BIOINFORMATICS

Vol. 00 no. 00 2005
Pages 1–8

Aligning short Reads to Reference Alignments and Trees

Simon A. Berger, Alexandros Stamatakis*

The Exelixis Lab, Scientific Computing Group, Heidelberg Institute for Theoretical Studies, Schloss-Wolfsbrunnenweg 35, D-69118, Heidelberg, Germany

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Motivation: Likelihood-based methods for placing short read sequences from metagenomic samples into reference phylogenies have been recently introduced. At present, it is unclear how to align those reads with respect to the reference alignment that was deployed to infer the reference phylogeny. Moreover, the adaptability of such alignment methods with respect to the underlying reference alignment strategies/philosophies has not been explored. It has also not been assessed if the reference phylogeny can be deployed in conjunction with the reference alignment to improve alignment accuracy in this context.

Results: We assess different strategies for short read alignment and propose a novel phylogeny-aware alignment procedure. Our alignment method can improve the accuracy of subsequent phylogenetic placement of the reads into a reference phylogeny by up to **5.8** times compared to phylogeny-agnostic methods. It can be deployed to align reads to alignments generated by using fundamentally different alignment strategies (e.g., PRANK$_{+F}$ versus MUSCLE).

Availability: `http://www.exelixis-lab.org/software.html`

Contact: Simon.Berger@h-its.org, Alexandros.Stamatakis@h-its.org

1 INTRODUCTION

Currently, bioinformatics is facing two challenges: the *many-core revolution* and the *biological data avalanche* that is driven by novel wet-lab sequencing techniques. In a single run, these new sequencing techniques can generate between hundreds of thousands up to several millions of short DNA reads with a length ranging between 30 to 450 nucleotides (Karow, 2010). One important application of next-generation sequencing methods is *in-vivo* sampling of microbial communities (e.g., in the human gut (Turnbaugh *et al.*, 2008) or on human hands (Fierer *et al.*, 2008)). For phylogenetic analysis of such meta-genomic environmental samples, new likelihood-based methods such as the Evolutionary Placement Algorithm (EPA) (Berger *et al.*, 2011; Stark *et al.*, 2010) and pplacer (Matsen *et al.*, 2010) have recently become available. These new placement algorithms help to establish the provenance of the anonymous and diverse environmental sample of short reads by means of assigning the reads to a given —fixed— reference phylogeny. The reference phylogeny is a fully resolved (strictly bifurcating) *unrooted phylogenetic reference tree* (RT) that is based on a fixed *multiple reference alignment* (RA) of the full length sequences in the RT (Fig. 1).

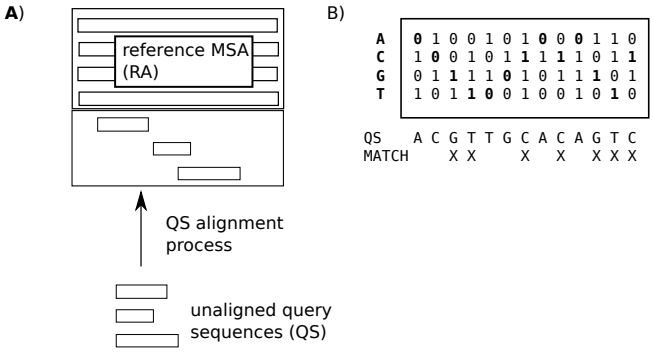

Fig. 1. (A) General scheme of the QS alignment procedure. (B) Matching a QS against an ancestral state vector.

Phylogenetic placement algorithms like the EPA or pplacer work by inserting and removing again one short read at a time into different edges (branches) of the RT. Thereby, they strive to find the optimal score of the extended (by one taxon) trees in order to individually determine the best insertion edge for each short read. The likelihood-based scoring of alternative short read insertion positions in EPA and pplacer is conducted under standard models of nucleotide substitution (e.g., generalized time reversible model using the Γ model of rate heterogeneity (GTR+Γ; Yang, 1994)). The accuracy of such a likelihood-based placement of reads depend upon the multiple sequence alignment, that entails the RA *and* the short sequence reads (henceforth denoted as query sequences (QS)). Therefore, a prerequisite for phylogenetic placement algorithms is, that the QS need to be aligned to the RA (Fig. 1A), before conducting a placement run. We investigate the problem of aligning short reads to a given reference alignment and compare alignment quality of HMMALIGN (Eddy, 1998) to a new phylogeny-aware short read alignment method by means of likelihood-based phylogenetic QS placement accuracy.

The most straight-forward approach to align QS with the RA (containing full length reference sequences) is to simply compute a new multiple sequence alignment (MSA) from scratch comprising the sequences in the RT *and* the QS (e.g., using MUSCLE (Edgar, 2004), MAFFT (Katoh *et al.*, 2005), or PRANK$_{+F}$ (Loytynoja and Goldman, 2008)). Because of the extremely large and continuously

*to whom correspondence should be addressed

growing number of QS, this de-novo alignment approach can be computationally prohibitive.

Alternatively, one can keep the existing (potentially manually curated) RA fixed, and only align the QS with respect to this RA, using dedicated QS alignment methods. One such method for aligning QS with respect to a RA is implemented in the HMMER (Eddy, 1998) tool-suite. HMMER initially builds a profile Hidden Markov Model (HMM) from the RA. Thereafter, the QS are aligned against the profile-HMM that represents the RA. HMMER implements a dedicated method, HMMALIGN that allows for aligning multiple QS (one at a time) against the *fixed* profile-HMM of the RA. HMMALIGN will then output an alignment that contains the RA and the QS that have been aligned with respect to the profile-HMM of the RA. Note that, HMMALIGN frequently also modifies the RA by inserting gaps, if needed. When using a profile-HMM, the entire RA is represented by a monolithic —flat— probabilistic profile that does not use the phylogenetic information of the RT. MUSCLE and MAFFT offer similar options to align sequences (in our case QS) against a monolithic profile that is derived from an existing RA. It has already been shown that QS alignment using HMMALIGN performs reasonably well with respect to phylogenetic placement accuracy (Berger *et al.*, 2011; Matsen *et al.*, 2010). However, depending on the specific alignment strategy/philosophy deployed to generate the RA, better alignment quality (as quantified by QS placement accuracy), can be achieved by incorporating the phylogenetic signal of the RT into the QS alignment process. Hence, we mainly focus on adaptability of QS alignment methods to the underlying, implicit RA structure.

We present PaPaRa (PArsimony-based Phylogeny-Aware short Read Alignment), a novel, phylogeny-aware method for QS alignment. To assess PaPaRa performance, we systematically evaluate phylogenetic QS placement accuracy of the EPA for different QS alignment methods. As baseline for comparisons, we present corresponding results for EPA-based placement accuracy based on QS alignments using HMMALIGN. While MUSCLE and MAFFT also offer modes for sequence-profile alignment (that can be deployed for QS alignment), we exclusively focus on HMMALIGN as a representative of monolithic profile-based approaches for the following reasons: MUSCLE offers an option to conduct profile-profile alignments which corresponds to aligning two MSAs. Thus, either all QS need to be represented by a single profile (i.e., they have to be 'pre-aligned' with respect to each other) or MUSCLE needs to be invoked separately for each QS and the individual results will have to be combined thereafter. Representing all QS by a single profile does not represent a good option, since it may be impossible to align the QS to each other if the short fragments do not exhibit sufficient overlap. For the second MUSCLE alternative, it is unclear, how the resulting individual per QS MSAs —possibly containing gaps in the RA as well— can be synthesized/merged into a single, comprehensive MSA. In contrast to MUSCLE, MAFFT offers an analogous option for QS alignment as considered here. However, in preliminary tests MAFFT returned considerably worse QS alignments than HMMALIGN, with respect to our evaluation criteria (placement accuracy; see below). For the above reasons, we focus on comparing phylogeny-agnostic HMMALIGN performance against phylogeny-aware performance of the PaPaRa method described in the following Section.

PaPaRa is available as open source code at `http://www.exelixis-lab.org/software.html`

2 ALGORITHM

PaPaRa is a novel method for short read alignment against a fixed reference MSA (RA) and the corresponding phylogenetic reference tree (RT). The underlying idea of PaPaRa is to align the QS against the ancestral state vector of each edge in the RT. These ancestral state sequences are conceptually similar to the profiles used in HMMER. However, we do not use a probabilistic model because of prohibitive run times (see below). A key difference to HMMALIGN is that, in our approach we derive one profile per edge (branch) in the RT, as opposed to the single, monolithic profile that represents the whole RA in HMMALIGN. Thus, given an RT with r taxa, m sites, and q QS, we need to execute $O(rq)$ alignment steps or $O(rqm^2)$ operations. Note that, q is typically significantly larger than r. Because of this high time complexity, we also introduce a proof-of-concept parallelization. The ancestral state vectors, as used here, provide two different types of information: the ancestral sequence profile and a tree-derived gap signal (see following paragraphs).

Ancestral Sequence Profile After reading the input data, our algorithm visits the $2r-3$ edges of the RT by means of a depth-first tree traversal, starting at an arbitrary terminal edge leading to a tip. At each edge, we compute the parsimony state-vectors (Fitch and Margoliash, 1967; Sankoff, 1975) of the RT at each end of the edge. The signal from those two state-vectors is then combined using parsimony, to obtain the ancestral parsimony state for an imaginary root-node located on the current insertion edge (Fig. 2). For DNA data, every edge b in the RT will thus be represented by a parsimony state vector $A_b = A_b^1, ..., A_b^n$, where the individual A_b^i are the parsimony states for each alignment site i of the RA. Each entry A_b^i is a bit-vector; each bit corresponds to a character in the sequence alphabet (see Fig. 1B). For DNA data, a bit vector at a site i can have the following state set: $A_b^i = a_b^i(A), a_b^i(C), a_b^i(G), a_b^i(T) \in \{0,1\}^4$, where the a_b^i are the bits which correspond to the four DNA characters. For practical reasons, the A_b^i are implemented using one 32-bit integer per site (e.g., $S_b^i = a_b^i(A) + 2a_b^i(C) + 4a_b^i(G) + 8a_b^i(T)$ for DNA data). This approach is not limited to DNA data; it can be extended to alphabet sizes with up to 32 states in the current implementation.

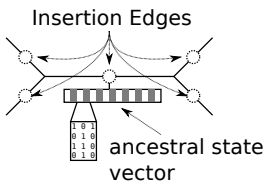

Fig. 2. Unrooted reference tree (RT) and possible query sequence (QS) insertion positions. The QS are aligned against the ancestral state vectors at the candidate insertion positions.

Gap Signal In addition to the parsimony states, we also use phylogenetic information on the gap structure as induced by the tree for our alignment process. This gap information is calculated in conjunction with the parsimony state vectors when the RT is traversed. For each alignment site we recursively compute two flags. One flag (denoted as 'consistent gap'; CGAP) is used for indicating that for a specific site in the RA, there consistently

appears a gap. The second flag (denoted as 'potential gap'; OPEN) is used to indicate if the gap status of a site i is inconsistent. This type of a tree-derived gap signal is based on similar ideas as used in PRANK$_{+F}$ (Loytynoja and Goldman, 2008), which has been designed for de-novo MSA. The two 'gap flags' are deployed in an analogous way as 'compulsory gaps' (CGAP) and 'potential free gaps' (OPEN) in PRANK$_{+F}$. Because the signal is calculated from the tips toward the current insertion edge (Fig. 3), we need to consider three cases for combining gap signals during a post-order tree traversal: TIP/TIP, TIP/INNER, and INNER/INNER. Thus, we need to devise rules for recursively combining the gap signals from the child nodes. In the TIP/TIP case (the children to the left and right of the node at which we intend to compute the 'ancestral' gap signal are tips) the gap signal coming from the two tips can either be gap or non-gap. If both tips have a gap, the result is CGAP, which indicates that in the subtree defined by the current ancestral node the two tips have a gap signal at site i. If only one tip has a gap, the outcome is OPEN, indicating a 'potential gap'. For the TIP/INNER case the flags are computed as follows: If either both child nodes have a gap or the tip has a gap and the ancestral child node has a potential gap (indicated by OPEN), the result is a CGAP. In this case, the 'potential' gap signal coming from the INNER side is upgraded (promoted) to a consistent gap. If only one child node signals a consistent or potential gap, the result is OPEN. Finally, for the INNER/INNER case (i.e., two ancestral child nodes), only two consistent CGAP signals will result in a CGAP at the ancestral node. If only one child node has a CGAP, the result at the ancestral node is OPEN. This rule set for combining and propagating the gap signal through the tree has been derived empirically. While the rule set can evidently be further re-fined, it already yields promising results on real biological data sets. Note that, we align the QS to ancestral states derived from the edges of the RT. Thus, for each edge, we combine the gap signals of the two adjacent nodes. This combination of gap signals is accomplished by using the same rules (TIP/INNER and INNER/INNER cases) as described above. Essentially, this corresponds to placing a temporary root in the middle of the insertion edge.

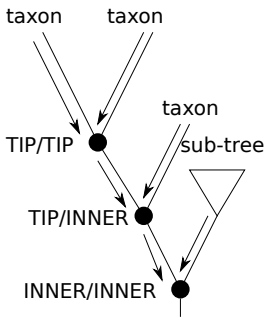

Fig. 3. Gap signal 'flow' from the tips towards the QS insertion position.

Dynamic Programming Alignment Once the gap signal and the ancestral parsimony state at the candidate insertion edge have been computed, they are deployed to calibrate the alignment scoring scheme for the QS at this edge by modifying the match/mismatch and gap open/extend penalties. Only the CGAP flag and *not* the OPEN flag influences the scoring scheme of the alignment algorithm (see below). In general, the CGAP flag will calibrate the scoring scheme such that, aligning QS characters against sites with a CGAP flag is strongly penalized. Opening and extending gaps at these CGAP positions will be preferred. Thereby, if we try to align a QS against the ancestral state of a tree region, where gaps are common for certain alignment sites, it is very likely that the QS alignment will also contain gaps at these sites.

The actual alignment of the QS against each ancestral state vector is carried out by a standard dynamic programming algorithm for pair-wise alignment using affine gap penalties (Gotoh, 1982). Pairwise alignment is conducted with two modifications. Firstly, we deploy a 'free shift' or overlapping alignment strategy (Huang, 1992), that is, gaps inserted at the beginning and/or end of the QS are not penalized. Secondly, the affine gap model is only used for inserting gaps into the QS (i.e., deletions in the QS). For inserting gaps in the RA (i.e., insertions in the QS), we deploy a flat gap penalty. In practice, instead of inserting gaps in the RA, we instead simply delete these insertion characters in the QS. The rationale for this is that, introducing gaps in the RA does not provide any additional information for QS placement using the EPA. In other words, 'empty' RA columns that entirely contain gaps (modeled as undetermined characters in standard ML implementations) will not affect the EPA placements, since we align only one QS at a time. While inserting gaps in the RA may be useful for aligning the QS with respect to each other, our focus here is on evolutionary placement of the QS relative to the RA.

The alignment scoring function is provided in equation 1. The equation recursively defines the score of the dynamic-programming matrix cell $D^{i,j}$ in column i and row j for aligning site A^i of the ancestral state vector against site B^j in the QS.

$$
\begin{aligned}
CG^i &= \begin{cases} 10 & \text{if CGAP is set for site } i \\ 0 & \text{otherwise} \end{cases} \\
(GP^i_{OE}, GP^i_E) &= \begin{cases} (2,1) & \text{if } CG^i = 0 \\ (0,0) & \text{otherwise} \end{cases} \\
S^{i,j} &= \begin{cases} 0 & \text{if } A^i \text{and} B^j \text{match} \\ 3 & \text{otherwise} \end{cases} \\
I^{i,j} &= D^{i,j-1} + 3 \\
D^{i,j}_E &= min \begin{cases} D^{i-1,j} + GP^i_{OE} \\ D^{i-1,j}_E + GP^i_E \end{cases} \\
D^{i,j} &= min \begin{cases} D^{i-1,j-1} + S^{i,j} + CG^i \\ D^{i,j}_E \\ I^{i,j} \end{cases}
\end{aligned} \tag{1}
$$

The term CG^i is used to adapt the scoring scheme for sites where the 'constant gap' (CGAP) flag is set. Thereby, we substantially penalize matching a QS site against such sites in the RT/RA and allow for free gap insertion in the QS at such positions. The remaining definitions correspond to a standard dynamic-programming implementation of the Gotoh (1982) algorithm for sequence alignment with affine gap penalties. As described above, every state A^i is a bit-vector with one bit per alphabet character. Thus, one may think of the ancestral parsimony state vector as

a simple profile, where the bits determine which character of the QS can be aligned for 'free' against an ancestral state character (Fig. 1B). If, for example $A^i = 1, 1, 0, 0$, this means that, As and Cs in the QS can be matched against alignment site i for this ancestral state vector without incurring a mismatch penalty. Thus, the score $S^{i,j}$ is 0 (i.e., no penalty is induced), if the bit corresponding to character j of the QS is set in A^i. Otherwise, the scoring scheme $S^{i,j}$ will return the default mismatch penalty of 3. Note that, the numerical values given in equation 1 represent the default parameters (used in all our experiments), which have been derived empirically. PaPaRa can also deploy user-defined parameters. While there exist more elaborate probabilistic methods (e.g., TKF92; Thorne *et al.*, 1992), 'ad-hoc' scoring schemes (e.g., BLAST or Smith-Waterman) are still widely used for bioinformatics analyses. Moreover, because of the high computational complexity of our approach ($O(rqm^2)$), it is currently not computationally feasible to explore more elaborate scoring schemes. In other words, there is a clear trade-off between model accuracy and execution times.

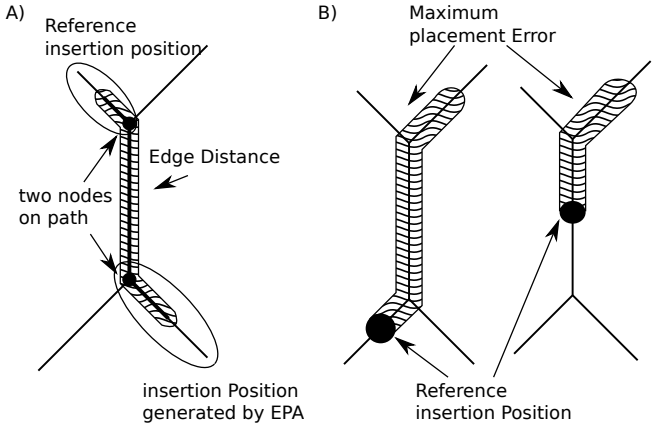

Fig. 4. (A) Distance measures: Node distance (ND) and edge distance (ED). (B) The maximum placement error for two exemplary reference insertion position.

2.1 Implementation

PaPaRa is implemented in C/C++ as experimental extension of RAxML (Stamatakis, 2006). It uses the existing routines for parsing alignment files and trees, as well as the existing parsimony implementation. Initially, the algorithm reads and parses the RT, RA, and, the QS. The taxon names in the RT (Newick format) and the RA (relaxed PHYLIP format, see RAxML v7.0.4 Manual) need to be consistent: all taxa in the RT must have a corresponding sequence in the RA. The QS that shall be assigned to the RT can be read from a separate FASTA file or be included in the RA (for details see PaPaRa README).

The aligner uses a custom-built sequential dynamic-programming implementation (i.e., the core alignment algorithm is single-threaded and not vectorized). However, as Farrar (2007) demonstrated for the smith-waterman algorithm (Smith and Waterman, 1981), dynamic programming algorithms can be significantly accelerated by means of vectorization. Therefore, we plan to also develop a vectorized version of PaPaRa. Further technical implementation details (e.g., cache utilization, parallelization) are described in the supplementary material.

We also implemented and tested a one-sided version of the alignment method, where gaps are only inserted in the QS and not the RA. The respective, simplified dynamic-programming algorithm exhibits fewer dependencies between matrix cell computations. This property can be exploited for further performance improvements. This comes at the cost of alignment quality if insertions (with respect to the sequences in the RA) are common in the QS. For further details please refer to the supplementary material.

As already mentioned, PaPaRa relies on a free-shift alignment strategy. Therefore, after the dynamic-programming matrix has been filled, we search for the optimal alignment score (minimum) in the last row of the dynamic-programming matrix. This allows for insertion of free gaps at the end of the QS. In standard free-shift alignment procedures, one has the search for the minimum score in the last row *and* the last column of the matrix, because it allows for free gaps at either end of both sequences. It is possible to deploy a local alignment scheme or the standard free shift alignment method in PaPaRa to handle cases where the procedure presented here is not applicable (e.g., QS not fully contained in the RA).

3 EXPERIMENTAL SETUP

The main application scenario for PaPaRa is for metagenomic analyses using phylogenetic placement methods such as the Evolutionary Placement Algorithm (EPA) (Berger *et al.*, 2011) or pplacer (Matsen *et al.*, 2010). As mentioned in section 1, for these algorithms the QS need to be in alignment with the RA. To this end, our performance evaluation is specifically designed to assess the accuracy of alignment methods (PaPaRa, HMMALIGN) with respect to analyzing (identifying) short reads by means of phylogenetic placement algorithms. In other words, we do not directly evaluate alignment quality. Instead, we analyze the impact of the QS alignment method on the phylogenetic placement quality/accuracy using the EPA. Therefore, we assess alignment quality by means of the calculated/inferred evolutionary position of the QS. In (Berger *et al.*, 2011) we devise measures and methods for assessing the placement accuracy of short reads using the EPA. We also carried out a basic assessment of QS placement accuracy when QS are re-aligned with HMMALIGN (v3.0), albeit in a different experimental setup and context. Here, we use the same distance/accuracy measures (Fig. 4A). The node distance (ND), which is defined as the number of nodes along the path between the 'true' placement position and the inferred placement position (see below) represents an absolute accuracy measure. The normalized edge distance (EDN%), is a relative measure between 'true' and inferred placement positions that is based on the actual edge-lengths in the RT. The EDN% reflects the relative evolutionary distance between the two positions. In contrast to Berger *et al.* (2011), we use a revised scheme for normalizing the edge distance: Rather than normalizing it by the tree-diameter (longest path in the tree), we now deploy a position-specific maximum possible placement error (Fig. 4B). This position-specific placement-error corresponds to the QS-specific worst-case scenario, that is, we normalize by the longest path from the 'true' insertion position to a terminal edge.

3.1 Realignment of simulated QS

The main part of our performance evaluation compares the placement accuracy of EPA-computed QS placement with respect to the placement position of the optimally aligned QS ('true' placement). The EPA placements obtained without QS alignment are regarded as the optimal ('true') reference placements, against which the phylogenetic placements *after* QS re-alignment with PaPaRa/HMMALIGN are compared. For such an evaluation, we require QS that are already in alignment with the RA in order to compute an optimal reference placement with the EPA that represents the 'true' placement. The QS, which are assumed to be correctly aligned in the reference QS alignment, are initially dis-aligned (we simply remove all gaps), and passed to the two QS alignment procedures (PaPaRa and HMMALIGN) for re-alignment. The thereby re-aligned QS are then used as input for the EPA.

The correctly aligned QS were extracted from 7 real-world full-length biological MSAs (termed original MSAs). The taxon set of each reference MSA is randomly split into two sub-alignments of equal size (each containing 50% of taxa from the original MSA). One half of the original MSA is then used as RA, on which we compute the best-known maximum likelihood (ML) tree with RAxML. This tree is then used as RT for the RA. The other half of the original RA MSA is used to generate a QS set. Because both sub-alignments originally formed part of the same MSA, all sequences in the QS set (and all sub-sequences of these sequences) are in alignment with the RA. The sequences in the QS set that are derived from a MSA of full length sequences are then reduced in length (see below for details) to emulate QS that resemble short sequence reads.

For each data set, we carried out our performance analysis using three common MSA methods to generate three original MSA versions respectively. We computed de-novo MSAs using MUSCLE (v3.70), MAFFT (v6.626), and PRANK$_{+F}$. We selected these three programs, because they are widely used state-of-the-art codes for MSA *and* because they are based on fundamentally different alignment philosophies. Since we adopt an agnostic view on what the best MSA strategy may be, we thereby intend to assess the flexibility and adaptability of PaPaRa to diverse MSA philosophies that are implicitly encoded in the underlying RAs. Finally, we also used the partially manually curated MSAs as provided by the authors of our test data sets. While manual curation is debatable, in particular in the light of reproducibility of results, we nonetheless used the given MSAs because hand-curation is still common practice and may encode empirical biological knowledge about the underlying data. We also conducted an experiment using simulated sequence data. On the simulated alignment, the performance of PaPaRa *and* HMMALIGN is considerably better than on the corresponding real-world alignment and tree (see supplementary material for details). Thus, the QS alignment problem is harder for real data than for simulated data. Therefore, we use real sequence data for our performance assessment. For each of the 7 data sets, we thus have 4 original MSA versions: manually curated (called ORIG throughout the paper), MUSCLE, MAFFT and PRANK$_{+F}$. Table 1 contains information about the length (number of RA columns) in the data sets as well as the number of taxa contained in the RAs and the respective number of QS.

In our experiments, we assume that the full length QS (and the derived short QS) obtained from the 4 MSA versions for each data set, represent ideally aligned QS, with respect to the corresponding RA. To the best of our knowledge HMMALIGN and PaPaRa currently represent the most suitable methods for aligning short-reads to a RA. Therefore, we specifically did not use real short read data, for which the correct alignment to the RA is not known. Our experiments are designed to systematically test the impact of QS alignment quality on the evolutionary placement process using the EPA. Thus, we did not consider alignment quality criteria, other than the relative QS placement error with respect to the reference QS placement obtained from the original MSA.

Data MSA	# sites ORIG	# sites MUSCLE	# sites MAFFT	# sites PRANK	# taxa	# QS
D150	1269	1272	1336	1939	75	1500
D218	2101	2044	1993	6425	109	2180
D500	1398	1402	1402	1479	250	5000
D628	1199	1761	1348	2437	314	6280
D714	1241	1341	1273	2205	357	7140
D855	1436	1469	1443	2208	427	8560
D1604	1271	1325	1278	2475	802	16040

Table 1. Data sets used for evaluation of the QS alignment algorithms. The the values in columns 2–5 correspond to the four RA per data set, which have been generated with the different MSA approaches (ORIG, MUSCLE, MAFFT and PRANK$_{+F}$.

From every full-length QS in the QS set, we randomly sub-sampled 20 contiguous QS with uniformly distributed position and normally distributed lengths (mean length: 100 ± 10 bp and 200 ± 60 bp). We have already used this method for QS generation in (Berger *et al.*, 2011) to create simulated short read sequences that emulate reads obtained from a high throughput sequencer. For each of the 20 sub-sampled QS, we computed an individual reference placement, because the EPA placement of the sub-sampled QS can differ from the placement of the full-length QS. Thereby, we can more accurately assess the QS alignment impact on placement accuracy, without the potential bias that is induced by QS length variation (see Berger *et al.* (2011)). To yield the evaluation more realistic, we then also modified the subsampled QS by introducing typical next-generation sequencer errors. Based on the methods implemented in Grinder (Angly *et al.*, 2009) and the empirical data by Balzer *et al.* (2010), we re-implemented an appropriate model for simulating representative 454 homopolymer sequencing errors. Each homopolymer (this also includes single characters) that is detected in the raw QS is randomly shortened or elongated, according to empirical probabilities provided by Balzer *et al.* (2010).

Because we derive the new RA by splitting the original MSA into two parts (i.e., the RA and the QS set), it is likely that the RA will contain sites that entirely consist of gaps. This is especially true for MSAs generated with PRANK$_{+F}$, that frequently comprise sites with only one or two non-gap characters. Since entirely 'empty' columns that only contain gaps are not present in real MSAs, such columns are completely removed from the RA *prior* to QS alignment (using HMMALIGN and PaPaRa) and placement in our experiments.

4 RESULTS & DISCUSSION

For each of the 7 data sets, we determined PaPaRa- and HMMALIGN-based QS placement accuracy for all 4 original MSA versions. Tables containing the results for all data sets are provided in the supplementary material. The values in the Tables indicate RT-based average ND and EDN% distances between the 'true' reference EPA placements, based on the QS alignment extracted from the original MSA and the respective EPA placements with QS re-alignment. Values for the two QS re-alignment methods (PaPaRa, HMMALIGN) are provided separately. Preliminary tests using MAFFT for QS alignment, generated placements that were at least 2 times further away from their reference position than those obtained with HMMALIGN (data not show). We therefore only report results for HMMALIGN and PaPaRa.

In Table 2 we provide results for the largest data set (D1604) in terms of number of taxa using QS with a mean length of 100 ± 10 bp. When HMMALIGN is used for re-alignment on the manually curated MSA on this data set, EPA placements are on average 1.35 nodes (column HMM, row ORIG) away from the reference placement position. For PaPaRa the corresponding node distance (ND) is 0.28 (column PA, row ORIG). When the relative distance (EDN%) is used, the corresponding values are 1.35 (HMMALIGN) and 0.71 (PaPaRa). Therefore, PaPaRa reduces the error in QS node placement distance by a factor of 4.87 (factor 1.90 for the relative distance) compared to HMMALIGN. For the automated MSA methods (MUSCLE, MAFFT, PRANK$_{+F}$), HMMALIGN and PaPaRa yield analogous accuracy differences. The EPA placements of re-aligned QS are on average 3.11–5.88 times closer (1.7–2.52 times for the EDN% distance) to the reference placements in terms of node distance (ND) for PaPaRa than for HMMALIGN. On some of the smaller data sets (D150, D218 and D714) the PaPaRa-aligned QS can produce worse placements than the HMMALIGN-aligned QS (see supplementary material). However, in most cases, PaPaRa only produces worse results with respect to the EDN% measure.

		ND		EDN %	
D	MA	PaPaRa	HMM	PaPaRa	HMM
100 ± 10	ORIG	0.28	1.35 (4.87)	0.71	1.35 (1.90)
	MUSCLE	0.43	1.35 (3.11)	0.87	1.48 (1.70)
	MAFFT	0.29	1.21 (4.12)	0.72	1.29 (1.80)
	PRANK$_{+F}$	0.41	2.43 (5.88)	0.95	2.41 (2.52)
200 ± 60	ORIG	0.25	0.90 (3.53)	0.63	0.80 (1.28)
	MUSCLE	0.40	1.03 (2.61)	0.76	0.92 (1.21)
	MAFFT	0.26	0.82 (3.18)	0.65	0.79 (1.21)
	PRANK$_{+F}$	0.34	1.65 (4.80)	0.84	1.39 (1.64)

Table 2. Placement accuracy for the two QS alignment methods on the largest data set (D1604). The relative accuracy of HMMALIGN compared to PaPaRa is given in parentheses.

For the longer QS of mean length 200, in most cases, placement accuracy increases for both alignment methods. The improvements are more pronounced for HMMALIGN, where the ND is improved by up to a factor of 2. Generally, accuracy differences between PaPaRa and HMMALIGN decrease. For 5 out of the 7 data sets PaPaRa produces worse results than HMMALIGN at least for some of the tests (i.e., for certain RA and distance measure combinations, see supplementary material). As with the shorter QS, this is especially pronounced on the smaller data sets with less taxa. Thus, apparently, the advantage of using a phylogeny-aware QS alignment strategy on data sets with few taxa is smaller. In contrast to PaPaRa, on small data sets HMMALIGN can take advantage of its more powerful probabilistic RA model and the stronger signal contained in the 200 bp long QS. However, the typical RT will be considerably larger than the smallest data sets in this study, because of the very dense taxon sampling of the 16S rRNA. Thus, while the accuracy improvement induced by PaPaRa is minor on small data sets, it substantially improves placement quality on the larger reference data sets in our experiments.

The rather pronounced difference between the two distance measures (i.e., when the ND is considered, the advantage of PaPaRa over HMMALIGN is larger than for the EDN%), can be attributed to the RT shape of this data set (D1604): Visual inspection revealed that, it contains a large number of closely related taxa which gives rise to a large number of relatively short edges (branches) near the tips of the tree. Thus, if a QS is misplaced within such a region of the tree, this can result in a relatively large ND (because there is a large number of nodes in the region), but a small EDN% since edges between the nodes are short. The HMMALIGN re-aligned QS tend to be misplaced in such 'dense' areas of the tree, which results in a relatively large average ND compared to PaPaRa re-aligned QS. To this end, by using a phylogeny-aware approach, PaPaRa can better use such densely sampled areas in the RT, while such a fine-grained resolution can not be achieved by using a 'flat' probabilistic profile (e.g., HMMALIGN). On smaller data sets the differences between the two distance measures are less pronounced (see supplementary material).

In most cases, the largest difference in placement accuracy between PaPaRa and HMMALIGN is observed for PRANK$_{+F}$-based MSAs. Because of the specific MSA approach in PRANK$_{+F}$, a strong and consistent gap signal is embedded into the original MSA. In contrast to HMMALIGN, PaPaRa is able to use this embedded gap-signal in combination with the respective RT. In Figure 5 we provide histograms of the average ND distribution for QS (with mean length 100bp) over all data sets and for all reference MSAs. PaPaRa-based QS alignments generate placements that are, on average, closer to the 'true' reference position. The histograms also show that for PRANK$_{+F}$-generated MSAs, the placement accuracy decrease induced by using HMMALIGN is more pronounced compared to other MSA methods. In general, PaPaRa is thus more robust with respect to different MSA philosophies and hence more adaptable.

For the above experiments, we knew a priori, that the QS had sufficiently closely related sequences in the RA. If this is not given (e.g., if reads from a distant clade not contained in the RT are sampled), according to some preliminary experiments, neither the QS alignment method nor the EPA can be expected to produce resonable results. This observation also holds when the QS stem from a different (e.g., non-orthologous) genomic region than the sequences in the RA. Therefore, we suggest that the QS should be checked beforehand, for example by doing a quick BLAST search against the sequences in the RA to reject completely unrelated sequences.

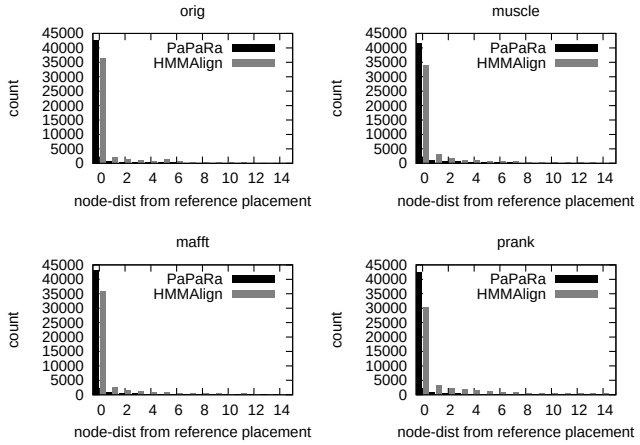

Fig. 5. Histograms showing the distribution of the placement error (ND) for PaPaRa and HMMALIGN aligned QS, over all data sets.

4.1 Execution times

We also carried out a runtime assessment of HMMALIGN and PaPaRa. A serial execution of PaPaRa requires 385s – 44,270s on the smallest (D150) and largest (D1604) data set respectively (using the ORIG MSA and QS of lengths 200 ± 60 bp on an 3.2 GHz Intel Core i5; compiled with `gcc 4.5.1` for Linux). The corresponding HMMALIGN times range between 61s and 1031s. Thus, HMMALIGN is 6.3 – 43 times faster than PaPaRa. This performance difference is not surprising, because PaPaRa runtimes depend on the number of QS *and* the number of taxa in the RT. In other words, PaPaRa exhibits a significantly higher theoretical runtime complexity than HMMALIGN. Therefore, performance optimization of the core alignment procedure is essential for overall PaPaRa performance. The inherent —significantly higher— time complexity of PaPaRa is also one main reason for aligning against ancestral parsimony state vectors (i.e., bit-vectors), instead of using a probabilistic approach that would require costly floating point arithmetics.

Currently, PaPaRa creates the QS alignments in two phases: Initially, all QS are aligned, and thereby scored, against all ancestral state vectors (insertion positions/edges of the RT). For performance reasons the actual alignments (i.e., the dynamic programming traceback) are not generated in this phase. After the best scoring insertion position has been determined for each QS, the actual alignments are then generated by aligning them again to the best positions in a second step. The initial step normally accounts for more than 99% of overall runtime. As already mentioned, the core alignment procedure could be further optimized by deploying 128-bit wide SSE or even 256-bit wide AVX vector instructions. One could also think of a more compact bit-level representation of the input data to reduce memory requirements and cache misses. Our current dynamic programming implementation can perform about 120 MCUPS (million cell updates/s) on the Intel Core i5. For comparison, Farrar (2007) reports more than 3000 MCUPS for his SSE optimized smith-waterman implementation on an older 2.0 GHz Intel Xeon Core 2 Duo processor. Note that HMMALIGN, as used here, already includes SSE vectorization in the alignment algorithm. We are therefore confident that, the run-time gap between HMMALIGN and PaPaRa can be significantly reduced in future versions of the code.

5 CONCLUSION & FUTURE WORK

We have conducted an experimental evaluation of methods for aligning short QS against a fixed RT and RA in the context of likelihood-based evolutionary QS placement methods. We also introduced PaPaRa, a novel phylogeny-aware method for this purpose. On short QS and large RAs, PaPaRa performs better than the currently best phylogeny-agnostic method (HMMALIGN). For longer QS and small RAs the performance of the current PaPaRa implementation is relatively poor. Apparently, the more powerful probabilistic model in HMMALIGN, is beneficial, if the RA is small enough to be represented by a single flat profile. For larger RAs, PaPaRa has the advantage of sampling different signals from different parts of the associated RT and performs well, despite using a simple model for ancestral states and an 'ad-hoc' scoring scheme. We intend to introduce additional heuristics for reducing the total number of ancestral state vectors against which individual QS need to be aligned. We also plan to exploit data-parallelism in the core alignment algorithm by using SSE and AVX vector instructions. PaPaRa can also be used to generate multiple 'candidate' alignments for each QS (it is likely that the current method generates multiple alignments with equal scores per QS because of the discrete scoring scheme). Those different per-QS alignment candidates could then be scored by their placement scores under ML to select the 'best candidate'.

As a more fundamental improvement, we will explore methods to refine the gap propagation in the tree based on a binary likelihood model. A further option, worth exploring, would be to use a probabilistic alignment approach, where the ancestral states resemble the probability vectors as used in likelihood-based tree inference. In the long term we will work on integrating the EPA with PaPaRa. One possible application would be dynamic alignment/tree extension by full length sequences as they appear in GenBank. This would represent a step towards simultaneous/integrated tree-building and alignment.

Acknowledgements

Funding: This work was partially funded under the auspices of the Emmy-Noether program by the German Science Foundation (DFG).

Conflict of Intererst: none declared.

REFERENCES

Angly, F. E., Willner, D., Prieto-Davó, A., Edwards, R. A., Schmieder, R., Vega-Thurber, R., Antonopoulos, D. A., Barott, K., Cottrell, M. T., Desnues, C., Dinsdale, E. A., Furlan, M., Haynes, M., Henn, M. R., Hu, Y., Kirchman, D. L., McDole, T., McPherson, J. D., Meyer, F., Miller, R. M., Mundt, E., Naviaux, R. K., Rodriguez-Mueller, B., Stevens, R., Wegley, L., Zhang, L., Zhu, B., and Rohwer, F. (2009). The GAAS metagenomic tool and its estimations of viral and microbial average genome size in four major biomes. *PLoS computational biology*, **5**(12), e1000593.

Balzer, S., Malde, K., Lanzn, A., Sharma, A., and Jonassen, I. (2010). Characteristics of 454 pyrosequencing dataenabling realistic simulation with flowsim. *Bioinformatics*, **26**(18), i420–i425.

Berger, S. A., , Krompass, D., and Stamatakis, A. (2011). Performance, accuracy and web-server for evolutionary placement of short sequence reads under maximum-likelihood. *Syst. Biol.*, **60**(3), 291–302.

Eddy, S. (1998). Profile hidden markov models. *Bioinformatics*, **14**(9), 755–763.

Edgar, R. C. (2004). Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucl. Acids Res.*, **32**(5), 1792–1797.

Farrar, M. (2007). Striped Smith-Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, **23**(2), 156–161.

Fierer, N., Hamady, M., Lauber, C., and Knight, R. (2008). The influence of sex, handedness, and washing on the diversity of hand surface bacteria. *Proc. Natl. Acad. Sci. USA*, **105**(46), 17994–17999.

Fitch, W. and Margoliash, E. (1967). Construction of phylogenetic trees. *Science*, **155**(3760), 279–284.

Gotoh, O. (1982). An improved algorithm for matching biological sequences. *J. Mol. Biol.*, **162**, 705–708.

Huang, X. (1992). A contig assembly program based on sensitive detection of fragment overlaps. *Genomics*, **14**(1), 18–25.

Karow, J. (2010). Survey: Illumina, solid, and 454 gain ground in research labs; most users mull additional purchases.

Katoh, K., Kuma, K., Toh, H., and Miyata, T. (2005). MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucl. Acids Res.*, **33**, 511–518.

Loytynoja, A. and Goldman, N. (2008). Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis. *Science*, **320**(5883), 1632.

Matsen, F., Kodner, R., and Armbrust, E. V. (2010). pplacer: linear time maximum-likelihood and bayesian phylogenetic placement of sequences onto a fixed reference tree. *BMC Bioinformatics*, **11**(1), 538.

Sankoff, D. (1975). Minimal mutation trees of sequences. *SIAM. J. Appl. Math.*, **28**, 35–42.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.

Stamatakis, A. (2006). RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, **22**(21), 2688–2690.

Stark, M., Berger, S. A., Stamatakis, A., and von Mering, C. (2010). MLTreeMap - accurate Maximum Likelihood placement of environmental DNA sequences into taxonomic and functional reference phylogenies. *BMC Genomics*, **11**(1), 461+.

Thorne, J. L., Kishino, H., and Felsenstein, J. (1992). Inching toward reality: An improved likelihood model of sequence evolution. *J. Mol. Evol.*, **34**(1), 3–16.

Turnbaugh, P., Hamady, M., Yatsunenko, T., Cantarel, B., Duncan, A., Ley, R., Sogin, M., Jones, W., Roe, B., Affourtit, J., *et al.* (2008). A core gut microbiome in obese and lean twins. *Nature*, **457**(7228), 480–484.

Yang, Z. (1994). Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites. *J. Mol. Evol.*, **39**, 306–314.