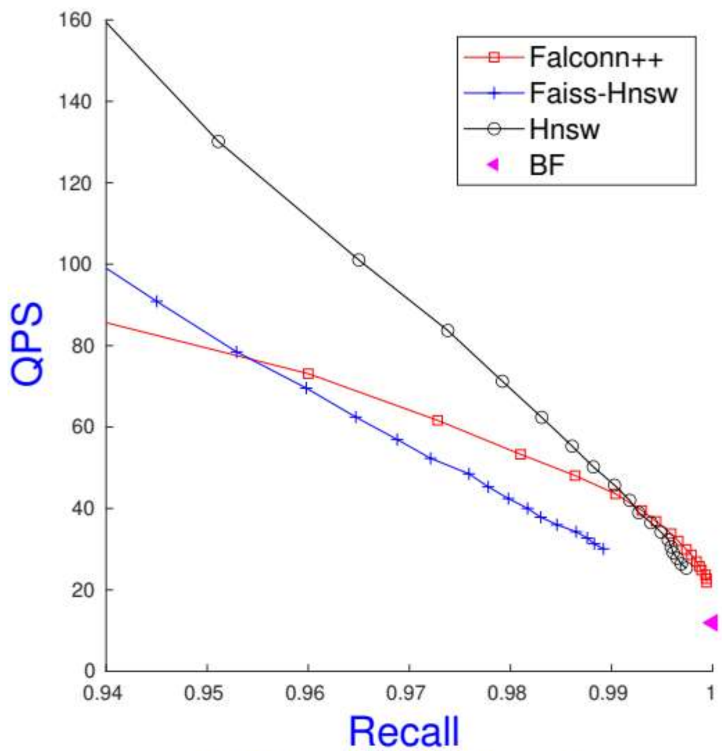
(c) Glove200

Falconn++ vs Hnsw

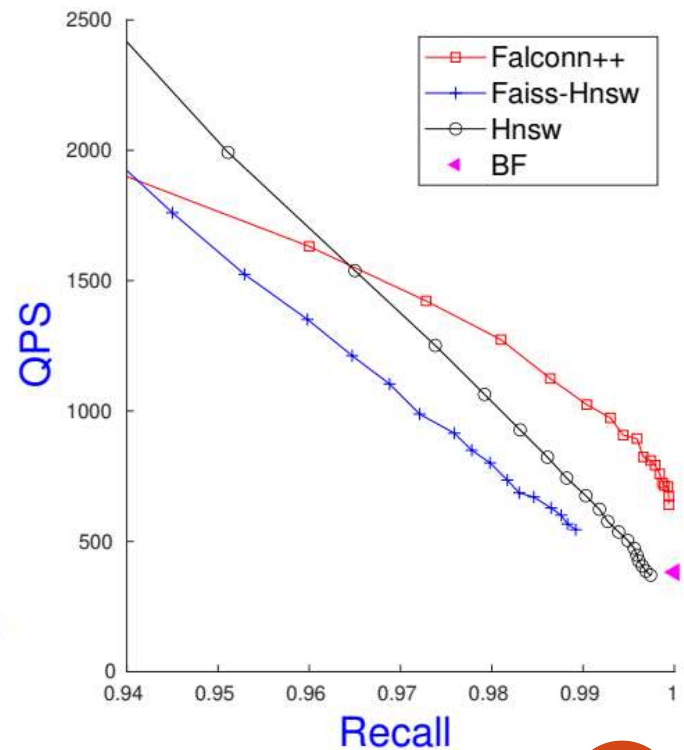
- Parameter settings:
  - Hnsw: **ef-index = 200, M = 512,** vary **ef-query**
  - Falconn++: **D = 256, L = 350, α = 0.01, iProbes = 3**, vary **qProbes**

Indexing

Space: **5.4GB**
Hnsw: **13.7 mins**
Falconn++: **1.1 mins**



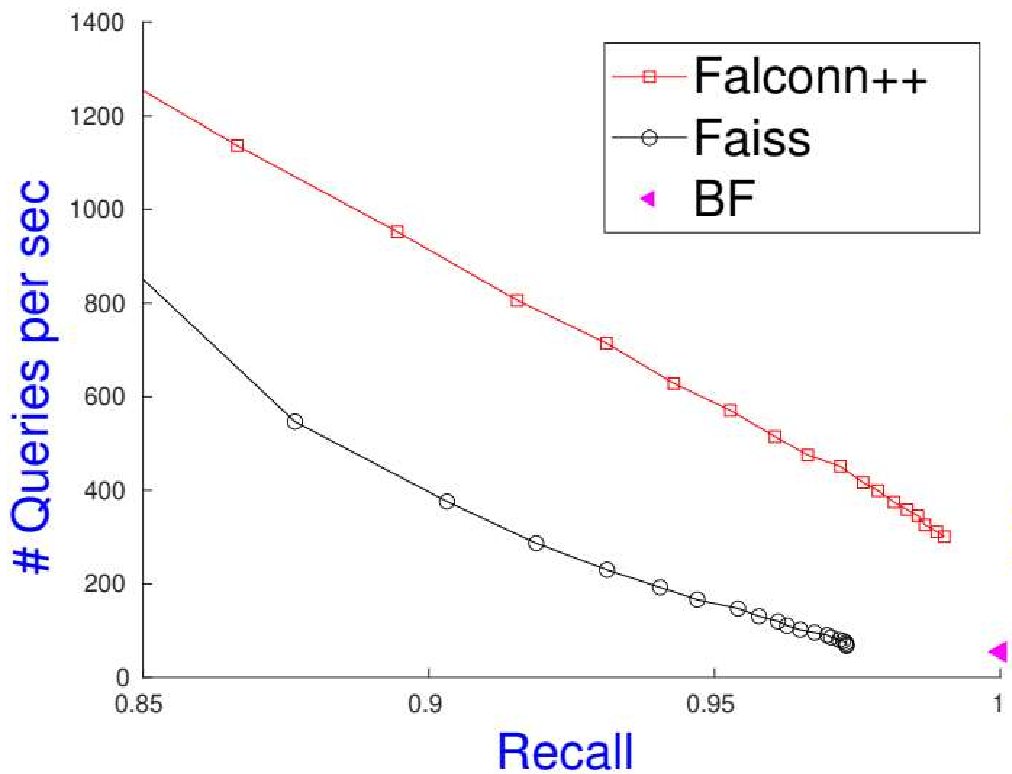
(c) Glove200, 1 thread



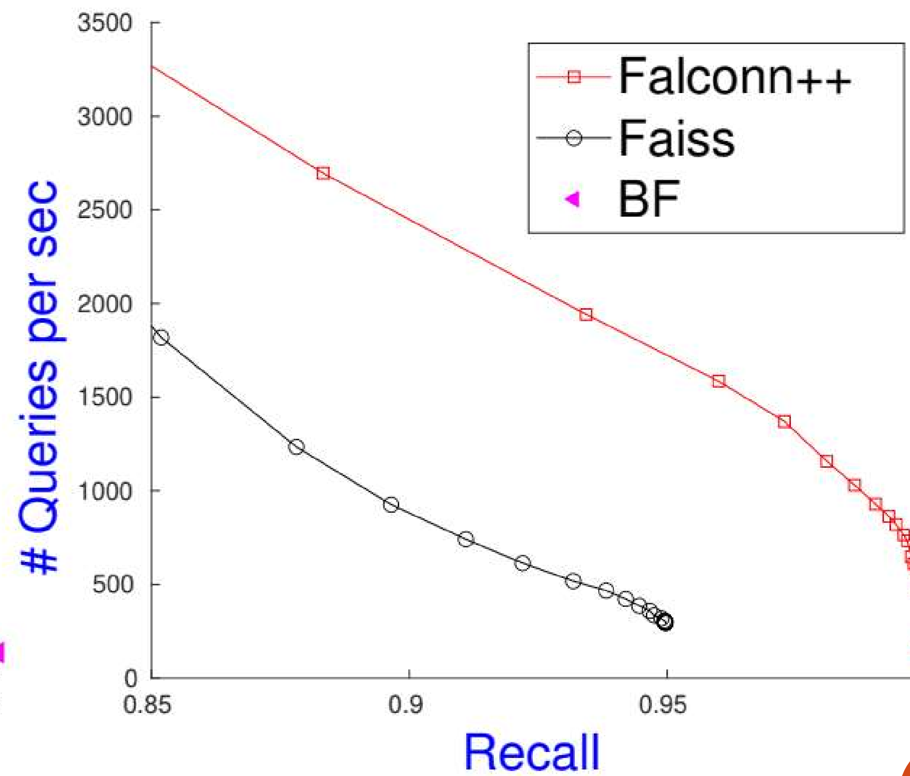
(d) Glove200, 64 threads

Falconn++ vs Faiss

- Parameter settings:
  - Faiss: **m = 256, nlist=1000, 8 bits/centroid**, vary **probe**
  - Falconn++: **D = 256, L = 350, α = 0.01, iProbes = 3**, vary **qProbes**



(b) Glove300



(c) Glove200

Open problem

- Practical LSH & LSF pattern:
  - Existing for Euclidean distance with $\boldsymbol{\rho = 1/c^2}$

- Characterize # random projections **D**
- Characterize the scaling factor $\boldsymbol{\alpha}$