Efficient and Robust Search of Microbial Genomes via Phylogenetic Compression

Karel Břinda [1, 2*], Leandro Lima [3], Simone Pignotti [2,4], Natalia Quinones-Olvera [2], Kamil Salikhov [4], Rayan Chikhi [5], Gregory Kucherov [4], Zamin Iqbal [3], Michael Baym [2*]

1 GenScale, Inria/IRISA Rennes, Campus de Beaulieu, 35042 Rennes Cedex, France

2 Department of Biomedical Informatics, Harvard Medical School, MA 02115 Boston, USA

3 EMBL-EBI, CB10 1SD Hinxton, UK

4 LIGM, CNRS, Univ. Gustave Eiffel, 77454 Marne-la-Vallée Cedex 2, France

5 Department of Computational Biology, Institut Pasteur, 75015 Paris, France

* Correspondence to karel.brinda@inria.fr and baym@hms.harvard.edu

ABSTRACT

Comprehensive collections approaching millions of sequenced genomes have become central information sources in the life sciences. However, the rapid growth of these collections makes it effectively impossible to search these data using tools such as BLAST and its successors. Here, we present a technique called phylogenetic compression, which uses evolutionary history to guide compression and efficiently search large collections of microbial genomes using existing algorithms and data structures. We show that, when applied to modern diverse collections approaching millions of genomes, lossless phylogenetic compression improves the compression ratios of assemblies, de Bruijn graphs, and *k*-mer indexes by one to two orders of magnitude. Additionally, we develop a pipeline for a BLAST-like search over these phylogeny-compressed reference data, and demonstrate it can align genes, plasmids, or entire sequencing experiments against all sequenced bacteria until 2019 on ordinary desktop computers within a few hours. Phylogenetic compression has broad applications in computational biology and may provide a fundamental design principle for future genomics infrastructure.

INTRODUCTION

The comprehensive collections of genomes have become an invaluable resource for research across life sciences. However, their exponential growth, exceeding improvements in computation, makes their storage, distribution, and analysis increasingly difficult [1]. As a consequence, traditional search approaches, such as the Basic Local Alignment Search Tool (BLAST) [2] and its successors, are becoming less effective with the available reference data, which poses a major challenge for organizations such as the National Center for Biotechnology Information (NCBI) or European Bioinformatics Institute (EBI) in maintaining the searchability of their repositories.

The key to achieving search scalability are compressive approaches that aim to store and analyze genomes directly in the compressed domain [3,4]. Genomic data have low fractal dimension and entropy [5], which guarantees the existence of efficient search algorithms [5]. However, despite the progress in compression-related areas of computer science [4–14], it remains a practical challenge to compute parsimonious compressed representations of the exponentially growing public genome collections, particularly in light of their heavily biased sampling.

Microbial collections are particularly difficult to compress due to the huge number and the exceptional levels of genetic diversity, which reflect the billions of years of evolution across the domain. Even though substantial efforts have been made to construct comprehensive collections of all sequenced microbial genomes, such as the 661k assembly collection [15] (661k pre-2019 bacteria) and the BIGSIdata de Bruijn graph collection [16] (448k de Bruijn graphs of all pre-2016 bacterial and viral raw sequence), the resulting data archives and indexes range from hundreds of gigabytes (661k) to tens of terabytes (BIGSIdata). This scale exceeds the bandwidth, storage, and data processing capacities of most users, making local computation on these data functionally impossible.

We reasoned that the redundancies among microbial genomes are efficiently predictable, as they reflect the underlying evolutionary and sampling processes. While genomes in nature can accumulate substantial diversity through vertical and horizontal mutational processes, this process is functionally sparse, and at the same time subjected to selective pressures and drift that limit their overall entropy. This is further limited by selective biases due to culture and research or clinical interests, resulting in sequencing efforts being predominantly focused on narrow subparts of the tree of life, associated with model organisms and human pathogens [15]. Importantly, such subtrees have been shown to be efficiently compressible when considered in isolation, as low-diversity groups of oversampled phylogenetically related genomes, such as isolates of the same species under epidemiological surveillance [17,18]. This suggests that the compression of comprehensive collections could be informed by their evolutionary

history. This would reduce the complex problem of general genome compression to a much more tractable problem of local compression of phylogenetically ordered genomes, identified for instance through phylogenetic trees.

Phylogenetic trees are effective at estimating the similarity and compressibility of microbial genomes and their data representations. The closer two genomes are within a phylogeny, the closer they are likely to be in terms of mathematical similarity measures, such as the edit distance or *k*-mer distances [19], and thus also more compressible. Importantly, this principle holds not only for genome assemblies, but also for their similarity-preserving representations, such as de Bruijn graphs or *k*-mer indexes [20]. Phylogenetic trees could be embedded into computational schemes in order to assort similar data together, as a preprocessing step for boosting local compressibility of data. The well-known Burrows-Wheeler Transform [21] has a similar purpose in a different context. Other related ideas have previously been used for scaling up metagenomic classification using taxonomic trees [22–25].

At present, the public version of BLAST is frequently used to identify the species of a given sequence by comparing it to exemplars, but it is impossible to align against *all* sequenced bacteria. Despite the increasing number of bacterial assemblies available in the NCBI repositories, the searchable fraction of bacteria is exponentially decreasing over time (**Fig. 1a**). This limits the ability of the research community to study bacteria in the context of their known diversity, as the gene content of different strains can vary substantially, and important hits can be missed due to the database being unrepresentative.

Here, we present a solution to the problem of searching vast libraries of microbial genomes: *phylogenetic compression*, a technique for an evolutionary-guided compression of arbitrarily sized microbial genome collections. We show that the underlying evolutionary structure of microbes can be efficiently approximated and used as a guide for existing compression and indexing tools. Phylogenetic compression can then be applied to collections of assemblies, de Bruijn graphs, and *k*-mer indexes, and can be run in parallel for efficient processing. The resulting compression yields benefits ranging from a quick download, through a reduction of Internet bandwidth and storage costs, to efficient search on personal computers. We show this by implementing BLAST-like search to all sequenced pre-2019 bacterial isolates, which allow us to align genes, plasmids, and sequencing reads on an ordinary laptop or desktop within a few hours, a task that was completely infeasible with previous techniques. Phylogenetic compression has wide applications in computational biology and may provide a fundamental design principle for future genomics infrastructure.

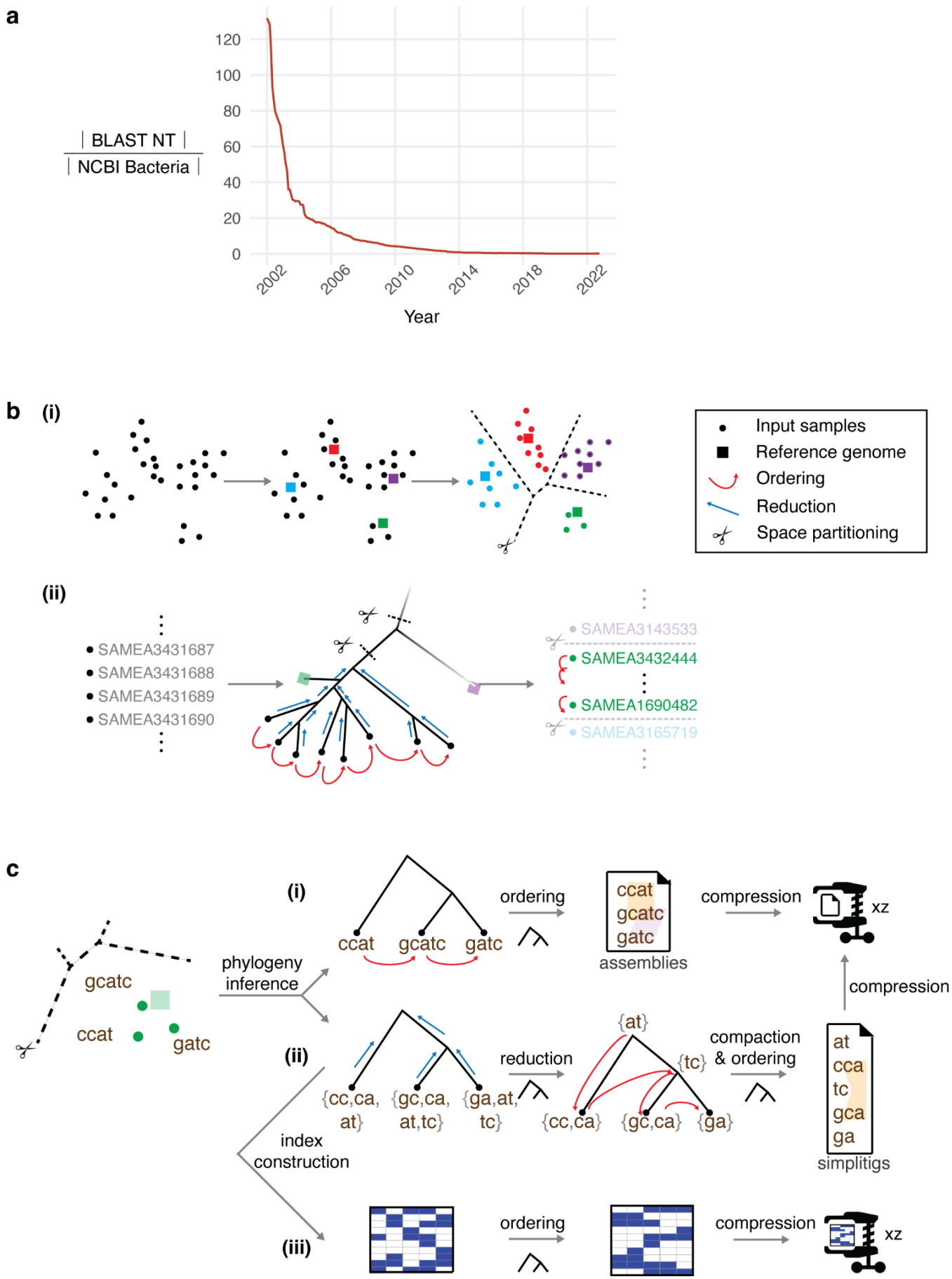
a
| BLAST NT |
| NCBI Bacteria |
120
100
80
60
40
20
0
2002
2006
2010
2014
2018
2022
Year
b
(i)
Input samples
Reference genome
Ordering
Reduction
Space partitioning
(ii)
SAMEA3431687
SAMEA3431688
SAMEA3431689
SAMEA3431690
SAMEA3143533
SAMEA3432444
SAMEA1690482
SAMEA3165719
c
gcatc
ccat
gatc
phylogeny inference
(i)
ccat
gcatc
gatc
ordering
ccat
gcatc
gatc
assemblies
compression
xz
compression
(ii)
{cc,ca, at}
{gc,ca, at,tc}
{ga,at, tc}
reduction
{at}
{tc}
{cc,ca}
{gc,ca}
{ga}
compaction & ordering
at
cca
tc
gca
ga
simplitigs
index construction
(iii)
ordering
compression
xz

Fig. 1: Overview of phylogenetic compression and its applications to different data types. **a)** Exponential decrease of data searchability over the past two decades – the size of the BLAST NT database divided by the size of the NCBI Bacterial Assembly database, as a function of time (Methods). **b)** The first three stages of phylogenetic compression before the application of a low-level compressor. **(i)** Partitioning genome collection into size- and diversity-balanced batches using metagenomic classification. **(ii)** Reversible reordering of input data according to their phylogeny, applied per batch. **c)** Examples of specific protocols of phylogenetic compression for individual data types, applied per batch. **(i)** For assemblies, data are sorted left-to-right according to the phylogeny and then compressed using a low-level compressor such as XZ or MBGC [17]. **(ii)** For de Bruijn graphs, *k*-mers are propagated bottom-up along the phylogeny, the newly obtained *k*-mer sets compacted into simplitigs, and compressed using XZ. **(iii)** For BIGSI *k*-mer indexes, Bloom filters (in columns) are ordered left-to-right according to the phylogeny and then compressed using XZ.

RESULTS

We developed a technique called phylogenetic compression for evolutionarily informed compression and search of microbial collections (**Fig. 1**). Phylogenetic compression combines four ingredients (**Fig. 1b**): 1) *clustering* of samples into phylogenetically related groups, followed by 2) inference of a *compressive phylogeny* that acts as a template for 3) *data reordering*, prior to an 4) application of a calibrated *low-level compressor/indexer* (Methods). This general scheme can be instantiated to individual protocols for various data types as we show in **Fig. 1c**; for instance, a set of bacterial assemblies can be phylogenetically compressed by XZ (the Lempel-Ziv Markov-Chain Algorithm [7], implemented in XZ utils, https://tukaani.org/xz/) by a left-to-right enumeration of the assemblies, with respect to the topology of their compressive phylogeny obtained through sketching [26].

We implemented phylogenetic compression for assemblies, de Bruijn graphs, and *k*-mer indexes in a framework called Microbes on a Flash Drive (MOF, http://karel-brinda.github.io/mof). We build upon the empirical observation that microbial genomes in public repositories usually form clusters corresponding to individual species [27], which we identify for individual genomes via standard metagenomic classification [28] (**Fig. 1b**, Methods). As some of the resulting clusters may be too large or too small, and thus unbalancing downstream parallelization, we further redistribute the clustered genomes into size- and diversity-balanced batches (Methods, **Supplementary Fig. 1**). This batching

enables compression and search in a constant time (using one node per batch on a cluster) or linear time (using a single machine) (Methods). For every batch, a compressive phylogeny is computed using Mashtree [26] and used for data reordering (Methods). Finally, the obtained reordered data are compressed per batch using particularly optimized XZ, and possibly further re-compressed or indexed using some general or specialized low-level tool, such as MBGC [17] or COBS [29] (Methods).

We calibrated and evaluated MOF using five microbial collections, selected as representatives of compression-related tradeoffs between characteristics including data quality, genetic diversity, genome size, and collection size (Methods, **Supplementary Table 1**). We quantified the distribution of their underlying phylogenetic signal (Methods, **Supplementary Table 2, Supplementary Fig. 2**), used them to calibrate the individual steps of the phylogenetic compression workflow (Methods, **Supplementary Fig. 3, Supplementary Fig. 4, Supplementary Fig. 5**), and evaluated the resulting performance, tradeoffs, and extremal characteristics (Methods, **Supplementary Table 3, Supplementary Fig. 6**). For instance, we found that, as one extreme, 591k SARS-CoV-2 genomes can be phylogenetically compressed using XZ to only 18.1 bytes/genome (Methods, **Supplementary Table 3**, **Supplementary Fig. 4,6**), resulting in a file size of 10.7 Mb (13× more compressed than GZip).

We found that phylogenetic compression improved the compression of genome assembly collections that comprise hundreds of thousands of isolates of over 1,000 species, by more than one order of magnitude compared to the state-of-the-art (**Fig. 2a**, **Supplementary Table 3**). As specialized compressors of high efficiency such as MBGC [17] are not applicable to highly diverse collections, the compression protocols deployed in practice for extremely large and diverse collections are still based on the standard GZip. One example is provided by the 661k datasets, containing all bacteria pre-2019 from ENA (n=661,405), which occupies 805 GB on a public FTP [15]. Here, MOF recompressed the collection to 29.0 GB (impr. 27.8×; 43.8 KB/genome, 0.0898 bits/bp, 5.23 bits/distinct *k*-mer) using XZ as the low-level tool, and even more to 20.7 GB (impr. 38.9×; 31.3 KB/genome, 0.0642 bits/bp, 3.74 bits/distinct *k*-mer) when combined with MBGC [17] that also accounts for reverse complements (**Fig. 2a**, **Supplementary Table 3**, Methods). Additionally, we found that the lexicographically ordered ENA datasets, as being partially phylogenetically ordered, can be used as a first-order approximation of phylogenetic compression, with compression performance, degraded only by a factor of 4.17 compared to the full phylogenetic compression (**Supplementary Table 3**, Methods). Phylogenetic compression proceeded through several hundred batches of at most 4k genomes per batch (**Supplementary Fig. 1**). The resulting compressed files are provided for download from Zenodo (**Supplementary Table 4**).

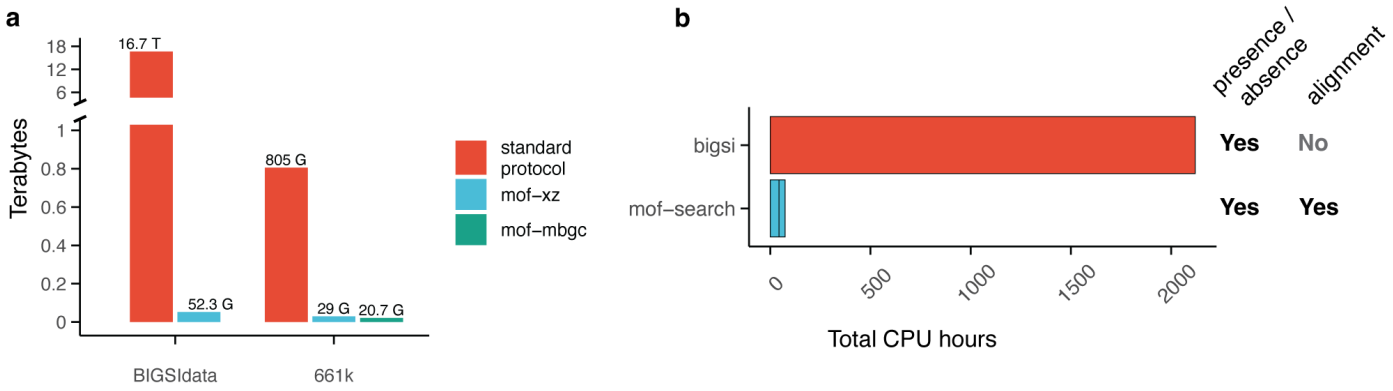

Fig. 2: Results of phylogenetic compression. a) Compression of the two comprehensive genome collections: BIGSI (425k de Bruijn graphs, the standard compression proceeds by McCortex binary files) and 661k (661k bacterial assemblies, the standard protocol is based on GZip). **b)** Comparison of the MOF vs. BIGSI methods on search of all plasmids from the EBI database. For MOF-Search, the split of the times of matching and alignment is denoted by a vertical bar.

We then studied the compression of de Bruijn graphs, which are a popular genome representation directly applicable to raw read data [16,30], and found that phylogenetic compression can improve state-of-the-art approaches by one to two orders of magnitude (**Fig. 2a**, **Supplementary Table 3,** Methods). As de Bruijn graphs lack practical methods for joint compression, single graphs are usually distributed individually [31]. For instance, the graphs of the BIGSIdata collection [16], comprising all viral and bacterial genomes from pre-2016 ENA (n=447,833), are provided in an online repository in the McCortex binary format [32] and occupy in total >16.7 TB (Methods). Here, we managed to retrieve n=425,160 graphs from the Internet (94.5% of the original count) (Methods) and losslessly recompressed them using the MOF methodology, with a bottom-up propagation of the *k*-mer content, to 52.3 GB (impr. 319×; 123. KB/genome, 0.166 bits/simplitig bp [33], 10.2 bits/distinct *k*-mer) (**Fig. 2a**, **Supplementary Table 3**, Methods). As recent advances in de Bruijn graph indexing [20] may lead to more efficient storage protocols in the future, we also compared MOF to MetaGraph [30], an optimized tool for indexing on high-performance servers with a large amount of memory. Here, we found that MOF still provided an improvement of nearly one order of magnitude (Methods).

Phylogenetic compression can be applied to any data structure as long as it is based on a similarity-preserving genome representation. We demonstrate this using the Bitsliced Genomic Signature Index (BIGSI) [16] (**Fig. 1c(iii)**), a *k*-mer indexing method using an array of Bloom filters, which is widely used

185 for large-scale genotyping and presence/absence queries of genomic elements [15,16]. Using the same data,
186 batches, and orders as inferred previously, we phylogenetically compressed the BIGSI indexes of the 661k
187 collection, computed using a modified version of COBS [29] (**Supplementary Table 5**, Methods).
188 Phylogenetic compression provided an 8.51× overall improvement compared to the original index (from
189 937 GB to 110 GB), making it finally applicable on ordinary computers. Removing the low-quality
190 genomes from the precomputed batches decreased the uncompressed index size by 4.9% (removing 3.7%
191 of genomes, **Supplementary Fig. 7**), but the resulting phylogenetic compression improved to 12.3×
192 (72.8 GB) (**Supplementary Table 5**).
193
194 We found that the most divergent genomes occupied 9.4× higher proportion of the database after
195 compression, both for assemblies and COBS *k*-mer indexes (**Supplementary Fig. 8**). On the other
196 hand, the top ten species (accounting for 80% of the genomic content) occupied less than half of the
197 compressed database after compression. The remarkable similarity of the post-compression species
198 ratios between assemblies and *k*-mer indexes suggests that compressibility is governed by the same rules,
199 regardless of the specific data representation used, with divergent genomes as a major driver of the final
200 size.
201
202 To demonstrate the utility of phylogenetic compression in practice, we implemented BLAST-like search
203 across all pre-2019 bacteria for standard desktop and laptop computers (MOF-search,
204 http://github.com/karel-brinda/mof-search). For a given batch of queries, MOF-search first filters
205 reference genomes using phylogenetically compressed COBS *k*-mer indexes [29], and then computes
206 alignment using Minimap 2 [34] while iterating over phylogenetically compressed genome assemblies
207 (Methods). The tool choice was arbitrary, and other programs could readily be used instead. Despite the
208 size of the original database, this resulted in total download and storage requirements of only 102 GB
209 (195 KB/genome, 0.329 bits/bp, 23.0 bits/distinct *k*-mer) and memory requirements starting from 12 GB
210 (user-specified) (**Supplementary Table 7**); therefore, the pipeline is deployable on all modern laptop
211 and desktop computers.
212
213 We first evaluated MOF-search with 661k-HQ using three different types of queries - resistance genes
214 (the ARG-ANNOT database of resistance genes [35], n=1,856), plasmids (EBI plasmid database, n=2,826),
215 and a nanopore sequencing experiment (n=158,583 reads), and found consistent performance with
216 results available within several hours (**Supplementary Table 2**). To benchmark against other tools, we
217 were unable to find any tool capable of aligning queries to 661k-HQ in a comparable setup (excluding
218 solutions based on an extensive parallelization on a compute cluster). We therefore used the EBI plasmid
219 dataset to compare MOF-Search to BIGSI with its original database of 448k genomes (which is
220 essentially a subset of the 661k-HQ) [16]. We found that MOF-search was over an order of magnitude faster

(**Fig. 2b, Supplementary Table 6**); the search required 74.1 CPU hours and provided an improvement in performance of a factor of 28.6× compared to the same BIGSI benchmark with its smaller database [16] (1.43× less genomes compared to 661k-HQ) (**Fig. 2b, Supplementary Table 6**), while providing the full alignments rather than presence/absence only (**Fig. 2b)**. This is to our knowledge the first time when alignment on this scale has been performed.

DISCUSSION

It is hard to overstate the impact on bioinformatics of BLAST [2], which has allowed biologists across the world to simply and rapidly compare their sequence of interest with essentially all known genomes – to the extent that the tool name has become a verb. The web version provided by NCBI/EBI is so standard that it is easy not to think how representative or complete its database is. However, twenty-three years on, sequencing data is far outstripping BLAST's ability to keep up, and in fact the publicly BLAST-able fraction of all sequenced microbes is shrinking exponentially (**Fig. 1a**). Much work has gone into approximate solutions [20], but full alignment to the complete corpus of bacterial genomes has remained completely impossible. We have addressed this problem and made significant progress, via phylogenetic compression, a highly efficient general technique using evolutionary history of microbes to improve existing algorithms and data structures. Performance of compression and search improves by one to two orders of magnitude. More concretely, BLAST-like search of all microbes moves from the impossible to the possible, not just for NCBI/EBI, but for anyone on their laptop. There are wide-ranging benefits, ranging from an easy and rapid download of large and diverse genome collections, through reductions in bandwidth, transmission/storage costs and computational time.

As with all compression, our capability to reduce data is fundamentally limited by the underlying information entropy. For genome collections, this is not just introduced by the underlying signal, but also tightly connected with the sequencing process and our ability to reconstruct the genomes from sequencing reads. The underlying *k*-mer histograms (**Supplementary Fig. 7**) suggest that any methods for compression or search will have to address noise in the form of contamination, missing regions, and technological artifacts, with legacy data being a major issue for both storage and analysis. Future methods may choose to incorporate stricter filtering, and as our experiments demonstrated, this will help not only to reduce the data volume, but also improve the quality of the search output. We note that this problem may be mitigated by novel computational approaches such as taxonomic filters [36] or sweep deconvolution [37].

Many elements of our approach have been used previously in other contexts. Reversible reordering for improving compression is in the core of the Burrows-Wheeler Transform [21] and its associated indexes [38,39], and it has also been used for read compression [40]. Tree hierarchies have been applied in metagenomics for lossy [22,23,41] and lossless [24] reference data compression. Finally, a divide-and-conquer methodology has been used for accelerating inference of species trees [42].

In the light of technological development, the benefits of phylogenetic compression will grow in time. Only a fraction of the world's microbial diversity has been sequenced, but as more is sequenced, the tree of life will not change, thus the relative advantage of phylogenetic compression will improve. We foresee its use from mobile devices to large distributed cloud environments, and anticipate promising applications in global epidemiological surveillance [43] and rapid diagnostics [44]. Overall, phylogenetic compression of data structures has broad applications across computational biology and provides a fundamental design principle for future genomics infrastructure.

REFERENCES

1. Stephens, Z. D. *et al.* Big Data: Astronomical or Genomical? *PLoS Biol.* **13**, e1002195 (2015).
2. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410 (1990).
3. Navarro, G. & Mäkinen, V. Compressed full-text indexes. *ACM Computing Surveys* **39**, 2-es (2007).
4. Loh, P.-R., Baym, M. & Berger, B. Compressive genomics. *Nat. Biotechnol.* **30**, 627–630 (2012).
5. Yu, Y. W., Daniels, N. M., Danko, D. C. & Berger, B. Entropy-Scaling Search of Massive Biological Data. *Cell Systems* **1**, 130–140 (2015).
6. Giancarlo, R., Scaturro, D. & Utro, F. Textual data compression in computational biology: a synopsis. *Bioinformatics* **25**, 1575–1586 (2009).
7. Salomon, D. & Motta, G. *Handbook of Data Compression*. (Springer London).
8. Daniels, N. M. *et al.* Compressive genomics for protein databases. *Bioinformatics* **29**, i283-90 (2013).
9. Deorowicz, S. & Grabowski, S. Data compression for sequencing data. *Algorithms Mol. Biol.* **8**, 25 (2013).
10. Giancarlo, R., Rombo, S. E. & Utro, F. Compressive biological sequence analysis and archival in the

288 era of high-throughput sequencing technologies. *Brief. Bioinform.* (2013) doi:10.1093/bib/bbt088.
289 11. Zhu, Z., Zhang, Y., Ji, Z., He, S. & Yang, X. High-throughput DNA sequence data compression. *Brief.*
290 *Bioinform.* **16**, 1–15 (2015).
291 12. Hosseini, M., Pratas, D. & Pinho, A. J. A Survey on Data Compression Methods for Biological
292 Sequences. *Information* **7**, 56 (2016).
293 13. Jayasankar, U., Thirumal, V. & Ponnurangam, D. A survey on data compression techniques: From
294 the perspective of data quality, coding schemes, data type and applications. *Journal of King Saud*
295 *University - Computer and Information Sciences* **33**, 119–140 (2021).
296 14. Navarro, G. Indexing Highly Repetitive String Collections, Part I: Repetitiveness Measures. *ACM*
297 *Comput. Surv.* **54**, 1–31 (2021).
298 15. Blackwell, G. A. *et al.* Exploring bacterial diversity via a curated and searchable snapshot of archived
299 DNA sequences. *PLoS Biol.* **19**, e3001421 (2021).
300 16. Bradley, P., den Bakker, H. C., Rocha, E. P. C., McVean, G. & Iqbal, Z. Ultrafast search of all
301 deposited bacterial and viral genomic data. *Nat. Biotechnol.* **37**, 152–159 (2019).
302 17. Grabowski, S. & Kowalski, T. M. MBGC: Multiple Bacteria Genome Compressor. *Gigascience* **11**,
303 (2022).
304 18. Deorowicz, S., Danek, A. & Li, H. AGC: compact representation of assembled genomes with fast
305 queries and updates. *Bioinformatics* **39**, (2023).
306 19. Zielezinski, A., Vinga, S., Almeida, J. & Karlowski, W. M. Alignment-free sequence comparison:
307 benefits, applications, and tools. *Genome Biol.* **18**, 186 (2017).
308 20. Marchet, C. *et al.* Data structures based on k-mers for querying large collections of sequencing data
309 sets. *Genome Res.* **31**, 1–12 (2021).
310 21. Burrows, M. & Wheeler, D. J. *A Block-sorting Lossless Data Compression Algorithm.* (1994).
311 22. Wood, D. E. & Salzberg, S. L. Kraken: ultrafast metagenomic sequence classification using exact
312 alignments. *Genome Biol.* **15**, R46 (2014).
313 23. Kim, D., Song, L., Breitwieser, F. P. & Salzberg, S. L. Centrifuge: rapid and sensitive classification of
314 metagenomic sequences. *Genome Res.* **26**, 1721–1729 (2016).

315 24. Břinda, K. Novel computational techniques for mapping and classification of Next-Generation
316 Sequencing data. (Université Paris-Est, 2016).
317 25. Břinda, K., Salikhov, K., Pignotti, S. & Kucherov, G. *ProPhyle: An accurate, resource-frugal and*
318 *deterministic DNA sequence classifier*. (Zenodo, 2017). doi:10.5281/zenodo.1045429.
319 26. Katz, L. *et al.* Mashtree: a rapid comparison of whole genome sequence files. *J. Open Source Softw.*
320 **4**, 1762 (2019).
321 27. Jain, C., Rodriguez-R, L. M., Phillippy, A. M., Konstantinidis, K. T. & Aluru, S. High throughput ANI
322 analysis of 90K prokaryotic genomes reveals clear species boundaries. *Nat. Commun.* **9**, 5114 (2018).
323 28. Breitwieser, F. P., Lu, J. & Salzberg, S. L. A review of methods and databases for metagenomic
324 classification and assembly. *Brief. Bioinform.* **20**, 1125–1136 (2019).
325 29. Bingmann, T., Bradley, P., Gauger, F. & Iqbal, Z. COBS: A Compact Bit-Sliced Signature Index. in
326 *String Processing and Information Retrieval* 285–303 (Springer International Publishing, 2019).
327 30. Karasikov, M. *et al.* MetaGraph: Indexing and Analysing Nucleotide Archives at Petabase-scale. *Cold*
328 *Spring Harbor Laboratory* 2020.10.01.322164 (2020) doi:10.1101/2020.10.01.322164.
329 31. Rahman, A., Chikhi, R. & Medvedev, P. Disk compression of k-mer sets. *Algorithms Mol. Biol.* **16**, 10
330 (2021).
331 32. Turner, I., Garimella, K. V., Iqbal, Z. & McVean, G. Integrating long-range connectivity information
332 into de Bruijn graphs. *Bioinformatics* **34**, 2556–2565 (2018).
333 33. Břinda, K., Baym, M. & Kucherov, G. Simplitigs as an efficient and scalable representation of de
334 Bruijn graphs. *Genome Biol.* **22**, 96 (2021).
335 34. Li, H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* 1–3 (2018).
336 35. Gupta, S. K. *et al.* ARG-ANNOT, a New Bioinformatic Tool To Discover Antibiotic Resistance Genes
337 in Bacterial Genomes. *Antimicrob. Agents Chemother.* **58**, 212–220 (2014).
338 36. Goig, G. A., Blanco, S., Garcia-Basteiro, A. L. & Comas, I. Contaminant DNA in bacterial sequencing
339 experiments is a major source of false genetic variability. *BMC Biol.* **18**, 24 (2020).
340 37. Mäklin, T. *et al.* Bacterial genomic epidemiology with mixed samples. *Microb Genom* **7**, (2021).
341 38. Ferragina, P. & Manzini, G. Opportunistic data structures with applications. in *Proceedings 41st*

342 *Annual Symposium on Foundations of Computer Science* 390–398 (IEEE Comput. Soc, 2000).
343 39. Gagie, T., Navarro, G. & Prezza, N. Fully Functional Suffix Trees and Optimal Text Searching in
344 BWT-Runs Bounded Space. *J. ACM* **67**, 1–54 (2020).
345 40. Chandak, S., Tatwawadi, K. & Weissman, T. Compression of genomic sequencing reads via hash-
346 based reordering: algorithm and analysis. *Bioinformatics* **34**, 558–567 (2018).
347 41. Ames, S. K. *et al.* Scalable metagenomic taxonomy classification using a reference genome database.
348 *Bioinformatics* **29**, 2253–2260 (2013).
349 42. Molloy, E. K. & Warnow, T. Statistically consistent divide-and-conquer pipelines for phylogeny
350 estimation using NJMerge. *Algorithms Mol. Biol.* **14**, 14 (2019).
351 43. Gardy, J. L. & Loman, N. J. Towards a genomics-informed, real-time, global pathogen surveillance
352 system. *Nat. Rev. Genet.* (2017) doi:10.1038/nrg.2017.88.
353 44. Břinda, K. *et al.* Rapid inference of antibiotic resistance and susceptibility by genomic neighbour
354 typing. *Nat Microbiol* **5**, 455–464 (2020).
355 45. Kitts, P. A. *et al.* Assembly: a resource for assembled genomes at NCBI. *Nucleic Acids Res.* **44**, D73-
356 80 (2016).
357 46. Tonkin-Hill, G., Lees, J. A., Bentley, S. D., Frost, S. D. W. & Corander, J. RhierBAPS: An R
358 implementation of the population clustering algorithm hierBAPS. *Wellcome Open Res.* **3**, 93 (2018).
359 47. Lu, J. *et al.* Metagenome analysis using the Kraken software suite. *Nat. Protoc.* **17**, 2815–2839
360 (2022).
361 48. Wood, D. E., Lu, J. & Langmead, B. Improved metagenomic analysis with Kraken 2. *Genome Biol.*
362 **20**, 257 (2019).
363 49. Lu, J., Breitwieser, F. P., Thielen, P. & Salzberg, S. L. *Bracken: Estimating species abundance in*
364 *metagenomics data*. 1–14 http://biorxiv.org/lookup/doi/10.1101/051813 (2016) doi:10.1101/051813.
365 50. Broder, A. Z. On the resemblance and containment of documents. in *Proceedings. Compression and*
366 *Complexity of SEQUENCES 1997 (Cat. No.97TB100171)* 21–29 (IEEE Comput. Soc, 1997).
367 51. Fan, H., Ives, A. R., Surget-Groba, Y. & Cannon, C. H. An assembly and alignment-free method of
368 phylogeny reconstruction from next-generation sequencing data. *BMC Genomics* **16**, 522 (2015).

369 52. Saitou, N. & Nei, M. The neighbor-joining method: a new method for reconstructing phylogenetic
370 trees. *Mol. Biol. Evol.* **4**, 406–425 (1987).
371 53. Gascuel, O. Neighbor-Joining Revealed. *Mol. Biol. Evol.* **23**, 1997–2000 (2006).
372 54. Howe, K., Bateman, A. & Durbin, R. QuickTree: building huge Neighbour-Joining trees of protein
373 sequences. *Bioinformatics* **18**, 1546–1547 (2002).
374 55. Brinda, K. Novel computational techniques for mapping and classification of Next-Generation
375 Sequencing data. (Université Paris-Est, 2016).
376 56. Marçais, G. & Kingsford, C. A fast, lock-free approach for efficient parallel counting of occurrences of
377 k-mers. *Bioinformatics* **27**, 764–770 (2011).
378 57. Köster, J. & Rahmann, S. Snakemake-a scalable bioinformatics workflow engine. *Bioinformatics* **28**,
379 2520–2522 (2012).
380 58. Grad, Y. H. *et al.* Genomic Epidemiology of Gonococcal Resistance to Extended-Spectrum
381 Cephalosporins, Macrolides, and Fluoroquinolones in the United States, 2000–2013. *J. Infect. Dis.*
382 **214**, 1579–1587 (2016).
383 59. Zerbino, D. R. & Birney, E. Velvet: algorithms for de novo short read assembly using de Bruijn
384 graphs. *Genome Res.* **18**, 821–829 (2008).
385 60. Elbe, S. & Buckland-Merrett, G. Data, disease and diplomacy: GISAID's innovative contribution to
386 global health. *Glob Chall* **1**, 33–46 (2017).
387 61. roblanf & Mansfield, R. *roblanf/sarscov2phylo: 13-11-20*. (Zenodo, 2020).
388 doi:10.5281/ZENODO.3958883.
389 62. Iqbal, Z., Caccamo, M., Turner, I., Flicek, P. & McVean, G. De novo assembly and genotyping of
390 variants using colored de Bruijn graphs. *Nat. Genet.* **44**, 226–232 (2012).
391 63. Bradley, P. & Iqbal, Z. Supplementary Info. (2017) doi:10.6084/M9.FIGSHARE.5702776.
392 64. Blackwell, G. *et al.* Additional material for article "Exploring bacterial diversity via a curated and
393 searchable snapshot of archived DNA sequences." (2021) doi:10.6084/M9.FIGSHARE.16437939.
394 65. Grad, Y. Data for "Genomic Epidemiology of Gonococcal Resistance to Extended-Spectrum
395 Cephalosporins, Macrolides, and Fluoroquinolones in the United States, 2000-2013." (2019)

396 doi:10.5281/ZENODO.2618836.
397 66. Corander, J., Marttinen, P., Sirén, J. & Tang, J. Enhanced Bayesian modelling in BAPS software for
398 learning genetic structures of populations. *BMC Bioinformatics* **9**, 539 (2008).
399 67. Croucher, N. J. *et al.* Rapid phylogenetic analysis of large samples of recombinant bacterial whole
400 genome sequences using Gubbins. *Nucleic Acids Res.* **43**, e15 (2015).
401 68. Lanfear, R. *A global phylogeny of SARS-CoV-2 sequences from GISAID*. (2020).
402 doi:10.5281/zenodo.4089815.
403 69. Tange, O. GNU Parallel: the command-line power tool. ;*login: The USENIX Magazine* **36**, 42–47
404 (2011).
405 70. Larsson, N. J. & Moffat, A. Off-line dictionary-based compression. *Proc. IEEE* **88**, 1722–1732
406 (2000).
407 71. Wan, R. Browsing and Searching Compressed Documents. (University of Melbourne, 2003).
408 72. Cock, P. J. A. *et al.* Biopython: freely available Python tools for computational molecular biology and
409 bioinformatics. *Bioinformatics* **25**, 1422–1423 (2009).
410 73. Grüning, B. *et al.* Bioconda: sustainable and comprehensive software distribution for the life
411 sciences. *Nat. Methods* **15**, 475–476 (2018).
412 74. Inouye, M. *et al.* SRST2: Rapid genomic surveillance for public health and hospital microbiology
413 labs. *Genome Med.* **6**, 90 (2014).
414

415 **METHODS**

416

417

418 ***Analysis of the decrease in bacteria BLAST searchability***

419

420 **BLAST NT size estimation.** The estimates of the size of the BLAST NT database (n=27) for the time
421 period between 2002-01-01 and 2022-11-01 were inferred from five types of online resources. First, most
422 recent values were recorded manually from the file size reported on the official NCBI website
423 https://ftp.ncbi.nih.gov/blast/db/FASTA/ (n=11, between 2020-04-05 and 2022-11-01); second,
424 additional values were obtained from the snapshots of this website and its other NCBI mirrors on
425 http://web.archive.org (n=7, between 2012-10-11 and 2022-06-06); third, the archived versions of the
426 NT database at selected time points were found in online repositories (n=3, between 2017-10-26 and
427 2021-01-15); fourth, the size of the database was also captured in a software documentation (n=1, 2013-
428 12-03); and fifth, the number of base pairs was also provided in scientific literature (n=5, between 2002-
429 01-01 and 2010-01-01) (**Supplementary Table 6**). To convert the size of the NT database between the
430 number of nucleotides and the size of the FASTA file after compressing using GZip; the compression
431 ratio was estimated using the NT version from 2022-06-20 to be approximately 2.04 bits per bp.

432

433 **NCBI Assembly DB size estimation.** The number of bacteria in the NCBI Assembly database and
434 their compressed size were estimated from the GenBank assembly summary file
435 (https://ftp.ncbi.nlm.nih.gov/genomes/genbank/bacteria/assembly_summary.txt, downloaded on
436 2022-11-02, n=1,280,758 records), and subsequently sorted according to the 'seq_rel_date' field. The
437 resulting file was used for calculating the number of published assemblies till a given date, further
438 aggregated per month. The total size of the assemblies was estimated from the average size of genome
439 assembly in the 661k collection, which is 3.90 Mbp, and the corresponding GZip size was estimated as
440 previously. We note that updates in the assembly_summary.txt file could affect the old statistics, such as
441 the removal of old contaminated records, but a manual inspection and comparison during a several-
442 months-long period revealed that these changes have only a negligible impact on the resulting statistics.

443

444 **Comparison.** The BLAST scalability plot (**Fig 1a**) depicts the estimated size of the BLAST NT database
445 (in the .fasta.gz format) divided by the estimated size of the bacteria in the NCBI Assembly Database [45]
446 (https://www.ncbi.nlm.nih.gov/assembly/) (also in the .fasta.gz format) at the same time, as a function
447 of time from 2002 to 2022. Both types of values, i.e., the sizes of the NT database and the bacteria in the
448 NCBI Assembly database, were interpolated in the logarithmic scale by piecewise linear functions. The
449 resulting interpolations were used for enumerating the estimated proportion of the sizes of NT and the
450 bacteria in the NCBI Assembly database, by calculating their values at regular intervals (for each month).

Although the provided calculation may involve little inaccuracies (for instance, the average bacterial genome size or GZip compression ratio might differ for the NCBI Assembly database), this would have only negligible impact on the overall exponential decrease of data searchability. The resulting approximations were used for plotting **Fig 1a**.

Conceptual overview of phylogenetic compression

General overview. Phylogenetic compression is a general approach for compressing arbitrarily-sized genome collections and indexes and to search them. While the existing compression techniques excel in local compression, they struggle with widely distributed redundancies. As genomic data result from a superposition of evolutionary and sampling processes, genomic collections feature a tree-like geometrical structure reflecting vertical descent and partially confounded less frequent horizontal transfer. Reordering according to the tree topology co-localizes correlated information within the input data, and thus increases the local compressibility of data – consecutive genomes in phylogenetic orders will often be highly similar. To organize input genomes into phylogenetic trees in a scalable manner, phylogenetic compression combines four conceptual steps.

Step 1: Clustering/batching (Fig. 1b(i)). The goal of this step is to separate genomes into batches of phylogenetically related genomes of limited size and diversity that can be easily compressed and searched together. In downstream compression, indexing, and analyses, individual batches are processed individually, in separation, and the guarantees on the maximum batch size and diversity enable us to establish upper bounds on the maximum time and space necessary for processing a single batch. The clustering and batching is achieved via metagenomic classification [28]; it is known from the literature that microbial genomes in public repositories form distinct clusters, usually (but not always) corresponding to individual species [27] and metagenomic classification assigns genomes to individual clusters, defined by the genomes used in the corresponding reference database (e.g., RefSeq). Clusters of divergent genomes that are too small are put into a separate joint pseudo-cluster called dustbin. As some of the obtained (pseudo)-clusters can be too big (such as the clusters corresponding to oversampled species; e.g., *S. enterica*), they are further divided into smaller batches in a way that provides guarantees on downstream computational resources. The clustering/batching step is not necessary if the number of genomes is sufficiently small (the order of thousands).

Step 2: Inference of compressive phylogeny (Fig. 1b(ii)). The second step, already performed per individual batches of a limited size and diversity independently, consists in inferring a so-called *compressive phylogeny* that sufficiently approximates the true phylogenetic signal for compression

487 purposes. While phylogenies computed using an accurate inference method such as BAPS [46] are
488 preferable, in most practical scenarios these are not available and would be too costly to compute or
489 require particular adjustments for different species. In such cases, a rapidly estimated phylogeny, for
490 instance using MashTree [26], is sufficient.
491
492 **Step 3: Data reduction/reordering (Fig. 1b(ii)).** The role of the computed compressive phylogeny
493 is to act as a template for the reduction and re-ordering of input data according to their evolutionary
494 history. This can have multiple different forms, based on the specific application and type of compression
495 (e.g., lossy vs. lossless), and it can involve two directions. Either the collection/batch is only reordered
496 left-to-right according to the topology of the compressive phylogeny, or the genomic data are propagated
497 bottom-up along the phylogeny (i.e., shared genomic content is propagated up, and thus reduced, before
498 the left-to-right enumeration is performed).
499
500 **Step 4: Compression or indexing using a calibrated low-level tool (Fig. 1c).** Once the data are
501 reordered (and possibly reduced) using the compressive phylogenies, the last step is the final
502 compression or indexing using a low-level tool that can exploit local redundancies in the data. At this
503 stage, all the data are highly locally compressible thanks to both the phylogeny-based clustering and
504 phylogeny-based reordering. Many general and specialized genome compressors are available and can be
505 used at this step; however, it is important to ensure that the parameters of the underlying algorithms
506 correspond to the characteristics of genome data; for instance, the window/dictionary of a Lempel-Ziv-
507 based compressor needs to be sufficiently large to span multiple genomes and to store a sufficient
508 amount of phrases (**Supplementary Fig. 3a**). General compressors usually need to be particularly
509 tested and calibrated, whereas specialized compressors for genomes are usually calibrated by default.
510 Furthermore, general compressors may require additional data re-formatting; for instance, for efficient
511 multi-genome compression using general compressors, it is important to ensure that FASTA files have
512 one sequence per one line (**Supplementary Fig. 3b**).
513
514
515 ***The Microbes on Flash Drive (MOF) workflow for phylogenetic compression***
516
517 MOF implements several protocols of phylogenetic compression for compression of assemblies, de Bruijn
518 graphs, and for search genome; more information and links can be found on the associated website
519 (http://karel-brinda.github.io/mof).
520
521 **Clustering/batching.** As individual genome collections encountered in practice can have very different
522 properties and associated data available, for the use with MOF the clustering and batching steps are

523 expected to be performed by the user. The recommended procedure is to identify species using standard
524 metagenomic approaches, such as those implemented in the Kraken software suite [47] (e.g., Kraken 2 [48]
525 and Bracken [49] applied on the original read sets) and divided into smaller batches analogically to the
526 examples in **Supplementary Figure 1**. The protocol can be further customized based on the specific
527 performance of algorithms downstream, e.g., by increasing/decreasing batch size or adjusting
528 parameters for building dustbin batches.
529
530 **Inference of a compressive phylogeny.** The user can either provide a custom tree, tailored for the
531 specific collection/batch, such as a tree computed by RHierBAPS [46], or leave MOF-Compress to compute
532 a compressive phylogeny by Mashtree [26], which is based on estimating *k*-mer-set Jaccard index using
533 locality sensitive hashing using MinHash sketches [50] and estimating mutation rate under a simple
534 evolutionary model [51] using the so-called Mash distance [50]; the obtained distances are then used for
535 estimating the likely phylogeny using the Neighbor-Joining algorithm [52,53] as implemented in
536 QuickTree [54].
537
538 **MOF-Compress** (http://github.com/karel-brinda/mof-compress)**.** This is a central package of MOF
539 that performs phylogenetic compression of a single batch and calculates the associated statistics. It
540 implements the following three protocols: 1) phylogenetic compression of assemblies based on a left-to-
541 right reordering, 2) phylogenetic compression of de Bruijn graphs represented by simplitigs [33] based on
542 the left-to-right reordering, and 3) phylogenetic compression of de Bruijn graphs using bottom-up *k*-mer
543 propagation using ProPhyle [25,55]. The *k*-mer propagation proceeds recursively inside the compressive
544 phylogeny in a bottom-up fashion – at every internal node, *k*-mer sets of the child nodes are loaded, their
545 intersection computed, stored at the node, the intersection subtracted from the child nodes, and all three
546 *k*-mer sets saved in the form of simplitigs. This progressively reduces the *k*-mer content within the
547 phylogeny in a lossless fashion. More details on this technique can be found in ref [55]. In all three
548 protocols, the output is a TAR file with ordered text files with sequences – for assemblies in the one-line
549 FASTA format and for simplitigs in a text file with eol-separated simplitigs. The TAR file is subsequently
550 compressed using XZ with the parameters 'xz -9 -T1' (see calibration). MOF-Compress also computes
551 extensive statistics for all three protocols, including the size of the corresponding *k*-mer multiset, *k*-mer
552 set, number of sequences, their cumulative length, and the resulting compressed sizes (see section
553 Statistics). The output .tar.xz file from MOF-Compress can be used for additional recompression or
554 indexing in the same order by other low-level tools.
555
556 **MOF-Compress statistics.** MOF-Compress computes a multitude of statistics characterizing the
557 compressibility using the three implemented protocols, and these are further used for computing global
558 statistics such as phylogeny-explained redundancy. For each of the three protocols, the following

statistics are calculated: set (the size of the *k*-mer set of all sequences), multiset (size of the *k*-mer multiset of all sequences), sum_ns (number of sequences), and sum_cl (total sequence length), recs (number of records), and xz_size (size after compression using XZ). The sizes of *k*-mer sets and multi-sets are obtained from *k*-mer histograms computed by JellyFish 2 [56]. Based on these numbers the various compression-related statistics used in this paper are computed, such as bits per distinct *k*-mer or kilobytes per genome.

Phylogeny-explained redundancy. Comparing the sizes of *k*-mer sets and multisets before and after reduction using *k*-mer propagation along phylogenies allows further quantification of the proportion of the *k*-mer signal that is explained by a given compressive phylogeny. The so-called *removed k-mer redundancy* quantifies the proportion of *k*-mer occurrences that were removed by *k*-mer propagation out of those that could be removed if the phylogeny perfectly explained the distribution of *k*-mers (i.e., every *k*-mer occurring only once after propagation), and the corresponding formula is

removed_redundancy = (|multiset_preprop|-|multiset_postprop|)/(|multiset_preprop|-|set|)

MOF-COBS-Build. MOF-COBS-build (https://github.com/leoisl/mof-cobs-build) is a pipeline that can be appended to MOF-Compress for constructing phylogenetically compressed ClaBS COBS indexes (Classical Bit-sliced index) for creating XZ-compressed COBS indexes from batches of phylogenetically-ordered samples. ClaBS is a mode of COBS that is conceptually analogous to the original BIGSI data structure [16], using Bloom filters of the same size, which is a key property that guarantees that Bloom filters of phylogenetically close datasets are mutually compressible (different sizes of Bloom filters would shift bits corresponding to the same *k*-mers to different positions). MOF-COBS-build is built as a Snakemake [57] workflow reading two directories: the first describes the sample batches as well as their ordering (in detail, this directory contains a list of text files with each such file listing the samples in the batch and their ordering, with one sample name per line). The second is a directory with the assemblies themselves. The workflow comprises four main steps: 1) creating groups of assemblies in the order specified by the input (in details this is done by creating a directory with symbolic links to the original assemblies, with these links having artificial names sorted by their phylogenetically order, forcing COBS to process them in such order); 2) building COBS classic indexes by “cobs classic-construct -T 8 {batch} {output}.cobs_classic”; 3) compressing the COBS classic indexes with “xz -9 -T1 -e -k -c --lzma2=preset=9,dict=1500MiB,nice=250”; 4) combining all compressed indexes into a single TAR file that can be further used for distribution.

594
595 ***Overview of the five test microbial collections***
596
597 **GISP.** The collection consists of 1,102 draft assemblies constructed from clinical isolates of *N.*
598 *gonorrhoeae* collected in the US from 2000 to 2013 by the Centers for Disease Control and Prevention
599 within the Gonococcal Isolate Surveillance Project [58]; the isolates were previously sequenced using
600 Illumina HiSeq and assembled using Velvet [59]. The collection presents a model of high-quality genomic
601 data from a low-diversity species sequenced and assembled using identical protocols.
602
603 **NCTC3k.** The collection consists of 1,065 draft and complete assemblies constructed from strains from
604 the National Collection of Type Cultures (NCTC) collection, analyzed by Public Health England, the
605 Wellcome Trust Sanger Institute, and Pacific Biosciences within the NCTC 3000 project; the isolates
606 were sequenced using the PacBio Single Molecule, Real-Time (SMRT) DNA Sequencing technology,
607 assembled using automated pipelines, and are provided online through the
608 https://www.sanger.ac.uk/resources/downloads/bacteria/nctc/ website. The collection presents a model
609 of nearly complete high-quality genomes of diverse species.
610
611 **SC2.** The collection is a snapshot of the GISAID database [60] from 2021-05-18 of 590,779 SARS-CoV-2
612 isolates (complete assemblies) with a known phylogeny and available complete genomic sequences,
613 collected and sequenced from 2020 to 2021 by various laboratories, and provided online through
614 https://gisaid.org/ and analyzed using the sarscov2phylo software
615 (https://github.com/roblanf/sarscov2phylo/, ref [61]). The collection presents a model of a large number of
616 genomes of varying quality from epidemiological surveillance of a single species collected across the
617 globe.
618
619 **BIGSIdata.** The BIGSIdata collection is a snapshot of bacterial and viral isolates present in the
620 European Nucleotide Archive (ENA) on December 2016 as published in ref [16], consisting of 425,160
621 cleaned de Bruijn graphs (k=31) that we managed to download from the associated FTP website
622 (http://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018, out of the original 447,833 that were
623 mentioned in ref [16]); the isolates had originally been collected and sequenced using various laboratories,
624 deposited to some repository that is synchronized with ENA (i.e., ENA, NCBI SRA, or DDBJ Sequence
625 Read Archive), downloaded and transformed into cleaned de Bruijn graphs using McCortex [32,62] by the
626 European Bioinformatics Institute (EBI) and provided on the FTP website together with metadata on
627 Figshare [63]. The collection presents a model of a large number of microbial isolates collected and
628 sequenced across the globe using various sequencing technologies that are represented in a searchable
629 representation other than genome assembly.

661k. This collection is an assembled snapshot [15] (draft assemblies) of all 661,405 Illumina-sequenced bacterial isolates present in the ENA on 2018-11-26; the isolates had originally been collected and sequenced using various laboratories, deposited to some repository that is synchronized with ENA (i.e., ENA, NCBI SRA, or DDBJ Sequence Read Archive), downloaded and assembled using a single unified pipeline (https://github.com/iqbal-lab-org/assemble-all-ena) based on Shovill (https://github.com/tseemann/shovill) by the European Bioinformatics Institute (EBI), and provided on FigShare [64] (metadata) on FTP (https://ftp.ebi.ac.uk/pub/databases/ENA2018-bacteria-661k/, assemblies). The collection presents a model of a large number of assembled microbial isolates collected and sequenced across the globe using a single sequencing technology, i.e., the state-of-the-art of the short read-assembly era.

Basic characteristics of the test collections, including the size of the original files, the number of samples, as well as the number of species, and the number of distinct *k*-mers are provided in **Supplementary Table 1**.

Acquisition of the test collections

BIGSIdata. The files corresponding to individual samples of the collection [16] were downloaded from the associated FTP (http://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018/), including cleaned de Bruijn graphs, and taxonomic information as inferred using metagenomic classification using Kraken [22] and abundance reports computed using Bracken [49]. The download was done in groups corresponding to individual EBI prefixes (e.g., DRR000) using RSync by

```
rsync -avP --min-size=1 --exclude '*stats*' --exclude '*uncleaned*' --exclude '*bloom*' --exclude
'*log* "rsync://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018/ctx/${prefix}"
```

The prefixes were further organized into batches by 100, which resulted in 15 batches in total. The batches were processed sequentially, and the individual contained groups were downloaded on a research computing cluster in parallel using Slurm, with jobs deployed using Snakemake [57] (between 2020-08-01 and 2020-09-15). From the downloaded McCortex files, unitigs were extracted using McCortex ("mccortex31 unitigs -m 3G -") and stored locally, after which the McCortex files were deleted. Only graphs with unitigs of length at least 2 kbp, with less then 15 M *k*-mers (to remove contaminated datasets), and without any file system errors were used in the subsequent processing. This resulted in n=425,161 de Bruijn graphs (out of the original 463,331 files) that were used in the subsequent analyses.

661k. All assemblies were retrieved in March 2022 from the official FTP repository provided in ref[15], by running

rsync -avp rsync://ftp.ebi.ac.uk/pub/databases/ENA2018-bacteria-661k/Assemblies/{pref}

for individual prefixes ranging from 000 to 661. This resulted in n=661,405 .fa.gz files, occupying in total 805,255,364,491 bytes (i.e., 805 GB).

GISP. The GISP collection was obtained from the https://github.com/c2-d2/rase-db-ngonorrhoeae-gisp, published in ref [44]; the original data were originally analyzed in ref [58] and the resulting data later provided for download also on Zenodo [65]. The GISP assemblies (n=1,102) were obtained from the "isolates/contigs" subdirectory of Github repository, and the associated phylogenetic tree, computed using BAPS [66] (Bayesian Analysis of Population Structure) after correction for recombination using Gubbins[67], downloaded from the "tree/" subdirectory of the same repository.

SC2. The following SARS-Cov-2 data were downloaded from the GISAID website (https://www.gisaid.org/, as of 2021-05-18): an assembly file (sequences_fasta_2021_05_18.tar.xz, n=1,593,858) and a Sarscov2phylo phylogeny[68] data file (gisaid-hcov-19-phylogeny-2021-05-11.zip, n=590,952). Both datasets were converted to the same set of identifiers, and isolates with missing data discarded. This resulted in 590,779 genomes accompanied with their corresponding phylogenetic tree (**Tab. 2**; the SC2 collection). Out of the downloaded 1,593,858 sequences that were available in May 2021, we first extracted those with known phylogenetic position within the global Sarscov2phylo phylogeny [68]; this resulted in 590,779 genomes accompanied with their corresponding phylogenetic tree (**Tab. 2**; the SC2 collection) (as of May 2021; n=590,779 sequences with phylogenetic information out of the total of 1,593,858; Methods)

NCTC3k. The assemblies were downloaded in the GFF format using FTP from ftp://ftp.sanger.ac.uk/pub/project/pathogens/NCTC3000 by

wget -m -np -nH --cut-dirs 3 --retr-symlinks ftp://ftp.sanger.ac.uk/pub/project/pathogens/NCTC3000 .

converted them to the FASTA format by any2fasta (https://github.com/tseemann/any2fasta, v0.4.2) parallelized by GNU Parallel [69], and finally uploaded to Zenodo (http://doi.org/10.5281/zenodo.4838517). Species were counted based on the data in the main Sanger/Public Health England assembly table for NCTC3000 as provided online (https://www.sanger.ac.uk/resources/downloads/bacteria/nctc/, retrieved on 2022-09-14). The HTML table was first manually exported to the XLSX, and then used for constructing a translation table from

701 NCTC accession numbers to the corresponding species. Finally, the accessions of the assemblies
702 contained in our collection were extracted from file names, translated to species, and the species counted.
703 Overall, this resulted in 1,065 assemblies of 259 species.
704
705
706 ***Calibration and evaluation of phylogenetic compression using the test collections***
707
708 For calibration, two collections from epidemiological surveillance (SC2 and GISP) were used to model
709 similar genomes, and one additional high-diversity isolate collection (NCTC3k) was used to model
710 divergent genomes (**Supplementary Tab. 1**). These three collections were used for calibrating and
711 comparing individual low-level compressors, as well as for evaluating the compressibility of the datasets
712 (**Supplementary Fig. 3–5**).
713
714 **Calibration of the XZ compressor (Supplementary Fig. 3).** The comparison was performed using
715 the GISP collection and the GZip, BZ2, and XZ low-level compressors with range of their presets, and in
716 the case of XZ with only 1 thread. For the right panel, line length was progressively modified using seqtk
717 seq (the "-l" parameter), the collection recompressed, and all the final results compared in function of
718 line length.
719
720 **Comparison of scaling modes (Supplementary Fig. 4).** The SC2 collection was provided in the
721 left-to-right order according to the topology of the phylogeny and compressed, with genomes being
722 progressively subsampled. The compression methods in this experiment included XZ ("xz -9 -T1"), BZip2
723 ("bzip2 --best"), GZip ("gzip -9"), and Re-Pair [70,71] (implementation from
724 https://github.com/rwanwork/Re-Pair, "repair -v -i", version "Oct 26 2021"). As Re-Pair was only little
725 scalable and suffered from various technical issues, the integrity of the output files was always verified via
726 their decompression. The comparison for the NCTC3k collection was done analogically via MOF-
727 Compress with individual subsampling, and additional re-recompression using GZip and BZip2 with the
728 same parameters as previously. In the case of the SC2 collection, sequence names were not included in
729 the benchmark to their long names given the short genomes.
730
731 **Order comparison (Supplementary Fig. 5).** For SC2, the isolates with phylogenetic information
732 (n=590,779) were used for the compression analysis using three orderings: the original ordering
733 (corresponding to the lexicographical ordering by sequence names), the left-to-right ordering of the
734 phylogeny, and a randomized order. In all cases, a custom Python script using BioPython [72] was used to
735 order the FASTA file and remove sequence names, and its output was compressed by the XZ compressor

using 1 thread and maximum compression ('xz -T1 -9'), and the sizes of the resulting files measured using wc ('wc -c'). The comparisons for GISP and NCTC3k was performed analogically.

Summary of the findings. The most popular method, GZip, always performed poorly for bacteria, but provided a moderate scaling for viruses. Stronger compressors such XZ achieved steep compression curves for low-diversity collections, with compression ratio improving by one order per one order increase of #genomes, for both viruses and bacteria. On the other hand, NCTC3k was little compressible even with the best approaches (<1 order of magnitude of compression after a 3 orders-of-magnitude increase of #genomes), indicative of that divergent genomes is the fundamental compression bottleneck within comprehensive collections. Finally, we compared the best experimental grammar-based compressor (Re-Pair [70,71]) to XZ, and found they achieved similar asymptotics, suggesting the potential of grammar compression for phylogenetic compression. As for the orders, we found that phylogeny reordering always boosted compression (reduction to 38%–67% compared to the random order), for both low- and high-diversity collections. We also found that trees computed using rapid heuristics (MashTree [26]) performed nearly as well as an accurate Bayesian approach (Bayesian Analysis of Population Structure [46]). Overall, based on the observed tradeoffs, we selected "XZ -9 -T1" as the compression procedure for MOF-Search and Mashtree as a sufficiently accurate method for generating compressive phylogenies.

Phylogenetic compression of the BIGSIdata collection of de Bruijn graphs

Clustering and batching. For every sample, the output of Kraken [22] and Bracken [49] were extracted from the downloaded data. Clusters were then defined based on the most prevalent species in a sample, as identified in the corresponding Bracken report and batching proceeded as depicted in **Supplementary Fig. 1**. The genomes of the 1,443 identified species (clusters) were redistributed into 568 regular batches and 6 dustbin batches, resulting in a total of 574 batches.

Phylogenetic compression. Phylogenetic compression first proceeded through a workflow that later resulted in MOF-Compress For individual batches, compressive phylogenies were computed using Mashtree with the default parameters. The resulting trees were then used with ProPhyle and unitig files to propagate *k*-mers along the compressive phylogenies and to compute simplitigs using ProphAsm [33]. After the resulting files were ordered and compressed by XZ ("xz -v -z -9 -T8 --stdout"), the resulting files (occupying 74.4 GB) were deposited on https://doi.org/10.5281/zenodo.4086456 and https://doi.org/10.5281/zenodo.4087330. Furthermore, an analogical version of the propagated simplitig files, but without sequence headers and with compression using a single thread only, was later

771 created using the MOF compress pipeline and resulted in files occupying in total 52.3 GB that were
772 subsequently deposited on https://doi.org/10.5281/zenodo.5555253.

773 **Phylogenetic decompression.** To decompress the files obtained through phylogenetic compression
774 based on *k*-mer propagation back to the original de Bruijn graphs, the original graphs need to be
775 reconstructed by collecting all *k*-mers along root-to-leaf paths, which we implemented a program called
776 MOF-Client (https://github.com/karel-brinda/mof-client). The program downloads individual data files
777 from Zenodo from the accessions above and decompresses them using the following procedure. For it
778 decompresses the XZ file of a given batch, splits it according to files corresponding to individual nodes of
779 the compressive phylogeny, recompressed individual nodes using GZip parallelized using GNU
780 Parallel [69], and for all leaves (genomes) it collects the corresponding *k*-mer sets from by merging all GZip
781 files along the corresponding root-to-leaf paths using the Unix cat command. The correctness of the
782 resulting files was confirmed using JellyFish [56].

783 **Comparison to the original compression.** As the samples in our BIGSIdata collection do not fully
784 correspond to the selected data used in the original publication [16], we calculated the original size of the
785 published McCortex files of our graphs based on the FTP listoff files as provided within individual
786 subdirectories of http://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018/ (as of 2021-08-27). These
787 were downloaded per individual prefix directories recursively using wget by

788 *wget -nv -e robots=off -np -r -A .html*
789 *"http://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018/ctx/${prefix}/",*

790 The corresponding parallelized Snakemake pipeline was run on a desktop computer. This resulted in a
791 table containing 484,463 files, out of which 162,645 had a bz2 suffix. The individual file records were
792 compared with the list of accessions of files that were previously retrieved and sorted in our BIGSIdata
793 collection, and the volume of the source graphs on FTP calculated to be 16.7 TB.

794 **Comparison to Metagraph** [30]. The size of the phylogenetically compression BIGSIdata collection was
795 compared to the size of an analogical Metagraph index from the original paper [30] based on the statistics
796 in Table 1 and Supplementary Table 1 (SRA-Microbe collection): n=446,506 indexed datasets, 39.5 G
797 canonical *k*-mers (with the same *k*-mer size *k*=31), and the size of the annotated de Bruijn graph being
798 291 GB (graph 30 GB + annotations 261 GB). This index was constructed from the same data as in the
799 original BIGSI paper [16], but using a slightly different computational methodology. In consequence, the
800 index of Metagraph contained a by 4% lower number of distinct *k*-mers compared to BIGSIdata as
801 constructed in this paper, indicative of either lower diversity of the samples included or of their
802 additional cleaning. To compare the two compression approaches (MOF with bottom-up *k*-mer
803 propagation and XZ as a low-level tool vs. SRA-Microbe compressed using Metagraph), both applied to
804 the similar but different input data, we used the number of bits per distinct *k*-mer as the statistics to

compare, which was found to be 10.2 and 58.9, respectively, Therefore, the MOF compression was more efficient by an estimated factor of 5.78, but this number might be underestimated due to a different noise level. We note that phylogenetic compression could be directly embedded into Metagraph in the future, which may help to reduce the size of its index substantially.

Phylogenetic compression of the 661k assembly collection

Batches. Clusters were identified based on the species identified using Kraken 2 [48] + Bracken [49], as provided in the supplementary materials in ref [15] (the File1_full_krakenbracken.txt file, the V2 column), and further split into batches as displayed in **Supplementary Fig. 1**.

Phylogenetic compression using MOF-Compress. The individual batches of the collection were compressed using the MOF-Compress pipeline, compressive phylogenies computed using MashTree [26], left-to-right reordering of the assemblies, left-to-right re-ordering of simplitigs of the de Bruijn graphs, bottom-up *k*-mer propagation and simplitig computation by ProPhyle, and storing simplitigs and assemblies as text and FASTA file, respectively, followed by a compression by 'xz -9 -T1'. The resulting files were deposited on https://doi.org/10.5281/zenodo.4602622.

Calculations of the statistics. All the statistics used in the plots and tables were calculated based on the numbers obtained from MOF-Compress. Additionally, the total number of *k*-mers was calculated using JellyFish [56] (v2.2.10) by

jellyfish count --mer-len 31 --size 200G --threads 32 --output kmer_counting.jf --out-counter-len=1 --canonical

which resulted in 44,349,827,744 distinct *k*-mers (28,706,296,898 unique *k*-mers) for the 661k collection and in 35,524,194,027 distinct *k*-mers (22,904,412,202 unique *k*-mers) for the 661k-HQ collection. We note that the files uploaded to https://doi.org/10.5281/zenodo.4602622 are higher by approximately 0.2 GB (approx. 0.7% of the total size) compared to the value **Supplementary Table 3** as the Zenodo submission was done with an older version of the pipeline with slightly different trees.

Recompression using MBGC. Individual phylogenetically compressed batches from the previous step were converted to single FASTA files by '*tar -xOvf {input.xz}*' and then compressed using MBGC v1.2.1 with 8 threads and the maximum compression level (3) by *'mbgc -i {input.fa} -c 3 -t 8 {output.mbgc}'*. This resulted in files occupying in total 20,726,725,129 bytes, which were then uploaded to Zenodo (https://doi.org/10.5281/zenodo.6347064).

840
841 **Compression in the lexicographic order.** As data in ENA and other similar repositories have
842 identifiers assigned in the order in which they are uploaded, individual uploads typically proceed by
843 uploading entire projects, and these typically involve phylogenetically very close genomes; for instance,
844 genomes from a study investigating a hospital outbreak often occupy a range of accessions. As such,
845 lexicographically sorted genomes from ENA can be considered as a first approximation of phylogenetic
846 compression. To compare the compressibility of the 661k collection in the ENA accession order to the full
847 phylogenetic compression, we streamed the genomes from the main collection file provided on
848 http://ftp.ebi.ac.uk/pub/databases/ENA2018-bacteria-661k/661_assemblies.tar, decompressed them
849 on-the-fly, converted them to the one-line FASTA format, and compressed using XZ with 32 threads, all
850 by
851 *pv 661_assemblies.tar | tar -xOf - | gunzip -c | seqtk seq | xz -9 -T32 -v*
852
853 The computation, performed on a dedicated server, required 23.5 h of wall clock time and 757 CPU
854 hours, and the resulting file had 120,701,329,280 bytes. We note that compression using a single thread
855 was not possible in this case due to the size of the file; nevertheless, as the individual blocks used for XZ
856 parallelization were guaranteed to be a multiple of the dictionary size (that is 68.7 MB with the '-9'
857 preset), ensuring their sufficient size for the comparison to be correct.
858
859
860 ***Phylogenetic compression of the 661k/661k-HQ k-mer indexes***
861
862 **Phylogenetic compression of 661k-HQ COBS index**. We built a phylogenetically compressed
863 COBS index from the 661-HQ dataset using the MOF-COBS-build pipeline. In short, individual COBS
864 indexes were constructed per individual batches, with low-quality genomes removed, with the variant of
865 the index called ClaBS (Classical Bit-sliced index), analogical to the original BIGSI data structure [16]. In
866 this index, all columns (Bloom filters) have the same size and the genomes were provided in the left-to-
867 right phylogenetic order as illustrated in **Fig. 1c(i)**, after which every index was compressed using XZ,
868 resulting in 72.8 GB (1.06 GB when uncompressed, 14.5× reduction) (**Supplementary Table 5**). See
869 the section about MOF-COBS-build for more information. The resulting indexes were then used in MOF-
870 Search for the initial filtration of reference.
871
872 **Phylogenetic compression of 661k-HQ COBS index.** To evaluate the gain of phylogenetic
873 compression in the specific case of COBS indexes, we performed a series of additional experiments (see
874 their overview in **Supplementary Table 5**). In particular, we also created a phylogenetically
875 compressed index of the entire 661k collection, including the low-quality genomes, resulting in 110. GB

876 (2.46 TB when uncompressed, 22.5× reduction); here, the compression optically seems to be more
877 efficient, but the only reason is that contaminated genomes too much increase the size of the Bloom
878 filters, adding many additional rows that are predominantly composed of zeros.
879
880 **Comparisons to baselines.** To evaluate the improvement of phylogenetic compression for COBS, we
881 needed also to construct the compacted indexes (the default mode of the COBS program), with adaptive
882 adjustments of Bloom filter sizes through subindexes. This was more challenging as this required to work
883 simultaneously with the entire 661k dataset at the same time. For the entire 661k compact COBS index,
884 we used the official one, available online (http://ftp.ebi.ac.uk/pub/databases/ENA2018-bacteria-
885 661k/661k.cobs_compact, retrieved on 2022-09-08), as published with the original manuscript [15]. The
886 index had originally been constructed by COBS, version 7c030bb, using the "compact-construct"
887 subcommand with default options; i.e., without any batches and with adaptively sized Bloom filters (937
888 GB). In a similar fashion, we also constructed an analogical compact COBS index for the 661K-HQ
889 collection (893 GB). Both indexes were then compressed on a highly performant server in a combination
890 with XZ parallelization using 32 cores ("xz -9 -T32", resulting in 3.86× and 4.35× reductions,
891 respectively). However, we note such indexes are not suitable for personal computers due to both the
892 space requirements and the necessity to uncompress the index in its entirety at once. The resulting
893 comparison of phylogenetic compression of COBS indexes is provided in **Supplementary Table 5**.
894
895
896 ***MOF-Search pipeline for BLAST-like search across all pre-2019 bacteria from ENA***
897
898 **Overview of the pipeline.** MOF-Search (https://github.com/karel-brinda/mof-search) uses
899 phylogenetically compressed assemblies (661k) and phylogenetically compressed COBS indexes (661-
900 HQ) as described in the corresponding sections. Upon first execution, the pipeline downloads all the
901 input reference files from the Internet (29.2 GB of assemblies and 72.8 GB of COBS indexes, in total 102
902 GB). The search then consists of two phases – matching of queries against the *k*-mer indexes using COBS
903 [29], and then aligning the identified candidates using Minimap 2 [34]. MOF-Search is developed as a
904 Snakemake [57] pipeline using Bioconda [73], with the standard Snakemake resource management [57] to
905 control the assignments of CPU cores and limiting the RAM usage (up to a user-specified threshold).
906
907 **Matching.** Matching of queries is performed iteratively per individual batches. Individual
908 phylogenetically compressed COBS indexes are decompressed either on-the-fly (faster, but requires
909 additional memory for decompression), or on disk, and then they are queried using a modified version of
910 COBS (see below, v0.2.1 with a pre-specified *k*-mer threshold (the minimum required proportion of
911 matching *k*-mer). The output matches are either or stored on disk entirely, or only a user-defined number

912 of best hits (in terms of the number of matching *k*-mers) of interest (plus ties). To balance resources, the
913 number of threads used by COBS is adjusted based on the size of individual batches (bigger batches are
914 processed using more threads). Finally, the obtained results are aggregated across batches and for every
915 query only the pre-specified number of best matches (plus ties) is kept.
916
917 **Alignment.** Alignment of queries is performed also iteratively per individual batches. For every batch, a
918 dedicated Python script iterates over the phylogenetically compressed genomes and if at least one of the
919 queries was identified in the previous step as a potential hit for the current genome, its Minimap 2 [34] (v
920 2.24) index is built on on-the-fly and all the relevant queries aligned with user-specified parameters and
921 the output provided to the user.
922
923 **Modified COBS**. To enable the integration of COBS into MOF-Search, a new major version of COBS [29]
924 was created (v2, https://github.com/iqbal-lab-org/cobs), fixing multiple bugs, implementing a support
925 for OS X, integrating more tests, and supporting streamed loading of indexes into memory. The new
926 versions of COBS are provided in the form of Github releases (https://github.com/iqbal-lab-
927 org/cobs/releases), as well as pre-built packages on Bioconda [73].
928
929
930 ***Evaluating MOF-search***
931
932 **Overview of the benchmarking procedure.** The search using MOF-search was evaluated using
933 three datasets, representative of different query scenarios: a database of antibiotic resistance genes, a
934 database of plasmids, and an Oxford nanopore sequencing experiment. In all cases, the search
935 parameters were adjusted to the query type, including the number of top hits, the COBS *k*-mer threshold,
936 and the Minimap preset. The experiments were run on an iMac18,3, Quad-Core Intel CPU i7, 4.2 GHz
937 with 42.9 GB (40 GiB) RAM with 4 physical (8 logical) cores.
938
939 **Time measurement.** The wall clock and CPU time were measured using GNU time, and were
940 calculated as 'real' and 'usr+sys', respectively. The measurements were done for both search phases
941 separately (matching and alignment).
942
943 **Memory measurement.** We have not found any reliable way of measuring peak memory consumption
944 on macOS: GNU time was systematically providing incorrect values for our Snakemake pipeline, with
945 values several times lower compared to the expected ones, as did another method based on the psutil
946 Python library. Therefore, we performed additional measurements using the SLURM job manager on a
947 Linux cluster using jobs allocated with a configuration similar to our iMac computer. We found that for

948 'max_ram_gb' equal to 30 GB, the peak memory consumption was 26.2 GB. This discrepancy is an
949 expected behavior because the 'max_ram_gb' parameter defines an upper bound for the Snakemake
950 resource management [57], corresponding to the worst-case scenario of parallel job combinations.
951
952 **Resistance genes – ARGannot.** The resistance genes search as performed using the ARG-ANNOT
953 database [35], as distributed within the the SRST2 software toolkit [74] in the file
954 https://github.com/katholt/srst2/blob/master/data/ARGannot_r3.fasta (retrieved on 2022-07-24),
955 consisting of 1,856 genes/alleles. The search parameters were set to require at least 50% matching *k*-
956 mers, 1,000 best hits taken for every gene (although yielding always a much higher number of them due
957 to a high number of equally scoring matches), and the Minimap present for short reads.
958
959 **Plasmids – the EBI plasmid database.** The list of EBI plasmid was downloaded from
960 https://www.ebi.ac.uk/genomes/plasmid.details.txt (retrieved on 2022-04-03), and individual plasmids
961 then downloaded from the EBI ENA using curl and GNU parallel [69] (size characteristics provided in
962 **Supplementary Table 6**). The search parameters were set to at least 40% matching *k*-mers (as in ref
963 [16]), 1,000 best hits taken for every plasmid, and the Minimap present for mapping long highly divergent
964 sequences ('asm20').
965
966 **Oxford Nanopore reads – ERR9030361.** This experiment corresponds to 159k nanopore reads of
967 an isolate of Mycobacterium tuberculosis; the reads were downloaded from SRA NCBI and the search
968 parameters were adjusted for a sensitive identification of the nearest neighbors in the database, with at
969 least 40% matching *k*-mers, 10 best hits taken, and the Minimap present for mapping nanopore reads
970 ('map-ont').
971
972 **Comparison to BIGSI.** As we were unable to reproduce the original plasmid search experiment [16] with
973 BIGSI on our iMac computer (due to the size of files to transfer exceeding 1.43 TB), we resorted to the
974 values provided in the paper [16] . For a fair comparison, we focused our comparison on the the total CPU
975 time (sys+usr), and also verified that our parallelization is close from the maximal possible (680% out of
976 800% possible, based on the values in **Supplementary Table 6**), thus well characterizing also the real
977 time (we previously observed in other literature COBS being applied incorrectly, resulting in very
978 diverging values of '(sys+usr)/cores' and 'real' due to swapping, thus providing unmeaningful results of
979 real and biasing comparisons).
980