Locality-sensitive hashing indexing schemes for compositional assembly-free metagenomics data

2018-11-28 | InTheArt@CEA | Jacques-Henri Sublemontier

OUTLINE

Metagenomics

1. Context 2. Clustering metagenomic reads 3. Assessing inter-reads connectivity

Locality Sensitive Hashing

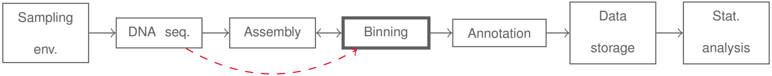
1. SimHash - LSH *via* random projections 2. Orthonormality and LSH 3. Structured binary embeddings 4. Fast Johnson-Lindenstrauss transform 5. Preconditioned binary embeddings 6. Discussion

LSH for metagenomics

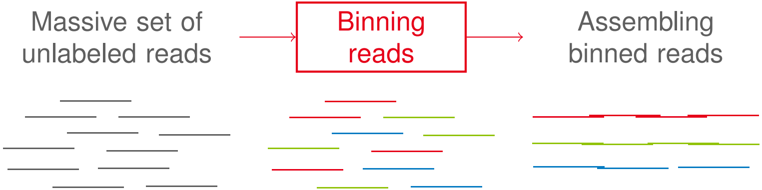
1. Indexing compositional profiles 2. Datasets 3. Implementation 4. Evaluations 5. k-NN retrieval performance 6. Bin sizes balance 7. Binning performance 8. Time performance

Conclusion and future work

CONTEXT

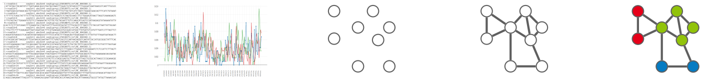


We consider *de novo* assembly-free binning



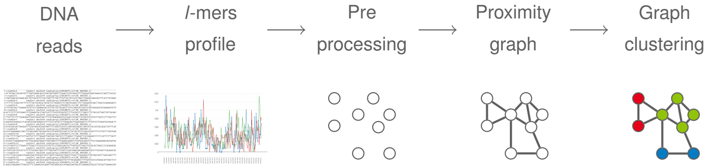
CLUSTERING METAGENOMIC READS - PROPOSED APPROACH

DNA reads → *l*-mers profile → Pre processing → Proximity graph → Graph clustering



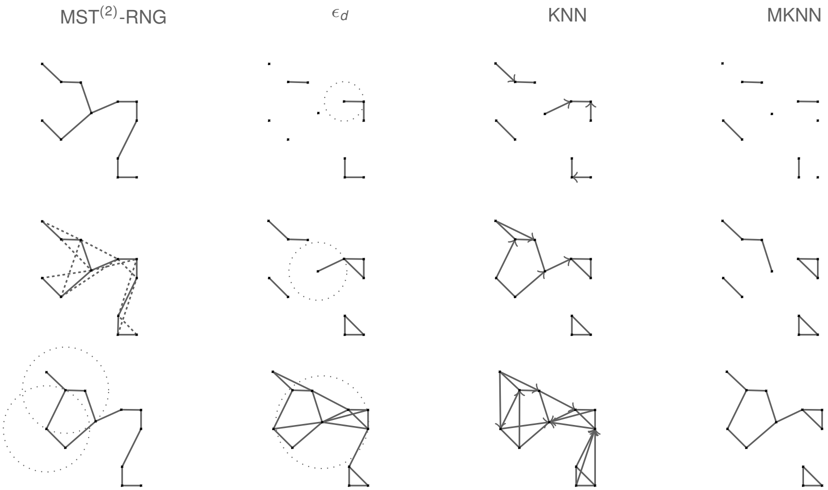
- Given a sparse structure (*e.g.* proximity graph) between reads, one can design specific and efficient clustering algorithm to partition the graph.

CLUSTERING METAGENOMIC READS - PROPOSED APPROACH



- Given a sparse structure (*e.g.* proximity graph) between reads, one can design specific and efficient clustering algorithm to partition the graph.
- Can we compute proximity graphs efficiently ?

ASSESSING INTER-READS CONNECTIVITY - PROXIMITY GRAPHS



PROXIMITY GRAPHS - A COMMON PROBLEM

The Nearest Neighbor Search (NNS)

Given a dataset $\mathcal{X} \in \mathbb{R}^p$ with $|\mathcal{X}| = n$ and $x_i \in \mathcal{X}$, find x_j that is closer to x_i *i.e.*

$$x_j = \operatorname*{argmin}_{x \in \mathcal{X}} d(x, x_i)$$

- can be extended to k-Nearest Neighbor Search
- can be done with tree-based algorithm : k-d trees, R-trees, *etc.*
- but this is not suitable with high-dimensional datasets (*e.g.* ≥ 20)

We are further interested in Approximate Nearest Neighbor problem

LSH - GENERALITIES

Idea

Let $h(x_i)$ be a **binary word** (hashcode) of length b.

1. Devise a data structure $\mathcal{H}$: hashcode $\rightarrow$ list(objects) (hash table) *s.t.* :
   - two close objects (*e.g.* reads) x_1 and x_2 have the same hashcode ($h(x_1) = h(x_2)$),
   - two distant objects have different hashcodes ($h(x_1) \neq h(x_2)$);

   $\rightarrow$ with high probability
2. Then, $\forall x_i$:
   - searching nearest neighbours of any x_i in input space leads to searching closests x_j in hashcodes space associated with the hamming distance d_H

SIMHASH - LSH *VIA* RANDOM PROJECTIONS[1]

$$Y = GX \; ; \; \overbrace{X \in \mathbb{R}^{p\times n}}^{\text{Data matrix}} \; ; \; \overbrace{G \in \mathbb{R}^{b\times p}}^{\text{Projector}}$$

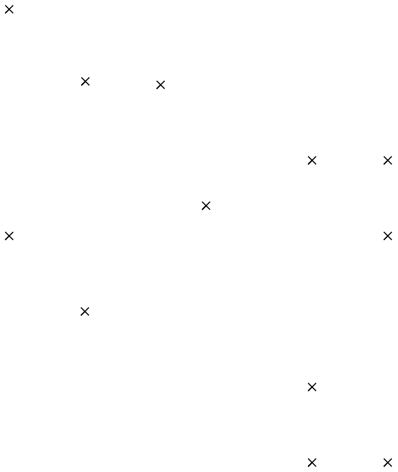
$$G = [\mathbf{g_1}, \ldots, \mathbf{g_b}]^\top \; ; \; \forall j \; \mathbf{g_j} \in \mathbb{R}^p, \; \mathbf{g_j} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I})$$

$$\forall \mathbf{x_i}, \; h(\mathbf{x_i}) = \mathbf{enc}(G\mathbf{x_i}) \; ; \; h(\mathbf{x_i}) = \overbrace{h_1(\mathbf{x_i}). \cdots .h_b(\mathbf{x_i})}^{\text{Concat. of } h_j(.)}$$

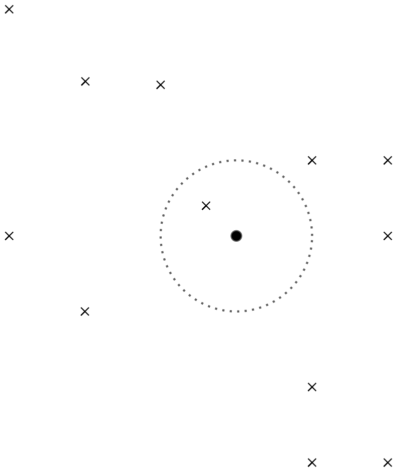
$$h_j(\mathbf{x}) = \mathrm{enc}(\mathbf{g_j}^\top \mathbf{x}) = \begin{cases} 1_2 & \text{if } \langle \mathbf{g_j}, \mathbf{x} \rangle \geq 0 \\ 0_2 & \text{otherwise} \end{cases}$$

[1] Moses S. Charikar. “Similarity Estimation Techniques from Rounding Algorithms”. In: *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*. STOC ’02. Montreal, Quebec, Canada: ACM, 2002, pp. 380–388. ISBN: 1-58113-495-9. DOI: 10.1145/509907.509965. URL: http://doi.acm.org/10.1145/509907.509965.

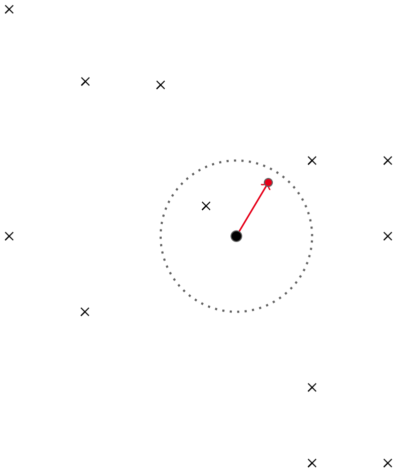
SIMHASH - LSH *VIA* RANDOM PROJECTIONS



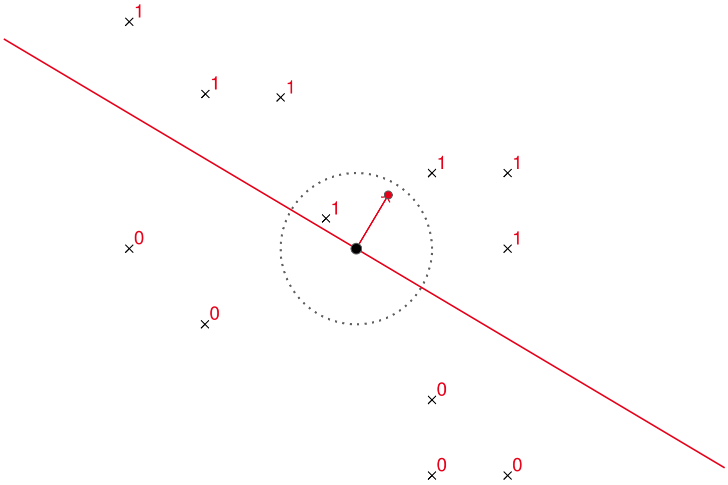
SIMHASH - LSH *VIA* RANDOM PROJECTIONS



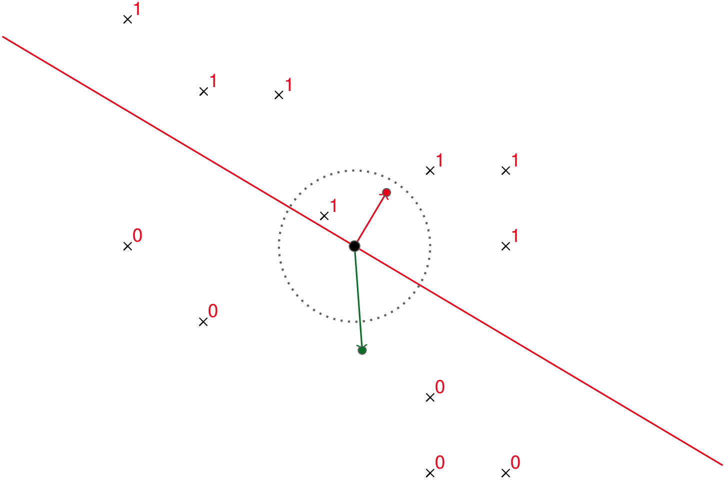
SIMHASH - LSH *VIA* RANDOM PROJECTIONS



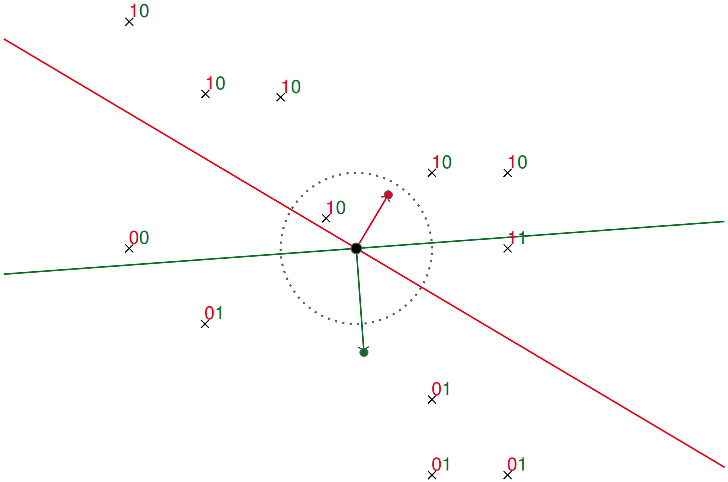
SIMHASH - LSH *VIA* RANDOM PROJECTIONS



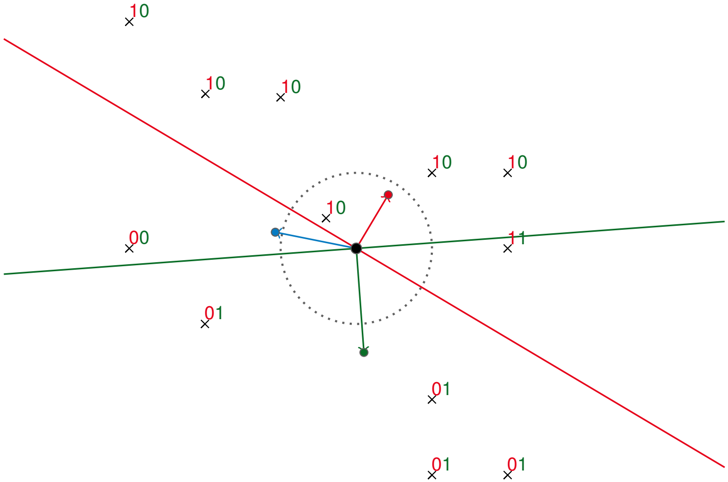
SIMHASH - LSH *VIA* RANDOM PROJECTIONS



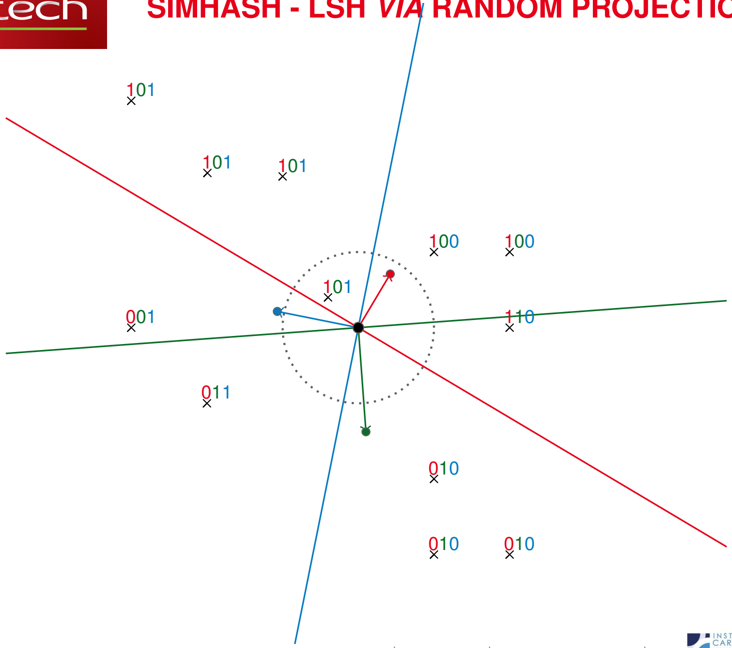
SIMHASH - LSH *VIA* RANDOM PROJECTIONS



SIMHASH - LSH *VIA* RANDOM PROJECTIONS



SIMHASH - LSH *VIA* RANDOM PROJECTIONS



SIMHASH - LSH *VIA* RANDOM PROJECTIONS

Operations

1. **Indexing**: store all $\mathbf{x_i}$ in $\mathcal{H}$ at key $h(\mathbf{x_i})$ (with chaining)
   1.1 **Projector sampling**: build G
   1.2 **Projecting**: compute $\mathbf{y_i} = G\mathbf{x_i}$, $\forall \mathbf{x_i}$
   1.3 **Rounding**: compute $h(\mathbf{x_i}) = \mathbf{enc}(\mathbf{y_i})$, $\forall \mathbf{x_i}$ (binarization)
2. **Querying**: retrieve k closest $\mathbf{x}$ from $\mathbf{x_{i'}}$: $\arg\min_{\mathbf{x}} {}_k \, d(\mathbf{x}, \mathbf{x_{i'}})$
   2.1 **In bin**: $\arg\min_{\mathbf{x}} {}_k \, d(\mathbf{x}, \mathbf{x_{i'}})$ *s.t.* $d_H(h(\mathbf{x}), h(\mathbf{x_{i'}})) = 0$ (if $|h(\mathbf{x_{i'}})| \geq k$)
   2.2 ***L*-radius bins**: $\arg\min_{\mathbf{x}} {}_k \, d(\mathbf{x}, \mathbf{x_{i'}})$ *s.t.* $d_H(h(\mathbf{x}), h(\mathbf{x_{i'}})) \leq L$ (else)
   2.3 **Search-based in bins (hamming) space** considering an appropriate order relation $\prec$ (heuristic)

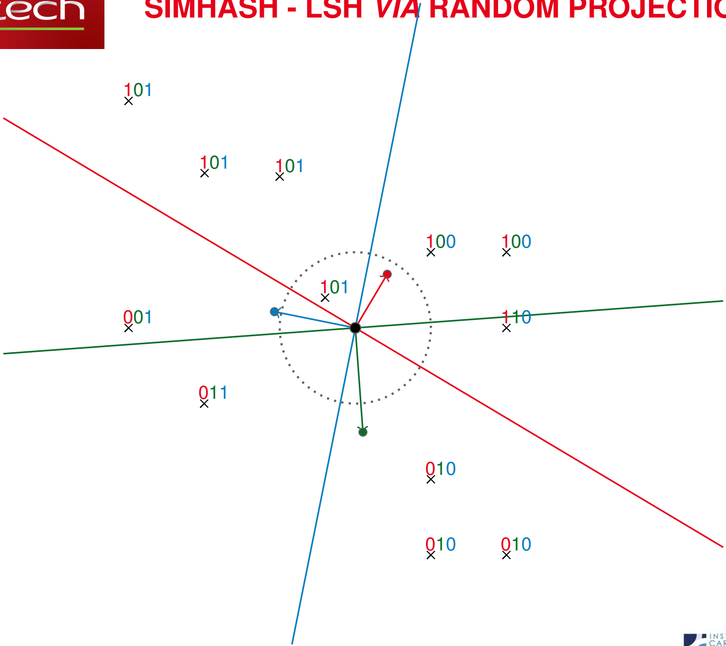
SIMHASH - LSH *VIA* RANDOM PROJECTIONS

Discussion

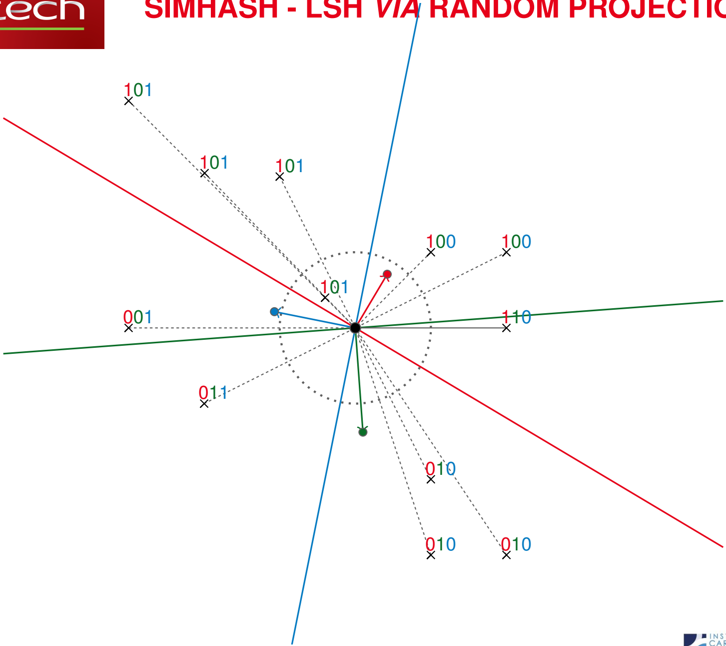
- G involved in indexing and querying
  $\rightarrow$ G should be fast sampled, with lowest storage cost
  $\rightarrow$ $G\mathbf{x}$ should be computed efficiently
- $\mathbb{P}[h_i(\mathbf{x_1}) = h_i(\mathbf{x_2})] = sim(\mathbf{x_1}, \mathbf{x_2}) = 1 - \frac{\theta(\mathbf{x_1}, \mathbf{x_2})}{\pi}$
- $\mathbb{E}[d_H(h(\mathbf{x_1}), h(\mathbf{x_2}))] = \frac{b}{\pi}\theta(\mathbf{x_1}, \mathbf{x_2}) = C\theta(\mathbf{x_1}, \mathbf{x_2})$

For any $\mathbf{x_1}, \mathbf{x_2}$, the hamming distance between their hashcodes is an estimate of their angle $\theta(\mathbf{x_1}, \mathbf{x_2})$
$\rightarrow$ For cosine or angular distances approximation.

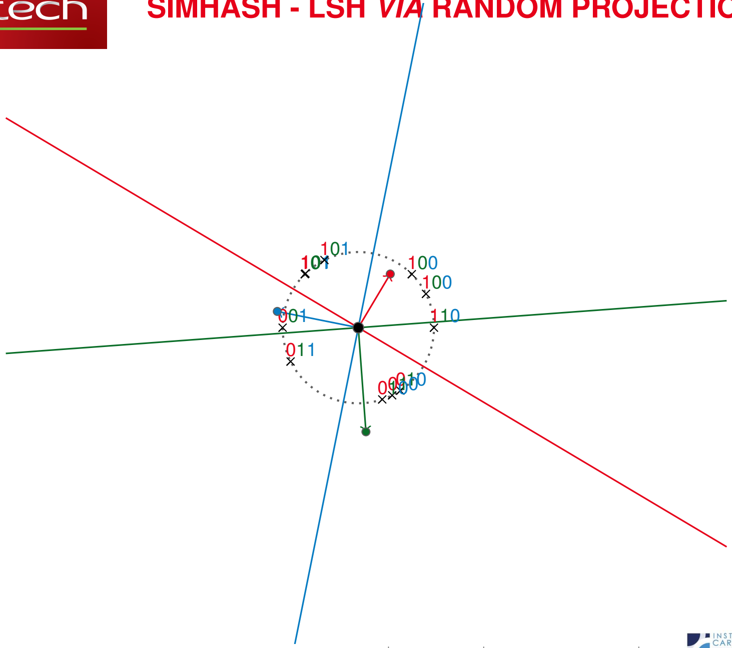
SIMHASH - LSH *VIA* RANDOM PROJECTIONS



SIMHASH - LSH *VIA* RANDOM PROJECTIONS



SIMHASH - LSH *VIA* RANDOM PROJECTIONS



ORTHOGONALITY AND LSH

$$Y = GX \ ; \ G \in \mathbb{R}^{b \times p}$$

G can be fully orthogonal

$$G = [\mathbf{g_1}, \ldots, \mathbf{g_b}]^\top \ ; \ \forall j \ \mathbf{g_j} \in \mathbb{R}^p, \ \mathbf{g_j} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I})$$

$GG^\top = \mathbf{I}$, with any process

ORTHOGONALITY AND LSH

$$Y = GX \; ; \; G \in \mathbb{R}^{b \times p}$$

G can be fully orthogonal

$$G = [\mathbf{g_1}, \dots, \mathbf{g_b}]^\top \; ; \; \forall j \; \mathbf{g_j} \in \mathbb{R}^p, \; \mathbf{g_j} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I})$$

$$GG^\top = \mathbf{I}, \text{ with any process}$$

⚠ $b \leq p$

ORTHOGONALITY AND LSH[2]

$$Y = GX \; ; \; G \in \mathbb{R}^{b \times p}, \; (GG^\top = \mathbf{I})$$

G can be batch-orthogonal, given L batches of size $\frac{b}{L} \leq p$

$$G' = [\mathbf{g_1}, \ldots, \mathbf{g_b}] \; ; \; \forall j \; \mathbf{g_j} \in \mathbb{R}^p, \; \mathbf{g_j} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I}) \; ; \; G' \in \mathbb{R}^{p \times b}$$

$$G = [Q_1, \ldots, Q_L]^\top \; ; \; \forall l \; Q_l \in \mathbb{R}^{p \times (\frac{b}{L})}$$

$$\forall l, \; G'_{\left[1:p, \; l \times \frac{b}{L}+1:(l+1)\frac{b}{L}\right]} = Q_l R_l \; ; \; Q_l Q_l^\top = \mathbf{I}$$

$$GG^\top \neq \mathbf{I}$$

[2] Jianqiu Ji et al. "Batch-orthogonal locality-sensitive hashing for angular similarity". In: *IEEE transactions on pattern analysis and machine intelligence* 36.10 (2014), pp. 1963–1974.

ORTHOGONALITY AND LSH[3]

$$Y = GX \; ; \; G \in \mathbb{R}^{b \times p}, \; GG^\top = \mathbf{I}$$

G can be orthogonal by kronecker product

$$G = G_1 \otimes \cdots \otimes G_L \; ; \; \forall l \; G_l \in \mathbb{R}^{2 \times 2}$$

$$\forall l, \; G_l = Q_l ; \; G'_l = Q_l R_l \; ; \; G'_l = [\mathbf{g_1}, \mathbf{g_2}], \; \mathbf{g_j} \sim \mathcal{N}_2(\mathbf{0}, \mathbf{I}), \; G'_l \in \mathbb{R}^{2 \times 2}$$

$$\forall l, \; G_l G_l^\top = \mathbf{I}$$

[3] X. Zhang et al. “Fast Orthogonal Projection Based on Kronecker Product”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2929–2937. DOI: 10.1109/ICCV.2015.335.

ORTHOGONALITY AND LSH[3]

$$Y = GX \; ; \; G \in \mathbb{R}^{b \times p}, \; GG^\top = \mathbf{I}$$

G can be orthonormal by kronecker product

$$G = G_1 \otimes \cdots \otimes G_L \; ; \; \forall l \; G_l \in \mathbb{R}^{2\times 2}$$

$$\forall l, \; G_l = Q_l; \; G_l' = Q_l R_l \; ; \; G_l' = [\mathbf{g_1}, \mathbf{g_2}], \; \mathbf{g_j} \sim \mathcal{N}_2(\mathbf{0}, \mathbf{I}), \; G_l' \in \mathbb{R}^{2\times 2}$$

$$\forall l, \; G_l G_l^\top = \mathbf{I}$$

⚠ Target G dimensions must be factorizable with a same number of factors L

[3] X. Zhang et al. "Fast Orthogonal Projection Based on Kronecker Product". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2929–2937. DOI: 10.1109/ICCV.2015.335.

ORTHOGONALITY AND LSH

Discussion

- Reduction of $Var[d_H(h(x_1), h(x_2))]$ (Batch-orthonormal LSH)
  $\rightarrow$ Leads to better accuracy in angle estimation, thus in approximate nearest neighbor query
  $\rightarrow$ The resulting binary codes are considered more informative
- Orthonormal matrix sampling from $\mathcal{O}(p^3)$ to $\mathcal{O}(\log(p))$ (Kronecker-based approach, for very small element matrices (2×2)). Also space complexity is $\mathcal{O}(\log p)$
  $\rightarrow$ Projection Gx_i can be done in $\mathcal{O}(p \log p)$
  $\rightarrow$ Care for existence of G built from kronecker product for specific target dimensions b and p

STRUCTURED BINARY EMBEDDINGS[4]

$$Y = GX \; ; \; G \in \mathbb{R}^{b \times p}, G = G'D$$

G' can be circulant (or Tœplitz or Hankel)

$$D = \operatorname{diag}(d_1, \ldots, d_p) \; ; \; d_j \sim \mathit{Rademacher}$$

$$\mathbf{g} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I}) \; ; \; \mathbf{g} = (g_1, \ldots, g_p)$$

$$G' = \begin{bmatrix} g_1 & g_2 & \cdots & g_{p-1} & g_p \\ g_p & g_1 & g_2 & & g_{p-1} \\ \vdots & g_p & g_1 & \ddots & \vdots \\ g_{p-b+3} & & \ddots & \ddots & g_2 \\ g_{p-b+2} & g_{p-b+3} & \cdots & g_{p-b} & g_{p-b+1} \end{bmatrix} \; ;$$

[4] Felix Yu et al. "Circulant Binary Embedding". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, 2014, pp. 946–954. URL: http://proceedings.mlr.press/v32/yub14.html.

Discussion

- $\mathcal{O}(p \log p)$ time complexity of the projection Gx_i (*via* FFT)
- $\mathcal{O}(p)$ space complexity
- $\mathbf{g_1}, \ldots, \mathbf{g_b}$ (rows of G) are not independent. Realizations are not orthogonal but unlikely correlated.

Expand G' in $C_p \in \mathbb{R}^{p \times p}$. We have $C_p = F_p^* . diag(F_p\mathbf{g})F_p$, then compute $\mathbf{y} = C_p\mathbf{x_i} = F_p^* . diag(F_p\mathbf{g})F_p\mathbf{x_i}$:

1. $\mathbf{a} = F_p\mathbf{x_i}$ ($fft(\mathbf{x_i})$ in $\mathcal{O}(p \log p)$)
2. $\mathbf{b} = F_p\mathbf{g}$ ($fft(\mathbf{g})$ in $\mathcal{O}(p \log p)$)
3. $\mathbf{z}^\top = [a_1 b_1, \cdots, a_p b_p]$ in $\mathcal{O}(p)$
4. $\mathbf{y} = \frac{1}{p} F_p^* \mathbf{z}$ ($ifft(\mathbf{z})$ in $\mathcal{O}(p \log p)$)

FJLT-BASED LSH

Lemma (Johnson-Lindenstauss[5,6])

Given $\epsilon > 0$ *and a positive integer* $b \geq \epsilon^{-2} \log n$*. For any set* $\mathcal{X}$ *of points in p dimensions,* $\exists \mathcal{G} : \mathbb{R}^p \mapsto \mathbb{R}^b$ *projection from p dimensions to b dimensions* s.t. $\forall x_1, x_2 \in \mathcal{X}$,

$$(1-\epsilon)||x_1 - x_2||_2^2 \leq ||\mathcal{G}(x_1) - \mathcal{G}(x_2)||_2^2 \leq (1+\epsilon)||x_1 - x_2||_2^2$$

[5] Dimitris Achlioptas. "Database-friendly random projections: Johnson-Lindenstrauss with binary coins". In: *Journal of Computer and System Sciences* 66.4 (2003). Special Issue on PODS 2001, pp. 671 –687. ISSN: 0022-0000. DOI: `https://doi.org/10.1016/S0022-0000(03)00025-4`. URL: `http://www.sciencedirect.com/science/article/pii/S0022000003000254`.

[6] William Johnson and Joram Lindenstrauss. "Extensions of Lipschitz mappings into a Hilbert space". In: *Conference in modern analysis and probability (New Haven, Conn., 1982)*. Vol. 26. Contemporary Mathematics. American Mathematical Society, 1984, pp. 189–206.

FJLT-BASED LSH[7]

$$Y = GX \; ; \; G \in \mathbb{R}^{b \times p} \; ; \; G = PHD$$

$$D = \mathrm{diag}(d_1, \ldots, d_p) \; ; \; d_j \sim \mathit{Rademacher}$$

$$H_p = H_2 \otimes H_{2^{m-1}} \; ; \; H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \; ; \; H_1 = \begin{bmatrix} 1 \end{bmatrix} \; ; \; H = \frac{1}{\sqrt{p}} H_p$$

$$P \in \mathbb{R}^{b \times p} \; ; \; P_{ij} \sim \mathit{Bernoulli}(q) \times \mathcal{N}(0, q^{-1})$$

$$q = \min \left\{ \Theta \left(\frac{\epsilon^{z-2} \log^z n}{p} \right), 1 \right\} \; ; \; z \in \{1, 2\} \; (||.||_z)$$

---

[7] Nir Ailon and Bernard Chazelle. “The Fast Johnson-Lindenstrauss Transform and Approximate Nearest Neighbors”. In: *SIAM J. Comput.* 39.1 (May 2009), pp. 302–322. ISSN: 0097-5397. DOI: 10.1137/060673096. URL: http://dx.doi.org/10.1137/060673096.

FJLT-BASED LSH[7]

$$Y = GX \; ; \; G \in \mathbb{R}^{b \times p} \; ; \; G = PHD$$

$$D = \mathrm{diag}(d_1, \ldots, d_p) \; ; \; d_j \sim \mathit{Rademacher}$$

$$H_p = H_2 \otimes H_{2^{m-1}} \; ; \; H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \; ; \; H_1 = \begin{bmatrix} 1 \end{bmatrix} \; ; \; H = \frac{1}{\sqrt{p}} H_p$$

$$P \in \mathbb{R}^{b \times p} \; ; \; P_{ij} \sim \mathit{Bernoulli}(q) \times \mathcal{N}(0, q^{-1})$$

$$q = \min \left\{ \Theta \left(\frac{\epsilon^{z-2} \log^z n}{p} \right), 1 \right\} \; ; \; z \in \{1, 2\} \; (||.||_z)$$

⚠ Target G dimension p must be a power of 2

[7] Nir Ailon and Bernard Chazelle. "The Fast Johnson-Lindenstrauss Transform and Approximate Nearest Neighbors". In: *SIAM J. Comput.* 39.1 (May 2009), pp. 302–322. ISSN: 0097-5397. DOI: 10.1137/060673096. URL: http://dx.doi.org/10.1137/060673096.

FJLT-BASED LSH

The role of *HD*

$$\max_{x \in \mathcal{X} \subset \mathbb{R}^p} ||HDx||_\infty = \mathcal{O}(p^{-1/2}\sqrt{\log n}) \text{ with high probability.}$$

- $\forall x \in \mathcal{X},\ HDx$ become smooth
- HDx can be computed in $\mathcal{O}(p \log p)$
- $HD(HD)^\top = \mathbf{I}$

FJLT-BASED LSH

The role of HD

$$\max_{x \in \mathcal{X} \subset \mathbb{R}^p} ||HDx||_\infty = \mathcal{O}(p^{-1/2}\sqrt{\log n}) \text{ with high probability.}$$

– $p = 2^m$ by definition of H_p

$$H_1 = \begin{bmatrix} 1 \end{bmatrix} \; ; \; H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \; ; \; H_p = H_2 \otimes H_{2^{m-1}}$$

Discussion

- Originally developped to find l_2 and l_1 (l_z) embeddings, without rounding (binarization *via* sign operator)
- One can binarize the output
- $\mathcal{O}(p \log p + \min\{p\varepsilon^{-2} \log n, \varepsilon^{z-4} \log^{z+1} n\})$ time complexity
- *HD* as a preconditioner (smoother) $\rightarrow$ also called ROS[8]
- *H* represents Walsh-Hadamard matrix

[8] F. Pourkamali-Anaraki and S. Becker. "Preconditioned Data Sparsification for Big Data With Applications to PCA and K-Means". In: *IEEE Transactions on Information Theory* 63.5 (2017), pp. 2954–2974. ISSN: 0018-9448. DOI: 10.1109/TIT.2017.2672725.

PRECONDITIONED BINARY EMBEDDINGS[9,10,11]

$$Y = GX \; ; \; G \in \mathbb{R}^{b \times p} \; ; \; G = G_{struct} HD$$

- G_{struct} may be any structured or unstructured gaussian
- HD is the preconditioner

$$G_{struct} = HGP_1{}^{9} \; ; \; G_{struct} = HD_3 HD_2{}^{10} \; ; \; G_{struct} = P_2 TD_2 P_1{}^{11}$$

- $P_i \in \mathbb{R}^{b' \times p}$ ($b \leq b' \leq p$) is a permutation matrix with rows selection ($b' = b$ when needed)

[9] Anirban Dasgupta, Ravi Kumar, and Tamas Sarlos. "Fast Locality-sensitive Hashing". In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '11. San Diego, California, USA: ACM, 2011, pp. 1073–1081. ISBN: 978-1-4503-0813-7. DOI: 10.1145/2020408.2020578. URL: http://doi.acm.org/10.1145/2020408.2020578.

[10] Alexandr Andoni et al. "Practical and Optimal LSH for Angular Distance". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 1225–1233. URL: http://dl.acm.org/citation.cfm?id=2969239.2969376.

[11] Xinyang Yi, Constantine Caramanis, and Eric Price. "Binary Embedding: Fundamental Limits and Fast Algorithm". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 2162–2170. URL: http://proceedings.mlr.press/v37/yi15.html.

PRECONDITIONED BINARY EMBEDDINGS

Discussion

- Lower bound on b length of binary word when using unstructured G to achieve a specific distortion δ for a given probability.

Given any $f : \mathbb{S}^{p-1} \rightarrow \{0,1\}^b$ and $g : \{0,1\}^b \times \{0,1\}^b \rightarrow \mathbb{R}$, *s.t.*

$\forall x_i, x_j \in \mathbb{S}^{p-1}, \; |g(f(x_i), f(x_j)) - d(x_i, x_j)| \leq \delta$ with probability $(1 - \varepsilon)$

Then

$$b = \Omega(\delta^{-2} \log(n/\varepsilon))$$

DISCUSSION

- Focus on high dimensional data
- Multiple hash tables[12] *vs.* multi-probing
- Data-independent LSH *vs.* Data-dependent LSH[13,14]
- Fully randomized *vs.* learned hash functions
- Short *vs.* long codes
- Preconditionner (*HD*) + Structured (fast) subspace projection is a regular scheme

[12] M. Norouzi, A. Punjani, and D. J. Fleet. "Fast Exact Search in Hamming Space With Multi-Index Hashing". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.6 (2014), pp. 1107–1119. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2013.231.

[13] J. Wang et al. "Learning to Hash for Indexing Big Data – A Survey". In: *Proceedings of the IEEE* 104.1 (2016), pp. 34–57. ISSN: 0018-9219. DOI: 10.1109/JPROC.2015.2487976.

[14] J. Wang et al. "A Survey on Learning to Hash". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP.99 (2018), pp. 1–1. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2699960.

INDEXING COMPOSITIONAL PROFILES

ATTGAC...

4-mer	frequency
...	...
ATTG/CAAT	0
TTGA/TCAA	0
TGAC/GTCA	0
...	...

INDEXING COMPOSITIONAL PROFILES

ATTGAC...

4-mer	frequency
...	...
ATTG/CAAT	+1
TTGA/TCAA	0
TGAC/GTCA	0
...	...

INDEXING COMPOSITIONAL PROFILES

ATTGAC. . .

4-mer	frequency
. . .	. . .
ATTG/CAAT	1
TTGA/TCAA	+1
TGAC/GTCA	0
. . .	. . .

INDEXING COMPOSITIONAL PROFILES

ATTGAC…

4-mer	frequency
…	…
ATTG/CAAT	1
TTGA/TCAA	1
TGAC/GTCA	+1
…	…

INDEXING COMPOSITIONAL PROFILES

ATTGAC. . .

4-mer	frequency
. . .	. . .
ATTG/CAAT	1
TTGA/TCAA	1
TGAC/GTCA	1
. . .	. . .

INDEXING COMPOSITIONAL PROFILES

ATTGAC…

4-mer	frequency
…	…
ATTG/CAAT	1
TTGA/TCAA	1
TGAC/GTCA	1
…	…

For each read compositional profile x_i:

1. $x_i \leftarrow x_i + \mathbf{1}$ (add-one smoothing)
2. $x_i \leftarrow x_i / \mathbf{gm}(x_i)$ (centered log-ratio[15])

$\mathbf{gm}(x_i)$ refers to the geometric mean of x_i

[15] J Aitchison. *The Statistical Analysis of Compositional Data*. London, UK, UK: Chapman & Hall, Ltd., 1986. ISBN: 0-412-28060-4.

INDEXING COMPOSITIONAL PROFILES

LSH & Metagenomics: Practical considerations

- Leverage *HD* preconditioner for $p \neq 2^m$

$$H_{p_{blk}} = \begin{bmatrix} \frac{1}{\sqrt{2^{t_1}}} H_{2^{t_1}} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\sqrt{2^{t_\tau}}} H_{2^{t_\tau}} \end{bmatrix} ; \; p = \sum_{i=t_1}^{t_\tau} 2^i$$

LSH & Metagenomics: Practical considerations

- Leverage *HD* preconditioner for $p \neq 2^m$

example: p = 136 (4-mers compositionnal vector dimension)

$$H_{136_{blk}} = \begin{bmatrix} \frac{1}{\sqrt{128}} H_{128} & 0 \\ 0 & \frac{1}{\sqrt{8}} H_8 \end{bmatrix} \; ; \; p = 128 + 8 = 2^7 + 2^3$$

Then select a random permutation of b rows (out of p) of (randomly column permuted) $H_{p_{blk}}$ to get $H_{p_{blk}} \in \mathbb{R}^{b \times p}$

INDEXING COMPOSITIONAL PROFILES

LSH & Metagenomics: Practical considerations

- Leverage *HD* preconditioner for $p \neq 2^m$
- Build a valid kronecker-product based $G \in \mathbb{R}^{b \times p}$ for any b and p

$$G_{(b \times p)_{blk}} = \begin{bmatrix} G_{(b_1 \times p_1(b_1))} & \cdots & G_{(b_1 \times p_{\tau_2}(b_1))} \\ \vdots & \vdots & \vdots \\ G_{(b_{\tau_1} \times p_1(b_{\tau_1}))} & \cdots & G_{(b_{\tau_1} \times p_{\tau_2}(b_{\tau_1}))} \end{bmatrix}$$

$$b = \sum_{r=1}^{\tau_1} b_r, \ \forall r \ b_r = 2^{m_r}, \ p = \sum_{c=1}^{\tau_2} p_c(b_r), \ p_c(b_r) = 2^{m_c}$$

INDEXING COMPOSITIONAL PROFILES

LSH & Metagenomics: Practical considerations

- Leverage *HD* preconditioner for $p \neq 2^m$
- Build a valid kronecker-product based $G \in \mathbb{R}^{b \times p}$ for any b and p

$$G_{(b \times p)_{blk}} = \begin{bmatrix} G_{(b_1 \times p_1(b_1))} & \cdots & G_{(b_1 \times p_{\tau_2}(b_1))} \\ \vdots & \vdots & \vdots \\ G_{(b_{\tau_1} \times p_1(b_{\tau_1}))} & \cdots & G_{(b_{\tau_1} \times p_{\tau_2}(b_{\tau_1}))} \end{bmatrix}$$

$$G_{(b_r \times p_c(b_r))} = G_{(b_r \times p_c(b_r))_1} \otimes \cdots \otimes G_{(b_r \times p_c(b_r))_L}$$

LSH & Metagenomics: Practical considerations

- Leverage *HD* preconditioner for $p \neq 2^m$
- Build a valid kronecker-product based $G \in \mathbb{R}^{b \times p}$ for any b and p

example: p = 136 (4-mers compositionnal vector dimension), $b = 12$

$$G_{(12 \times 136)_{blk}} = \begin{bmatrix} G_{(8 \times 128)} & G_{(8 \times 8)} \\ G_{(4 \times 128)} & G_{(4 \times 8)} \end{bmatrix}$$

$$G_{(8 \times 8)} = G_{(2 \times 2)} \otimes G_{(2 \times 2)} \otimes G_{(2 \times 2)} \; ; \; G_{(4 \times 128)} = G_{(2 \times 16)} \otimes G_{(2 \times 8)}$$

LSH & Metagenomics: Practical considerations

- Leverage *HD* preconditioner for $p \neq 2^m$
- Build a valid kronecker-product based $G \in \mathbb{R}^{b \times p}$ for any b and p

example: p = 136 (4-mers compositionnal vector dimension), $b = 12$

$$G_{(12 \times 136)_{blk}} = \begin{bmatrix} G_{(8 \times 128)} & G_{(8 \times 8)} \\ G_{(4 \times 128)} & G_{(4 \times 8)} \end{bmatrix}$$

$$G_{(8 \times 8)} = G_{(2 \times 2)} \otimes G_{(2 \times 2)} \otimes G_{(2 \times 2)} \; ; \; G_{(4 \times 128)} = G_{(2 \times 16)} \otimes G_{(2 \times 8)}$$

$$G^{\top}_{(b' \times p')} G_{(b' \times p')} = I \; (G_{(b' \times p')} \; QR \text{ dec.}) \; ; \; G^{\top}_{(b \times p)} G_{(b \times p)} \approx I$$

DATASETS

n: number of reads. Reads are $\sim$ 600 b.p. length

Name	n	$n.species$	$n.genus$	$coverage$
MC5	25K	5	3	1X
MC10	50K	10	10	1X
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
MC100	500K	100	$\leq$ 67	1X
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
MC700	35M	700	$\leq$ 321	10X

- *n.species*: number of classes at species level
- *n.genus*: number of classes at genus level

IMPLEMENTATION

Cluster Node N0			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

IMPLEMENTATION

Cluster Node N0			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

Cluster Node N1			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

Cluster Node N2			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

Cluster Node N3			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

Cluster Node N4			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

Cluster Node N5			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

IMPLEMENTATION

Cluster Node N0			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

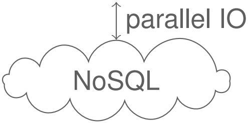
Cluster Node N1			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

Cluster Node N2			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

Cluster Node N3			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

Cluster Node N4			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

Cluster Node N5			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}



IMPLEMENTATION

Cluster Node N0			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

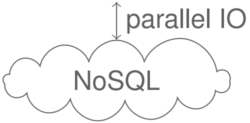
Cluster Node N1			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

Cluster Node N2			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

Cluster Node N3			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

Cluster Node N4			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}

Cluster Node N5			
Socket 0		Socket 1	
T_0	T_1	T_0	T_1
p_0	p_{16}	p_1	p_{17}
p_2	p_{18}	p_3	p_{19}
p_4	p_{20}	p_5	p_{21}
p_6	p_{22}	p_7	p_{23}
p_8	p_{24}	p_9	p_{25}
p_{10}	p_{26}	p_{11}	p_{27}
p_{12}	p_{28}	p_{13}	p_{29}
p_{14}	p_{30}	p_{15}	p_{31}



+ Randomized (data-independent) LSH are 1-pass EP jobs !

DESIRED PROPERTIES

- Ability to retrieve true neighbors of any read, within a bin or nearby bins
- Ability to have balanced bin sizes
- Ability to correctly bin reads *w.r.t.* species/genus
- Ability to fast compute the index

K-NN RETRIEVAL PERFORMANCE

Protocol

1. Build exact k-NN graph $\mathcal{G}_j$

K-NN RETRIEVAL PERFORMANCE

Protocol

1. Build exact k-NN graph $\mathcal{G}_j$
2. Compute retrieved reads around x_i: $Ret_L(x_i, \mathcal{G}_j)$

$$Ret_L(x_i, \mathcal{G}_j) = \{(x_i, y) \mid d_H(h_s(y), h_s(x_i)) \leq L\} \cap E(\mathcal{G}_i)$$

$E(\mathcal{G}_j)$ is the set of edges of $\mathcal{G}_j$ as pairs of reads.

K-NN RETRIEVAL PERFORMANCE

Protocol

1. Build exact k-NN graph $\mathcal{G}_j$
2. Compute retrieved reads around x_i: $Ret_L(x_i, \mathcal{G}_j)$
3. Compute $LB_L(\mathcal{H}_s)$, $EB_L(\mathcal{H}_s)$, $UB_L(\mathcal{H}_s)$, $Ret_{succ_L}(\mathcal{H}_s, \mathcal{G}_j)$

$$LB_L(\mathcal{H}_s) = \frac{1}{n} \min_{x_i} |\{x \mid d_H(h_s(x), h_s(x_i)) \leq L\}|$$

$\rightarrow$ corresponds to the (empirical) smallest search space ratio, parameterized by L radius of a ball in hamming space.

K-NN RETRIEVAL PERFORMANCE

Protocol

1. Build exact k-NN graph $\mathcal{G}_j$
2. Compute retrieved reads around x_i: $Ret_L(x_i, \mathcal{G}_j)$
3. Compute $LB_L(\mathcal{H}_s)$, $EB_L(\mathcal{H}_s)$, $UB_L(\mathcal{H}_s)$, $Ret_{succ_L}(\mathcal{H}_s, \mathcal{G}_j)$

$$EB_L(\mathcal{H}_s) = \frac{1}{n} \underset{x_i}{\text{avg}} \left| \{ x \mid d_H(h_s(x), h_s(x_i)) \leq L \} \right|$$

$\rightarrow$ corresponds to the (empirical) expected search space ratio, parameterized by L radius of a ball in hamming space.

K-NN RETRIEVAL PERFORMANCE

Protocol

1. Build exact k-NN graph $\mathcal{G}_j$
2. Compute retrieved reads around x_i: $Ret_L(x_i, \mathcal{G}_j)$
3. Compute $LB_L(\mathcal{H}_s)$, $EB_L(\mathcal{H}_s)$, $UB_L(\mathcal{H}_s)$, $Ret_{succ_L}(\mathcal{H}_s, \mathcal{G}_j)$

$$UB_L(\mathcal{H}_s) = \frac{1}{n} \max_{x_i} |\{x \mid d_H(h_s(x), h_s(x_i)) \leq L\}|$$

$\rightarrow$ corresponds to the (empirical) largest search space ratio, parameterized by L radius of a ball in hamming space.

K-NN RETRIEVAL PERFORMANCE

Protocol

1. Build exact k-NN graph $\mathcal{G}_j$
2. Compute retrieved reads around x_i: $Ret_L(x_i, \mathcal{G}_j)$
3. Compute $LB_L(\mathcal{H}_s)$, $EB_L(\mathcal{H}_s)$, $UB_L(\mathcal{H}_s)$, $Ret_{succ_L}(\mathcal{H}_s, \mathcal{G}_j)$

$$Ret_{succ_L}(\mathcal{H}_s, \mathcal{G}_j) = \frac{1}{n.k} \sum_{x_i} |Ret_L(x_i, \mathcal{G}_j)|$$

$\rightarrow$ corresponds to the set of correctly identified neighborhood relations (True Positives) among all reads.

K-NN RETRIEVAL PERFORMANCE

Protocol

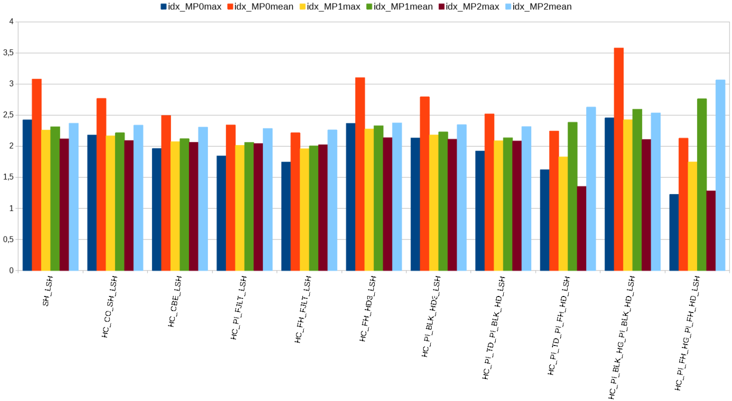
1. Build exact k-NN graph $\mathcal{G}_j$
2. Compute retrieved reads around x_i: $Ret_L(x_i, \mathcal{G}_j)$
3. Compute $LB_L(\mathcal{H}_s)$, $EB_L(\mathcal{H}_s)$, $UB_L(\mathcal{H}_s)$, $Ret_{succ_L}(\mathcal{H}_s, \mathcal{G}_j)$

$$EBI_L = \frac{Ret_{succ_L}(\mathcal{H}_s, \mathcal{G}_j)}{EB_L(\mathcal{H}_s)} \; ; \; UBI_L = \frac{Ret_{succ_L}(\mathcal{H}_s, \mathcal{G}_j)}{UB_L(\mathcal{H}_s)}$$

$\rightarrow$ also called idx_MPLmean (idx_MPLmax respectively) in the following plots. It corresponds to a *MultiProbing ANN performance index*.

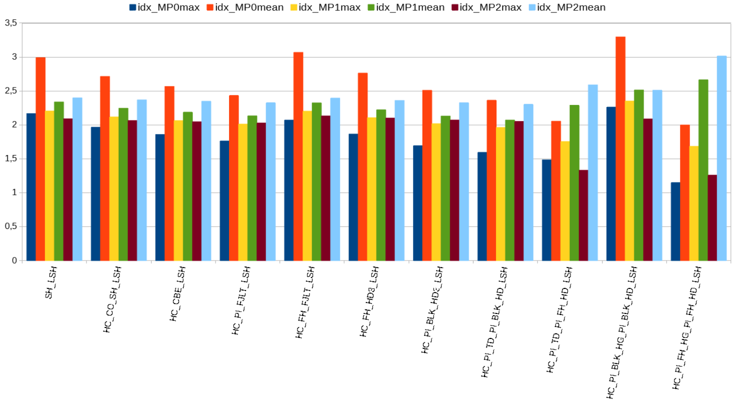
K-NN RETRIEVAL PERFORMANCE

MC5 - 5 species, 3 genuses, 25K reads, $b = 4$, 5-NN



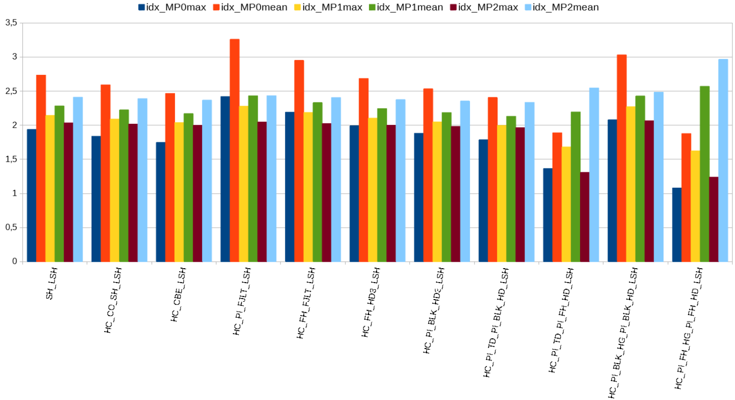
K-NN RETRIEVAL PERFORMANCE

MC5 - 5 species, 3 genuses, 25K reads, $b = 4$, 10-NN



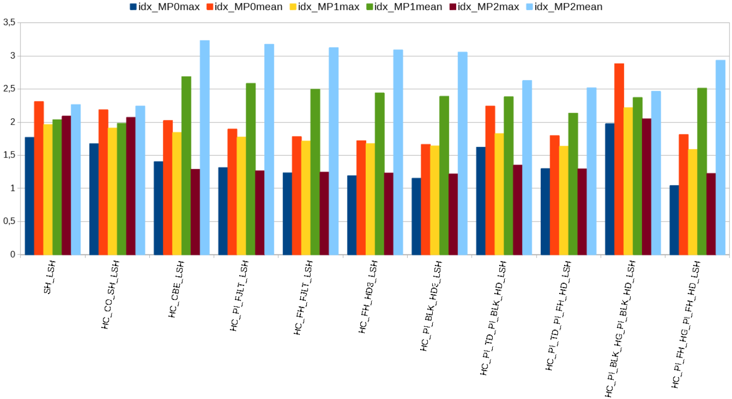
K-NN RETRIEVAL PERFORMANCE

MC5 - 5 species, 3 genuses, 25K reads, $b = 4$, 25-NN



K-NN RETRIEVAL PERFORMANCE

MC5 - 5 species, 3 genuses, 25K reads, $b = 4$, 50-NN

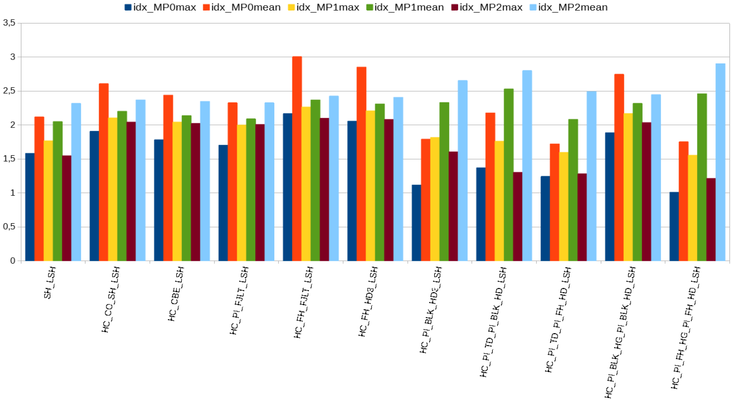


FROM RESEARCH TO INDUSTRY

cea tech

K-NN RETRIEVAL PERFORMANCE

MC5 - 5 species, 3 genuses, 25K reads, $b = 4$, 100-NN



INSTITUT CARNOT TN@UPSaclay

université PARIS-SACLAY

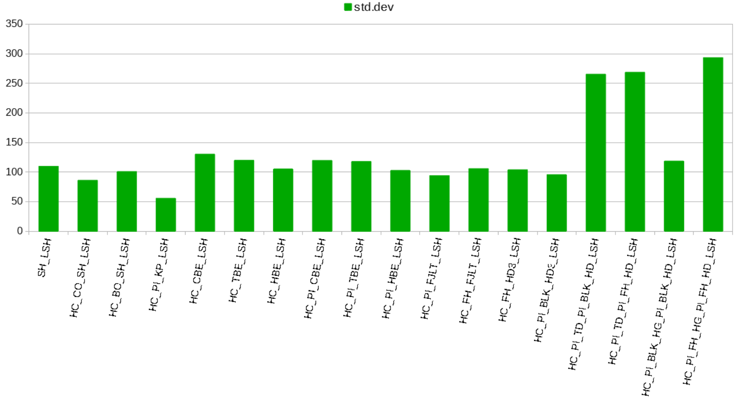
BIN SIZES BALANCE

Protocol

1. Compute bin sizes: $Counts(\mathcal{H}_s)$
2. Compute summary statistics (*min*, *max*, *mean*, <u>*sd*</u>, *med*) of $Counts(\mathcal{H}_s)$

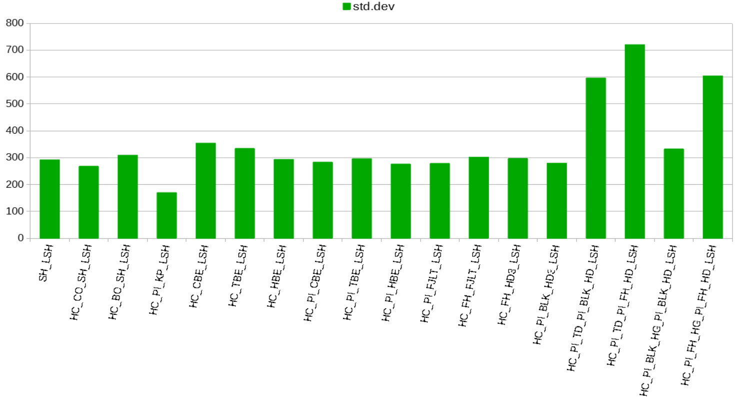
BIN SIZES BALANCE

MC5 - 5 species, 3 genuses, 25K reads, $b = 7$



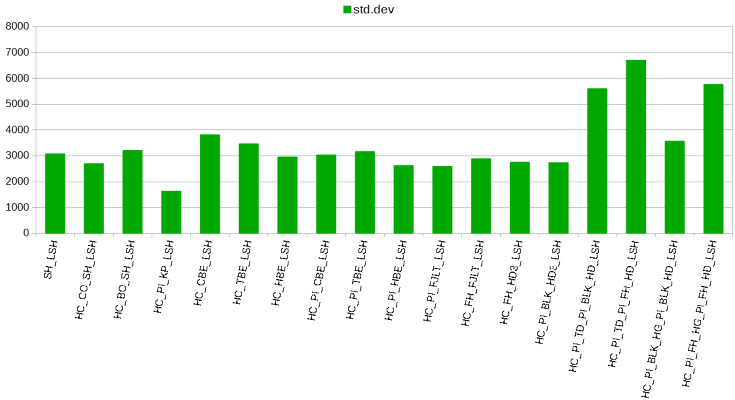
BIN SIZES BALANCE

MC10 - 10 species, 10 genuses, 50K reads, $b = 7$



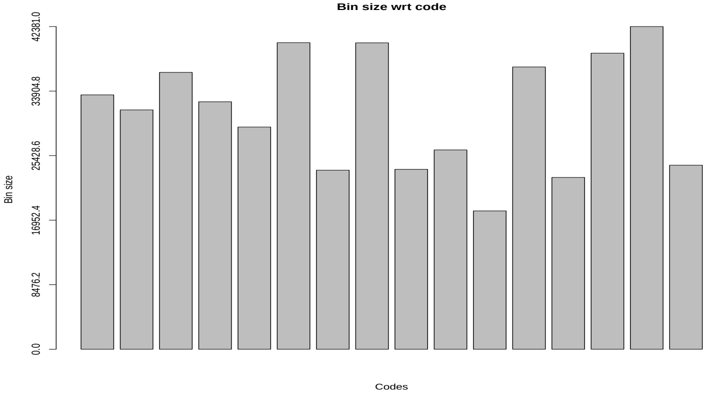
BIN SIZES BALANCE

MC100 - 100 species, $\leq$ 67 genuses, 500K reads, $b = 7$



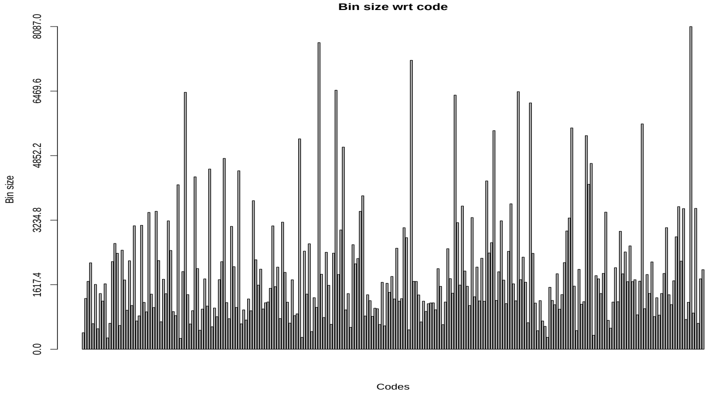
BIN SIZES BALANCE

MC100 - 100 species, $\leq$ 67 genuses, 500K reads, $b = 4$, SimHash



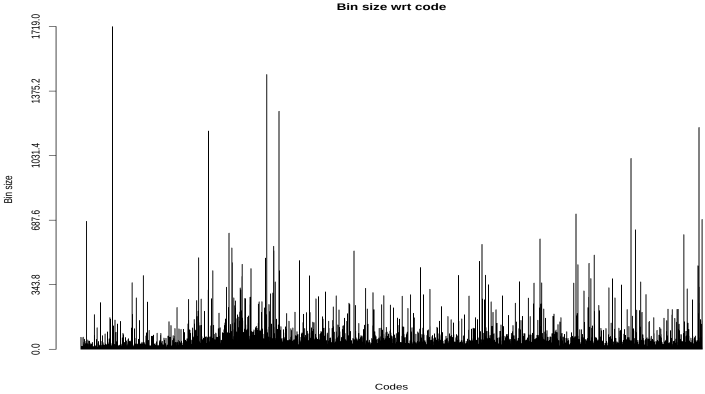
BIN SIZES BALANCE

MC100 - 100 species, $\leq$ 67 genuses, 500K reads, $b = 8$, SimHash



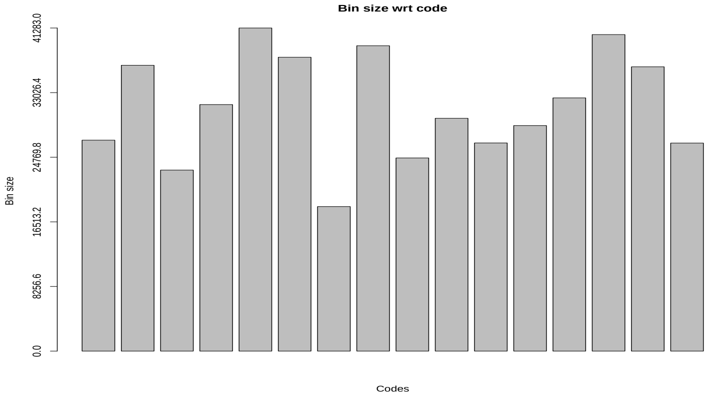
BIN SIZES BALANCE

MC100 - 100 species, $\leq$ 67 genuses, 500K reads, $b = 16$, SimHash



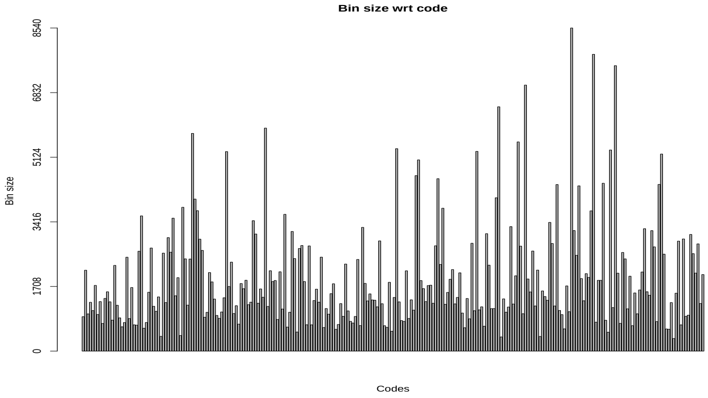
BIN SIZES BALANCE

MC100 - 100 species, $\leq$ 67 genuses, 500K reads, $b = 4$, SimHash - complete orthonormal



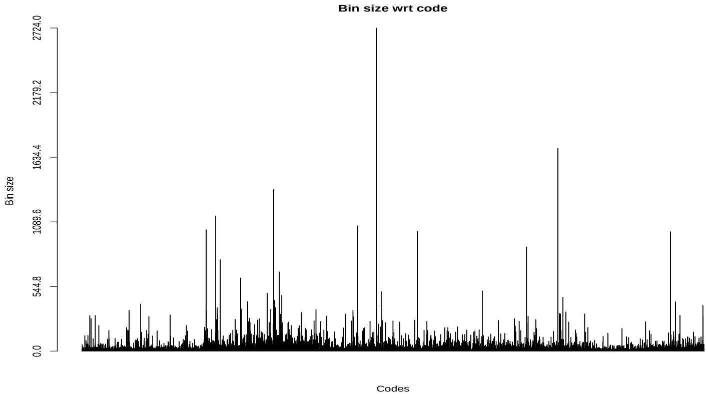
BIN SIZES BALANCE

MC100 - 100 species, $\leq$ 67 genuses, 500K reads, $b = 8$, SimHash - complete orthonormal



BIN SIZES BALANCE

MC100 - 100 species, $\leq$ 67 genuses, 500K reads, $b = 16$, SimHash - complete orthonormal



BINNING PERFORMANCE

Protocol

1. Compute binning scores (*Precision*, *Homogeneity*) of $\mathcal{H}_s$ at species and genus level

$$Precision(\mathcal{H}_s, \mathcal{C}) = \left(\sum_k \max_c T_{ck} \right) / n$$

T: confusion matrix hashcodes in $\mathcal{H}_s \times$ classes in $\mathcal{C}$

T_{ck}: number of objects from class c (species) hashcoded by k

BINNING PERFORMANCE

Protocol

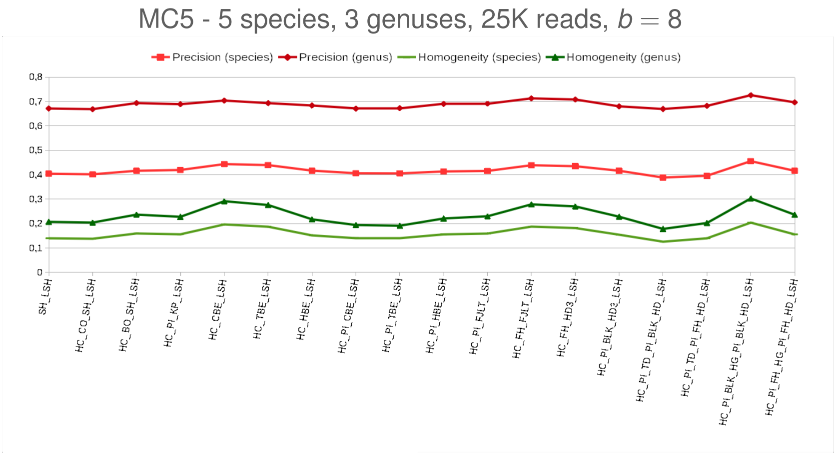
1. Compute binning scores (*Precision*, *Homogeneity*) of $\mathcal{H}_s$ at species and genus level

$$Homogeneity(\mathcal{H}_s, \mathcal{C}) = 1 - \frac{\sum_k \sum_c \frac{T_{ck}}{n} \log \frac{T_{ck}}{\sum_c T_{ck}}}{\sum_c \sum_k \frac{T_{ck}}{n} \log \frac{\sum_k T_{ck}}{n}}$$

T: confusion matrix hashcodes in $\mathcal{H}_s \times$ classes in $\mathcal{C}$

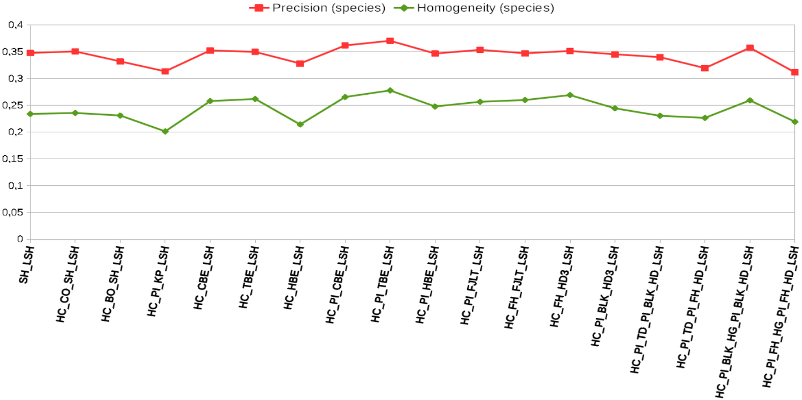
T_{ck}: number of objects from class c (species) hashcoded by k

BINNING PERFORMANCE

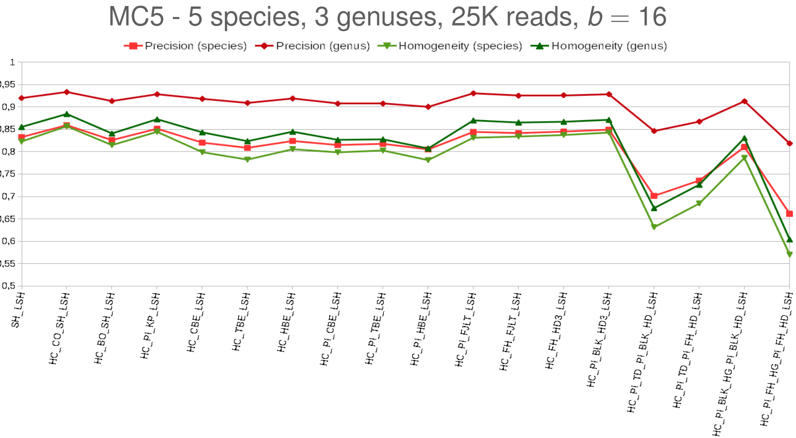


BINNING PERFORMANCE

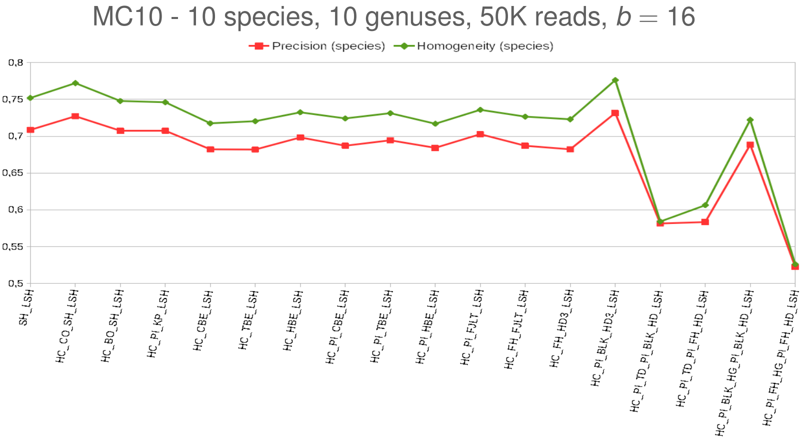
MC10 - 10 species, 10 genuses, 50K reads, $b = 8$



BINNING PERFORMANCE



BINNING PERFORMANCE



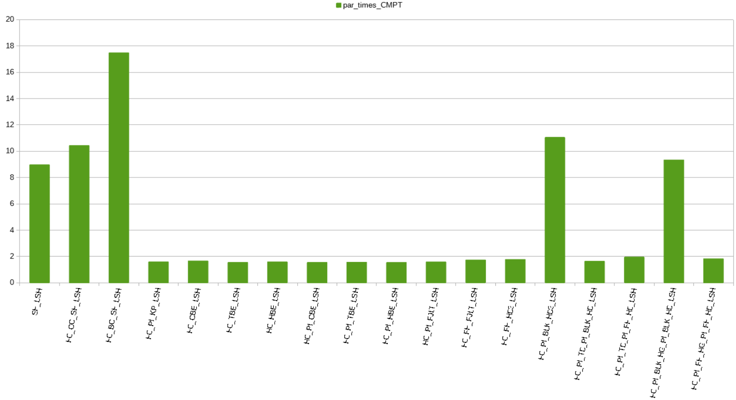
TIME PERFORMANCE

Protocol

1. Compute binning time in a parallel setting
2. Provide an estimate for sequential time

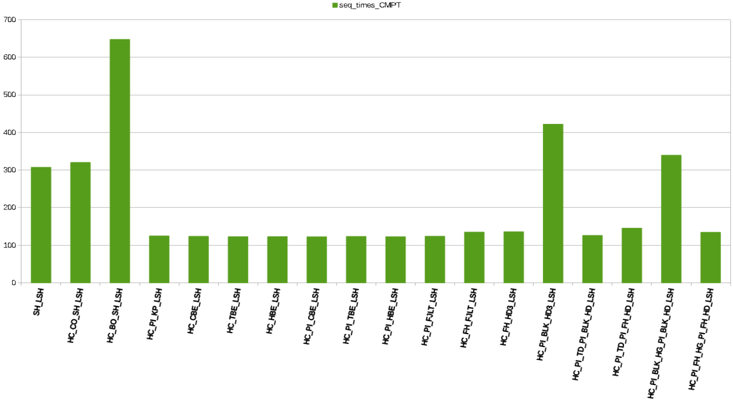
TIME PERFORMANCE

MC100 - 100 species, $\leq$ 67 genuses, 500K reads, $b = 16$, parallel time (seconds)



TIME PERFORMANCE

MC100 - 100 species, $\leq$ 67 genuses, 500K reads, $b = 16$, sequential time (seconds)



CONCLUSION AND FUTURE WORK

- Exploit LSH indexes for reads clustering
- Implement and test multi-index search algorithms
- Introduce learning, explore semi-supervised extensions
- Visualizing LSH indexes

CONCLUSION AND FUTURE WORK

- Exploit LSH indexes for reads clustering
- Implement and test multi-index search algorithms
- Introduce learning, explore semi-supervised extensions
- Visualizing LSH indexes

SPHERICAL LSH[16]

Idea

- Partition the surface of the *b*-dimensional hypersphere ($\mathbb{S}^{b-1}$)
- Leverage regular structures in *b*-dimensional space $\Rightarrow$ regular polytopes (simplex, orthoplex and hypercube only for $b \geq 5$)
- Hashing is performed by identifying the closest vertex of the chosen regular polytope $\forall x \in \mathcal{X}$

[16] Kengo Terasawa and Yuzuru Tanaka. “Spherical LSH for Approximate Nearest Neighbor Search on Unit Hypersphere.” In: *WADS*. ed. by Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Norbert Zeh. Vol. 4619. Lecture Notes in Computer Science. Springer, Aug. 24, 2007, pp. 27–38. ISBN: 978-3-540-73948-7. URL: `http://dblp.uni-trier.de/db/conf/wads/wads2007.html#TerasawaT07`.

SPHERICAL LSH[17]

Idea

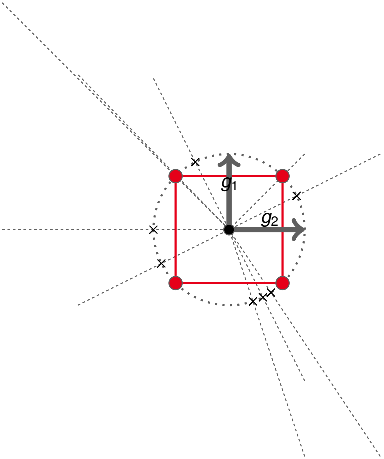
- Remains a data-independent approach that involve randomization in terms of a random rotation applied to the regular polytope
- Then hashing is obtained by assigning the closest vertex id among the vertices of the regular polytope

Let $\{v_1, v_2, \cdots, v_N\}$ with $||v_i||_2^2 = 1, \forall v_i$, the set of vertices that forms a regular polytope in $\mathbb{R}^b$, and $G \in \mathbb{R}^{b \times b}$ a rotation matrix ($G^\top G = I_b$)

$$h(x) = \operatorname*{argmin}_i ||Gv_i - x||_2^2$$

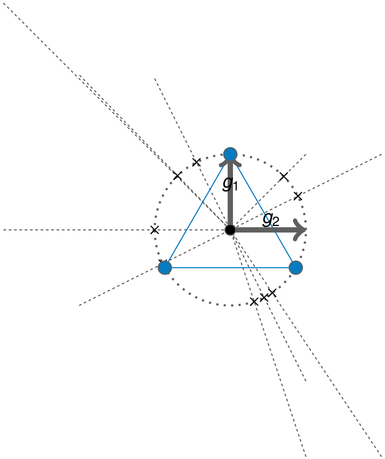
[17] Kengo Terasawa and Yuzuru Tanaka. "Spherical LSH for Approximate Nearest Neighbor Search on Unit Hypersphere." In: *WADS*. ed. by Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Norbert Zeh. Vol. 4619. Lecture Notes in Computer Science. Springer, Aug. 24, 2007, pp. 27–38. ISBN: 978-3-540-73948-7. URL: http://dblp.uni-trier.de/db/conf/wads/wads2007.html#TerasawaT07.

SPHERICAL LSH[18]



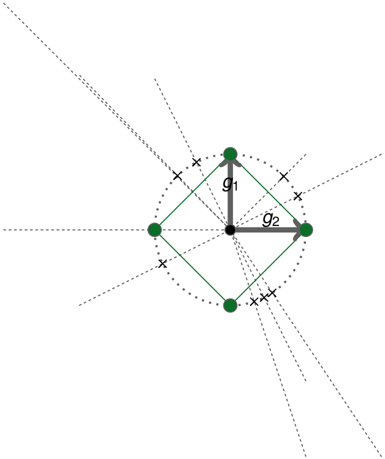
[18] Kengo Terasawa and Yuzuru Tanaka. "Spherical LSH for Approximate Nearest Neighbor Search on Unit Hypersphere." In: *WADS*. ed. by Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Norbert Zeh. Vol. 4619. Lecture Notes in Computer Science. Springer, Aug. 24, 2007, pp. 27–38. ISBN: 978-3-540-73948-7. URL: http://dblp.uni-trier.de/db/conf/wads/wads2007.html#TerasawaT07.

SPHERICAL LSH[18]



[18] Kengo Terasawa and Yuzuru Tanaka. “Spherical LSH for Approximate Nearest Neighbor Search on Unit Hypersphere.” In: *WADS*. ed. by Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Norbert Zeh. Vol. 4619. Lecture Notes in Computer Science. Springer, Aug. 24, 2007, pp. 27–38. ISBN: 978-3-540-73948-7. URL: http://dblp.uni-trier.de/db/conf/wads/wads2007.html#TerasawaT07.

SPHERICAL LSH[18]



[18] Kengo Terasawa and Yuzuru Tanaka. “Spherical LSH for Approximate Nearest Neighbor Search on Unit Hypersphere.” In: *WADS*. ed. by Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Norbert Zeh. Vol. 4619. Lecture Notes in Computer Science. Springer, Aug. 24, 2007, pp. 27–38. ISBN: 978-3-540-73948-7. URL: `http://dblp.uni-trier.de/db/conf/wads/wads2007.html#TerasawaT07`.

ITERATIVE QUANTIZATION[19]

Idea

- Follow the idea of spherical LSH but introduce (unsupervised) learning yielding a data-dependent approach
- **Learn** the rotation (of an hypercube) that best fit the data lying on the hypersphere instead of randomly rotating it

[19] Y. Gong et al. "Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.12 (2013), pp. 2916–2929. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.193.

ITERATIVE QUANTIZATION[20]

$$Y = GX \ ; \ X \in \mathbb{R}^{p \times n} \ ; \ G \in \mathbb{R}^{b \times p}$$

$$G = RG' \ ; \ R \in \mathbb{R}^{b \times b}, R^\top R = I_b \ ; \ G' = \underset{\{P \in \mathbb{R}^{b \times p}, P^\top P = I_b\}}{\text{argmax}} ||PX||^2 \ (PCA)$$

We aim to solve :

$$R = \underset{R'}{\text{argmin}} || \,\mathbf{sign}(Y) - R'G'X||_F^2 \ ; \ R'^\top R' = I_b$$

sign applies sign component-wise on each matrix element

$$\text{sign}(Y_{ij}) = \begin{cases} 1 & \text{if } Y_{ij} \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

[20] Y. Gong et al. "Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.12 (2013), pp. 2916–2929. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.193.

ITERATIVE QUANTIZATION[21]

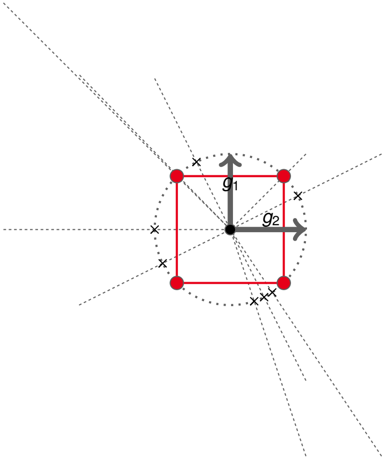
$$R = \underset{R',Y}{\operatorname{argmin}} ||\,\mathbf{sign}(Y) - R'G'X||_F^2\ ;\ R'^\top R' = I_b$$

Let R be a random rotation matrix

1. **Fix** R, compute $\mathbf{sign}(Y) = \mathbf{sign}(R'G'X)$ (recall that $G'X$ is the projected X on the subspace spanned by the top b leading eigenvectors)
2. **Fix** Y, compute $R = \underset{R'}{\operatorname{argmin}} ||\,\mathbf{sign}(Y) - R'G'X||_F^2\ ;\ R'^\top R' = I_b$
   $\Rightarrow$ Let $Y' = \mathbf{sign}(Y)$, the solution involve computing the SVD yielding $R = V^\top U$ where U and $V^\top$ are left (resp. right) singular vectors of the matrix $Y'X^\top G'^\top$.

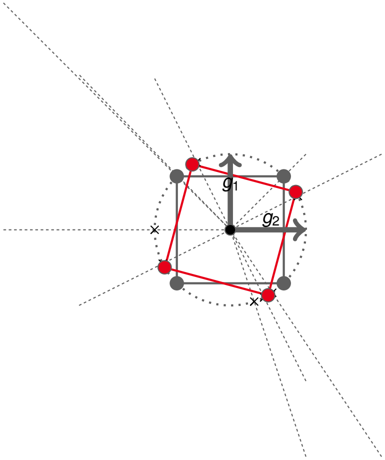
[21] Y. Gong et al. "Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.12 (2013), pp. 2916–2929. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.193.

ITERATIVE QUANTIZATION[22]



[22] Y. Gong et al. "Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.12 (2013), pp. 2916–2929. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.193.

ITERATIVE QUANTIZATION[22]



[22] Y. Gong et al. “Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.12 (2013), pp. 2916–2929. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.193.

DISCUSSION

- Focus on high dimensional data
- Multiple hash tables[23] *vs.* multi-probing
- Data-independent LSH *vs.* Data-dependent LSH[24,25]
- Fully randomized *vs.* learned hash functions
- Short *vs.* long codes
- Preconditionner (*HD*) + Structured (fast) subspace projection + Learned rotation become a standard for data-dependent hashing

[23] M. Norouzi, A. Punjani, and D. J. Fleet. "Fast Exact Search in Hamming Space With Multi-Index Hashing". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.6 (2014), pp. 1107–1119. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2013.231.

[24] J. Wang et al. "Learning to Hash for Indexing Big Data – A Survey". In: *Proceedings of the IEEE* 104.1 (2016), pp. 34–57. ISSN: 0018-9219. DOI: 10.1109/JPROC.2015.2487976.

[25] J. Wang et al. "A Survey on Learning to Hash". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP.99 (2018), pp. 1–1. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2699960.

jacques-henri.sublemontier@cea.fr

Commissariat à l'énergie atomique et aux énergies alternatives
Institut List | CEA SACLAY NANO-INNOV | BAT. 861 – PC142
91191 Gif-sur-Yvette Cedex - FRANCE
www-list.cea.fr