PanGraph: scalable bacterial pan-genome graph construction

Nicholas Noll,[1] Marco Molari,[2,3] and Richard A. Neher[2,3]
[1] *Kavli Institute for Theoretical Physics, University of California, Santa Barbara*
[2] *Swiss Institute of Bioinformatics, Basel, Switzerland*
[3] *Biozentrum, University of Basel, Basel, Switzerland*

The genomic diversity of microbes is commonly parameterized as population genetic polymorphisms relative to a reference genome of a well-characterized, but arbitrary, isolate. Reference genomes contain a fraction of the microbial *pangenome*, the set of genes observed within *all* isolates of a given species, and are thus blind to both the dynamics of the accessory genome, as well as variation within gene order and copy number. With the wide-spread usage of long-read sequencing, the number of high-quality, complete genome assemblies has increased dramatically. Traditional computational approaches towards whole-genome analysis either scale poorly, or treat genomes as dissociated bags of genes, and thus are not suited for this new era. Here, we present *PanGraph*, a Julia based library and command line interface for aligning whole genomes into a graph, wherein each genome is represented as an undirected path along vertices, which in turn, encapsulate homologous multiple sequence alignments. The resultant data structure succinctly summarizes population-level nucleotide and structural polymorphisms and can be exported into a several common formats for either downstream analysis or immediate visualization.

During evolution, microbial genomes change by both local modifications and large-scale alterations (Arnold *et al.*, 2021). Local mutations conserve sequence architecture and only change a few nucleotides by substitution, insertion or deletion. Conversely, large-scale mutations reorganize the sequence, and involve either the homologous recombination of large segments or wholesale gene loss or gain from the environment. The accumulation of such polymorphisms over time complicates comparative genomic analyses of present day isolates; microbial genomes are mosaics of homologous sequence, locally related by an asexual phylogenetic tree (Sakoparnig *et al.*, 2021).

Recent advances in long-read sequencing have enabled the low-cost assembly of isolated genomes at the quality of reference databases (Whibley *et al.*, 2021). The accumulation of many complete genomes promises to rapidly improve our ability to quantify the evolutionary dynamics that drive microbial diversity in natural populations. Unfortunately, comparative genomic analyses have not kept pace with the technological advance, opting either for reference-based approaches that only partially capture microbial diversity (Tettelin *et al.*, 2008) or to construct a *pangenome* that accurately captures nucleotide polymorphisms within genes but approximates structural polymorphisms as simple gene presence-absence relationships (Ding *et al.*, 2018; Page *et al.*, 2015) . This new era of pangenomics demands novel data structures to encapsulate the *complete* diversity of a given genomic sample set.

In recent years, efforts have focused on generalizing the *pangenome* framework of microbial diversity to graphical models (Eizenga *et al.*, 2020). At a high level, pangenome graphs generalize the reference sequence coordinate system conventionally used; microbial diversity is, instead, encapsulated by a graph-based data structure, with vertices or edges labelled by homologous DNA sequences, in which a walk recapitulates a subset of mosaic genomes. As such, pangenome graphs can be thought of as an analogous construct to the well-known assembly graph, in which whole genomes are individual "reads" of the underlying microbial structure (Myers, 2005).

While easy to conceptualize, the construction of pangenome graphs has proven computationally challenging. Colored generalizations of the de Bruijin graph-based assemblers have been successively used to build graphs from large sequence sets, however the underlying efficiency derives from a fixed kmer size which prevents modelling long-range homology (Iqbal *et al.*, 2012; Muggli *et al.*, 2017). An orthogonal approach has been to formulate the inference of the pangenome graph as a multiple genome alignment. However, current methods either scale poorly to large sets of genomes (Darling *et al.*, 2010), focus on comparisons across diverse sets of species across the tree of life at the cost of memory (Armstrong *et al.*, 2020), or utilize a reference-guided approach by partitioning genomes first into annotated genes (Colquhoun *et al.*, 2021; Gautreau *et al.*, 2020).

Here we present PanGraph, a Julia (Bezanson *et al.*, 2017) library and command line interface, designed to efficiently align large sets of closely related genomes into a pangenome graph on personal computers. The resulting graph both compresses the input sequence set and succinctly captures the population diversity at multiple scales ranging from nucleotide mutations and indels to structural polymorphisms driven by inversions, rearrangements, and gene gain/loss. The underlying graph data structure can be exported into numerous formats for downstream analysis and visualization in software such as Bandage (Wick *et al.*, 2015).

I. ALGORITHMS AND IMPLEMENTATION

PanGraph transforms an arbitrary set of genomes into a *graph* that simultaneously compresses the collection of sequences and exhaustively summarizes both the structural and nucleotide-level polymorphisms. The graph is composed of *pancontigs*, which represent linear multiple-sequence alignments of homologous sequence found within one or more input genomes. *Pancontigs* are connected by an edge if they are syntenic on at least one input sequence; individual sequences are then recapitulated by contiguous *paths* through the graph.

PanGraph's overarching strategy is to approximate multiple-genome alignment by iterative pairwise alignment, in the spirit of progressive alignment tools (Armstrong *et al.*, 2020; Darling *et al.*, 2010). A guide tree is utilized to linearize the problem's complexity by approximating multiple-sequence alignment as a quenched order of successive pairwise alignments (Feng and Doolittle, 1987). Pairwise graph alignment is performed by an all-to-all alignment of the *pancontigs* between both graphs.

A. Guide tree construction

The alignment guide tree is constructed subject to three design constraints: (i) sequences are aligned sequentially based upon similarity, (ii) the similarity computation scales sub-quadratically with the number of input sequences, and (iii) the resultant tree is balanced. To this end, we formulate the algorithm as a two step process. The initial guide tree is constructed by neighbor-joining (Saitou and Nei, 1987); the pairwise distance between sequences is approximated by the Jaccard distance between sequence minimizers (Roberts *et al.*, 2004). Computationally, each sequence can be sketched into its set of minimizers in linear time while the cardinality of all pairwise intersections can be computed by sorting the list of all minimizers to efficiently count overlaps. Hence, the pairwise distance matrix is estimated in log-linear time. The final guide tree is constructed as the balanced binary tree constrained to reproduce the topological ordering of leaves found initially. See Fig 1A for a graphical depiction of the guide tree.

B. Iterative graph alignment

The iterative pangraph construction is illustrated in Fig. 1. The pangraph is initialized as one subgraph per input genome, each representing its respective genome as a single *pancontig*. Pairs of subgraphs are aligned in a postorder traversal of the constructed guide tree. The identified homologous intervals of the pancontig are then merged, thereby creating shorter contigs that represent homologous sections of multiple input genomes, see Fig. 1B for a graphical deption. These steps of pairwise graph alignment and merging of homologous intervals are repeated until the root of the guide tree is reached.

a. Pairwise alignment. Full genome alignment between two closely related isolates is a well-studied problem with many high-quality tools available (Li, 2018; Marçais *et al.*, 2018). We chose to use *minimap2* as the core pairwise genome aligner for its proven speed, sensitivity, and easy-to-use, exposed library API (Li, 2018). However, we note that *PanGraph* is written to be modular; in the future, additional alignment kernels can be added with ease. The *minimap2* alignment kernel is included within a custom Julia wrapper, available at `github.com/nnoll/minimap2_jll.jl`. The kernel interface expects two lists of *pancontigs* to align as input and will output a list of putative overlapping alignments between both input lists.

b. Merging of homologous sequence. *Pancontigs* encapsulate linear multiple-sequence alignments which are modelled internally by a star phylogeny, i.e. are assumed to be well-described by a reference sequence augmented by independent SNPs and indels for each contained isolate. Importantly, during the all-to-all alignment phase, *pancontigs* are aligned based *solely* upon their consensus sequence. All putative homologous alignments found by *pancontig* alignment are ranked according to the pseudo-energy

$$E = -\ell + \alpha N_c + \beta N_m \tag{1}$$

where ℓ, N_c, and N_m denote the alignment length, number of *pancontigs* created by the merger, and number of polymorphisms per genome in the newly created pancontig respectively. See Fig 1B for a graphical definition for each term. Additionally, α and β are hyperparameters of the algorithm, respectively controlling the tradeoff between fragmentation of the graph and the maximum diversity within each block. Only alignments with negative pseudo-energy are performed.

At the graph level, the merger of two *pancontigs* defines a new *pancontig*, connected on both sides by edges to the neighboring *pancontigs* of both inputs, and thus locally collapses the two graphs under consideration. At the nucleotide level, the pairwise alignment of two *pancontigs* maps the reference of one onto the other; the merger of two *pancontigs* requires the application of the map onto the underlying multiple-sequence alignment. Once both sets of sequences are placed onto a common coordinate system, the resultant consensus sequence, and thus polymorphisms, are recomputed. This procedure can be viewed as an online multiple sequence alignment algorithm.

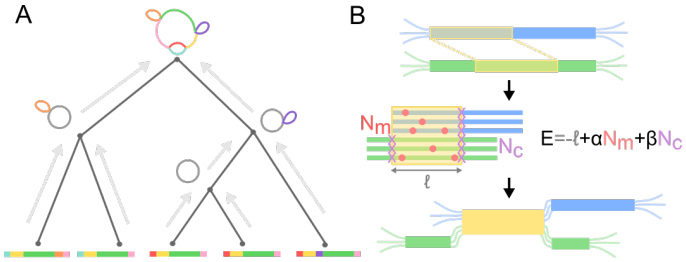

FIG. 1 **Overview of PanGraph algorithm**. (A) The alignment graph is constructed progressively by aligning graphs pairwise up a guide tree constructed from neighbor-joining the minimizer overlap between strains. (B) During pairwise alignment, *pancontigs* (blue and green) are merged by identifying homologous intervals (shown in yellow). If the underlying alignments are viewed compatible, i.e. the energy is less than 0, the pancontigs are merged.

The above procedure is repeated until no alignments with negative energy remain. Upon completion, transitive edges within the graph are removed by merging adjacent *pancontigs*.

C. Parallelism

PanGraph is designed with a message-passing architecture to enable scalable parallelism, as shown in Fig 1A for a cartoon example. Each internal node of the guide tree represents a job that performs a single pairwise graph alignment between its two children. The process will block until both of its children processes have completed and subsequently pass the result of their pairwise graph alignment up to the parent. All jobs run concurrently from the start of the algorithm; the Julia scheduler resolves the order of dependencies naturally (Bezanson *et al.*, 2017). As such, the number of parallel computations is automatically scaled to the number of available threads allocated by the user at the onset of the alignment.

D. Graph Export and Availability

The constructed pangenome graph can be exported in a variety of file formats for downstream analysis and visualization. In addition to a custom JSON schema, PanGraph can export the alignment as a GFA file, where each *pancontig* is represented as a segment and each genome as a path. This allows for visualization in software such as Bandage (Wick *et al.*, 2015). Lastly, we provide functionality to export as a conventional pangenome, however wherein *pancontigs* supplant putative gene clusters, that can be visualized directly by the PanX toolkit (Ding *et al.*, 2018).

PanGraph is published under an MIT license with source code, extensive documentation, examples, and instructions for installation available at `github.com/neherlab/pangraph`. Additionally, pre-built binaries are available as GitHub releases. All data and scripts used to validate PanGraph are available within the same repository.

II. VALIDATION AND PERFORMANCE

In order to assess the accuracy of the pangenome graph inferred by PanGraph, and to quantify its performance characteristics as a function of input size, we generated synthetic data. Specifically, we simulated populations of size $N = 100$ and sequence length $L = 50000$ utilizing a Wright-Fisher model (Hudson, 2002) evolved for $T = 50$ generations. In addition to nucleotide mutations that occur at rate μ per generation, we modelled inversions, deletions (each with independent rates set to $1e-2$ per generation), and horizontal sequence transfer that occurs with tunable rate r per generation per genome. The ancestral state for each sequence is tracked through each evolutionary event so that the true mosaic relatedness structure can easily be converted to a graph. The simulation framework is distributed within the PanGraph command line toolsuite for external use.

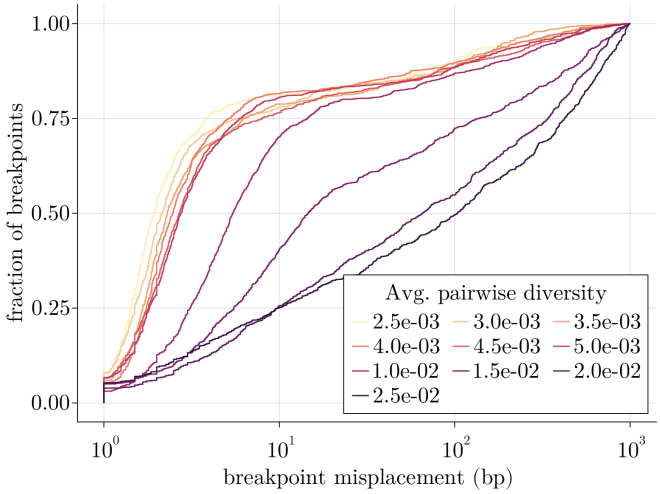

FIG. 2 **Accuracy against synthetic data**. Cumulative density functions for breakpoint accuracy comparing simulated and reconstructed pangenome graphs for various SNP rates, μ.

We quantified the accuracy of PanGraph algorithm by computing the distance, δ, of the inferred breakpoints relative to their known locus stored from the evolutionary simulation. Critically, we found that the accuracy was *independent* of the rate of horizontal gene transfer r and thus underlying graph complexity. The predominant determinant of accuracy was the sample diversity, shown in Fig. 3 which displays the distributions of δ for 100 independent simulations ran for each mutation rate μ. For

approximately clonal populations, $\sim 80\%$ of breakpoints were estimated to be within 10bp away from the "true" location. Conversely, for samples with average pairwise diversity above 2%, $\sim 80\%$ of breakpoints are estimated only within 100 bp. This limitation is primarily driven by lack of sensitivity in the pairwise alignment phase performed using *minimap2* (Li, 2018).

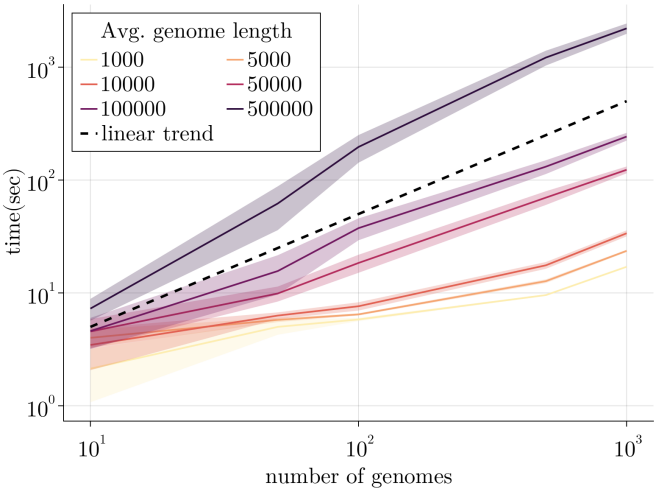

FIG. 3 **Algorithm performance**. PanGraph scales linearly with the number of input genomes. This is a direct result of the guide tree simplification. The solid line and ribbons display the mean and standard deviation respectively. All runs were performed on an AMD Ryzen 7 2700X utilizing 8 cores.

The algorithmic complexity was measured empirically by simulating populations of increasing size and average sequence length. The mean and standard deviation of results obtained from 25 iterations are shown in Fig. 3. Importantly, PanGraph's computational complexity grows *linearly* with the number of input genomes. We note that the PanGraph scales approximately log-linear with average sequence length, as expected from the underlying algorithmic complexity of Minimap2 (Li, 2018).

We additionally validated PanGraph on natural populations sampled from RefSeq (O'Leary *et al.*, 2016). Obviously the "true" breakpoints are not known *a priori* and thus the direct analog of Fig. 2 is not possible. However, we postulate that most *bone fide* breakpoints generated from evolutionary events will happen at or near gene boundaries; there is preliminary evidence for this assumption (Oliveira *et al.*, 2017). We compared the empirical distribution of distances to gene boundaries to a null distribution generated from the data, in which breakpoints are sampled uniformly from each genome, constrained to produce equivalent number of partitions as was measured by PanGraph. For all tested cases, we find our boundaries cluster closer to genes than would be expected by chance. The median distance from a Pangraph boundary to annotated gene ranges between 50-100 bp, depending on the species, while the random breakpoint is, on average, 600 bp off.

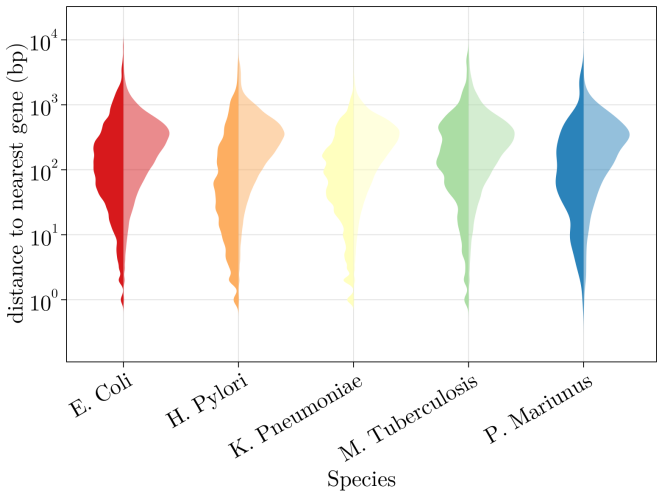

FIG. 4 **Pancontigs cluster towards gene boundaries**. The distribution of the nearest gene boundary for each pancontig breakpoint, shown for five species, with samples obtained from RefSeq (O'Leary *et al.*, 2016). (Left) Empirical distribution obtained over all genomes. (Right) Null distribution constructed by randomly sampling positions on each genome, constrained to conserve the number of breakpoints.

The computational time and maximum memory required to build each alignment is summarized in Table I.

TABLE I **Real-world Performance**. Elapsed time and maximum RAM usage needed to compute the pangraph for the following collections of RefSeq genomes. All runs were performed on an AMD Ryzen 7 2700X utilizing 8 cores.

Species	Sample size	Wall time (min)	Max memory (GB)
E. coli	307	426	11.7
H. pylori	85	102	1.2
K. pneumoniae	109	132	2.1
M. tuberculosis	51	91	0.4
P. marinus	10	12	0.1

III. DISCUSSION

While single nucleotide differences in the core genome are straightforward to analyze with existing tools, these analysis miss the great majority of genetic diversity. The ability to rapidly align large sets of closely related, fully assembled genomes is a critical tool for the investigation of the processes governing microbial evolution. We hope PanGraph will spur novel insight microbial diversity and the processes by which bacteria adapt and change.

Currently, PanGraph is limited to collections of

genomes that are closely related with typical sequence divergence smaller that than 3%. This limitation originates from the core tool used for pairwise alignment, `minimap2`, that was determined to lose sensitivity if sequences are sufficiently diverged under the chosen parameters. Homologous sequence that is more diverged than this cut-off will not be reliably merged by pangraph. This effectively limits the TMRCA of sequences within a block. More diverged sequence, for example originating from a cross-species transfer event, will thus be represented as alternative path. In addition to the intrinsic limit of `minimap2`, this cut-off can be tuned with the parameter α and can for example be used to restrict merging to very similar sequences and thus capture all recent horizontal events as divergent paths in the graph.

We emphasize that PanGraph was designed as a modular tool. In the future, we anticipate utilizing other pairwise "alignment" kernels in addition to minimap2 that will allow the user to trade-off speed for sensitivity.

ACKNOWLEDGEMENT

We are grateful to Boris Shraiman for stimulating discussions. This study was funded by the University of Basel and the NSF.

REFERENCES

Armstrong, J., G. Hickey, M. Diekhans, I. T. Fiddes, A. M. Novak, A. Deran, Q. Fang, D. Xie, S. Feng, J. Stiller, *et al.* (2020), Nature **587** (7833), 246.

Arnold, B. J., I. Huang, W. P. Hanage, *et al.* (2021), Nature Reviews Microbiology , 1.

Bezanson, J., A. Edelman, S. Karpinski, and V. B. Shah (2017), SIAM review **59** (1), 65.

Colquhoun, R. M., M. B. Hall, L. Lima, L. W. Roberts, K. M. Malone, M. Hunt, B. Letcher, J. Hawkey, S. George, L. Pankhurst, *et al.* (2021), Genome biology **22** (1), 1.

Darling, A. E., B. Mau, and N. T. Perna (2010), PloS one **5** (6), e11147.

Ding, W., F. Baumdicker, and R. A. Neher (2018), Nucleic acids research **46** (1), e5.

Eizenga, J. M., A. M. Novak, J. A. Sibbesen, S. Heumos, A. Ghaffaari, G. Hickey, X. Chang, J. D. Seaman, R. Rounthwaite, J. Ebler, *et al.* (2020), Annual review of genomics and human genetics **21**, 139.

Feng, D.-F., and R. F. Doolittle (1987), Journal of molecular evolution **25** (4), 351.

Gautreau, G., A. Bazin, M. Gachet, R. Planel, L. Burlot, M. Dubois, A. Perrin, C. Médigue, A. Calteau, S. Cruveiller, *et al.* (2020), PLoS computational biology **16** (3), e1007732.

Hudson, R. R. (2002), Bioinformatics **18** (2), 337.

Iqbal, Z., M. Caccamo, I. Turner, P. Flicek, and G. McVean (2012), Nature genetics **44** (2), 226.

Li, H. (2018), Bioinformatics **34** (18), 3094.

Marçais, G., A. L. Delcher, A. M. Phillippy, R. Coston, S. L. Salzberg, and A. Zimin (2018), PLoS computational biology **14** (1), e1005944.

Muggli, M. D., A. Bowe, N. R. Noyes, P. S. Morley, K. E. Belk, R. Raymond, T. Gagie, S. J. Puglisi, and C. Boucher (2017), Bioinformatics **33** (20), 3181.

Myers, E. W. (2005), Bioinformatics **21** (suppl_2), ii79.

O'Leary, N. A., M. W. Wright, J. R. Brister, S. Ciufo, D. Haddad, R. McVeigh, B. Rajput, B. Robbertse, B. Smith-White, D. Ako-Adjei, *et al.* (2016), Nucleic acids research **44** (D1), D733.

Oliveira, P. H., M. Touchon, J. Cury, and E. P. Rocha (2017), Nature communications **8** (1), 1.

Page, A. J., C. A. Cummins, M. Hunt, V. K. Wong, S. Reuter, M. T. Holden, M. Fookes, D. Falush, J. A. Keane, and J. Parkhill (2015), Bioinformatics **31** (22), 3691.

Roberts, M., W. Hayes, B. R. Hunt, S. M. Mount, and J. A. Yorke (2004), Bioinformatics **20** (18), 3363.

Saitou, N., and M. Nei (1987), Molecular biology and evolution **4** (4), 406.

Sakoparnig, T., C. Field, and E. van Nimwegen (2021), Elife **10**, e65366.

Tettelin, H., D. Riley, C. Cattuto, and D. Medini (2008), Current opinion in microbiology **11** (5), 472.

Whibley, A., J. L. Kelley, and S. R. Narum (2021), "The changing face of genome assemblies: Guidance on achieving high-quality reference genomes,".

Wick, R. R., M. B. Schultz, J. Zobel, and K. E. Holt (2015), Bioinformatics **31** (20), 3350.