

Comparative genomics with succinct colored de Bruijn graphs

Lucas P. Ramos^{1*†}, Felipe A. Louza^{2†} and Guilherme P. Telles^{1†}

^{1*}Instituto de Computação, UNICAMP, Campinas, SP, Brazil.

²Faculdade de Engenharia Elétrica, UFU, Uberlândia, MG, Brazil.

*Corresponding author(s). E-mail(s): lucaspr98@gmail.com;

Contributing authors: louza@ufu.br; gpt@ic.unicamp.br;

[†]These authors contributed equally to this work.

Abstract

DNA technologies have evolved significantly in the past years enabling the sequencing of a large number of genomes in a short time. Nevertheless, the underlying computational problem is hard, and many technical factors and limitations complicate obtaining the complete sequence of a genome. Many genomes are left in a draft state, in which each chromosome is represented by a set of sequences with partial information on their relative order. Recently, some approaches have been proposed to compare draft genomes by comparing paths in de Bruijn graphs, which are constructed by many practical genome assemblers. In this article we introduce **gcBB**, a method for comparing genomes represented as succinct colored de Bruijn graphs directly, without resorting to sequence alignments, by means of the entropy and expectation measures based on the Burrows-Wheeler Similarity Distribution. We also introduce an improved version of **gcBB**, called **multi-gcBB**, that improves the time performance considerably through the selection of different data structures. We have compared phylogenies of genomes obtained by other methods to those obtained with **gcBB**, achieving promising results.

Keywords: Succinct de Bruijn graphs, BOSS, BWSD, Genomic comparison, Phylogenetics

1 Introduction

Genome assembly is the task of reconstructing the sequence of nucleotides in molecules of DNA that follows the DNA sequencing process, in which a large amount of short

strings representing fragments of consecutive nucleotides in the target DNA molecule (reads) is first obtained. The reads cover each DNA nucleotide many times, varying across sequencing projects and may be as high as 200 times per nucleotide. The reads must then be assembled based on the overlaps among them. This is a hard computational problem, further complicated by the huge number of reads that may be obtained with the current DNA sequencing technologies, by the presence of repetitions in the target DNA, by sequencing errors and by other sources of ambiguities and technical limitations. Completely assembling a genome also requires intensive wet-lab work, then many genomes are left in a draft state after sequencing, that is, instead of a single string for each chromosome there is a set of strings (contigs) that may include information on their relative order (scaffolds) [31].

Several approaches have been proposed to assemble genomes. Some of the most used are based on different assembly graphs, such as overlap graphs [32], de Bruijn graphs [8], string graphs [38] and repeat graphs [13]. These assembly graphs can also be useful in gene discovery, structural variation analysis, hybrid assembly and other applications [36].

Comparative genomics aims at identifying similar and dissimilar regions among genomes [21]. Through comparison it is possible, for instance, to identify conserved regions across species that may be related to cellular processes, to identify regions involved in mutational events, to build phylogenies based on similarity among sequences, etc.

In Bioinformatics, the most widely employed means of calculating similarity between biological sequences is through alignments [12]. When used with evolution models for amino acids and nucleotides, alignments provide a similarity measure that reflects the evolutionary distance between molecules and is supported by statistics that are readily understood by practitioners [34]. Alignments also provide a natural layout for visualizations and visual data exploration [41].

On the other hand, the quadratic computational cost of the algorithms to calculate alignments between strings ($O(nm)$ time for two strings of lengths n and m) coupled with the huge amount of data currently available in public and private repositories push the need for faster alternatives such as heuristics, parallel algorithms and alternative distance measures, including alignment-free strategies [21]. When it comes to the comparison of multiple genomes, producing a multiple sequence alignment is even harder computationally, and the use of heuristics is widespread.

Similarity measures for strings in general, not only DNA or protein, may be computed in many ways, for example as distance among vector representation of strings, as statistics calculated on groups of symbol co-occurrences, as edit distance, as alignment score, as substring tiling and others [1, 37]. Compression-based measures also exist, such as the NCD [6], whose idea originates in the works on minimum algorithmic descriptions of strings. Similarity measures based on the Burrows-Wheeler transform (BWT) [4], as the eBWT-based distances [23, 24] and the Burrows-Wheeler Similarity Distribution (BWSD) [40], are particularly attractive because the BWT provides a self-index [26] that can be computed in linear time on the string length.

Many genome assemblers are based on the de Bruijn graph (*e.g.* [14, 16, 38]), that may be stored succinctly using the BOSS representation [3]. The BOSS representation

is based on the BWT and enables the assembly of larger sets of reads. Colors may be added to the edges of a de Bruijn graph, allowing the representation of a set of strings from distinct genomes on colored de Bruijn graphs. Recent approaches have been proposed for the comparison of genomes by the extraction of paths from their colored de Bruijn graphs [20, 28].

In this paper we introduce `gcBB`¹, a space-efficient algorithm for comparing genomes using succinct de Bruijn graphs and BWSD. Given a set of genomes, each one represented by a set of unmounted raw reads, a colored de Bruijn graph in the BOSS representation is built for all the genomes and BWSD-based measures are evaluated to assess the similarity between all genomes in the set. Our algorithm computes the colored de Bruijn graph for all genomes only once, avoiding a pairwise construction, and the BWSD-based measures are computed using compressed data structures as proposed in [18]. Our method showed promising results in experiments that compared the phylogenies for genomes of 12 *Drosophila* species built with `gcBB` and with the methods by Lyman *et al.* [20] and by Pevikov and Kolmogorov [28].

2 Definitions and notation

A string is the juxtaposition of symbols from an ordered alphabet Σ of size σ . Let S be a string of length n . We index its symbols from 1 to n . A substring of S is $S[i, j] = S[i] \dots S[j]$ with $1 \leq i \leq j \leq n$. Whenever $i > j$ then $S[i, j]$ denotes the empty string. Any substring $S[1, i]$ is referred to as a prefix of S and $S[i, n]$ is referred to as a suffix of S . The concatenation of string and symbols will be denoted by juxtaposition.

For clearer definitions we assume that the last symbol of a string S is the special end-marker symbol $\$$, that does not occur elsewhere in S and is the smallest symbol in Σ . This way, all suffixes of S are distinct.

The suffix array [22] of a string S of length n is the array SA_S containing the permutation of $\{1, \dots, n\}$ that gives the suffixes of S in lexicographic order, that is, $S[\text{SA}_S[1], n] < S[\text{SA}_S[2], n] < \dots < S[\text{SA}_S[n], n]$.

By $\text{lcp}(S_1, S_2)$ we denote the length of the longest common prefix of strings S_1 and S_2 . The LCP array for a string S of length n is the array of integers containing the lcp of consecutive suffixes in SA_S . Formally, $\text{LCP}_S[i] = \text{lcp}(S[\text{SA}_S[i], n], S[\text{SA}_S[i-1], n])$ for $1 < i \leq n$ and $\text{LCP}_S[1] = 0$. Figure 1 shows the suffix array and the LCP array for the string $S = \text{abracadabra}\$$.

Let $\mathcal{S} = \{S_1, S_2, \dots, S_d\}$ be a collection of d strings of lengths n_1, n_2, \dots, n_d . We define the concatenation of all strings in \mathcal{S} as $S^{\text{cat}} = S_1[1, n_1 - 1]\$_1 S_2[1, n_2 - 1]\$_2 \dots S_d[1, n_d - 1]\$_d$, that is, the end-marker symbol $\$$ of each string is replaced by a separator symbol $\$_i$ such that $\$_i < \$_j$ if $i < j$ and every $\$_i < \$$. The length of the concatenated string S^{cat} is $N = \sum_{i=1}^d n_i$. The suffix and LCP arrays for a collection \mathcal{S} correspond to the arrays $\text{SA}_{\mathcal{S}}[1, N]$ and $\text{LCP}_{\mathcal{S}}[1, N]$ computed for S^{cat} .

We define the context of a suffix $S^{\text{cat}}[i, N]$ as the substring $S^{\text{cat}}[i, j]$ such that $S^{\text{cat}}[j]$ is the leftmost occurrence of some $\$_k$ in $S^{\text{cat}}[i, N]$. The document array is an array of integers $\text{DA}_{\mathcal{S}}$ of length n that stores to which string each suffix in $\text{SA}_{\mathcal{S}}$ belongs. Formally, $\text{DA}_{\mathcal{S}}[i] = j$ if the context of $S^{\text{cat}}[\text{SA}_{\mathcal{S}}[i], N]$ ends with $\$_j$. The suffix, LCP

¹A preliminary version of this work appeared in [30].

i	SA_S	LCP_S	BWT_S	$S[SA_S[i], n]$
1	12	0	a	\$
2	11	0	r	a\$
3	8	1	d	abra\$
4	1	4	\$	abracadabra\$
5	4	1	r	acadabra\$
6	6	1	c	adabra\$
7	9	0	a	bra\$
8	2	3	a	bracadabra\$
9	5	0	a	cadabra\$
10	7	0	a	dabra\$
11	10	0	b	ra\$
12	3	2	b	racadabra\$

Fig. 1: Suffix array, LCP array and BWT for string $S = \text{abracadabra}\$$. The last column shows the suffixes of S in the order provided by the suffix array.

and document arrays for a collection \mathcal{S} can be computed in linear time using constant workspace [17].

When clear from the context, we drop the subscripts in SA_S , LCP_S , SA_S and DA_S . Without loss of generality, we assume that the alphabet Σ of a string is $\{1, \dots, \sigma\}$ or has been implicitly mapped onto $\{1, \dots, \sigma\}$.

2.1 BWT and BWS

The Burrows-Wheeler Transform (BWT) [4] of a string S is a reversible transformation that permutes its symbols such that the resulting string, denoted by BWT_S or simply by BWT when the context is clear, often allows better compression because equal symbols tend to be clustered. The BWT is the core of many indexing structures for text [21, 26, 27].

For $0 \leq i < n$, the i -th circular rotation (or conjugate or simply rotation) of a string S is the string $S[i+1, n]S[1, i]$. As $S[n] = \$$, its rotations are distinct. The BWT is the last column of a matrix \mathcal{M} having the sorted rotations of S as rows. In \mathcal{M} , the first column is called F and the last column is called L . Since $S[n] = \$$, sorting the rotations of S is equivalent to sorting the suffixes of S and then the BWT may be defined in terms of the suffix array as $BWT[i] = S[SA[i] - 1]$ if $SA[i] \neq 1$ or $BWT[i] = \$$ otherwise. The BWT for $S = \text{abracadabra}\$$ is shown in Figure 1. The BWT for a collection of strings \mathcal{S} may be obtained from the SA of S^{cat} as well [5].

The BWT of strings S_1 and S_2 may be used to compute similarity measures between them based on the observation that the amount of symbols of S_1 intermixed with symbols of S_2 in the BWT of S_1S_2 is related to the amount of substrings shared by S_1 and S_2 [23].

The Burrows-Wheeler similarity distribution (BWS) [40] between S_1 and S_2 , denoted by $BWS(S_1, S_2)$, is a probability mass function defined as follows. Given the BWT of $\mathcal{S} = \{S_1, S_2\}$, we define a bitvector α of size $n_1 + n_2$ such that $\alpha[p] = 0$ if $BWT[p] = \$_2$ or $BWT[p] \in S_1$ and $\alpha[p] = 1$ if $BWT[p] = \$_1$ or $BWT[p] \in S_2$. The

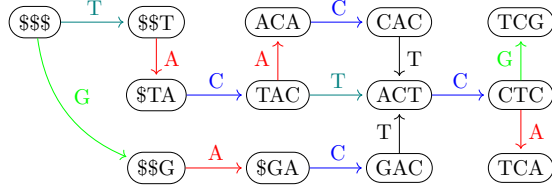


Fig. 2: De Bruijn graph for $\mathcal{S} = \{\$$$TACACT, $$$TACTCA, $$$GACTCG\}$ and $k = 3$.

bitvector α can be represented as a sequence of runs

$$r = 0^{k_1} 1^{k_2} 0^{k_3} 1^{k_4} \dots 0^{k_m} 1^{k_{m+1}}$$

where i^{k_j} means that i repeats k_j times and only k_1 and k_{m+1} may be zero. The largest possible value for k_j is $\max\{n_1, n_2\}$. We denote $\max\{k_j | i^{k_j} \in r\}$ by k_{\max} .

Let t_k be the number of occurrences of an exponent k in r . Let $s = t_1 + t_2 + \dots + t_{k_j} + \dots + t_{k_{\max}}$. The $BWSD(S_1, S_2)$ is the probability mass function

$$P\{k_j = k\} = t_k/s \text{ for } k = 1, 2, \dots, k_{\max}$$

Two distance measures were defined on the $BWSD$ of S_1 and S_2 [40].

Definition 1. The expectation distance is $D_M(S_1, S_2) = E(k_j) - 1$, where $E(k_j)$ is the expectation of $BWSD(S_1, S_2)$.

Definition 2. The entropy distance is $D_E(S_1, S_2) = \mathcal{H}(BWSD(S_1, S_2))$, where $\mathcal{H}(BWSD(S_1, S_2)) = -\sum_{k \geq 1, t_k \neq 0} (t_k/s) \log_2(t_k/s)$ is the Shannon entropy of $BWSD(S_1, S_2)$.

Note that if S_1 and S_2 are equal then $k_{\max} = 1$, $P\{k_j = 1\} = \frac{n_1+n_2}{n_1+n_2} = 1$, $D_M(S_1, S_2) = 0$ and $D_E(S_1, S_2) = 0$. Also, if α for $BWT(S_1, S_2)$ is equal to α for $BWT(S_2, S_1)$, then both have the same $BWSD$, and $D_E(S_1, S_2) = D_E(S_2, S_1)$ and $D_M(S_1, S_2) = D_M(S_2, S_1)$.

2.2 Succinct de Bruijn graphs

Let $\mathcal{S} = \{S_1, S_2, \dots, S_d\}$ be a collection of strings (reads of a genome). Assume that \mathcal{S} is modified by concatenating k symbols $\$$ at the beginning of each string in \mathcal{S} . We will refer to a string of length k as a k -mer.

A de Bruijn graph (of order k) for \mathcal{S} has one vertex for each distinct k -mer in a string of \mathcal{S} . We say that the k -mer related to a vertex u is its vertex label, denoted by \vec{u} . There is an edge labeled $v[k]$ from vertex u to vertex v if the substring $u[1]u[2] \dots u[k]v[k]$ occurs in a string of \mathcal{S} . For example, given $\mathcal{S} = \{\$$$TACACT, $$$TACTCA, $$$GACTCG\}$ and $k = 3$, Figure 2 illustrates the de Bruijn graph of order k for \mathcal{S} .

Notice that an edge from u to v corresponds to the existence of an overlap of length $k - 1$ between the suffix of \vec{u} and the prefix of \vec{v} and also that the concatenation of edge labels along a path of length k that arrives at a vertex v whose label does not

have a $\$$ will be \vec{v} . For example, in Figure 2, starting at node $\$$$$ and traversing the edges labelled T, A, C, A, C, T successively, we obtain the first string in the collection.

BOSS [3] is a succinct representation of the de Bruijn graph that enables efficient navigation across vertices and edges. Let n and m be respectively the number of vertices and edges of a de Bruijn graph G . Assume that the vertices v_1, v_2, \dots, v_n in G are sorted according to the co-lexicographic order of their labels, *i.e.*, the lexicographic order of the reverse of their labels, $\overleftarrow{v}_i = \overrightarrow{v}_i[k] \dots \overrightarrow{v}_i[1]$ for each vertex v_i .

We define *Node* as a conceptual matrix containing the co-lexicographically sorted set with the distinct k -mers in \mathcal{S} and with, for each vertex that has $t > 1$ outgoing edges with distinct labels, $t - 1$ additional copies of that label. Let m be the number of rows in *Node*.

For each vertex v_i , we define W_i as the sequence of symbols of the outgoing edges of v_i in lexicographic order. If v_i has no outgoing edges then $W_i = \$$.

The BOSS representation is composed by the following components:

1. The string $W[1, m] = W_1 W_2 \dots W_n$. Observe that $|W| = |\text{Node}|$ and $\text{Node}[i]$ denotes the vertex from which $W[i]$ leaves.
2. The bitvector $W^-[1, m]$ such that $W^-[i] = 0$ if there exists $j < i$ such that $W[j] = W[i]$ and the suffixes of length $k - 1$ of $\text{Node}[j]$ and of $\text{Node}[i]$ are identical, or $W^-[i] = 1$ otherwise.
3. The bitvector $\text{last}[1, m]$ such that $\text{last}[i] = 1$ if $i = n$ or $\text{Node}[i]$ is different from $\text{Node}[i + 1]$, or $\text{last}[i] = 0$ otherwise.
4. The counter array $C[1, \sigma]$ such that $C[c]$ stores the number of symbols smaller than c in the last column of the conceptual matrix *Node*.

For example, the succinct representation of the de Bruijn graph for $\mathcal{S} = \{\$$$TACACT, $$$TACTCA, $$$GACTCG\}$ is illustrated in Figure 3 augmented with the *Node* matrix and with edges of the de Bruijn graph to ease the understanding.

For DNA sequences, the alphabet is $\Sigma = \{A, T, C, G, N, \$\}$ with size $\sigma = 6$. Storing the string W requires $m \lceil \log_2 \sigma \rceil = 3m$ bits, the bitvectors W^- and last require $2m$ bits and the counter array C requires $\sigma \log_2 m = 6 \log_2 m$ bits. Therefore, the overall space to store the BOSS structure is $5m + 6 \log_2 m$ bits.

Egidi *et al.* [9] proposed an algorithm called eGap for computing the multi-string BWT and the LCP array in external memory. As an application the authors showed how to compute the BOSS representation with a sequential scan over the BWT and the LCP array built for collection \mathcal{S} with all strings reversed in $O(N)$ time.

The *colored de Bruijn graph* [11] generalizes the formulation of a de Bruijn graph for a set $\mathcal{S} = \{\mathcal{S}_1, \dots, \mathcal{S}_g\}$ of g string collections. In the colored de Bruijn graph the set of vertices includes all strings in \mathcal{S} and there may be parallel edges with the same label, but with distinct colors. Formally, the colored de Bruijn graph (of order k) for $\mathcal{S} = \{\mathcal{S}_1, \dots, \mathcal{S}_g\}$ has one vertex for each distinct k -mer in a string of \mathcal{S} . There is an edge labeled $v[k]$ and colored i from vertex u to vertex v if the substring $u[1]u[2] \dots u[k]v[k]$ occurs in a string of \mathcal{S}_i .

For example, for collections $\mathcal{S}_1 = \{\$$$TACACT, $$$TACTCA\}$ and $\mathcal{S}_2 = \{\$$$GACTCG\}$ and $k = 3$, Figure 4 shows the de Bruijn graphs for \mathcal{S}_1 and for \mathcal{S} , and the colored de Bruijn graph for $\{\mathcal{S}_1, \mathcal{S}_2\}$.

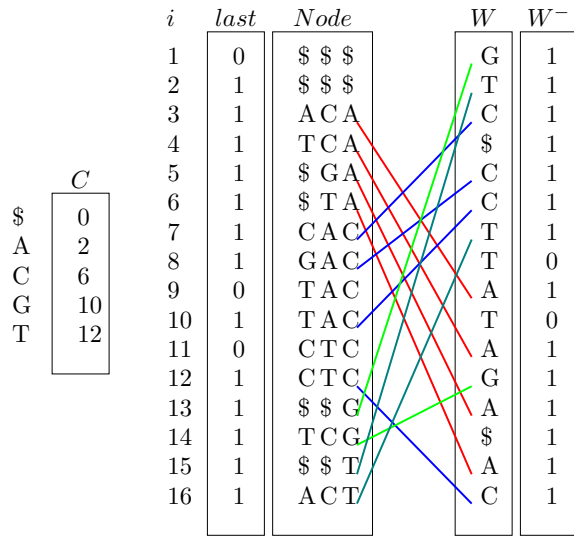


Fig. 3: (a) BOSS representation for $\mathcal{S} = \{\$ \$ \$ T A C A C T, \$ \$ \$ T A C T C A, \$ \$ \$ G A C T C G\}$ augmented with the $Node$ matrix and with edges of the de Bruijn graph; edges with the same color have the same symbol.

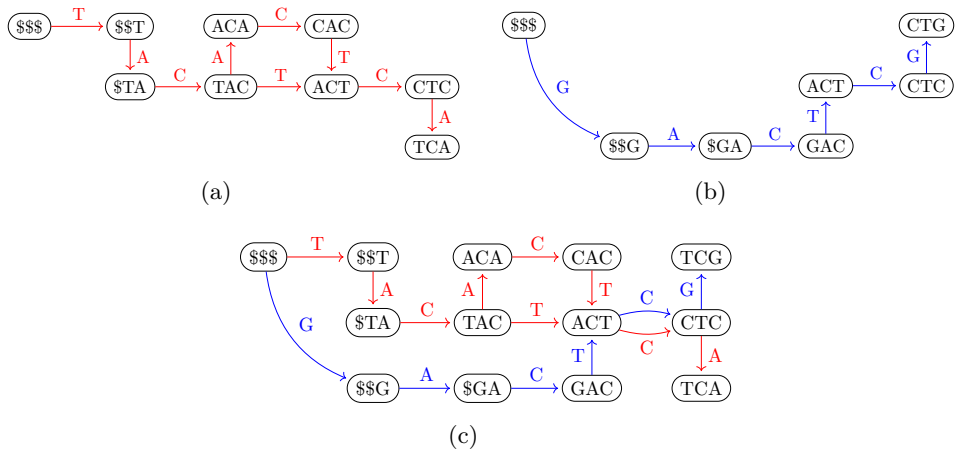


Fig. 4: (a) De Bruijn graph for $\mathcal{S}_1 = \{\$ \$ \$ T A C A C T, \$ \$ \$ T A C T C A\}$. (b) De Bruijn graph for $\mathcal{S}_2 = \{\$ \$ \$ G A C T C G\}$. (c) Colored de Bruijn graph for $\{\mathcal{S}_1, \mathcal{S}_2\}$, where red edges are from \mathcal{S}_1 and blue edges are from \mathcal{S}_2 . We remark that only the graph for $\{\mathcal{S}_1, \mathcal{S}_2\}$ has colored edges, \mathcal{S}_1 and \mathcal{S}_2 edges are colored for example purposes.

When $|\mathcal{S}| = 2$, a bitvector `colors` $[1, m]$ is added to obtain the BOSS representation of a colored de Bruijn graph, indicating from which genome each edge came from. We deal with the case $|\mathcal{S}| > 2$ in Section 3.1.

The number of reads that include a given nucleotide in the target DNA is referred to as coverage, and may be as high as 200 with current sequencing technologies. The coverage is used by genome assemblers to solve ambiguities during the reconstruction. Coverage is also directly related to the existence of repeated regions in the genome and to sequencing errors.

This information is handled using a `coverage`. The `coverage` $[1, m]$ array stores the number of times a $(k + 1)$ -mer represented by an edge occurs in its genome and can be computed from the LCP array during BOSS construction.

3 gcBB

The input for our algorithm, called `gcBB` (genome comparison using BOSS and BWSD), is a set of genomes (each one as a FASTQ file of reads) and a value for k . For each pair of genomes `gcBB` constructs the colored BOSS and computes the BWSD, producing two distance matrices, D_M and D_E , with the expectation and entropy distances among all pairs of genomes in the input set.

The intuition is that intermixed edges in the colored BOSS are related to shared nodes in their graphs and to similarities in the genomes.

The pseudo-code for `gcBB` is shown in Algorithm 1. `gcBB` has three phases, indicated in the pseudo-code and detailed below.

Phase 1:

First, `gcBB` constructs the BWT and the LCP array for each genome $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_g$ in external memory using `eGap` [9]. It also computes an auxiliary array with the length of each context, called `CL`. We remark that one could use any other tool to construct these data structures, for example [2, 19, 29].

For each pair of genomes \mathcal{S}_i and \mathcal{S}_j , the corresponding arrays are merged with `eGap` while the document array $DA_{i,j}$ is computed. Note that $DA_{i,j}$ can be stored in a bitvector, since we merge only pairs of genomes. The resulting arrays are written to external memory.

For genomes $\mathcal{S}_1 = \{\text{TACTCA}, \text{TACACT}\}$ and $\mathcal{S}_2 = \{\text{GACTCG}\}$, Figure 5 shows the output of `eGap` for each genome and the resulting merge.

Phase 2:

For each pair of genomes \mathcal{S}_i and \mathcal{S}_j , `gcBB` constructs the colored BOSS representation for \mathcal{S}_i and \mathcal{S}_j from the merged $BWT_{i,j}$ and $LCP_{i,j}$ array as described in [9]. The bitvector `colors` $[1, m]$ and the array `coverage` $[1, m]$ are computed, where m is the number of edges in the colored de Bruijn graph.

`gcBB` also computes two extra arrays, $LCS[1, m]$ and $KL[1, m]$. The LCS array contains the longest common suffix between consecutive k -mers in $Node$ and the KL array contains the length of each vertex label not including the $\$$ symbols. These arrays are easily obtained from $LCP_{i,j}$ and $CL_{i,j}$ arrays.

Algorithm 1: gcBB

Input: Collection $\mathcal{S} = \{\mathcal{S}_1, \dots, \mathcal{S}_g\}$ and k -mer length
Output: Matrices D_M and D_E of double precision numbers

```
// Phase 1
1 for each genome  $\mathcal{S}_i$  in  $\mathcal{S}$  do
2   | eGap( $\mathcal{S}_i$ ); // compute LCP $_i$ , BWT $_i$  and CL $_i$ 
3 end
4 for each pair of genomes  $\{\mathcal{S}_i, \mathcal{S}_j\}$  in  $\mathcal{S}$  do
5   | eGap( $\mathcal{S}_i, \mathcal{S}_j$ ); // merge LCP, BWT and CL and generate DA
6 end
7 double  $D_M[1..g][1..g] = 0.0$ ;
8 double  $D_E[1..g][1..g] = 0.0$ ;
9 for each pair of genomes  $\{\mathcal{S}_i, \mathcal{S}_j\}$  in  $\mathcal{S}$  do
   | // Phase 2
10  | bitvector colors initialized with 0;
11  | colors = colored_BOSS_construction(LCP $_{i,j}$ , BWT $_{i,j}$ , CL $_{i,j}$ , DA $_{i,j}$ ,  $k$ );
   | // Phase 3
12  | double expectation = 0.0;
13  | double entropy = 0.0;
14  | {expectation, entropy} = BWS computation(colors, LCP $_{i,j}$ , CL $_{i,j}$ ,  $k$ );
15  |  $D_M[i][j] = \text{expectation}$ ;
16  |  $D_E[i][j] = \text{entropy}$ ;
17 end
18 return  $D_M, D_E$ ;
```

Consider the merged arrays of genomes \mathcal{S}_1 and \mathcal{S}_2 obtained in Phase 1 and $k = 3$. The resulting colored BOSS representation is shown in Figure 6.

Phase 3:

The distances between each pair of genomes \mathcal{S}_i and \mathcal{S}_j are computed by evaluating the BWS on the `colors` bitvector, obtaining the corresponding entry in the expectation and entropy distance matrices.

Note that the colored BOSS representation contains the edges of every k' -mer from the merged genomes, for $1 \leq k' \leq k$. These edges are part of the BOSS representation and are needed by the navigation operations (see [3]). Since we are interested only in the k -mers for the comparisons, we filtered out all the edges of the colored BOSS where $\text{KL}[j] < k$, for $1 \leq j \leq m$, during the BWS computation.

From the colored BOSS shown in Figure 6, filtering edges representing k' -mers of size smaller than k and using the bitvector `colors` as the bitvector α of the BWS, we have $\alpha = \{0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1\}$, $r = 0^3 1^1 0^3 1^2 0^2 1^1$, $t_1 = 2, t_2 = 2, t_3 = 2$ and $s = 6$. Hence, the BWS($\mathcal{S}_1, \mathcal{S}_2$) is

$$P\{k_j = 1\} = \frac{2}{6}, P\{k_j = 2\} = \frac{2}{6}, P\{k_j = 3\} = \frac{2}{6}$$

i	BWT ₂	LCP ₁	CL ₁	context
1	T	0	1	$\$1$
2	T	0	1	$\$2$
3	C	0	5	ACAT $\$2$
4	$\$2$	2	7	ACTCAT $\$1$
5	C	1	3	AT $\$1$
6	C	2	3	AT $\$2$
7	T	0	6	CACAT $\$2$
8	A	2	4	CAT $\$2$
9	T	3	4	CAT $\$1$
10	A	1	6	CTCAT $\$1$
11	A	0	2	T $\$1$
12	A	1	2	T $\$2$
13	$\$1$	1	7	TCACAT $\$2$
14	C	3	5	TCAT $\$1$

(a)

i	BWT ₂	LCP ₂	CL ₂	context
1	G	0	1	$\$1$
2	C	0	3	AG $\$1$
3	T	0	4	CAG $\$1$
4	G	1	6	CTCAG $\$1$
5	A	0	2	G $\$1$
6	$\$1$	1	7	GCTCAG $\$1$
7	C	0	5	TCAG $\$1$

(b)

i	BWT _{1,2}	LCP _{1,2}	CL _{1,2}	DA _{1,2}	context
1	T	0	1	0	$\$1$
2	T	0	1	0	$\$2$
3	G	0	1	1	$\$3$
4	C	0	5	0	ACAT $\$2$
5	$\$3$	2	7	0	ACTCAT $\$1$
6	C	1	3	1	AG $\$3$
7	C	1	3	0	AT $\$1$
8	C	2	3	0	AT $\$2$
9	T	0	6	0	CACAT $\$2$
10	T	2	4	1	CAG $\$3$
11	A	2	4	0	CAT $\$1$
12	T	3	4	0	CAT $\$2$
13	G	1	6	1	CTCAG $\$3$
14	A	4	6	0	CTCAT $\$1$
15	A	0	2	1	G $\$3$
16	$\$2$	1	7	1	GCTCAG $\$3$
17	A	0	2	0	T $\$1$
18	A	1	2	0	T $\$2$
19	$\$1$	1	7	0	TCACAT $\$2$
20	C	3	5	1	TCAG $\$3$
21	C	3	5	0	TCAT $\$1$

(c)

Fig. 5: The BWT, LCP and CL arrays output by eGap for genomes (a) \mathcal{S}_1 and (b) \mathcal{S}_2 . (c) Merged BWT, LCP, CL arrays and DA for $\mathcal{S}_1\mathcal{S}_2$. The context column is not produced by eGap.

Computing the distances we have $D_M(\mathcal{S}_1, \mathcal{S}_2) = 1$ and $D_E(\mathcal{S}_1, \mathcal{S}_2) = 1.584$.

Coverage.

We can use the coverage information in the BWSO to weight the edges of the graph, aiming at improving the accuracy of the distance measures.

The same $(k + 1)$ -mer from distinct genomes can be detected in the colored BOSS using the LCS array and the `colors` bitvector. These repeated $(k + 1)$ -mers will appear in the α array with a 0 followed by a 1. Note that this happens only once independently of the number of times these $(k + 1)$ -mers occurred in both genomes. Whenever these repeated $(k + 1)$ -mers occurred many times in both genomes, their distance should be decreased.

For example, we added the string ACTC in sets \mathcal{S}_1 and \mathcal{S}_2 from the previous example. Let $\mathcal{S}_1' = \{\text{TACTCA}, \text{TACACT}, \text{ACTC}, \text{ACTC}, \text{ACTC}\}$ and $\mathcal{S}_2' = \{\text{GACTCG}, \text{ACTC}, \text{ACTC}\}$. In both genomes we have to increment the coverage information of the k -mers ACT with the outgoing edge C. The updated lines of the BOSS representation are shown in Figure 7.

i	$last$	$Node$	W	W^-	colors	coverage	LCS	KL
1	1	\$ ₁	T	1	0	2	0	0
2	1	\$ ₃	G	1	1	1	0	0
3	1	ACA	C	1	0	1	0	3
4	1	TCA	\$ ₃	1	0	1	2	3
5	1	\$ ₃ GA	C	1	1	1	1	2
6	1	\$ ₁ TA	C	1	0	2	1	2
8	1	CAC	T	1	0	1	0	3
9	1	GAC	T	0	1	1	2	3
10	0	TAC	A	1	0	1	2	3
11	1	TAC	T	0	0	1	3	3
12	0	CTC	A	1	0	1	1	3
13	1	CTC	G	1	1	1	4	3
14	1	\$ ₃ G	A	1	1	1	0	1
15	1	TCG	\$ ₂	1	1	1	1	3
16	1	\$ ₁ T	A	1	0	1	0	1
17	0	ACT	\$ ₁	1	0	1	1	3
18	0	ACT	C	1	0	1	3	3
19	1	ACT	C	0	1	1	3	3

Fig. 6: $\mathcal{S}_1\mathcal{S}_2$ merged colored BOSS representation with $k = 3$. Lines where KL values are colored red represent edges that will be filtered out in the BWSD computation.

i	$last$	$Node$	W	W^-	color	coverage	LCS	KL
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
18	0	ACT	C	1	0	4	3	3
19	1	ACT	C	0	1	3	3	3

Fig. 7: Lines with **coverage** incremented for the $(k + 1)$ -mer ACTC in the colored BOSS for $\{\mathcal{S}_1', \mathcal{S}_2'\}$.

Let $\alpha' = \{0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1\}$ be a bitvector equal to α from the previous example. The last 0 and 1 values from α' represent the $(k + 1)$ -mer ACTC from both genomes. We apply the coverage value to the positions of r' where these values occurred. That is, $r' = 0^31^10^31^20^{1+3}1^{1+2} = 0^31^10^31^20^41^3$. Finally, we expand r' in the positions of the equal $(k + 1)$ -mers while merging them, that is $r' = 0^31^10^31^20^10^11^10^11^10^11^10^10$. Then, we have $t_1 = 8$, $t_2 = 1$, $t_3 = 2$ and $s = 11$. And the $BWSD(\mathcal{S}_1', \mathcal{S}_2')$ is

$$P\{k_j = 1\} = \frac{9}{12}, P\{k_j = 2\} = \frac{1}{12}, P\{k_j = 3\} = \frac{2}{12}$$

Computing the distances we have $D_M(\mathcal{S}_1'\mathcal{S}_2') = 0.41666$ and $D_E(\mathcal{S}_1'\mathcal{S}_2') = 1.04085$. The effect of coverage on the similarity is analysed in our experiments.

Time and space analysis.

Let N_1 and N_2 be the sizes of two genomes. Phase 1 takes $O((N_1 + N_2)\text{maxlcp})$ time to construct and merge the BWT, LCP and CL in external memory with eGap, where maxlcp is the maximum in LCP.

Phase 2 takes $O(N_1 + N_2)$ time to construct the BOSS representation.

Let m be the number of edges in the colored BOSS. The space required for the BOSS representation is $5m + 6 \log_2 m$ bits, as shown in Section 2.2. The `colors` bitvector and the `coverage` array require extra m bits and $4m$ bytes respectively. For reads with less than 65K symbols both LCS and KL can be stored in arrays of short integers, that is, $2m$ bytes for each one. Therefore, the overall space required is $8m$ bytes plus $6m + 6 \log_2 m$ bits.

Phase 3 takes $O(m)$ time to compute the BWSD from the `colors` bitvector, LCS and KL arrays. The arrays r and t require $O(m)$ bytes.

3.1 An improved algorithm: multi-gcBB

In this section, we describe an extension to `gcBB`, called `multi-gcBB`, that computes the colored BOSS for all genomes $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_g$ only once, instead of constructing pairwise as above, and computes the distance matrices D_M and D_E using compressed data structures, as proposed in [18] for the BWSD. The pseudo-code for `multi-gcBB` is shown in Algorithm 2.

In Phase 1, `multi-gcBB` constructs the BWT and the arrays LCP and CL for each genome $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_g$ with eGap (as in the previous version). Then, all these arrays are merged only once, computing the document array DA for all genomes as well.

In Phase 2, `multi-gcBB` computes the colored BOSS for all genomes, modifying `colors[1, m]` to be an integer array and m the number of edges in the colored de Bruijn graph for $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_g$.

In Phase 3, `multi-gcBB` builds g bitvectors B_i , with $|B_i| = m$, where $B_i[j] = 1$ if `colors[j] = i` or $B_i[j] = 0$ otherwise. For each bitvector B_i , an $O(1)$ rank/select structure [25] is built. The algorithm then proceeds line by line on the matrix. To evaluate the distances among \mathcal{S}_i and $\mathcal{S}_{j>i}$, the algorithm selects the intervals over `colors` that contain consecutive occurrences of i . For each interval the algorithm counts the k_j occurrences of j , which corresponds to the existence of the run 1^{k_j} in the sequence of runs for \mathcal{S}_i and \mathcal{S}_j . The runs 0^{ℓ_j+1} are computed when ℓ_j consecutive intervals do not contain any occurrence of j .

Time and space analysis.

Let \mathcal{S} be a collection of g genomes and N be the total length of all genomes. Phase 1 takes time $O(N\text{maxlcp})$ time to construct and merges all arrays with eGap. Phase 2 takes $O(N)$ time to construct the BOSS representation. The `colors` array now requires $m \log_2 g$ bits.

Then, the overall space required for the BOSS representation with LCS and KL arrays is $5m + 6 \log_2 m + m \log_2 g$ bits plus $8m$ bytes.

Phase 3 takes $O(dN)$ time to compute the BWSD from the `colors` array, LCS and KL arrays. The bitvectors B_i with support to rank/select queries require $dN + o(dN)$ bits. The arrays r and t require $O(N)$ bytes.

Algorithm 2: mgcBB

Input: Collection \mathcal{S} and selected k
Output: Matrices D_M and D_E of double precision numbers
// Phase 1
1 **for** each genome S_i in \mathcal{S} **do**
2 | eGap(S_i); // compute LCP, BWT and CL
3 **end**
4 eGap_merge(\mathcal{S}); // merge LCP, BWT and CL and generate DA
5 double $D_M[1..d][1..d] = 0.0$;
6 double $D_E[1..d][1..d] = 0.0$;
// Phase 2
7 colored_boss_construction(LCP, BWT, CL, DA, k); // compute colors
// Phase 3
8 bwsd_all(colors, LCP, CL, D_M , D_E);
9 **return** D_M, D_E ;

4 Experiments

We evaluated gcBB by reconstructing the phylogeny of the 12 *Drosophila* species in Table 1, obtained from FlyBase [39]. The reads were obtained with a NextSeq 500 sequencer² and have 302 bp on average, except that reads of *D. grimshawi* were obtained with a MinION sequencer³ and have 6,520 bp on the average. The phylogenies in the sequel were drawn using iTOL [15].

Table 1: Information on the genomes of *Drosophilas*, that can be accessed through their Run (SRR) or BioSample (SAMN) accessions at <https://www.ncbi.nlm.nih.gov/genbank/>. The Bases column has the number of sequenced bases in Gbp. The Reference column has the size of the complete genome in Mb.

Organism	Run	BioSample	Bases (Gbp)	Reference (Mb)
<i>D. melanogaster</i>	SRR6702604	SAMN08511563	6.20	138.93
<i>D. ananassae</i>	SRR6425991	SAMN08272423	7.13	215.47
<i>D. simulans</i>	SRR6425999	SAMN08272428	9.22	131.66
<i>D. virilis</i>	SRR6426000	SAMN08272429	11.16	189.44
<i>D. willistoni</i>	SRR6426003	SAMN08272432	11.66	246.98
<i>D. pseudoobscura</i>	SRR6426001	SAMN08272435	12.28	163.29
<i>D. mojavensis</i>	SRR6425997	SAMN08272426	12.45	163.17
<i>D. yakuba</i>	SRR6426004	SAMN08272438	12.78	147.90
<i>D. persimilis</i>	SRR6425998	SAMN08272433	13.32	195.51
<i>D. erecta</i>	SRR6425990	SAMN08272424	14.01	146.54
<i>D. sechellia</i>	SRR6426002	SAMN08272427	14.44	154.19
<i>D. grimshawi</i>	SRR13070661	SAMN16729613	14.50	191.38

²<https://www.illumina.com/systems/sequencing-platforms/nextseq.html>

³<https://nanoporetech.com/products/minion>

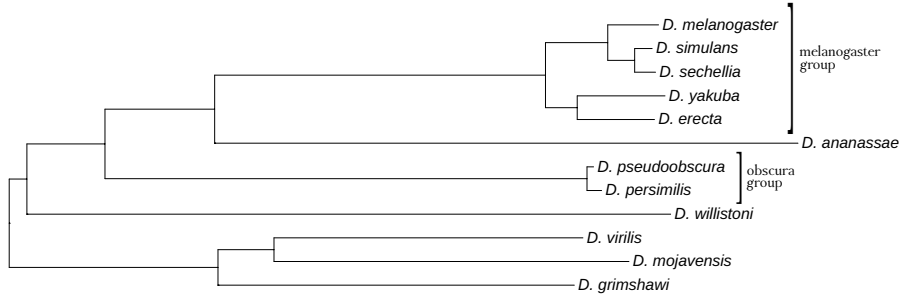


Fig. 8: Drosophila phylogeny, after [7].

Table 2: Construction information on data structures for the Drosophilas genomes.

Organism	BWT	LCP	CL	Running time	LCP avg	LCP max
<i>D. melanogaster</i>	5.9 GB	12 GB	12 GB	17.30h	61.54	302
<i>D. ananassae</i>	6.7 GB	14 GB	14 GB	18.37h	55.56	302
<i>D. simulans</i>	8.7 GB	18 GB	18 GB	24.18h	58.96	302
<i>D. virilis</i>	11 GB	21 GB	21 GB	28.45h	55.06	302
<i>D. willistoni</i>	11 GB	22 GB	22 GB	37.31h	57.79	302
<i>D. pseudoobscura</i>	12 GB	23 GB	23 GB	40.07h	58.72	302
<i>D. mojavensis</i>	12 GB	24 GB	24 GB	32.65h	58.09	302
<i>D. yakuba</i>	12 GB	24 GB	24 GB	42.42h	59.98	302
<i>D. persimilis</i>	13 GB	25 GB	25 GB	35.30h	58.76	302
<i>D. erecta</i>	14 GB	27 GB	27 GB	42.92h	60.47	302
<i>D. sechellia</i>	14 GB	27 GB	27 GB	37.88h	61.25	302
<i>D. grimshawi</i>	14 GB	27 GB	27 GB	57.40h	43.07	2648

Figure 8 shows a phylogeny for the 12 Drosophila genomes [7, 10] built using Neighbor-Joining on distances inferred on alignments among families of genes. This phylogeny was as reference, that is, we analysed if the distances computed by gcBB lead to a phylogeny that agrees with this phylogeny.

Our algorithms were implemented in C and compiled with gcc version 4.9.2. Our prototype implementation can be accessed at <https://github.com/lucaspr98/gcBB>. As previously mentioned, we used eGap [9] to construct and merge the data structures during Phase 1. The experiments were conducted on a system with Debian GNU/Linux 4.9.2 64 bits on Intel Xeon E5-2630 v3 20M Cache 2.40 GHz processors, 378 GB of RAM and 13 TB SAS storage. Our experiments were limited to 48 GB of RAM.

4.1 Running time

The times to run eGap to compute the BWT and the LCP and CL arrays for each genome $S_i \in \mathcal{S}$ in Phase 1 are shown in Table 2. The longest running time was approximately 57 hours, with the resulting arrays taking about 68 GB of space on disk. The sizes of the BWT, the sizes of the LCP and CL arrays, and the average and maximum LCP values are also shown in Table 2. Both LCP and CL arrays use 2 bytes per entry.

The overall time to run eGap to merge the computed data structures, for all pairs of genomes $\{S_i, S_j\} \in \mathcal{S}$, with the pairwise approach (Algorithm 1) was approximately 154 days, whereas the overall time to run eGap to merge the data structures of all genomes in \mathcal{S} at once with the all-vs-all approach (Algorithm 2) was approximately 20 days. The size of the merged files was approximately the sum of the sizes of the input files. The document array file has the same size of the merged BWT file, since both store each value using one byte.

In Phase 2, the overall running time to build all pairs of colored BOSS data structures with the pairwise approach was approximately 57 hours, while the average time to build the colored BOSS for a pair $\{S_i, S_j\}$ was about 52 minutes. The time to build the colored BOSS for all genomes at once with the all-vs-all approach was approximately 5.5 hours.

In Phase 3, the overall running time to compute all pairs of BWSD distance matrices entries with the pairwise approach was approximately 1.7 hours. The average time was about 2 minutes for each pair. The time to compute the BWSD distance matrices at once with the all-vs-all approach was approximately 10.8 hours.

Therefore, we spent approximately 156.4 days with gcBB against 20.6 days using multi-gcBB. The running times for Phases 1 and 2 were obtained with coverage information and $k = 31$.

4.2 Phylogenetic trees

We ran gcBB for $k = 15, 31$ and 63 , producing entropy and expectation BWSD distance matrices for the 12 *Drosophilas*, with and without coverage information.

We used the Neighbor-Joining [35] on the distance matrices to reconstruct the phylogenies. Following the construction for the reference phylogeny, we executed Neighbor-Joining down to $n = 2$ to root the phylogeny. We used the Robinson-Foulds [33] distance to compare our phylogenies with the reference phylogeny.

Let T be a phylogenetic tree with n vertices labeled by $U = \{1, 2, \dots, n\}$. If an edge is removed from T then it induces a bipartition of U . If every edge in the set E of edges of T is removed in turn, a set of induced bipartitions $T(E)$ is defined. For a pair of phylogenetic trees T_1 and T_2 with the same set of labelled leaves, the Robinson-Foulds distance between T_1 and T_2 is the size of the symmetric difference of $T_1(E_1)$ and $T_2(E_2)$; it is a metric whose values vary from 0 to $2n - 6$.

Figures 9 and 10 show the phylogenies for $k = 15$ with and without coverage information, respectively. The pair *D. pseudoobscura* and *D. persimilis* from the *obscura* group agrees with the reference phylogeny. With coverage information there is a clear separation between the *melanogaster* group and the outer groups. However, in general, the placement of the other genomes disagree with the reference phylogeny.

Figures 11 and 12 show the phylogenies for $k = 31$ with and without coverage information, respectively. There is one inconsistency involving *D. grimshawi* and *D. virilis*, which were swapped in our phylogenies, but are in the same subtree. Nonetheless, the high level groups division agrees with the reference phylogeny.

Figures 13 and 14 show the phylogenies for $k = 63$ with and without coverage information, respectively. They are very similar to those resulting with $k = 31$.

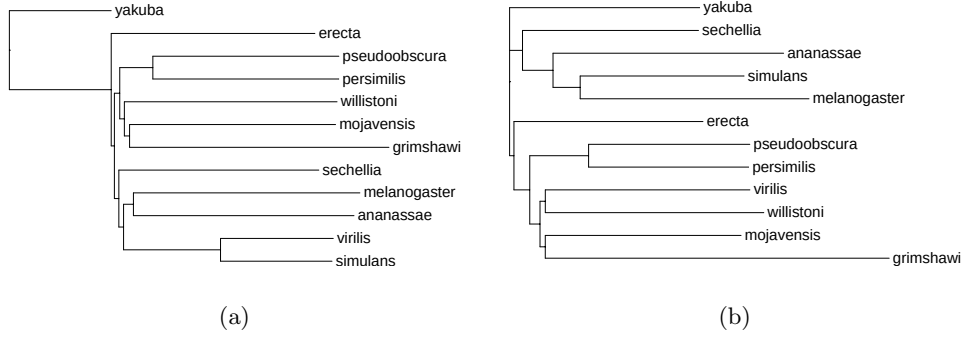


Fig. 9: gcBB phylogenies with $k = 15$, (a) using entropy, (b) using expectation.

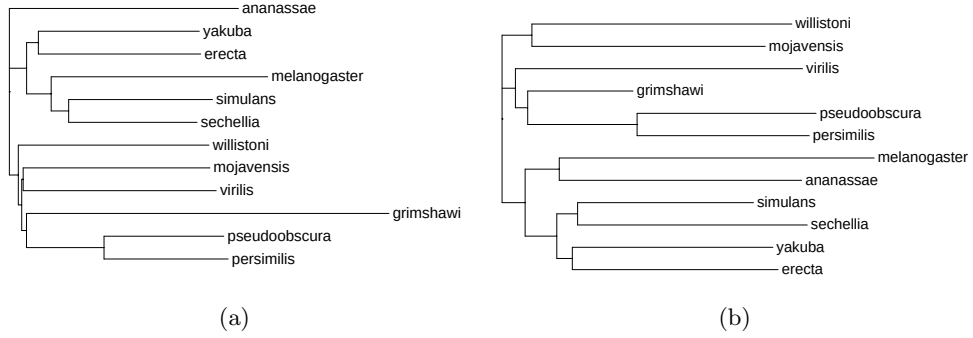


Fig. 10: gcBB phylogenies with $k = 15$ and coverage information, (a) using entropy, (b) using expectation.

Table 3: Robinson-Foulds distances computed between phylogenies by gcBB and the reference phylogeny of *Drosophila* genomes in Table 1. The symbol c indicates the phylogenies constructed by gcBB using coverage information.

	15	15c	31	31c	63	63c
Entropy	7	2	2	1	2	1
Expectation	6	5	2	3	2	2

Table 3 shows the Robinson-Foulds distance evaluated between the phylogenies by gcBB and the reference phylogeny. The phylogenies by gcBB which are closer to the reference were constructed using $k = 31$ and $k = 63$, with coverage information and entropy distance.

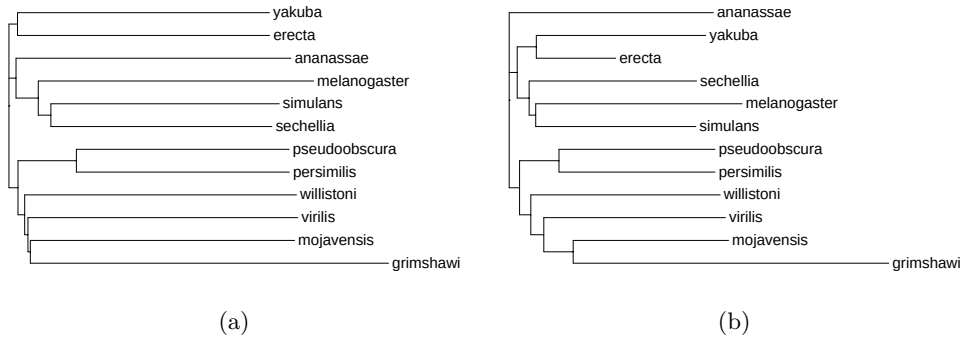


Fig. 11: gcBB phylogenies with $k = 31$, (a) using entropy, (b) using expectation.

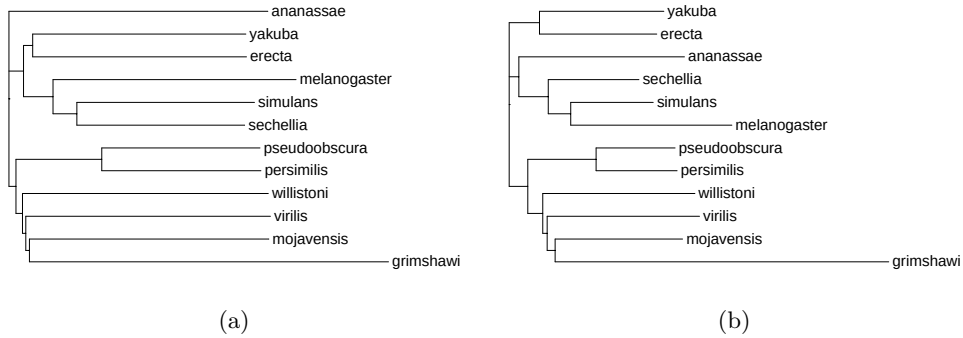


Fig. 12: gcBB phylogenies with $k = 31$ and coverage information, (a) using entropy, (b) using expectation.

Effect of data size

In order to evaluate the effect of read sizes in the resulting phylogenies, we considered sequencing data from an Illumina HiSeq 2000 for *D. grimshawi*. The information on this genome, the running time taken by eGap to construct the data structures and their sizes are shown in Table 4.

Table 4: Information on the genome of *D. grimshawi*, that can be accessed through its Run (SRR) or BioSample (SAMN) accessions at <https://www.ncbi.nlm.nih.gov/genbank/>. The Bases column has the number of sequenced bases in Gbp. The Reference column has the size of the complete genome in Mb. The sizes of data structures in Gb are shown in columns BWT, LCP and CL, and the average LCP is shown in column LCP avg.

Organism	Run	BioSample	Bases	Reference	BWT	LCP	CL	LCP avg
<i>D. grimshawi</i>	7642855	09764638	1.80	191.38	1.80	3.5	3.5	28.74

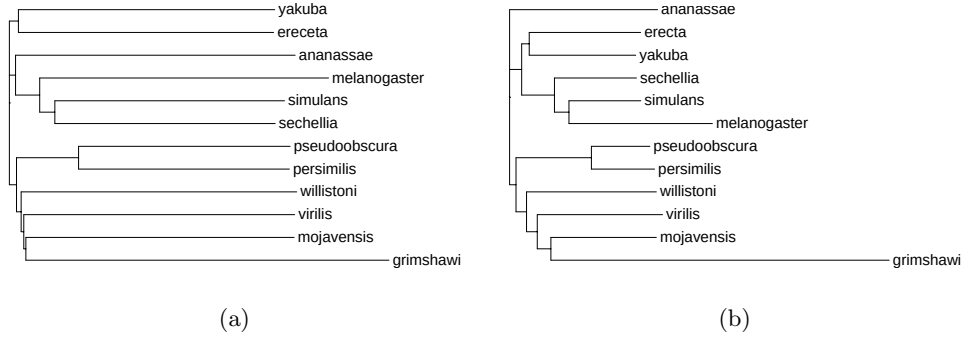


Fig. 13: gcBB phylogenies with $k = 63$, (a) using entropy, (b) using expectation.

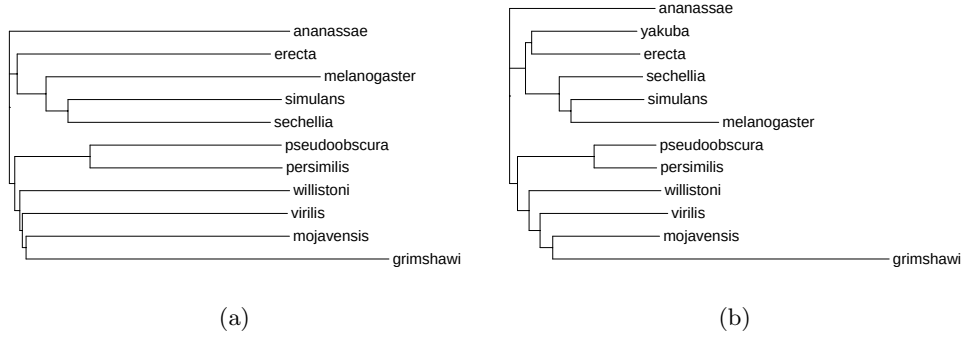


Fig. 14: gcBB phylogenies with $k = 63$ and coverage information, (a) using entropy, (b) using expectation.

Table 5: Robinson-Foulds distance computed between phylogenies with *D. grimshawi* from another experiment and the reference phylogeny. The symbol c represents the phylogenies constructed using coverage information.

	15	15c	31	31c	63	63c
Entropy	7	5	5	5	5	5
Expectation	6	7	6	5	6	6

We executed gcBB using the same parameters and values of k . The best phylogeny was obtained with $k = 15$ using coverage information. It is shown in Figure 15. By computing the Robinson-Foulds distance between these phylogenies and the reference phylogeny we obtained the values in Table 5.

We believe that significantly different amounts of sequenced bases impairs gcBB in its current form. In this experiment *D. grimshawi* has 1.8 Gbp, while *D. sechellia*

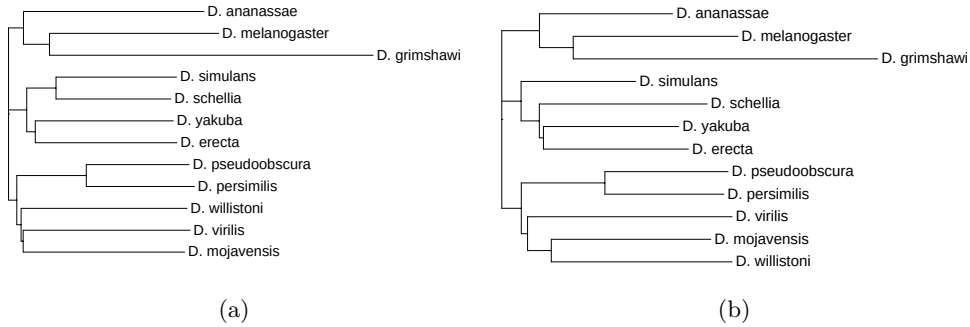


Fig. 15: gcBB phylogenies for *Drosophila*s with the alternative *D. grimshawi* with $k = 15$ and coverage information, (a) using entropy, (b) using expectation.

and *D. simulans* have more than 14 Gbp. When constructing the colored de Bruijn graph for *D. grimshawi* and *D. sechellia* there will be much more edges from *D. sechellia* than from *D. grimshawi*, and the similarity between these genomes tends to be smaller than it should be. Moreover, when constructing the colored de Bruijn graph for *D. grimshawi* and *D. melanogaster* there will also be much more edges from *D. melanogaster*. The difference between the amount of bases in *D. melanogaster* and *D. grimshawi* is around 4 Gbp, while from *D. sechellia* to *D. grimshawi* is around 12 Gbp.

These results suggest that our algorithms produce reasonable phylogenies when k is closer to the average LCP of the reads. Also, the coverage information reduced the Robinson-Foulds distance to the reference in most cases. Finally, the fact that all reads in the dataset were obtained using similar sequencing protocols and, on average, have a similar number of sequenced bases may have helped obtaining favorable results.

5 Conclusions

In this work we introduced a new method to compare genomes prior to assembly using space-efficient data structures implemented as gcBB and multi-gcBB, algorithms to compare sets of reads of genomes using the BOSS representation and to compute the similarity measures based on the BWS.

We evaluated our algorithms reconstructing the phylogeny of 12 *Drosophila* genomes. We used Neighbor-Joining over the distance matrices output by gcBB to reconstruct phylogenetic trees. Then we computed the Robinson-Foulds distance between the phylogenies by gcBB and a reference phylogeny. One issue when working with the de Bruijn graph is setting the value of k . We observed that values over the average LCP of the genomes lead to reasonable results. We observed better results using the entropy measure and coverage information in the BWS computation.

Future research may investigate different strategies for dealing with coverage information, as the experiments indicate a positive contribution of coverage to the resulting phylogenies. The quality of sequenced bases may also be investigated in future work as a means to improve the method.

Acknowledgements.

The authors thank Prof. Marinella Sciortino for helpful discussions and thank Prof. Nalvo Almeida for granting access to the computer used in the experiments.

Funding.

L.P.R. acknowledges that this study was financed by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil, Financing Code 001. F.A.L. acknowledges the financial support from CNPq (grants 408314/2023-0 and 406418/2021-7) and FAPEMIG (grant APQ-01217-22). G.P.T. acknowledges the financial support of Brazilian agencies CNPq (grants 408314/2023-0 and 317249/2021-5) and CAPES.

References

- [1] Baeza-Yates R, Ribeiro-Neto B, et al (1999) Modern information retrieval, vol 463. ACM press New York
- [2] Bonizzoni P, Vedova GD, Pirola Y, et al (2019) Multithread multistring Burrows-Wheeler Transform and longest common prefix array. *J Comput Biol* 26(9):948–961
- [3] Bowe A, Onodera T, Sadakane K, et al (2012) Succinct de Bruijn graphs. In: *Proc. WABI 2017*, vol 7534. Springer, pp 225–235
- [4] Burrows M, Wheeler DJ (1994) A block-sorting lossless data compression algorithm. Tech. Rep. 124, Systems Research Center
- [5] Cenzato D, Lipták Z (2022) A theoretical and experimental analysis of BWT variants for string collections. In: *33rd Annual Symposium on Combinatorial Pattern Matching*, pp 25:1–25:18
- [6] Cilibrasi R, Vitanyi PMB (2005) Clustering by compression. *IEEE Transactions on Information Theory* 51(4):1523–1545
- [7] Clark AG, Pachter L (2007) Evolution of genes and genomes on the drosophila phylogeny. *Nature* 450(7167):203–218
- [8] De Bruijn NG (1946) A combinatorial problem. In: *Proc. Koninklijke Nederlandse Academie van Wetenschappen*, pp 758–764
- [9] Egidi L, Louza FA, Manzini G, et al (2019) External memory BWT and LCP computation for sequence collections with applications. *Algorithms for Molecular Biology* 14(1):1–15
- [10] Hahn MW, Han MV, Han SG (2007) Gene family evolution across 12 drosophila genomes. *PLoS Genet* 3(11):e197

- [11] Iqbal Z, Caccamo M, Turner I, et al (2012) De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nature genetics* 44(2):226–232
- [12] Kim J, Ji M, Yi G (2020) A review on sequence alignment algorithms for short reads based on next-generation sequencing. *IEEE Access* 8:189811–189822
- [13] Kolmogorov M, Bickhart DM, Behsaz B, et al (2020) metaFlye: scalable long-read metagenome assembly using repeat graphs. *Nature Methods* 17(11):1103–1110
- [14] Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. *Nature Methods* 9(4):357
- [15] Letunic I, Bork P (2021) Interactive Tree Of Life (iTOL) v5: an online tool for phylogenetic tree display and annotation. *Nucleic Acids Research* 49(W1):W293–W296
- [16] Li H, Durbin R (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* 25(14):1754–1760
- [17] Louza FA, Gog S, Telles GP (2017) Inducing enhanced suffix arrays for string collections. *Theor Comput Sci* 678:22–39
- [18] Louza FA, Telles GP, Gog S, et al (2019) Algorithms to compute the Burrows-Wheeler similarity distribution. *Theor Comput Sci* 782:145–156
- [19] Louza FA, Telles GP, Gog S, et al (2020) gsufsort: constructing suffix arrays, LCP arrays and bwts for string collections. *Algorithms Mol Biol* 15(1):18
- [20] Lyman CA, Fujimoto MS, Suvorov A, et al (2017) Whole genome phylogenetic tree reconstruction using colored de Bruijn graphs. In: *IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE)*, IEEE, pp 260–265
- [21] Mäkinen V, Belazzougui D, Cunial F, et al (2015) *Genome-Scale Algorithm Design*. Cambridge University Press
- [22] Manber U, Myers G (1993) Suffix arrays: A new method for on-line string searches. *SIAM Journal on Computing* 22(5):935–948
- [23] Mantaci S, Restivo A, Rosone G, et al (2007) An extension of the Burrows-Wheeler transform. *Theoretical Computer Science* 387(3):298–312
- [24] Mantaci S, Restivo A, Sciortino M (2008) Distance measures for biological sequences: some recent approaches. *International Journal of Approximate Reasoning* 47(1):109–124
- [25] Munro JI (1996) Tables. In: *16th Conference on Foundations of Software Technology and Theoretical Computer Science*, pp 37–42

- [26] Navarro G (2016) Compact data structures: a practical approach. Cambridge University Press
- [27] Ohlebusch E (2013) Bioinformatics algorithms. Enno Ohlebusch
- [28] Polevikov E, Kolmogorov M (2019) Synteny paths for assembly graphs comparison. In: 19th International Workshop on Algorithms in Bioinformatics (WABI 2019), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik
- [29] Prezza N, Rosone G (2021) Space-efficient construction of compressed suffix trees. *Theor Comput Sci* 852:138–156
- [30] Ramos LP, Louza FA, Telles GP (2022) Genome comparison on succinct colored de bruijn graphs. In: Proc. SPIRE 2022, pp 165–177
- [31] Rice ES, Green RE (2019) New approaches for genome assembly and scaffolding. *Annual Review of Animal Biosciences* 7(1):17–40
- [32] Rizzi R, Beretta S, Patterson M, et al (2019) Overlap graphs and de Bruijn graphs: data structures for de novo genome assembly in the big data era. *Quantitative Biology* 7(4):278–292
- [33] Robinson DF, Foulds LR (1981) Comparison of phylogenetic trees. *Mathematical Biosciences* 53(1-2):131–147
- [34] Rosenberg MS (2005) Evolutionary distance estimation and fidelity of pair wise sequence alignment. *BMC Bioinformatics* 6:102
- [35] Saitou N, Nei M (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution* 4(4):406–425
- [36] Scheibye-Alsing K, Hoffmann S, Frankel A, et al (2009) Sequence assembly. *Computational biology and chemistry* 33(2):121–136
- [37] Setubal JC, Meidanis J (1997) Introduction to computational molecular biology. PWS Pub. Boston
- [38] Simpson JT, Durbin R (2012) Efficient de novo assembly of large genomes using compressed data structures. *Genome Research* 22(3):549–556
- [39] Thurmond J, Goodman JL, Strelets VB, et al (2018) FlyBase 2.0: the next generation. *Nucleic Acids Research* 47(D1):D759–D765
- [40] Yang L, Zhang X, Wang T (2010) The Burrows–Wheeler similarity distribution between biological sequences based on Burrows–Wheeler transform. *Journal of Theoretical Biology* 262(4):742–749

- [41] Zhu Z, Niu B, Chen J, et al (2013) Mgaviewer: a desktop visualization tool for analysis of metagenomics alignment data. *Bioinform* 29(1):122–123