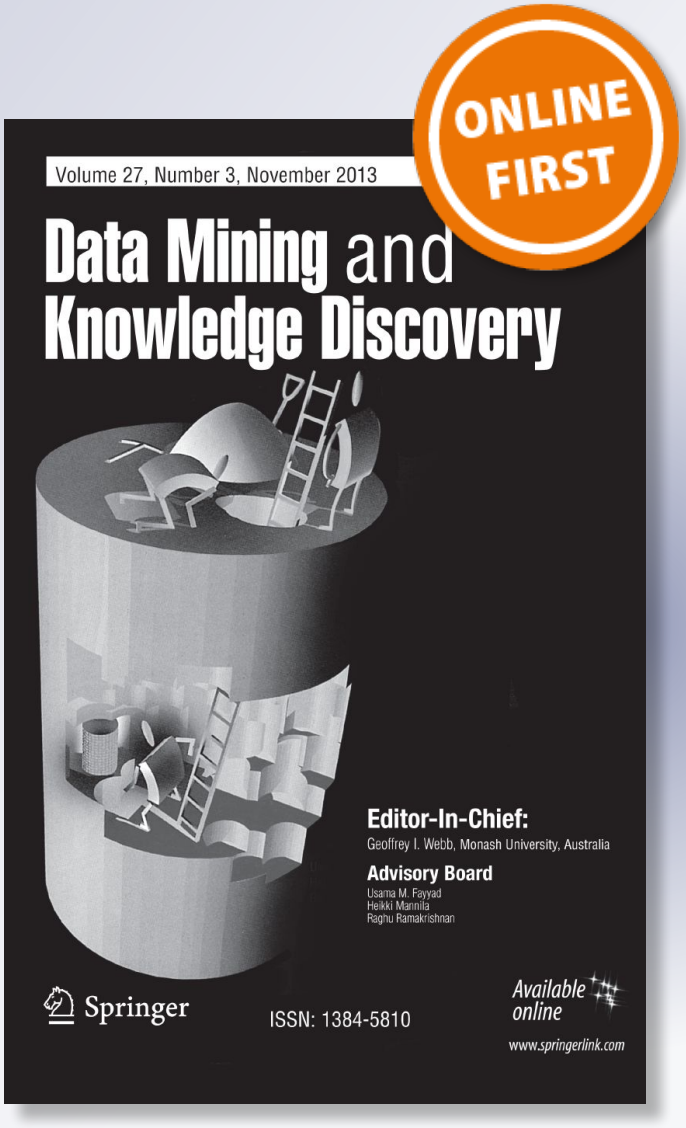
Estimating sequence similarity from read sets for clustering next-generation sequencing data

Petr Ryšavý & Filip Železný

Data Mining and Knowledge Discovery

ISSN 1384-5810

Data Min Knowl Disc
DOI 10.1007/s10618-018-0584-8



Springer

Data Mining and Knowledge Discovery
https://doi.org/10.1007/s10618-018-0584-8

Estimating sequence similarity from read sets for clustering next-generation sequencing data

Petr Ryšavý[1] · **Filip Železný**[1]

Received: 28 August 2017 / Accepted: 16 July 2018

Abstract

Computing mutual similarity of biological sequences such as DNA molecules is essential for significant biological tasks such as hierarchical clustering of genomes. Current sequencing technologies do not provide the content of entire biological sequences; rather they identify a large number of small substrings called reads, sampled at random places of the target sequence. To estimate similarity of two sequences from their read-set representations, one may try to reconstruct each one first from its read set, and then employ conventional (dis)similarity measures such as the edit distance on the assembled sequences. Due to the nature of data, sequence assembly often cannot provide a single putative sequence that matches the true DNA. Therefore, we propose instead to estimate the similarities directly from the read sets. Our approach is based on an adaptation of the Monge-Elkan similarity known from the field of databases, avoiding the sequence assembly step. For low-coverage (i.e. small) read set samples, it yields a better approximation of the true sequence similarities. This in turn results in better clustering in comparison to the first-assemble-then-cluster approach. Put differently, for a fixed estimation accuracy, our approach requires smaller read sets and thus entails reduced wet-lab costs.

Keywords Read sets · Similarity · Hierarchical clustering · Biological sequences

Responsible editor: Pierre Baldi.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10618-018-0584-8) contains supplementary material, which is available to authorized users.

✉ Petr Ryšavý
petr.rysavy@fel.cvut.cz

Filip Železný
zelezny@fel.cvut.cz

[1] Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic

Published online: 04 August 2018

1 Introduction

Sequencing means reading the sequence of elements that constitute a polymer, such as the DNA. The *human genome project* (Lander et al. 2001) completed in 2003 was a prime example of sequencing, resulting in the identification of almost the entire genomic sequence (over 3 billion symbols) of a single human.

Sequencing becomes technologically difficult as the length of the read sequence grows. The common principle of *next-generation sequencing* (NGS) (Goodwin et al. 2016) is that only very short substrings (10's to 100's of symbols) are read at random positions of the sequence of interest. It is usually required that the number of such scanned substrings, called *reads*, is such that with high probability each position in the sequence is contained in multiple reads; the number of such reads is termed *coverage*. The complete sequence is determined by combinatorial assembly of the substrings guided by their suffix-prefix overlaps. For example, one possible assembly of reads {AGGC, TGGA, GCT} is AGGCTGGA. Short reads imply low cost of wet-lab sequencing traded off with high computational cost of assembly. Indeed, the assembly task can be posed as the NP-complete problem of searching the Hamiltonian path in a graph of mutual overlaps.

One of the central tasks in computational biology is to infer phylogenetic trees. In its basic form, this task translates to hierarchical clustering (Saitou and Nei 1987) of genomes, which are sequences of nucleotides. When the genomes are represented only through sequenced read-sets, the bioinformatician is forced to reconstruct the sequences from the read-sets prior to clustering. This requires solving the sequence assembly problem for each data instance with the possibility of introducing new errors into the final assembly. Assembly algorithms usually fail to provide a single putative sequence, and several contigs are produced instead. The relative order and gaps between contigs are unknown without additional sequencing. According to Haiminen et al. (2011), up to 20 % of the true sequence may remain uncovered when working with reads 50-symbols long and with coverage (average number of times each nucleotide is read) 50. This motivates the question whether the assembly step could be entirely avoided. We address this question here by proposing a similarity function computable directly on the read sets, that should approximate the true similarities on the original sequences.

The topic of clustering NGS data has been elaborated in prior related work such as the papers of Bao et al. (2011), Malhotra et al. (2014), Kchouk and Elloumi (2016). However, these studies deal with clustering of *reads*. The paper (Weitschek et al. 2014) proposes a similarity measure for NGS data, but again it operates on the level of reads. The previous studies by Železný et al. (2014) and Jalovec and Železný (2014) also aim at avoiding the assembly step in learning from NGS data but these concern supervised classification learning, and they do not elaborate on read-set similarity.

Also of relevance is the collection of *alignment-free* measures used to cluster sequences. The original purpose of the latter was to avoid sequence alignment in hierarchical clustering. This area was started by Edwin Blaisdell (1986), which proposed so called D_2 statistics for the correlation of two sequences. In Reinert et al. (2009), some of the drawbacks of the original measure for DNA sequences were rectified, while Song et al. (2013) proposed modified measures d_2, d_2^S, and d_2^*, which are

applicable to read data, and thus we include them in comparative experiments. Comin et al. (2015) modified those measures to reflect the quality values in the sequencing data. Further different alignment-measures were proposed, for example co-phylog (Yi and Jin 2013), Mash (Ondov et al. 2016), and $\overline{Under_2}$ (Comin and Schimd 2014), out of which we embrace Ondov et al. (2016) and Yi and Jin (2013) in the experiments.

From the perspective of sequence analysis, our work relates to the area of *Approximate string matching*, where the goal is to find strings similar to a pattern in a dictionary. Specifically, we employ the concept of q-gram distance (Ukkonen 1992). Our method also employes the Monge-Elkan distance (Monge and Elkan 1996) known from the field of databases.

The present work extends the initial inquiry (Ryšavý and Železný 2016) both in methodological aspects and experimental evaluation. The most significant methodological improvement consists of sampling- and embedding-based procedures for fast approximations of the basic exact computation algorithms. Furthermore, the current paper proposes a new strategy for detecting reads with missing counterparts, superseding an analogical procedure by Ryšavý and Železný (2016). Finally, unlike in Ryšavý and Železný (2016), the present approach deals with the fact that the orientation of the input reads is unknown, just as is the DNA strand from which a read originates. The theoretical analysis of the proposed algorithm's properties has also been extended so that it includes the new embedding-based approximation. Regarding the experiments, we have adopted more real datasets and a wider range baselines for comparative evaluations including more assembly algorithms as well as two strategies for alignment-free similarity estimation.

The rest of the paper is organized as follows. In the next section, we provide a formal framework for the problem to be addressed. Section 3 then offers a solution in an iterative manner, by successively incorporating a number of ideas. In Sect. 4 we explain some details important for the implementation of the solution and also provide optimizations to increase its efficiency. Section 5 then elaborates on certain theoretical properties of the proposed methods. In Sect. 6 we evaluate several variants of our method in comparison to the conventional (assembly-based) approach on several real genomic data sets. Section 7 concludes the study a proposes a direction for further research.

2 Problem statement

We will work with strings over alphabet Σ. With no loss of generality, we will fix Σ to {A, T, C, G} corresponding to nucleotide types in genomes. If a_i is a read, then $\overline{a_i}$ denotes its *complement*, in which A (C, respectively) and T (G) are interchanged. For example, the complement of read ATTCG is read TAAGC. Furthermore, we define a reversed read, denoted $\underline{a_i}$. For instance, the reverse of ATTCG is GCTTA. The reverse complement of a_i is marked as $\overline{\underline{a_i}}$.

The functor $|.|$ will denote the absolute value, cardinality, and length (respectively) for a number, set, and string argument. Let $\mathrm{dist}(A, B)$ denote the Levenshtein distance (Levenshtein 1966) between strings A and B. The function measures the minimum number of edits (symbol insertions, deletions, and substitutions) needed to make the

strings identical, and is a typical example of a sequence dissimilarity measure used in bioinformatics. It is a property of the distance that

$$\text{dist}(A, B) \leq \max\{|A|, |B|\}. \tag{1}$$

A *read bag* R_A is a multiset of $|R_A|$ substrings, each of length

$$l \ll |A| \tag{2}$$

sampled uniformly with replacement from among all $|A| - l + 1$ substrings of length l of A. *Coverage* α of R_A is defined as

$$\alpha = l \frac{|R_A|}{|A|}. \tag{3}$$

Informally, the coverage α indicates the average number of reads covering a particular position in A.

The read length l and coverage α are determined by a particular sequencing technology and experimental setup. In this paper, we consider them constant.

Our goal is to propose a distance function $\text{Dist}(R_A, R_B)$ that approximates $\text{dist}(A, B)$ for read-sets R_A and R_B of arbitrary strings A and B. We also want $\text{Dist}(R_A, R_B)$ to be more accurate and less complex to calculate than a natural estimate $\text{dist}(\hat{A}, \hat{B})$ in which the arguments represent putative sequences reconstructed from R_A and R_B using *assembly algorithms* such as those of Simpson et al. (2009), Hernandez et al. (2008), Warren et al. (2007), Nurk et al. (2013), or Zerbino and Birney (2008).

3 Proposed solution

A natural approach to instantiate $\text{Dist}(R_A, R_B)$ is to exploit the $|R_A||R_B|$ pairwise Levenshtein distances between the reads in R_A and R_B. Most of those values are useless because they match reads from completely different parts of sequences A and B. Therefore we want to account only for those pairs which likely belong together.

If we seek a read from R_B that matches a read $a_i \in R_A$, we make the assumption that the most similar read $b_j \in R_B$ is the one that we look for (see Fig. 1), i.e.,

$$b_j = \arg\min_{b_k \in R_B} \text{dist}(a_i, b_k).$$

To calculate the distance from R_A to R_B, we average over all reads from R_A:

$$\underset{\text{ME}}{\text{Dist}}(R_A, R_B) = \frac{1}{|R_A|} \sum_{a_i \in R_A} \min_{b_j \in R_B} \text{dist}(a_i, b_j). \tag{4}$$

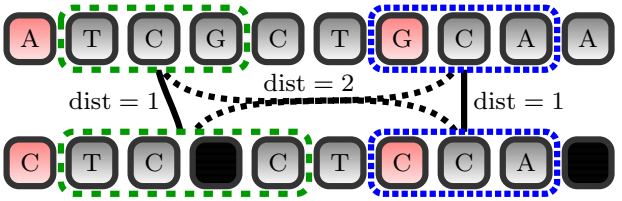


Fig. 1 We calculate read-read distances in order to find matching pairs of reads. For each read from the first sequence we find the least distant read in the second sequence. We see optimal alignment of ATCGCTGCAA and CTCCTCCA. Read TCG is paired with TCC

This idea was presented by Monge and Elkan (1996) for searching duplicates in database systems. The method is known as the *Monge-Elkan similarity*[1] (hence the ME label) and entails a simple but effective approximation algorithm.

In practical settings, to apply the above distance function as well as its variants elaborated further, a preprocessing step needs to be applied, following from the consideration in turn. First, the DNA molecule has two complementary strands in reverse directions, i.e. A and $\overline{A}$, and usually the bioinformatician does not know, which strand a read comes from. When matching a read a_i with read b_j, the matching of a_i with $\overline{b_j}$ should then also be considered. Furthermore, some sequencing techniques do not provide read *orientation* with respect to the 3'-end to 5'-end orientation of the molecule. In this case, we need to consider matching a_i with all of $b_j, \overline{b_j}, \underline{b_j}, \underline{\overline{b_j}}$. We deal with this by formally expanding the R_B set above by the explained variants of its elements.

3.1 Symmetry

$\mathrm{Dist}_{\mathsf{ME}}(R_A, R_B)$ is non-symmetric in general, which is undesirable given that the approximated distance $\mathrm{dist}(A, B)$ is known to be symmetric. Therefore we define a symmetric version by averaging both directions

$$\underset{\mathsf{MES}}{\mathrm{Dist}}(R_A, R_B) = \frac{1}{2}\left(\underset{\mathsf{ME}}{\mathrm{Dist}}(R_A, R_B) + \underset{\mathsf{ME}}{\mathrm{Dist}}(R_B, R_A)\right). \tag{5}$$

3.2 Distance scale

Consider duplicating a non-empty string A into AA and assume $R_{AA} = R_A \cup R_A$. Typically for a B similar to A, we expect that $\mathrm{dist}(AA, B) > \mathrm{dist}(A, B)$ but the (symmetric) Monge-Elkan distance will not change, i.e., $\mathrm{Dist}_{\mathsf{MES}}(R_{AA}, R_B) = \mathrm{Dist}_{\mathsf{MES}}(R_A, R_B)$, indicating a discrepancy that should be rectified.

In fact, $\mathrm{Dist}_{\mathsf{MES}}$ has the constant upper bound l, which is because it is the average (c.f. (4) and (5)) of numbers no greater than l (see (1)). On the other hand, $\mathrm{dist}(A, B)$ has a non-constant upper bound $\max\{|A|, |B|\}$ as by (1).

[1] Here we alter the Monge-Elkan similarity into a distance measure. The standard way of using Monge-Elkan is as a similarity measure with min replaced by max and distance calculation by similarity calculation.

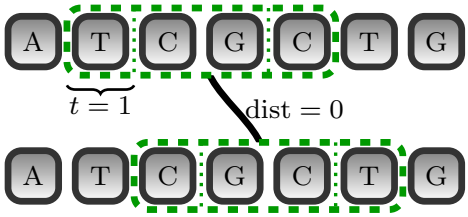


Fig. 2 Because read locations in sequences are random, we do not want to penalize small leading or trailing gaps

To bring $\text{Dist}_{\text{MES}}(A, B)$ on the same scale as $\text{dist}(A, B)$, we should therefore multiply it by the factor $\max\{|A|, |B|\}/l = \max\{|A|/l, |B|/l\}$. By (3) we have $|A| = \frac{l}{\alpha}|R_A|$, yielding the factor $\max\{|R_A|/\alpha, |R_B|/\alpha\}$. Under the assumption that α is a constant same for all read sets, we can multiply only by the factor $\max\{|R_A|, |R_B|\}$ as hierarchical clustering algorithms are scale independent. If coverage is different for R_A and R_B, we have to keep α as a divisor. Therefore, we modify the read distance into

$$\underset{\text{MESS}}{\text{Dist}}(R_A, R_B) = \max\{|R_A|, |R_B|\}\underset{\text{MES}}{\text{Dist}}(R_A, R_B). \tag{6}$$

3.3 Margin gaps

Consider the situation in Fig. 2 showing two identical sequences each with one shown read. The Levenshtein distance between the two reads is non-zero due to the one-symbol trailing (leading, respectively) gap of the top (bottom) read caused only by the different random positions of the reads rather than due to a mismatch between the sequences. Thus there is an intuitive reason to pardon margin gaps up to certain size t

$$t < \frac{l}{2} \tag{7}$$

when matching reads. Here, t should not be too large as otherwise the distance could be nullified for pairs of long reads with small random prefix-suffix overlaps.

To estimate an appropriate value for t, consider sequence A and its sampled read-set R_A. We now sample an additional read a of length l from A. Ideally, there should be a zero-penalty match for a in R_A as a was sampled from the same sequence as R_A was. This happens iff there is a read in R_A sampled from the same position in A as a, or from a position shifted by up to t symbols to the left or right as then the induced gaps are penalty-free. Since R_A is an uniform-probability independent and identically distributed (i.i.d.) sample from A, the probability that a particular read from R_A starts at one of these $1 + 2t$ positions is[2] $\frac{1+2t}{|A|}$. We want to put an upper bound $\varepsilon > 0$ on the probability that this happens for none of the $|R_A| = \frac{\alpha}{l}|A|$ reads in R_A:

$$p = \left(1 - \frac{1 + 2t}{|A|}\right)^{\frac{|A|\cdot\alpha}{l}} \leq \varepsilon.$$

[2] Strictly speaking, this reasoning is incorrect if read a is drawn from a place close to A's margins, more precisely, if it starts in fewer than t ($t + l$, respectively) symbols from A's left (right) margin, as then not all of the $2t$ shifts are possible. This is however negligible due to (2).

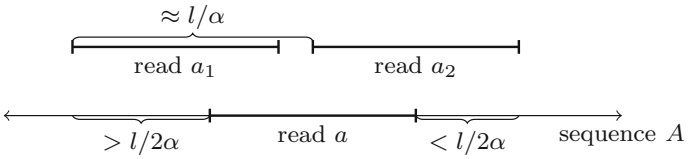


Fig. 3 Illustration to reasoning in Sect. 3.3

Consider the first-order Taylor approximation $(1+x)^n = 1 + nx + \varepsilon'$ where the difference term $\varepsilon' > 0$ decreases with decreasing $|x|$. Due to (7) and (2), $\frac{1+2t}{|A|}$ is small and we can apply the approximation on the above formula for p, yielding

$$p = 1 - \frac{2t+1}{|A|}\frac{|A|\cdot\alpha}{l} + \varepsilon' = 1 - (2t+1)\frac{\alpha}{l} + \varepsilon' \leq \varepsilon.$$

For simplicity, we choose $\varepsilon = \varepsilon'$. The smallest gap size t for which the inequality is satisfied is obtained by solving $1 - (2t+1)\frac{\alpha}{l} = 0$, yielding

$$t = \frac{1}{2}\left(\frac{l}{\alpha} - 1\right). \tag{8}$$

This choice of t matches intuition in that with larger read-length l we can allow a larger grace gap t but with larger coverage α, t needs not be so large as there is a higher chance of having a suitably positioned read in the read-set. Another way to look at it is to realize that reads in a read-set are approximately $\frac{l}{\alpha}$ positions from each other. Consider matching read a to reads from R_A. If there is a read $a_1 \in R_A$ requiring gap larger than $\frac{l}{2\alpha}$ to match a, then there will typically be another read $a_2 \in R_A$ requiring gap at most $\frac{l}{2\alpha}$ (see Fig. 3). From (7) and (8) we see that this method is applicable only when $\alpha > (\frac{1}{l}+1)^{-1}$, which is bit less than 1. However, the results start to be nonzero for $t < l$, which is by (8) equivalent to $\alpha > (\frac{1}{l}+2)^{-1}$, which is bit less than 0.5.

$\text{Dist}_{\text{MESS}}$ equipped with the margin gap technique is denoted $\text{Dist}_{\text{MESSG}}$.

3.4 Missing reads

Sometimes there is no good match for read a_i in R_B. During evolution the substring that contained a_i may have been inserted into A or may have vanished from B. We would like to detect such a situation. Denote

$$\text{BM} = \bigcup_{a_i \in R_A} \left\{ \min_{b_j \in R_B} \text{dist}(a_i, b_j) \right\}$$

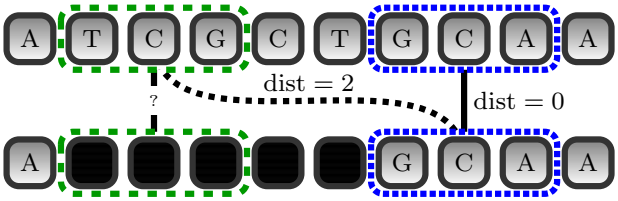


Fig. 4 If the distance between a read and its closest counterpart is an outlier, we assume that the read matches to a gap in the sequence alignment

the set of all distances from reads in R_A to their closest reads in R_B. If a read a_i has no counterpart in R_B, then its distance to the closest $b_j \in R_B$ should be larger than other distances in the set BM. In other words, $\text{dist}(a_i, b_j)$ is an outlier in BM.[3]

To detect outliers, we split the set BM to two clusters using divisive hierarchical single linkage clustering. In the case that the smaller of the clusters contains at most 25 % of the high-distance data, and the gap separating the clusters is at least one standard deviation of data in BM, we mark those distances as outliers. As each corresponding read a_i has no close counterpart in R_B, we force its distance to the closest read in R_B to be l. (See Fig. 4.) Out of curiosity we tested a z-score filtering for outliers, however, the optimal z-score threshold varied between 1 and 3 for different datasets.

The distance function equipped with the missing read detection as described gives rise to the version denoted as $\text{Dist}_{\text{MESSGM}}$.

4 Implementation and optimizations

4.1 Grace margin gaps

Implementing the grace margin gaps in function $\text{Dist}_{\text{MESSG}}$ requires only a small change to the standard Wagner–Fischer algorithm (Wagner and Fischer 1974).[4] When the algorithm fills the first and the last row and column of the table, margin gaps up to t symbols are not penalized. Larger margin gaps are penalized so that a constraint that the distance between a string a and an empty word is $|a|$. In particular, the standard linear gap penalty on margins is replaced with a piecewise linear function $g(x)$ that gives cost of a gap at x-th position

$$g(x) = \begin{cases} 0, & \text{if } 0 \le x \le t-1, \\ 2\frac{x-t+1}{l+1-2t}, & \text{if } t-1 < x \le l-t, \\ 2, & \text{if } l-t < x < l. \end{cases} \tag{9}$$

[3] The idea of finding outliers in BM was proposed by one of the reviewers of the paper and it turned out to work better than the original version by Ryšavý and Železný (2016).

[4] The dynamic programming algorithm for calculating the Levenshtein distance (Levenshtein 1966) is commonly called Wagner–Fischer algorithm (Wagner and Fischer 1974). When we refer to sequence alignment problem in bioinformatics, this algorithm is often called Needleman–Wunsch algorithm (Needleman and Wunsch 1970).

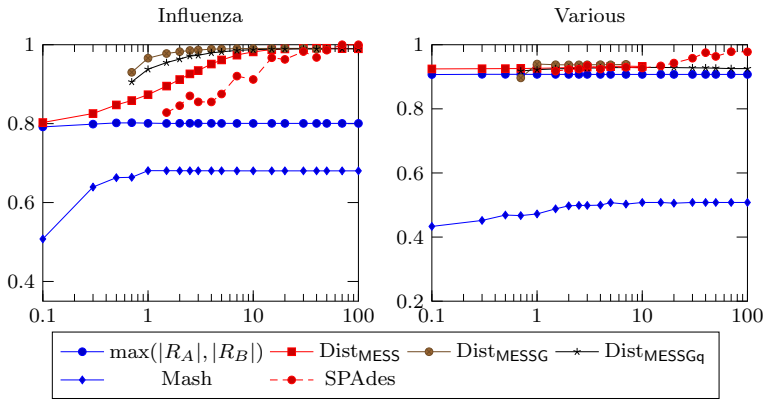


Fig. 5 Plot of average Pearson's correlation coefficient for several choices of coverage values

4.2 Sampling

Data sets collected for conventional assembly tasks usually exhibit high coverage α as the latter is essential for sufficient read overlaps, which in turn are significant for the quality of assembly. For example, study (Haiminen et al. 2011) shows that for de novo assembly, coverage up to 50 may be needed. On the other hand, the distance estimation approach elaborated here does not rely on overlaps, and thus high coverage is less essential. Figure 5 shows that our methods provide good quality results for coverage 2 and higher. This observation motivates us to conduct a pre-processing step in which high-coverage input data are replaced by a smaller random sample thereof.

4.3 Embedding

Given the heuristic nature of our method, the minimum searched in (4) need not be exact; an approximate value is acceptable if that allows a significant efficiency gain. We achieve such an approximation through a *read-embedding* technique based on the following basic idea. The high-dimensional space of reads equipped with the Levenshtein distance is mapped to a lower-dimensional space with another distance function, which is easier to compute. The two representations correspond mutually in that elements close (distant) in the original space are also close (distant) in the target space. In the latter, we first find all the elements minimizing the distance to the image of a_i in (4). For this small set of candidates, we finally compute their true Levenshtein distance to a_i in the original space to determine the closest one.

The particular embedding we use is based on the q-gram profile and q-gram distance as formalized by Ukkonen (1992). A q-gram is a string from Σ^q. The *q-gram profile* of string a is a vector $\mathbf{Q}(a) \in \mathbb{N}_0^{|\Sigma|^q}$ whose components are the counts of occurrences for all possible q-grams in a. Finally, the *q-gram distance* defined in the embedding space is simply the Manhattan distance $\mathrm{dist}_q(\mathbf{Q}(a), \mathbf{Q}(b)) = \|\mathbf{Q}(a) - \mathbf{Q}(b)\|_1$. For notational brevity later in this paper, we extend dist_q formally to the source string space in the obvious way: $\mathrm{dist}_q(a, b) \equiv \mathrm{dist}_q(\mathbf{Q}(a), \mathbf{Q}(b))$.

We instantiate q to $q = 3$ as this value has been known to provide a good balance between the runtime and quality of estimates in the context of the BLAST algorithm (Altschul et al. 1990). Navarro (2001) recommends to use $q = \log_{|\Sigma|} l$. The dimension of the embedding space is thus $|\Sigma|^q = 64$. A lower q would mean a significant loss of information because each 2-gram can be found six times on average in a random sequence of length 100. On the opposite, $q = 4$ or higher is inefficient as at most 97 fields out of 256 in 4-gram profile can be non-zero for read length 100. The distance measure stemming from $\text{Dist}_{\text{MESSG}}$ ($\text{Dist}_{\text{MESSGM}}$ respectively) while adopting the embedding technique will be referred to as $\text{Dist}_{\text{MESSGq}}$ ($\text{Dist}_{\text{MESSGMq}}$).

5 Theoretical analyses

5.1 Asymptotic complexity

Calculating $\text{dist}(A, B)$ for sequences A and B requires $\Theta(|A||B|)$ operations if we use the standard Wagner-Fischer dynamic programming algorithm (Wagner and Fischer 1974). This algorithm also requires $\Theta(\min(|A|, |B|))$ memory as we are interested only in distance and not in the alignment. To calculate Dist_{ME} we need to know the distances between all pairs of reads, so we have to evaluate (see (3)) $\frac{\alpha}{l}|A|\frac{\alpha}{l}|B|$ distances where each one requires l^2 operations. Therefore $\alpha^2|A||B|$ operations are required. For the symmetric version Dist_{MES}, we make $2\alpha^2|A||B|$ operations, which can be reduced to $\alpha^2|A||B|$ operations and $\Theta(l + \frac{\alpha}{l}(|A| + |B|))$ memory. Further modifications (MESS, MESSG, MESSGM) do not change the asymptotic complexity.

Sampling, as described in Sect. 4.2 reduces the complexity by α^2 factor to $\Theta(|A||B|)$, assuming that the final coverage is a constant. Method MESSGq does not give any theoretical guarantee on the number of pairwise edit distances that need to be explored. However, assuming this number to be a small constant yields the runtime of $\Theta(|R_A||R_B| + l^2(|R_A| + |R_B|))$.

The constants α and l are determined by the sequencing technology and the independent complexity factors are $|A|$ and $|B|$. To calculate the distance in the conventional way as $\text{dist}(\hat{A}, \hat{B})$ requires to reconstruct $\hat{A}$ and $\hat{B}$ from the respective read-sets through an assembly algorithm. This is an NP-hard problem which becomes non-tractable for large $|A|$ and $|B|$, and which is avoided by our approach.

To calculate distance matrix for n sequences, $\Theta(n^2)$ pairwise comparisons are needed. The time needed to build a hierarchical clustering is $\Theta(n^3)$ with a straightforward implementation of neighbor-joining or UPGMA. However, this time is neglectable compared to the distance matrix calculation time.

5.2 Metric properties

Dist_{MES} as well as the subsequent versions are all symmetric and non-negative but none of the proposed versions satisfies the identity condition ($\text{dist}(a, b) = 0$ iff $a = b$) or the triangle inequality, despite being based on the Levenshtein distance dist, which is a metric. For example, let $R_A = \{\text{ATC}, \text{ATC}, \text{GGG}\}$, let $R_B = \{\text{ATA}, \text{GGG}\}$, and

let $R_C = \{\mathsf{CTA}, \mathsf{GGG}\}$. Then $\mathrm{Dist}_{\mathsf{MES}}(R_A, R_B) = \frac{7}{12}$, and $\mathrm{Dist}_{\mathsf{MES}}(R_B, R_C) = \frac{1}{2}$ but $\mathrm{Dist}_{\mathsf{MES}}(R_A, R_C) = \frac{14}{12} > \frac{7}{12} + \frac{1}{2}$. While this might lead to counter-intuitive behavior of the proposed distances in certain applications, the violated conditions are not requirements assumed by clustering algorithms.

5.3 Embedding-based approximation

The q-gram distance proposed in Sect. 4.3 as an approximation to the edit distance turns out to be a lower bound on the latter. In particular, for any reads a and b and $q = 3$, we have

$$\mathrm{dist}(a, b) \geq \frac{1}{6}\mathrm{dist}_q(a, b).$$

This can be proven through mathematical induction on the Levenshtein distance between a and b. Suppose that $\mathrm{dist}(a, b) = k$.

If $k = 0$, then both $\mathrm{dist}(a, b) = \mathrm{dist}_q(a, b) = 0$ and the inequality holds.

Now suppose the bound holds for any reads with distance at most $k - 1$. There exists a sequence of k operations *insert*, *delete* and *replace* that transforms a into b. Denote b' the result that we obtain after applying the first $k - 1$ operations. Then $\mathrm{dist}(a, b') = \mathrm{dist}(a, b) - 1$. Consider the last operation and calculate the maximal value of $\mathrm{dist}_q(a, b) - \mathrm{dist}_q(a, b')$.

In the case of insertion (or, vice versa, deletion), we obtain in the worst case instead of q-grams abc,bcd, q-grams abX,bXc,Xcd (or vice versa). The q-gram distance grows by at most 5. In case of a mismatch, q-grams abX,bXc,Xcd are replaced by q-grams abY,bYc,Ycd. The q-gram distance grows at most by 6. Therefore $\mathrm{dist}_q(a, b) - \mathrm{dist}_q(a, b') \leq 6$. Making the induction step results in

$$\begin{aligned}\mathrm{dist}(a, b) &= 1 + \mathrm{dist}(a, b') \geq 1 + \frac{1}{6}\mathrm{dist}_q(a, b') \\ &\geq 1 + \frac{1}{6}\left(\mathrm{dist}_q(a, b) - 6\right) = \frac{1}{6}\mathrm{dist}_q(a, b).\end{aligned}$$

The proof is then finished by the standard mathematical induction axiom.

6 Experimental evaluation

The purpose of the experiments is to compare different methods for estimating the Levenshtein distance $\mathrm{dist}(A, B)$ for various real DNA sequences A, B from their read sets R_A, R_B. The methods include

- our newly proposed distance variants (MES, MESS, MESSG, MESSGM, MESSGq, MESSGMq) applicable directly on R_A, R_B
- the conventional method based on assembling estimates $\hat{A}$, $\hat{B}$ of the original sequences A, B using five common de-novo gene assemblers [ABySS (Simpson et al. 2009), Edena (Hernandez et al. 2008), SPAdes (Nurk et al. 2013), SSAKE

(Warren et al. 2007), and Velvet (Zerbino and Birney 2008)] and then estimating $\mathrm{dist}(A, B)$ as in (10)
- alignment-free measures D_2, d_2, d_2^*, D_2^*, their quality-based versions D_2^q, d_2^q, d_2^{q*}, D_2^{q*}, and Mash distance measure (Ondov et al. 2016)
- a trivial baseline method estimating $\mathrm{dist}(A, B)$ as $\max\{|R_A|, |R_B|\}$.

Our code was single-threaded and implemented in Java with maximum of shared code.[5] In the case of all d-type measures, we used implementation provided with paper (Comin and Schimd 2016). Mash measure was implemented by the authors of paper (Ondov et al. 2016). All the five assembly algorithms were configured with the default parameters and the current official version was used. When a result of an assembly procedure consists of multiple contigs, we select two contigs $\hat{A}$, $\hat{B}$ that are the most similar in terms of the post-normalized Levenshtein distance $\mathrm{dist}(\hat{A}, \hat{B})/\max\{|\hat{A}|, |\hat{B}|\}$. In other words, their distance is supposed to be small and at the same time their length as large as possible. The length of the selected contigs depends on other factors than the true sequence lengths. Therefore a scaling based on (1) needs to be done. We divide by the maximum of contig lengths and consequently multiply by sequence length estimates originated from (3), which gives

$$\mathrm{dist}(A, B) \approx \frac{\mathrm{dist}(\hat{A}, \hat{B})}{\max\{|\hat{A}|, |\hat{B}|\}} \cdot \frac{l}{\alpha} \max\{|R_A|, |R_B|\}, \tag{10}$$

where $\hat{A}$, $\hat{B}$ minimize $\mathrm{dist}(\hat{A}, \hat{B})/\max\{|\hat{A}|, |\hat{B}|\}$ over all contigs produced by assembly of R_A and R_B respectively.

6.1 Evaluation criteria

The evaluation criteria consist of

- the Pearson's correlation coefficient measuring the similarity of the distance matrix produced by the respective method to the true distance matrix;
- the Fowlkes–Mallows index (Fowlkes and Mallows 1983) measuring the similarity between the hierarchical clustering tree built using the true distance matrix, and the tree induced from a distance matrix estimated by the respective method;
- runtime needed for assembly (when applicable), distance-matrix calculation, and clustering time.

For hierarchical clustering, we used the UPGMA algorithm (Sokal and Michener 1958) and the neighbor-joining algorithm (Saitou and Nei 1987). The Fowlkes–Mallows index shows how much the resulting trees differ in structure. Both trees are first cut into k clusters for $k = 2, 3, \ldots, n-1$. Then the clusterings are compared based on the number of common objects among each pair of clusters. By this procedure we obtain a set of values B_k that shows how much the trees differ at various levels.

[5] Implementation and more detailed experimental results are available on https://github.com/petrrysavy/readsDAMI2017.

6.2 Testing data

The testing data contains four datasets. They are summarized in Table 1. The "influenza" dataset[6] contains 12 influenza virus genome sequences plus an outgroup sequence. The "various" dataset[7] contains 17 genomes of different viruses.

Those two datasets contain artificial reads sampled under the assumption that reads are i.i.d. and that α and l are constant. We sampled the datasets several times with an extensive range of coverage and read length values.[8] For each choice of α and l, and for each sequence A, we selected i.i.d. with replacement $|R_A|$ reads from the set of $|A| - l + 1$ possible reads. For averaging, we left out three most outlying values of α and l.

The "hepatitis" dataset contains 81 hepatitis A segments. This time we used the ART (Huang et al. 2012) program to simulate sequencing to obtain read data for each choice of $(\alpha, l) \in \{10, 30, 50\} \times \{30, 70, 100\}$. When using the methods proposed in this paper, we down-sampled to coverage $\alpha = 2$.

The "chromosomes" dataset contains 23 regions of the human genome. For each chromosome, we selected one 20 kbp long region and obtained reads that were sequenced from this region. We used the Ensembl (Hubbard et al. 2002) reference human genome to calculate the reference distance matrix. In this dataset, the overall average coverage was 4.32. However, for each chromosome the coverage was different, which breaks our assumptions. Therefore, methods MESS, MESSG, MESSGM, MESSGq, MESSGMq were provided per-dataset coverage in formula (6). In other words, the multiplication factor in (6) was $\max\{|R_A|/\alpha_A, |R_B|/\alpha_B\}$, where α_A (α_B respectively) is coverage for read set R_A (R_B). The same holds for (10).

The sequences in "influenza", "various" and "hepatitis" datasets were downloaded from the ENA repository[9] (Leinonen et al. 2011). The human chromosomes data were obtained from the *1000 Genomes Project* (2015).

6.3 Procedure

A time limit was applied on assembly step and distance matrix calculation. Whenever an algorithm did not finish in time, assembly step failed to produce any contig, or all distances were equal, we marked the attempt as unsuccessful and did not count it towards the average.

[6] AF389115, AF389119, AY260942, AY260945, AY260949, AY260955, CY011131, CY011135, CY011143, HE584750, J02147, K00423 and outgroup AM050555. The genomes are available at http://www.ebi.ac.uk/ena/data/view/<accession>.

[7] AB073912, X98292, AM050555, D13784, EU376394, FJ560719, GU076451, JN680353, JN998607, M14707, U06714, U46935, U66304, U81989, X05817, Y13051 and outgroup AY884005.

[8] $(\alpha, l) \in \{0.1, 0.3, 0.5, 0.7, 1, 1.5, 2, 2.5, 3, 4, 5, 7, 10, 15, 20, 30, 40, 50, 70, 100\} \times \{3, 5, 10, 15, 20, 25, 30, 40, 50, 70, 100, 150, 200, 500\}$.

[9] http://www.ebi.ac.uk/ena.

Table 1 Overview of the datasets used in the experiments

Name	Source	Read generation	Strand known	$5'$ *to* $3'$ known	n	α	l	Time-limit
Influenza	ENA	i.i.d., uniform distr.	✗	✗	13	0.1–100	3–500	2 h
Various	ENA	i.i.d., uniform distr.	✗	✗	17	0.1–100	3–500	2 h
Hepatitis	ENA	Huang et al. (2012)	✗	✓	81	10, 30, 50	30, 70, 100	1 day
Chromosomes	1000 Genomes Project Consortium et al. (2015)	Real-world	✗	✓	23	4.32	76	1 day
E. coli	SRA NCBI	Real-world	✗	✓	15	Around 100	200	–

i.i.d. independent and identically distributed

6.4 Second real-world dataset

Outside of the previous scheme we used a fifth dataset in the experiments. In paper (Yi and Jin 2013), the authors tested their algorithm on "Escherichia coli" dataset. We used this dataset in our experiments; however, due to the assumption of constant l, we selected from the original 29 read-sets 15 that had $l = 200$. Those read-sets are available on NCBI SRA archive (Wheeler et al. 2008).[10]

As complete sequences for those data are not available, the reference cannot be calculated. We run MESSG(M)qα, co-phylog, d-type measures, and Mash methods and compare pairwise correlations between the distance matrices and runtime. Because of the genome sizes, we tested only method MESSG(M)qα ($\alpha = 2$) out of the methods proposed in this paper. Exact evaluation of (4) would be too slow on bacteria DNA without improvements from Sects. 4.2 and 4.3. Only on "E. coli" dataset, parallelization was used. The evaluation was done on 32-cores CPU.

6.5 Results

The main experimental results are shown in Table 2. The four partitions of the table correspond to the average results on the four datasets. The Pearson's correlation coefficient (column 'corr.') demonstrates that the most developed versions of our approach (MESSG and MESSGM) are competitive with the other approaches. This finding is generally supported by the Fowlkes–Mallows index (last four columns) shown for two levels of trees learned by two methods. Figure 6 provides a more detailed insight into the Fowlkes–Mallows values graphically for all the tree levels.

Figure 5 analyses the accuracy of different algorithms in dependence on coverage. We see that our approaches produce good results for coverage around 2. Therefore, on the "hepatitis" dataset, we sampled down to $\alpha = 2$. The assembly algorithms, however, require coverage $\alpha = 10$ and more to produce results of the same accuracy. Figure 7 shows that our method produces good estimates for shorter reads than the assembly methods. For example, method MESSG reaches the same correlation for read length $l = 20$, as the best assembly algorithm for read length $l = 500$ on the "influenza" dataset.

The overall correlation in Table 2 needs to be studied together with the number of successful attempts. For example, our methods have lower average correlation than SPADES algorithm on the "various" dataset. The overall average correlation of our method is small, as it finishes even on low coverage/read length datasets, while SPADES finished successfully only in 34 cases out of 112. If we compare with Velvet, the algorithm that finished in a similar number of cases, our methods give approximately 0.03 higher correlation.

Columns 4-5 of Table 2 indicate that the exact variants (MES, MESS, MESSG, MESSGM) of our approach were systematically slower in terms of absolute runtime than the approaches based on sequence assembly, despite the NP-hard complexity of the lat-

[10] Accessions of the used read-sets are SRX036766, SRX036767, SRX036766, SRX036767, SRX036772, SRX036774, SRX036775, SRX036942, SRX036776, SRX036777, SRX036779, SRX036943, SRX036780, SRX036781, SRX036802, SRX036803, SRX036945.

Table 2 Runtime, Pearson's correlation coefficient between distance matrices and Fowlkes–Mallows index for $k = 4$ and $k = 8$

Dataset	Method	Finished	Assem. ms	Distances ms	UPGMA ms	NJ ms	Corr.	UPGMA B_4	UPGMA B_8	NJ B_4	NJ B_8
Influenza	Reference	112/112	0	3991	4.59	3.25	1	1	1	1	1
	$\max(\lvert R_A\rvert, \lvert R_B\rvert)$	112/112	0	337	1.08	3.25	.801	.67	.319	.658	.319
	$\text{Dist}_{\text{MESS}}$	112/112	0	829,411	0.24	0.26	.945	1	.866	1	.84
	$\text{Dist}_{\text{MESSG}}$	104/112	0	986,757	0.13	0.36	.981	.995	1	.998	.993
	$\text{Dist}_{\text{MESSGq}}$	112/112	0	49,260	0.09	0.53	.971	.999	.992	.999	.985
	Mash	112/112	0	117	1.53	8.59	.679	.476	.575	.438	.61
	d_2^*	111/112	0	352	4.86	3.36	.837	.378	.712	.403	.898
	SPAdes	43/112	12,230	4644	0.33	1.07	.928	.965	.752	.94	.781
Various	Reference	112/112	0	59,602	5.21	3.40	1	1	1	1	1
	$\max(\lvert R_A\rvert, \lvert R_B\rvert)$	112/112	0	596	1.95	2.35	.907	.671	.655	.846	.924
	$\text{Dist}_{\text{MESS}}$	76/112	0	1,302,199	0.36	0.53	.93	.627	.804	.873	.933
	$\text{Dist}_{\text{MESSG}}$	70/112	0	1,575,721	0.29	0.64	.933	.621	.884	.932	.93
	$\text{Dist}_{\text{MESSMq}}$	110/112	0	570,361	0.29	0.79	.927	.657	.771	.842	.972
	Mash	112/112	0	238	4.88	11.26	.498	.408	.267	.428	.326
	d_2^*	109/112	0	689	4.84	19.32	.442	.378	.189	.453	.317
	SPAdes	34/112	18,675	177,821	0.21	0.79	.942	.698	.91	.961	.949

Table 2 continued

Dataset	Method	Finished	Assem. ms	Distances ms	UPGMA ms	NJ ms	Corr.	UPGMA B_4	UPGMA B_8	NJ B_4	NJ B_8
Hepatitis	Reference	9/9	0	1,759,470	25.00	44.44	1	1	1	1	1
	$\max(\lvert R_A\rvert, \lvert R_B\rvert)$	9/9	0	18,913	7.11	14.00	.181	.553	.368	.724	.828
	Dist_{MES}	9/9	0	10,994,207	1.11	3.56	.833	1	.952	1	.961
	$\text{Dist}_{\text{MESSGM}}$	9/9	0	20,489,458	4.78	3.78	.965	.994	.946	1	.903
	$\text{Dist}_{\text{MESSGMq}}$	9/9	0	697,464	1.56	5.78	.9	.915	.947	1	.944
	Mash	9/9	0	3788	23.00	141.33	.967	.964	.966	1	.918
	d_2^q	9/9	0	26,301	47.11	397.00	.973	.984	.96	1	.87
	Velvet	9/9	17,774	2,398,724	1.00	3.67	.782	.803	.846	.964	.847
Chromosomes	Reference	1/1	0	653,909	7.00	4.00	1	1	1	1	1
	$\max(\lvert R_A\rvert, \lvert R_B\rvert)$	1/1	0	1247	1.00	1.00	.331	.64	.404	.613	.298
	Dist_{MES}	1/1	0	10,645,321	1.00	0.00	.886	.42	.263	.596	.276
	$\text{Dist}_{\text{MESSG}\alpha}$	1/1	0	20,713,067	1.00	1.00	.848	.408	.227	.585	.26
	$\text{Dist}_{\text{MESSGq}\alpha}$	1/1	0	178,840	1.00	1.00	.841	.673	.301	.9	.262
	Mash	1/1	0	261	1.00	4.00	.33	.588	.307	.599	.382
	d_2^*	1/1	0	1768	0.00	2.00	.302	.503	.328	.805	.303
	SSAKEα	1/1	46,853	55,131	1.00	1.00	.652	.528	.17	.805	.255

The 'reference' method calculates distances from the original sequences. We show only assembly algorithm that gave the highest correlation, the best d-type method, and the better algorithm of pairs MES/MESS, MESSG/MESSGM, and MESSGq/MESSGMq

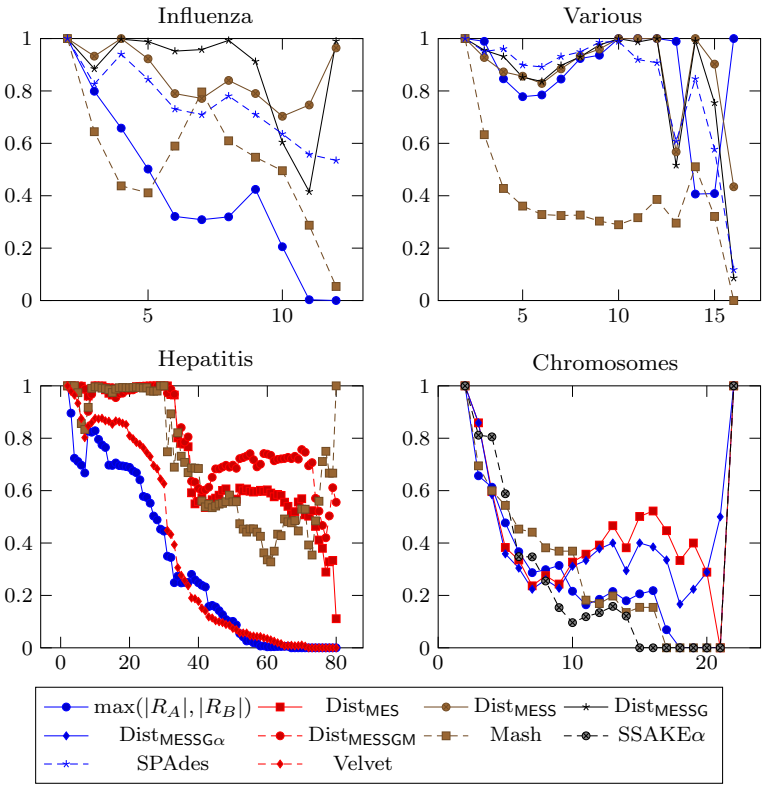


Fig. 6 Plots of Fowlkes–Mallows index B_k versus k. The index compares trees generated by the neighbor-joining algorithm. The tree is compared with the tree generated from the original sequences. If all values are equal to 1, the structures of the trees are the same

ter task. However, the approximated versions (MESSGq, MESSGMq, including sampling) of our approach did produce results in time comparable to the assembly time. The cost for this runtime improvement was only a small decrease of accuracy. Moreover, the approximated versions were faster than calculating the reference distance matrix on larger datasets. This makes methods MESSGq and MESSGMq suitable for real-world situations. The numbers also show that our asymptotic complexity estimate in Sect. 5.1 is generally correct: the ratio between the time spent on calculating the distances on one hand, and the runtime of the reference method on the other hand, is approximately α^2.

The "chromosomes" dataset shows how our method is dependent on the assumption that estimates of the original sequences lengths are good. While method MES provides good estimates, the accuracy drops after applying the scaling from Sect. 3.2. The coverage is not constant on all read bags and therefore the results are poor knowing that all the original sequences had the same length. Therefore we provide results with per-sample coverage known to the algorithms. This fact is marked by α in the experimental results.

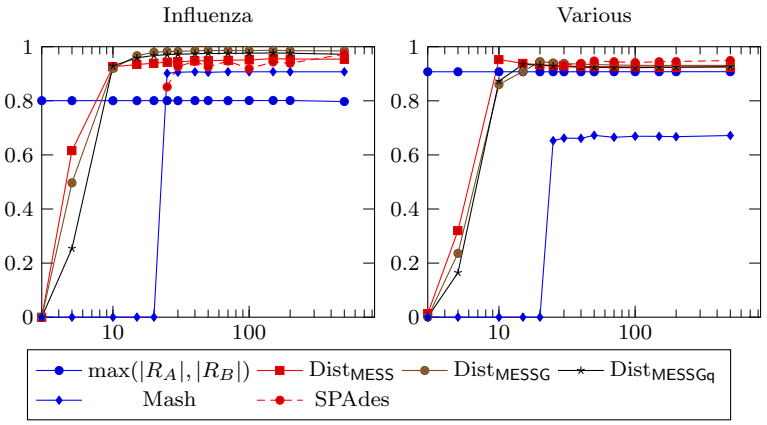


Fig. 7 Plot of average Pearson's correlation coefficient for several choices of read length

Table 3 Runtime on "E. coli" dataset

Method	Time (in s)
$\text{Dist}_{\text{MESSG(M)q}\alpha}$	11073
co-phylog	583
Mash	480
d_2	3221
d_2^*	3235
d_2^q	3228
d_2^{q*}	3225
D_2	3235
D_2^*	3301
D_2^q	3224
D_2^{q*}	3227

Assembly time (without distance matrix calculation) on the same dataset is 18,844 s (ABySS), 18,606 s (Edena), 33,545 s (SPAdes), 298,564 s (SSAKE), and 17,701 s (Velvet)

Our approaches outperform the alignment-free measures concerning correlation on "influenza", "various" and "chromosomes" datasets. The alignment-free measures are approximately 1000 times faster. This speed is paid by the fact that they have to break reads into shorter k-mers, and as a result, they lose some information. Due to this speed, they are however able to use all reads in the "hepatitis" dataset, while our methods have to sample to coverage $\alpha = 2$.

The results on "E. coli" dataset are presented in Tables 3 and 4. Concerning the runtime, the proposed method with all technical improvements is 4 to 23 times slower than alignment-free approaches. On the other hand, the runtime is still 30% less compared to the sequence assembly itself without any distance matrix calculation. The Pearson's correlation coefficient between the respective methods in Table 4 shows that the proposed method does not deviate from the other approaches.

Table 4 Pairwise correlations of distance matrices on "E. coli" dataset

	$\text{Dist}_{\text{MESSG(M)q}\alpha}$	co-phylog	Mash	d_2	d_2^*	d_2^q	d_2^{q*}	D_2	D_2^*	D_2^q	D_2^{q*}
$\text{Dist}_{\text{MESSG(M)q}\alpha}$	1.000	0.829	0.942	0.269	0.302	0.269	0.304	0.683	0.866	0.684	0.868
co-phylog	0.829	1.000	0.925	0.161	0.215	0.161	0.216	0.711	0.813	0.712	0.813
Mash	0.942	0.925	1.000	0.286	0.339	0.286	0.341	0.734	0.877	0.735	0.878
d_2	0.269	0.161	0.286	1.000	0.956	1.000	0.956	0.240	0.246	0.243	0.249
d_2^*	0.302	0.215	0.339	0.956	1.000	0.956	1.000	0.322	0.303	0.325	0.307
d_2^q	0.269	0.161	0.286	1.000	0.956	1.000	0.956	0.240	0.246	0.243	0.249
d_2^{q*}	0.304	0.216	0.341	0.956	1.000	0.956	1.000	0.323	0.305	0.326	0.308
D_2	0.683	0.711	0.734	0.240	0.322	0.240	0.323	1.000	0.949	1.000	0.949
D_2^*	0.866	0.813	0.877	0.246	0.303	0.246	0.305	0.949	1.000	0.950	1.000
D_2^q	0.684	0.712	0.735	0.243	0.325	0.243	0.326	1.000	0.950	1.000	0.949
D_2^{q*}	0.868	0.813	0.878	0.249	0.307	0.249	0.308	0.949	1.000	0.949	1.000

7 Conclusions and future work

We have proposed and evaluated several variants of a method for estimating edit distance between sequences, given only read sets sampled from these sequences. In empirical experiments, our approach produced estimates better than a conventional approach, in which the sequences are first estimated from the read sets using assembly algorithms, and the distances are then computed from these estimates.

Specifically, the experiments have demonstrated that the approach based on conventional assembly algorithms requires a higher coverage α as well as read-length l to produce estimates comparable to our method. Thus the latter brings an obvious and tangible benefit of reducing the cost of wet-lab sequencing procedures, if the experimental aim is to determine mutual sequence distances as in the task of genome clustering.

Our approach offers directions for further improvements. For example, one may consider a *partial assembly* approach, in which sets of a few (up to a constant) reads would be pre-assembled and the Monge-Elkan distance would be applied on such partial assemblies. This view would open a 'continuous' spectrum between our approach on one hand, and the conventional assembly-based approach, on which the optimal trade-off could be identified.

Acknowledgements The authors acknowledge the support of the OP VVV project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics". Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

References

1000 Genomes Project Consortium et al. (2015) A global reference for human genetic variation. Nature 526(7571):68–74

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. J Mol Biol 215(3):403–410

Bao E, Jiang T, Kaloshian I, Girke T (2011) SEED: efficient clustering of next-generation sequences. Bioinformatics 27(18):2502–2509

Blaisdell BE (1986) A measure of the similarity of sets of sequences not requiring sequence alignment. Proc Natl Acad Sci 83(14):5155–5159

Comin M, Leoni A, Schimd M (2015) Clustering of reads with alignment-free measures and quality values. Algorithms Mol Biol 10(1):4

Comin M, Schimd M (2014) Assembly-free genome comparison based on next-generation sequencing reads and variable length patterns. BMC Bioinformatics 15(9):S1

Comin M, Schimd M (2016) Fast comparison of genomic and meta-genomic reads with alignment-free measures based on quality values. BMC Med Genomics 9(1):36

Fowlkes EB, Mallows CL (1983) A method for comparing two hierarchical clusterings. J Am Stat Assoc 78(383):553–569

Goodwin S, Mcpherson J, Richard Mccombie W (2016) Coming of age: ten years of next-generation sequencing technologies. Nat Rev Genet 17:333–351 05

Haiminen N, Kuhn DN, Parida L, Rigoutsos I (2011) Evaluation of methods for de novo genome assembly from high-throughput sequencing reads reveals dependencies that affect the quality of the results. PLOS ONE 6(9):1–9 09

Hernandez D, Franois P, Farinelli L, sters M, Schrenzel J (2008) De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. Genome Res 18(5):802–809

Huang W, Li L, Myers JR, Marth GT (2012) ART: a next-generation sequencing read simulator. Bioinformatics 28(4):593–594

Hubbard T, Barker D, Birney E, Cameron G, Chen Y et al (2002) The Ensembl genome database project. Nucl Acids Res 30(1):38–41

Jalovec K, Železný F (2014) Binary classification of metagenomic samples using discriminative DNA superstrings. In: MLSB 2014: 8th International workshop on machine learning in systems biology, pp 44–47

Kchouk M, Elloumi M(2016) A clustering approach for denovo assembly using next generation sequencing data. In: 2016 IEEE international conference on bioinformatics and biomedicine (BIBM), IEEE, pp 1909–1911

Lander ES, Linton LM, Birren B, Nusbaum C, Zody MC, Baldwin J, Devon K, Dewar K, Doyle M, FitzHugh W et al (2001) Initial sequencing and analysis of the human genome. Nature 409(6822):860–921

Leinonen R, Akhtar R, Birney E, Bower L, Cerdeno-Trraga A, Cheng Y, Cleland I, Faruque N, Goodgame N, Gibson R, Hoad G, Jang M, Pakseresht N, Plaister S, Radhakrishnan R, Reddy K, Sobhany S, Ten Hoopen P, Vaughan R, Zalunin V, Cochrane G (2011) The European Nucleotide Archive. Nucl Acids Res 39(suppl–1):D28–D31

Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions, and reversals. Sov Phys Dokl 10(8):707

Malhotra R, Elleder D, Bao L, Hunter DR, Acharya R, Poss M (2014) Clustering pipeline for determining consensus sequences in targeted next-generation sequencing. ArXiv preprint

Monge AE, Elkan CP (1996) The field matching problem: algorithms and applications. In: Proceedings of the second international conference on knowledge discovery and data mining, KDD'96, AAAI Press, pp 267–270

Navarro G (2001) A guided tour to approximate string matching. ACM Comput Surv 33(1):31–88

Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48(3):443–453

Nurk Sergey, Bankevich Anton, et al (2013) Assembling genomes and mini-metagenomes from highly chimeric reads. In: Deng M, Jiang R, Sun F, Zhang X, (eds) 17th Annual international conference on research in computational molecular biology, RECOMB 2013, Beijing, China, April 7–10, 2013. Proceedings, Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 158–170

Ondov BD, Treangen TJ, Melsted P, Mallonee AB, Bergman NH, Koren S, Phillippy AM (2016) Mash: fast genome and metagenome distance estimation using MinHash. Genome Biol 17(1):132

Reinert G, Chew D, Sun F, Waterman MS (2009) Alignment-free sequence comparison (I): statistics and power. J Comput Biol 16(12):1615–1634

Ryšavý Petr, Železný Filip (2016) Estimating sequence similarity from read sets for clustering sequencing data. In: Boström H, Knobbe A, Soares C, Papapetrou P (eds) 15th International symposium on advances in intelligent data analysis XV, IDA 2016, Stockholm, Sweden, October 13–15, 2016, Proceedings, Cham, Springer International Publishing, pp 204–214

Saitou N, Nei M (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol 4(4):406–425

Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJM, nan Birol (2009) ABySS: a parallel assembler for short read sequence data. Genome Res 19(6):1117–1123

Sokal RR, Michener CD (1958) A statistical method for evaluating systematic relationships. Univ Kans Sci Bull 38:1409–1438

Song K, Ren J, Zhai Z, Liu X, Deng M, Sun F (2013) Alignment-free sequence comparison based on next-generation sequencing reads. J Comput Biol 20(2):64–79

Ukkonen E (1992) Approximate string-matching with q-grams and maximal matches. Theor Comput Sci 92(1):191–211

Wagner RA, Fischer MJ (1974) The string-to-string correction problem. J Assoc Comput Mach 21(1):168–173

Warren RL, Sutton GG, Jones SJM, Holt RA (2007) Assembling millions of short DNA sequences using SSAKE. Bioinformatics 23(4):500–501

Weitschek E, Santoni D, Fiscon G, De Cola MC, Bertolazzi P, Felici G (2014) Next generation sequencing reads comparison with an alignment-free distance. BMC Res Notes 7:869

Wheeler DL, Barrett T, Benson DA, Bryant SH, Canese K, Chetvernin V, Church DM, DiCuccio M, Edgar R, Federhen S, Feolo M, Geer LY, Helmberg W, Kapustin Y, Khovayko O, Landsman D, Lipman DJ, Madden TL, Maglott DR, Miller V, Ostell J, Pruitt KD, Schuler GD, Shumway M, Sequeira E, Sherry ST, Sirotkin K, Souvorov A, Starchenko G, Tatusov RL, Tatusova TA, Wagner L, Yaschenko E (2008) Database resources of the national center for biotechnology information. Nucl Acids Res 36(suppl–1):D13–D21

Yi H, Jin L (2013) Co-phylog: an assembly-free phylogenomic approach for closely related organisms. Nucl Acids Res 41(7):e75

Zerbino DR, Birney E (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res 18(5):821–829

Železný F, Jalovec K, Tolar J (2014) Learning meets sequencing: a generality framework for read-sets. In: ILP 2014: 24th Internation conference on inductive logic programming, Late-Breaking Papers