Lightweight Metagenomic Classification via eBWT*

Veronica Guerrini[0000−0001−8888−9243] and Giovanna Rosone**[0000−0001−5075−1214]

Department of Computer Science
University of Pisa
Largo B. Pontecorvo 3, 56127 Pisa, Italy
veronica.guerrini@di.unipi.it
giovanna.rosone@unipi.it

Abstract. The development of Next Generation Sequencing has had a major impact on the study of genetic sequences, and in particular, on the advancement of metagenomics, whose aim is to identify the microorganisms that are present in a sample collected directly from the environment. In this paper, we describe a new lightweight alignment-free and assembly-free framework for metagenomic classification that compares each unknown sequence in the sample to a collection of known genomes. We take advantage of the combinatorial properties of an extension of the Burrows-Wheeler transform, and we sequentially scan the required data structures, so that we can analyze unknown sequences of large collections using little internal memory. For the best of our knowledge, this is the first approach that is assembly- and alignment-free, and is not based on k-mers. We show that our experiments confirm the effectiveness of our approach and the high accuracy even in negative control samples. Indeed we only classify 1 short read on $5,726,358$ random shuffle reads. Finally, the results are comparable with those achieved by read-mapping classifiers and by k-mer based classifiers.

Keywords: Metagenomics · Next-generation sequencing · Classification · Alignment-free · Assembly-free · eBWT · LCP Array.

1 Introduction

The advent of "next-generation" DNA sequencing (NGS) technologies has meant that collections of hundreds of millions of DNA sequences are now commonplace

* The final authenticated publication is available online at https://doi.org/10.1007/978-3-030-18174-1_8. Thanks to Published source. Please, cite the publisher version: Veronica Guerrini, Giovanna Rosone: Lightweight Metagenomic Classification via eBWT. Algorithms for Computational Biology. AlCoB 2019. Lecture Notes in Computer Science , vol 11488. Springer, Cham. DOI: https://doi.org/10.1007/978-3-030-18174-1_8. VG is totally and GR is partially supported by the project MIUR-SIR CMACBioSeq ("Combinatorial methods for analysis and compression of biological sequences") grant n. RBSI146R5L.

** Corresponding author

in bioinformatics. One research field that has grown extraordinarily in recent years is metagenomics [26]. The problem of comparing sequences is of fundamental importance in this field: indeed, the metagenomic studies require computational tools that are able to analyze large datasets and to extract correct information about the community under investigation. There exist many metagenomics statistical/computational tools (among them [30,25,22,11]) and recent surveys (for instance [13,21]) that offer a thorough benchmarking analysis by comparing the majority of the state-of-the-art tools.

We propose a new alignment-free and assembly-free method for comparing sequences (cf. [29,20]), which is combinatorial by nature and allows us to use little internal memory with respect to other approaches, such as those k-mer based. Our method is based on an extension of the Burrows-Wheeler Transform (shortly eBWT) to a collection of sequences. The eBWT has been used in several application contexts as the circular pattern matching (cf. [9]) and the alignment-free methods for comparing sequences (cf. [17,18,31,23,14]). Different distance measures have been defined and successfully applied to several biological datasets, as for instance mitochondrial DNA genomes [17,18], expressed sequence tags [23] and proteins [31]. Usually, when the eBWT is applied to a collection $\mathcal{S}$ of sequences, its output string $\mathsf{ebwt}(\mathcal{S})$ is enriched by another data structure: the document array $\mathsf{da}(\mathcal{S})$, i.e. a different color can be assigned to each element of $\mathcal{S}$ and each symbol of $\mathsf{ebwt}(\mathcal{S})$ is associated with a color in $\mathsf{da}(\mathcal{S})$. In other words, the array $\mathsf{da}(\mathcal{S})$ contains a sequence of colors that depends on how the suffixes of the sequences in $\mathcal{S}$ are mixed in the sorted list. In [17,18], the authors define a class of dissimilarity measures that, by using the eBWT, formalize the intuitive idea that the greater is the number of substrings shared by two sequences u and v, the smaller is the "distance" between u and v.

In this paper, inspired by the same intuitive idea, we define a new similarity measure that is based on an important property of the eBWT (the clustering effect [28,19] of the input symbols) together with the information on the length of the contexts that follow them. Then we describe how to apply this notion of similarity to perform metagenomic classifications.

Finally, in the last section, we describe the results of preliminary experiments on simulated metagenome collections: we obtain similar or better precision than CLARK-S [24] (a k-mer based approach), yet using a smaller memory footprint.

We show, moreover, that the sensitivity and precision obtained are similar to those achieved by Centrifuge [11], a read-mapping classifier. Nevertheless, our method gets better results than Centrifuge on the negative control datasets comprising random shuffled reads that do not belong to any known genome. Indeed, in metagenomic samples, a large number of reads are from "unknown" organisms whose genomes are not present in any reference database, and thus they cannot be given a taxonomic assignment. To mimic these reads, negative control datasets have been designed as to test the reliability of a method [13].

2 Preliminaries and materials

Let S be a string (or sequence), n its length, and Σ its alphabet set, with $\sigma = |\Sigma|$. We denote the i-th symbol of S by $S[i]$. We denote by $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ a collection of m strings. We assume that each string $S_i \in \mathcal{S}$ of length n_i is followed by a special symbol $S_i[n_i + 1] = \$_i$, which is lexicographically smaller than any other characters in $\mathcal{S}$, and do not appear in $\mathcal{S}$ elsewhere — for implementation purposes, we may simply use a unique end-marker \$ for all strings in $\mathcal{S}$. A *substring* of any $S \in \mathcal{S}$ is denoted as $S[i, j] = S[i] \cdots S[j]$, with $S[1, j]$ being called a *prefix* and $S[i, n + 1]$ a *suffix* of S. A *range* is delimited by a square bracket if the correspondent endpoint is included.

The *Burrows-Wheeler Transform* (BWT) [4] is a well known reversible string transformation widely used that can be extended to a collection of strings. Such an extension, known as eBWT or multi-string BWT, is a reversible transformation that produces a string (denoted by $\mathsf{ebwt}(\mathcal{S})$) that is a permutation of the symbols of all strings in $\mathcal{S}$ [17] (see also [2,16,7,6]). The length of $\mathsf{ebwt}(\mathcal{S})$ is denoted by $N = \sum_{i=1}^{m} n_i + m$ and $\mathsf{ebwt}(\mathcal{S})[i] = x$, with $1 \leq i \leq N$, if x circularly precedes the i-th suffix $S_j[k, n_j + 1]$ (for some $1 \leq j \leq m$ and $1 \leq k \leq n_j+1$), according to the lexicographic sorting of the suffixes of all strings in $\mathcal{S}$. In this case we say that the suffix $S_j[k, n_j + 1]$ is associated with the position i in $\mathsf{ebwt}(\mathcal{S})$ and with the color $j \in \{1, 2, \ldots, m\}$. The output string $\mathsf{ebwt}(\mathcal{S})$ is enhanced with the array $\mathsf{da}(\mathcal{S})$ of length N where $\mathsf{da}(\mathcal{S})[i] = j$, with $1 \leq j \leq m$ and $1 \leq i \leq N$, if $\mathsf{ebwt}(\mathcal{S})[i]$ is a symbol of the string $S_j \in \mathcal{S}$. See Figure 1 for an example.

The *longest common prefix* (LCP) array of $\mathcal{S}$ is the array $\mathsf{lcp}(\mathcal{S})$ of length $N + 1$, such that $\mathsf{lcp}(\mathcal{S})[i]$, with $2 \leq i \leq N$, is the length of the longest common prefix between the suffixes associated with the positions i and $i - 1$ in $\mathsf{ebwt}(\mathcal{S})$, and $\mathsf{lcp}(\mathcal{S})[1] = \mathsf{lcp}(\mathcal{S})[N + 1] = 0$ by default.

The set $\mathcal{S}$ will be omitted if it is clear from the context. Moreover, for clarity of description, we denote by $\mathcal{L}(\mathcal{S})$ the sorted list of the suffixes in $\mathcal{S}$, although we do not need it for our computation.

We call *u-occurrence* any substring u that occurs in any sequence of $\mathcal{S}$.

Remark 1. Recall that $\mathsf{ebwt}(\mathcal{S})$ is implicitly associated with $\mathcal{L}(\mathcal{S})$ and all the suffixes in $\mathcal{S}$ starting with the same substring u, with $|u| = k$, must be consecutive entries in $\mathcal{L}(\mathcal{S})$ in the range $[h, j]$. Moreover, $\mathsf{lcp}[i] \geq k$ for $i = h + 1, \ldots, j$ and the symbols of $\mathcal{S}$ that are followed by u-occurrences coincide with $\mathsf{ebwt}[h, j]$.

Remark 2. Let ℓ be the total number of u-occurrences in $\mathcal{S}$, with $|u| = k$, there exist $k - 1$ substrings (*i.e.* all suffixes of u that are not equal to u) which appear *at least* ℓ times in $\mathcal{S}$.

Example 3 (running example). Let $\mathcal{S} = \{S_1 = A\mathbf{CGT}CGCATTAA, S_2 = \mathbf{CGT}CACATNA\}$. The substring CGT appears exactly once in both sequences. The two suffixes of S_1 and S_2 starting with CGT-occurrences occupy consecutive positions, precisely 14 and 15, and $\mathsf{lcp}[15] = 4$. Moreover, according to Remark 2 the number of GT-occurrences is 2 and the one of T-occurrences is 5.

index	da($\mathcal{S}$)	lcp($\mathcal{S}$)	ebwt($\mathcal{S}$)	$\mathcal{L}(\mathcal{S})$
1	1	0	A	$\$_1$
2	2	0	A	$\$_2$
3	1	0	A	$A\$_1$
4	2	1	N	$A\$_2$
5	1	1	T	$AA\$_1$
6	2	1	C	$ACATNA\$_2$
7	1	2	$\$_1$	$ACGTCGCATTAA\$_1$
8	2	1	C	$ATNA\$_2$
9	1	2	C	$ATTAA\$_1$
10	2	0	T	$CACATNA\$_2$
11	2	2	A	$CATNA\$_2$
12	1	3	G	$CATTAA\$_1$
13	1	1	T	$CGCATTAA\$_1$
14	2	2	$\$_2$	$CGTCACATNA\$_2$
15	1	4	A	$CGTCGCATTAA\$_1$
16	1	0	C	$GCATTAA\$_1$
17	2	1	C	$GTCACATNA\$_2$
18	1	3	C	$GTCGCATTAA\$_1$
19	2	0	T	$NA\$_2$
20	1	0	T	$TAA\$_1$
21	2	1	G	$TCACATNA\$_2$
22	1	2	G	$TCGCATTAA\$_1$
23	2	1	A	$TNA\$_2$
24	1	1	T	$TTAA\$_1$

Fig. 1. The required data structures for our running example, where $\mathcal{S}$ is the set $\{S_1 = ACGTCGCATTAA, S_2 = CGTCACATNA\}$

3 Method

In this section, we introduce a new strategy to tackle the problem of metagenomic classification that is assembly- and alignment-free, not based on k-mer, and uses a little amount of memory.

We suppose that $\mathcal{S} = \{S_1, \ldots, S_m\}$ is a collection of biological sequences comprising r reads and g genomes, where $m = r + g$. More in details, $S_i \in \mathcal{S}$ is a read if $1 \leq i \leq r$ and $S_j \in \mathcal{S}$ is a genome if $r + 1 \leq j \leq m$. For simplicity, we denote by $\mathcal{R}$ the subset of reads and by $\mathcal{G}$ the subset of genomes. Assume that Σ is the biological alphabet of these sequences.

We introduce a method that classifies any read S_i in $\mathcal{R}$ by assigning it to a unique genome $S_j \in \mathcal{G}$ through reading in sequential way ebwt($\mathcal{S}$), da($\mathcal{S}$) and lcp($\mathcal{S}$).

We define a notion of similarity between sequences that exploits the underlying properties of the eBWT: the clustering effect, i.e. the fact that this transformation tends to group together symbols that occur in similar contexts in the input strings collection. Indeed, when applying eBWT to a collection, if a substring u occurs in one or more sequences, then the suffixes of the collection starting with u are likely to be close in the sorted list of suffixes. This implies that the greater the number of substrings shared by two sequences is, the more they are similar. Roughly speaking, we consider the symbols of $\mathcal{S}$ followed by a same substring (i.e. context) that are clustered together in ebwt($\mathcal{S}$) and match one-to-one the symbols belonging to $\mathcal{R}$ to the symbols belonging to $\mathcal{G}$.

Our method works in three steps: (1) we detect and keep some blocks of ebwt($\mathcal{S}$) in which the suffixes in $\mathcal{L}(\mathcal{S})$ share a common context of a minimum

length α, and to which sequences both in $\mathcal{R}$ and $\mathcal{G}$ belong; (2) we analyze these interesting blocks in order to evaluate a degree of similarity between any read and any genome in $\mathcal{S}$; (3) we perform the read assignment: for every read in $\mathcal{R}$, either we retrieve the unique genome of belonging, or we report that it is not possible to identify it.

Build α-clusters collection — In step (1), inspired by Remark 1, we build a collection $\mathcal{C}_\alpha$ of blocks in $\mathsf{ebwt}(\mathcal{S})$, which are delimited by pairs of indices called α-clusters, that are associated with LCP-values exceeding a threshold value α.

Definition 4. *Let α be a positive integer, $\mathsf{lcp}[1, N+1]$ be the* LCP*-array and $\mathsf{da}[1, N]$ the document array associated with $\mathsf{ebwt}[1, N]$. An* α-cluster *of $\mathsf{ebwt}(\mathcal{S})$ of size $pE - pS + 1$ is any pair of indices (pS, pE) in $[1, N]$ such that*
- $\mathsf{lcp}[pS] < \alpha$, *and* $\mathsf{lcp}[pE + 1] < \alpha$,
- $\mathsf{lcp}[i] \geq \alpha$, *for every* $pS < i \leq pE$,
- *there exist two indices* s, t, $pS \leq s, t \leq pE$, *such that* $\mathsf{da}[s] \leq r$ *and* $\mathsf{da}[t] > r$, *where r is the total number of reads in $\mathcal{S}$.*

Example 5 (running example). For $\alpha = 2$, the set $\mathcal{C}_2$ of 2-clusters of the $\mathsf{ebwt}(\mathcal{S})$ in Figure 1 is $\mathcal{C}_2 = \{(6, 7), (8, 9), (10, 12), (13, 15), (17, 18), (21, 22)\}$.

In other words, we discard the blocks of $\mathsf{ebwt}(\mathcal{S})$ whose associated suffixes do not share a prefix of length at least α. This step requires a sequential scan of $\mathsf{lcp}(\mathcal{S})$ and $\mathsf{da}(\mathcal{S})$ allowing us to use only a small amount of memory to detect α-clusters.

Remark 6. It is to see that we are computing the similarity between a read $S_j \in \mathcal{R}$ and a genome $S_k \in \mathcal{G}$ by analyzing the entire set of sequences $\mathcal{S}$, not only the two sequences S_j and S_k. Indeed, let (pS, pE) be an α-cluster of $\mathsf{ebwt}(\mathcal{S})$ that contains at least a symbol of S_j and at least a symbol of S_k. Other symbols that belong to sequences in $\mathcal{S}$ apart from S_j and S_k may also appear in $\mathsf{ebwt}[pS, pE]$. Nevertheless, we can implicitly get a new cluster (pS', pE') by deleting from the $\mathsf{ebwt}(\mathcal{S})$ all symbols not belonging to S_j and S_k, and for the properties of the LCP array, it is easy to verify that (pS', pE') forms an α-cluster.

Build similarity arrays — During the second step, we refine each α-cluster of the $\mathsf{ebwt}(\mathcal{S})$ by splitting it according to its symbols, and then we measure the degree of similarity between the sequences in $\mathcal{R}$ and the genomes in $\mathcal{G}$.

We split the alphabet Σ of $\mathcal{S}$ in two subsets. We include the DNA bases in $\Sigma' = \{A, C, G, T\}$ and the end-marker symbols, the (rare) occurrences of N and other degenerate base symbols (see IUPAC nomenclature) in $\Sigma'' = \Sigma \setminus \Sigma' \cup \{\$\}$.

Definition 7. *Let a be any symbol in Σ'. The* a-refinement of an α-cluster (pS, pE) *of $\mathsf{ebwt}(\mathcal{S})$ is the set of indices $\{j_1, \ldots, j_q\}$ in the range $[pS, pE]$, such that $\mathsf{ebwt}[j_\ell] = a$, for any $1 \leq \ell \leq q$.*

Example 8 (running example). The C-refinement of the cluster $(6, 7) \in \mathcal{C}_2$ is the singleton $\{6\}$, while the a-refinement, for any $a \in \{A, G, T\}$, is the empty set, since neither A nor G nor T appear in $\mathsf{ebwt}[8, 9]$. On the other hand, the C-refinement of the cluster $(8, 9) \in \mathcal{C}_2$ is the set $\{8, 9\}$.

Now, we define a similarity between two sequences $S_j, S_k \in \mathcal{S}$ by using the notion of α-cluster and a-refinement.

Definition 9. *Let $\mathcal{C}_\alpha$ be the set of all the α-clusters associated with $\mathsf{ebwt}(\mathcal{S})$. We define the* α-similarity *between two sequences $S_j \in \mathcal{R}$ and $S_k \in \mathcal{G}$ as the quantity $\mathfrak{S}_\alpha(S_j, S_k) = \sum_{x \in \mathcal{C}_\alpha} Q_{j,k}(x)$, where*

$$Q_{j,k}(x) = \sum_{a \in \Sigma'} \min\left(n_{(j,x,a)}, n_{(k,x,a)}\right), \tag{1}$$

with $n_{(j,x,a)}$ (resp. $n_{(k,x,a)}$) being the number of symbols belonging to S_j (resp. S_k) in the a-refinement of the α-cluster x.

Intuitively, during the computation of our measure, we count the symbols of each read that we can associate with the same symbols of each genome in the α-cluster, or vice versa. In particular, if the symbol belongs to Σ', we associate the nucleotide of a read in $\mathcal{R}$ with the exact nucleotide of a genome in $\mathcal{G}$. Whereas if the symbol belongs to Σ'', we consider it as placeholder, in the sense that we associate it with any nucleotide of the sequence of the other collection in order to maximize the quantity $Q_{i,j}(x)$ in Eq. (1) (see Example 10 below).

More precisely, let $m_{(j,x)}$ (resp. $m_{(k,x)}$) be the number of Σ''-symbols belonging to S_j (resp. S_k) and appearing in an α-cluster x. For any $a \in \Sigma'$, if $|n_{(j,x,a)} - n_{(k,x,a)}| > 0$ (i.e. the minimum between the number of a-symbols of S_j appearing in x and the number of a-symbols of S_k appearing in x can be increased), then we convert some placeholders to a-symbols and decrease the quantities $m_{(j,x)}$ and $m_{(k,x)}$ accordingly. Note that the symbol $a \in \Sigma'$ to which we convert any placeholder symbol appearing in x is unique.

Furthermore, if n_j (resp. n_k) is the length of S_j (resp. S_k), then the quantity $\mathfrak{S}_\alpha(S_j, S_k)$ ranges between 0 and $\min(n_j, n_k) + 1 - \alpha$. We can normalize dividing by $\min(n_j, n_k) + 1 - \alpha$, as to obtain a similarity value within the range $[0, 1]$.

Example 10 (running example). The 2-similarity between S_1 and S_2 is given by $\mathfrak{S}_2(S_1, S_2) = 1 + 1 + 0 + 1 + 1 + 1 = 5$, by setting $\$_1 = C$ and $\$_2 = T$, and by normalizing $\mathfrak{S}_2(S_1, S_2)/9 = 0,56$. On the other hand, for $\alpha = 3$, the normalized similarity $\mathfrak{S}_3(S_1, S_2)$ is equal to $0,25$ if and only if $\$_2 = A$.

Concerning the metagenome analysis and the set $\mathcal{S}$, we build a set of similarity arrays $\{Sim_1, \ldots, Sim_g\}$. More precisely, for each genome $S_k \in \mathcal{G}$, we define the array Sim_{k-r} of length r, whose entry $Sim_{k-r}[j]$ stores the normalized similarity value $\mathfrak{S}_\alpha(S_j, S_k)$, for every $S_j \in \mathcal{R}$.

In order to use only a sequential scan of $\mathsf{ebwt}(\mathcal{S})$ and $\mathsf{da}(\mathcal{S})$, we analyze the α-clusters in $\mathcal{C}_\alpha$ one by one through $|\mathcal{C}_\alpha|$ iterations. At each iteration, we consider

an α-cluster x and we evaluate the quantities $Q_{j,k}(x)$ of Eq. (1), for every index $j \leq r$ and $k > r$ appearing in x, maximizing them by means of placeholder symbols (if there are any). Then, we update each corresponding entry $Sim_{k-r}[j]$ by adding the quantity $Q_{j,k}(x)$.

Finally, once all the α-clusters in $\mathcal{C}_\alpha$ have been examined, we normalize each entry of $\{Sim_1, \ldots, Sim_g\}$ completing the construction of the similarity arrays.

Classification— The last step consists in assigning a unique provenance to any read S_j ($j \leq r$) with respect to the normalized values $\mathfrak{S}_\alpha(S_j, S_k)$, $r < k \leq r + g$.

For every $j \leq r$, we compute the set $\mathcal{I}$ of indices q such that the normalized similarity $\mathfrak{S}_\alpha(S_j, S_q)$ is close to the maximum normalized similarity $M = \max_i \mathfrak{S}_\alpha(S_j, S_i)$, i.e.

$$\mathfrak{S}_\alpha(S_j, S_q) \sim M, \text{ for all } q \in \mathcal{I}\,. \tag{2}$$

Moreover, in order to control the assessment score, we set a threshold value β ($0 \leq \beta < 1$) that the maximum value M of Eq. (2) has to exceed in order to classify the read S_j with respect to $\mathcal{I}$.

We assign the read S_j (or its reverse complement) to S_q if $q \in \mathcal{I}$, $|\mathcal{I}| = 1$ and $\mathfrak{S}_\alpha(S_j, S_q) > \beta$. Whereas, the read S_j is said to be *not classified* if $\max_i \mathfrak{S}_\alpha(S_j, S_i) \leq \beta$. Finally, the read classification of S_j is said to be *ambiguous* if $\mathfrak{S}_\alpha(S_j, S_q) > \beta$ and $|\mathcal{I}| > 1$. In the last case, if our strategy is used for the analysis of a paired-end collection, we use the sum of the assignment scores of the individual mates and assign the read to the genome that obtains the maximum score. Note that if more than one genome obtains the maximum score, we could classify the read at higher taxonomic ranks.

4 Results

In this section we evaluate our alignment-free strategy against other tools. We choose two tools: the first is alignment-free and is based on the use of k-mers, and the second is based on a read-mapping strategy. To assess the performance of our sequence analysis method, we have implemented a prototype C++ tool, named LightMetaEbwt[1].

A recent evaluation of the state-of-the-art tools for metagenome classification [13] presents the most widely used tools tested on complex and realistic datasets which have been designed ad hoc for this analysis[2]. According to this benchmarking analysis, kraken [30] and CLARK [25] result to be top-performing tools in terms of both similarity to the correct answer and the fraction of reads classified [13]. Note that both tools are k-mer based. However, for our evaluation, we selected the new version of CLARK, called CLARK-S [24], that uses spaced k-mers rather than simple k-mers, and achieves higher sensitivity than both CLARK and kraken, while maintaining high precision. Nevertheless, the tool

[1] `https://github.com/veronicaguerrini/LightMetaEbwt`

[2] `http://www.gardner-binflab.org/our_research/`

CLARK-S, as well as CLARK and kraken, is extremely memory-consuming, and the results obtained by running its lightweight version CLARK-l are indicated to be a "draft, or first order approximation" of those obtained by running CLARK or CLARK-S.

We also compare our results with a recent metagenomics classifier, named Centrifuge [11]. It adapts the data structures of read-mapping algorithms based on the BWT and the FM-index [8], such as Bowtie [12], which provide very fast alignment with a relatively small memory footprint. We observe that LightMetaEbwt, unlike Centrifuge, processes all reads at the same time.

In order to guarantee a fair evaluation, we use the custom reference database for the three tools. Notice that a like-for-like comparison on the time-consuming between LightMetaEbwt, CLARK-S and Cenfrifuge is not possible, since CLARK-S and Centrifuge are multi-thread and our tool is currently able to use one core only. In order to run CLARK-S, we use a machine with 128 GB of RAM. All tests were done on a DELL PowerEdge R630 machine, 24-core machine with Intel(R) Xeon(R) CPU E5-2620 v3 at 2.40 GHz, with 128 GB of shared memory, used in not exclusive mode. The system is Ubuntu 14.04.2 LTS.

Dataset description. The reference database $\mathcal{G}$ we use for our experiments comprises 930 genomes from 686 species belonging to 17 phyla as indicated in [13].

We perform validation of our approach by using two sets of metagenomes among those provided by Lindgreen et al. [13]: the two datasets of paired end reads *setA2* and *setB2* reproduce the size, complexity and characteristics of real metagenomic samples containing around 20 millions of sequences of length 100 belonging to $\mathcal{G}$. Some phyla are included in equal proportions, whereas some others vary more substantially between the two sets.

Moreover, as to test the reliability of the tools, each dataset includes a subset of simulated negative control sequences to mimic sequences from "unknown" organisms (i.e. their genomes are not present in the reference database) that are likely to appear in metagenome samples – see [13] for further details. Each of these negative control datasets, called *setA2_Ran* and *setB2_Ran* in our experiments, includes around 5 million of random shuffled reads.

We precise that the original datasets, downloadable from [13], are not exactly the datasets *setA2* and *setB2* we use for our evaluations[3]. In fact, we first removed a group of reads associated with the phylum of Eukaryotes whose species provenance was not specified in [13]. Second, since we use a custom database and CLARK-S downloads up-to-date taxonomy data (such as taxonomy id, or accession numbers) from the NCBI website ignoring expiring entries, we preferred not to include in sets *setA2* and *setB2* a group of reads associated with 3 genomes whose entries in the NCBI database have been indicated as obsolete.

Preprocessing step. This task for our tool can be achieved using, for example, BCR [5], Egsa [16], gsacak [15], GAP [7] or eGAP [6]. As the set $\mathcal{G}$ of genomes is the same for each experiment, we can build the data structures of $\mathcal{G}$ only once,

[3] `https://github.com/veronicaguerrini/LightMetaEbwt/tree/master/Datasets`

Table 1. Results of metagenome analysis at species level of *setA2* and *setB2* for positive control, and *setA2_Ran* and *setB2_Ran* for negative control. Best scores are in bold

setA2	REAL	CLARK-S –highconfidence	LightMetaEbwt α 16 β 0.25	LightMetaEbwt α 16 β 0.35	Centrifuge min-hitlen 16	Centrifuge min-hitlen 22
TP	21,461,160	19,789,944	19,908,394	19,815,751	20,062,940	19,897,787
FP	0	187,386	37,232	34,408	485,353	68,722
FN	0	1,483,830	1,515,534	1,611,001	912,867	1,494,651
SEN (%)	100.000	93.025	92.926	92.481	**95.648**	93.013
PREC (%)	100.000	99.062	99.813	**99.827**	97.638	99.656
F1 (%)	100.000	95.949	**96.247**	96.014	**96.633**	96.220
setB2						
TP	20,249,373	18,644,316	18,922,266	18,819,348	18,913,373	18,766,021
FP	0	167,709	73,208	68,154	450,209	58,766
FN	0	1,437,348	1,253,899	1,361,871	885,791	1,424,586
SEN (%)	100.000	92.842	93.785	93.251	**95.526**	92.944
PREC (%)	100.000	99.109	99.615	**99.639**	97.675	99.688
F1 (%)	100.000	95.873	**96.612**	96.340	96.589	96.198
setA2_Ran						
TN	5,726,358	5,726,336	5,726,294	5,726,357	150,971	5,712,085
FP	0	22	64	1	5,575,387	14,273
SPEC (%)	100.00	99.99	99.99	**100.00**	2.64	99.75
setB2_Ran						
TN	5,406,659	5,406,642	5,406,601	5,406,658	141,994	5,393,260
FP	0	17	58	1	5,264,665	13,399
SPEC (%)	100.00	99.99	99.99	**100.00**	2.63	99.75

by using GAP[4]. Then we can use BCR[5] (it is a tool for very large collection of short reads) for building the data structures for $\mathcal{R}$ and use eGAP[6] for merging them obtaining the data structures for the entire collection $\mathcal{S}$. On the other hand, exploiting the mathematical properties of the permutation associated with the eBWT and LCP array, by using BCR [2, Remark 3.6] [5], we can update the data structures of $\mathcal{G}$ (without constructing the BWT from scratch) in order to obtain the data structures for $\mathcal{S}$. To find the best method for building our data structures is a non-trivial problem and it is not in the aim of this paper.

Validation step. As the provenance of simulated reads is known, we can set TP as the number of reads correctly classified (i.e. assigned to their right provenance), FP as the number of reads erroneously classified, and FN as the number of reads unassigned, from which we can calculate the quality metrics: sensitivity $SEN = \frac{TP}{TP+FN}$, precision $PREC = \frac{TP}{TP+FP}$, and F1 score $F_1 = \frac{2TP}{2TP+FP+FN}$. In these experiments, we do not handle the reads classified as ambiguous or assigned to taxonomic level higher than species (i.e. more species could be assigned to them), and we count them among unclassified reads in FN. For simulated negative control sequences that do not exist in any known species, we can set TN as the number of random reads that are not mapped to any genome and FP as the number of random reads that are erroneously mapped to some genome, and calculate the specificity $SPEC = \frac{TN}{TN+FP}$.

[4] `http://people.unipmn.it/manzini/gap`
[5] `https://github.com/giovannarosone/BCR_LCP_GSA`
[6] `https://github.com/felipelouza/egap`

Experiments Our tool is able to classify the reads to several taxonomic levels such as genomes, species or phylum. For the experiments reported in Table 1, we choose a deep taxonomic level, i.e. we classify each read to the species level.

CLARK-S runs with default values and the results are filtered by using the recommended option `--highconfidence` (e.g., assignment with confidence score < 0.75 and gamma score < 0.03 are discarded).

For LightMetaEbwt, we set the minimum length of the common context $\alpha = 16$, since the length of each paired end read is 100, and provide results for minimum similarity scores $\beta = 0.25$ and $\beta = 0.35$. Our similarity score ranges between 0 and 1: clearly the greater the value is, the higher the read similarity is. Thus, for $\beta = 0.25$ the sensitivity increases and the precision slightly decreases.

Centrifuge begins with a short exact match (16-bp minimum) and extends the match as far as possible. Based on the exact matches found in the read and its reverse complement, Centrifuge classifies each read using only those mappings with at least one match of k bases. This parameter k (named `--min-hitlen`) is comparable with α used in our tool. Hence, we perform a first experiment where we set it to 16 and a second experiment to 22 (default value). For both *setA2* and *setB2*, the highest sensitivity achieved is given by choosing `--min-hitlen`=16. Nevertheless, for `--min-hitlen`=16 such a higher sensitivity alters the correct metagenomic classification as the percentage of random shuffled reads classified (i.e. the specificity) dramatically decreases up to 2.63%. The higher sensitivity for `--min-hitlen`=16 increases the F1 score, which is the harmonic mean of precision and sensitivity. In fact, for *setA2* the best F1 score is obtained by Centrifuge for `--min-hitlen`=16 followed by LightMetaEbwt with $\beta = 0.25$. Further experiments with $\beta = 0.1$ show that the F1 score obtained by LightMetaEbwt increases to 97.1% at the cost of a slightly low specificity (98.4%).

Without considering the pre-processing steps, we observe that the RAM usage of our tool (by using a semi-external memory approach) is about 17-18GB for *setA2* and *setB2* and about 9-10GB for *setA2_Ran* and *setB2_Ran*. CLARK-S uses about 120GB for any dataset, whereas Centrifuge uses less than 2GB. Moreover, we observe that our method scans sequentially the required data structures, so that we could analyze unknown sequences of large collections in external memory by reducing the internal memory usage to a minimum. We have also observed that our tool is slower than the other two tools, but a more engineered implementation of our algorithm would improve our performance in terms of time and space, that we leave as further work.

Overall accuracy for the three tools was very similar, but the highest precision (keeping high sensitivity) values are obtained by our tool even in the random shuffled samples (*setA2_Ran* and *setB2_Ran*).

5 Conclusions and discussion

In this paper, we present a versatile, alignment-free, lightweight method that by sequentially scanning some data structures eBWT, LCP and DA array allows us to identify the genome to which each read belongs. We focused the attention

on species level classifications, but LightMetaEbwt can also work at higher taxonomic levels such as genus, family, class or phylum. Preliminary experiments show that the relative phylum abundance estimated meets the real dataset composition with very high precision. For instance, we obtain only $31,666$ ambiguous reads and $868,456$ not classified reads and we correctly classify $19,349,193$ in *setB2*.

Furthermore, we have considered the sequences classified as ambiguous as those not classified, but we leave a more in-depth analysis of the ambiguous reads for a further work, for instance using our tool with stronger parameters.

The idea of building the clusters of the eBWT with/without LCP is not new (see [17,18]). However, we want to specify that our notion differs from the notion of LCP-interval in the literature [1]: indeed, a LCP-interval is a particular α-cluster (pS, pE) in which at least an index i, $pS < i \leq pE$, is equal to α. It is well worth mentioning also the difference with the strategies used in [27,10], where the partitions of $\mathsf{ebwt}(\mathcal{S})$ determined by LCP-values are filtered according to their size. Here, we do not impose any constraint on the α-cluster size. Moreover, to the best of our knowledge, it is the first time that the notion of cluster is used on metagenomic classification problems.

Furthermore, it is interesting to note that the data structures used by our strategy are intrinsically dynamic: the collection $\mathcal{S}$ can be modified by inserting or removing sequences [2, Remark 3.6] [5] exploiting the mathematical properties of the permutation associated with the eBWT and LCP array (for instance by using BCR), allowing us to modify α-clusters accordingly. On the other hand, one can build and store the data structures for the genome database and then, for each new experiment, build the data structures for the read collection. To merge them and obtain $\mathsf{ebwt}(\mathcal{S})$, $\mathsf{da}(\mathcal{S})$ and $\mathsf{lcp}(\mathcal{S})$, one could use eGAP.

Finally, note that LightMetaEbwt allows a certain degree of parallelization. The analysis of the clusters is independent of each other and is thus easily parallelizable. This allows us to use multiple processors on multi-core servers that are commonplace nowadays while keeping the computational requirements low. Moreover, we note that in the recent literature there are several papers with the aim of introducing new lightweight and parallel computational strategies for building the data structures we use in our tool, for instance see [3].

In conclusion, we believe that our tool can be useful in a variety of applications in metagenomics and genomics.

References

1. Abouelhoda, M.I., Kurtz, S., Ohlebusch, E.: Replacing suffix trees with enhanced suffix arrays. Journal of Discrete Algorithms **2**(1), 53 – 86 (2004)
2. Bauer, M., Cox, A., Rosone, G.: Lightweight algorithms for constructing and inverting the BWT of string collections. Theoret. Comput. Sci. **483**(0), 134 – 148 (2013)
3. Bonizzoni, P., Della Vedova, G., Nicosia, S., Pirola, Y., Previtali, M., Rizzi, R.: Divide and conquer computation of the multi-string BWT and LCP array. In: CiE 2018. pp. 107–117. Springer International Publishing (2018)

4. Burrows, M., Wheeler, D.: A Block Sorting data Compression Algorithm. Tech. rep., DIGITAL System Research Center (1994)
5. Cox, A., Garofalo, F., Rosone, G., Sciortino, M.: Lightweight LCP construction for very large collections of strings. J. Discrete Algorithms **37**, 17–33 (2016)
6. Egidi, L., Louza, F.A., Manzini, G., Telles, G.P.: External memory BWT and LCP computation for sequence collections with applications. In: WABI 2018. LIPIcs, vol. 113, pp. 10:1–10:14 (2018)
7. Egidi, L., Manzini, G.: Lightweight BWT and LCP merging via the Gap algorithm. In: SPIRE. pp. 176–190. LNCS 10508 (2017)
8. Ferragina, P., Manzini, G.: Opportunistic data structures with applications. In: FOCS. pp. 390–398 (2000)
9. Hon, W.K., Ku, T.H., Lu, C.H., Shah, R., Thankachan, S.V.: Efficient Algorithm for Circular Burrows-Wheeler Transform. In: CPM, LNCS, vol. 7354, pp. 257–268. Springer (2012)
10. Janin, L., Rosone, G., Cox, A.J.: Adaptive reference-free compression of sequence quality scores. Bioinformatics **30**(1), 24–30 (2014)
11. Kim, D., Song, L., Breitwieser, F.P., Salzberg, S.L.: Centrifuge: rapid and sensitive classification of metagenomic sequences. Genome Res. **26**(12), 1721–1729 (2016)
12. Langmead, B., Trapnell, C., Pop, M., Salzberg, S.L.: Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol **10**(3), R25 (2009)
13. Lindgreen, S., Adair, K.L., Gardner, P.P.: An evaluation of the accuracy and speed of metagenome analysis tools. Scientific Reports **6**, 19233 (2016)
14. Louza, F.A., Telles, G.P., Gog, S., Zhao, L.: Computing burrows-wheeler similarity distributions for string collections. In: SPIRE. pp. 285–296 (2018)
15. Louza, F., Gog, S., Telles, G.: Inducing enhanced suffix arrays for string collections. Theor. Comput. Sci. **678**, 22–39 (2017)
16. Louza, F., Telles, G., Hoffmann, S., Ciferri, C.: Generalized enhanced suffix array construction in external memory. Algorithms Mol Biol **12**(1), 26 (2017)
17. Mantaci, S., Restivo, A., Rosone, G., Sciortino, M.: An extension of the Burrows-Wheeler Transform. Theoret. Comput. Sci. **387**(3), 298–312 (2007)
18. Mantaci, S., Restivo, A., Rosone, G., Sciortino, M.: A new combinatorial approach to sequence comparison. Theory Comput. Syst. **42**(3), 411–429 (2008)
19. Mantaci, S., Restivo, A., Rosone, G., Sciortino, M., Versari, L.: Measuring the clustering effect of BWT via RLE. Theoret. Comput. Sci. **698**, 79 – 87 (2017)
20. Mantaci, S., Restivo, A., Sciortino, M.: Distance measures for biological sequences: Some recent approaches. Int. J. Approx. Reasoning **47**(1), 109–124 (2008)
21. McIntyre, A.B.R., et al.: Comprehensive benchmarking and ensemble approaches for metagenomic classifiers. Genome Biol. **18**(1), 182 (2017)
22. Menzel, P., Ng, K.L., Krogh, A.: Fast and sensitive taxonomic classification for metagenomics with kaiju. Nature Communications (2016)
23. Ng, K.H., Ho, C.K., Phon-Amnuaisuk, S.: A hybrid distance measure for clustering expressed sequence tags originating from the same gene family. PLoS ONE **7**(10) (2012)
24. Ounit, R., Lonardi, S.: Higher classification sensitivity of short metagenomic reads with CLARK-S. Bioinformatics **32**(24), 3823–3825 (2016)
25. Ounit, R., Wanamaker, S., Close, T.J., Lonardi, S.: CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. BMC Genomics **16**(1), 236 (2015)
26. Pedersen, M., et al: Ancient and modern environmental DNA. Philos. Trans. R. Soc. Lond., B, Biol. Sci. **370**(1660) (2015)

27. Prezza, N., Pisanti, N., Sciortino, M., Rosone, G.: Detecting Mutations by eBWT. In: WABI 2018. LIPIcs, vol. 113, pp. 3:1–3:15 (2018)
28. Restivo, A., Rosone, G.: Balancing and clustering of words in the Burrows-Wheeler transform. Theoret. Comput. Sci. **412**(27), 3019 – 3032 (2011)
29. Vinga, S., Almeida, J.: Alignment-free sequence comparisona review. Bioinformatics **19**(4), 513–523 (2003)
30. Wood, D.E., Salzberg, S.L.: Kraken: ultrafast metagenomic sequence classification using exact alignments. Genome Biol **15**(3), R46 (2014)
31. Yang, L., Zhang, X., Wang, T.: The Burrows-Wheeler similarity distribution between biological sequences based on Burrows-Wheeler transform. Journal of Theoretical Biology **262**(4), 742–749 (2010)