

BinDash 2.0: New MinHash Scheme Allows Ultra-fast and Accurate Genome Search and Comparisons

3

4 Jianshu Zhao^{1,2,6}, Xiaofei Zhao^{3,6}, Jean Pierre-Both⁴, Konstantinos T. Konstantinidis^{1,2,5,*}

5

6

7 ¹Center for Bioinformatics and Computational Genomics, Georgia Institute of Technology,
8 Atlanta, Georgia, USA

9 ²School of Biological Sciences, Georgia Institute of Technology, Atlanta, Georgia, USA

10 ³MOE Key Laboratory of Bioinformatics and Bioinformatics Division, Department of
11 Automation, Tsinghua University, Beijing, China

12 ⁴Université Paris-Saclay, CEA, List, Palaiseau, France

13 ⁵School of Civil and Environmental Engineering, Georgia Institute of Technology, Atlanta,
14 Georgia, USA

15 ⁶Those authors contribute equally

16 *Corresponding author Konstantinos T Konstantinidis

17 (kostas.konstantinidis@gatech.edu)

18

19 **Running title:** Ultra-fast and Accurate Genome Comparisons via new MinHash
20 algorithms

21

22

23

24 **Abstract**

25 **Motivation:** Comparing large number of genomes in term of their genomic distance is
26 becoming more and more challenging because there is an increasing number of microbial
27 genomes deposited in public databases. Nowadays, we may need to estimate pairwise
28 distances between millions or even billions of genomes. Few softwares can perform such
29 comparisons efficiently.

30 **Results:** Here we update the multi-threaded software BinDash by implementing several
31 new MinHash algorithms and computational optimization (e.g. Simple Instruction Multiple
32 Data, SIMD) for ultra-fast and accurate genome search and comparisons at trillion scale.

33 That is, we implemented b-bit one-permutation rolling MinHash with optimal/faster
34 densification with SIMD. Now with BinDash 2, we can perform 0.1 trillion (or $\sim 10^{11}$) pairs
35 of genome comparisons in about 1.8 hours on a descent computer cluster or several
36 hours on personal laptops, a $\sim 50\%$ or more improvement over original version. The ANI
37 (average nucleotide identity) estimated by BinDash is well correlated with other accurate
38 but much slower ANI estimators such as FastANI or alignment-based ANI. In line with the
39 findings from comparing 90K genomes ($\sim 10^9$ comparisons) via FastANI, the 85% \sim 95%
40 ANI gap is consistent in our study of $\sim 10^{11}$ prokaryotic genome comparisons via
41 BinDash2, which indicates fundamental ecological and evolutionary forces keeping
42 species-like unit (e.g., > 95% ANI) together.

43 **Availability and implementation:** BinDash is released under the Apache 2.0 license at:
44 <https://github.com/zhaoxiaofei/bindash>

45 **Contact:** kostas.konstantinidis@gatech.edu

46 **Supplementary information:** Supplementary data are available at Bioinformatics
47 online.

48 **Keywords:** MinHash, One Permutation Hashing, densification, SIMD, ANI

49

50 **Introduction**

51 MinHash, originally developed for detecting duplicate webpages (Broder, et al., 1998),
52 turned out to be a powerful strategy when applied for genome comparisons in the
53 pioneering work from Ondov, et al. (2016): genome kmer set Jaccard index estimated via
54 MinHash can be accurate estimation of genomic distance or ANI via Mash equation
55
$$\text{ANI or } (1 - \text{Mash}_D) = 1 + \frac{1}{k} \log\left(\frac{2^*J}{1+J}\right)$$
. In the original MinHash work, many hash functions
56 are required under the locality sensitivity scheme. An alternative to k-minwise when
57 estimating set similarity with Minwise sketches is bottom-k implementation, where we use
58 one hash function and maintain the sketch as the keys with the k smallest hash values
59 (Broder, 1997). However, it needs a priority queue to maintain the k smallest hash values,
60 and this leads to a non-constant worst-case time per element (overall complexity is
61 $O(n * \log(k))$), which may be a problem in real-time processing of high-volume data
62 streams. More importantly, since only one hash function was used, we are not able to
63 encode set-similarity as an inner product of two sketch vectors. This is because the
64 elements lose their “alignment” – that is, the key with the smallest hash value in one set
65 might have the 10th smallest hash value in another set (Dahlgaard, et al., 2015). Or in
66 other words, the alignment property is preserved if the same components (e.g., value at

67 a given position) of two different sketches are equal with a collision probability that is a
68 monotonic function of some similarity measure, also called a locality sensitive hashing
69 (Shrivastava, 2017). In many real-world applications, such as nearest neighbor search,
70 this property will guarantee theoretically optimal accuracy and search recall will
71 deteriorate significantly if not preserving LSH (Shrivastava, 2017). Another interesting
72 alternative is called One Permutation Hashing, which also applies only one hash function
73 with time complexity $O(n + s)$). However, it has much larger variance than traditional k-
74 minwise hashing because there might be empty slots in the sketch vector after splitting
75 the sketch into buckets. Densification, that is to fill in the empty slots with some non-empty
76 slots, chosen with certain rules, has greatly improved the accuracy of one permutation
77 hashing and proved to be theoretical equivalent to that of traditional k-minwise hashing.
78 Based on one permutation hashing, original densification (Shrivastava and Li, 2014),
79 improved densification (Shrivastava and Li, 2014) and optimal densification (Shrivastava,
80 2017) are proposed and have all been proven to be locality sensitive theoretically. We
81 implemented so called original densification and optimal densification in the first version
82 of BinDash (Zhao, 2019). However, more densification strategies have been proposed
83 since the publication of original BinDash, e.g., faster densification (Mai, et al., 2020), re-
84 randomized densification (Li, et al., 2019), bidirectional densification (Jia, et al., 2021)
85 with even better run time behavior. Specifically, faster densification improved the worse-
86 case densification computational complexity in optimal densification from $O(n + s^2)$ to
87 $O(n + s \log(s))$ with the same average-case $O(n + s)$ (Figure 1b and c) while re-
88 randomized densification further improves accuracy for optimal densification at the cost

89 of additional computation since rerun MinHash within previously empty bins after optimal
90 densification is computationally expensive when there are many empty bins, see detailed
91 complexity analysis for re-randomization densification (Li, et al., 2019). BinDash 2
92 implemented all flavors and variants of MinHash presented in Broder (1997), Li and König
93 (2010), Li, et al. (2012), Shrivastava and Li (2014), Shrivastava and Li (2014), Shrivastava
94 (2017) (Figure 1a and b) and Mai, et al. (2020) (Figure 1a and c) with SIMD (see below).
95 The implementation detail is presented in Supplementary Material.
96 The limiting step of BinDash is popcount, which counts the number of non-zero elements
97 in large bit vectors for estimation of collision probability (after XOR operation). However,
98 recent algorithmic advancements in Simple Instruction Multiple Data has provided the
99 opportunity to further speed up popcount for many instruction sets (e.g., AVX2, AVX512
100 and SVE512) (Langarita, et al., 2023; Muła, et al., 2018). Since we use 64-bit integer type
101 as sketch vector to store hashes from kmers, it possible to use SIMD to count the number
102 of non-zero elements in parallel after rearranging the sketch vector in a way such that
103 each small portion of sketch vector fill into AVX instructions (e.g., 512 bit for AVX512 or
104 SVE512, 256 bit for AVX2) and also take care of the remaining part that does not fit for
105 any size of sketch vectors.
106 Genomic distance, measured via Jaccard index can be transformed into genome average
107 nucleotide identity via the Mash equation mentioned above. However, it has been recently
108 proved that the Poisson model assumption of sequence evolution can also be replaced
109 by Binomial model, which give more accurate estimation of ANI for distantly related
110 genomes (e.g., below 85% ANI) (Belbasi, et al., 2022): $ANI \text{ or } (1 - Mash_D) = \left(\frac{2^*J}{1+J}\right)^{\frac{1}{k}}$. We also

111 add this Binomial model option in BinDash2. To further improve the accuracy of BinDash
112 2 estimated ANI when compared to alignment-based ANI, we implemented a supervised
113 leaning step, which minimize the RMSE between BinDash 2 estimated ANI according to

114 the RMSE equation: $RMSE = \sqrt{\frac{\sum_{t=1}^T (\bar{y}_t - y_t)^2}{T}}$ where \bar{y}_t and y_t are BinDash 2 estimated
115 ANI and orthoANI (usearch) ANI for each pair, respectively. T is the total number of pairs
116 of genomes in the training dataset. We use a large collection of training genomes (10,000)
117 extracted from NCBI/RefSeq, covering ANI from 75% to 100%.

118 Similar to the first version, for each set, BinDash2 first applies one-permutation MinHash
119 (Li, et al., 2012), which use one predefined hash function to all elements/kmers in the set.
120 Then, one-permutation MinHash deterministically partitions the hash value universe into
121 a predefined maximum number B of buckets, extracts the smallest hash value in each
122 bucket, and extracts the b lowest bits (b=14 in practise) of each smallest hash value (Li
123 and König, 2010). These B*b bits are use as the signature of the set. Usually, a hash
124 value v is assigned to the $\lceil v/(M/B) \rceil$ bucket, where M is the maximum possible value for
125 v. One-permutation MinHash may produce a bucket that contains hash values for one set
126 but no hash values for another set, we then apply densification algorithms (Shrivastava
127 and Li, 2014). In additional to all MinHash schemes implemented in original version
128 (original MinHash, bottom-k MinHash and optimal densification), we then implemented
129 faster densification in BinDash2. That is, we now implemented b-bit one-permutation
130 rolling MinHash with optimal/faster densification with SIMD. Specifically, cyclic-
131 polynomial rolling hash based on iterated string hashing is much faster than
132 MurMurHash3 for DNA strings as in Mash (Lemire, 2012; Lemire and Kaser, 2014). We

133 also added nearest neighbor search option to only report nearest genomic distance to
134 query genomes with the consideration that the three densification schemes are all LSH,
135 a property important for genome search and classifications, questionable for other similar
136 softwares without LSH properties such as Mash, Dashing and Sourmash (Baker and
137 Langmead, 2019; Brown and Irber, 2016; Ondov, et al., 2016).

138

139 **Evaluation**

140 We compared BinDash2 with BinDash, Mash, Dashing 1 and 2, the state-of-the-art
141 MinHash-based bioinformatics softwares. On Dec. 22, 2023, the 315,686 assemblies of
142 bacterial and archaea genomes in RefSeq were downloaded. The downloaded genomes
143 consist of 412.7 GB of gzip-compressed FASTA files (~2.1 terabytes for raw fasta files).
144 The 315,686 compressed genomes are used as input data to each software. For each
145 software, we recorded the following: total size of the files used to represent sketches and
146 wall-clock runtime of each command. Each software ran, with its default parameter values
147 (other than a cutoff of 0.2 for mutation rate) with 24 threads, on a Red Hat Enterprise
148 Linux (Intel(R) Xeon(R) Gold 6226 CPU @ 2.70GHz, supports AVX2 and AVX512
149 instruction sets). BinDash2, dashing 1 and 2 and Mash are both composed of two
150 commands: sketch and dist. The command sketch compute sketches based on hashing
151 kmers from genomes. The command dist compares each sketch used as query to each
152 sketch used as target. The total runtime of these two commands is the total runtime of
153 the corresponding software. In all comparisons, we use sketch size 10,000 instead of the
154 default 2000 to have accuracy at 99% ANI or above, sketch size is important for real-

155 world genomic distance comparisons for species level ANI comparisons (e.g., 95% or
156 99%) as shown in many places (Jain, et al., 2018). For 318,756 bacterial and archaea
157 genomes, BinDash 2 is 47.8% faster than original BinDash, ~80 time faster than Mash,
158 ~8 and ~17 times faster than Dashing 1 (HLL) and 2 (ProbMinHash or SetSketch)
159 respectively, for the dist command (Table S1) while for sketch, BinDash 2 is sighly slower
160 than Dashing 1 (HLL, the fastest). However, since sketch step is always less than 20
161 minutes for ~318K genomes, it is not a limiting step for all tools mentioned above (Table
162 S2). In terms of sketch size stored on disk, BinDash 2 and BinDash is about 3 times
163 smaller than Mash but larger than Dashing (Table S1).

164 We did a theoretical analysis of RMSE for all MinHash-like and HyperLogLog-like
165 algorithms and showed that MinHash-like algorithms are generally more accurate than
166 HyperLogLog-like algorithms in terms of estimating Jaccard index (Table S2). We
167 computed the true pairwise Jaccard indices of the 120 reference genomes chosen among
168 these 318,756 genomes with ANI above 80%. The true Jaccard indices serve as ground
169 truth and root-mean-square error (RMSE) was used to measure the accuracy of all tools.
170 BinDash 2 remains almost the same accuracy with original BinDash, with RMSE better
171 than both Mash and Dashing (Table S3). More importantly, BinDash 2 and Mash RMSE
172 converges to 0 as sketch size increases in theory while for Dashing 1 and 2
173 (HyperLogLog), there is no such guarantee (Table S2) (Flajolet, et al., 2007; Gakhov,
174 2022).

175 To show the real-world application of BinDash 2, we first compare the ANI from BinDash
176 2 with orthoANI after correction: A correcting factor σ is obtained via the supervised

177 learning step to correct the final ANI or 1-Mash_D so that the final output ANI value
178 correlates well with orthoANI (usearch) ANI (Figure 1d). Then we applied it to defining
179 bacterial genome species boundaries by performing all versus all comparisons among
180 ~318K genomes. We see a clear 85% to 95% ANI gap (Figure 1e), consistent with more
181 accurate but much slower software called fastANI. We believe this consistent gap is not
182 sampling bias or cultivation bias because many genomes deposited recently are
183 environmental genomes obtained by metagenomics and they reject cultivation and
184 isolation.

185

186 **Discussion**

187 BinDash 2 implemented the fastest densification idea called faster densification, which
188 has the same theoretical RMSE with traditional MinHash. Overall, MinHash-like
189 algorithms for estimation of Jaccard index are more accurate than HyperLogLog-like
190 algorithms as implemented in Dashing 1 and Dashing 2. To further improve the accuracy
191 of MinHash estimated Jaccard index or ANI, it is possible to explore new MinHash
192 algorithms like re-randomized MinHash (Li, et al., 2019) and circulant MinHash (Li and
193 Li, 2022), which are all theoretical breakthroughs very recently. However, both algorithms
194 achieved smaller RMSE (using same sketch size) at the expense of additional
195 computations. For example, re-randomized MinHash requires additional MinHash step
196 within each empty bin after running optimal densification, which can be several times
197 slower than faster densification according to theoretical analysis (Li, et al., 2019).
198 Circulant MinHash requires large number of random permutations for the second

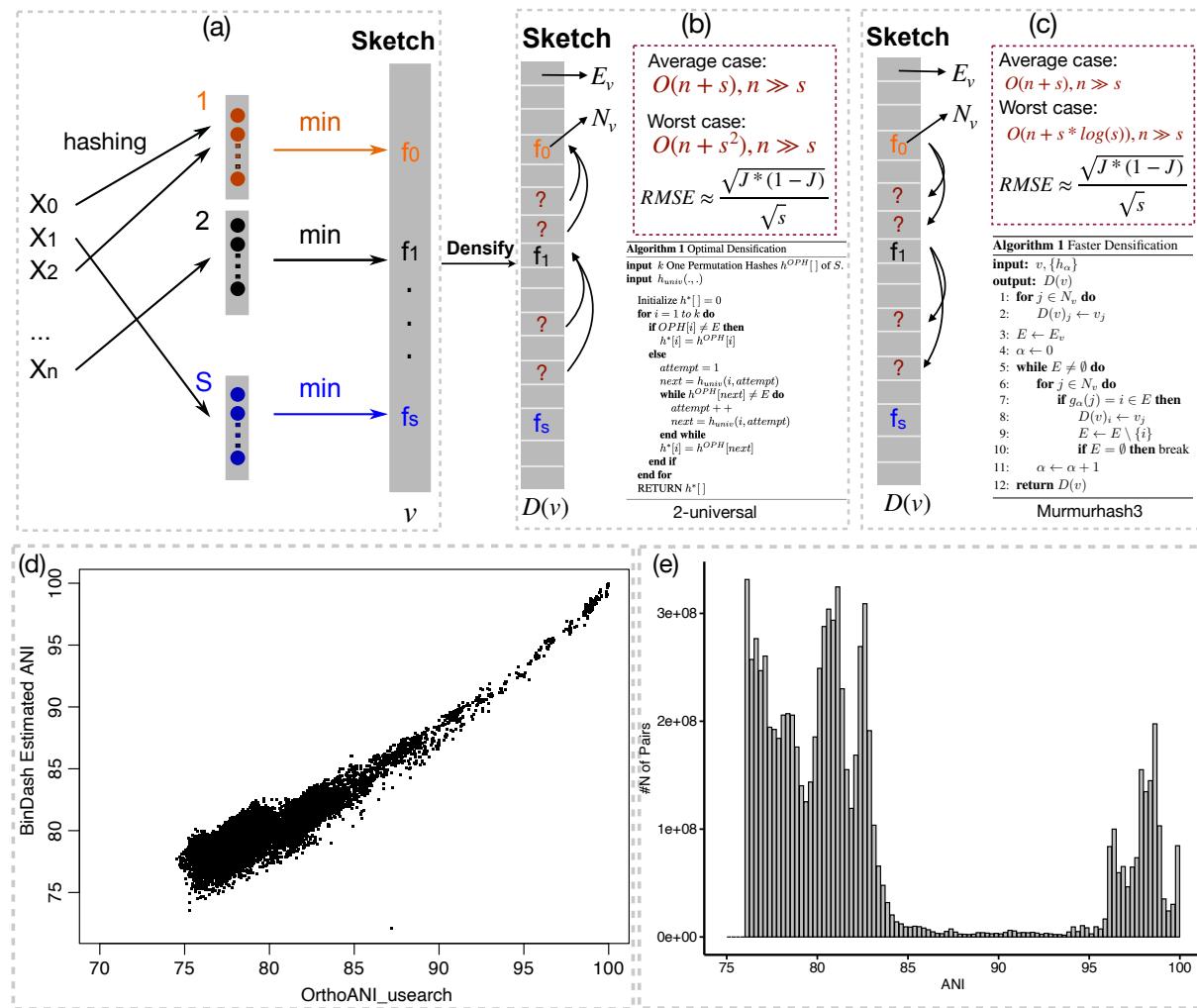
199 permutation procedure, which is not so efficient for large number of bins in sketch vector
200 in practice. However, we can use slightly larger sketch size in fast densification to achieve
201 similar accuracy with re-randomized MinHash or Circulant MinHash because the running
202 time of faster densification is not compromised due to the fact that average-case $O(n + s)$
203 is not affected by sketch size s (n is always more than 100 times larger than s for genomic
204 applications). In this regard, BinDash 2 achieved the best running time and accuracy
205 trade-off among all MinHas-like algorithms, including newly invented ones.
206 We have also showed that BinDash 2 can be used to perform large scale genome
207 comparisons and help define prokaryotic genome species. Taken together, via
208 implementing new algorithms and computational optimization, we believe BinDash 2 will
209 be a practical alternative to similar tools and will help very large-scale microbial genome
210 search and comparisons while maintaining accuracy for biological knowledge discovery.

211

212 **Conflict of Interest:** none declared.

213

214



215
216
217
218
219
220
221
222
223
224

Figure 1. (a) One Permutation Hashing; **(b)** Optimal Densification. Mapping empty bins to non-empty bins and copy values from non-empty bins to empty bins, carefully designed 2-universal hashing is required for mapping, implemented in BinDash 1; **(c)** Faster Densification or Reverse Optimal Densification. Mapping non-empty bins to empty bins and copy values to empty bins from non-empty bins, random hashing library can be used, e.g., murmurhash3, implemented in BinDash 2; **(d)** BinDash estimated ANI vs. OrthoANI for randomly selected 3,000 prokaryotic genomes after supervised learning; **(e)** Distribution of ANI pairs with >75% ANI for all vs. all comparisons of ~318K NCBI/RefSeq prokaryotic genomes. The histogram follows a bimodal distribution with less than 0.01% of pairs fall between ANI 85% and 95%.

225

References

226 Baker, D.N. and Langmead, B. Dashing: fast and accurate genomic distances with HyperLogLog. *Genome Biology*
227 2019;20(1):265.

228 Belbasi, M., et al. The minimizer Jaccard estimator is biased and inconsistent. *Bioinformatics*
229 2022;38(Supplement_1):i169-i176.

230 Broder, A.Z. On the resemblance and containment of documents. In, *Proceedings. Compression and Complexity of*
231 *SEQUENCES 1997 (Cat. No. 97TB100171)*. IEEE; 1997. p. 21-29.

232 Broder, A.Z., et al. Min-wise independent permutations. In, *Proceedings of the thirtieth annual ACM symposium on*
233 *Theory of computing*. 1998. p. 327-336.

234 Brown, C.T. and Irber, L. sourmash: a library for MinHash sketching of DNA. *Journal of open source software*
235 2016;1(5):27.

236 Dahlgaard, S., et al. Hashing for statistics over k-partitions. In, *2015 IEEE 56th Annual Symposium on Foundations of*
237 *Computer Science*. IEEE; 2015. p. 1292-1310.

238 Flajolet, P., et al. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete Mathematics*
239 and *Theoretical Computer Science* 2007:137-156.

240 Gakhov, A. Probabilistic data structures and algorithms for big data applications. BoD—Books on Demand; 2022.

241 Jain, C., et al. High throughput ANI analysis of 90K prokaryotic genomes reveals clear species boundaries. *Nature*
242 *Communications* 2018;9(1):5114.

243 Jia, P., et al. Bidirectionally densifying LSH sketches with empty bins. In, *Proceedings of the 2021 International*
244 *Conference on Management of Data*. 2021. p. 830-842.

245 Langarita, R., et al. Porting and Optimizing BWA-MEM2 Using the Fujitsu A64FX Processor. *IEEE/ACM Transactions*
246 *on Computational Biology and Bioinformatics* 2023.

247 Lemire, D. The universality of iterated hashing over variable-length strings. *Discrete Applied Mathematics*
248 2012;160(4-5):604-617.

249 Lemire, D. and Kaser, O. Strongly universal string hashing is fast. *The Computer Journal* 2014;57(11):1624-1638.

250 Li, P. and König, C. b-Bit minwise hashing. In, *Proceedings of the 19th international conference on World wide web*.
251 2010. p. 671-680.

252 Li, P., Li, X. and Zhang, C.-H. Re-randomized densification for one permutation hashing and bin-wise consistent
253 weighted sampling. *Advances in Neural Information Processing Systems* 2019;32.

254 Li, P., Owen, A. and Zhang, C.-H. One permutation hashing. *Advances in Neural Information Processing Systems*
255 2012;25.

256 Li, X. and Li, P. C-MinHash: Improving Minwise Hashing with Circulant Permutation. In, *International Conference on*
257 *Machine Learning*. PMLR; 2022. p. 12857-12887.

258 Mai, T., et al. On densification for minwise hashing. In, *Uncertainty in Artificial Intelligence*. PMLR; 2020. p. 831-840.

259 Muła, W., Kurz, N. and Lemire, D. Faster population counts using AVX2 instructions. *The Computer Journal*
260 2018;61(1):111-120.

261 Ondov, B.D., et al. Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biology*
262 2016;17(1):132.

263 Shrivastava, A. Optimal densification for fast and accurate minwise hashing. *International Conference on Machine*
264 *Learning* 2017:3154-3163.

265 Shrivastava, A. and Li, P. Densifying one permutation hashing via rotation for fast near neighbor search. In,
266 *International Conference on Machine Learning*. PMLR; 2014. p. 557-565.

267 Shrivastava, A. and Li, P. Improved densification of one permutation hashing. In, *Proceedings of the Thirtieth*
268 *Conference on Uncertainty in Artificial Intelligence*. 2014. p. 732-741.

269 Zhao, X. BinDash, software for fast genome distance estimation on a typical personal laptop. *Bioinformatics*
270 2019;35(4):671-673.

271