Bioinformatics, 31(8), 2015, 1169–1175
doi: 10.1093/bioinformatics/btu815
Advance Access Publication Date: 10 December 2014
Original Paper

Genome analysis

andi: Fast and accurate estimation of evolutionary distances between closely related genomes

Bernhard Haubold[1,*], Fabian Klötzl[1,2] and Peter Pfaffelhuber[3]

[1]Department of Evolutionary Genetics, Max-Planck-Institute for Evolutionary Biology, 24306 Plön, Germany, [2]Institue for Neuro- and Bioinformatics, Lübeck University, 23562 Lübeck, Germany and [3]Mathematical Stochastics, Mathematical Institute, Freiburg University, Germany

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on July 1, 2014; revised on November 6, 2014; accepted on December 7, 2014

Abstract

Motivation: A standard approach to classifying sets of genomes is to calculate their pairwise distances. This is difficult for large samples. We have therefore developed an algorithm for rapidly computing the evolutionary distances between closely related genomes.

Results: Our distance measure is based on ungapped local alignments that we anchor through pairs of maximal unique matches of a minimum length. These exact matches can be looked up efficiently using enhanced suffix arrays and our implementation requires approximately only 1 s and 45 MB RAM/Mbase analysed. The pairing of matches distinguishes non-homologous from homologous regions leading to accurate distance estimation. We show this by analysing simulated data and genome samples ranging from 29 *Escherichia coli*/*Shigella* genomes to 3085 genomes of *Streptococcus pneumoniae*.

Availability and implementation: We have implemented the computation of anchor distances in the multithreaded UNIX command-line program andi for ANchor DIstances. C sources and documentation are posted at http://github.com/evolbioinf/andi/

Contact: haubold@evolbio.mpg.de

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The spread of infectious diseases is nowadays often monitored by sequencing the genomes of the outbreak strains. Since a given pandemic is usually caused by the rapid expansion of a single clone, monitoring by sequencing leads to the accumulation of hundreds to thousands of similar genome sequences. For example, Petty *et al.* (2014) studied the spread of the multi-drug resistant *Escherichia coli* strain ST131, which causes extra-intestinal infections in humans. The authors sequenced 99 outbreak strains and reconstructed their phylogeny. This revealed that the outbreak was caused by a single lineage of ST131. On an even larger scale, Chewapreecha *et al.* (2014) studied pneumococcal carriage in a refugee camp by sequencing 3085 strains of *Streptococcus pneumoniae*, which causes pneumoniae in humans. Again, phylogeny reconstruction based on these genomes was an early step in the study.

Classifying bacteria by clustering their genomes is set to become routine. For this purpose, Petty *et al.* (2014) computed a multiple sequence alignment of their *E.coli* ST131 strains using the program mugsy (Angiuoli and Salzberg, 2011). It is based on the MUMmer software (Kurtz *et al.*, 2004), which makes mugsy one of the fastest multiple genome aligners available: it took only 19 h to align 57 complete *E.coli* genomes. However, the run time of mugsy becomes unacceptable when applied to the recently collected samples of hundreds or even thousands of bacterial genomes.

For their study of 3085 pneumococcal isolates, Chewapreecha *et al.* (2014) mapped the sequencing reads onto a reference genome, thereby approximating a multiple sequence alignment. Such alignment by mapping is widely used, and Bertels *et al.* (2014) have shown that its accuracy can be further improved by mapping against multiple reference genomes instead of the customary single reference. Their implementation of this idea, `REALPHY`, requires 2 min and 250 MB/Mbase analysed. The genome of *S.pneumoniae* is 2.2 Mbase long, so `REALPHY` would run 9.4 days on the 3085 *S.pneumoniae* isolates. However, a more prohibitive aspect of `REALPHY` might be the RAM requirement of 1.7 TB.

Perhaps surprisingly, it is not necessary to compute an explicit alignment for phylogeny reconstruction. This insight has sparked interest in devising alignment-free methods for rapidly calculating pairwise distances between genomes (Haubold, 2014), which can then be clustered using various quick algorithms (Felsenstein, 2004).

Alignment-free distance computation is either based on counting words of a certain length or recording match lengths (Haubold, 2014). When counting words, there is a choice between the traditional approach of counting exact words and a more recent method of looking for words that enclose one mismatch. The latter is implemented in the program `co-phylog` (Yi and Jin, 2013), which gives better distances than exact word counting while requiring only moderate additional resources (Haubold, 2014).

Haubold *et al.* (2009) devised an alignment-free estimator of genetic distance based on match lengths. The expected match length is the inverse of the proportion of mismatches. For example, if 1% of positions between two genomes are mismatched, the expected match length is 100. Domazet-Lošo and Haubold (2009) implemented this idea using a generalized suffix tree of all input sequences to look up the match lengths. Their program `kr` computes the distances between the complete genomes of 29 *E.coli/Shigella* strains in just 5.5 min on a single processor. However, this takes 5.3 GB RAM and `kr` has been criticized for excessive RAM utilization (Cohen and Chor, 2012). Moreover, Yi and Jin (2013) noted that `co-phylog` gave a better phylogeny when applied to the benchmarking sample of 29 *E.coli/Shigella* genomes.

This has prompted us to devise a replacement for `kr`. Our new strategy is to look for mismatches that are bracketed by long exact matches, which we call *anchors*. We show through simulation that the resulting program `andi`, for ANchor DIstances, is accurate, fast and memory efficient. Moreover, we apply `andi` to three sets of bacterial genomes: the 29 genomes of *E.coli/Shigella* Yi and Jin (2013) used for benchmarking `co-phylog`, the 109 genomes of *E.coli* ST131 studied by Petty *et al.* (2014)—of which they sequenced 99 strains—and the 3085 genomes of *S.pneumoniae* sequenced by Chewapreecha *et al.* (2014). In each case, `andi` quickly recovers pairwise distances. For the *E.coli* samples, we compare the new distances to alignment-based distances and find they are so similar that they yield almost identical phylogenies.

2 Methods

2.1 Defining anchor distances

We compute anchor distances using maximal matches by imposing three criteria on them: uniqueness, equidistance and a minimum length. In this section, we explain each of these criteria in turn, which sets the stage for the description of our algorithm in Section 2.2.

Consider two DNA sequences, a query Q and a subject S. Taking our cue from genome alignment tools such as `MUMmer` (Kurtz *et al.*, 2004) and `mauve` (Darling *et al.*, 2004), we call a unique maximal match between Q and S an *anchor*, if it has some minimum length. We look for pairs of anchors that have the same distance in Q and S, as shown in Figure 1A. Such anchor pairs approximate ungapped alignments, and we count the mismatches in the intervening segment. In contrast, if the anchors are unequally spaced as shown in Figure 1B, the regions they bracket are either not homologous or contain indels. We ignore such anchor pairs in our analysis. The total number of mismatches bracketed by equidistant anchors divided by the number of nucleotides covered by the anchors and the bracketed regions is our estimate of the number of mismatches per site, $d_m(Q,S)$. This is converted to the number of substitutions per site using the correction by Jukes and Cantor (1969):

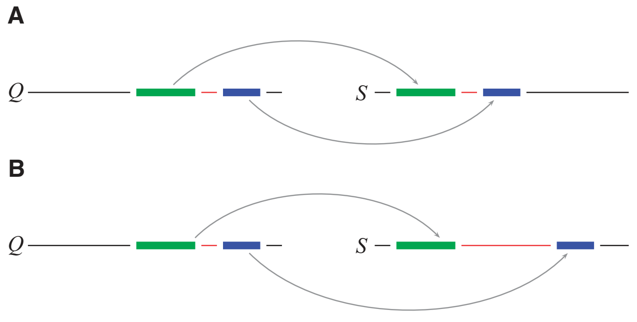


Fig. 1. Exact matches of a given minimum length (anchors) between query (*Q*) and subject (*S*) sequences. **A**: Equally spaced anchors; here we count the mismatches in the intervening segment. **B**: Unequally spaced anchors; these are ignored in the distance computation. (Color version of this figure is available at *Bioinformatics* online.)

$$K(Q,S) = -\frac{3}{4}\ln\left(1 - \frac{4}{3}d_m(Q,S)\right).$$

$K(Q, S)$ is not symmetrical, that is, $K(Q,S) \neq K(S,Q)$. We therefore define the anchor distance between two sequences i and j as the average Jukes–Cantor distance computed from the two possible labellings of i and j:

$$d_a(i,j) = \frac{K(Q,S) + K(S,Q)}{2}.$$

A critical parameter in the computation of $d_a(i,j)$ is the minimum anchor length, l. We compute this as a function of GC content and subject length using equation (6) by Haubold *et al.* (2009):

$$\mathbb{P}\{X_i^* \le x\} = \sum_{k=0}^{x} 2^x \binom{x}{k} p^k \left(\tfrac{1}{2} - p\right)^{x-k} \left(1 - p^k \left(\tfrac{1}{2} - p\right)^{x-k}\right)^{|S|},$$

where X_i^* is the length of a match starting at position i in Q and any position in S, and $2p$ is the GC content of S. We define l to be the 97.5% quantile of the distribution of X_i^*.

2.2 Algorithm and implementation

For computing anchor distances, we first order alphabetically all suffixes contained in the forward and reverse strands of S. From this suffix array, we compute the array of common prefix lengths between consecutive suffixes using the Φ-algorithm listed as Algorithm 4.4 by Ohlebusch (2013). Together with the suffix array, this longest common prefix array forms the enhanced suffix array, E, which is the central input for computing $K(Q, S)$. Algorithm 1 uses the function `getMatch(E, string)` to look for the longest prefix of string that matches somewhere in *S*. In Line 8, string is the suffix of *Q* that has not yet been matched against *S*. Function getMatch (not shown) is a slight variation on Algorithm 5.2 by Ohlebusch (2013). The matching step is repeated one residue beyond the mismatched nucleotide that terminates the previous match until an equidistant pair of anchors (Fig. 1A) is found in Line 12. The mutations in the intervening segment are counted (Line 13) and the distance between the current and the previous match is added to the homologous nucleotide counter (Line 14). The search for equidistant anchor pairs continues until the complete forward strand of *Q* has been traversed. Then $K(Q, S)$ is computed as the ratio of mutations to homologous nucleotides (Line 24). This streaming of *Q* against *S* using an enhanced suffix array is an idea we took from vmatch (http://www.vmatch.de).

For our suffix array computation, we use the libdivsufsort library (http://homepage3.nifty.com/wpage/software/). The function getMatch is based on range minimum queries, for which we use an algorithm and corresponding implementation by Fischer and Heun (2007). The memory requirement of the resulting program andi is dominated by the computation of the enhanced suffix array. To minimize the memory footprint of andi, it initially streams all sequences against the enhanced suffix array of the first sequence, then against the enhanced suffix array of the second sequence, and so on. Thus at any one time, only the enhanced suffix array for a single sequence is kept in memory. This approach also allows for easy parallelization, which we implemented using the OpenMP framework. andi runs fastest when the number of taxa is equal to the number of processors. In that situation, all rows of the distance matrix are filled in simultaneously and the program takes time proportional to the length of the longest genome.

Algorithm 1 Estimate substitutions per site

```
Require: $Q$ {query sequence}
Require: $S$ {subject sequence}
Require: $E$ {enhanced suffix array of $S$, forward & reverse}
Require: $l$ {minimum anchor length}
Ensure: $K$ {$K(Q, S)$, substitutions per site between $Q$ and $S$}
 1: $q_p \leftarrow q_c \leftarrow 0$ {previous and current position in $Q$}
 2: $s_p \leftarrow 0$ {previous position in $S$}
 3: $l_p \leftarrow 0$ {previous jump length}
 4: $s \leftarrow 0$ {number of mutations (segregating sites)}
 5: $n \leftarrow 0$ {number of homologous nucleotides}
 6: $a \leftarrow$ false {no anchor found yet}
 7: while $q_c < |Q|$ do
 8:     m ← getMatch(E, Q[$q_c$..|Q|])
 9:     $l_c \leftarrow$ m.length + 1 {jump by at least one position}
10:     if m.isUnique and m.length $\geq l$ then
11:         $s_c \leftarrow E$.position($m$) {find position of match in $S$}
12:         if $q_c - q_p = s_c - s_p$ then
13:             $s \leftarrow s +$ countDiff($Q[q_p..q_c - 1], S[s_p..s_c - 1]$)
14:             $n \leftarrow n + q_c - q_p$
15:             $a \leftarrow$ true
16:         else
17:             if $a =$ true then
18:                 $n \leftarrow n + l_p - 1$
19:             $a \leftarrow$ false
20:         $q_p \leftarrow q_c$
21:         $s_p \leftarrow s_c$
22:         $l_p \leftarrow l_c$
23:     $q_c \leftarrow q_c + l_c$
24: $K \leftarrow s/n$
```

2.3 Simulations

For simulating pairs of related DNA sequences, we used our program simK, which is linked from the andi web page. Here is the command for a typical simulation run

```
simK -l 1000000 -k 0.01 | andi
```

where -l is the sequence length and -k is the number of substitutions per site. Time and memory consumption was measured using commands such as

```
/usr/bin/time -f "\n %E elapsed, \n%M memory" \
andi sim.fa > sim.dist 2> andi.res
```

on a 32 core 2.3 GHz AMD Opteron system with 256 GB of RAM.

2.4 Datasets

Apart from simulated data, we analysed three sets of genomes of increasing size:

1. Twenty-nine *E.coli/Shigella* genomes used by Yi and Jin (2013) for benchmarking, average length 4.9 Mbase;
2. One hundred and nine *E.coli* ST131 genomes, average length 5.2 Mbase (Petty *et al.*, 2014);
3. Three thousand and eighty-five *S.pneumoniae* genomes, average length 2.2 Mbase (Chewapreecha *et al.*, 2014).

Links to these datasets are also posted on the andi web site.

2.5 Alignment

The two *E.coli* genome samples were aligned with mugsy, which generates output in 'mutation annotation format' (maf) (Angiuoli and Salzberg, 2011). We converted this to PHYLIP format with our script maf2phy.awk, also posted on the andi web site. Jukes–Cantor distances were computed using the program dnadist, which is part of the PHYLIP package (Felsenstein, 2005).

2.6 Phylogeny reconstruction

Distances were clustered using neighbor and the trees midpoint rooted using retree, both also part of PHYLIP. Trees were plotted using njplot (Perrière and Gouy, 1996) or drawgram (PHYLIP). Topological distances between trees were computed using the programs rspr (Whidden *et al.*, 2013) or treedist (PHYLIP).

3 Results

3.1 Simulations

Figure 2A shows our new distance measure d_a as a function of the number of substitutions per site, K, for simulated pairs of 100 kbase sequences, which implies a minimum anchor length of 8. Under these ideal conditions, d_a is an excellent estimator for a wide range of divergence values. However, for $K \geq 0.5$ the probability increases that no anchor pair is found and d_a cannot be computed. The proportion of failed estimations therefore grows from 0.7% for $K = 0.5$ to 94% for $K = 0.65$ (Fig. 2B, open circles). This might suggest that a lower minimum anchor length yields better estimates. However, it leads to underestimation of distances (Supplementary Fig. S1). We thus recommend using andi only for sequences with $K \leq 0.5$.

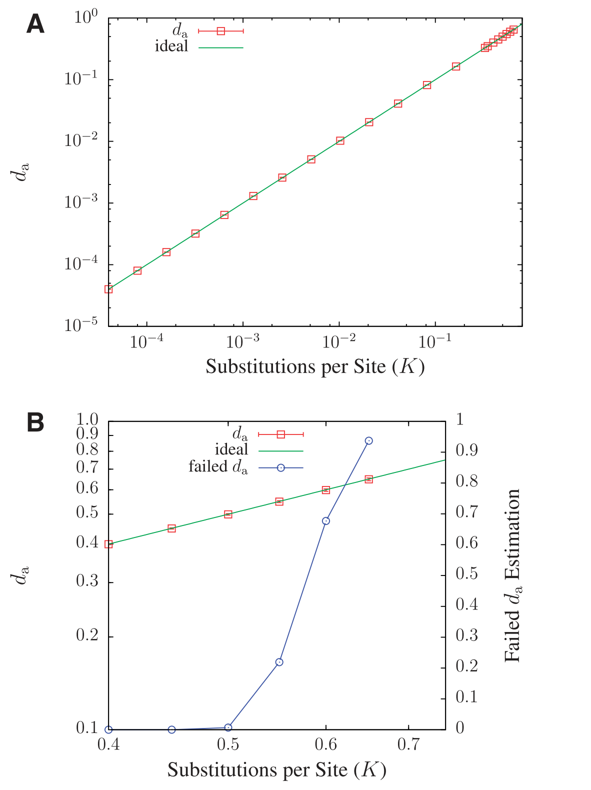


Fig. 2. Estimation of the rate of substitution on simulated pairs of 100 kbase sequences; shown are mean ± standard deviation of 1000 iterations. **A**: Substitution rates between 4×10^{-5} and 0.65. **B**: Substitution rates ≥0.4 and proportion of failed estimates. (Color version of this figure is available at *Bioinformatics* online.)

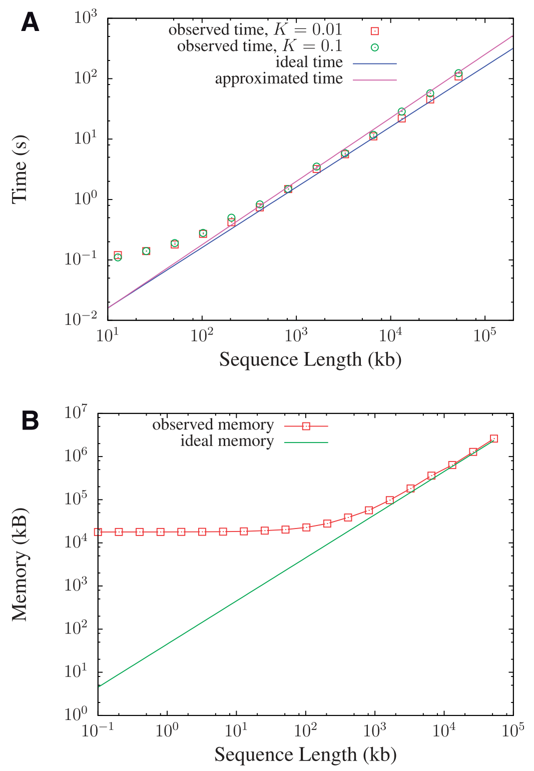


Fig. 3. Time (**A**) and memory (**B**) consumption as a function of sequence length. Single pairs of sequences were simulated with 0.01 or 0.1 substitutions per site and analysed. (Color version of this figure is available at *Bioinformatics* online.)

Our main motivation for developing d_a is efficiency. Figure 3 shows the time and memory consumption of `andi` as a function of sequence length. The run time would ideally be linear in the size of the input sequence, that is, time ∝ length, which is close to the observed run time $O(\text{length}^{1.05})$. Notice also that the more divergent sequences with $K = 0.1$ take slightly longer to analyse than those with $K = 0.01$. The reason for this is that the streaming of the query against the enhanced suffix array of the subject takes time proportional to the number of calls to the matching function. This in turn depends on the number of substitutions, with divergent sequences requiring more matching steps. Nevertheless, as a rule of thumb `andi` takes 1 s/Mbase. The memory consumption shown in Figure 3B is initially constant in the sequence length reflecting program overheads. For longer sequences, memory consumption is exactly linear in the size of the input data, as expected. In fact, we observe that 45 bytes memory are used per base pair.

We conclude from our analysis of simulated data that `andi` is accurate and efficient. Next, we apply `andi` to three samples of genomes: 29 *E.coli*/*Shigella* genomes, 109 genomes of *E.coli* ST131 (Petty *et al.*, 2014) and 3085 genomes of *S.pneumoniae* (Chewapreecha *et al.*, 2014). Where appropriate, we compare the results obtained by `andi` with those of `co-phylog` and alignment-based distances.

3.2 Application to genomes

Figure 4 shows the phylogeny of 29 *E.coli*/*Shigella* strains computed from their complete genomes. These genomes are often used for benchmarking (Bertels *et al.*, 2014; Yi and Jin, 2013) and are on average 4.9 Mbase long. Aligning them to compute the phylogeny in Figure 4A took `mugsy` 5 h, 33 min and 2.9 GB RAM. The corresponding `co-phylog` computation took only 9 min, 21 s and 156.8 MB RAM. The resulting tree in Figure 4B is shorter than the reference from which it is separated by a topological distance of 3. Two of these topological differences affect short branches in clade C. The other difference concerns the position of *E.coli* strain UMN026, which switches between the two most basal clades. With 29 threads `andi` took 19.8 s and 7.2 GB RAM to compute Figure 4C. Its branch lengths are almost indistinguishable from the reference tree and its topological distance from the reference is only one due to a difference in clade C, where strain 536 should branch off at a more basal position. However, the position of strain UMN026 is correct.

Figure 5 shows the `andi` tree constructed from 109 *E.coli* ST131 strains (Petty *et al.*, 2014) in 1 min 21 s using 30 CPUs and 7.7 GB RAM. The 99 strains sequenced by Petty *et al.* (2014) fall into three clades, A, B, and C, shown online in red, orange and green, respectively. The clades identified by `andi` are identical to those reported in the original publication based on a `mugsy` alignment computed on our hardware in 5.6 days using 52.7 GB RAM. That is, `andi` analyses the 109 *E.coli* genomes approximately 6000 times faster than `mugsy` and uses seven times less RAM.

As the third and final challenge, we applied `co-phylog` and `andi` to 3085 complete genomes of the human bacterial pathogen *S.pneumoniae*. Its genome is 2.2 Mbase long, amounting to a dataset comprising 6.7 Gbase. `co-phylog` took 36.5 days and 2.3 GB RAM to compute the pairwise distances shown in Figure 6A. With 32 threads `andi` took 7 h, 35 min and 23.8 GB RAM to carry out the same computation (Fig. 6B). Unfortunately, we cannot compare these two trees to a reference tree. Moreover, their

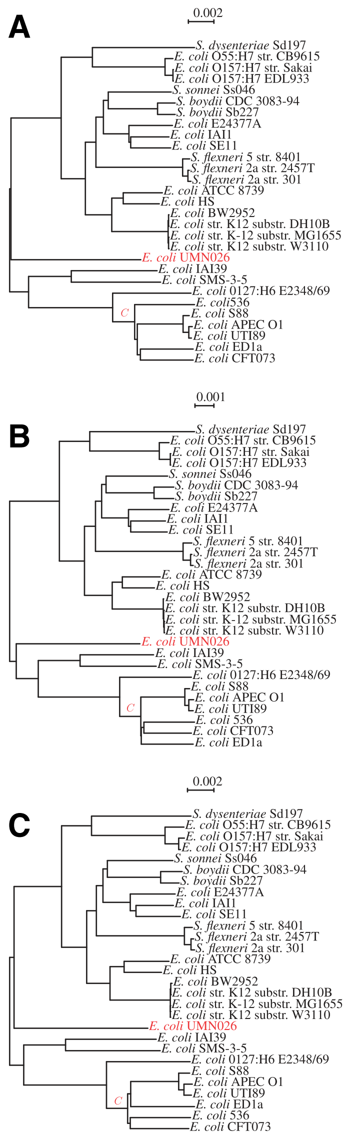


Fig. 4. Phylogenies computed from the complete genomes of 29 strains of *E.coli*/*Shigella*. **A**: Based on a `mugsy` alignment (Angiuoli and Salzberg, 2011); **B**: based on distances computed using `co-phylog` (Yi and Jin, 2013); **C**: based on d_a. Clade *C* differs between all three trees; the position of strain UMN026 differs between A and B. (Color version of this figure is available at *Bioinformatics* online.)

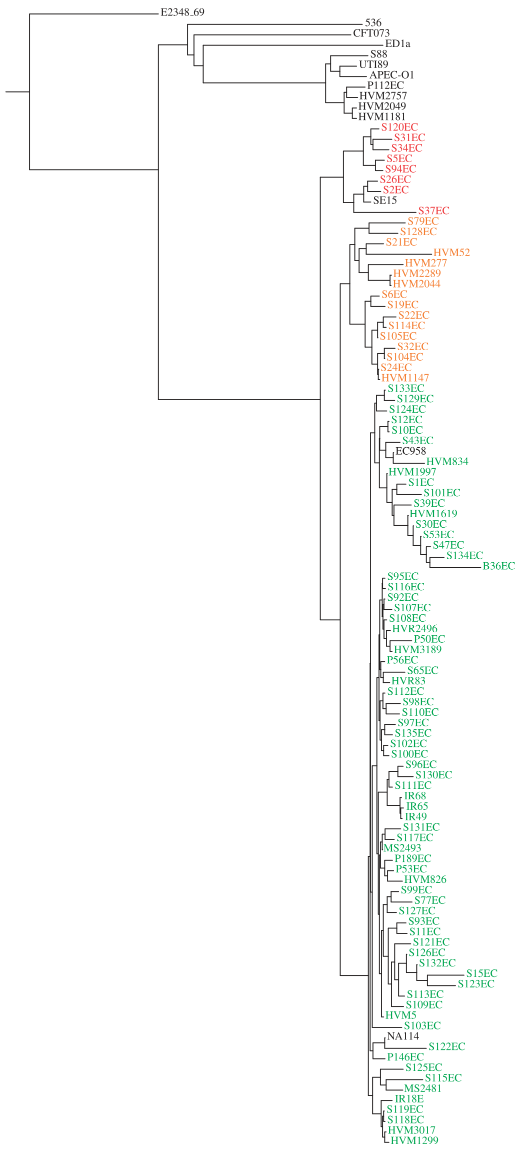


Fig. 5. The phylogeny of 109 *E.coli* ST131 strains based on d_a computed from complete genome sequences. Clades A–C colored online as defined by Petty *et al.* (2014). (Color version of this figure is available at *Bioinformatics* online.)

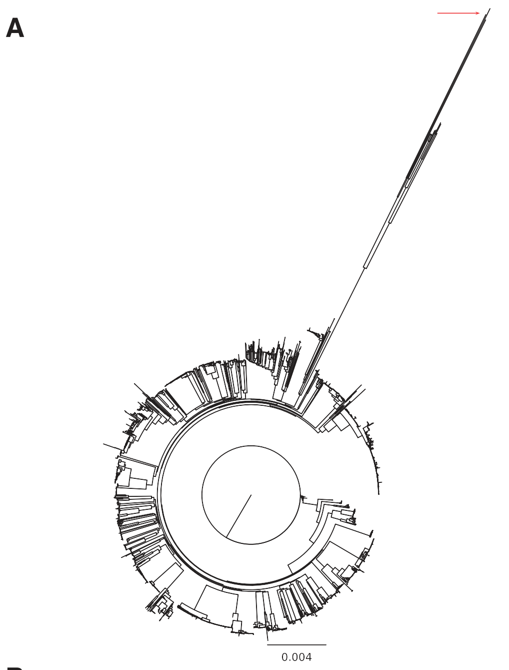
Robinson–Foulds distance (Robinson and Foulds, 1981) is 4570, which is disconcertingly large. However, we found that the average Robinson–Foulds distance between 10 random trees with 3085 leaves is 6166. Also, the longest branches indicated by arrows in Figures 6A and B harbour the same three strains sequenced in lanes 6680_6#10, 6775_1#8 and 6823_7#22. It would be interesting to further investigate what sets these strains apart.

4 Discussion

Our new distance measure, d_a, approximates local alignments by anchoring them with long, unique matches (Fig. 1). The requirement that the matches are equidistant in the query and the subject (Fig. 1A) is equivalent to restricting the analysis to ungapped alignments. `andi` is therefore a cross between the early version of BLAST (Altschul *et al.*, 1990) and the genome aligner `MUMmer` (Kurtz *et al.*, 2004): From early BLAST it inherits the idea of ungapped local alignments, from `MUMmer` the idea of looking up unique matches by indexing the subject.

Domazet-Lošo and Haubold (2009) had previously used the power of indexing algorithms to estimate the number of substitutions from the match length distribution. Their program `kr` works on the same principle as the average common substring distance (Cohen and Chor, 2012; Ulitsky *et al.*, 2006), except that `kr` implements theory by Haubold *et al.* (2009) to transform common substring lengths to mutation rates.

Fast as the average common substring methods are, they suffer from two disadvantages: First, local fluctuations in the mutation rate affect the average match length. As a result, the same number of mutations can lead to different distances depending on the degree to which the mutations are clustered. Haubold *et al.* (2013) have used this property to devise a test for recombination. `andi` does not have this problem as it counts mutations directly rather than inferring them from match lengths. The second disadvantage of match-length based methods is that matches induced by non-homologous



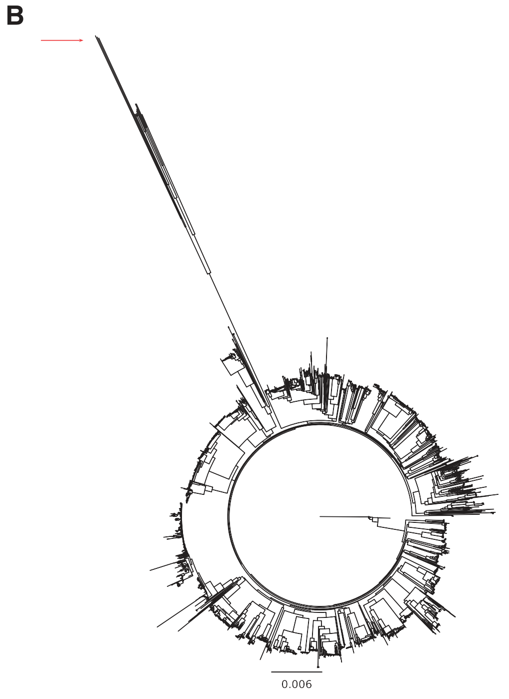


Fig. 6. Phylogeny of 3085 stains of *S.pneumoniae* (Chewapreecha *et al.*, 2014). **A**: Based on distances computed using co-phylog (Yi and Jin, 2013); **B**: based on d_a. Arrows highlight strains discussed in the text. (Color version of this figure is available at *Bioinformatics* online.)

regions are hard to distinguish from matches induced by highly divergent regions. Even a moderate divergence of $K = 0.1$ implies an average match length of 10. Compare this to the expected length of a random match in a 1 Mbase sequence, which according to the theory by Haubold *et al.* (2009) is 10.4. To overcome this limitation of match length distances, Leimeister and Morgenstern (2014) proposed a *k*-mismatch generalization. They show that this outperforms the classical zero-mismatch version of their distance. However, it remains unclear how to choose the critical parameter *k* when applying this method.

Instead of a generalized mismatch approach, we bracket mutations with paired anchors. This should give more accurate results than kr, and andi did compute a better tree for 29 *E.coli* genomes than co-phylog, which in turn gave a better tree than kr. Hence, andi is substantially more accurate than kr.

The accuracy of andi is excellent when applied to simulated sequences with a wide range of substitution rates, *K* (Fig. 2). However, at $K > 0.5$ the search for suitable anchors fails increasingly often (Fig. 2B), which cannot be overcome by lowering the minimum anchor length (Supplementary Fig. S1). Hence, our method is effectively limited to $K \leq 0.5$.

To get an intuition for the evolutionary times implied by $K = 0.5$, consider the average synonymous substitution rate in mammals of 3.51×10^{-9} (Li, 1997, p. 420). The last common ancestor that can occur in a tree restricted to $K \leq 0.5$ lived $0.5/3.51/10^{-9}/2 = 71 \times 10^6$ years ago. This would allow the analysis of great apes, which diverged 15.7×10^6 years ago (Hedges *et al.*, 2006) and mice (*Muridae*, 26.9×10^6 years), but not of these two groups together (92.3×10^6 years).

Apart from maximizing accuracy, we strove to minimize time and memory usage by implementing three ideas: (i) Streaming of query against subject as first implemented in vmatch speeds up suffix array construction compared with the suffix array of all input sequences underlying kr; it also uses much less memory. Our second idea was (ii) to construct only as many enhanced suffix arrays as there are genomes in the sample, rather than constructing an enhanced suffix array for each pairwise comparison. This means that for a sample of *n* genomes andi requires only *n* suffix array constructions, whereas a program like mugsy requires the computation of $O(n^2)$ suffix trees. The third idea was (iii) multithreading, which allows access to the multi-processor architecture of modern computers. However, other programmers might have chosen a different combination of time/memory consumption. For example, vmatch uses half as much memory as andi for suffix array construction, but is slower than the libdivsufsort library we used.

When clustering hundreds of genomes, efficiency becomes paramount. As shown in Figure 3, andi uses only 1 s/Mbase and 45 bytes/bp when applied to simulated sequences. There is an intimate connection between the efficiency of andi and its limitation to closely related sequences: andi approximates local alignments by concatenating exact matches. Looking up exact matches is fast, but this strategy breaks down for divergent sequences where homologous matches become shorter than random matches. This phenomenon is also the reason why fast genome alignment programs like mauve and mugsy work best when applied to closely related genomes (Angiuoli and Salzberg, 2011; Darling *et al.*, 2004).

The accuracy and efficiency observed with simulated data carried over to the analysis of genomes. Here, we compared andi to co-phylog as Haubold (2014) had found this to be the best alignment-free distance estimator for long sequences. However, andi gave a more accurate tree when compared to the tree based on the mugsy alignment (Fig. 4). This improvement in accuracy came without a time penalty as co-phylog computed its tree 36 times faster than mugsy, while andi was a thousand times faster than the alignment. The superior speed of andi comes from the structure of its algorithm and the multithreading; without multithreading, andi would still be 1.6 times faster than co-phylog when applied to the 29 *E.coli/Shigella* genomes. The memory consumption of andi is strictly linear in the number of threads, while time is roughly inversely proportional to the number of threads. This gives the user the opportunity to trade speed for memory and processors, depending on the hardware available.

Aligning the 109 genomes of *E.coli* ST131 took `mugsy` 5.6 days and 52.7 GB RAM. Compare this to the 5 h 33 min it took `mugsy` to align 29 *E.coli* genomes. In other words, a 3.8 times larger sample took 24.6 times longer to align. In contrast, `andi` took with 1 min, 21 s only 4.1 times longer, yielding the correct classification of strains into clades A–C in Figure 5. Moreover, the memory requirement of `mugsy` grew 18-fold, while that of `andi` grew by only 7% to 7.7 GB. These comments are not meant to imply that the `mugsy` alignment computed in the original study was superfluous; it was used for a number of analyses apart from phylogeny reconstruction, including the detection of horizontal gene transfer (Petty *et al.*, 2014). However, quick clustering of genomes is useful, if only as a quality control step.

For our final application, we chose the set of 3085 genomes of *S.pneumoniae*, because here an alignment program like `mugsy` would run far longer than anyone is willing to wait. The current method for comparing sets of bacterial genomes this size is mapping the reads to a reference genome. There is some debate as to the accuracy of the resulting trees (Bertels *et al.*, 2014). `andi` takes as input assembled contigs, which are generated from the raw reads early on in all genome sequencing projects. Given these contigs, `andi` analyses them in 7 h and 35 min using 23.8 GB RAM on a 32 processor computer. Such computing resources are available in most genomics labs. The three outlier strains identified by `andi` are identical to those found by `co-phylog` after a 36.5 days run. This is gratifying and underlines the usefulness of our program.

Acknowledgement

We thank Angelika Börsch-Haubold and Frederic Bertels for comments on the manuscript.

Funding

PP is supported by the Deutsche Forschungsgemeinschaft through grant Pf672/3-1.

Conflict of Interest: none declared.

References

Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.,* **215**, 403–410.

Angiuoli,S.V. and Salzberg, S.L. (2011) Mugsy: fast multiple alignment of closely related whole genomes. *Bioinformatics,* **27**, 334–342.

Bertels,F. *et al.* (2014) Automated reconstruction of whole-genome phylogenies from short-sequence reads. *Mol. Biol. Evol.,* **31**, 1077–1088

Chewapreecha,C. *et al.* (2014) Dense genomic sampling identifies highways of pneumococcal recombination. *Nat. Genet.,* **46**, 305–309.

Cohen,E. and Chor,B. (2012) Detecting phylogenetic signals in eukaryotic whole genome sequences. *J. Comput. Biol.,* **19**, 945–956.

Darling,A.C.E. *et al.* (2004) Mauve: multiple alignment of conserved genomic sequence with rearrangement. *Genome Res.,* **14**, 1394–1403.

Domazet-Lošo,M. and Haubold,B. (2009) Efficient estimation of pairwise distances between genomes. *Bioinformatics,* **25**, 3221–3227.

Felsenstein,J. (2004) *Inferring Phylogenies*. Sinauer, Sunderland, MA.

Felsenstein,J. (2005) PHYLIP (phylogeny interference package) version 3.6.

Fischer,J. and Heun,V. (2007) A new succinct representation of rmq-information and improvements in the enhanced suffix array. *Lect. Notes Comput. Sc.,* **4614**, 459–470.

Haubold,B. (2014) Alignment-free phylogenetics and population genetics. *Brief. Bioinform.,* **15**, 407–418.

Haubold,B. *et al.* (2009) Estimating mutation distances from unaligned genomes. *J. Comput. Biol.,* **16**, 1487–1500.

Haubold,B. *et al.* (2013) An alignment-free test for recombination. *Bioinformatics,* **29**, 3121–3127.

Hedges,S.B. *et al.* (2006) TimeTree: a public knowledge-base of divergence times among organisms. *Bioinformatics,* **22**, 2971–2972.

Jukes,T.H. and Cantor,C.R. (1969) Evolution of protein molecules. In: H.N., Munro (ed.) *Mammalian Protein Metabolism*, **vol. 3**, Academic Press, NY, pp. 21–132.

Kurtz,S. *et al.* (2004) Versatile and open software for comparing large genomes. *Genome Biol.,* **5**, R12.

Leimeister,C.-A. and Morgenstern,B. (2014) kmacs: the *k*-mismatch average common substring approach to alignment-free sequence comparison. *Bioinformatics,* **30**, 2000–2008.

Li,W.-H. (1997) *Molecular Evolution*. Sinauer, Sunderland, MA.

Ohlebusch,E. (2013) *Bioinformatics Algorithms*. Enno Ohlebusch.

Perrière,G. and Gouy,M. (1996) WWW-Query: an on-line retrieval system for biological sequence banks. *Biochimie,* **78**, 364–369.

Petty,N.K. *et al.* (2014) Global dissemination of a multidrug resistant *Escherichia coli* clone. In *Proceedings of the National Academy of Sciences,* 111, 5694–5699, *USA*.

Robinson,D.F. and Foulds,L.R. (1981) Comparison of phylogenetic trees. *Math. Biosci.,* **53**, 514–525.

Ulitsky,I. *et al.* (2006) The average common substring approach to phylogenomic reconstruction. *J. Comput. Biol.,* **13**, 336–350.

Whidden,C. *et al.* (2013) Fixed-parameter algorithms for maximum agreement forests. *SIAM J. Comput.,* **42**, 1421–1466.

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