Alignment-free Phylogeny Reconstruction Based On Quartet Trees

Dissertation
for the award of the degree
„Doctor rerum naturalium“
of the Georg-August-Universität Göttingen
within the doctoral program
„Computer Science“ (PCS)
of the Georg-August-University School of Science (GAUSS)

submitted by
Thomas Dencker

from
Nordenham

January 27, 2020

Members of the Thesis Committee

Prof. Dr. Burkhard Morgenstern (1st Referee)
Institute for Microbiology and Genetics, Georg-August-Universität Göttingen

Prof. Dr. Stephan Waack (2nd Referee)
Institute for Computer Science, Georg-August-Universität Göttingen

Dr. Johannes Söding
Quantitative and Computational Biology, Max-Planck Institute for Biophysical Chemistry Göttingen

Further Members of the Examination Board

Prof. Dr. Christoph Bleidorn
Johann-Friedrich-Blumenbach Institute for Zoology & Anthropology, Georg-August-Universität Göttingen

Prof. Dr. Anja Sturm
Institute for Mathematical Stochastics, Georg-August-Universität Göttingen

Prof. Dr. Jan de Vries
Institute for Microbiology and Genetics, Georg-August-Universität Göttingen

Date of the oral examination: March 4 th 2020

Affidavit

I hereby confirm that this thesis has been written independently and with no other sources and aids than quoted.

Göttingen, 27. January 2020

Thomas Dencker

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich in den letzten Jahren begleitet und unterstützt haben.

Zuallererst bedanke ich mich bei meinem Doktorvater Prof. Dr. Burkhard Morgenstern für die Betreuung dieser Doktorarbeit und die andauernde Unterstützung seit der Bachelorarbeit. Ohne ihn wäre ich gar nicht in der Bioinformatik gelandet. Besonders dankbar ich bin für die vielen Ratschläge, mit denen ich vor allem meine Präsentationsfähigkeiten verbessern konnte. Außerdem hat er mir die Möglichkeit gegeben, an der RECOMB-CG in Kanada teilzunehmen und viele neue Erfahrungen zu sammeln. An die gemeinsame Zeit werde ich mich noch lange positiv erinnern. Außerdem danke ich Dr. Soeding und Prof. Dr. Waack für die konstruktive Kritik in den Komiteetreffen. Des Weiteren möchte ich mich auch bei Dr. Peter Meinecke, Heiner Klingenberg, Lars Hahn und Matthias Blanke für die vielen hilfreichen Diskussion und das gute Feedback bei meinen Präsentationen bedanken.

Ganz besonders bedanke ich mich bei meinem Freund und ehemaligen Kollegen Chris Leimeister dafür, dass er mir immer mit gutem Rat zur Seite stand und für die vielen hilfreichen Gesprache, die mich immer wieder motiviert haben. Mein ganz besonderer Dank gilt auch meinen Eltern, die mich während des gesamten Studiums unterstützt und immer an mich geglaubt haben.

Contents

Publication List

Journal Papers

- **Thomas Dencker**, Chris-André Leimeister, Michael Gerth, Christoph Bleidorn, Sagi Snir, Burkhard Morgenstern. ‘Multi-SpaM’: a maximum-likelihood approach to phylogeny reconstruction using multiple spaced-word matches and quartet trees *NAR Genomics and Bioinformatics*, 2:lqz013, 2020.
- **Thomas Dencker**, Niklas Birth und Burkhard Morgenstern. Insertions and deletions as a phylogenetic signal in an alignment-free context. Prepared for submission.
- Chris-Andre Leimeister, **Thomas Dencker**, and Burkhard Morgenstern. Accurate multiple alignment of distantly related genome sequences using filtered spaced word matches as anchor points. *Bioinformatics*, 35(2):211–218, 2019.
- Andrzej Zielezinski, Hani Z Girgis, Guillaume Bernard, Chris-Andre Leimeister, Kujin Tang, **Thomas Dencker**, Anna Katharina Lau, Sophie Röhling, Jae Jin Choi, Michael S Waterman, Matteo Comin, Sung-Hou Kim, Susana Vinga, Jonas S Almeida, Cheong Xin Chan, Benjamin T James, Fengzhu Sun, Burkhard Morgenstern, Wojciech M Karlowski Benchmarking of alignment-free sequence comparison methods. *Genome Biol.*, 20(1):144, 2019.
- Sophie Röhling, Alexander Linne, Jendrik Schellhorn, Morteza Hosseini, **Thomas Dencker** and Burkhard Morgenstern. The number of k-mer matches between two DNA sequences as a function of k and applications to estimate phylogenetic distances. *PLOS ONE*, in press, 2020.

Conference Papers (peer reviewed)

- **Thomas Dencker**, Chris-Andre Leimeister, Michael Gerth, Christoph Bleidorn, Sagi Snir, and Burkhard Morgenstern. Multi-SpaM: a Maximum-Likelihood approach to phylogeny reconstruction using multiple spaced-word matches and quartet trees. In *Comparative Genomics*, pages 227–241. Springer International Publishing, 2018.

Abstract

Traditional methods for phylogeny reconstruction are based on multiple sequence alignments and character-based methods. This combination of computationally expensive methods leads to very accurate results, but it is ill-suited to handle the enormous amount of sequence data that is available today. As a consequence, very fast alignment-free methods have been developed. These methods calculate pairwise distances in order to build phylogenetic trees. However, current alignment-free methods are generally less accurate than traditional methods.

In this thesis, I developed *Multi-SpaM* which is a novel alignment-free approach that tries to combine the best of both worlds. This method quickly finds small gap-free 'micro-alignments' – so-called blocks – involving four sequences. A binary pattern defines at which positions the nucleotides have to match. At the remaining *don't care* positions, the possibly mismatching nucleotides are first used to remove random matches with a filtering procedure previously introduced by *Filtered Spaced-Word Matches (FSWM)*. Then, the character-based method *RAxML* is used to find the optimal quartet tree for each block. Subsequently, all quartet trees are amalgamated into a supertree with *Quartet MaxCut*. This approach can be used to build phylogenetic trees of high quality.

Furthermore, I showed multiple ways that could help to improve *Multi-SpaM*. The distances between two adjacent blocks involving the same four sequences can be used to identify putative insertions and deletions from which accurate quartet trees can be derived. These trees could be used both on their own and in combination with the quartet trees produced by *Multi-SpaM* to build or improve phylogenetic trees using *Quartet MaxCut*. As an alternative, we also used *Maximum-Parsimony* to infer accurate phylogenies from these putative insertions and deletions. In other experiments, I tried to give the individual quartet trees weights based on SH-like support values and tried to use *Neighbor-Joining* in order to speed up *Multi-SpaM*.

Moreover, I contributed to another extension of the *FSWM* approach. Here, we used these matches as anchor points for a genome alignment tool called *mugsy*. We found that a higher number of homologous pairs could be aligned for more distantly related species in comparison to other anchor points used with the same alignment program.

1 Introduction

Evolution is a central concept in biology. A formal theory of evolution was first formulated by Charles Darwin in his book "On the Origin of Species" [23]. We now believe that all species [89] ultimately evolved from a common ancestor [137]. The evolutionary history of a group of species is called a *phylogeny*. It is commonly visualized as a *phylogenetic tree*. In some cases, the phylogeny is depicted as a phylogenetic network [58] to take even more complex relationships into account. Reconstructing phylogenies is a fundamental task in the life sciences. Ultimately, it is one major goal to identify the evolutionary relationships between all species and thus reconstruct the tree of life [53].

For the longest time, relationships between species were determined purely by morphological features such as bone structures. Based on these features, species can be assigned to certain *taxa*. Taxa are groups of species on different levels (domain, kingdom, phylum, class, order, family, genus, and species) and can be thought of as subtrees of the tree of life. Reconstructing phylogenies based on morphological features can be a challenging task. One problem is that a feature can appear independently in different taxa. The ability to fly was developed by both birds and bats and is a common example for convergent evolution. Such a feature is also called a *homoplasy* [8]. In contrast, features are called *homologies* when they exist due to divergent evolution, i.e. they were inherited from a shared ancestor. In order to reconstruct the correct phylogeny, it has to be based on sufficiently many homologies.

However, the morphological approach reaches its limits when homologies are hard to find. For example, for certain taxa, such as closely related bacteria, it is difficult to make out any differences. In such a case, the genetic material can provide evidence of the phylogenetic relationship. The first step towards this goal was achieved when the structure of deoxyribonucleic acid (DNA) was described in 1953 by Watson and Crick [146]. The DNA consists of two strands that form the widely known double helix structure. Each strand is made out of four different nucleotides: adenine, guanine, cytosine, and thymine. The nucleotides from both strands normally form two base pairs, adenine–thymine and guanine–cytosine. Thus, one strand is complementary to the other strand and it can be used to reconstruct the other one. As each strand is translated in the opposite direction, it is common to also consider the reverse complement of a DNA sequence.

In order to analyze the DNA, it has to be sequenced first. Historically, sequencing methods like the Sanger Sequencing [115] were slow and expensive. Ever since, the cost of DNA sequencing has only been going down. *Next generation sequencing* methods [84] produce short fragments of DNA, so called *reads*. Depending on the method, they can have different lengths and error rates. The reads can be *assembled* [122] in order to obtain the whole genome.

The sequence data of assembled genomes can be used in a similar way as with morphological features. Genomes consist of individual genes that can be translated into proteins. The presence or absence of genes as well as molecular changes within the genes can be used as features for sequence comparison. For that, homologous genes need to be identified. However, this can be difficult because genes can be duplicated during evolution. Therefore, there are two types of homologies. Genes that evolved from the same copy in a shared ancestor are called *orthologous* genes. In contrast, *paralogous* genes evolved from different copies. Clearly, orthologous genes should be used for phylogeny reconstruction. However, finding orthologous genes [117, 145, 57] can be challenging. Thus, it is desirable to compare different taxa based on the sequence data alone. In the following, I will describe different ways to reconstruct a phylogenetic tree based on nucleotide data.

1.1 Sequence alignment

It has long been known that homologous nucleotides or amino acids can be used to gain information about the evolutionary relationship of different taxa [157]. Sequence *alignments* are an arrangement of DNA or protein sequences that make it possible to identify and compare homologous characters. In the following, I will assume that all sequences consist of nucleotides. An alignment is a matrix in which two or more sequences are arranged in order. Every row of the matrix has the same length. Therefore, gap characters are inserted in such a way that homologous characters can appear in the same column. Under the assumption that all sequences in the alignment share a common ancestor, mismatching nucleotides in the same column can be interpreted as substitutions. The gap characters show putative insertions or deletions (*indels*). An example is shown in Table 1.

Intuitively, an optimal alignment should match identical nucleotides and have as few substitutions and gaps as possible. An optimal alignment can be found algorithmically with

A	C	C	-	-	G	A	T
A	C	G	A	T	G	-	T

Table 1: Example for a pairwise sequence alignment. There are two putative indel events at position 4 and 7 as well as a substitution at position 3.

regard to a scoring scheme. It is possible to simply score matches and mismatches, but there are substitution matrices that account for different substitution rates for pairs of nucleotides as they appear in reality [19]. Similarly, gaps can be punished by a linear gap penalty or there can be a higher penalty for opening a gap and a lower penalty for elongating a gap. This affine-linear gap penalty is motivated by the fact that a longer gap is more likely to happen in nature than multiple small ones.

Under such a scoring scheme, the *Needleman-Wunsch algorithm* [97] constructs the optimal *global* alignment, i.e. an alignment over the entire length of two sequences. It solves the problem in time proportional to the product of the sequence lengths by *dynamic programming*. All pairs of prefixes of the two sequences are aligned and their scores are stored. To calculate an alignment, the score of a previous calculated smaller alignment can be used and adjusted whether the last position is a match, mismatch or a gap in either sequence. Thus, the algorithm exhaustively evaluates all possible alignments and is guaranteed to find the optimal one. However, global alignments are not meaningful when the sequences are only related locally. Therefore, the *Smith-Waterman algorithm* [124] was developed. This algorithm is a modification of the Needleman-Wunsch algorithm that limits alignments to regions with high similarity.

The Smith-Waterman algorithm, or a variant thereof, is heavily used for homology detection. Tools like *BLAST* [2, 3] can quickly find local homologies in large databases. If a large number of sequences need to be compared, then the algorithm is no longer fast enough. While there are faster implementations of the Smith-Waterman algorithm that utilize vector parallelization effectively [32], it is more common to find (inexact) word matches to identify sequences that have potentially high scoring local alignments. This way, the Smith-Waterman algorithm needs to be used only on relatively few sequences. A popular tool that implements such an approach is *diamond* [17].

Oftentimes, it is not enough to find homologous sequences and pairwise alignments. In order to compare multiple sequences at the same time on a molecular level, *multiple se-*

quence alignments are necessary. Such an alignment is significantly more complex. In fact, multiple sequence alignments are known to be NP-hard [143]. Therefore, heuristical methods are needed. These methods can generally be divided into two classes. Some methods align sequences progressively [38] along a guide tree. Previously aligned nucleotides are not realigned when new sequences are added. Thus, the quality of such a multiple sequence alignment strongly depends on the quality of the guide tree and the initial alignments. The most commonly used multiple sequence alignment tools, *ClustalW* [138] and *Clustal Omega* [120], fall into this category and are available on widely used public webservers. Other popular progressive alignment methods are *MAFFT* [64, 63] and *T-Coffee* [99]. The other group of methods improve alignments iteratively, i.e. the alignment can be improved in a later iteration based on some objective function. One popular method, that can refine its alignments, is *MUSCLE* [29]. Furthermore, *dialign* [93] follows an unusual approach. The multiple sequence alignment is based on short pairwise alignments without gaps. Thus, only parts that are locally related are aligned. Therefore, dialign can outperform other methods if the sequences are not related over their entire length.

Multiple sequence alignments are very useful to accurately compare sequences. However, there are several limitations [154]. The algorithms for multiple alignments only deal with the basic evolutionary events. However, there are other events that can not be addressed properly. During evolution, genes can be duplicated or inserted due to horizontal gene transfer [43]. In the latter case, genes are transferred to other genomes without a parent-child relationship. These events can lead to paralogous or unrelated genes being aligned. Moreover, the entire genome can be rearranged which cannot be represented in a classical alignment at all where the sequences are always aligned in order. There are even more reasons why alignments might be inaccurate. The alignment depends on the scoring scheme which is often rather arbitrary. Different parameters can lead to substantially different solutions [148]. Furthermore, the accuracy of the alignments can also be effected by the relatedness of the sequences. More distantly related sequences are harder to align, especially if the sequences are not related over their entire lengths.

Apart from these problems, the most important limitation is the high runtime and memory usage. Thus, alignment methods do not scale well to whole-genome data. In order to be able to align whole genomes, several genome aligners have been developed, such as *Cactus* [102] and *mugsy* [5]. These tools only align parts of the sequences and are thus much faster and can even deal with genome rearrangements. For instance, *mugsy* uses maximal unique words that appear in multiple sequences as starting points for their alignment. Using word matches in multiple sequences is a common strategy to speed up or improve the quality of alignments [94, 74].

1.2 Tree building methods

Traditionally, phylogenetic trees are reconstructed using multiple sequence alignments. It is common practice to calculate alignments for a set of genes that are found in multiple or even all taxa. Since the sequences that need to be aligned are much shorter, this procedure is computationally less expensive than a full sequence alignment. Moreover, the individual alignments can be more accurate if the genes are known to be orthologous. Several alignments can be concatenated to form a *supermatrix*. In case not all alignments contain the same group of taxa, special characters are inserted to denote missing information. In order to reconstruct the phylogenetic tree from a multiple sequence alignment or supermatrix, there are generally two approaches which have very different runtime requirements. The first class of methods are called character-based methods. These methods use the nucleotide characters directly to evaluate a given tree topology. While this step is usually very fast, it is very hard to find the optimal tree topology. In fact, character-based inference of phylogenies is known to be a NP-hard problem [41, 20]. Despite this fact, trees built by such methods are generally considered to be the most accurate and are the method of choice in many studies. In the following, I will describe some commonly used character-based methods.

The first one is *Maximum-Parsimony* [33, 40]. In this method, the optimal tree is the one that requires the least amount of substitution events to explain a given alignment. This optimality criterion applies *Ockham's Razor [100]* (or *lex parsimonae*) to evolutionary events. This principal states that among multiple hypotheses that can explain the observed data, the simplest hypothesis should be chosen. Following this principle, there is a simple way to determine the minimum number of substitution events that must have happened during evolution to explain the data in a column of a given alignment for a given tree (see Figure 1 for an example).

By summing up the scores of all columns, the *length* of a tree is calculated. Clearly, the

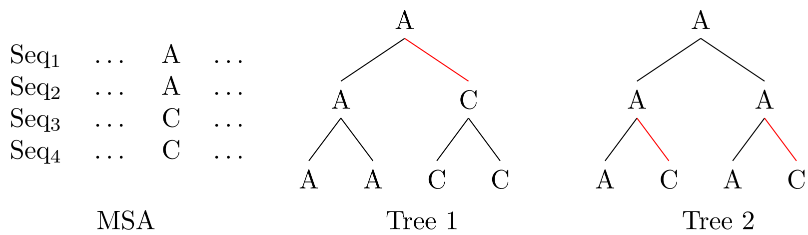

Figure 1: For a row in the given multiple sequence alignment, there are two trees that could explain the given data. The substitutions, that must have happened during evolution, are shown with red edges. The first tree requires only one substitution and is thus the most parsimonous tree.

tree with the shortest length is the optimal tree. The most challenging part of *Maximum-Parsimony* is to find the optimal tree topology in the tree space. There are $\frac{(2n-5)!}{(n-3)!2^{n-3}}$ possible unrooted tree topologies for $n > 2$ species [37]. This number grows very quickly too large to examine every possible tree topology. In fact, *PAUP** [133], a commonly used implementation of *Maximum-Parsimony*, does not allow exhaustive search for any datasets with more than 12 species. For slightly larger datasets, the *branch and bound* algorithm [52] can be used which can still reconstruct the optimal tree. Larger datasets require heuristical algorithms. In this case, *hill climbing* is used which evaluates "neighboring" trees, until no better tree can be found. The most common strategies to explore the tree space are *nearest-neighbor-interchange* (NNI) [92, 107], *subtree prune-and-regraft* (SPR) [51] and *tree bisection and reconnection* (TBR) [27]. With these strategies, the resulting tree might not be the optimal tree, but the computation time is reduced drastically. It can be reduced even further, especially for very large datasets, if the characters are sampled. Such an approach has been implemented with the *Parsimony-ratchet* [98].

One important characteristic of this method is that it does not assume a model of evolution. This can, of course, be considered a downside, but it does allow for many different types of characters such as present/absent encoding of some features. In case such characters are used, *Maximum-Parsimony* is the obvious method to use. These use cases are also one reason for the popularity of the method. One example of uncommon characters are gap characters. In many implementations, gaps are treated as missing information. Even if this is not the case, insertions and deletions could only be considered for length 1 since every column is evaluated individually. Longer insertion and deletion need to be encoded separately as special characters [121].

A drawback of the simple *Maximum-Parsimony* approach is that every substitution is assumed to be equally likely. In general, this is not very realistic. Thus, many statistical models of evolution have been developed. A commonly used model is the *GTR* (generalized time reversible) model [135]. This model considers different substitution probabilities for every pair of nucleotides. Moreover, it is also possible to take variable substition rates into account [150]. Apart from these common models, there are many more, some of which even consider the likelihood of insertions and deletions [139, 140]. *Maximum-Likelihood* is a tree building method that can utilize such a model. This method is very similar to *Maximum-Parsimony*. Instead of the length of a given tree, the likelihood, that the data in the alignment is observed under a chosen model, is calculated. The tree with the highest likelihood is accepted as the optimal tree. One of the most commonly used tools that implement *Maximum-Likelihood* is *RAxML* [128].

Another popular character-based method is Bayesian Inference [110]. This method uses *markov chain monte carlo sampling* to calculate the tree with the highest likelihood based on prior probabilities.

Even though these methods are considered to produce highly accurate trees, they are also computationally expensive. In order to reduce the runtime, distance-based methods have been developed. Based on a distance matrix consisting of pairwise similarity (or dissimilarity) measures, the phylogenetic tree can be built very quickly. In practice, these methods are extremely fast and can be used on far larger datasets than *Maximum-Parsimony* or *Maximum-Likelihood*. While the distance measures can be arbitrary, they are usually based on a multiple sequence alignment. The multiple sequence alignment implies pairwise sequence alignments from which pairwise distances can be calculated by counting the mismatches. The mismatches per position are, however, not an accurate measure of how many substitutions have happened since the two species diverged from a common ancestor. It is always possible that multiple consecutive substitutions happen at the same position in the sequence. In this case, the true number of substitution is not reflected in the distance measure. To fix this issue, the distance needs to be corrected according to a model of evolution. For distance-based methods, simple models are generally preferred. The simplest model is the *Jukes&Cantor* [62] model which assumes equal base frequencies and substitution rates for all pairs of nucleotides. However, there are many more models that could be used be to correct the distances.

In order to build a phylogenetic tree from a distance matrix, hierarchical clustering is used. In the beginning, every cluster consists of a single taxon. The two clusters with the shortest (corrected) distance are joined into a new cluster. This new cluster represents an inner node in the phylogenetic tree. Afterwards, distances between the new cluster and all other clusters are calculated. This procedure is repeated until all taxa are joined in a single cluster. A simple hierarchical clustering method is *unweighted pair group method with arithmetic mean* (UPGMA) [127]. This algorithm assumes a *molecular clock*, i.e. constant mutations rates for all taxa. This makes it possible to calculate a rooted tree. Other popular methods assume minimum evolution [111]. The correct tree is assumed to be the tree with the lowest sum of all branch lengths. A commonly used method is *Neighbor-Joining* (NJ) [112]. This method does not assume a molecular clock and thus builds unrooted trees. As long as the input distance matrix is correct, the algorithm will produce the optimal tree. Even though this is usually not the case, the algorithm will find the optimal tree most of the time. It is also possible to assure that the optimal tree under the minimum evolution assumption is found [70]. Furthermore, there is a popular modification of the *Neighbor-Joining* called *BIONJ* [42]. This method can lead to improved trees in practice.

Tree building methods can return different trees even for small changes in the input. Therefore, it is common practice to calculate bootstrap values [35] for all branches in the phylogenetic tree. One way of doing this is to generate 100 datasets from sampled columns of a multiple sequence alignment. For every dataset, a tree is reconstructed using any of the methods described above. Based on the 100 trees, a consensus tree [16] is built and every branch is assigned the percentage of trees in which this branch appears. Thus, the stability of the phylogenetic tree can be assessed.

1.3 Supertree methods

Is is also possible to build multiple trees, and then built a *supertree* [44, 12] from multiple overlapping phylogenetic trees. As with other problems that need to search the complete tree space, this problem is also NP-hard [129]. Thus, this problem has to be solved heuristically for large trees. One such heuristical method called *ASTRAL* [153] tries to find the correct phylogenetic tree in case of *incomplete lineage sorting* (ILS) [87]. For a set of genes, the individual gene trees may be different from each other, for example due to horizontal gene transfer. This tool reconstructs the phylogeny under the multi-species coalescent model [65] and can thus lead to better results than a simple super matrix approach. The supertree is inferred from the sets of quartet trees that are given by the gene trees. Quartet trees are the smallest trees that contain phylogenetic information in an unrooted setting. In case the outgroup is known, a rooted triplet tree could be considered the fundamental phylogenetic unit. Information for the supertree can be obtained from a quartet tree as it indicates a split between the two sister nodes from one side and the two sister nodes on the other side. There are three possible splits that can be indicated by a quartet tree topology.

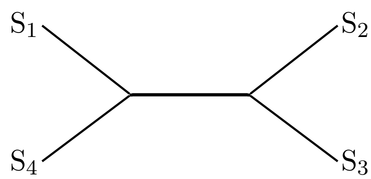

Figure 2: Example of a quartet tree.

There are also other use cases for supertree methods. It is possible to calculate supertrees from a lot of small trees that do not necessarily correspond to any gene. One summary method that also uses quartet trees is *Quartet MaxCut* [126, 125]. This method attempts to find the tree that is consistent with as many input quartet trees as possible. Since this *maximum quartet consistency* problem is NP-hard [129] and can therefore not be solved directly, Quartet MaxCut tries to solve a related problem heuristically. Removing any edge in a tree will split or *cut* the tree into subtrees. For every non-trivial split, i.e. each subtree consists of at least two nodes, there can be potentially many quartet trees with exactly two species from each subtree. If the split indicated by the quartet tree coincides with an edge in the supertree, then a cut caused by removing this edge is *satisfied* by the quartet tree. Otherwise, the quartet tree is *violated* by the cut. Finding the edge that has the highest ratio of satisfied to violated quartet trees, or respectively the cut that has the maximum support, is called the *MaxCut* problem.

Algorithmically, this problem is solved by defining good and bad edges for every input

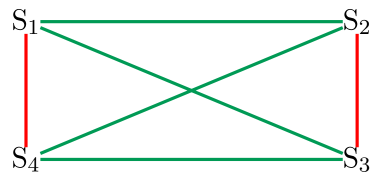

Figure 3: Based on the quartet tree in Figure 2, good are edges are shown in green and bad edges are shown in red. Bad edges should not be cut as this would result in a split that is not indicated by the underlying quartet tree.

quartet tree, as shown in Figure 3. All four nodes are connected with an edge. These edges should signal whether this quartet tree is being satisfied or violated by a cut in the supertree. As splitting sister nodes on either sides would clearly violate the quartet tree, the two edges connecting these nodes are considered to be bad edges. All other edges are good edges. In order to find the super tree, an empty multigraph is created that has one node for every species. In the next step, the previously defined edges from all input quartet trees are inserted into the multigraph. As shown in Figure 4, the algorithm tries to find the maximum cut with the best ratio of good to bad edges. This is solved efficiently with semidefinite programming.

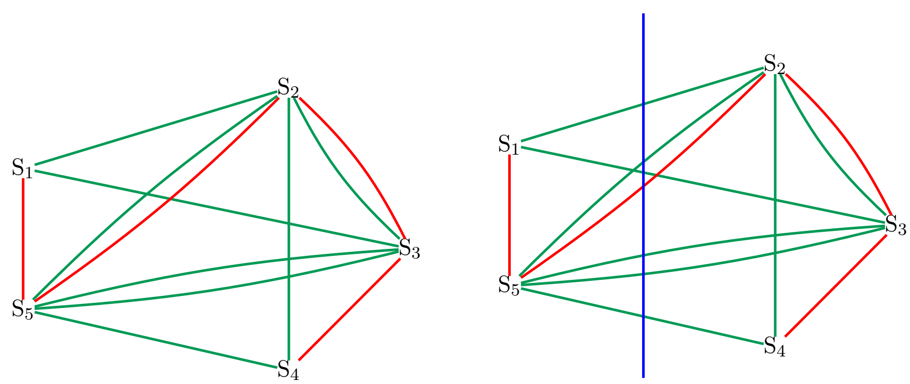

Figure 4: Good and bad edges are inserted into a multigraph with 5 sequences. On the right, the maximum cut is shown in blue. This cut has the ratio of 6 good edges to 1 bad edge. It indicates an edge in the supertree, between the sequences (S_1, S_5) and (S_2, S_3, S_4).

After finding the maximum cut, the set of sequences are divided into two subsets according to the cut. Further, the corresponding edge is inserted into the supertree. Afterwards, the algorithm is repeated on each subtree until the supertree is fully resolved or the information contained in the input quartet trees is not sufficient to find more maximum cuts. Thus, *Quartet MaxCut* is a divide-and-conquer algorithm. It can correctly reconstruct trees for 100 taxa with an input of 1 million quartet trees drawn uniformly at random, even if the error rate is 30% [126]. In practice, quartet trees that do not conform to the supertree topology do not have to be wrong because the conflicting topologies may be caused by ILS. In order to give confidence values to the input quartets, it is also possible to specify weights for every quartet tree [6]. Another extension to this idea is *Triplet MaxCut* [118].

Quartet MaxCut has been compared to other supertree methods [132, 7] and it scales well with regard to number of quartets, error rates and number of taxa. The most significant advantage of this method is its speed. For millions of input quartets, *Quartet MaxCut* terminates in less than a second while other methods require many hours or even days. *Quartet MaxCut* has been compared to the most used supertree method *matrix representations with parsimony (MRP)* [105, 9]. This method defines a $n \times m$ matrix where n is the number of species and m the number of splits in any of the input trees. In every column, every species on one side of the split is marked with a '0' and all other species with a '1'. The supertree is then built with *Maximum-Parsimony*. Due to the fact that this method uses MP, it is prohibitively slow for large numbers of input trees.

The scalability of supertree methods can be improved. *SuperFine* [131] merges the input trees together with a *strict consensus merger* [59] which is very fast. However, there will likely be conflicts which will occur for a usually small subset of species. In this case, a more sophisticated method such as *Quartet MaxCut* or *MRP* is used to resolve these conflicts. Other methods try to divide a large set of species, e.g. to create something like the tree of life, into smaller sets and then attempt to amalgamate the smaller trees into a final supertree which is different from the usual supertree task where the input trees have to be overlapping. A big challenge of this approach is that the input trees have to be very accurate as errors cannot be corrected. Furthermore, a distance matrix is necessary to combine the input trees as *NJMerge* [91] is based on *Neighbor-Joining*.

It is difficult to measure the quality of a phylogenetic tree. This is largely due to the fact that accurate reference trees are necessary in order to even begin a meaningful comparison. It is possible to sidestep this problem by simulating sequences. However, even though the correct phylogenetic tree is known, it is questionable whether the simulated sequences could be considered realistic. Thus, researchers have to rely on widely accepted reference trees.

One of the most commonly used metrics to compare phylogenetic trees is the *Robinson-Foulds distance* [108] which is solely based on the tree topology. In order to calculate this distance, every possible bipartition of the trees have to be considered. A bipartition is created by removing a single edge from the tree. Then, the Robinson-Foulds distance is the number of bipartitions implied by the reconstructed tree, but not implied by the reference plus the number of the reverse. Thus, if both trees are binary trees, i.e. no multifurcations, the distance will always be an even number. The Robinson-Foulds distance can also be normalized by dividing it by $2n-6$ where n is the number of leaves in the tree. This way, the performance of phylogenetic methods can be compared in a more meaningful way. Many more distance measures have been proposed, such as quartet and triplet distances. [113]

1.4 Alignment-free methods

So far, I described different ways to reconstruct accurate phylogenetic trees which are based on sequence alignments in one way or another. While this alignment-based approach is still the preferred way to built trees of high quality, most of the previously described methods try to solve problems that are NP-hard and can therefore not be solved efficiently. Many tools use heuristics and are highly optimized by using as many forms of parallelization as possible. This includes using special processor instructions, so-called vector parallelization, or distributing the work load to computer clusters. Even though this improves the scalability of these tools greatly, nowadays we are in the age of *next generation sequencing* and big data [130], resulting in a tremendous amount of sequence data to be analysed. This has led to a lot of research into how sequences can be compared without requiring a full sequence alignment. Even though I will focus on methods for phylogeny reconstruction in this thesis, alignment-free methods have been developed for many different fields such as metagenomics [4, 18, 78, 90, 134, 136, 144, 149], sequence assembly [152], identification of biomarkers [28], isoform quantification from RNAseq reads [103], analysis of regulatory elements [34, 66, 77], read alignment [1, 68, 79] and protein classification [21, 76, 81, 82]. The main goal of alignment-free methods is to be magnitudes faster than alignment-based approaches while being easy to use. Thus, the user should not be required to identify and align orthologous genes manually. Instead, most tools are used on whole-genome data. Some methods even go one step further and also try to eliminate the genome assembly step. These assembly-free methods [101, 31, 10, 151, 116] can be applied to partially sequenced genomes or even directly to unassembled reads and lead to an even faster sequence analysis.

In general, alignment-free methods calculate pairwise distances. Then, fast methods like *Neighbor-Joining* [112] are used on the distance matrices to built a phylogenetic tree. This is significantly faster than more traditional methods. However, without an alignment, it is a challenging task to find evidence for evolutionary events based on the sequence data. Thus, many methods estimate distances that are not based on a model of evolution and therefore do not reflect the substitutions per position. Instead, they rely on the comparison of simple sequence features such as word frequencies or lengths of common substrings. These features usually have an intuitive correlation with the evolutionary distance. The advantage of this approach is that these distances can be calculated very fast. However, even when reflect the substitions per position under some model of evolution, alignment-free methods are still less accurate than alignment-based methods. In the following, I want to give a broad overview over alignment-free methods and describe some of the methods, that are relevant for the rest of the thesis, in more detail. For a more detailed overview, there are several reviews of alignment-free methods [142, 47, 11]. Furthermore, the *AF-Project* [155] is a comprehensive benchmark of alignment-free methods for various different settings.

One of the most commonly used sequence features are k-mer frequencies. k-mers are words of length k. The early alignment-free methods mostly used short k-mers, i.e. dinucleotide (2-mers) and trinucleotide (3-mers) frequencies. The distances can be calculated e.g. with the X_2 [13] or D_2 [83] statistic. The latter distance is the inner product of two frequency vectors which is also the number of k-mer matches. Especially comparing 3-mer frequencies is an intuitively meaningful feature as 3-mers correspond to codons and are thus translated to amino acids. However, if non-coding regions are included, which is the case for whole-genome data, then the accuracy of using short k-mers decreases. The optimal length of k-mers has been investigated [123], but no clear answer has emerged. *Feature frequency profiles* (FFP) [123] is a very fast method that uses a range of word lengths which was found empirically. The authors used the *Jensen-Shannon divergence* [80] to calculate a distance between the frequency profiles. Apart from this commonly used distance, it is also possible to use the *Euclidean distance*. Even more distance measures are provided in *CAFE* [85].

These methods have one problem in common. The frequency measure is increasingly unreliable when the sequences have different background frequencies which is the case when there are unrelated regions in the sequences. One methods that tries to correct the frequency measure by subtracting predicted background frequencies is *CVTree* [104].

These alignment-free methods are already very fast. However, it is possible to improve the runtime even more if the sequences are not compared in their entirety. A popular method that makes use of this approach and can thus handle even very large datasets is *Mash* [101]. For every sequence, only a small *sketch* is stored which can be very small in comparison to the whole sequence. The sketches are based on MinHashes [15] and are the smallest k-mers according to a simple hash function. The distances are then calculated using the Jaccard index which is the fraction of shared k-mers. This distance can be accurate even for a relatively small sample of k-mers. At some point, using more k-mers will most likely not influence the distance anymore.

Instead of contiguous k-mers, it is also possible to compare inexact k-mers. For database searches, word matches are used as a starting point – or seed – for a specialized version of the Smith-Waterman algorithm. It has been shown that inexact word matches, so-called *spaced seeds*, can be used to improve homology searches [86]. This finding has then inspired the use of *spaced-words* [14] (see Table 2 for an example).

Sequence	A	C	A	G	T	T	A	C	A	G	C	T
Pattern			1	1	0	1	0	0	1	1		
Spaced-Word			A	G	*	T	*	*	A	G		

Table 2: For a given sequence and binary pattern, the *spaced-word* starting at position 3 is shown. The characters at the *don't care positions* are replaced by wildcard characters.

A spaced-word is defined together with a binary pattern of the same length. This pattern defines *don't care positions* and match positions. The number of match positions is called the weight of pattern. For a given pattern, a spaced-word is a substring of a sequence where the characters at the *don't care positions* are replaced by a wildcard symbol. The distance based on spaced-word frequencies can be calculated similar to the contiguous case. The accuracy of this distance measure can be improved if multiple patterns are used [72], even though this leads to increased runtimes. Furthermore, it is possible to estimate the substitutions per position under the *Jukes&Cantor* model, based on spaced-word matches [95]. A third way of estimating distances from spaced-words is based on the fact that the number of spaced-words decreases when the weight of the pattern is increased. This can be used to define a function and distances can be estimated based on the slope in a certain range [109].

As the pattern can influence the performance of such a method, it is an obvious question whether patterns can be improved. *SpEED* [61] and *rasbhari* [46] are two tools that can optimize pattern sets. These methods optimize the overlap complexity [60]. This is a measure based on the number of overlapping match positions when the patterns are shifted against each other. This can lead to more stable values for the number of spaced-word matches [95]. However, the effect of this optimization is marginal for a single pattern. Moreover, there is no known algorithm that can optimize a pattern for a specific dataset.

The number of word matches is not the only sequence feature that can be used to estimate distances. Another class of alignment-free methods is based on the lengths of word matches. Intuitively, there should be longer words when the sequences are closely related because every substitution, insertion or deletion in one sequence breaks up previous word matches. In order to calculate distances, one possible approach is to use data compression algorithms such as the *Lempel-Ziv factorization* [156]. This has been utilized in *average common substring* (ACS) [141] which finds the longest common substring starting from each position in both sequences. The average length of these common substrings is then used as the distance between the two sequences.

S_1	C	G	A	A	T	C	C	A	C	G	A
S_2		C	A	C	G	G	A	T	A	C	

Table 3: The longest common substring with $k = 2$ mismatches for position 1 in the first sequence is found at position 3 in the second sequence. The length of this substring is 6. The average length of these k-mismatch common substrings is the distance calculated by *kmacs*.

This approach has been extended to use inexact common substrings. In *kmacs* [71], each substring can have up to k mismatches where k is a user-defined parameter. Since these substrings cannot be found efficiently, *kmacs* first finds the longest common substring for each position and then extends these substrings until k mismatches are found (see Table 3 for an example). Then, the distance is calculated analogous to *ACS*. By using inexact common substrings, the results can be improved. However, it is unclear how a good value for k can be determined. *kmacs* is also available as a webserver [55], as well as *spaced* (word frequencies). Neither *ACS* nor *kmacs* estimate evolutionary distances. In case of *kmacs*, it has been shown that the length distribution of the extension after the longest common substring can be used to estimate substitutions per position accurately [96]. This is similar to the first method that tried to estimate evolutionary distances, *kr* [49]. This method uses the lengths of *shortest unique substrings*, so-called shustrings [48], between two sequences for its estimation. Shustrings are an extension of the longest common substring by one.

There is one more group of alignment-free methods that follow a different approach. They find pairwise gap-free micro-alignments. Instead of counting these matches as in some previously described methods, the distances are calculated from the number of observed mismatches per position in the micro-alignments. This is analogous to the distances based on pairwise sequence alignments. Obviously, these tools do not conform perfectly to the definition of alignment-free methods. However, they do not calculate a full sequence alignment. Instead, the micro-alignments are based on word matches in one way or another which can be found quickly. Thus, it is justified to still consider them to be alignment-free methods. One major challenge for such an approach is that the micro-alignments have to be homologous. Otherwise, the distances would be based on incidental alignments and likely be inaccurate. Thus, these approaches have to find a way to deal with random matches as well as homoplasy. In the following, I will describe three tools in more detail.

The first method to utilize micro-alignments was *co-phylog* [151]. The micro-alignment follows a structure consisting of an *object* (O_i) which denotes the i central position(s) of the micro-alignment. This object is flanked by a *context* ($C_{j,j}$) which consists of two j-mers (see Table 4 for an example). The structure is written as $C_{j,j}O_i$ which inspired the name *co-phylog*.

S_1		T	G	C	A	A	G	C	G
S_2	G	C	G	C	T	A	G	C	

Table 4: Example of a microalignment in *co-phylog*. The structure is $C_{2,2}O_1$. The context is colored in orange and the object is colored in violet.

For two sequences, all k-mers are rapidly found and split according to the predefined structure. k-mers are matched according to the contexts. Then, the mismatches per position can be observed from the objects. In this method, this value is used as the distance and not corrected with e.g. the *Jukes&Cantor* model. Since the context pairs are supposed to make up a micro-alignment, the context has to be unique in both sequences. Otherwise, it cannot be established which two contexts should be matched with each other. *co-phylog* discards any context that appears multiple times in a sequence to solve this issue. For large sequences, there is a high number of expected random matches which can lead to homologous matches to be overlooked or random matches to be used for the distance estimation, depending on the structure that is being used. In order to circumvent this problem, *co-phylog* uses contexts that are sufficiently large. By default, the structure used in the program is $C_{9,9}O_1$. Thus, the length of the k-mers is 19. While this length is sufficient to ensure that the matching context are homologous, the expected number of micro-alignments for more distantly related sequences is fairly low. This can reduce the accuracy of the distance to the point where distances can no longer be estimated. However, it should be noted that *co-phylog* is intended to be used on more closely related sequences. The structures are likely inspired by SNP's (single nucleotide polymorphisms) since the default size of the objects is 1.

S_1		G	C	T	A	G	C	G	A	A	T	C	T
S_2	C	G	C	T	A	G	T	C	A	A	T	C	

Table 5: Example of a microalignment in *andi*. The two anchors are colored in orange. They are equidistant, i.e. 4 positions apart. The violet part of the alignment is used to estimate the substitutions per position under the *Jukes&Cantor* model.

A slightly more flexible, yet very similar method is called *andi* [50] which is an acronym for *anchor distance*. It is similar in the sense that it uses flanking word matches - or *anchors* - to ensure that the micro-alignments are homologous and it calculates pairwise distances based on the possibly mismatching nucleotides in-between. However, there is no fixed pattern or length of the micro-alignments. Instead, the anchors have to be maximal unique word matches that have at least a certain length. The minimum length is based on a statistical analysis of the distribution of how many word matches are expected to be found in unrelated sequences. The length of the anchors is required to be longer than at least 97.5 % of this distribution. As the anchors are maximal and unique, there is no ambiguity as to which words are matched with each other. The two anchors have to be equidistant from each other, i.e. the interjacent part has to have the same length in both sequences. Otherwise, it would be impossible to arrange them in a micro-alignment and calculate a meaningful distance. In order to ensure that the entire micro-alignment is homologous, there is also a maximum distance that the anchors can be apart from each other. *andi* is similar to *co-phylog* and it has the same drawbacks. Therefore, it is also intended to be used on closely related species. In fact, the authors use their method on different strains of the same species of bacteria, *Escherichia coli*, in their evaluation. An example is shown in Table 5.

Unlike *co-phylog*, *andi* uses the *Jukes&Cantor* model to account for back substitutions. Thus, the distance accurately reflects the substitutions per positions, at least up to 0.5 substitutions per position. Above this threshold, less and less equidistant anchor pairs can be found and at some point, the distance cannot be estimated. The algorithm of *andi* is based on suffix arrays [88] and *range minimum queries* [39]. It is one of the fastest alignment-free methods and, unlike *co-phylog*, it is efficiently parallelized and can be used on large datasets. Moreover, it is a much more user-friendly tool. It can, therefore, be seen as a generalization and an overall improvement over *co-phylog*. However, one advantage of *co-phylog* is that it can be applied to unassembled reads and can thus be considered assembly-free.

1.5 Filtered spaced word matches

The previously mentioned methods that utilized spaced-words, either compared their frequencies or used the number of spaced-word matches to estimate distances. *Filtered spaced word matches* (FSWM) [73] is a different approach that is based on micro-alignments. These micro-alignments are *spaced-word matches* (see Table 6 for an example). For a spaced-word match to occur, the match positions of two spaced-words have to coincide. The match positions are defined by a single binary pattern. The remaining positions, the *don't care positions*, can have mismatching nucleotides. The structure used in *co-phylog* can also be called a spaced-word match with a fixed pattern. The advantage of using spaced-word matches is that, while the expected number of matches is the same as with exact word matches, the matches are more evenly spread across the sequences. This is because two consecutive spaced-words are statistically less dependent on each other.

In order to find a spaced-word match in a quick and easy way, a list of all spaced-words for two sequences is sorted lexicographically. All occurrences of a spaced-word are then next to each other.

S_1		T	G	C	A	A	G	C	T	G	T
S_2	G	C	G	C	T	A	G	A	T	C	
Pattern			1	0	0	1	1	0	1		

Table 6: Example of a microalignment in *FSWM*. There are two mismatches at the *don't care positions* and only one match. Therefore, such a match would be discarded as a random match.

The nucleotides at the *don't care positions* are used to score possible spaced-word matches, before the distance is calculated. The score is the sum of substitution scores defined in a substitution matrix for alignments of DNA sequences [19]. A histogram of these scores is shown in Figure 6. Random spaced-word matches are expected to have a negative score depending on the number of *don't care positions*. These matches constitute the peak on the left of the histogram. In most cases, the scores of the background matches are not positive. Therefore, by discarding matches with negative scores, the background noise can be filtered out effectively. This is the novelty introduced in *FSWM*. The filtering step makes it possible to use patterns with lower weights. This is due to the fact that the expected number of background matches grows quadratically with the length of the sequences while the expected number of homologous matches only grows linearly. Thus, the distances would be dominated by background noise for longer and more distantly related sequences

Spaced-word	Position
⋮	
A∗∗TC∗C∗A	1053
A∗∗TC∗C∗T	236
A∗∗TC∗C∗T	843
A∗∗TC∗C∗T	54
A∗∗TC∗C∗T	2524
A∗∗TC∗C∗T	214
A∗∗TC∗G∗A	843
⋮	

Spaced-word	Position
⋮	
A∗∗TC∗C∗A	648
A∗∗TC∗C∗T	45
A∗∗TC∗C∗T	765
A∗∗TC∗C∗T	55
A∗∗TC∗C∗A	1555
A∗∗TC∗C∗A	187
A∗∗TC∗G∗A	698
⋮	

Figure 5: For two sequences, a lexicographically sorted list of spaced-words can be used to find all possible spaced-word matches. In this example, there are 5 occurrences of the spaced-word 'A∗∗TC∗C∗T' in the first sequence and 3 occurrences in the second one. Thus, there can be at most 3 homologous micro-alignments.

if patterns with a low weight are used without the filtering step.

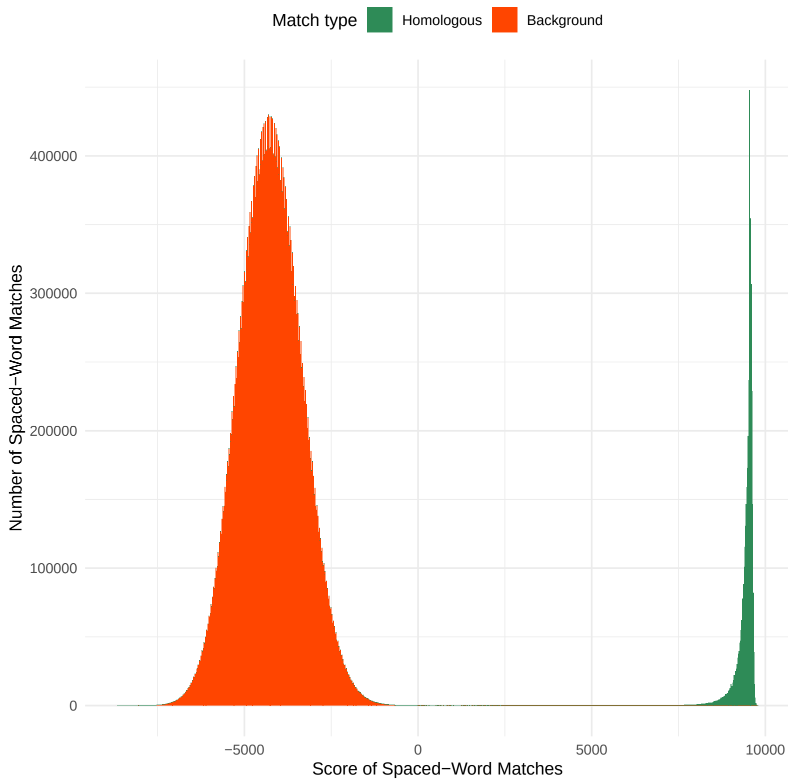

Figure 6: Every spaced-word match can be scored according to the nucleotides at the *don't care positions* with respect to a pre-defined binary pattern. This figure shows a histogram for all spaced-word matches from a comparison of two bacterial genomes, *Escherichia coli UMN026* and *Escherichia coli IAI39*. The left peak shows the distribution of random spaced-word matches which is approximately normally-distributed. On the right, there is a peak for the homologous spaced-word matches. Spaced-word matches with negative scores are shown in red. Removing these matches can reliably eliminate the background noise.

In contrast to *co-phylog* and *andi*, *FSWM* does not require the spaced-words to be unique in both sequences. The assignment is resolved with *1-to-1 mapping*. Since a score for every potential spaced-word match is calculated in the filtering step, this information can also be used to match each spaced-word with at most one spaced-word in the other sequence. This is done with a greedy algorithm. Afterwards, the substitutions per position are estimated based on the *Jukes&Cantor* model.

While *FSWM* is still a fast alignment-free method, the filtering step is $\mathcal{O}(n^2)$ for n occurrences of a given spaced-word. For large datasets, this can significantly slow down this method. On the plus side, *FSWM* can estimate evolutionary distance accurately up to 1 substitutions per position. The *FSWM* approach has led to a lot of further research. It has been applied to protein sequences in *Prot-SpaM* [75]. Furthermore, it has been shown that *FSWM* can also be used as an assembly-free method in *Read-SpaM* [69].

1.6 Objectives of this thesis and overview

So far, all alignment-free methods estimate pairwise distances and build a phylogenetic tree based on a distance matrix. In this thesis, I pursue the question whether multiple sequence comparison can be used for an alignment-free method. While there have been attempts in that direction [106], the goal of this thesis is to develop a completely different approach based on micro-alignments. This method extends *FSWM* to multiple sequence comparison. Its filtering procedure is used to ensure that so-called *blocks* – small, gap-free micro-alignments involving four different taxa – are homologous. I tried to apply slower, but more accurate character-based methods to these blocks in order to find out whether the accuracy of alignment-free methods can be improved this way. Something similar has been tried in a supermatrix approach [114]. Here, I developed *Multi-SpaM* [26] which calculates the optimal tree topology for every block of spaced-word matches and uses *Quartet MaxCut* to amalgamate the quartet trees into a supertree. This method is described in Chapter 2.

In Chapter 3, we compared pairs of adjacent blocks involving the same four sequences. We calculated the distance between two blocks in each sequence and use this information in order to find putative insertions and deletions. This information can be used as parsimony-informative characters or to define quartet trees. We used this novel approach to build or improve phylogenetic trees.

Chapter 4 describes another method based on spaced-words to which I contributed. Here, spaced-word matches are used as anchor points for a genome aligner. These anchor could be used to improve the alignments for distantly related species.

Furthermore, I show a few experiments that could improve *Multi-SpaM* under some conditions. In a modified version of *Multi-SpaM*, *Neighbor-Joining* is used instead of *RAxML* to find the optimal quartet tree topologies. Moreover, I gave each quartet tree a weight in order to improve the overall phylogenetic tree. The results can be found in Chapter 5.

Finally, the overall results are discussed in Chapter 6. I discuss severall limitations and show a few options on how they can be remedied. This outlook is given in Chapter 7.

2 'Multi-SpaM': a maximum-likelihood approach to phylogeny reconstruction using multiple spaced-word matches and quartet trees

Reference

Thomas Dencker, Chris-André Leimeister, Michael Gerth, Christoph Bleidorn, Sagi Snir, Burkhard Morgenstern, 'Multi-SpaM': a maximum-likelihood approach to phylogeny reconstruction using multiple spaced-word matches and quartet trees, NAR Genomics and Bioinformatics, Volume 2, Issue 1, March 2020, `https://doi.org/10.1093/nargab/lqz013`. [26]

We published an earlier version of this paper at the 'RECOMB-CG' conference [25].

Original Contribution

TD designed the study, developed and implemented *Multi-SpaM* and performed the program evaluation. CAL and SS assisted in the study design. MG and CB contributed to the program evaluation. BM supervised the project. TD wrote the manuscript and BM improved parts of the manuscript. All authors read and approved the final version of the manuscript.

Published online 30 October 2019 *NAR Genomics and Bioinformatics, 2020, Vol. 2, No. 1* **1**
doi: 10.1093/nargab/lqz013

'Multi-SpaM': a maximum-likelihood approach to phylogeny reconstruction using multiple spaced-word matches and quartet trees

Thomas Dencker[1], Chris-André Leimeister[1], Michael Gerth[2], Christoph Bleidorn[3,4], Sagi Snir[5] and Burkhard Morgenstern[1,6,*]

[1]Department of Bioinformatics, Institute of Microbiology and Genetics, Universität Göttingen, Goldschmidtstr. 1, 37077 Göttingen, Germany, [2]Institute for Integrative Biology, University of Liverpool, Biosciences Building, Crown Street, L69 7ZB Liverpool, UK, [3]Department of Animal Evolution and Biodiversity, Universität Göttingen, Untere Karspüle 2, 37073 Göttingen, Germany, [4]Museo Nacional de Ciencias Naturales, Spanish National Research Council (CSIC), 28006 Madrid, Spain, [5]Institute of Evolution, Department of Evolutionary and Environmental Biology, University of Haifa, 199 Aba Khoushy Ave. Mount Carmel, Haifa, Israel and [6]Göttingen Center of Molecular Biosciences (GZMB), Justus-von-Liebig-Weg 11, 37077 Göttingen, Germany

Received April 30, 2019; Revised July 31, 2019; Editorial Decision October 03, 2019; Accepted October 13, 2019

ABSTRACT

Word-based or 'alignment-free' methods for phylogeny inference have become popular in recent years. These methods are much faster than traditional, alignment-based approaches, but they are generally less accurate. Most alignment-free methods calculate 'pairwise' distances between nucleic-acid or protein sequences; these distance values can then be used as input for tree-reconstruction programs such as neighbor-joining. In this paper, we propose the first word-based phylogeny approach that is based on 'multiple' sequence comparison and 'maximum likelihood'. Our algorithm first samples small, gap-free alignments involving four taxa each. For each of these alignments, it then calculates a quartet tree and, finally, the program 'Quartet MaxCut' is used to infer a super tree for the full set of input taxa from the calculated quartet trees. Experimental results show that trees produced with our approach are of high quality.

INTRODUCTION

Sequence-based phylogeny reconstruction is a fundamental task in computational biology. Standard phylogeny methods rely on 'sequence alignments' of either entire genomes or sets of orthologous genes or proteins. 'Character-based' methods such as 'Maximum Parsimony' (1,2) or 'Maximum Likelihood' (3) infer trees based on evolutionary substitution events that may have happened since the species evolved from their last common ancestor. These methods are generally considered to be accurate as long as the underlying alignment is of high quality and as long as suitable substitution models are used. However, for the task of multiple alignment no exact polynomial-time algorithm exists, and even heuristic approaches are relatively time consuming (4). Similarly, exact algorithms for character-based approaches are known to be 'NP hard' (5,6).

'Distance' methods, by contrast, infer phylogenies by estimating evolutionary distances for all pairs of input taxa. Here, pairwise alignments are sufficient and can be faster calculated than multiple alignments, but still require run time proportional to the product of the lengths of the aligned sequences. However, there is a loss in accuracy compared to character-based approaches, as all information about evolutionary events is reduced to a single number for each pair of taxa, and not more than two sequences are considered simultaneously, as opposed to character-based approaches, where all sequences are examined simultaneously. The final trees are obtained by clustering based on the distance matrices, most commonly with 'Neighbor Joining (NJ)' (7) or 'BIONJ' (8). Since both pairwise and multiple sequence alignment is computationally expensive, they are ill-suited for the increasingly large data sets that are available today due to the next-generation sequencing techniques.

In recent years, a large number of fast 'alignment-free' methods have been proposed for phylogeny reconstruction, see (9–15) for review articles. Some of these approaches are using some sort of 'micro-alignments' and infer phylogenetic distances from the number of mismatches in these simplified alignments. So, strictly-spoken, these methods are not 'alignment-free', but most authors refer to them as 'alignment-free' anyway, since 'micro-alignments' can be

*To whom correspondence should be addressed. Tel: +49 551 39 14628; Email: bmorgen@gwdg.de

found by rapid pattern-matching algorithms, avoiding the need to calculate full alignments of the compared sequences.

Another advantage of the so-called 'alignment-free' methods for genome comparison is that they can circumvent common problems of alignment-based approaches such as genome rearrangements and duplications. Moreover, many alignment-free methods can be applied not only to entire genomes, but also to partially sequenced genomes or even to unassembled reads (16–22). A disadvantage of these methods is that they are often considerably less accurate than slower, alignment-based methods. A systematic evaluation of existing alignment-free methods for a variety of different application scenarios has been carried out in the 'AFproject' (23).

'co-phylog' (18) is a recently proposed 'alignment-free' method that is based on 'micro alignments'. This approach finds short, gap-free alignments of a fixed length, consisting of matching nucleotide pairs only—except for the middle position in each alignment, where mismatches are allowed. Phylogenetic distances are estimated from the fraction of such alignments for which the middle position is a mismatch. As a generalization of this approach, 'andi' (24) uses pairs of maximal exact word matches that have the same distance to each other in both sequences and uses the frequency of mismatches in the segments between those matches to estimate the number of substitutions per position between two input sequences. A further development of this approach is 'phylonium' (25).

Since 'co-phylog' and 'andi' require a minimum length of the flanking word matches in order to reduce the number of matches that are mere random background matches, they do not perform well on distantly related sequences where fewer exact matches with the required minimum length can be found, if any at all. Moreover, the number of random segment matches grows quadratically with the length of the input sequences while the expected number of homologous matches grows only linearly. Thus, the minimum match length must be increased in these approaches if long sequences are to be compared to limit the number of background matches. This, in turn, reduces the number of homologous segment matches that are found, and therefore the amount of information that is available to estimate phylogenetic distances.

Other alignment-free approaches are based on the length of maximal common substrings between sequences that can be rapidly found using suffix trees or related data structures (26,27). As a generalization of this approach, some methods use longest common substrings with a certain number of mismatches (28–32). Finally, methods have been proposed that estimate phylogenetic distances from the decay of the number of word matches as a function of the word length (33,34).

In previous publications, we proposed to use words with 'wildcard characters'—so-called 'spaced words'—for alignment-free sequence comparison (35–37). Here, a binary pattern of 'match' and 'don't-care' positions specifies the positions of the 'wildcard' characters, see also (38–40). In 'Filtered Spaced-Word Matches (FSWM)' (41) and 'Proteome-based Spaced-Word Matches (Prot-SpaM)' (42), alignments of such spaced words are used where sequence positions must match at the 'match' positions while mismatches are allowed at the 'don't-care positions', see Figure 1. A score is calculated for every such spaced-word match in order to remove—or 'filter out'—'background' spaced-word matches; the mismatch frequency of the remaining 'homologous' spaced-word matches is then used to estimate the number of substitutions per position that happened since two sequences evolved from their last common ancestor. The filtering step allows us to use patterns with fewer match positions in comparison to above mentioned methods 'co-phylog' and 'andi', since the vast majority of the background noise can be eliminated reliably. Consequently, phylogenetic distances calculated with 'FSWM' and 'Prot-SpaM' are still accurate, if large and distantly related sequences are compared.

S_1: T A **C T** A **G** C G **T** C G
S_2: A C T C **C T** A **G** T G **T** T G

Figure 1. Spaced-word match W with respect to a pattern $P = 1101001$ of weight $w = 4$. W can be seen as a gap-free pairwise alignment that has the same length as P, with matching nucleotide at the four 'match positions' and possible mismatches at the three 'don't-care' positions.

S_1: C **C C** A A **G** G A C
S_2: A A C T A C G T A C C T
S_3: A A C T A C G T A C C
S_4: **C C** A C **G** T C C G C G
S_5: A G A C T C **C C** A A **G** G A
S_6: T C **C C** A T **G** G A C C
S_7: A A C T A C G T A C C A
1 2 3 4 5 6 7 8 9 10 11 12 13

Figure 2. P-block for a pattern $P = 11001$: the spaced word $W = CC{*}{*}G$ occurs at [S_1, 2], [S_4, 1], [S_5, 7] and [S_6, 3].

In the present paper, we introduce a novel approach to phylogeny reconstruction called 'Multiple Spaced-Word Matches (Multi-SpaM)' that combines the 'speed' of the so-called 'alignment-free' methods with the 'accuracy' of the 'Maximum-Likelihood' approach. While other alignment-free methods are limited to 'pairwise' sequence comparison, we generalize our previous 'FSWM' approach to 'multiple' sequence comparison. For a binary pattern P representing 'match' and 'don't-care' positions, 'Multi-SpaM' identifies so-called 'P-blocks' consisting of four matching spaced words from four different sequences each. That is, a P-block can be seen as a gap-free 'micro alignment' of four different sequences, with matching nucleotides at the 'match' positions of the underlying binary pattern and possible mismatches at its 'don't-care' positions, see Figure 2 for an example. For each such P-block, an optimal 'Maximum-Likelihood' tree topology is calculated with the software 'RAxML' (43). We then use the 'Quartet MaxCut' algorithm (44) to obtain a super tree from the calculated quartet tree topologies. We show that on both simulated and real data, 'Multi-SpaM' produces phylogenetic trees of high quality and often outperforms other alignment-free methods. An earlier version of the present paper has been published in the proceedings of the 'RECOMB-CG' conference (45).

MATERIALS AND METHODS

Spaced words and *P*-blocks

To describe 'Multi-SpaM', we first need to introduce some formal definitions. We want to compare sequences over an alphabet $\mathcal{A}$; since our approach is dealing with DNA sequences, our alphabet is $\mathcal{A} = \{A, C, G, T\}$. For a pattern length ℓ and a binary pattern $P \in \{0, 1\}^\ell$, a 'spaced word' with respect to P is a word W of length ℓ over $\mathcal{A} \cup \{*\}$, such that $W(i) = *$ if and only if $P(i) = 0$. A spaced word W can be considered as a regular expression where '*' is a 'wildcard character'. A position $i \in \{1, \ldots, \ell\}$ is called a 'match position' if $P(i) = 1$ and a 'don't-care position' otherwise. The number of match positions in P is called the 'weight' of P. For a DNA Sequence S of length n and a position $1 \leq i \leq n - \ell + 1$, we say that a 'spaced word' W with respect to P occurs in S at position i – or that $[S, i]$ is an 'occurrence' of W – if $S(i + j - 1) = W(j)$ for all match positions j of P. This corresponds to the definition previously used in (35) and (37).

A pair $([S, i], [S', i'])$ of occurrences of the same spaced word W is called a 'spaced-word match'. For a substitution matrix assigning a 'score' $s(X, Y)$ to every pair (X, Y) of nucleotides, we define the 'score' of a spaced-word match $([S, i], [S', i'])$ of length ℓ as

$$\sum_{1 \leq k \leq \ell} s(S(i + k - 1), S'(i' + k - 1))$$

That is, if we align the two occurrences of W to each other, the score of the spaced-word match is the sum of the scores of the nucleotides aligned to each other. In 'Multi-SpaM', we are using the following nucleotide substitution matrix that has been proposed in (46):

$$\begin{array}{lrrrr} & A & C & G & T \\ A & 91 & -114 & -31 & -123 \\ C & & 100 & -125 & -31 \\ G & & & 100 & -114 \\ T & & & & 91 \end{array} \qquad (1)$$

'Multi-SpaM' starts with generating a binary pattern P with user-defined length ℓ and weight w; by default, we use values $\ell = 110$ and $w = 10$, i.e. by default the pattern has 10 'match positions' and 100 'don't-care' positions. We are using a low 'weight' to obtain a large number of spaced-word matches when comparing two sequences. This includes necessarily a high proportion of random spaced-word matches. The high number of 'don't-care' positions, on the other hand, allows us to accurately distinguish between 'homologous' and 'background' spaced-word matches.

Given these parameters, a pattern P with minimal 'overlap complexity (OC)' is calculated by running our previously developed software tool 'rasbhari' (47). The OC of a pattern or a set of patterns is defined in terms of the number of overlapping 1's if the patterns are shifted against themselves and against each other, for multiple pattern sets. It has been shown that the OC of patterns is closely related to their 'sensitivity' in database searching (48,49) and to the statistical stability of the number of spaced-word matches (37).

As a basis for phylogeny reconstruction, we are using four-way 'micro alignments' consisting of occurrences of the same spaced word with respect to P in four different sequences or their reverse complements. We call such an alignment a 'quartet P-block' or a P-block, for short. A P-block is, thus, a gap-free alignment of length ℓ where in the k-th column identical nucleotides are aligned if k is a 'match' position in P, while mismatches are possible if k is a 'don't-care' position (see Figure 2). 'Multi-SpaM' considers P-blocks involving spaced words from both strands of the input sequences. It is clear that the number of P-blocks can be very large: if there are n occurrences of a spaced-word W in n different sequences, then this gives rise to $\binom{n}{4}$ different P-blocks. Thus, instead of using all possible P-blocks, 'Multi-SpaM' randomly samples a limited number of P-blocks to keep the program run time under control.

For phylogeny reconstruction, we want to use P-blocks that are likely to represent true homologies. Therefore, we introduce the following definition: a P-block is called 'homologous' if it contains at least 'one' spaced-word occurrence $[S, i]$, such that each of the three spaced-word matches of $[S, i]$ with the remaining occurrences has a positive score. Note that a 'homologous' P-block in the sense of this formal definition is, of course, not necessarily 'homologous' in the usual sense, i.e. the four involved sequence segments are not necessarily derived from one common anchestral segment. To sample a list of homologous P-blocks in the sense of our definition, we randomly select spaced-word occurrences with respect to P from the input sequences and their reverse complements. For each selected $[S, i]$, we then randomly select occurrences of the same spaced word from sequences $S' \neq S$, until we have found three occurrences of W from three different sequences that all have positive scores with $[S, i]$.

To find spaced-word matches efficiently, we first sort the list of all spaced-word occurrences with respect to P in lexicographic order, such that all occurrences of the same spaced word appear as a contiguous section of the list. Once we have sampled a homologous P-block as described, we remove the four spaced-word occurrences from our list, so no two of the sampled P-blocks can contain the same occurrence of a spaced word. The algorithm continues to sample P-blocks until no further P-blocks can be found, or until a maximal number M of P-blocks is reached. By default, 'Multi-SpaM' uses a maximum of $M = 1\,000\,000$ P-blocks, but this parameter can be adjusted by the user.

Quartet trees

For each of the sampled quartet P-blocks, we infer an unrooted binary tree topology. This most basic phylogenetic unit is called a 'quartet' topology; there are three different quartet topologies for a set of four taxa. To identify the best of these three topologies, we use the 'Maximum-Likelihood' software 'RAxML' (43) with the 'GTR' model (50). This corresponds to using the commandline version of 'RAxML' with the option `'-m GTRGAMMA -f q -p 12345'`. We note that 'RAxML' is a general 'maximum-likelihood' software, its use in our context is fairly degenerated, as we only use it to infer optimal quartet topologies.

After obtaining the optimal quartet topology for each of the sampled *P*-blocks, we need to amalgamate them into a single tree spanning the entire taxa set. This task is called the 'Supertree Task' (51) and is known to be 'NP hard', even for the special case where the input is limited to quartets topologies, as in our case (52). Nevertheless there are several heuristics for this task, with 'MRP' (53,54) the most popular. Here we chose to use 'Quartet MaxCut' (44,55) that proved to be faster and more accurate for this kind of input (56). In brief, 'Quartet MaxCut' recursively partitions the taxa set, where each such partition defines a split in the final tree. If, during this process, a set *A* of taxa is to be split into two subsets, the program tries to put neighboring taxa from the quartet trees into the same subset while non-neighboring taxa can end up in different subsets. To achieve this, a multi-graph is defined where the taxa in the set *A* are represented as nodes, and each pair of taxa in each quartet tree is represented as an edge. That is, each quartet tree defines six edges in the multi-graph. Edges between neighboring taxa in a quartet tree are seen as 'good' edges that are to be retained, if possible, while edges between non-neighboring taxa are 'bad' edges that can be removed by the partition. The program then finds a partition that minimizes the ratio between good and bad edges that are to be removed, see (44,55) for details.

Implementation

To keep the run time of our software manageable, we integrated the 'RAxML' code directly into our program code. We parallelized our program with 'openmp' (57).

TEST RESULTS

To evaluate 'Multi-SpaM' and to compare it to other fast, alignment-free methods, we applied these approaches to both simulated and real sequence data and compared the resulting trees to reference trees. In phylogeny reconstruction, artificial benchmark data are often used since here, 'correct' reference trees are known. For the real-world sequence data that we used in our study, we had to rely on reference trees that are believed to reflect the true evolutionary history, or on trees calculated using traditional, alignment-based methods that can be considered to be reasonably accurate. In our test runs, we used standard parameters for all methods, if such parameters were suggested by the respective program authors. The program 'kmacs' (28) that was one of the programs that we evaluated, has no default value for its only parameter, the number k of allowed mismatches in common substrings. Here, we chose a value of $k = 4$. While 'Multi-SpaM' produces tree topologies without branch lengths, all other methods that we evaluated produce distance matrices. To generate trees from these matrices, we used 'Neighbor-joining' (7).

To compare the trees produced by the different alignment-free methods to the respective benchmark trees, we used the 'Robinson-Foulds (RF)' metric (58), a standard measure to compare how different two tree topologies are. The smaller the 'RF' distances between the reconstructed trees and the corresponding reference trees are, the better a method is. To calculate 'Neighbor-joining' trees and to calculate 'RF' distances between the obtained trees and the respective reference trees, we used the 'PHYLIP' package (59).

As explained above, both 'FSWM' and 'Multi-SpaM' rely on binary patterns of 'match' and 'don't-care' positions; the results of these programs therefore depend on the underlying patterns. Both programs use the software 'rasbhari' (47) to calculate binary patterns. 'rasbhari' uses a probabilistic algorithm, so different program runs usually return different patterns and, as a result, different program runs with 'FSWM' and 'Multi-SpaM' may produce slightly different distance estimates, even if the same parameter values are used. To see how 'FSWM' and 'Multi-SpaM' depend on the underlying binary patterns, we ran both programs ten times on each data set. The figures in the 'Results' section report the 'averageRF'-distance for each data set over the ten program runs. Error bars indicate the highest and lowest RF-distances, respectively, for the 10 program runs.

Simulated sequences

At first, we evaluated 'Multi-SpaM' on data sets generated with the 'Artificial Life Framework (ALF)' (60). 'ALF' starts by simulating an ancestral genome that includes a number of genes. According to a guide tree that is either provided by the user or randomly generated, ALF simulates speciation events and other evolutionary events such as substitutions, insertions and deletions for nucleotides, as well as duplications, deletions and horizontal transfer of entire genes. A large number of parameters can be specified by the user for these events. We used parameter files that were used in a study by the authors of ALF (61). This way, we generated six data sets, three with simulated γ-proteobacterial genomes ($b1$, $b2$, $b3$), and three with simulated mammalian genomes ($m1$, $m2$, $m3$).

We used the base parameter sets for each data set and only slightly modified them to generate *DNA* sequences for roughly 1000 genes per taxon that we then concatenated to full genomes. As in (61), we used parameter values 7.2057 and 401.4189 for the length distribution of the simulated bacterial sequences and 1.7781 and 274.1061, respectively, for the length distribution of the simulated mammalian sequences. Within each data set, we used the same rate for gene duplication, gene loss and horizontal gene transfer, but we used different rates for different data sets. For the six data sets, the corresponding rates were set to 0.0025 ($b1$), 0.0018 ($b2$), 0.0017 ($b3$), 0.0058 ($m1$), 0.0068 ($m2$), 0.011 ($m3$), respectively. Each data set uses a different guide tree that was sampled from known topologies. The average pairwise distances in our simulated sequence sets are as follows (average number of substitutions per position as estimated with *FSWM*): *m1: 0.11; m2: 0.12; m3: 0.07; b1: 0.30; b2: 0.23; b3: 0.25.*

Each data generated in this way set contains 30 genomes (taxa) and has a size of around 10 *mb*. As shown in Figure 3, none of the tools that we evaluated was able to exactly reconstruct the reference tree topologies for the simulated bacterial genomes. In some cases, the average normalized RF distance to the reference trees was 1.0, the maximum possible dissimilarity value. Therefore, we also calculated the triplet distance between the reconstructed trees and the

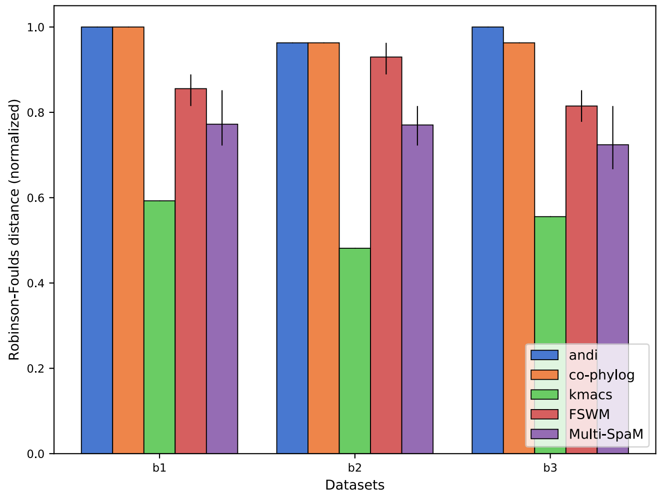

Figure 3. Average 'normalized Robinson-Foulds (RF)' distances between trees calculated with alignment-free methods and reference trees for three sets of simulated bacterial genomes. 'FSWM' and 'Multi-SpaM' were run 10 times, with different patterns *P* generated (see the main text). Error bars indicate the lowest and highest RF distances, respectively.

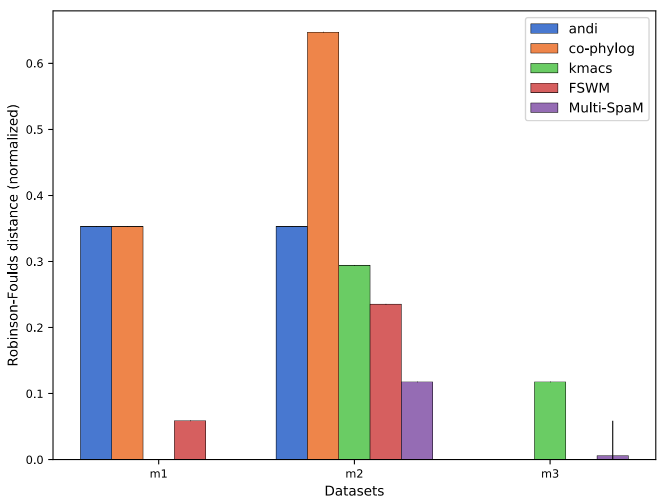

Figure 4. 'Normalized RF' distances for three sets of simulated mammalian genomes. If no bar is shown, the RF distance is zero for the respective method and data set. For example, the RF distance between the tree generated by 'kmacs' for data set *m1* and the reference tree is zero, i.e. here the reference tree topology was precisely reconstructed. Error bars for 'FSWM' and 'Multi-SpaM' are as in Figure 3.

reference trees by running the program 'tqDist' (62). The results are shown in the Supplementary Data. Reference topologies for the simulated mammalian genomes could be reconstructed by some tools, although no method could reconstruct all three reference topologies exactly, see Figure 4.

We also evaluated how the parameters of the genome sequence simulator 'ALF' affect the performance of 'Multi-SpaM' on the simulated genomes. A figure showing the influence of these parameters is given in the Supplementary Data. In short, the rate of 'horizontal gene transfer (HGT)' has a larger influence on the quality of the resulting trees than other parameters of 'ALF'. This is not surprising, since 'HGT' events can lead to false quartet tree topologies, whereas the other program parameters mostly affect the 'number' of *P*-blocks that can be used by 'Multi-SpaM', but not so much the resulting quartet topologies. Even so, the 'HGT' rate in 'ALF' had only minor influence on the quality of the resulting trees, compared to the guide tree that is used in the simulation.

Real genomes

We also applied the programs that we evaluated to real genomes to see if the results are similar to our results on simulated genomes. Here, our first data set were 29 *Escherichia coli* and *Shigella* genomes that are commonly used as a benchmark data set to evaluate alignment-free methods (24). As a reference, we used a tree calculated with 'Maximum Likelihood', based on a 'mugsy' alignment (63). The data set is 144 *mb* large and the average distance between two sequences in this set is ∼0.0166 substitutions per sequence position.

Next, we used 19 *Wolbachia* genomes that have been analyzed by (64); we used the phylogeny published in their paper as a reference. The total size of this sequence set is 25 *mb*, the average pairwise distance is 0.06 substitutions per position. The results of these three series of test runs are summarized in Figure 5.

As a third real-world test case, we used a much larger sequence set, namely a set of 14 plant genomes with a total length of 4.8 *gb*. This data set was originally used by Hatje and Kollmar (65) and has been subsequently used as benchmark data in other publications on alignment-free methods. Figure 6 shows the resulting trees. For this data set, we used a pattern with a weight of $w = 12$ instead of the default value $w = 10$, to keep the number of background spaced-word matches manageable. As can be seen in Figure 6, 'Multi-SpaM' and 'FSWM' produced fairly accurate trees for this data set, with only minor differences to the reference tree: 'Multi-SpaM' misclassified 'Carica papaya', whereas 'FSWM' failed to classify *Brassica rapa* correctly. None of the other alignment-free tools that we evaluated could produce a reasonable tree for this data set: 'andi' returned a tree that is rather different to the reference tree, while 'kmacs' and 'co-phylog' could not finish the program runs in a reasonable time frame.

In addition, we used three real-world data sets that were used as benchmark data in the 'AFproject' paper (23): another data set of 27 *E. coli*/*Shigella* genomes, a set of mitochondrial genomes from 25 fish species, and a set of 8 strains of *Yersinia*.

As explained in the 'Materials and Methods' section, 'Multi-SpaM' calculates an optimal tree topology for each of the sampled 'quartet *P*-blocks'. Here, it can happen that no single best topology is found. In particular for closely related sequences, this happens for a large fraction of the sampled quartet *P*-blocks. For the *E. coli*/*Shigella* data set, for example, ∼50% of the *P*-blocks were inconclusive, i.e. 'RAxML' could find no single best tree topology. We observed a similar result for a data set of 13 *Brucella* genomes where the pairwise phylogenetic distances are even smaller than for the *E. coli*/*Shigella* data set, namely 0.0019 substitutions per site, on average. Here, roughly 80% of the blocks were inconclusive. For all other data sets, the fraction of in-

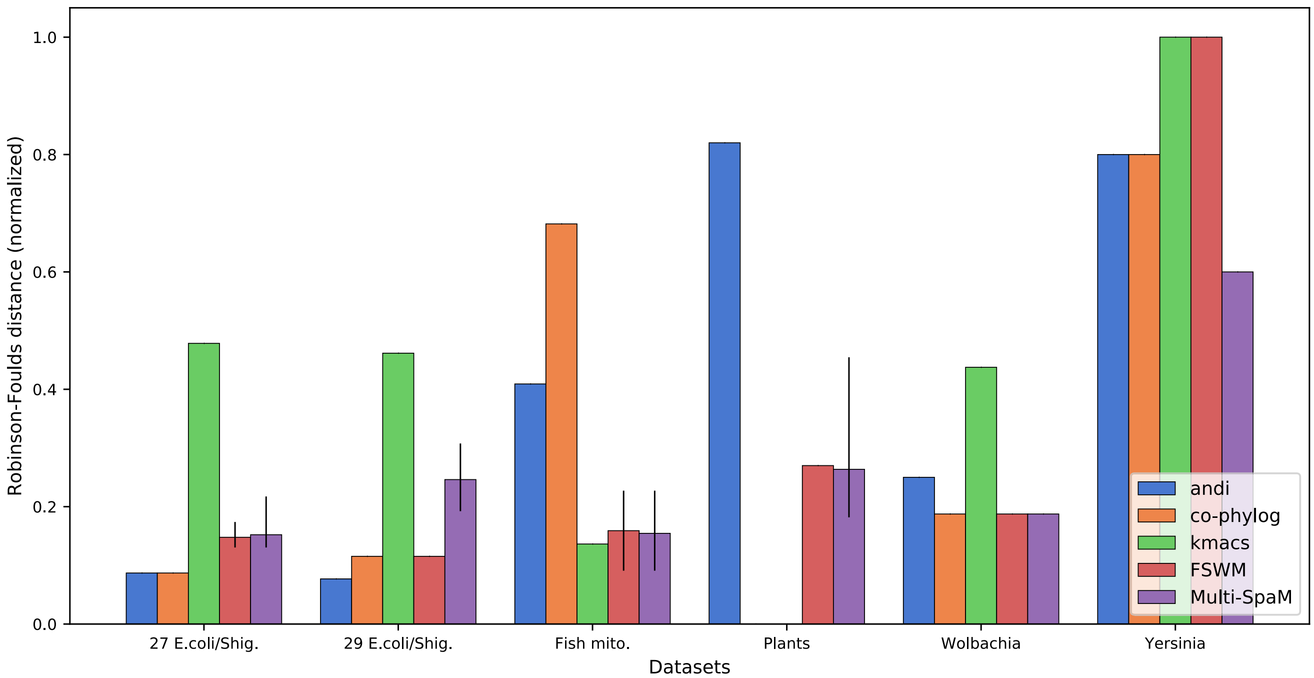

Figure 5. 'Normalized RF' distances for six sets of benchmark genomes: 29 *E. coli/Shigella* genomes, another set of 27 *E. coli/Shigella* genomes, mitochondrial genomes from 25 different fish species, 14 plant genomes, 19 *Wolbachia* genomes and 8 Yersinia genomes. Error bars for 'FSWM 'and 'Multi-SpaM' as in Figure 3. Unlike in Figure 4, missing bars for the plant data sets in this figure mean that the programs in question, *co-phylog* and *kmacs*, did not terminate on this data set.

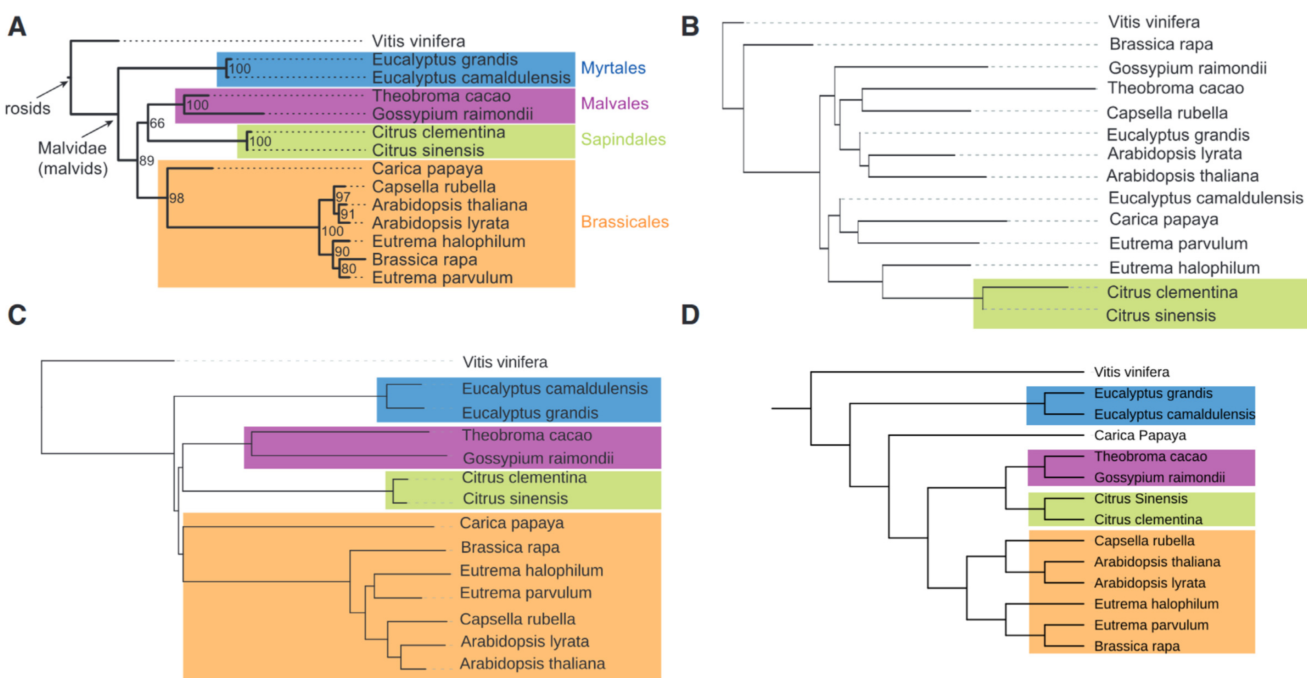

Figure 6. Reference tree (**A**) from (65) and trees reconstructed by 'andi' (**B**), 'FSWM' (**C**) and 'Multi-SpaM' (**D**) for a set of 14 plant genomes.

conclusive quartet *P*-blocks was negligible. For example, for the set of 14 plant genomes, only ∼250 out of the 1 000 000 sampled *P*-blocks were inconclusive.

Program run time and memory usage

Table 1 shows the program run time for 'Multi-SpaM', 'FSWM', 'kmacs', 'andi' and 'co-phylog' on all six real-world data sets in our program comparison. The test runs were done on a 5 x Intel(R) Xeon(R) CPU E7-4850 with 2.00 GHz, a total of 40 threads (20 cores). Some of the programs that we evaluated have been parallelized. For these programs, both 'wall clock time' and 'CPU' time are reported. For the largest data set in our study, the set of 14 plant genomes, the peak 'RAM' usage was 76 GB for 'FSWM', 110 GB for 'andi' and 142 GB for 'Multi-SpaM'.

In memory saving mode, the peak 'RAM' usage of 'Multi-SpaM' could be reduced to 10.5 *GB*, but this roughly doubles the program run time. To achieve this, the list of spaced words is not kept in memory in its entirety, but rather in 16 chunks based on the first two match positions. At any given time, there is only one chunk kept in main memory in addition to the sequences itself and the list of *P*-blocks. The overhead, such as additional comparisons, results in increased run times.

DISCUSSION

Standard software tools for phylogeny reconstruction are relatively slow, because they rely on multiple sequence alignments and on time-consuming probabilistic calculations. Therefore, a variety of so-called 'alignment-free' methods have been proposed recently, which are orders of magnitudes faster than those alignment-based approaches. Existing alignment-free methods calculate 'distances' between DNA or protein sequences that can be used as a basis for phylogeny reconstruction. In general, however, distance-based phylogeny methods are considered to be less accurate than 'character-based' methods. In this paper, we introduced a novel approach to phylogeny reconstruction called 'Multi-SpaM' that combines the speed of alignment-free methods with the accuracy of 'Maximum Likelihood'. To our knowledge, this is the first alignment-free approach that uses multiple sequence comparison and likelihood.

Our test runs show that 'Multi-SpaM' can produce phylogenetic trees of high quality. It outperforms other alignment-free methods on a number of test data sets, in particular on sequences with large evolutionary distances. On sets of very similar sequences, such as different strains of the same bacterial species, however, our approach was sometimes outperformed by other alignment-free methods. As shown in Figure 5, the programs 'andi', 'co-phylog' and 'FSWM' produce better results than 'Multi-SpaM' on a set of *E. coli*/*Shigella* genomes. This may be due to our above-mentioned observation that there is often no single best tree topology for a 'quartet *P*-block', if the compared sequences are very similar to each other.

As mentioned in the 'Results' section, we used 'Neighbor-Joining (NJ)' (7) in order to obtain phylogenetic trees from the distance matrices produced by the competing alignment-free programs 'andi', 'co-phylog', 'FSWM' and 'kmacs'. As an alternative, we also ran the program 'BIONJ' (8). It should be mentioned that, in the majority of test runs, '*BIONJ*" produced slightly better results than 'NJ', especially on the distance matrices produced by 'FSWM'. In our program evaluation, we used 'NJ' anyway, since this program is used in most other studies to evaluate alignment-free methods, e.g. in the recently published 'AFproject' benchmark study (23), so using 'NJ' makes it easier to compare our results to other studies.

Calculating optimal tree topologies for the sampled 'quartet *P*-blocks' is a relatively time-consuming step in 'Multi-SpaM'. In fact we observed that, for a given set of input sequences, the program run time of 'Multi-SpaM' is roughly proportional to the number of *P*-blocks for which topologies are calculated. However, the maximal number of 'quartet blocks' that are sampled is a user-defined parameter. By default we sample up to $M = 1\ 000\ 000$ quartet blocks; in our test runs, the quality of the resulting trees could not be significantly improved by further increasing M (test results with different values of M are shown in the Supplementary Data). Consequently, our method is relatively fast on large data sets, where only a small fraction of the possible quartet-blocks is sampled. By contrast, on small data sets, 'Multi-SpaM' is slower than other alignment-free methods. To further speed-up 'Multi-SpaM', we have parallelized our software to run on multiple cores; in Table 1, we report both wall-clock and 'CPU' run times. It should be straight-forward to adapt our software to run on distributed systems, as has been done for other alignment-free approaches (66,67).

Apart from the maximum number of sampled quartet blocks, the only relevant parameters of our approach are the 'length' and the 'weight' (number of 'match positions') of the underlying binary pattern. For 'Multi-SpaM', we used similar default values as in 'Filtered Spaced Word Matches (FSWM)' (41), namely a weight of $w = 10$ and a pattern length of $\ell = 110$, so our default patterns have 100 'don't-care' positions. As mentioned in the 'Materials and Methods' section, a large number of 'don't-care' positions is important in 'Multi-SpaM' as well as in our previous approach 'FSWM', since this makes it easier to distinguish homologous from random background spaced-word matches. Also, a large number of 'don't-care' positions helps to reduce the number of 'inconclusive' quartet *P*-blocks, where no single best quartet tree exists, on data sets where sequences are closely related to each other.

Our default pattern length $\ell = 110$ limits, on the other hand, the number of homologous quartet blocks that can be found. Since 'Multi-SpaM' is based on 'gap-free' four-way alignments of length ℓ, *P*-blocks with positive scores can only be expected in sequence regions without insertions or deletions. For real-world data sets, it is difficult to tell how exactly the number of possible homologous *P*-blocks depends on the pattern length—to find out, one would need either a reliable multiple alignment of the sequences or the full list of homologous *P*-blocks. But both are impossible to calculate for large genome sequences. As a proxy, to get an idea how the number of homologous 'quartet' *P*-blocks is affected by the pattern length ℓ, we counted the number of 'pairwise' spaced-word matches with positive scores for different patterns with a fixed weight and variable length.

Table 1. Run time in seconds for different alignment-free approaches on six sets of real-world genomes. On the largest data set, the 14 plant genomes, 'kmacs' and 'co-phylog' did not terminate the program run. On this data set, we increased the pattern weight for 'Multi-SpaM' from the default value of $w = 10$ to $w = 12$, in order to reduce the run time. Note that 'Multi-SpaM', 'FSWM' and 'andi' are parallelized, so we could run them on multiple processors, while 'kmacs' and 'co-phylog' had to be run on single processors. The reported run times are 'wall clock times'.

	FSWM		andi		co-phylog	kmacs	Multi-SpaM	
	wall clock	CPU	wall clock	CPU			wall clock	CPU
27 *E.coli*/*Shigella*	710	27,075	15	291	704	5,247	603	22,185
29 *E.coli*/*Shigella*	860	32,798	16	325	533	55,736	611	21,973
25 fish mitochondria	2	8	<1	3	9	5	27	1,054
14 plants	1,107,720	28,690,489	1,808	13813	-	-	12,516	389,770
19 *Wolbachia*	65	2,185	3	42	113	24,961	484	15,804
8 *Yersinia*	91	3,333	5	34	50	1,083	183	6,182

For various real-world genomes, we found that, with our default length $\ell = 110$, the number of spaced-word matches with positive scores is only slightly smaller than with a pattern length of, for example, $\ell = 60$. Details are shown in the Supplementary Data.

In 'Multi-SpaM', we are using by default a relatively low 'weight' of the underlying pattern P, to obtain a sufficiently large number of P-blocks. On very large data sets, on the other hand, it is advisable to increase the weight of P, in order to reduce the number of the spaced-word matches that are considered, and thereby the program run time. For the largest data set our study, the set of plant genomes, we increased the pattern weight in our test runs from the default value $w = 10$ to $w = 12$. A table in the Supplementary Data shows that increasing the pattern weight can slightly deteriorate the quality of the resulting trees, so one should be careful with this option.

We should mention that it is, in general, not possible to predict the run time of 'Multi-SpaM' from the program parameters and the size of the input data alone. A relatively time-consuming step of our algorithm is sampling homologous P-blocks. As detailed above, this is done by iteratively picking a random spaced-word occurrence, and by looking at other random occurrences of the same spaced word in different sequences, until three spaced-word matches with positive scores are found, i.e. until a homologous P-block is found. Since we are using patterns with a low weight, most random spaced-word matches have negative scores. The number of spaced-word matches that have to be evaluated, until a homologous P-block is found, depends on the input sequences and can vary considerably. This may be the reason why the relative run time of 'Multi-SpaM', compared to other methods, is rather variable, as can be seen in Table 1. The instability of the program run time is a certain disadvantage of our approach.

To distinguish between homologous and background spaced-word matches, we are using a nucleotide substitution matrix that has been published by Webb Miller's group (46), the same matrix that we are using in 'Filtered Spaced Word Matches' (41). As we have shown in this previous paper, homologous and background spaced-word matches can be easily distinguished if the number of 'don't-care positions' is sufficiently large. The performance of our program is, thus, hardly affected by the specific substitution matrix that we are using; on most sequence sets one can expect to obtain similar results with an alternative matrix, or even by simply counting matches and mismatches.

To calculate supertrees from quartet tree topologies, the current implementation of 'Multi-SpaM' uses the previously developed software 'Quartet MaxCut' (44,55). We are using this program since it is faster and produced better results on our data than other supertree approaches. A drawback of this approach is that the current version of 'Multi-SpaM' generates tree 'topologies' only, i.e. trees without branch lengths. We will investigate in the future, if our approach can be extended to calculate full phylogenetic trees with branch lengths, based on the same 'quartet' P-blocks that we have used in the present study.

DATA AVAILABILITY

The source code of the program is available at https://github.com/tdencker/multi-SpaM.

Contact: thomas.dencker@stud.uni-goettingen.de

SUPPLEMENTARY DATA

Supplementary Data are available at NARGAB Online.

FUNDING

W Foundation [VWZN3157]. We acknowledge support by the Open Access Publication Funds of Göttingen University.

Conflict of interest statement. None declared.

REFERENCES

1. Farris,J.S. (1970) Methods for computing wagner trees. *System. Biol.*, **19**, 83–92.
2. Fitch,W. (1971) Toward defining the course of evolution: minimum change for a specific tree topology. *System. Zool.*, **20**, 406–416.
3. Felsenstein,J. (1981) Evolutionary trees from DNA sequences:a maximum likelihood approach. *J. Mol. Evol.*, **17**, 368–376.
4. Sievers,F., Wilm,A., Dineen,D., Gibson,T.J., Karplus,K., Li,W., Lopez,R., McWilliam,H., Remmert,M., Söding,J. *et al.* (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol. Syst. Biol.*, **7**, 539.
5. Chor,B. and Tuller,T. (2005) Maximum Likelihood of Evolutionary Trees Is Hard. In: Miyano,S, Mesirov,J, Kasif,S, Istrail,S, Pevzner,PA and Waterman,M. (eds). *Research in Computational Molecular Biology*. Springer, Berlin, Heidelberg, pp. 296–310.
6. Foulds,L. and Graham,R. (1982) The steiner problem in phylogeny is NP-complete. *Adv. Appl. Math.*, **3**, 43–49.
7. Saitou,N. and Nei,M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, **4**, 406–425.

8. Gascuel,O. (1997) BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Mol. Biol. Evol.*, **14**, 685–695.
9. Haubold,B. (2014) Alignment-free phylogenetics and population genetics. *Brief. Bioinform.*, **15**, 407–418.
10. Song,K., Ren,J., Reinert,G., Deng,M., Waterman,M.S. and Sun,F. (2014) New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. *Brief. Bioinform.*, **15**, 343–353.
11. Zielezinski,A., Vinga,S., Almeida,J. and Karlowski,W.M. (2017) Alignment-free sequence comparison: benefits, applications, and tools. *Genome Biol.*, **18**, 186.
12. Bernard,G., Chan,C.X. and Ragan,M.A. (2016) Alignment-free microbial phylogenomics under scenarios of sequence divergence, genome rearrangement and lateral genetic transfer. *Sci. Rep.*, **6**, 28970.
13. Ren,J., Bai,X., Lu,Y.Y., Tang,K., Wang,Y., Reinert,G. and Sun,F. (2018) Alignment-Free Sequence Analysis and Applications. *Ann. Revi. Biomed. Data Sci.*, **1**, 93–114.
14. Bernard,G., Chan,C.X., Chan,Y.-B., Chua,X.-Y., Cong,Y., Hogan,J.M., Maetschke,S.R. and Ragan,M.A. (2019) Alignment-free inference of hierarchical and reticulate phylogenomic relationships. *Brief. Bioinform.*, **22**, 426–435.
15. Kucherov,G. (2019) Evolution of biosequence search algorithms: a brief survey. *Bioinformatics*, **35**, 3547–3552.
16. Roychowdhury,T., Vishnoi,A. and Bhattacharya,A. (2013) Next-Generation Anchor Based Phylogeny (NexABP): Constructing phylogeny from Next-generation sequencing data. *Sci. Rep.*, **3**, 2634.
17. Song,K., Ren,J., Zhai,Z., Liu,X., Deng,M. and Sun,F. (2013) Alignment-Free Sequence Comparison Based on Next-Generation Sequencing Reads. *J. Comput. Biol.*, **20**, 64–79.
18. Yi,H. and Jin,L. (2013) Co-phylog: an assembly-free phylogenomic approach for closely related organisms. *Nucleic Acids Res.*, **41**, e75.
19. Comin,M. and Schimd,M. (2014) Assembly-free genome comparison based on next-generation sequencing reads and variable length patterns. *BMC Bioinform.*, **15**, S1.
20. Ondov,B.D., Treangen,T.J., Melsted,P., Mallonee,A.B., Bergman,N.H., Koren,S. and Phillippy,A.M. (2016) Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biol.*, **17**, 132.
21. Lau,A.K., Leimeister,C.-A. and Morgenstern,B. (2019) Read-SpaM: assembly-free and alignment-free comparison of bacterial genomes. bioRxiv doi: http://dx.doi.org/10.1101/550632, 12 October 2019, preprint: not peer reviewed.
22. Sarmashghi,S., Bohmann,K., Gilbert,M.T.P., Bafna,V. and Mirarab,S. (2019) Skmer: assembly-free and alignment-free sample identification using genome skims. *Genome Biol.*, **20**, 34.
23. Zielezinski,A., Girgis,H.Z., Bernard,G., Leimeister,C.-A., Tang,K., Dencker,T., Lau,A.K., Röhling,S., Choi,J., Waterman,M.S. *et al.* (2019) Benchmarking of alignment-free sequence comparison methods. *Genome Biol.*, **20**, 144.
24. Haubold,B., Klötzl,F. and Pfaffelhuber,P. (2015) andi: Fast and accurate estimation of evolutionary distances between closely related genomes. *Bioinformatics*, **31**, 1169–1175.
25. Klötzl,F. and Haubold,B. (2018) Fast and Accurate Distance Computation from Unaligned Genomes. In: *Proceedings German Conference on Bioinformatics GCB'18, Poster Abstracts, September 25-28, 2018*. Vienna.
26. Ulitsky,I., Burstein,D., Tuller,T. and Chor,B. (2006) The average common substring approach to phylogenomic reconstruction. *J. Comput. Biol.*, **13**, 336–350.
27. Haubold,B., Pfaffelhuber,P., Domazet-Loso,M. and Wiehe,T. (2009) Estimating Mutation Distances from Unaligned Genomes. *J. Comput. Biol.*, **16**, 1487–1500.
28. Leimeister,C.-A. and Morgenstern,B. (2014) *kmacs*: the k-mismatch average common substring approach to alignment-free sequence comparison. *Bioinformatics*, **30**, 2000–2008.
29. Thankachan,S.V., Apostolico,A. and Aluru,S. (2016) A Provably Efficient Algorithm for the k-Mismatch Average Common Substring Problem. *J. Comput. Biol.*, **23**, 472–482.
30. Thankachan,S.V., Chockalingam,S.P., Liu,Y. and Aluru,A.K.S. (2017) A greedy alignment-free distance estimator for phylogenetic inference. *BMC Bioinformatics*, **18**, 238.
31. Morgenstern,B., Schöbel,S. and Leimeister,C.-A. (2017) Phylogeny reconstruction based on the length distribution of k-mismatch common substrings. *Algorithms Mol. Biol.*, **12**, 27.
32. Ayad,L.A., Charalampopoulos,P., Iliopoulos,C.S. and Pissis,S.P. (2018) Longest Common Prefixes with k-Errors and Applications. arXiv doi: https://arxiv.org/abs/1801.04425, 13 January 2018, preprint: not peer reviewed.
33. Bromberg,R., Grishin,N.V. and Otwinowski,Z. (2016) Phylogeny Reconstruction with Alignment-Free Method That Corrects for Horizontal Gene Transfer. *PLoS Comput. Biol.*, **12**, e1004985.
34. Röhling,S., Dencker,T. and Morgenstern,B. (2019) The number of k-mer matches between two DNA sequences as a function of k. bioRxiv doi: http://dx.doi.org/10.1101/527515, 30 April 2019, preprint: not peer reviewed.
35. Leimeister,C.-A., Boden,M., Horwege,S., Lindner,S. and Morgenstern,B. (2014) Fast Alignment-Free sequence comparison using spaced-word frequencies. *Bioinformatics*, **30**, 1991–1999.
36. Horwege,S., Lindner,S., Boden,M., Hatje,K., Kollmar,M., Leimeister,C.-A. and Morgenstern,B. (2014) *Spaced words* and *kmacs*: fast alignment-free sequence comparison based on inexact word matches. *Nucleic Acids Res.*, **42**, W7–W11.
37. Morgenstern,B., Zhu,B., Horwege,S. and Leimeister,C.-A. (2015) Estimating evolutionary distances between genomic sequences from spaced-word matches. *Algorithms Mol. Biol.*, **10**, 5.
38. Ounit,R. and Lonardi,S. (2015) Algorithms in Bioinformatics: 15th International Workshop. *WABI 2015, Atlanta, GA, USA, September 10-12, 2015, Proceedings chapter Higher Classification Accuracy of Short Metagenomic Reads by Discriminative Spaced* k-*mers*. Springer, Berlin, Heidelberg, pp. 286–295.
39. Noé,L. (2017) Best hits of 11110110111: model-free selection and parameter-free sensitivity calculation of spaced seeds. *Algorithms Mol. Biol.*, **12**, 1.
40. Girotto,S., Comin,M. and Pizzi,C. (2018) Efficient computation of spaced seed hashing with block indexing. *BMC Bioinformatics*, **19**, 441.
41. Leimeister,C.-A., Sohrabi-Jahromi,S. and Morgenstern,B. (2017) Fast and Accurate Phylogeny Reconstruction using Filtered Spaced-Word Matches. *Bioinformatics*, **33**, 971–979.
42. Leimeister,C.-A., Schellhorn,J., Dörrer,S., Gerth,M., Bleidorn,C. and Morgenstern,B. (2019) Prot-SpaM: Fast alignment-free phylogeny reconstruction based on whole-proteome sequences. *GigaScience*, **8**, doi:10.1093/gigascience/giy148.
43. Stamatakis,A. (2014) RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, **30**, 1312–1313.
44. Snir,S. and Rao,S. (2012) Quartet MaxCut: A fast algorithm for amalgamating quartet trees. *Mol. Phylogenet. Evol.*, **62**, 1–8.
45. Dencker,T., Leimeister,C.-A., Gerth,M., Bleidorn,C., Snir,S. and Morgenstern,B. (2018) *Multi-SpaM*: a Maximum-Likelihood approach to phylogeny reconstruction using multiple spaced-word matches and quartet trees. In: Blanchette,M and Ouangraoua,A (eds). *Comparative Genomics*. Springer International Publishing, Cham, pp. 227–241.
46. Chiaromonte,F., Yap,V.B. and Miller,W. (2002) Scoring Pairwise Genomic Sequence Alignments. In: Altman,R.B., Dunker,A.K., Hunter,L. and Klein,T.E., (eds). *Pacific Symposium on Biocomputing*. Lihue, Hawaii, pp. 115–126.
47. Hahn,L., Leimeister,C.-A., Ounit,R., Lonardi,S. and Morgenstern,B. (2016) *rasbhari*: optimizing spaced seeds for database searching, read mapping and alignment-free sequence comparison. *PLOS Comput. Biol.*, **12**, e1005107.
48. Ilie,L., Ilie,S. and Bigvand,A.M. (2011) SpEED: fast computation of sensitive spaced seeds. *Bioinformatics*, **27**, 2433–2434.
49. Ilie,S. (2012) Efficient Computation of Spaced Seeds. *BMC Res. Notes*, **5**, 123.
50. Tavaré,S. (1986) Some probabilistic and statistical problems on the analysis of DNA sequences. *Lect. Math. Life Sci.*, **17**, 57–86.
51. Bininda-Emonds,O. (2004) Phylogenetic supertrees: Combining information to reveal the Tree of Life. *Computational Biology*. Springer, Heidelberg, NY.
52. Steel,M. (1992) The Complexity of Reconstructing Trees from Qualitative Characters and Subtress. *J. Classifi.*, **9**, 91–116.
53. Baum,B. (1992) Combining trees as a way of combining data sets for phylogenetic inference. *Taxon*, **41**, 3–10.

54. Ragan,M. (1992) Matrix representation in reconstructing phylogenetic-relationships among the eukaryotes. *Biosystems*, **28**, 47–55.
55. Snir,S. and Rao,S. (2010) Quartets MaxCut: A Divide and Conquer Quartets Algorithm. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **7**, 704–718.
56. Avni,E., Yona,Z., Cohen,R. and Snir,S. (2018) The Performance of Two Supertree Schemes Compared Using Synthetic and Real Data Quartet Input. *J. Mol. Evol.*, **86**, 150–165.
57. OpenMP Forum, OpenMP C and C++ Application Program Interface, Version 2.0. (2002) http://www.openmp.org. Technical report, (March, 2002).
58. Robinson,D.F. and Foulds,L. (1981) Comparison of phylogenetic trees. *Math. Biosci.*, **53**, 131–147.
59. Felsenstein,J. (1989) PHYLIP - Phylogeny Inference Package (Version 3.2). *Cladistics*, **5**, 164–166.
60. Dalquen,D.A., Anisimova,M., Gonnet,G.H. and Dessimoz,C. (2012) ALF - A Simulation Framework for Genome Evolution. *Mol. Biol. Evol.*, **29**, 1115–1123.
61. Dalquen,D.A., Altenhoff,A.M., Gonnet,G.H. and Dessimoz,C. (2013) The Impact of Gene Duplication, Insertion, Deletion, Lateral Gene Transfer and Sequencing Error on Orthology Inference: A Simulation Study. *PLOS ONE*, **8**, 1–11.
62. Sand,A., Holt,M.K., Johansen,J., Brodal,G.S., Mailund,T. and Pedersen,C. N.S. (2014) tqDist: a library for computing the quartet and triplet distances between binary or general trees. *Bioinformatics*, **30**, 2079–2080.
63. Angiuoli,S.V. and Salzberg,S.L. (2011) Mugsy: fast multiple alignment of closely related whole genomes. *Bioinformatics*, **27**, 334–342.
64. Gerth,M. and Bleidorn,C. (2016) Comparative genomics provides a timeframe for *Wolbachia* evolution and exposes a recent biotin synthesis operon transfer. *Nat. Microbiol.*, **2**, 16241.
65. Hatje,K. and Kollmar,M. (2012) A phylogenetic analysis of the brassicales clade based on an alignment-free sequence comparison method. *Front. Plant Sci.*, **3**, 192.
66. Cattaneo,G., Ferraro Petrillo,U., Giancarlo,R. and Roscigno,G. (2017) An Effective Extension of the Applicability of Alignment-free Biological Sequence Comparison Algorithms with Hadoop. *J. Supercomput.*, **73**, 1467–1483.
67. Petrillo,U.F., Guerra,C. and Pizzi,C. (2017) A new distributed alignment-free approach to compare whole proteomes. *Theor. Computer Sci.*, **698**, 100–112.

3 Insertions and deletions as a phylogenetic signal in an alignment-free context

We prepared the following manuscript for submission. It also contains several results that will not be part of the published manuscript.

Reference

Thomas Dencker, Niklas Birth and Burkhard Morgenstern. Insertions and deletions as a phylogenetic signal in an alignment-free context. Prepared for submission, 2020.

Original Contribution

TD designed the study and performed the program evaluation. NB implemented the software. TD assisted in the developement of the software. BM supervised the project. TD wrote the manuscript. BM improved parts of the manuscript.

Insertions and deletions as a phylogenetic signal in an alignment-free context

Thomas Dencker[1], Niklas Birth[1] and Burkhard Morgenstern[1,2]

[1]Department of Bioinformatics, Institute of Microbiology and Genetics, University of Göttingen, Goldschmidtstr. 1, 37077 Göttingen, Germany
[2]Göttingen Center of Molecular Biosciences (GZMB), Justus-von-Liebig-Weg 11, 37077 Göttingen, Germany

January 27, 2020

Abstract

Most methods for phylogeny inference are based on sequence alignments; they infer the phylogeny of a set of input taxa based on aligned nucleotide or amino-acid residues. Gaps in alignments are usually not used as phylogenetic signal even though they can, in principle, provide valuable information. In this paper, we explore how information about insertions and deletions can be utilized for phylogenetic tree inference. Our approach does not need a full sequence alignment of the compared sequences. Instead, we are using our previously developed approach *Multi-SpaM*, to generate local gap-free four-way alignments, so-called *blocks*. For adjacent blocks involving the same four sequences, we consider the distances between these blocks in all four sequences, to obtain information about insertions or deletions that have happened since the four sequences evolved from their last common ancestor. This way, a pair of adjacent blocks can support one of the three possible quartet topologies for the four involved sequences. We are using this information as input for *Maximum-Parsimony* and for the software program *Quartet MaxCut* to reconstruct phylogenetic trees based on insertions and deletions.

1 Introduction

The foundation of most phylogenetic studies are multiple sequence alignments (MSAs), either of partial or complete genomes or of individual genes or proteins. If MSAs of multiple genes or proteins are used, there are two possibilities to construct a phylogenetic tree: (1) the alignments can be concatenated to form a so-called *superalignment* or *supermatrix*. Tree building methods such as *Maximum-Likelihood* [45, 14], *Bayesian Approaches* [39] or *Maximum-Parsimony* [10, 12, 47] can then be applied to these superalignments. (2) One can calculate a separate tree for each gene or protein family and then use a *supertree approach* [4] to amalgamate these different trees into one final tree, with methods such as *ASTRAL* [53], *MRP* [36] or *Quartet MaxCut* [44].

Multiple sequence alignments usually contain gaps representing insertions or deletions (*indels*) that are assumed to have happened since the aligned sequences evolved from their last common ancestor. Gaps, however, are usually not used for phylogeny reconstruction. Most of the above tree-reconstruction methods are based on substitution models for nucleotide or amino-acid residues. Here, alignment columns with gaps are either completely ignored, or gaps are treated as 'missing information', for example in the frequently used tool *PAUP** [47]. Some models have been proposed that can include gaps in the *Maximum-Likelihood* analysis such as *TKF91* [48] and TKF92 [49], see also [18, 1, 29]. Unfortunately, these models do not scale well to genomic data. Thus, indels are rarely used as a source of information for the phylogenetic analysis.

In those studies that actually make use of indels, this additional information is usually encoded in some simple manner. The most straightforward encoding is to treat the gap character as a fifth character for DNA comparison, or as a 21st character in protein comparison, respectively. Clearly, the lengths of gaps are not explicitly considered in such an approach. That is, for a gap of length ℓ in an MSA, the ℓ gap characters are treated as independent insertion or deletion events. Some more downsides of this approach are discussed in [41]; these authors introduced the "simple encoding" of indel data as an alternative. For every indel in the multiple sequence alignment, an additional column is appended. This column contains a present/absent encoding for an indel event which is defined as a gap with the same start and end positions. If a longer gap is fully contained in a shorter gap in another sequence, it is considered as *missing information*. Such a simple binary encoding is an effective way of using the length of the indels to gain additional information and can be used in some *maximum-parsimony* framework. A disadvantage of these approaches is their relatively long runtime. The above authors also proposed a more complex encoding of gaps [41] which they further refined in a subsequent paper [33]. The commonly used approaches to encode gaps for phylogeny reconstruction are compared in [34].

The “simple encoding” of gaps has been used in many studies; one recent study obtained additional information on the phylogeny of Neoaves which was hypothesized to have a ”hard polytomy” [20]. Despite such successes, indel information is still largely ignored in phylogeny reconstruction. Oftentimes, it is unclear whether using indels is worth the large overhead and increased runtime. On the other hand, it has also been shown that gaps can contain substantial phylogenetic information [8].

All of the above mentioned approaches to use indel information for phylogeny reconstruction require MSAs of the compared sequences. Nowadays, the amount of the available molecular data is rapidly increasing, due to the progress in next-generation sequencing technologies. If the size of the analyzed sequences increases, calculating multiple sequence alignments quickly becomes too time-consuming. Thus, in order to provide faster and more convenient methods to phylogenetic reconstruction, many alignment-free approaches have been proposed in recent years. Most of these approaches calculate pairwise distances between sequences, based on sequence features such as k-mer frequencies [42, 35, 24] or the number [27] or length [25, 50, 30] of word matches. Distance methods such as *Neighbor-Joining* [40] or *BIONJ* [13] can then reconstruct phylogenetic trees from the calculated distances. For a thorough overview, the reader is referred to recent reviews of alignment-free methods [51, 16, 3].

Some of the more recent alignment-free methods use inexact word matches between pairs of sequences [52, 17, 26], where mismatches are allowed to some degree. Such word matches can be considered as pairwise, gap-free “micro-alignments”. So, strictly spoken, these methods are not ‘’alignment-free”. In the literature, they are still called “alignment-free”, as they circumvent the need to calculate full sequence alignments of the compared sequences. The advantage of such “micro-alignments” is that inexact word matches can be found almost as efficiently as exact word matches, by adapting standard word-matching algorithms.

A number of these methods use so-called *spaced-words* [19, 24, 32]. A spaced-word is a word composed of nucleotide or amino-acid symbols that contains additional *wildcard* characters at certain positions, specified by a pre-defined binary pattern P representing ‘match positions’ and ‘don’t-care positions’. If the same ‘spaced word’ occurs in two different sequences, this is called a *Spaced-word Match* or *SpaM*, for short. One way of using spaced-word matches – or other types of inexact word matches – in alignment-free sequence comparison is to use them as a proxy for full alignments, to estimate the number of mismatches per position in the (unknown) full sequence alignment. This idea has been implemented in the software *Filtered Spaced Word Matches (FSWM)* [26]; it has also been applied to protein sequences [23], and to unassembled reads [22]. Other approaches have been proposed recently, that use the *number* of *SpaMs* to estimate phylogenetic distances between DNA sequences [32, 38], see [31] for a review of the various *SpaM*-based methods.

The *SpaM* approach has also been applied to been applied to *multiple* sequence comparison. *Multi-SpaM* [7] is a recent extension of the *FSMW* idea that finds *multiple* spaced-word matches with respect to some binary pattern P. Such a multiple *SpaM* is called a P-*block*. Each P-*block*, thus, consists four occurrences of the same spaced-word, with respect to the selected pattern P. For each such block, the program then identifies the optimal quartet tree topology, using the program *RAxML* [45]. Finally, a super tree is found based on a large sample of quartet trees with the program *Quartet MaxCut* [44].

In the present paper, we use the *blocks* identified by *Multi-SpaM* to use insertions and deletions as phylogenetic signal. More specifically, for pairs of adjacent blocks that involve the same four sequences, we consider the distances between the two blocks in these four sequences. If this distance is the same for two of the four sequences, but different for the remaining two sequences, this indicates that the two sequences with the same distance should be grouped together, in the sense of *maximum parsimony*. This way, a pair of adjacent blocks may support one of the three possible quartet topologies for the four involved sequences.

We investigate multiple ways of reconstructing phylogenetic trees based on quartet topologies inferred in this way. In these experiments, we are using indel information (A) as the sole source of phylogenetic information, and (B) in conjunction with the quartet topologies that are constructed by *Multi-SpaM*. To construct phylogenetic trees from indel information, we use the *Quartet MaxCut* program that we previously used in *Multi-SpaM*, but we also use *Maximum-Parsimony* methods to find trees solely based on indel information. We show that under certain circumstances, the indel data is highly accurate and can improve the quality of the phylogenetic trees in some cases.

2 Design and Implementation

2.1 Spaced words, P-blocks and distances between P-blocks

We are using standard notation from stringology as defined, for example, in [15]. For a sequence S over some alphabet, $S(i)$ denotes the i-th symbol of S. In order to investigate the information that can be obtained from putative indels in an alignment-free context, we use the *P-blocks* generated by the program *Multi-SpaM* [7]. At the start of every run, a binary pattern $P \in \{0,1\}^\ell$ is specified for some integer ℓ. Here, a "1" in P denotes a *match position*, a "0" stands for a *don't-care position*. The number of *match positions* in P is called its *weight* and is denoted by w. By default, we are using parameter values $\ell = 110$ and $w = 10$, so by default the pattern P has 100 *don't-care* positions.

A spaced-word W with respect to a pattern P is a word over the alphabet $\{A,C,G,T\} \cup \{*\}$ with $W(i) = *$ if and only if i is a *don't care position* of P, i.e. if $P(i) = 0$. If S is a sequence of length N over the nucleotide alphabet $\{A,C,G,T\}$, and W is a spaced word, we say that W *occurs* at some position $i \in \{1,\dots,\ell\}$, if $S(i+j-1) = W(j)$ for every match position j in P. For two sequences S and S' and positions i and i' in S and S', respectively, we say that there is a *spaced-word match (SpaM)* between S and S' at (i,i'), if the same spaced word W occurs at i in S and at i' in S'. A *SpaM* can be considered as a local pairwise alignment without gaps. Given a nucleotide substitution matrix, the *score* of a spaced-word match is defined as the sum of the substitution scores of the nucleotides aligned to each other at the *don't-care* positions of the underlying pattern P. In *FSWM* and *Multi-SpaM*, we are using a substitution matrix described in [5]. In *FSWM*, only *SpaMs* with positive scores are used. It has been shown that this *SpaM-filtering* step can effectively eliminate most random spaced-word matches [26].

The program *Multi-SpaM* is based on so-called *P-blocks*, where a P-block is defined as four occurrences of some spaced word W in four different sequences. For a set of $N \geq 4$ input sequences, a P-block can be thus considered as a local gap-free four-way alignment. To exclude spurious random P-blocks, *Multi-SpaM* removes P-blocks with low scores. More precisely, a P-block is required to contain one occurrence of the spaced-word W, such that the other three occurrences of W have positive scores with this first occurrence. In this paper, we are considering *pairs* of P-blocks involving the same four sequences, and we are using the distances between the two blocks in these sequences as phylogenetic signal.

2.2 Phylogeny inference using distances between P-blocks

Let us consider two P-blocks B_1 and B_2 involving the same four sequences S_s, S_r, S_t, S_u, and let D_s, D_r, D_t, D_u be the distances between B_1 and B_2 in the four sequences. More specifically, we define D_r as the length of the segment in S_r between the two spaced-word occurrences corresponding to B_1 and B_2, see Figures 1 and 2 for examples. If we find that two of these distances, say D_s and D_r, are different from each other, this would imply that an insertion or deletion has happened in S_s and S_r between B_1 and B_2, since the two sequences evolved from their last common ancestor. If D_r and D_s are equal, no such insertion or deletion has to be assumed. It is therefore possible that the distances between B_1 and B_2 support one of three possible binary quartet topologies for the four involved sequences, in the sense of the *parsimony* principle, applied to insertions and deletions.

A binary quartet tree or topology corresponds to a split of the involved taxa, with two taxa on each side of the split: for taxa A, B, C, D, the split $AB|CD$ would correspond to the topology where A and B are at neighboring leaves, as well as C and D, and an internal edge would separate A and B from C and D. There are two situations where the above distances $D_r, \ldots, D_u$ would support one of the three possible binary quartet topologies for the respective sequences. (1) Two distances, say D_r, D_s, are equal to each other, and the other distances, D_t, D_u are also equal to each other, but different from D_r and D_s, as for example in Figure 1. We say that this situation would *strongly* support the split $S_rS_s|S_tS_u$, and we call (B_1, B_2) a *type 1 pair of blocks*. (2) Two distances, D_r and D_s are equal and D_t, D_u would be different from D_r and D_s, but also different from each other, as shown, for example, in Figure 2. Here, we would say that this constellation would *weakly* support the same split $S_rS_s|S_tS_u$; in this case, we call (B_1, B_2) a *type 2 pair of blocks*. By contrast, if the four distances are either equal to each other, or if they have four different values, no quartet topology would be supported. If a pair of P-blocks (B_1, B_2) is either a type 1 or a type 2 pair of blocks, we call (B_1, B_2) an *informative P-block pair*.

We implemented two different ways of inferring phylogenetic trees from a set $\mathcal{B}$ of *informative P-block pairs*. First, we calculated the *quartet topology* for each informative P-block pair in $\mathcal{B}$, and we applied a *super-tree* approach to infer a topology for a set of input sequences from these quartet topologies. Here, we used the program *Quartet MaxCut* [43, 44] which is very fast and accurate. Furthermore, this method can compensate for inaccurate quartet topologies, as long as there are sufficiently many quartet topologies in total [46, 2]. This approach is similar to the algorithm implemented in our

Sequence													Distance
S_r	A	G	G	C	A	A	C	G	G	T			2
S_s	A	G	G	C	A	T	C	G	G	T			2
S_t	A	G	G	C	A	A	C	T	C	G	G	T	4
S_u	A	G	G	C	A	A	C	T	C	G	G	T	4

Figure 1: Distances between two P-blocks involving the same four sequences. In the sense of *maximum parsimony* with respect to insertions and deletions, these distances would *strongly* support the split $S_rS_s|S_tS_u$.

Sequence													Distance
S_r	A	G	G	C	A	A	C	G	G	T			2
S_s	A	G	G	C	A	T	C	G	G	T			2
S_t	A	G	G	C	A	A	C	T	C	G	G	T	4
S_u	A	G	G	C	A	A	T	C	G	G	T		3

Figure 2: Distances between two P-blocks as in Figure 1. Here, the distances would *weakly* support the split $S_rS_s|S_tS_u$.

previous software program *Multi-SpaM* where we inferred quartet topologies from the nucleotides aligned at the *don't-care* positions of P-blocks.

As an alternative, we used the distances in informative P-block pairs from $\mathcal{B}$ as input for *Maximum-Parsimony* [10, 12]. To this end, we generate a character matrix M as follows: the rows of M correspond, as usual, to the input sequences, and each informative P-block pair corresponds to one column of M. The distances between the two P-blocks are encoded by characters 0, 1 and 2, such that equal distances in an informative P-block pair were represented by the same character. For sequences not involved in an informative P-block pair, the entry in M is empty and considered as 'missing information'. In Figure 1, for example, the entries for S_r, S_s, S_t, S_u would be 0, 0, 1, 1; in Figure 2, the entries would be 0, 0, 1, 2. Although encoding P-block distances as a character matrix M and using this matrix as input for parsimony software is much slower, compared to the *Quartet MaxCut*, it is more intuitive approach. Additionally, the differences between both types of informative P-blocks are automatically considered using this approach whereas both types are treated the same when we infer quartet topologies. Furthermore, this approach is not restricted to quartet blocks, but could be directly generalized to distances between "blocks" involving more than four sequences. If there are only strongly supported splits, then our encoding is identical to the approach in *MRP* [36].

Sequences	Data matrix M				
S_1	1	1	-	1	1
S_2	1	-	-	-	0
S_3	0	1	0	-	-
S_4	-	0	1	0	1
S_5	-	2	1	0	0
S_6	0	-	0	1	-

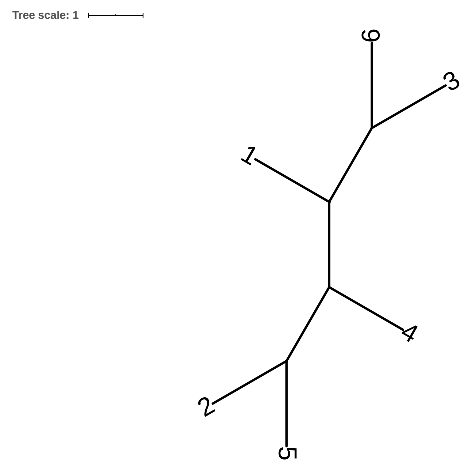

Figure 3: Data matrix M encoding distances for a set $\mathcal{B}$ of 5 informative P-block pairs for a set of 6 sequences, and tree topology reconstructed from M. Dashes represent 'missing information' in sequences not involved in a P-block pair. The matrix represents four *type 1* P-block pairs (columns 1, 3, 4, 5) and one *type 2* P-block pair (column 2). The resulting tree topology was calculated from the matrix with the program *pars* form the *PHYLIP* package [11].

2.3 Implementation

In order to find a suitable set $\mathcal{B}$ of P-block pairs for our approaches, we are using our software *Multi-SpaM*. This program samples up to 1 million P-blocks. Then, we generate lists of P-blocks involving the same four sequences. P-blocks in each such list are sorted according to their positions in one of the four sequences. For neighboring P-blocks in these lists, we calculate the distance S_r for each involved sequence S_r.

3 Comparison to a multiple sequence alignment

When an optimal multiple sequence alignment is available, it is possible to compare this new approach to the more traditional "simple encoding" of indel data. To this end, we generated artificial sequences with *ALF* [6] for which we know the optimal MSA. Since we are only interested in indels, we chose a very simple set of parameters and only simulated substitution and indel events. We used *ALF* to evolve 1000 randomly generated genes along simple phylogenetic trees. These trees were generated under a birth-and-death model with a birth rate of 0.1, a death rate of 0.01 and a mutation rate of 0.1. Each tree consists of 30 species.

Overall, we generated 30 datasets. We did 10 runs each for three different indel rates: 0.1, 0.01 and 0.001. The size of the indels was randomized under the Zipfian distribution with an exponent of 1.821 and a max length of 50. For every dataset, the indels were encoded using the "simple encoding" [41]. For this, we used the tool *2xread* [28]. Since the resulting matrix also contains the nucleotide data, we removed anything but the indel encoding. This encoding accurately represents the indels in the optimal alignment which we compared to the informative P-block pairs on these datasets. To this end, we considered every 4-set of sequences. Then, we found the most parsimonious quartet tree based on the "simple encoding". For our approach, we inferred quartet trees from the informative P-block pairs and found the most common topology for every 4-set. Then, we determined whether our approach finds the same topology that can be inferred from the multiple sequence alignment. The results can be found in in Figure 4. Here, we also checked for 4-sets of sequences for which there is no most parsimonious tree based on the true indel data. In these case, the quartet topology found by our approach is likely due to chance. We found that strongly supported splits are much more likely to reflect the true indel information. Weakly supported splits are much more abundant, but also found more often due to chance. Furthermore, we found that a high number of indels results in more splits conflicting with the true indel data.

4 Test results

In order to evaluate the above described approaches to phylogeny reconstruction, we used sets of genome sequences from *AF-Project* [54]. These sequences are frequently used as benchmark data for alignment-free methods; they were also used to evaluate *Multi-SpaM*. We ran *Multi-SpaM* with the default parameters except for a data sets of 14 plant genomes from *AF-project* for which we increased the pattern weight to 12 in order to keep the runtime manageable. First, we used informative P-block distances to infer quartet topologies for these genomes. Then, we used the two above described approaches, namely *Quartet MaxCut* and *Maximum-Parsimony*, to infer tree topologies for the entire input data sets.

Data set	Non-informative P-block pairs	Informative P-block pairs	
		type 2	*type 1*
27 *E.coli/Shigella*	0.866	0.118	0.016
29 *E.coli/Shigella*	0.859	0.123	0.017
25 fish mitochondria	0.819	0.161	0.019
14 plants	0.996	0.004	0.000
19 *Wolbachia*	0.916	0.062	0.021
8 *Yersinia*	0.991	0.008	0.001

Table 1: Propoprtion of non-informative and informative P-block pairs in the data sets from *AFproject*.

4.1 Quartet trees from P-block distances

First, we tested, how many of the P-block pairs that we found with our approach are *informative* of *type 1* or *type 2*, respectively. Table 1 shows the proportion of informative P-block pairs for these data sets. Next, we evaluated the correctness of the obtained quartet topologies by comparing them to the respective topologies of reference trees. For this, we used the ETE3 toolkit [21] to compute the Robinson-Foulds distance [37] between the two trees. If this distance is zero, then the quartet tree has the correct topology. However, a set of correct quartet trees is not sufficient to find a correct super tree [2, 46]. For every data set with n taxa, there are $\binom{n}{4}$ possible sets of four sequences. Ideally, for every such set, there should be at least one quartet tree in order to find the correct super tree. Thus, we also considered the *quartet coverage*, i.e. the fraction of these sets that have at least one quartet tree. The results can be found in Table 2. As can be seen, for most data sets, the quartet trees inferred from informative P-block distances are more accurate than the quartet trees found by *Multi-SpaM*. However, the number of quartet trees and quartet coverage is rather low. Especially, for the plants and *Yersinia* data sets, barely any quartet trees are found.

Data set	Method	Correctness	# quartet trees	Quartet coverage
27 *E.coli/Shigella*	*Multi-SpaM*	0.6942	597700	0.925
	all splits	0.7407	117800	0.658
	strong splits	**0.8046**	17000	0.287
29 *E.coli/Shigella*	*Multi-SpaM*	0.6655	587900	0.916
	all splits	0.763	130000	0.55
	strong splits	**0.8279**	15000	0.157
25 fish mitochondria	*Multi-SpaM*	0.6403	42500	0.704
	all splits	0.5919	6400	0.241
	strong splits	**0.6591**	1000	0.041
14 plants	*Multi-SpaM*	0.4676	998000	1
	all splits	**0.766**	4000	0.194
	strong splits	0.7196	<1000	0.021
19 *Wolbachia*	*Multi-SpaM*	0.7935	452400	1
	all splits	0.85	83900	0.885
	strong splits	**0.9567**	21500	0.308
8 *Yersinia*	*Multi-SpaM*	0.3665	35000	1
	all splits	0.3511	9000	1
	strong splits	**0.4158**	1000	1

Table 2: For 6 data sets from the AF-Project, it is shown how many quartet trees are correct according to the reference trees. Further, the number of quartet trees as well as the fraction of sets of four sequences with at least one quartet tree is given.

Comparison to *Multi-SpaM*

It is an interesting question whether our approach finds the same quartet topologies as the corresponding *Multi-SpaM* run. To this end, we considered every 4-set of sequences and found the most common quartet topology for each set. As shown in Table 3, the quartet trees are very accurate when both methods agree on the topology. On the other hand, the accuracy is particularly low when the quartet trees from our approach disagree with the topology of the *Multi-SpaM* quartet trees. The only exception is the *Yersinia* data set. However, the quartet trees of *Multi-SpaM* were already only slightly better than random quartet trees to begin with for this dataset.

Additional P-blocks

It should be noted that there may be many 4-sets of sequences with no P-blocks or only one single P-block. This can reduce the number of informative P-block pairs to the point where it is hard to accurately reconstruct the phylogeny. Thus, in cases, where *Multi-SpaM* found a single P-block for a 4-set of sequences, we searched for additional P-blocks in the vincinity of the P-block found by *Multi-SpaM*, to obtain additional informative P-block pairs. These additional P-blocks have to be within a range of 10000 and can be significantly shorter. Here, we used patterns without *don't care* positions and with a weight between 6 and 10. We also tried to use this approach with lower search ranges. However, this led to very few additional P-blocks being found in many cases. The only exception was the fish mitochondria dataset.

We tested how many of the informative P-block distances produce correct quartet trees with respect to reference tree when we include the additional P-blocks. Additionally, we checked if the quartet coverage could be improved.

Table 4 shows that for the two *E.coli/Shigella* datasets as well as the fish mitochondria, a lot of additional P-blocks could be found. In these cases, the correctness was reduced while the quartet coverage was improved. For a weight of 6, less additional P-blocks could be found. This is caused by words that appear twice in one sequence which are consequently removed. Interestingly, for the *Wolbachia* dataset, only very few additional P-blocks could be found which resulted in a slight overall improvement. For the other two datasets, we found no additional P-blocks with our search.

4.2 Full phylogeny reconstruction

Finally, we applied our approach to reconstruct full phylogenetic trees for the benchmark sequences from *AFproject*. Here, we used the two approaches

Quartet MaxCut and *Maximum-Parsimony*, as described above. We ran *Multi-SpaM* 10 times and checked whether the results could be improved by using P-block distances. The results of other alignment-free methods on these data are reported in [7, 54]. Since from the plant and *Yersinia* genomes we could not obtain a sufficient number of informative P-block pairs, we were unable to produce trees for these data sets.

Reconstructed phylogenies were measured against the respective reference trees. For that, we used the Robinson-Foulds distance [37] and normalized them so that they can be compared across multiple data sets. For a data set with n taxa, the RF distances are divided by $2 * n - 6$ in order to normalize them.

Quartet MaxCut

We applied the program *Quartet MaxCut* first to the quartet topologies derived from *all* informative P-block pairs. Alternatively, we restricted us to the set of *type-1* P-block pairs, *i.e.* the P-block pairs that *strongly* support one of the three possible topologies for the four involved sequences.

Next, we combined our new approach with *Multi-SpaM*. Since this tool also uses *Quartet MaxCut*, it is straightforward to combine the sets of quartet trees calculated with our new method and with *Multi-SpaM*. Again, we used both, the full set of *informative P-block pairs*, and the *type 1* P-block pairs only. As shown in Table 3, the resulting quartet trees are highly accurate in those situations where the quartet trees from *Multi-SpaM* agree with the quartet topologies inferred with informative P-block pairs. Therefore, we also tried to use these quartet trees to build a phylogenetic tree ("intersection"). Lastly, we tested the quartet trees generated with the additional P-blocks that were found in order to increase the quartet coverage. The results can be found in Figure 5. The error bars show the maximum and minimum of all runs.

For both *E.coli/Shigella* data sets, the indel data could be used to improve the normalized RF distance on average or to decrease the variance. However, the indel data alone was not enough to built accurate phylogenetic trees. The only exception was the *Wolbachia* data set. For most test cases, the indel data could be used to reconstruct more accurate trees than *Multi-SpaM*. Overall, the additional P-blocks could not be used to improve the quality of the trees.

Maximum-Parsimony

Next, we generated a character matrix from the informative P-block pairs, as described in the Method section. Subsequently, we used *Maximum-Parsimony* to infer a phylogenetic tree. Here, we used *PAUP** [47] to calculate the most parsimonious tree with the *TBR* [9] heuristic. In this case, we did not differentiate between strongly and weakly supported splits as this is taken into account automatically due to the way we defined our matrix. In some cases, *PAUP** returned multiple most parsimonious trees. We calculated the average Robinson-Foulds distance and the error bars over all resulting trees.

Figure 6 shows the comparison of *Maximum-Parsimony* with *Quartet MaxCut* and the corresponding *Multi-SpaM* run. For all data sets except for the fish mitochondria, the average normalized RF distances are lower with *Maximum-Parsimony*. However, the variance is much larger due to multiple most parsimonious trees being found. Additionally, the runtime is considerable higher, especially when there is not enough data to find a single most parsimonious tree.

5 Discussion

Indel information is largely neglected in phylogenetic studies, despite evidence of its usefulness. For more traditional methods that rely on multiple sequence alignment, it is fairly straightforward to postulate a phylogenetic relationship based on the position and size of the gaps in the alignment. In recent years, many alignment-free methods have been proposed to tackle the ever-increasing amount of sequence data with high speed. Without an alignment, inferring information about indel events is a much more challenging task. In this paper, we proposed a way to gather such information with an alignment-free method. To our knowledge, this is the first attempt being made in this direction.

Most of the alignment-free methods calculate pairwise distances, e.g. from the number of word matches. In such a setting, it is hard to find and utilize evidence of indel events. First of all, only homologous word matches can be taken into consideration. Recently, some alignment-free methods have been proposed that use micro-alignments. These methods require matches to be sufficiently long such that homologies can be found with high probability. Alternatively, some methods use spaced-word matches which can be scored in order to discard random matches. More recently, *Multi-SpaM* extended this approach and used so-called P-blocks, matching spaced-word occurrences in four sequences, to reconstruct phylogenies.

In this paper, we compared the distances between pairs of P-blocks to find putative indel events. If the distances were the same in two sequences while being different in the other two, then we use such a split to derive a quartet tree. We showed in our evaluation that these quartet trees are in most cases highly accurate when compared to the reference trees. The quartet trees were particularly accurate when they coincided with the topology of the *Multi-SpaM* quartet trees. Thus, the indel information could be used to reinforce existing information, e.g. by giving these trees a higher weight when trying to reconstruct the phylogenetic tree.

One might think that pairs of P-blocks should only be compared in relatively close proximity in order to reduce the noise. However, we derived many correct quartet trees from informative P-block distances even over large distances. In most cases, the distances are different in all sequences for pairs of P-blocks when there is nothing to find. While we did find evidence of quartet trees being found by chance when we compared them against an alignment, we found that the most effective way of reducing this noise is to only consider strongly supported splits, i.e. only splits where the distances are equal on each side of the split.

Choosing whether to use all or only strongly supported splits is the only important setting for our approach. Other than that, it depends on the parameters of *Multi-SpaM*. Here, we relied on the parameter settings used in the evaluation of *Multi-SpaM*. The high number of *don't care positions* is necessary to distinguish homologous spaced-word matches from background noise. Apart from that, it is only important to have a high enough number of P-blocks such that sufficiently many informative pairs of P-blocks can be found. The potential is, however, limited as the weight of the pattern cannot be reduced further without increasing the number of random matches drastically. Moreover, the number of samples cannot be increased further for some of the smaller datasets.

Finding enough informative pairs of P-blocks is a major challenge for our approach. For many 4-sets of sequences, we could not find any informative pairs of P-blocks. This is especially problematic when we tried to build phylogenetic trees with *Quartet MaxCut*. This challenge is even harder when only strongly supported splits are taken into account. Even though these splits are highly accurrate, the usefulness of them is rather limited when their number is too low. We tried to find additional P-blocks as a countermeasure. However, we found that the additional quartet trees were not of much use as the much higher number of trees came at the cost of lower accuracy.

Another limitation that we observed was that our approach only worked for closely related species. We tried to find evidence of indel events in a dataset of 14 plant genomes which are relatively distantly related. Here, we only found a very small number of informative pairs of P-blocks as most of the distances were different in all sequences when we compared pairs of P-blocks. One reason could be a large number of indel or other evolutionary events. Further, it should be noted that the plant dataset is by far the largest one with 4.5 *gb*. The P-blocks are therefore likely to be very far apart in comparison to the other datasets. In contrast, we found many informative pairs of P-blocks for closely related species. For the *Wolbachia* dataset, we even managed to infer the same topology as *Multi-SpaM* based on the putative indels alone.

We used informative P-block distances to reconstruct phylogenetic trees with both *Quartet MaxCut* and *Maximum-Parsimony*. When we used *Quartet MaxCut*, the best results could be achieved when we combined the quartet trees from the corresponding run of *Multi-SpaM* with the quartet trees derived from the P-block distances. Here, the overall quality of the trees could be improved in some cases even though there was no consistent method of achieving this. With *Maximum-Parsimony*, we could in some cases improve the phylogenetic tree in comparison to *Multi-SpaM* just by using the quartet trees derived from the P-block distances. On the downside, oftentimes there are not enough parsimony-informative characters which results in many most-parsimonous trees being calculated. Furthermore, the indel data cannot be combined as easily with the quartet trees from *Multi-SpaM*.

The focus of this study were quartet trees which is rather arbitrary. With *Maximum-Parsimony*, this approach could easily be extended to larger P-blocks. For us, this was a first step in this direction which allowed us to showcase the quality of the indel information in more detail. Overall, the best application of this method is to use the information as weights or in addition to other information such as quartet trees as in this study.

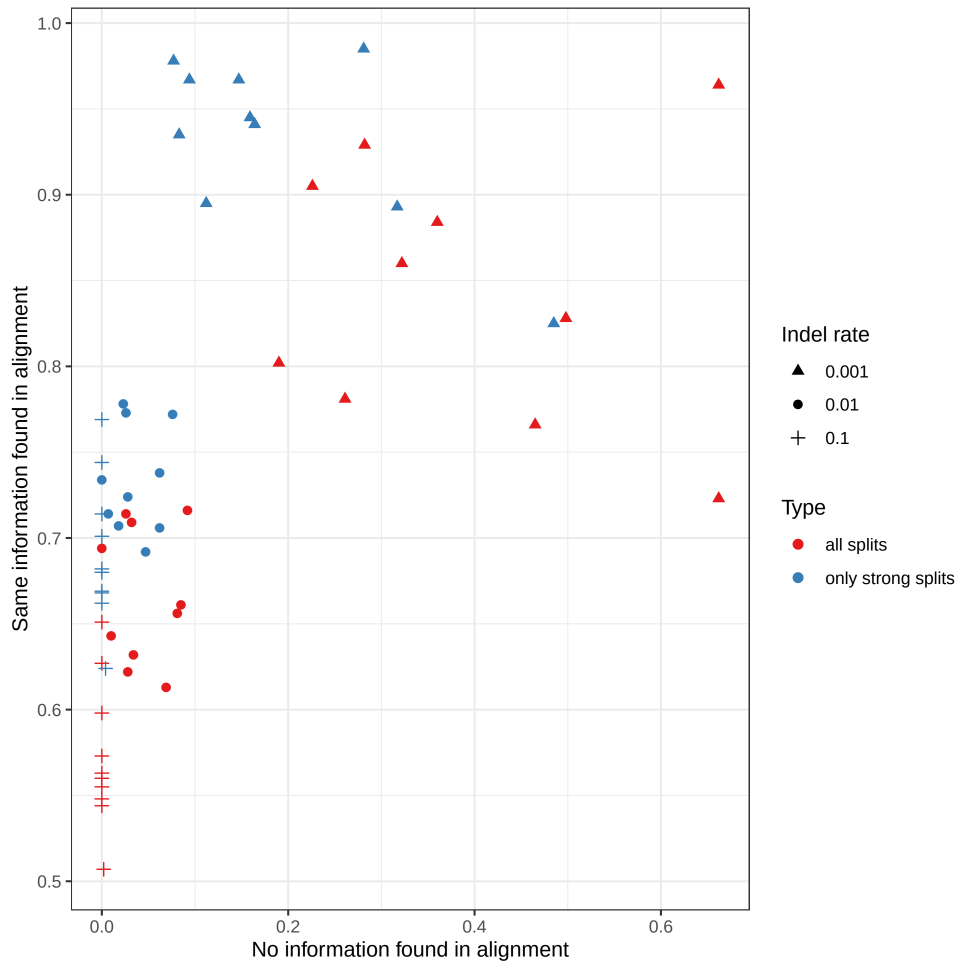

Figure 4: The information contained in the alignment is compared to the most common quartet tree found by our approach for all possible 4-sets of sequences. If for such a set, there is no indel data to support any split, i.e. no information was found in the alignment, then the split is likely to be noise. We ran 10 runs for three different indel rates. The red symbols show the values for all splits, while the blue symbols only show the data for strongly supported splits.

Data set	Same topology	*Multi-SpaM*	Gap sizes	Total
Most common quartet tree found by both methods				
27 *E.coli/Shigella*	7153 (**0.948**)	2013 (0.697)	2013 (0.229)	16244
29 *E.coli/Shigella*	8652 (**0.949**)	1888 (0.725)	1888 (0.159)	21763
25 fish mitochondria	1410 (**0.933**)	588 (0.684)	588 (0.194)	8911
14 plants	137 (**0.952**)	33 (0.746)	33 (0.138)	1001
19 *Wolbachia*	2848 (**0.974**)	213 (0.645)	214 (0.213)	3876
8 *Yersinia*	39 (0.457)	14 (0.482)	14 (0.150)	70
Most common quartet tree found by only one method				
27 *E.coli/Shigella*		5677 (0.794)	346 (0.491)	16244
29 *E.coli/Shigella*		8999 (0.766)	461 (0.418)	21763
25 fish mitochondria		5247 (0.639)	401 (0.539)	8911
14 plants		801 (0.713)	-	1001
19 *Wolbachia*		620 (0.866)	46 (0.496)	3876
8 *Yersinia*		-	-	70

Table 3: The quartet trees inferred from informative P-block pairs are compared to the quartet trees from *Multi-SpaM*. Here, we consider only the most common quartet tree topology for every 4-set of sequences. For every case, the correctness of the quartet trees (with respect to the reference tree) is shown next to the number of quartet trees (in brackets).

Dataset	Method	Correctness	# quartet trees	Quartet coverage
27 *E.coli/Shigella*	default	0.7407	117800	0.658
	weight = 6	0.5648	3838170	0.945
	weight = 7	0.5672	5544670	0.945
	weight = 8	0.5922	3028000	0.945
	weight = 9	0.6238	1112000	0.945
	weight = 10	0.6572	439333	0.945
29 *E.coli/Shigella*	default	0.763	130000	0.55
	weight = 6	0.5233	863800	0.55
	weight = 7	0.5089	1232100	0.573
	weight = 8	0.5392	690500	0.573
	weight = 9	0.6122	300600	0.573
	weight = 10	0.6952	176800	0.573
25 fish mitochondria	default	0.5919	6400	0.241
	weight = 6	0.5025	1803500	0.581
	weight = 7	0.4972	1809000	0.581
	weight = 8	0.5179	801300	0.58
	weight = 9	0.5393	236200	0.58
	weight = 10	0.5397	99300	0.579
14 plants	default	0.766	4000	0.194
	weight = 6 – 10	0.766	4000	0.194
19 *Wolbachia*	default	0.85	83900	0.885
	weight = 6	0.8486	86000	0.886
	weight = 7	0.8495	87400	0.886
	weight = 8	0.8513	86400	0.886
	weight = 9	0.8511	84900	0.886
	weight = 10	0.8503	84300	0.886
8 *Yersinia*	default	0.3511	9000	1
	weight = 6 – 10	0.3511	9000	1

Table 4: For 6 datasets from the AF-Project, the number of additional quartet trees, that can be found with a pattern of a certain weight, are shown. The correctness with regard to the reference tree as well as the quartet coverage is also given. The default values are from the set of quartet trees before the search.

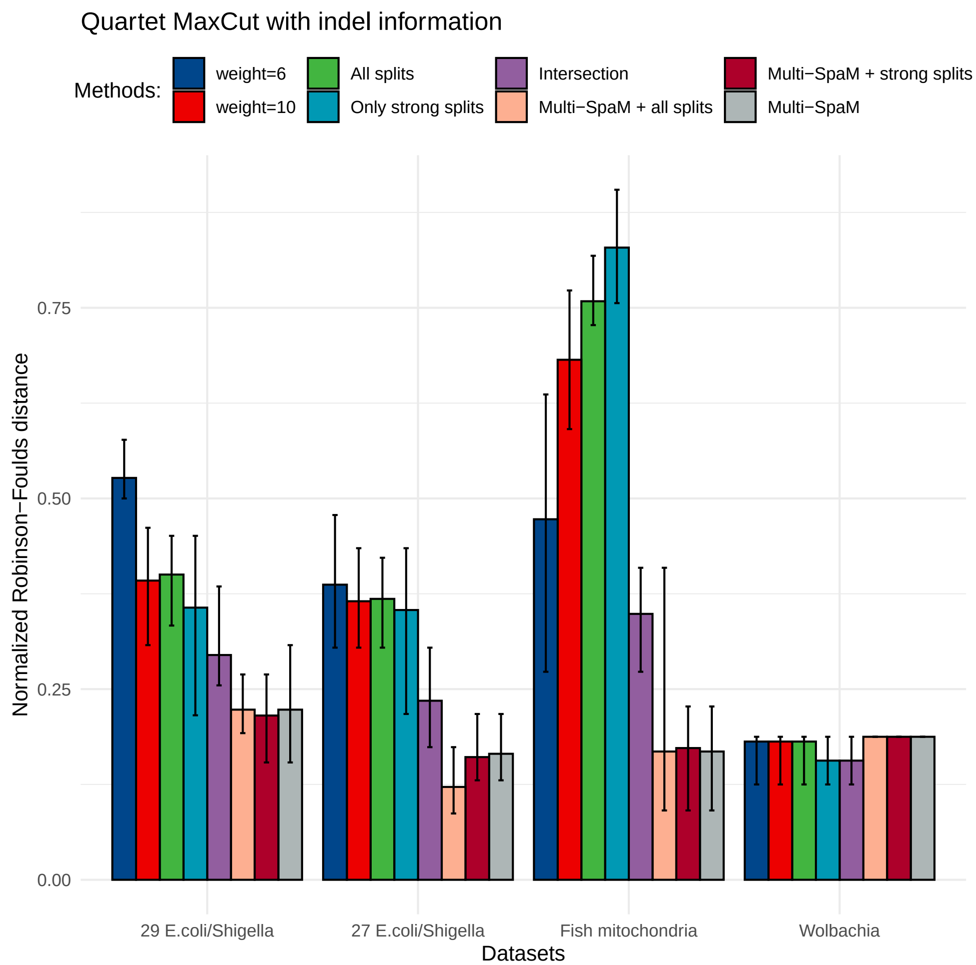

Figure 5: We derived quartet trees from the informative P-block pairs. We used *Quartet MaxCut* to build phylogenetic tree with multiple different sets of quartet trees. First, we used all quartet trees including derived from all informative P-block pairs. Then, we only used the quartet trees based on strongly supported splits. Both of these sets were also used in combination with the quartet trees of the corresponding run of *Multi-SpaM*. Furthermore, we tried to find additional quartet trees by searching for additional blocks with patterns of weight 6 and 10. Lastly, we calculated the most common quartet topologies for every 4-set of sequences. We used the set of quartet trees where the most common topology agreed with the most common topologies of the *Multi-SpaM* trees. We called this set "intersection".

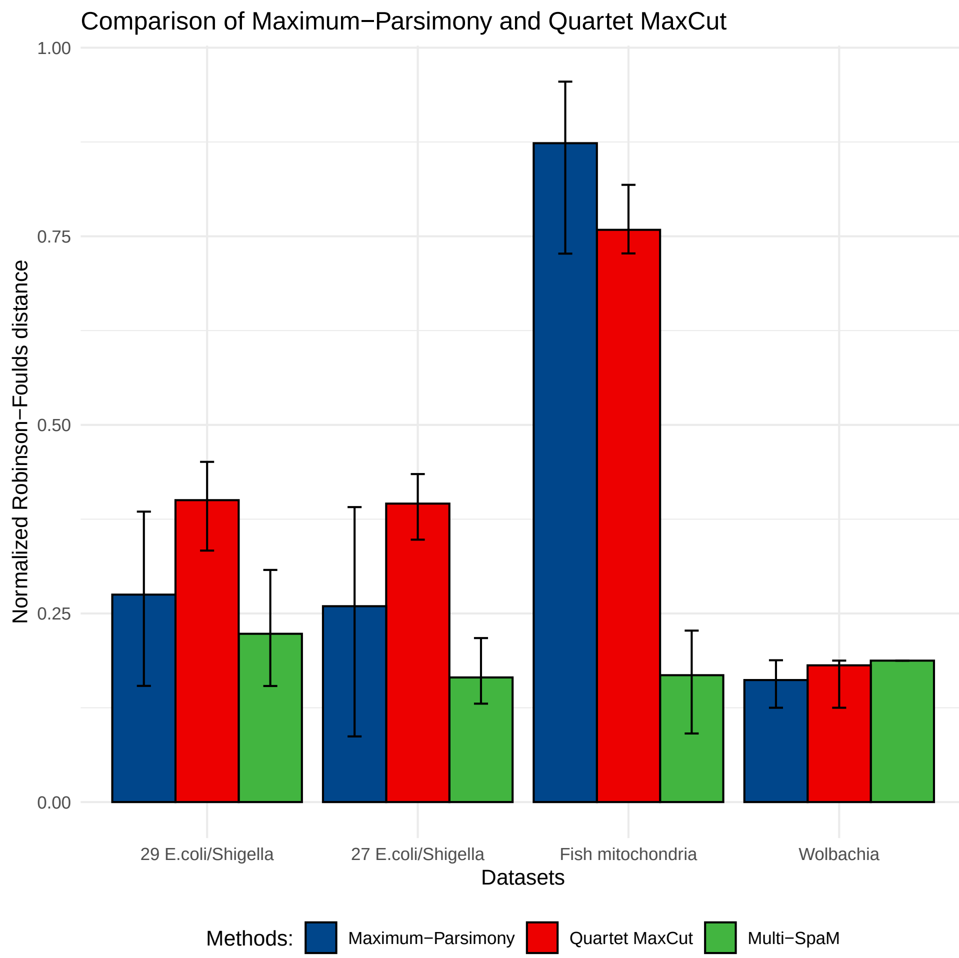

Figure 6: We encoded the gap sizes between adjacent blocks as parsimonious-informative characters and built phylogenetic trees with *Maximum-Parsimony*. We compared it to the corresponding runs of *Multi-SpaM* and to *Quartet MaxCut* which was used on the set of all splits. The variance of the Robinson-Foulds distances is higher for *Maximum-Parsimony* because multiple most parsimonious trees were found in several cases.

References

[1] A. V. Alekseyenko, C. J. Lee, and M. A. Suchard. Wagner and Dollo: a stochastic duet by composing two parsimonious solos. *Syst. Biol.*, 57(5):772–784, Oct 2008.

[2] E. Avni, Z. Yona, R. Cohen, and S. Snir. The Performance of Two Supertree Schemes Compared Using Synthetic and Real Data Quartet Input. *J. Mol. Evol.*, 86(2):150–165, 02 2018.

[3] G. Bernard, C. X. Chan, Y. B. Chan, X. Y. Chua, Y. Cong, J. M. Hogan, S. R. Maetschke, and M. A. Ragan. Alignment-free inference of hierarchical and reticulate phylogenomic relationships. *Brief. Bioinformatics*, 20(2):426–435, 03 2019.

[4] Olaf R.P. Bininda-Emonds. The evolution of supertrees. *Trends in Ecology and Evolution*, 19:315 – 322, 2004.

[5] Francesca Chiaromonte, Von Bing Yap, and Webb Miller. Scoring pairwise genomic sequence alignments. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, and Teri E. Klein, editors, *Pacific Symposium on Biocomputing*, pages 115–126, Lihue, Hawaii, 2002.

[6] Daniel A. Dalquen, Maria Anisimova, Gaston H. Gonnet, and Christophe Dessimoz. ALF - a simulation framework for genome evolution. *Molecular Biology and Evolution*, 29:1115–1123, 2012.

[7] Thomas Dencker, Chris-Andr Leimeister, Michael Gerth, Christoph Bleidorn, Sagi Snir, and Burkhard Morgenstern. Multi-SpaM: a maximum-likelihood approach to phylogeny reconstruction using multiple spaced-word matches and quartet trees. *NAR Genomics and Bioinformatics*, 2(1), 3 2020. lqz013.

[8] C. Dessimoz and M. Gil. Phylogenetic assessment of alignments reveals neglected tree signal in gaps. *Genome Biol.*, 11(4):R37, 2010.

[9] GJ Olsen DL Swofford. Phylogeny reconstruction. In DM Hillis and C. Moritz, editors, *Molecular Systematics*, pages 411–501. Sinauer Associates: Sunderland, Massachusetts, 1990.

[10] James S. Farris. Methods for computing Wagner trees. *Systematic Biology*, 19:83–92, 1970.

[11] Joseph Felsenstein. PHYLIP - Phylogeny Inference Package (Version 3.2). *Cladistics*, 5:164–166, 1989.

[12] Walter Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Zoology*, 20:406–416, 1971.

[13] O. Gascuel. BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Mol. Biol. Evol.*, 14(7):685–695, Jul 1997.

[14] S. Guindon, J. F. Dufayard, V. Lefort, M. Anisimova, W. Hordijk, and O. Gascuel. New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0. *Syst. Biol.*, 59(3):307–321, May 2010.

[15] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, Cambridge, UK, 1997.

[16] B. Haubold. Alignment-free phylogenetics and population genetics. *Brief. Bioinformatics*, 15(3):407–418, May 2014.

[17] Bernhard Haubold, Fabian Klötzl, and Peter Pfaffelhuber. andi: Fast and accurate estimation of evolutionary distances between closely related genomes. *Bioinformatics*, 31:1169–1175, 2015.

[18] I. H. Holmes. Solving the master equation for Indels. *BMC Bioinformatics*, 18(1):255, May 2017.

[19] Sebastian Horwege, Sebastian Lindner, Marcus Boden, Klaus Hatje, Martin Kollmar, Chris-André Leimeister, and Burkhard Morgenstern. *Spaced words* and *kmacs*: fast alignment-free sequence comparison based on inexact word matches. *Nucleic Acids Research*, 42:W7–W11, 2014.

[20] Peter Houde, Edward L. Braun, Nitish Narula, Uriel Minjares, and Siavash Mirarab. Phylogenetic signal of indels and the neoavian radiation. *Diversity*, 11(7), 2019.

[21] J. Huerta-Cepas, F. Serra, and P. Bork. ETE 3: Reconstruction, Analysis, and Visualization of Phylogenomic Data. *Mol. Biol. Evol.*, 33(6):1635–1638, 06 2016.

[22] Anna Katharina Lau, Svenja Dörrer, Chris-Andre Leimeister, Christoph Bleidorn, and Burkhard Morgenstern. Read-spam: assembly-free and alignment-free comparison of bacterial genomes with low sequencing coverage. *bioRxiv*, 2019.

[23] C. A. Leimeister, J. Schellhorn, S. Dorrer, M. Gerth, C. Bleidorn, and B. Morgenstern. Prot-SpaM: fast alignment-free phylogeny reconstruction based on whole-proteome sequences. *Gigascience*, 8(3), 03 2019.

[24] Chris-André Leimeister, Marcus Boden, Sebastian Horwege, Sebastian Lindner, and Burkhard Morgenstern. Fast alignment-free sequence comparison using spaced-word frequencies. *Bioinformatics*, 30:1991–1999, 2014.

[25] Chris-André Leimeister and Burkhard Morgenstern. *kmacs*: the k-mismatch average common substring approach to alignment-free sequence comparison. *Bioinformatics*, 30:2000–2008, 2014.

[26] Chris-André Leimeister, Salma Sohrabi-Jahromi, and Burkhard Morgenstern. Fast and accurate phylogeny reconstruction using filtered spaced-word matches. *Bioinformatics*, 33:971–979, 2017.

[27] Ross A. Lippert, Haiyan Huang, and Michael S. Waterman. Distributional regimes for the number of k-word matches between two random sequences. *Proceedings of the National Academy of Sciences*, 99:13980–13989, 2002.

[28] D.P. Little. 2xread: a simple indel coding tool. https://www.nybg.org/files/scientists/2xread.html., 2005. [Online; accessed December-2019].

[29] I. Mikls, G. A. Lunter, and I. Holmes. A Long Indel Model For Evolutionary Sequence Alignment. *Molecular Biology and Evolution*, 21(3):529–540, 03 2004.

[30] B. Morgenstern, S. Schobel, and C. A. Leimeister. Phylogeny reconstruction based on the length distribution of k-mismatch common substrings. *Algorithms Mol Biol*, 12:27, 2017.

[31] Burkhard Morgenstern. Sequence comparison without alignment: The spam approaches. *bioRxiv*, 2019.

[32] Burkhard Morgenstern, Bingyao Zhu, Sebastian Horwege, and Chris-André Leimeister. Estimating evolutionary distances between genomic sequences from spaced-word matches. *Algorithms for Molecular Biology*, 10:5, 2015.

[33] K. Muller. Incorporating information from length-mutational events into phylogenetic analysis. *Mol. Phylogenet. Evol.*, 38(3):667–676, Mar 2006.

[34] T. H. Ogden and M. S. Rosenberg. How should gaps be treated in parsimony? A comparison of approaches using simulation. *Mol. Phylogenet. Evol.*, 42(3):817–826, Mar 2007.

[35] Ji Qi, Hong Luo, and Bailin Hao. CVTree: a phylogenetic tree reconstruction tool based on whole genomes. *Nucleic Acids Research*, 32(suppl 2):W45–W47, 2004.

[36] M. A. Ragan. Phylogenetic inference based on matrix representation of trees. *Mol. Phylogenet. Evol.*, 1(1):53–58, Mar 1992.

[37] David F Robinson and Les Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131–147, 1981.

[38] Sophie Röhling, Alexander Linne, Jendrik Schellhorn, Morteza Hosseini, Thomas Dencker, and Burkhard Morgenstern. The number of k-mer matches between two dna sequences as a function of k and applications to estimate phylogenetic distances. *bioRxiv*, 2019.

[39] Fredrik Ronquist and John P. Huelsenbeck. MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19:1572–1574, 2003.

[40] Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

[41] M. P. Simmons and H. Ochoterena. Gaps as characters in sequence-based phylogenetic analyses. *Syst. Biol.*, 49(2):369–381, Jun 2000.

[42] Gregory E. Sims, Se-Ran Jun, Guohong A. Wu, and Sung-Hou Kim. Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. *Proceedings of the National Academy of Sciences*, 106:2677–2682, 2009.

[43] S. Snir and S. Rao. Quartets MaxCut: a divide and conquer quartets algorithm. *IEEE/ACM Trans Comput Biol Bioinform*, 7(4):704–718, 2010.

[44] Sagi Snir and Satish Rao. Quartet MaxCut: A fast algorithm for amalgamating quartet trees. *Molecular Phylogenetics and Evolution*, 62:1 – 8, 2012.

[45] Alexandros Stamatakis. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30:1312–1313, 2014.

[46] M. S. Swenson, R. Suri, C. R. Linder, and T. Warnow. An experimental study of Quartets MaxCut and other supertree methods. *Algorithms Mol Biol*, 6:7, Apr 2011.

[47] D. Swofford. Paup*. phylogenetic analysis using parsimony (*and other methods). version 4.0b10. *Sinauer Associates, Sunderland, Massachusetts*, 2003.

[48] J. L. Thorne, H. Kishino, and J. Felsenstein. An evolutionary model for maximum likelihood alignment of DNA sequences. *J. Mol. Evol.*, 33(2):114–124, Aug 1991.

[49] Jeffrey L. Thorne, Hirohisa Kishino, and Joseph Felsenstein. Inching toward reality: An improved likelihood model of sequence evolution. *Journal of Molecular Evolution*, 34(1):3–16, Jan 1992.

[50] Igor Ulitsky, David Burstein, Tamir Tuller, and Benny Chor. The average common substring approach to phylogenomic reconstruction. *Journal of Computational Biology*, 13:336–350, 2006.

[51] S. Vinga and J. Almeida. Alignment-free sequence comparison-a review. *Bioinformatics*, 19(4):513–523, Mar 2003.

[52] Huiguang Yi and Li Jin. Co-phylog: an assembly-free phylogenomic approach for closely related organisms. *Nucleic Acids Research*, 41:e75, 2013.

[53] C. Zhang, M. Rabiee, E. Sayyari, and S. Mirarab. ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. *BMC Bioinformatics*, 19(Suppl 6):153, 05 2018.

[54] A. Zielezinski, H. Z. Girgis, G. Bernard, C. A. Leimeister, K. Tang, T. Dencker, A. K. Lau, S. Rohling, J. J. Choi, M. S. Waterman, M. Comin, S. H. Kim, S. Vinga, J. S. Almeida, C. X. Chan, B. T. James, F. Sun, B. Morgenstern, and W. M. Karlowski. Benchmarking of alignment-free sequence comparison methods. *Genome Biol.*, 20(1):144, 07 2019.

4 Accurate multiple alignment of distantly related genome sequences using filtered spaced word matches as anchor points.

Reference

Chris-André Leimeister, Thomas Dencker, Burkhard Morgenstern, Accurate multiple alignment of distantly related genome sequences using filtered spaced word matches as anchor points, Bioinformatics, Volume 35, Issue 2, 15 January 2019, Pages 211–218, `https://doi.org/10.1093/bioinformatics/bty592`. [74]

Original Contribution

CAL designed the study, implemented most of the modified version of *FSWM* and performed the program evaluation. TD contributed to the implementation. BM supervised the project. CAL and BM wrote the manuscript. All authors read and approved the final version of the manuscript.

Bioinformatics, 35(2), 2019, 211–218
doi: 10.1093/bioinformatics/bty592
Advance Access Publication Date: 10 July 2018
Original Paper

OXFORD

Genome analysis

Accurate multiple alignment of distantly related genome sequences using filtered spaced word matches as anchor points

Chris-André Leimeister[1,*], Thomas Dencker[1] and Burkhard Morgenstern[1,2,*]

[1]Department of Bioinformatics, Institute of Microbiology and Genetics and [2]Center for Computational Sciences, University of Goettingen, 37077 Goettingen, Germany

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on May 23, 2017; revised on October 3, 2017; editorial decision on June 29, 2018; accepted on July 9, 2018

Abstract

Motivation: Most methods for pairwise and multiple genome alignment use fast local homology search tools to identify *anchor points*, i.e. high-scoring local alignments of the input sequences. Sequence segments between those anchor points are then aligned with slower, more sensitive methods. Finding suitable anchor points is therefore crucial for genome sequence comparison; speed and sensitivity of genome alignment depend on the underlying anchoring methods.
Results: In this article, we use *filtered spaced word matches* to generate anchor points for genome alignment. For a given binary pattern representing *match* and *don't-care* positions, we first search for *spaced-word matches*, i.e. ungapped local pairwise alignments with matching nucleotides at the *match* positions of the pattern and possible mismatches at the *don't-care* positions. Those spaced-word matches that have similarity scores above some threshold value are then extended using a standard *X*-drop algorithm; the resulting local alignments are used as anchor points. To evaluate this approach, we used the popular multiple-genome-alignment pipeline *Mugsy* and replaced the exact word matches that *Mugsy* uses as anchor points with our spaced-word-based anchor points. For closely related genome sequences, the two anchoring procedures lead to multiple alignments of similar quality. For distantly related genomes, however, alignments calculated with our *filtered-spaced-word matches* are superior to alignments produced with the original *Mugsy* program where exact word matches are used to find anchor points.
Availability and implementation: http://spacedanchor.gobics.de
Contact: chris.leimeister@stud.uni-goettingen.de or bmorgen@gwdg.de
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The most fundamental task in biological sequence analysis is to *align* two or several nucleic-acid or protein sequences—either *globally*, over their entire length, or *locally*, by restricting the alignment to a single region of homology. Standard approaches to global alignment assume that the input sequences derived from a common ancestor, and that evolutionary events are limited to substitutions and small insertions and deletions. In this case, sequence homologies can be represented by *global sequence alignments*, that is, by inserting *gap characters* into the sequences such that evolutionarily related sequence positions are arranged on top of each other. Under most scoring schemes, calculating an *optimal* alignment of two sequences takes time proportional to the product of their lengths and is therefore limited to rather short sequences (Durbin *et al.*, 1998; Gotoh, 1982; Morgenstern, 2002; Needleman and Wunsch, 1970; Smith and Waterman, 1981).

With the rapidly increasing number of partially or fully sequenced genomes, alignment of *genomic* sequences has become an important field of research in bioinformatics, see Earl *et al.* (2014) for a recent review and evaluation of some of the most popular approaches. Here, the first challenge is the sheer size of the input sequences which makes it impossible to use traditional algorithms with quadratic run time. A second challenge is the fact that related genomes often share *multiple* local homologies, interrupted by non-conserved parts of the sequences where no significant similarities can be detected. This means that neither *global* alignment methods (Needleman and Wunsch, 1970) nor strictly *local* methods (Altschul *et al.*, 1990; Smith and Waterman, 1981) are appropriate to represent the homologies between entire genomes. Finally, homologies do not generally occur in the same relative order in different genomes, because of duplications and large-scale genome rearrangements. Since it is not possible, in general, to represent homologies among genomes in one single alignment, advanced genome aligners return alignments of so-called *Locally Collinear Blocks*, i.e. blocks of segments of the input sequences where orthologous genes appear in the same linear order.

Since the late 1990s, efforts have been made to a address the above issues, and many approaches to genome-sequence alignment have been published. One of the first multiple-alignment programs that could be applied to genomic sequences was *DIALIGN* (Morgenstern *et al.*, 1996, 2002). This program composes multiple alignments from chains of local pairwise alignments, and it does not penalize gaps; it is therefore able to align sequences where local homologies are separated by non-homologous regions. The program was initially not designed for large genomic sequences, though, and it is limited to sequences up to around 10 *kb*. Moreover, *DIALIGN* is not able to deal with duplications, rearrangements or homologies on different strands of the DNA double helix.

To align longer sequences, most programs for genomic alignment rely on some sort of *anchoring* (Huang *et al.*, 2006; Morgenstern *et al.*, 2006). In a first step, they use a fast local alignment method to identify high-scoring local homologies, so-called *anchor points*. Next, *chains* of such local alignments are calculated and, finally, sequence segments between the selected anchor points are aligned with a slower but more sensitive alignment method. For multiple sequence sets, either pairwise or multiple local alignments can be used as anchor points. A pioneering tool to find anchor points for genomic alignment is *MUMmer* (Delcher *et al.*, 1999); the current version of the program is considered the state-of-the-art in alignment anchoring (Kurtz *et al.*, 2004). MUMmer uses *maximal unique matches* as pairwise anchor points. The genome aligner *MGA*, by contrast, uses *maximal exact matches* involving *all* input sequences (Höhl *et al.*, 2002). Both *MUMmer* and *MGA* use *suffix trees* (Kurtz, 1999) and related data structures to rapidly identify the pairwise or multiple word matches. *MUMmer* and *MGA* can rapidly align entire bacterial genomes; *MUMmer* was also used in the *A. thaliana* genome project (The Arabidopsis Genome Initiative, 2000). However, since the number of exact word matches decreases with increasing evolutionary distances, these approaches are most useful if closely related genomes are to be compared, such as different strains of *E. coli*.

Other approaches to genome alignment are *OWEN* (Ogurtsov *et al.*, 2002), *AVID* (Bray *et al.*, 2003), *MAVID* (Bray and Pachter, 2003), *LAGAN and Multi-LAGAN* (Brudno *et al.*, 2003b), *CHAOS/DIALIGN* (Brudno *et al.*, 2003a), the *VISTA genome pipeline* (Dubchak *et al.*, 2009), *TBA* (Blanchette *et al.*, 2004) and *Mauve* (Darling *et al.*, 2004), see Dewey and Pachter (2006) and Batzoglou (2005) for review. All of these methods use anchor points, and most of them are able to deal with duplications and genome rearrangements. Some genome aligners use *statistical* properties of the sequences (Bradley *et al.*, 2009; Darling *et al.*, 2004); other methods are based on *graphs*, for example on *A-Bruijn graphs* (Raphael *et al.*, 2004) or on *cactus graphs* (Paten *et al.*, 2011). A further development of *Mauve*, called *progressiveMauve* (Darling *et al.*, 2010), uses palindromic *spaced seeds* (Darling *et al.*, 2006) instead of exact word matches as anchor points. Spaced seeds are used for sequence-analysis tasks such as database searching (Choi *et al.*, 2004; Ma *et al.*, 2002; Noé, 2017; Xu *et al.*, 2006), read mapping (Břinda *et al.*, 2015; David *et al.*, 2011; Langmead *et al.*, 2009; Noé *et al.*, 2010; Ounit and Lonardi, 2015), alignment-free sequence comparison (Leimeister *et al.*, 2014) or pathogen detection Deneke *et al.* (2017). Such pattern-based approaches are often superior to methods based on contiguous words or word matches, see for example Li *et al.* (2006). In *Mauve*, palindromic patterns are used to cover both DNA strands of the input sequences.

Mugsy (Angiuoli and Salzberg, 2011) is a popular software pipeline for multiple genome alignment. In a first step, this program uses *nucmer* (Kurtz *et al.*, 2004) to construct all pairwise alignments of the input sequences. *Nucmer*, in turn, uses *MUMmer* to find exact unique word matches which are used as alignment anchor points. An *alignment graph* is constructed from these pairwise alignments using the *SeqAn* software (Döring *et al.*, 2008), and *Locally Collinear Blocks* are constructed. Finally, a multiple alignment is calculated using *SeqAn:: TCoffee* (Rausch *et al.*, 2008). *Mugsy* has been designed to rapidly align closely related genomes, such as different strains of a bacterium. Here, it produces alignments of high quality. On more distantly related genomes, however, the program is often outperformed by other multiple aligners (Earl *et al.*, 2014).

Finding *anchor points* is the most important step in whole-genome sequence alignment. Here, a trade-off between *speed*, *sensitivity* and *precision* has to be made. A sufficient number of anchor points is necessary to reduce the run time of the subsequent, more sensitive alignment routine. Wrongly chosen anchor points, on the other hand, can substantially deteriorate the quality of the final output alignment. They may not only lead to misalignments of non-homologous parts of the sequences but may also prevent biologically relevant, true homologies from being aligned. Also, if the number of anchor points is too large, finding optimal chains of anchor points can become computationally expensive.

In this article, we apply the *filtered spaced word matches (FSWM)* approach (Leimeister *et al.*, 2017) to find pairwise anchor points for genomic alignment. We use a *hit-and-extend* approach where high-scoring spaced-word matches are used as *seeds*. More precisely, for a given binary pattern of length ℓ representing *match* and *don't care* positions, we identify *spaced-word matches*—i.e. pairs of length-ℓ segments from the input sequences with matching nucleotides at the *match* positions and possible mismatches at the *don't care* positions. For each such spaced-word match, we then calculate a similarity score, and we keep only those spaced-word matches that have a score above a certain threshold. These matches are then extended to gap-free alignments, similar as in *BLAST* (Altschul *et al.*, 1990). To evaluate the anchor points generated by our approach, we modified the *Mugsy* pipeline by using our anchoring procedure instead of the original anchor points in *Mugsy* that are based on exact word matches. For closely related input sequences, these two different anchoring procedures lead to alignments of similar quality. Our anchor points are clearly superior, however, if distal sequences are to be aligned, where most other alignment approaches either fail to produce meaningful alignments or require an unacceptable amount of time.

Through our website at http://spacedanchor.gobics.de, we provide the modified *Mugsy* pipeline with our anchoring approach, as a pipeline for genome-sequence alignment that can be readily installed. In addition, we provide a stand-alone version of our software, such that software developers can integrate our anchor points into their own sequence-analysis pipelines.

2 Results

2.1 Filtered spaced word matches

For a sequence S of length L over an alphabet Σ and $0 < i \leq L$, $S[i]$ denotes the ith symbol of S, and $|S|$ denotes the *length* of S. Throughout this article, a *pattern* is a word over {0, 1}. For a pattern P, a position i is called a *match position* if $P[i] = 1$ and a *don't-care positions* otherwise. The number of match positions in a pattern P is called the *weight* of P. For an alphabet Σ, a pattern P, and a wildcard character '*' not contained in Σ, a *spaced word* with respect to P is a word w over $\Sigma \cup \{*\}$, such that $w[k] = *$ if and only if k is a *don't-care position*, see also Leimeister *et al.* (2014) and Horwege *et al.* (2014). We say that a spaced word w with respect to a pattern P occurs in a sequence S at some position i, if $i \leq |S| - |P| + 1$, and if $S[i + k - 1] = w[k]$ for all match positions k of P.

For sequences S_1 and S_2, a pattern P, and positions i and j, we say that there is *spaced-word match* between S_1 and S_2 at (i, j) with respect to P if the same spaced word occurs at i in S_1 and at j in S_2—in other words, if for all match positions k in P, one has

$$S_1[i + k - 1] = S_2[j + k - 1].$$

For the two sequences S_1 and S_2 below, for example, there is a spaced-word match with respect to the pattern $P = 1100101$ at (5, 2):

S_1:	G	C	T	G	T	A	T	A	C	G	T	C		
S_2:				A	T	A	C	A	C	T	T	A	T	
P:					1	1	0	0	1	0	1			

as the same spaced word '$TA**C*T$' occurs at positions 5 in S_1 and at position 2 in S_2.

In a previous article, we used spaced-word matches to estimate phylogenetic distances between genomic sequences, by considering at the nucleotides aligned to each other at the *don't care* positions of selected spaced-word matches (Leimeister *et al.*, 2017). To remove spurious random spaced-word matches, we applied a simple *filtering procedure*. Based on the following substitution matrix (Chiaromonte *et al.*, 2002)

	A	C	G	T
A	91	−114	−31	−123
C		100	−125	−31
G			100	−114
T				91

we calculated for each spaced-word match the sum of substitution scores of the nucleotide pairs aligned at the *don't-care* positions, and we removed all spaced-word matches with a score below zero; compare also Brejova *et al.* (2005).

A graphical representation of the spaced-word matches between two sequences shows that this procedure can clearly separate random spaced-word matches from true homologies. If we plot for each possible score value s the number of spaced-word matches with score equal to s, we obtain a bimodal distribution with one peak for random matches and a second peak for true homologies. We call such a plot a *spaced-words histogram*, see Figure 1 for an example. For simulated sequence pairs under a simple model of evolution, and with a sufficient number of *don't-care* positions in the underlying pattern, both peaks are approximately normally distributed. For real-world sequences, the random peak is still normally distributed, but the 'homologous' peak is more complex. Even so, using a suitable cut-off value, one can easily distinguish between random matches and true homologies; for the above matrix, a cut-off of zero works well. More examples for *spaced-words histograms* are given in Leimeister *et al.* (2017).

Herein, we propose to use spaced-word matches to calculate *anchor points* for pairwise alignment of genomic sequences. To distinguish between spaced-word matches representing *true homologies* and random *background* matches, we use the above filtering criterion. More precisely, our approach to find anchor points for genomic alignment is as follows. For given parameters ℓ and w, we first calculate a pattern P with length ℓ and weight w—i.e. with w *match positions*—using our recently developed software *rasbhari* (Hahn *et al.*, 2016). We then identify all spaced-word matches with respect to P. Based on the above substitution matrix, we calculate the *score* of each spaced-word match, and we discard all spaced-word matches with a score below zero, as we did in our previous article (Leimeister *et al.*, 2017). By default, our program uses only *unique* spaced-word matches. That is, if a spaced word w occurs n times in one sequence and m times in a second sequence, we only use the best-scoring of the $n \times m$ resulting spaced-word matches. But as an alternative, it is also possible to use *all* spaced-word matches with a score above zero.

To find homologies even for distantly related sequences, we use patterns with a low weight; by default, we use a weight of $w = 10$. On the other hand, we use a large number of *don't-care* positions, since this makes it easier to distinguish true homologies from random spaced-word matches. By default, we use a pattern length of $\ell = 110$, so our patterns contain 10 match positions and 100 don't-care positions.

Next, we do gap-free extensions of the identified local similarities in both directions using a standard X-drop approach. As starting points for these extensions, we do not use the full spaced-word matches, but their midpoints. The reason for this is that, with our long patterns, even high-scoring spaced-word matches may not represent true homologies over their entire length. It often happens that parts of a spaced-word aligns homologous nucleotides, but one or both ends of the aligned segments extend into non-homologous regions. There is a high probability, however, that the midpoint of a long, high-scoring spaced-word match is located within a region of true homology. As a result, it is possible that an 'extended' match in our approach is shorter than the initial spaced-word match that was used to define the starting point for the X-drop extension. Also, it can happen that a spaced-word match is located within the 'extension' of a previously processed match. Such matches are redundant and are therefore discarded by our algorithm. Finally, we use the extended gap-free alignments as anchor points for alignment.

2.2 Evaluation

To evaluate *FSWM* and to compare it to a state-of-the-art approach to alignment anchoring, we used the *Mugsy* software system. Here, we used the default version of *FSWM* with *unique* matches, i.e. for each distinct spaced word, only the highest-scoring spaced-word match is used. As mentioned above, the original *Mugsy* uses *MUMmer* to find pairwise anchor points. We replaced *MUMmer* in

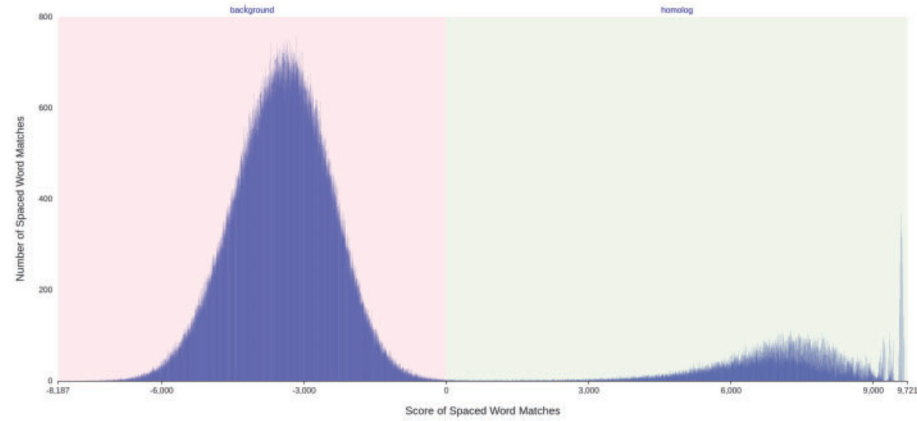

Fig. 1. *Spaced-words histogram* for a comparison of two bacterial genomes, *Phaeobacter gallaeciensis* 2.10 and *Rhodobacterales bacterium Y4I*. All possible spaced-word matches with respect to a given binary pattern *P* are identified, and their scores are calculated as explained in the main text. The number of spaced-word matches with a score *s* is plotted against *s*. Two peaks are visible, an approximately normally distributed peak for background spaced-word matches, and a more complex peak for spaced-word matches representing homologies. With a cut-off value of zero, background and homologous spaced-word matches can be reliably separated

the *Mugsy* pipeline by our *FSWM*-based anchor points and evaluated the resulting multiple alignments. In addition, we compared these alignments to alignments produced by the multiple genome aligner *Cactus* (Paten *et al.*, 2011). *Cactus* is known to be one of the best existing tools for multiple genome alignment; it performed excellently in the *Alignathon* study (Earl *et al.*, 2014). To measure the performance of the compared methods, we used simulated genomic sequences as well as three sets of real genomes. To make *MUMmer* directly comparable to *FSWM*, we used a minimum length of 10 *nt* for maximum unique matches, corresponding to the default *weight* (sum of *match positions*) used in *Spaced Words*. Note that, by default, *MUMmer* uses a minimum length of 15 *nt*. With this default value, however, we obtained alignments of much lower quality. *Cactus* was run with default values.

2.2.1 Simulated genomic sequences

To simulate genomic sequences, we used the *artificial life framework (ALF)* developed by Dalquen *et al.* (2012). *ALF* generates artificial gene families along a randomly generated tree, according to a probabilistic model of evolution. During this process, evolutionary events are logged so the *true* MSA is known for each simulated gene family and can be used as reference to assess the quality of automatically generated alignments.

We generated a series of 14 datasets, each one based on a randomly generated tree with 30 leaves, representing different species. Each dataset consists of 750 simulated gene families, evolved along the respective tree, such that exactly one gene from each family is present in each of the 30 'species'. Within each dataset, we used a fixed mutation rate for all gene families, but we used different mutation rates for different datasets. For all other parameters in *ALF*, we used the default settings. We varied the mutation rates between an *average* of 0.1013 substitutions per position for the first dataset to an average of 0.8349 substitutions per position for the 14th dataset. Here, the average is taken over all pairs of 'species' within the respective dataset. The *maximal* pairwise distance between all pairs of sequences within a dataset ranges from 0.1640 for the first to 1.0923 for the 14th dataset. The simulated genes have an average length of about 1500 *bp*, summing up to a total size of about 32 *MB* per dataset.

For simplicity, we did not concatenate the 750 genes in one 'species'. Instead, we applied the alignment programs that we evaluated to compare all genes from one 'species' to all genes from all other 'species' within the same dataset. Concatenating the sequences would have led to the same results. To assess the quality of the produced alignments, we calculated *recall* and *precision* values in the usual way. If, for one given dataset, S is the set of all positions of the 30×750 simulated gene sequences, we denote by $A \subset \binom{S}{2}$ the set of all pairs of positions aligned to each other by the alignment that is to be evaluated, while $R \subset \binom{S}{2}$ denotes the set of all pairs of positions aligned to each other in the reference alignment. *Recall* and *precision* are then defined as

$$\text{Recall} = \frac{|A \cap R|}{|R|}, \quad \text{Precision} = \frac{|A \cap R|}{|A|} \tag{1}$$

The harmonic mean of *recall* and *precision* is called the *balanced F-score* and is often used as an overall measure of accuracy; it is thus defined as

$$F_{\text{score}} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

To estimate these three values, we used the tool *mafComparator* which was also used in the *Alignathon* study (Earl *et al.*, 2014). Since it is prohibitive to consider *all* pairs of positions of the test sequences, we sampled 10 million pairs of positions for each dataset. This corresponds to the evaluation procedure used in *Alignathon*.

For the simulated sequence sets, their *recall* and *precision* values are shown in Figures 2 and 3. For datasets with smaller mutation rates, the quality of alignments obtained with *FSWM* and *MUMmer* is comparable (Fig. 4). However, if the mutation rate increases, our spaced-words approach clearly outperforms the original version of *Mugsy* where exact word matches are used to find anchor points. With *FSWM*, not only more homologies are detected, compared to *Mummer*, but also the *precision* of *Mugsy* is slightly improved.

2.2.2 Real-world genome sequences

For real-world genome families, it is usually not possible to calculate the *precision* of MSA programs because it is, in general, not known which sequence positions exactly are homologous to each other and which ones are not. If there are *core blocks* of the sequences for which biologically correct alignments are known, at least *recall* values can be calculated for these core blocks. For most genome

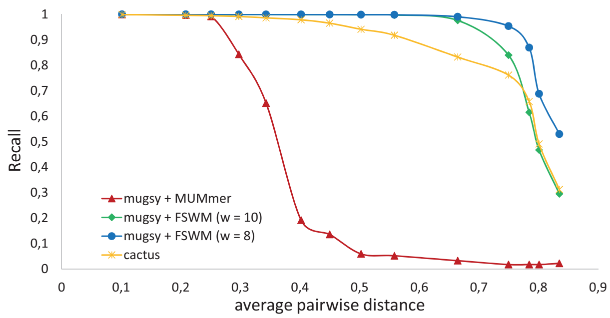

Fig. 2. *Recall* values for *Mugsy* using anchor points generated with *FSWM* and with *MUMmer*, respectively, as well as for *Cactus*. Test data were simulated genomic sequences generated with *ALF*, see main text for details. *FSWM* was run with the default *weight* $w = 10$, i.e. with 10 *match positions* in the underlying pattern, and with $w = 8$

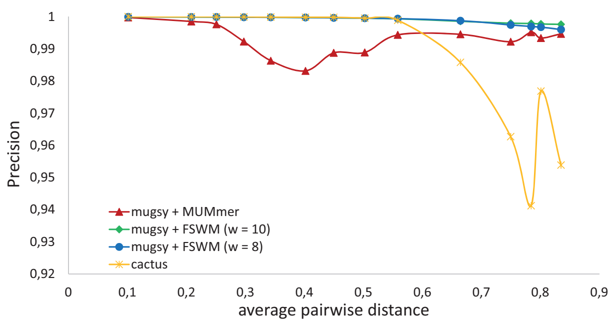

Fig. 3. *Precision* values for *Mugsy* with *FSWM* and *MUMmer* anchor points respectively, and for *Cactus*. Test data and parameter values as in Figure 2

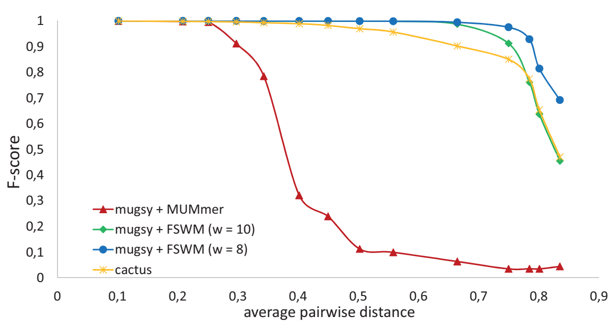

Fig. 4. *F-Score* values for *Mugsy* with *FSWM* and *MUMmer* anchor points, respectively, and for *Cactus*. Test data and parameter values as in Figure 2

sequences, however, not even such core blocks are available. To evaluate *Mugsy*, the authors of the program therefore used the *number of core columns* of the produced alignments as a criterion for alignment quality (Angiuoli and Salzberg, 2011). Here, a *core column* is defined as a column that does not contain gaps, i.e. a column in which nucleotides from *all* of the input sequences are aligned. In addition, the authors of *Mugsy* used the *number of pairs of aligned positions* of the aligned sequences as an indicator of alignment quality. In this article, we use the same criteria to evaluate multiple alignments of real-world genomes.

As a first real-word example, we used a set of 29 *E. coli/Shigella* genomes that has been used in the original *Mugsy* paper, see Supplementary Material for details; these sequences have also been used to evaluate alignment-free methods (Haubold *et al.*, 2015; Morgenstern *et al.*, 2015; Yi and Jin, 2013). The total size of this dataset is about 141 *MB*. As a second test set, we used another prokaryotic dataset, namely a set of 32 complete *Roseobacter* genomes (details in the Supplementary Material); these genomes are more distantly related than the *E. coli/Shigella* strains. The total size of this dataset is about 135 *MB*. To test our approach on eukaryotic genomes, we used as a third test case a set of nine fungal genomes, namely *Coprinopsis cinerea*, *Neurospora crassa*, *Aspergillus terreus*, *Aspergillus nidulans*, *Histoplasma capsulatum*, *Paracoccidioides brasiliensis*, *Saccharomyces cerevisiae*, *Schizosaccharomyces pombe* and *Ustilago maydis* (genbank accession numbers are given in the Supplementary Material). The total size of this third dataset is about 253 *MB*.

Table 1. Evaluation of multiple alignments of 29 *E. coli/Shigella* genomes, 32 *Roseobacter* genomes and 9 fungal genomes, obtained with *Mugsy*, using anchor points calculated with *FSWM* and with *MUMmer*, respectively

	Core LCBs	Aligned pairs	Core col.	LCBs
29 *E. coli/Shigella* genomes				
Mugsy + MUMmer	539	1,61E+09	2,827,115	4138
Mugsy + FSWM	664	1,63E+09	2,867,432	5906
Cactus	20,163	1,48E+09	2,663,750	56,592
32 *Roseobacter* genomes				
Mugsy + MUMmer	39	3,63E+08	13,654	13,501
Mugsy + FSWM	859	7,15E+08	824,054	30,836
Cactus	5984	4,95E+08	280,085	337,320
9 fungal genomes				
Mugsy + MUMmer	9	5,88E+06	2097	4252
Mugsy + FSWM	2590	1,18E+08	718,176	89,555
Cactus	31,589	1,33E+08	828,680	848,242

Note: As a comparison, the table contains the results obtained with *Cactus*. The first column contains the number of *core columns*, i.e. the number of columns in the multiple alignments that do not contain gaps; the second column contains the total number of aligned pairs of positions in the alignment. The third column contains the number of *core Locally Collinear Blocks (LCBs)* i.e. the number of *LCBs* that involve *all* of the aligned genomes ('core LCBs'), while the last column contains the total number of *LCBs*.

The results of *Mugsy* with *MUMmer* and *FSWM*, respectively, for the three real-world datasets are shown in Table 1, together with the results obtained with *Cactus*. In addition to the number of *core columns* and the number of aligned pairs of positions, the table contains the number of *core Locally Collinear Blocks*, i.e. the number of *Locally Collinear Blocks* involving all of the input sequences, and the total number of *Locally Collinear Blocks* returned by the alignment programs. For the *E. coli/Shigella* sequences, the two anchoring methods, *MUMmer* and *FSWM*, led to alignments of comparable quality when used with *Mugsy*; the genome sequences in this dataset are very similar to each other. For the *Roseobacter* and fungal genomes, however, the *FSWM* anchor points led to much better alignments than the default anchor points generated with *MUMmer*. The sequences in these sets are far more apart from each other than the sequences in the *E. coli/Shigella* set, so the results on these three datasets confirm our above results on simulated sequences.

2.2.3 Program run time

Table 2 reports the program run times of *Mugsy* with *FSWM*, *Mugsy* with *MUMmer* and *Cactus* on the above three real-world sequence sets. In addition, the table contains the run times for *FSWM* and *MUMmer* alone. A program run of *Mugsy* with *FSWM* on a set

Table 2. Run time in minutes for three different multiple genome-alignment methods applied to the three test datasets that we used in our program evaluation

	E. coli/Shigella	*Roseobacter*	*fungal genomes*
FSWM	59	83	110
FSWM + Mugsy	638	6428	1488
MUMmer	73	63	43
MUMmer + Mugsy	286	1099	63
Cactus	714	1775	775

of five mammalian sequences of length 200 *mb* each from Earl *et al.* (2014) took around 7 days, and 5 h with $k = 10$ and two days with $k = 12$.

3 Discussion

In this article, we proposed a novel approach to calculate anchor points for genome alignment. Finding suitable anchor points is a critical step in all methods for genome alignment, since the selected anchor points determine which regions of the sequences can be aligned to each other in the final alignment. A sufficient number of anchor points is necessary to keep the search space and run time of the main alignment procedure manageable, so *sensitive* methods are needed to find anchor points. Wrongly selected anchor points, on the other hand, can seriously deteriorate the quality of the final alignments, so anchoring procedures must also be highly *specific*.

Earlier approaches to genomic alignment used exact word matches as anchor points (Delcher *et al.*, 1999; Höhl *et al.*, 2002), since such matches can be easily found using suffix trees and related indexing structures. These approaches are limited, however, to situations where closely related genomes are to be aligned, for example different strains of a bacterium. In modern approaches to database searching, *spaced seeds* are used to find potential sequence homologies (Buchfink *et al.*, 2015; Hauswedell *et al.*, 2014; Li *et al.*, 2003). Here, binary patterns of *match* and *don't care* positions are used, and two sequence segments of the corresponding length are considered to match if identical residues are aligned at the *match* positions, while mismatches are allowed at the *don't care* positions. Such pattern-based approaches are more *sensitive* than previous methods that relied on exact word matches.

We previously proposed to apply the 'spaced-seeds' idea to alignment-free sequence comparison, by replacing contiguous words by so-called *spaced words*, i.e. by words that contain wildcard characters at certain pre-defined positions (Leimeister *et al.*, 2014). More recently, we introduced *FSWM* (Leimeister *et al.*, 2017) to estimate the average number of substitutions per sequence position between two genomes. In the latter approach, we first identify spaced-word matches using relatively long patterns with only few *match* positions. For the identified matching segments, we look at the nucleotides that are aligned to each other at the *don't-care* positions, and we discard spaced-word matches for which the similarity at the *don't-care* positions is below a threshold. Substitution frequencies are then estimated based on the aligned nucleotides at the don't-care positions of the remaining spaced-word matches. We showed that this procedure is fast and highly sensitive, and it can reliably distinguish between true homologies and spurious sequence similarities.

In the present study, we used FSWM to calculate anchor points for genomic sequence alignment. Instead of using the selected spaced-word matches directly as anchor points, we extend the identified hits into both directions, similar to the *hit-and-extend* approach to database searching. In view of speed and accuracy, this approach is somewhere between exact word matching and gapped local alignment. As in our previous paper on filtered spaced words (Leimeister *et al.*, 2017), we use binary patterns with a large number of *don't-care* positions. This way, the 'homologous' and 'background' peaks in the *spaced-word histograms* (Fig. 1) are far enough apart, since the distance between them is proportional to the number of *don't-care* positions in the underlying patterns. With a large number of *don't-care positions*, it is therefore easier to distinguish between homologous and background spaced-word matches.

One might think that, with our long patterns, we might miss too many shorter local homologies. We do not see this as a problem, though. Our goal is not to find *all* local homologies between two sequences, but to output a sufficient number of anchor points to make the final alignment procedure feasible. Moreover, our algorithm is well able to find gap-free homologies that are shorter than the specified pattern length, as long as the sequence similarity between these homologies is strong enough. As explained above, we do not start the *X*-drop extension at the end positions of the identified hits, but in the middle; this way we can find spaced-word matches that cover short homologies, but reach into gapped or non-homologous sequence regions to the left and to the right. In such cases, it can happen that the 'extended' hits are *shorter* than the respective initial spaced-word matches.

To evaluate these anchor points, we integrated them into the popular genome-alignment pipeline *Mugsy*. Test runs on simulated genome sequences show that, for closely related sequences, *Mugsy* produces alignments of high quality with both types of anchor points. For more distantly related sequences, however, the *recall* values of the program drop dramatically if anchor points are calculated with *MUMmer* while, with our spaced-word matches, one observes recall values close to 100% for distances up to around 0.7 substitutions per position.

For real-world genomes, it is more difficult to evaluate the performance of genome aligners since there is only limited information available on which positions are homologous to each other and which ones are not. Angiuoli and Salzberg (2011) therefore used the number of aligned pairs of positions as an indicator of alignment quality, together with the size of the 'core alignment', i.e. the number of alignments columns that do not contain gaps. At first glance, these criteria might seem questionable; it would be trivial to maximize these values, simply by aligning sequences without internal gaps, by adding gaps only at the ends of the shorter sequences. However, as shown in Figure 3, all MSA programs in our study have high *precision* values, i.e. positions aligned by these programs are likely to be true homologs. In this situation, the number of aligned position pairs and size of the 'core alignment' can be considered as a proxy for the *recall* of the applied methods i.e. the proportion of homologies that are correctly aligned.

As shown in Table 2, the program run time to generate anchor points is comparable for *FSWM* and *MUMmer*. For distantly related sequence sets, however, the *total* run time of *Mugsy* is much higher with our *FSWM* anchoring approach than with anchor points from *MUMmer*. A possible explanation for the difference in run time is that *FSWM* is more sensitive, so a larger number of anchor points are produced. Table 1 shows that, with our FSWM, more *Locally Collinear Blocks* are found than with the exact word matches that are found with *MUMmer*—especially for distantly related sequences where exact word matching is not very sensitive. One way of reducing the program run time would be to apply a cut-off value to reduce the number *Locally Collinear Blocks* that are to be aligned in the main alignment procedure. Further research efforts are necessary to balance speed and accuracy of multiple genome alignment algorithms.

Acknowledgements

We thank Rasmus Steinkamp for IT support. Three anonymous reviewers read the manuscript very carefully and made many detailed and helpful comments.

Funding

The project was partially funded by the VW Foundation, project VWZN3157. We acknowledge support by the *Open Access Publication Funds* of the *Göttingen University*.

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.

Batzoglou,S. (2005) The many faces of sequence alignment. *Brief. Bioinformatics*, **6**, 6–22.

Blanchette,M. *et al*. (2004) Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res*., **14**, 708–715.

Bradley,R.K. *et al*. (2009) Fast statistical alignment. *PLoS Comput. Biol*., **5**, e1000392.

Bray,N. and Pachter,L. (2003) MAVID multiple alignment server. *Nucleic Acids Res*., **31**, 3525–3526.

Bray,N. *et al*. (2003) AVID: a Global Alignment Program. *Genome Res*., **13**, 97–102.

Brejova,B. *et al*. (2005) Vector seeds: an extension to spaced seeds. *J. Comp. Syst. Sci*., **70**, 364–380.

Břinda,K. *et al*. (2015) Spaced seeds improve *k*-mer-based metagenomic classification. *Bioinformatics*, **31**, 3584–3592.

Brudno,M. *et al*. (2003a) Fast and sensitive multiple alignment of large genomic sequences. *BMC Bioinformatics*, **4**, 66.

Brudno,M. *et al*. (2003b) LAGAN and multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res*., **13**, 721–731.

Buchfink,B. *et al*. (2015) Fast and sensitive protein alignment using DIAMOND. *Nat. Methods*, **12**, 59–60.

Chiaromonte,F. *et al*. (2002) Scoring pairwise genomic sequence alignments. In: Altman, R. B., Dunker, A. K., Hunter, L., and Klein, T. E. (eds.) *Pacific Symposium on Biocomputing*. Lihue, Hawaii, pp. 115–126.

Choi,K.P. *et al*. (2004) Good spaced seeds for homology search. *Bioinformatics*, **20**, 1053.

Dalquen,D.A. *et al*. (2012) ALF: a simulation framework for genome evolution. *Mol. Biol. Evol*., **29**, 1115–1123.

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

Darling,A.E. *et al*. (2006) Procrastination leads to efficient filtration for local multiple alignment. In: Bucher, P. and Moret, B. M. E. (eds.) *Algorithms in Bioinformatics, Lecture Notes in Bioinformatics*. Springer, Berlin, Germany, pp. 126–137.

Darling,A.E. *et al*. (2010) progressiveMauve: multiple genome alignment with gene gain, loss and rearrangement. *PLoS One*, **5**, e11147.

David,M. *et al*. (2011) SHRiMP2: sensitive yet practical short read mapping. *Bioinformatics*, **27**, 1011–1012.

Delcher,A.L. *et al*. (1999) Alignment of whole genomes. *Nucleic Acids Res*., **27**, 2369–2376.

Deneke,C. *et al*. (2017) PaPrBaG: a machine learning approach for the detection of novel pathogens from NGS data. *Sci. Rep*., **7**, 39194.

Dewey,C.N. and Pachter,L. (2006) Evolution at the nucleotide level: the problem of multiple whole-genome alignment. *Hum. Mol. Genet*., **15**, R51–R56.

Döring,A. *et al*. (2008) SeqAn—an efficient, generic C++ library for sequence analysis. *BMC Bioinformatics*, **9**, 11.

Dubchak,I. *et al*. (2009) Multiple whole-genome alignments without a reference organism. *Genome Res*., **19**, 682–689.

Durbin,R. *et al*. (1998) *Biological Sequence Analysis*. Cambridge University Press, Cambridge, UK.

Earl,D. *et al*. (2014) Alignathon: a competitive assessment of whole-genome alignment methods. *Genome Res*., **24**, 2077–2089.

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

Hahn,L. *et al*. (2016) *rasbhari*: optimizing spaced seeds for database searching, read mapping and alignment-free sequence comparison. *PLoS Comput. Biol*., **12**, e1005107.

Haubold,B. *et al*. (2015) andi: fast and accurate estimation of evolutionary distances between closely related genomes. *Bioinformatics*, **31**, 1169–1175.

Hauswedell,H. *et al*. (2014) Lambda: the local aligner for massive biological data. *Bioinformatics*, **30**, i349–i355.

Höhl,M. *et al*. (2002) Efficient multiple genome alignment. *Bioinformatics*, **18**, S312–320S.

Horwege,S. *et al*. (2014) *Spaced words* and *kmacs*: fast alignment-free sequence comparison based on inexact word matches. *Nucleic Acids Res*., **42**, W7–W11.

Huang,W. *et al*. (2006) Accurate anchoring alignment of divergent sequences. *Bioinformatics*, **22**, 29–34.

Kurtz,S. (1999) Reducing the space requirement of suffix trees. *Softw. Pract. Exp*., **29**, 1149–1171.

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

Langmead,B. *et al*. (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*., **10**, R25.

Leimeister,C.-A. *et al*. (2014) Fast Alignment-Free sequence comparison using spaced-word frequencies. *Bioinformatics*, **30**, 1991–1999.

Leimeister,C.-A. *et al*. (2017) Fast and accurate phylogeny reconstruction using filtered spaced-word matches. *Bioinformatics*, **33**, 971–979.

Li,M. *et al*. (2003) PatternHunter II: highly sensitive and fast homology search. *Genome Informatics*, **14**, 164–175.

Li,M. *et al*. (2006) Superiority and complexity of the spaced seeds. In: *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, SODA '06*, pp. 444–453. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Ma,B. *et al*. (2002) PatternHunter: faster and more sensitive homology search. *Bioinformatics*, **18**, 440–445.

Morgenstern,B. (2002) A simple and space-efficient fragment-chaining algorithm for alignment of DNA and protein sequences. *Appl. Math. Lett*., **15**, 11–16.

Morgenstern,B. *et al*. (1996) Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl. Acad. Sci. USA*, **93**, 12098–12103.

Morgenstern,B. *et al*. (2002) Exon discovery by genomic sequence alignment. *Bioinformatics*, **18**, 777–787.

Morgenstern,B. *et al*. (2006) Multiple sequence alignment with user-defined anchor points. *Algorith. Mol. Biol*., **1**, 6.

Morgenstern,B. *et al*. (2015) Estimating evolutionary distances between genomic sequences from spaced-word matches. *Algorith. Mol. Biol*., **10**, 5.

Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol*., **48**, 443–453.

Noé,L. (2017) Best hits of 11110110111: model-free selection and parameter-free sensitivity calculation of spaced seeds. *Algorith. Mole. Biol*., **12**.

Noé,L. *et al*. (2010) Designing efficient spaced seeds for SOLiD read mapping. *Adv. Bioinformatics*, **2010**, 1–12.

Ogurtsov,A.Y. *et al*. (2002) OWEN: aligning long collinear regions of genomes. *Bioinformatics*, **18**, 1703–1704.

Ounit,R. and Lonardi,S. (2015) Higher classification accuracy of short metagenomic reads by discriminative spaced k-mers. In: *Algorithms in Bioinformatics: 15th International Workshop, WABI 2015, Atlanta, GA, USA, September 10–12, 2015, Proceedings*, pp. 286–295. Springer, Berlin, Germany.

Paten,B. *et al*. (2011) Cactus: algorithms for genome multiple sequence alignment. *Genome Res*., **21**, 1512–1528.

Raphael,B. *et al*. (2004) A novel method for multiple alignment of sequences with repeated and shuffled elements. *Genome Res*., **14**, 2336–2346.

Rausch,T. *et al*. (2008) Segment-based multiple sequence alignment. *Bioinformatics*, **24**, i187–i192.

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

The Arabidopsis Genome Initiative (2000) Analysis of the genome sequence of the flowering plant *Arabidopsis thaliana*. *Nature*, **408**, 796–815.

Xu,J. *et al*. (2006) Optimizing multiple spaced seeds for homology search. *J. Comput. Biol*., **13**, 1355–1368.

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

5 Further improvements and experiments

As shown in Chapter 2, *Multi-SpaM* is an effective tool for phylogeny reconstruction. Regardless, we tried to improve the method further. The quartet trees need to be correct with high probability in order to reconstruct an accurate phylogeny. This can be achieved by removing incorrect quartet trees. Alternatively, quartet trees, which are very likely to be correct, could be prioritized by giving them a higher weight.

5.1 SH-like support values

I followed this approach in an experiment where I tried to assess how much confidence can be placed into the topology of an individual quartet tree. To this end, I calculated a support value for every quartet tree. These support values are computed similarly to a test proposed by Shimodaira and Hasegawa [119]. This non-parametric test checks the null hypothesis that all three possible quartet tree topologies are equally likely. To this end, it considers the difference of the likelihood of the best and the second-best quartet tree topology. If this difference is significantly larger than the variance of the likelihood values, the SH test is passed. The variance is simulated in a similar way to bootstrapping. A fixed number of sites in the input alignment are sampled with replacement in order to generate a number of test sets. The fraction of test sets, for which the test was passed, is the SH-like support value of the quartet tree. If the support value is zero, then all three possible quartet tree topologies are equally likely. The implementation of this test is described in more detail in [45]. I calculated the SH-like support values with *RAxML*. This algorithm is activated with the flag '-F j'.

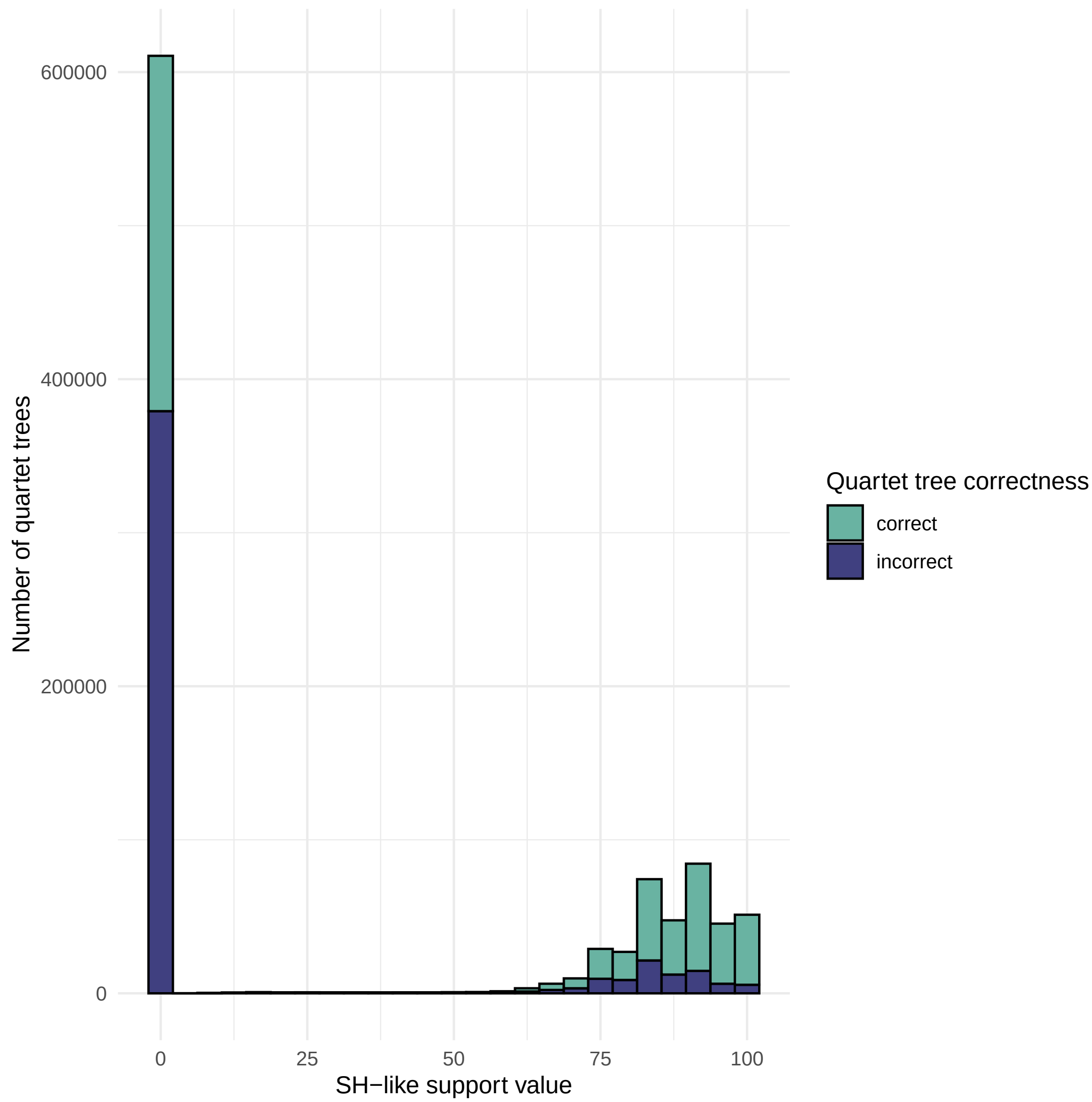

Figure 7: For a dataset of 29 *E.coli/Shigella* genomes, SH-like support values are calculated for every quartet tree. The number of quartet trees is shown for each possible support value. The quartet tree topologies are compared against a reference tree in order to assess their correctness.

5.1.1 Correctness and RF

In order to evaluate the effectiveness of SH-like support values, I compared the quartet tree topologies with a reference tree using the ETE-toolkit [56]. For each set of quartet trees with the same support value, I calculated the fraction of correct quartet trees. The results for a dataset consisting of 29 *E.coli/Shigella* genomes can be found in Table 7. A large number of quartet trees have a support value of zero. These quartet trees were calculated from micro-alignments which, most likely, consist of four identical sequences. In the original version of *Multi-SpaM*, these P-blocks would be discarded before the optimal quartet tree topology would be calculated. It is important to remove these quartet trees as the topology returned by *RAxML* is random. Thus, a large fraction of quartet trees would be incorrect as can be seen in Table 7. In contrast, quartet trees with high support values are also very likely to be correct.

In order to utilize these support values to improve the phylogenetic tree returned by *Multi-SpaM*, I implemented two different strategies. First, I used a threshold for the support values. All quartet trees with support values below that threshold were removed. For the second strategy, I also used this treshold. In addition, I used the support value to give each quartet tree a weight that is used during the supertree calculation with *Quartet MaxCut* [6]. For three datasets, I applied these two strategies to a single run of *Multi-SpaM*. In the following three figures, the results are shown. For both the fish mitochondria dataset (see Figure 10) and the 29 *E.coli/Shigella* genomes (see Figure 8), it was possible to improve the phylogenetic tree of *Multi-SpaM*. However, this could only be achieved with dataset-specific thresholds. It should be noted that with a threshold of zero, many incorrect quartet trees were included which causes the high Robinson-Foulds distances. Furthermore, if the threshold is too high, then the number of quartet trees will be reduced to the point where it affects *Quartet MaxCut* negatively. The difference between the two strategies were mostly negligible. Only for the fish mitochondria, there was a consistent improvement when using the SH-like support values as weights, at least for low threshold values. The relatively low effect of this strategy could be explained by the fact that a quartet tree topology can be output multiple times. This will likely have a bigger effect on the overall weight than the weight of a single quartet tree.

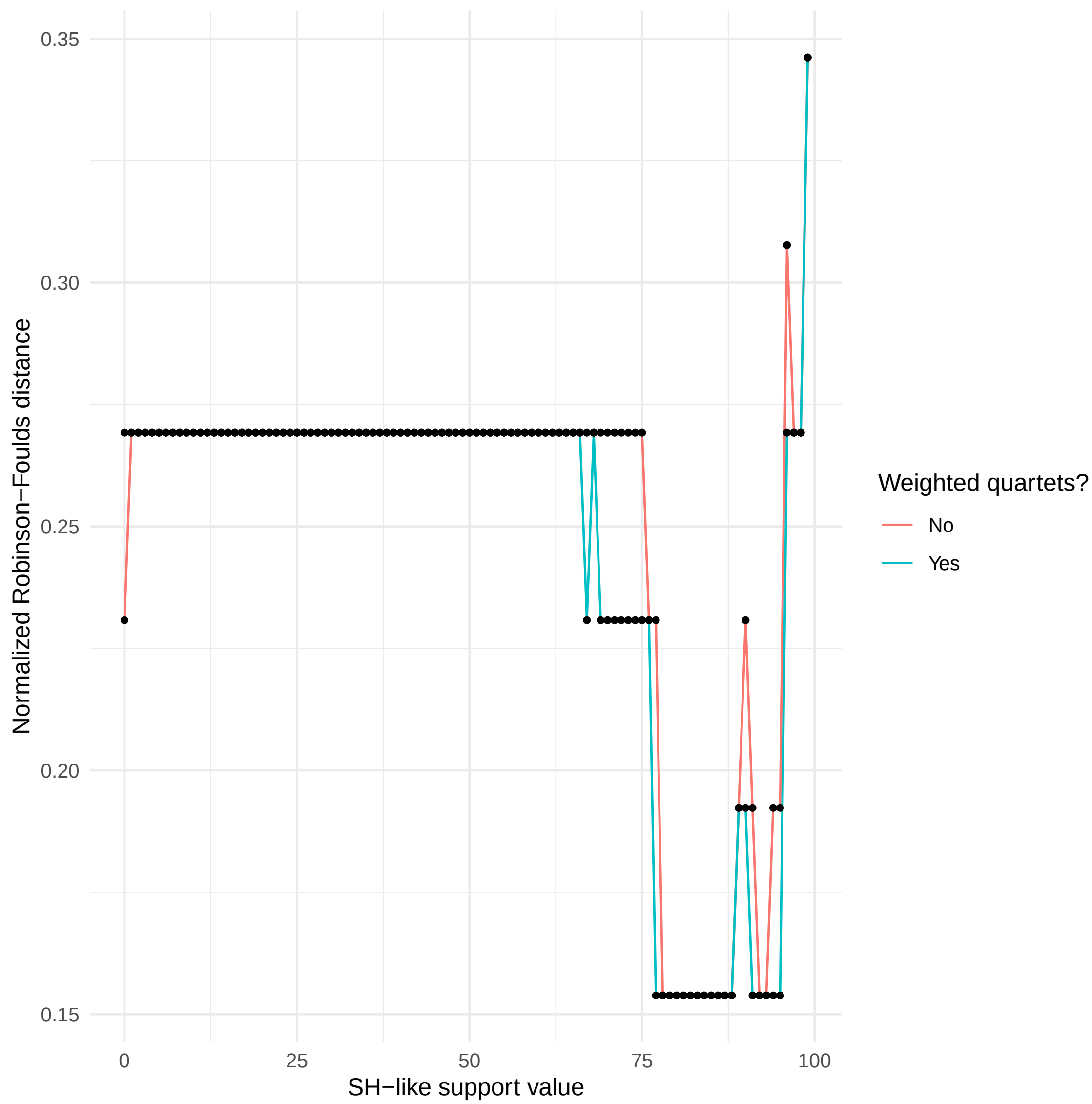

Figure 8: For a dataset of 29 *E.coli/Shigella* genomes, phylogenetic trees were built with the set of quartet trees whose support values surpass the treshold shown on the x axis. With a treshold between 75 and 85, the normalized Robinson-Fould distances of the trees can be improved.

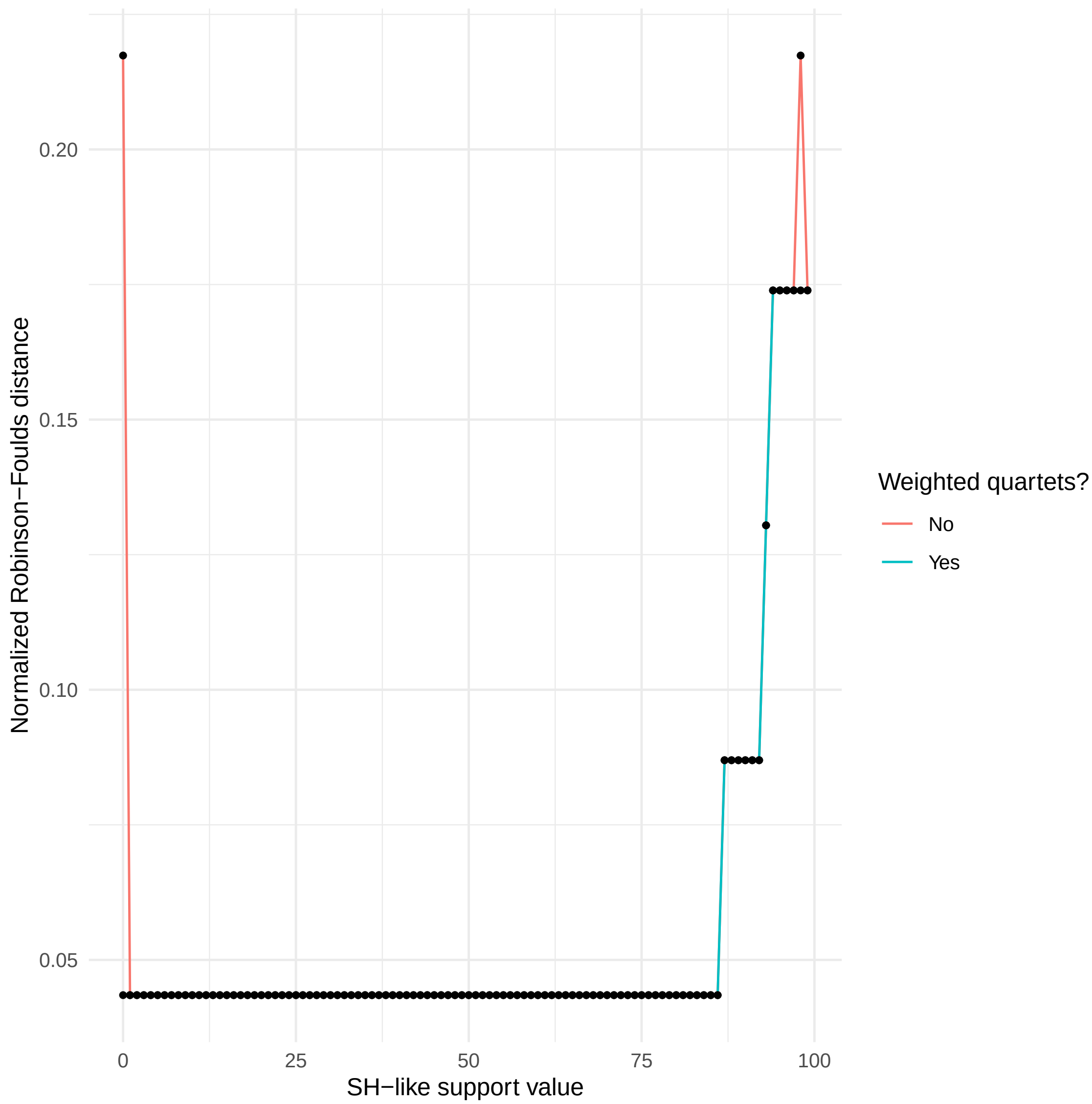

Figure 9: For a dataset of 27 *E.coli/Shigella* genomes, phylogenetic trees were built with the set of quartet trees whose support values surpass the treshold shown on the x axis. With a threshold of zero, many incorrect quartet trees are included which explains the high Robinson-Foulds distance. When the support values are used as weights, these quartet trees are effectively removed. Overall, there is no benefit in using support values for this particular run.

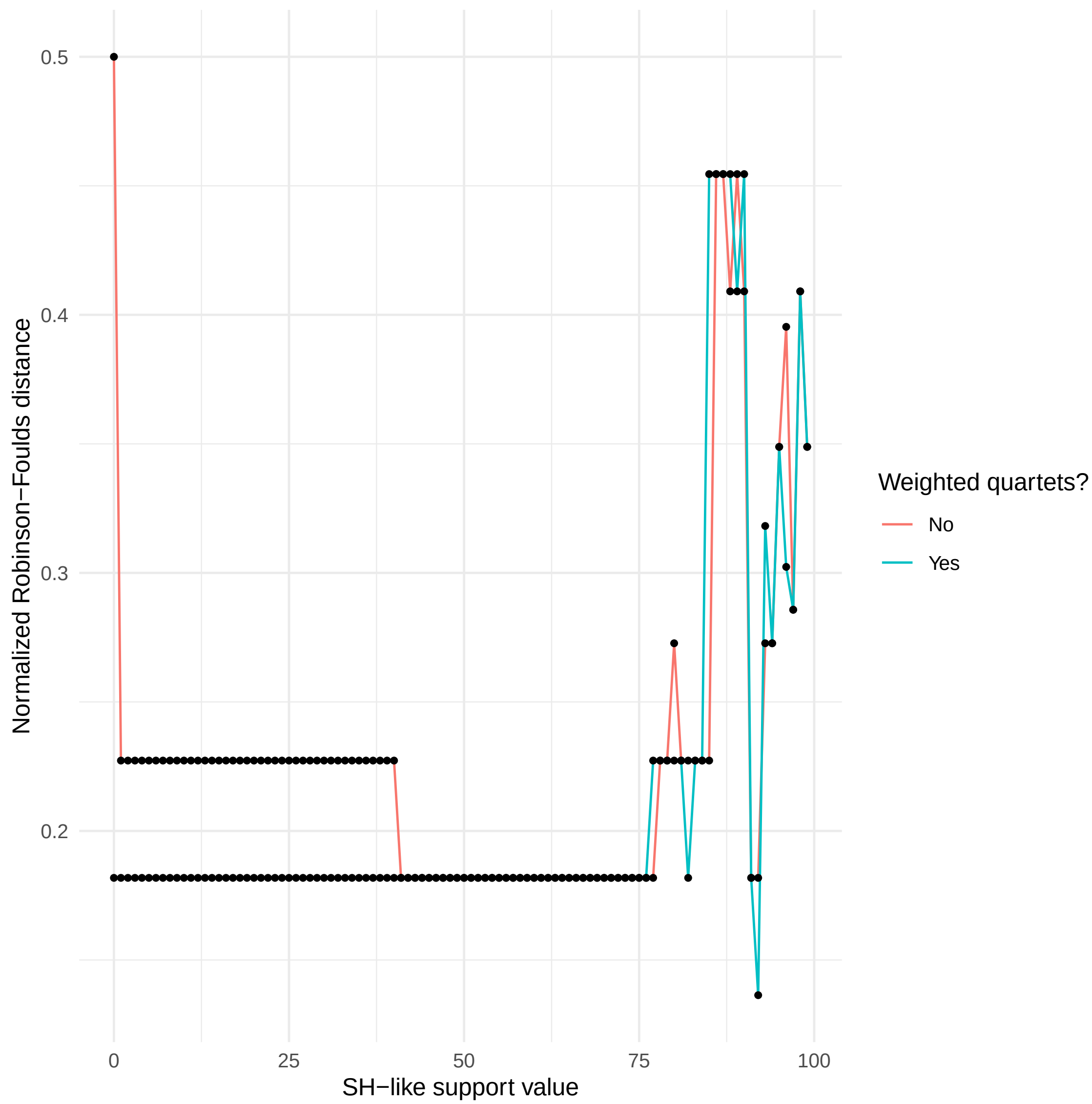

Figure 10: For a dataset of 25 fish mitochondria genomes, phylogenetic trees were built with the set of quartet trees whose support values surpass the treshold shown on the x axis. For this particular run, the Robinson-Foulds distance of the trees can be improved by using the SH-like support values as weight. The same effect can be achieved with a threshold between 40 and 75.

5.2 Neighbor-Joining

Apart from the accuracy, I also tried to improve the runtime of *Multi-SpaM*. The main motivation for developing *Multi-SpaM* was to utilize the high accuracy of *Maximum-Likelihood* in an alignment-free method. In Chapter 2, we showed that this is indeed possible. However, *RAxML* [128] requires a large portion of the program's runtime. Thus, we tried to replace the *Maximum-Likelihood* step of the algorithm with *Neighbor-Joining* [112]. For most datasets, this step requires the largest portion of the runtime. Thus, a very fast method such as *Neighbor-Joining* would drastically improve the overall runtime of *Multi-SpaM*. Since the P-blocks form rather small micro-alignments, it is not clear if the quartet tree topologies differ depending on the method used to calculate them. Preliminary tests have been done by Mats Kastner. Later, I implemented a *Neighbor-Joining step* in *Multi-SpaM* and further improved it.

By default, *Multi-SpaM* does not calculate any distances. However, this is a requirement for *Neighbor-Joining*. For every P-block, the pairwise distances can be computed in a straightforward manner using the *Jukes&Cantor* [62] model. Based on the distance matrix, the quartet tree can be calculated. However, there are some special cases where *Neighbor-Joining* would return incorrect quartet tree topologies which might affect the performance of the method negatively. Obviously, if all distances are zero, the quartet tree can be discarded right away. In other cases, *Neighbor-Joining* might calculate negative branch lengths. This can be the case when two sequences are identical and the other two sequences are more closely related to the other two sequences than to each other. In such a case, the quartet tree cannot reflect the true relationsships and can therefore be removed.

5.2.1 Test results

We applied the modified version of *Multi-SpaM* to the same datasets that were used for the evaluation in Chapter 2. The results for the simulated datasets can be found in Figure 11. For the real-word datasets, the results can be found in Figure 12. With *Neighbor-Joining*, the normalized Robinson-Foulds distance [108] to the reference trees could be improved for all datasets, except for the *Yersinia* datsets for which the results are the same. Interestingly, the opposite results could be observed for the simulated datasets. Here, the phylogeny could be reconstructed more accurately with *RAxML*.

The runtime comparison can be found in Table 7. Clearly, the version with *Neighbor-Joining* is much faster. In fact, calculating the optimal topology for 1 million quartet trees can be done near instantly. Thus, the remaining runtime is split between creating the list of spaced-words, sorting this list and sampling the P-blocks. For the larger datasets, the sampling step is the most time-consuming step as there is a larger number of spaced-word matches with a negative score and, therefore, more scores need to be calculated in total. However, the *Maximum-Likelihood* step requires most of the runtime for smaller datasets. Overall, the modified version of *Multi-SpaM* can be seen as an improvement over the original version.

Datasets	RAxML	Neighbor-Joining
29 *E.coli/Shigella*	611	116
25 fish mitochondria	27	2.7
19 *Wolbachia*	484	70
8 *Yersinia*	183	71

Table 7: Runtime comparison in for the original version of *Multi-SpaM* using *RAxML* and the modified version with *Neighbor-Joining*. The reported times are *wall clock times*.

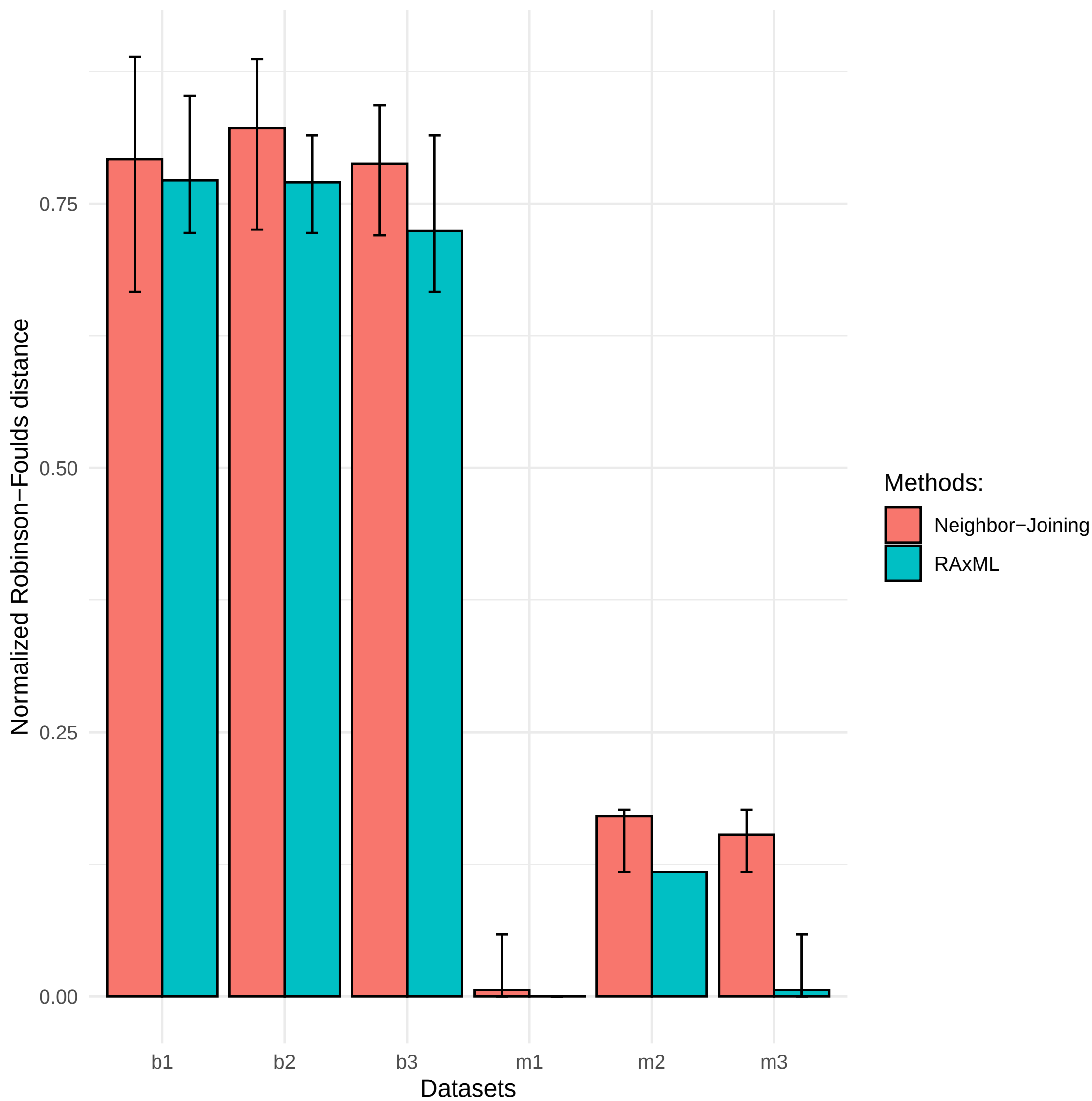

Figure 11: The datasets were simulated using *ALF* [22]. The datasets b1-3 are supposed to resemble bacterial genomes. The other three datasets consists of supposedly mammal-like genes. These datasets were used during the evaluation of *Multi-SpaM*. If *Neighbor-Joining* is used to calculate the optimal quartet tree topologies, the quality of the phylogenetic trees decreased for all datasets.

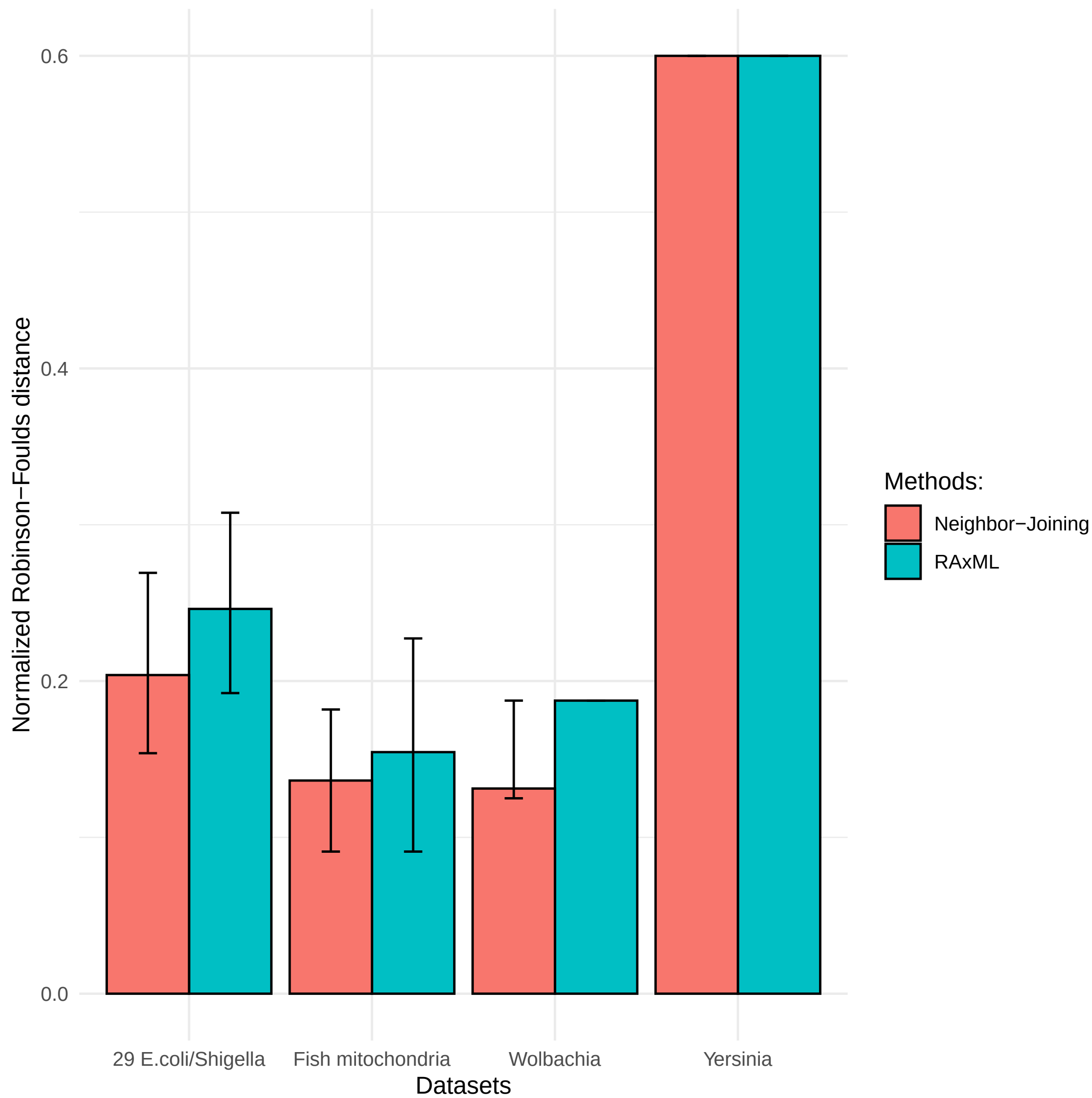

Figure 12: The topologies of the quartet trees were calculated with *Neighbor-Joining*. The phylogenetic trees were compared to the trees from the original *Multi-SpaM* version. For all four real-world datasets, the quality of the trees improved when *Neighbor-Joining* was used.

5.3 Bootstrapping

Bootstrapping is a common strategy to assess the stability of a method for phylogeny reconstruction. One way of doing this is to create 100 datasets by sampling columns from a given multiple sequence alignment and applying the method to each of these datasets. Then, a bootstrap value can be attributed to each branch in the phylogenetic tree. This value shows how many times this particular branch appeared in the 100 phylogenetic trees.

Since the results of *Multi-SpaM* are non-deterministic, it is possible to run the tool multiple times on the same dataset in order to assess the stability of the method. However, this would drastically increase the runtime. Therefore, I chose a different approach. I ran *Multi-SpaM* a single time and used the set of quartet trees to sample 100 sets consisting of 1 million quartet trees each. For each set, a supertree is built with *Quartet MaxCut*. Then, I calculated the consensus tree with the program *consense* from the *PHYLIP* package [36]. This tool also shows the bootstap values for each branch. One phylogenetic tree with support values is shown in Figure 13. For the dataset consisting of 29 *E.coli/Shigella* genomes, there are multiple low support values, the lowest being 52. This explains the relatively high variance which we observed in the evaluation of *Multi-SpaM* (see Chapter 2). However, there are also datasets for which the bootstrap values are high. For the *Wolbachia* dataset, the lowest bootstrap value was 99 using this approach.

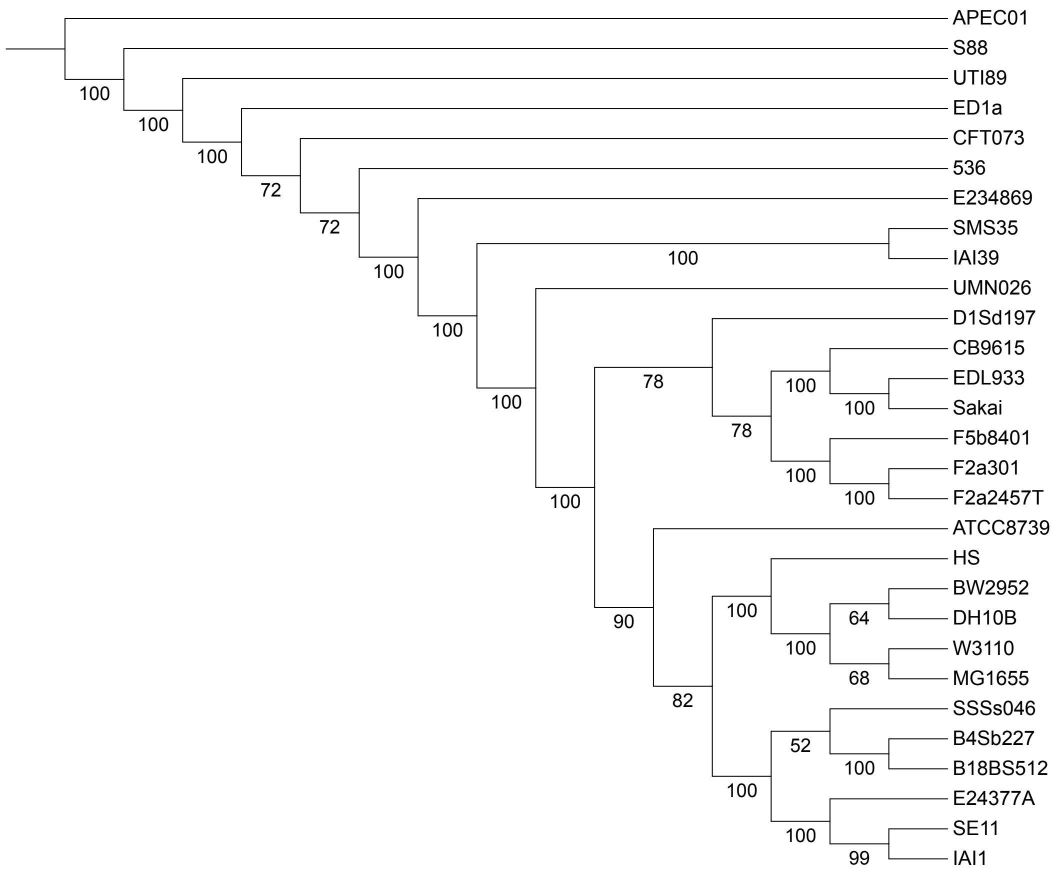

Figure 13: The bootstrap values are shown for a datasets of 29 *E.coli/Shigella* genomes.

6 Discussion

The main focus and contribution of this thesis is a novel alignment-free method for phylogeny reconstruction called *Multi-SpaM* (see Chapter 2). Existing alignment-free methods (see Chapter 1.4) usually calculate pairwise distances. The resulting phylogenetic trees are, in general, not as accurate as trees built by character-based methods that are used in more traditional approaches. Thus, I developed a method that extends the spaced-word approach used by *FSWM* [73] to multiple sequence comparison and uses a character-based method in hopes of improving the quality of the phylogenetic tree while retaining the speed advantage of an alignment-free method. *Multi-SpaM* uses the filtering approach introduced by *FSWM* that can reliably remove random spaced-word matches and thus ensure that the remaining matches are homologous. Four homologous spaced-word occurrences from different sequences constitute a P-block, with respect to a pre-defined binary pattern P. *Multi-SpaM* samples up to a million P-blocks. The P-blocks can be seen as 'micro-alignments' with possible mismatches in some columns. *RAxML* [128] is used to calculate the optimal quartet tree topology for each P-block. Subsequently, the quartet trees are amalgamated into a supertree. We showed that the resulting phylogenies are of high quality.

In Chapter 3, I showed another use case for the P-blocks that were generated by *Multi-SpaM*. We considered pairs of adjacent P-blocks that involve the same set of four sequences. Then, we calculated the distance between the spaced-word occurrences for each sequence. This distance can be encoded as an arbitrary character and subsequently be used to find the most parsimonious phylogenetic tree. Unless the distances are all equal or all different, it is possible to find putative indels with this novel approach. Furthermore, we tried to infer quartet trees from informative P-block distances where possible and observed that these trees are highly accurate for many datasets. In particular, quartet trees which topologies coincided with quartet trees produced by *Multi-SpaM* were very accurate. In several cases, we could improve the phylogenetic trees in comparison to the tree built with *Multi-SpaM*. So far, this has been fundamental research. With a more sophisticated approach, putative indels could be used to improve *Multi-SpaM* even further, for example by giving weights to the quartet trees.

Apart from *Multi-SpaM*, I also contributed to another extension of *FSWM*. In Chapter 4, we showed that spaced-word matches can be used as anchor points [94] for a multiple sequence alignment. Anchor points are high-scoring local alignments of some input sequences which are likely to be part of the final alignment. In this publication, we used *FSWM* to find spaced-word matches. Here, we only used unique spaced-word matches, i.e. only the match with the highest score, which is calculated during the filtering procedure, is used for each spaced-word. These matches were then extended with a standard X-drop approach. We used the extended matches as anchor points to reduce the runtime of slower tools, e.g. for genome alignment. This is possible because only the sequence segments between the anchor points are taken into consideration during the subsequent alignment procedure. *mugsy* [5, 67] is one such genome aligner that uses anchor points. By default, it uses *maximal unique matches* found by *MUMmer* [24]. The maximal unique matches work well as anchor points for closely related species. For distantly related species, there are not enough anchor points that can be found with *MUMmer*. In order to solve this problem, we replaced *MUMmer* with the modified version of *FSWM* in the *mugsy* pipeline. Then, we compared our approach with the default version of *mugsy* and other state-of-the-art methods. We found that the anchor points from *FSWM* led to much better alignments for distantly related species in comparison to the default version of *mugsy*. Overall, the performance was competetive with other genome aligners. Furthermore, our approach led to the highest number of aligned pairs due to the high number of anchor points. As a consequence, the runtime was also the highest for most datasets.

Lastly, I showed a few modifications of *Multi-SpaM* that could improve its performance. In Chapter 5.1, I calculated SH-like support values for every quartet tree. I tried to use them as weights for the quartet trees. Furthermore, I removed quartet trees with low support values. In principle, this approach can improve the performance of *Multi-SpaM*. However, the results were dataset-dependent and required a longer runtime than the original version. In another experiment, I found the optimal quartet tree topologies with *Neighbor-Joining* [112] instead of *RAxML* [128]. This way, the runtime of *Multi-SpaM* could be improved significantly. Furthermore, the quality of the phylogenetic trees also improved for most datasets. Thus, it can be said that character-based methods do not lead to increased accuracy as we assumed in the beginning of this project. However, it should be noted that the *Maximum-Likelihood* approach we used in our method was also heavily limited by the small size of the micro-alignments. Lastly, I implemented bootstrapping for *Multi-SpaM* which can help the user to grasp the stability of the phylogenetic tree.

6.1 Alignment-free methods based on micro-alignments

Many alignment-free methods for phylogeny reconstruction have been proposed in recent years in order to deal with the large amounts of sequence data which are available today (see Chapter 1.4). Earlier methods relied on simple sequence features such as k-mer frequencies. Some recent alignment-free methods estimate pairwise distances from the observed mismatches within so-called micro-alignments. In this section, I want to discuss advantages and disadvantages of these methods in comparison to *Multi-SpaM*.

The micro-alignments consist of (inexact) word matches. They can only be used in a meaningful way if the aligned sequences are homologous. Thus, all these methods have to make sure that there are as few random matches as possible. *andi* and *co-phylog* require the word matches to be unique and sufficiently long so that random word matches are highly unlikely. However, it becomes increasingly challenging to find long micro-alignments for more distantly related sequences. This can drastically reduce the accuracy of the distances. *FSWM* introduced a filtering procedure that can be used to find homologous spaced-word matches while allowing for lower pattern weights, i.e. the number of match positions. Therefore, it is better suited to estimate accurate distances which are higher than 0.5 substitutions per position. Furthermore, the spaced-word matches are not required to be unique. Thus, there can be many possible spaced-word matches for a given spaced-word. This issue is resolved with a greedy *1-to-1 mapping* based on the score used in the filtering procedure. Each spaced-word occurrence can only be matched once. *Multi-SpaM* distinguishes homologous spaced-word matches from background noise with the same approach which is generally beneficial to the performance. However, the additional score calculation results in increased runtimes in comparison to *andi* and *co-phylog*.

In contrast to *FSWM*, *Multi-SpaM* does not exhaustively calculate scores between all spaced-word occurrences. Instead, an initial spaced-word occurrence is chosen randomly. Then, scores are calculated with other occurrences of the same spaced-word until the P-block is complete. Under the assumption that homologous matches have the highest score, it is possible that suboptimal matches are chosen for a P-block. As shown in Chapter 2, the phylogenetic trees of both methods are of similar quality. Thus, it is not clear whether it is worthwhile to calculate scores for all possible spaced-word matches. For n occurrences of a given spaced-word, the required runtimes is $\mathcal{O}(n^2)$ for *FSWM*. For large datasets, this can lead to fairly high runtimes. In contrast, Multi-SpaM only requires $n-1$ score calculations in the worst case in the same scenario. The effect can be seen for the 4.8 *gb* dataset of 14 plant genomes (see Chapter 2). The runtime of *FSWM* is particularly high for this dataset. *FSWM* is 88 times slower than *Multi-SpaM* whereas it is mostly faster for smaller datasets. Partially, the lower runtime of *Multi-SpaM* is also due to the sampling of P-blocks. These two factors make datasets consisting of large sequences very favorable for *Multi-SpaM* in terms of runtime. For small sequences, the number of spaced-word occurrences is relatively low and hence much less score calculations are necessary.

Despite striving for fast runtimes, few alignment-free methods make use of sampling strategies. After some point, the results will not be affected much – if at all – by considering more data. This fact is used by *Mash* [101] which calculates pairwise distances only from a small fraction of all k-mers which is selected by a hash function. This approach has the additional advantage that only a small amount of k-mers have to be kept in memory. A similar sampling strategy has been applied to *FSWM* recently [30]. However, using a small fraction of spaced-words reduces the chance that P-blocks can be found by *Multi-SpaM*. For many small datasets, the number of sampled P-blocks is already much lower than the target of 1 million. A low number of quartet trees can negatively impact the performance of *Quartet MaxCut*. That is why we chose a different sampling strategy for *Multi-SpaM*. However, if the genomes are large enough, it would be possible to use both sampling strategies at the same time.

Sampling is a straight-forward approach to reduce the high memory usage which is a concern for all alignment-free methods that use micro-alignments. For large datasets, the memory requirements can easily exceed the main memory available in most computers. The vast majority of this space is occupied by a list of words or a suffix array. Additionally, these methods take the reverse complement into consideration which further increases the memory requirements. For example, the peak memory usage of these methods ranged between 76 *gb* and 146 *gb* in our evaluation of *Multi-SpaM* (see Chapter 2). *Multi-SpaM* is on the high end in terms of memory usage because it stores a list of spaced-words from all sequences as well as their reverse complements at the same time. In contrast, *FSWM* only needs to store the reverse complement of one sequence for each pairwise sequence comparison. All methods use efficient data structures to keep the memory requirements relatively low. A slight improvement could be achieved by using *canocial k-mers* (or spaced-words) which are, for example, implemented in *Mash*. Thereby, the reverse complement is not stored extra. Instead, every word is compared with its reverse complement and only one of them is stored, depending on which is lexicographically smaller (or larger). However, the list of spaced-words might still not fit into the main memory. In order to achieve a large reduction in memory usage, we implemented a memory saving mode. If activated, the list of words is divided into 16 chunks, depending on the two initial characters of every spaced-word. At every point in time, only one chunk is kept in memory. Thus, *Multi-SpaM* can handle far larger sequences than competing methods on systems with limited main memory. However, it should be noted that this approach increases the runtime of the method. Of course, the number of chunks could be increased to reduce the memory requirements even further.

Dataset	Rank	Number of ranks
29 *E.coli/Shigella*	6	20
27 *E.coli/Shigella*	3	16
14 plants	1	11
25 fish mitochondria	4	18
8 *Yersinia*	4	6

Table 8: This table shows the results of the *AF-project*. For 5 benchmark datasets, *Multi-SpaM* ranks relatively high among all alignment-free methods that participated in this study. The ranks are based on the Robinson-Foulds distance. Multiple tools can have the same distance. Thus, the number of ranks is lower than the total number of tools (between 70 and 90). More details on the performance of *Multi-SpaM* on these datasets can be found in Chapter 2.

As I pointed out in the introduction, it is not an easy task to evaluate methods for phylogeny reconstruction. If the test datasets are simulated, then they are usually based on simplified models of evolution. Thus, they can hardly match the complexity of real-world datasets. These datasets, on the other hand, are much harder to evaluate as there is no known ground truth. The reference trees, that the developers have to rely on, may not reflect the true evolutionary relationships. They are often built with more traditional methods of phylogeny reconstruction, such as *Maximum-Likelihood* on an alignment of selected marker genes. In order to make our evaluation more meaningful, we worked together with researchers of the *Wolbachia* bacteria which we used as one of the datasets in our evaluations. Additionally, we contributed to the *AF-project* [155] which provides trusted benchmarking datasets for alignment-free methods. In Table 8, the results of this benchmark are shown which also includes other alignment-free methods that are not based on micro-alignments.

6.2 Limitations

So far, I have shown that *Multi-SpaM* builts phylogenetic trees of high quality and is particularly well-suited for large sequences. However, there are also some limitations that I want to mention. Some of these drawbacks also apply to similar methods.

Several of the alignment-free methods based on micro-alignments use spaced-words. Consequently, they depend on the underlying binary pattern. Usually, the patterns are optimized with *rasbhari* [46] with regard to the *overlap complexity*. This optimization is not deterministic. Thus, the patterns are different for every run which causes the phylogenetic trees to vary, at times significantly. We took the variance into consideration during our evaluation of *Multi-SpaM* by showing the minimum and maximum values of the *Robinson-Foulds* distance for *FSWM* and *Multi-SpaM* in Chapter 2. A similar variance can also be observed for all other extensions of *FSWM*, such as *Prot-SpaM* [75]. In contrast, other alignment-free methods such as *andi* and *co-phylog* are stable. In order to get a stable result, we could use a fixed pattern. However, this would, in turn, lead to another problem. No binary pattern is ideal for every dataset and a fixed pattern might be particularly bad for the dataset that is being analyzed. Unfortunately, there is no known way to optimize a pattern according to a given dataset. Therefore, all methods based on spaced-words use more or less random patterns. In order to reduce the variance of the distances based on spaced-word frequencies [72], multiple patterns have been used. However, pattern sets also increase the runtime. Furthermore, the variance may still be relatively high which was the case for *Prot-SpaM*. With regard to *Multi-SpaM*, we did not try to use multiple patterns because there is, in addition to the already stated drawbacks, also the random sampling step which is another source of variance. Morever, the variance is also a relevant information for an analysis of a phylogeny. Even though a stable result is desirable, it does not mean that the result is correct. Thus, we chose to give the user a clue of how stable the phylogenetic tree is by adding bootstrap values, as described in Section 5.3. The patterns do not only cause variance, the weight and number of *don't care positions* can also affect the result. We use a relatively high number of *don't care positions* to reliably remove background matches. In some cases, homologous matches can be missed due to overly long patterns. Conversely, accepted matches may not be homologous over their entire length. This issue has been addressed in order to find anchor points in Chapter 4. Here, the spaced-word matches have been extended with a standard X-drop approach starting from the middle position. However, even without such an approach, the default parameters of *Multi-SpaM* work reasonable well for all datasets. In some cases, the phylogenetic trees can be improved by using a different weight or length of the pattern. However, it is not possible to determine optimal parameters without a reference tree. Hence, the user has to rely on the default parameters. Only for large datasets, it might be beneficial to increase the weight in order to decrease the number of random spaced-word matches and thus the runtime.

In contrast to other alignment-free methods, *Multi-SpaM* does not calculate distances in order to build the phylogenetic tree. Thus, it only provides the topology of the tree, but not the branch lengths. Even though the topology of a tree is the most important information, branch lengths might still be of interest to better understand the phylogenetic relationships. If we use *Neighbor-Joining* instead of *RAxML* to calculate the optimal quartet tree topology, it would be possible to use the branch lengths of the quartet trees. However, there is no way to use them with *Quartet MaxCut*. Furthermore, there is method I know of that can fit branch lengths to a given tree with regard to a distance matrix. If there was such a method, a distance matrix could be calculated similar to *FSWM*.

The last limitation that I want to mention is the scalability to datasets with large amounts of sequences. While *Multi-SpaM* can handle long sequences very well, the number of quartet trees that is necessary to reconstruct the correct phylogeny increases very quickly with the number of sequences. A study by the developers of *Quartet MaxCut* recommends $n^{2.8}$ quartet trees for a dataset with n sequences [126] which was enough to reconstruct the correct phylogeny even with up to 30% incorrect quartet trees. Thus, the number of samples might need to be increased drastically for a high number of sequences. Additionally, the quartet trees should ideally be spread somewhat evenly across the quartet space. In Chapter 3, we ran into the problem that large portions of this space were not covered. This issue is also much more likely for a large number of sequences. In a recent paper, *Multi-SpaM* has been applied to a dataset of 220 *Salmonella* genomes [147]. Even when the number of samples was increased, the phylogenetic trees were of relatively poor quality.

7 Conclusion and outlook

In this thesis, I showed that it is possible to reconstruct accurate phylogenies with an alignment-free method that does not follow a standard approach based on pairwise distances. Multiple sequence comparison can be done effectively with a quartet tree based approach while keeping the runtime low. I showed that the quartet trees can be inferred from both nucleotide data and from putative insertions or deletions. For that purpose, we found blocks consisting of spaced-word occurrences. Similar blocks can also be used effectively as anchor points in order to speed up genome alignment.

One goal of this thesis was to find out whether character-based methods can be used to improve alignment-free methods. While the phylogenetic trees are of high quality, our limited *Maximum-Likelihood* approach did not outperform a simple *Neighbor-Joining* approach.

I already mentioned a few ways *Multi-SpaM* could be improved, e.g. by giving weights to the quartet trees. In the last part of this thesis, I want to give an outlook on possible further research. First, I discuss a few more ways to improve *Multi-SpaM*. In the second part, I suggest a few directions the tool could be developed further.

7.1 Possible improvements

As previously stated, a big limitation of *Multi-SpaM* is the scalability to large amounts of taxa. Thus, it should be the focus of further improvements of the method. The number of quartet trees could be increased in a fairly straightforward way by searching for larger P-blocks. In the original version, a random spaced-word occurrence is chosen which is compared to other occurrences of the same spaced-word. After four homologous occurrences are added to the P-block, the search is stopped. Alternatively, the search could be continued in order to find all possible homologous spaced-word occurrences. For a P-block spanning n sequences, up to $\binom{n}{4}$ quartet trees could be inferred. This would increase the total number of quartet trees drastically. Additionally, this could increase the quartet coverage for datasets with a high number of taxa. However, using RAxML [128] to infer that many quartet trees would slow down *Multi-SpaM* significantly. In order to keep the runtimes at an acceptable level, *Neighbor-Joining* needs to be used for such an approach. As shown in Chapter 5.2, doing so could even improve the accuracy of the quartet trees. The quartet coverage could also be increased in another way. Instead of sampling randomly, a weighted sampling could be used that would favor sets of four sequences for which no quartet tree has been found so far.

Since *Multi-SpaM* works well for datasets with relatively few sequences, a divide-and-conquer approach could also be an option. *Quartet MaxCut* already follows this approach. Thus, *Multi-SpaM* could be applied to multiple subsets of sequences. The individual phylogenetic trees could then be amalgamated into a super tree. This could be done in a similar way as in *SuperFine* [131]. A fast and scalable method could be used to built a phylogenetic tree for the entire dataset. Then, *Multi-SpaM* can be applied to the sequences in the individual clades of this tree. Alternatively, a scalable method could be run multiple times. Then, the resulting trees could be merged into a consensus tree. In some parts of the tree, there might be conflicts. These conflicts could be resolved by using *Multi-SpaM*.

The dataset could also be divided into non-overlapping subsets. For each subset, *Multi-SpaM* could build a phylogenetic tree. Then, the resulting trees could be used as constraint trees for *NJMerge* [91]. This method is an extension to *Neighbor-Joining*. It builts a phylogenetic tree that is compatible with a set of constraints. As a distance-based method, it also requires a distance matrix of all sequences. This matrix can be calculated by a highly scalable method. The constraint trees would then be used to improve the phylogenetic tree.

Lastly, I want to discuss how the overall performance of *Multi-SpaM* could be enhanced, even for smaller datasets. If the quality of the quartet trees could be assessed reliably, it is straightforward to use this information as weights for *Quartet MaxCut* [6]. In Chapter 5.1, I showed that support values can be calculated for every quartet tree using *RAxML*. These values can be used as weights. Similarly, if *Neighbor-Joining* is used to calculate the optimal quartet tree topologies, then a measure of treelikeness [54] could be calculated. In this case, higher weights would be given if the distances are (nearly) additive. Alternatively, if some information besides the mismatches in the micro-alignments can be found that supports a quartet tree, it could be used to define weights. As shown in Chapter 3, the sizes of the gaps between two adjacent P-blocks hold information about the phylogenetic relationships. In particular, if these trees coincide with the quartet trees of *Multi-SpaM*, then there is a high probability that these quartet trees are correct. Thus, these trees could be given a higher weight.

7.2 Further applications

So far, *Multi-SpaM* is only available for DNA sequences. Similar to *FSWM*, it could be applied to proteomes, as well. *Prot-SpaM* [75] is a very fast tool that often outperforms other alignment-free methods. P-blocks found in protein sequences may correspond to protein domains. Thus, this could be a good prospect for *Multi-SpaM*. However, protein sequences are much shorter which counteracts one of the biggest advantages of *Multi-SpaM*. Furthermore, *FSWM* has been applied to unassembled genomes. *Read-SpaM* [69] is one of several assembly-free methods that were published in recent years. Since this is a desirable application for alignment-free methods, it would make sense to develop *Multi-SpaM* in this direction.

There are also some applications for the other two projects of this thesis. The information contained in the size of the gaps between two adjacent P-blocks could be investigated for larger or even maximal P-blocks. In this case, it is not clear which P-blocks should be compared as they may not contain spaced-word occurrences from the same set of sequences. This can be solved by comparing all pairs of P-blocks. Alternatively, P-blocks that involve an identical subset of at least four sequences could be compared. In Chapter 4, we used spaced-word matches to find anchor points for a genome alignment tool. It might be possible to use spaced-word matches to calculate an entire (genome) alignment. Such an approach would follow a similar strategy as *dialign* [93]. It would align spaced-word matches as long as they are consistent with the rest of the alignment. In order to cover large portions of the genomes, it would be necessary to use multiple patterns with different weights. Longer spaced-word matches need to be found first as they are less likely to be random matches. In order to increase the likelihood of homologous matches further, spaced-words that occur in multiple sequences with a positive score to each other could be prioritized. As such, P-blocks from *Multi-SpaM* could be used as a starting point for the alignment.

References

[1] Athena Ahmadi, Alexander Behm, Nagesh Honnalli, Chen Li, Lingjie Weng, and Xiaohui Xie. Hobbes: optimized gram-based methods for efficient read alignment. *Nucleic acids research*, 40(6):e41, 2012.

[2] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3): 403–410, 1990.

[3] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17): 3389–3402, Sep 1997.

[4] Sasha K Ames, David A Hysom, Shea N Gardner, G Scott Lloyd, Maya B Gokhale, and Jonathan E Allen. Scalable metagenomic taxonomy classification using a reference genome database. *Bioinformatics*, 29(18):2253–2260, Sep 2013.

[5] Samuel V Angiuoli and Steven L Salzberg. Mugsy: fast multiple alignment of closely related whole genomes. *Bioinformatics*, 27(3):334–342, Feb 2011.

[6] Eliran Avni, Reuven Cohen, and Sagi Snir. Weighted quartets phylogenetics. *Systematic biology*, 64(2):233–242, Mar 2015.

[7] Eliran Avni, Zahi Yona, Reuven Cohen, and Sagi Snir. The Performance of Two Supertree Schemes Compared Using Synthetic and Real Data Quartet Input. *Journal of molecular evolution*, 86(2):150–165, 2018.

[8] E. Ray Lankester B.A. On the use of the term homology in modern zoology, and the distinction between homogenetic and homoplastic agreements. *Annals and Magazine of Natural History*, 6(31):34–43, 1870.

[9] Bernard R. Baum. Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. *Taxon*, 41(1):3–10, 1992.

[10] Gaëtan Benoit, Pierre Peterlongo, Mahendra Mariadassou, Erwan Drezen, Sophie Schbath, Dominique Lavenier, and Claire Lemaitre. Multiple comparative metagenomics using multiset k-mer counting. *PeerJ Computer Science*, 2:e94, 2016.

[11] Guillaume Bernard, Cheong Xin Chan, Yao-ban Chan, Xin-Yi Chua, Yingnan Cong, James M Hogan, Stefan R Maetschke, and Mark A Ragan. Alignment-free inference of hierarchical and reticulate phylogenomic relationships. *Briefings in Bioinformatics*, 20(2):426–435, 03 2019.

[12] Olaf R.P. Bininda-Emonds. The evolution of supertrees. *Trends in Ecology and Evolution*, 19(6):315 – 322, 2004.

[13] B Edwin Blaisdell. A measure of the similarity of sets of sequences not requiring sequence alignment. *Proceedings of the National Academy of Sciences*, 83(14):5155–5159, 1986.

[14] Marcus Boden, Martin Schöneich, Sebastian Horwege, Sebastian Lindner, Chris-André Leimeister, and Burkhard Morgenstern. Alignment-free sequence comparison with spaced k-mers. In Tim Beißbarth, Martin Kollmar, Andreas Leha, Burkhard Morgenstern, Anne-Kathrin Schultz, Stephan Waack, and Edgar Wingender, editors, *German Conference on Bioinformatics 2013*, volume 34 of *OpenAccess Series in Informatics (OASIcs)*, pages 24–34. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.

[15] Andrei Z Broder. Identifying and filtering near-duplicate documents. In *Annual Symposium on Combinatorial Pattern Matching*, pages 1–10. Springer, 2000.

[16] David Bryant. A classification of consensus methods for phylogenetics. In *Bioconsensus*, pages 163–183, 2003.

[17] Benjamin Buchfink, Chao Xie, and D. H Huson. Fast and sensitive protein alignment using DIAMOND. *Nature Methods*, 12:59–60, 2015.

[18] Sourav Chatterji, Ichitaro Yamazaki, Zhaojun Bai, and Jonathan A Eisen. Compostbin: A dna composition-based algorithm for binning environmental shotgun reads. In *Annual International Conference on Research in Computational Molecular Biology*, pages 17–28. Springer, 2008.

[19] Francesca Chiaromonte, Von Bing Yap, and Webb Miller. Scoring pairwise genomic sequence alignments. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, and Teri E. Klein, editors, *Pacific Symposium on Biocomputing*, pages 115–126, Lihue, Hawaii, 2002.

[20] Benny Chor and Tamir Tuller. Maximum likelihood of evolutionary trees is hard. In *Annual International Conference on Research in Computational Molecular Biology*, pages 296–310, Berlin, Heidelberg, 2005. Springer.

[21] Matteo Comin and Davide Verzotto. The irredundant class method for remote homology detection of protein sequences. *Journal of Computational Biology*, 18(12): 1819–1829, Dec 2011.

[22] Daniel A. Dalquen, Maria Anisimova, Gaston H. Gonnet, and Christophe Dessimoz. ALF - a simulation framework for genome evolution. *Molecular Biology and Evolution*, 29:1115–1123, 2012.

[23] Charles Darwin. *On the origin of species by means of natural selection, or, The preservation of favoured races in the struggle for life*. John Murray, London, 1859.

[24] Arthur L Delcher, Simon Kasif, Robert D Fleischmann, Jeremy Peterson, Owen White, and Steven L Salzberg. Alignment of whole genomes. *Nucleic acids research*, 27(11):2369–2376, Jun 1999.

[25] Thomas Dencker, Chris-André Leimeister, Michael Gerth, Christoph Bleidorn, Sagi Snir, and Burkhard Morgenstern. Multi-spam: a maximum-likelihood approach to phylogeny reconstruction using multiple spaced-word matches and quartet trees. pages 227–241, 2018.

[26] Thomas Dencker, Chris-André Leimeister, Michael Gerth, Christoph Bleidorn, Sagi Snir, and Burkhard Morgenstern. 'Multi-SpaM': a maximum-likelihood approach to phylogeny reconstruction using multiple spaced-word matches and quartet trees. *NAR Genomics and Bioinformatics*, 2(1), 3 2020. lqz013.

[27] GJ Olsen DL Swofford. Phylogeny reconstruction. In DM Hillis and C. Moritz, editors, *Molecular Systematics*, pages 411–501. Sinauer Associates: Sunderland, Massachusetts, 1990.

[28] Alexandre Drouin, Sébastien Giguère, Maxime Déraspe, Mario Marchand, Michael Tyers, Vivian G Loo, Anne-Marie Bourgault, François Laviolette, and Jacques Corbeil. Predictive computational phenotyping and biomarker discovery using reference-free genome comparisons. *BMC Genomics*, 17(1):754, Sep 2016.

[29] Robert C Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, 32(5):1792–1797, 2004.

[30] Christoph Elfmann. Implementation of sampling strategies for Filtered Spaced-Word Matches. Bachelor thesis, University of Göttingen, Germany, July 2019.

[31] Huan Fan, Anthony R Ives, Yann Surget-Groba, and Charles H Cannon. An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data. *BMC Genomics*, 16:522, Jul 2015.

[32] Michael Farrar. Striped Smith–Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, 23(2):156–161, 11 2006.

[33] James S. Farris. Methods for computing Wagner trees. *Systematic Biology*, 19:83–92, 1970.

[34] Maria Federico, Mauro Leoncini, Manuela Montangero, and Paolo Valente. Direct vs 2-stage approaches to structured motif finding. *Algorithms for Molecular Biology*, 7(1):20, Aug 2012.

[35] Joseph Felsenstein. Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, 39(4):783–791, 1985.

[36] Joseph Felsenstein. PHYLIP - Phylogeny Inference Package (Version 3.2). *Cladistics*, 5:164–166, 1989.

[37] Joseph Felsenstein. *Inferring phylogenies*, volume 2. MA: Sinauer associates, Sunderland, 2004.

[38] Da-Fei Feng and Russell F. Doolittle. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351–360, Aug 1987.

[39] Johannes Fischer and Volker Heun. A new succinct representation of rmq-information and improvements in the enhanced suffix array. In *International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, pages 459–470. Springer, 2007.

[40] Walter Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Zoology*, 20:406–416, 1971.

[41] Les R Foulds and Ronald L Graham. The steiner problem in phylogeny is np-complete. *Advances in Applied mathematics*, 3(1):43–49, 1982.

[42] Olivier Gascuel. BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Molecular biology and evolution*, 14(7):685–695, 1997.

[43] J Peter Gogarten and Jeffrey P Townsend. Horizontal gene transfer, genome innovation and evolution. *Nature Reviews Microbiology*, 3(9):679–687, 2005.

[44] Allan D Gordon. Consensus supertrees: The synthesis of rooted trees containing overlapping sets of labeled leaves. *Journal of Classification*, 3(2):335–348, Sep 1986.

[45] Stéphane Guindon, Jean-François Dufayard, Vincent Lefort, Maria Anisimova, Wim Hordijk, and Olivier Gascuel. New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of phyml 3.0. *Systematic biology*, 59(3):307–321, 2010.

[46] Lars Hahn, Chris-André Leimeister, Rachid Ounit, Stefano Lonardi, and Burkhard Morgenstern. *rasbhari*: optimizing spaced seeds for database searching, read mapping and alignment-free sequence comparison. *PLOS Computational Biology*, 12(10): e1005107, 2016.

[47] Bernhard Haubold. Alignment-free phylogenetics and population genetics. *Briefings in Bioinformatics*, 15:407–418, 2014.

[48] Bernhard Haubold, Nora Pierstorff, Friedrich Möller, and Thomas Wiehe. Genome comparison without alignment using shortest unique substrings. *BMC Bioinformatics*, 6:123, 2005.

[49] Bernhard Haubold, Peter Pfaffelhuber, Mirjana Domazet-Loso, and Thomas Wiehe. Estimating mutation distances from unaligned genomes. *Journal of Computational Biology*, 16:1487–1500, 2009.

[50] Bernhard Haubold, Fabian Klötzl, and Peter Pfaffelhuber. andi: Fast and accurate estimation of evolutionary distances between closely related genomes. *Bioinformatics*, 31:1169–1175, 2015.

[51] Jotun Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Mathematical Biosciences*, 98(2):185 – 200, 1990.

[52] Michael D. Hendy and David Penny. Branch and bound algorithms to determine minimal evolutionary trees. *Mathematical Biosciences*, 59(2):277–290, 1982.

[53] Cody E. Hinchliff, Stephen A. Smith, James F. Allman, J. Gordon Burleigh, Ruchi Chaudhary, Lyndon M. Coghill, Keith A. Crandall, Jiabin Deng, Bryan T. Drew, Romina Gazis, Karl Gude, David S. Hibbett, Laura A. Katz, H. Dail Laughinghouse, Emily Jane McTavish, Peter E. Midford, Christopher L. Owen, Richard H. Ree, Jonathan A. Rees, Douglas E. Soltis, Tiffani Williams, and Karen A. Cranston. Synthesis of phylogeny and taxonomy into a comprehensive tree of life. *Proceedings of the National Academy of Sciences*, 112(41):12764–12769, 2015.

[54] Barbara R Holland, Katharina T Huber, Andreas Dress, and Vincent Moulton. δ plots: a tool for analyzing phylogenetic distance data. *Molecular biology and evolution*, 19(12):2051–2059, 2002.

[55] Sebastian Horwege, Sebastian Lindner, Marcus Boden, Klaus Hatje, Martin Kollmar, Chris-André Leimeister, and Burkhard Morgenstern. *Spaced words* and *kmacs*: fast alignment-free sequence comparison based on inexact word matches. *Nucleic Acids Research*, 42:W7–W11, 2014.

[56] Jaime Huerta-Cepas, François Serra, and Peer Bork. ETE 3: reconstruction, analysis, and visualization of phylogenomic data. *Molecular biology and evolution*, 33(6):1635–1638, 2016.

[57] Jaime Huerta-Cepas, Damian Szklarczyk, Kristoffer Forslund, Helen Cook, Davide Heller, Mathias C Walter, Thomas Rattei, Daniel R Mende, Shinichi Sunagawa, Michael Kuhn, Lars J Jensen, Christian von Mering, and Peer Bork. eggNOG 4.5: a hierarchical orthology framework with improved functional annotations for eukaryotic, prokaryotic and viral sequences. *Nucleic Acids Research*, 44(D1):D286–D293, Jan 2016.

[58] Daniel H Huson and Celine Scornavacca. A survey of combinatorial methods for phylogenetic networks. *Genome biology and evolution*, 3:23–35, 2011.

[59] Daniel H Huson, Scott M Nettles, and Tandy J Warnow. Disk-covering, a fast-converging method for phylogenetic tree reconstruction. *Journal of Computational Biology*, 6(3-4):369–386, 1999.

[60] Lucian Ilie and Silvana Ilie. Multiple spaced seeds for homology search. *Bioinformatics*, 23:2969–2977, 2007.

[61] Lucian Ilie, Silvana Ilie, and Anahita M. Bigvand. SpEED: fast computation of sensitive spaced seeds. *Bioinformatics*, 27:2433–2434, 2011.

[62] Thomas H. Jukes and Charles R. Cantor. Evolution of protein molecules. In *Mammalian Protein Metabolism*, pages 21 – 132. Academic Press, 1969.

[63] Kazutaka Katoh and Daron M. Standley. MAFFT multiple sequence alignment software version 7: Improvements in performance and usability. *Molecular Biology and Evolution*, 30(4):772–780, 2013.

[64] Kazutaka Katoh, Kazuharu Misawa, Kei-Ichi Kuma, and Takashi Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic Acids Research*, 30(14):3059–3066, 2002.

[65] John Frank Charles Kingman. The coalescent. *Stochastic Processes and their Applications*, 13(3):235 – 248, 1982.

[66] Hashem Koohy, Nigel P Dyer, John E Reid, Georgy Koentges, and Sascha Ott. An alignment-free model for comparison of regulatory sequences. *Bioinformatics*, 26(19): 2391–2397, Oct 2010.

[67] Stefan Kurtz, Adam Phillippy, Arthur L Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, and Steven L Salzberg. Versatile and open software for comparing large genomes. *Genome Biol.*, 5(2):R12, 2004.

[68] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10:R25, 2009.

[69] Anna-Katharina Lau, Svenja Dörrer, Chris-André Leimeister, Christoph Bleidorn, and Burkhard Morgenstern. Read-SpaM: assembly-free and alignment-free compari-

son of bacterial genomes with low sequencing coverage. *BMC bioinformatics*, 20(20): 638, 2019.

[70] Vincent Lefort, Richard Desper, and Olivier Gascuel. FastME 2.0: A Comprehensive, Accurate, and Fast Distance-Based Phylogeny Inference Program. *Molecular Biology and Evolution*, 32(10):2798–2800, 06 2015.

[71] Chris-André Leimeister and Burkhard Morgenstern. *kmacs*: the *k*-mismatch average common substring approach to alignment-free sequence comparison. *Bioinformatics*, 30:2000–2008, 2014.

[72] Chris-André Leimeister, Marcus Boden, Sebastian Horwege, Sebastian Lindner, and Burkhard Morgenstern. Fast alignment-free sequence comparison using spaced-word frequencies. *Bioinformatics*, 30:1991–1999, 2014.

[73] Chris-André Leimeister, Salma Sohrabi-Jahromi, and Burkhard Morgenstern. Fast and accurate phylogeny reconstruction using filtered spaced-word matches. *Bioinformatics*, 33:971–979, 2017.

[74] Chris-André Leimeister, Thomas Dencker, and Burkhard Morgenstern. Accurate multiple alignment of distantly related genome sequences using filtered spaced word matches as anchor points. *Bioinformatics*, 35(2):211–218, 01 2019.

[75] Chris-Andre Leimeister, Jendrik Schellhorn, Svenja Dörrer, Michael Gerth, Christoph Bleidorn, and Burkhard Morgenstern. Prot-spam: Fast alignment-free phylogeny reconstruction based on whole-proteome sequences. *GigaScience*, 8(3): giy148, 2019.

[76] Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. pages 564–575, 2001.

[77] Garmay Leung and Michael B Eisen. Identifying cis-regulatory sequences by word profile similarity. *PLoS ONE*, 4(9):e6901, Sep 2009.

[78] Henry CM Leung, Siu-Ming Yiu, Bin Yang, Yu Peng, Yi Wang, Zhihua Liu, Jingchi Chen, Junjie Qin, Ruiqiang Li, and Francis YL Chin. A robust and accurate binning algorithm for metagenomic sequences with arbitrary species abundance ratio. *Bioinformatics*, 27(11):1489–1495, Jun 2011.

[79] Ruiqiang Li, Chang Yu, Yingrui Li, Tak-Wah Lam, Siu-Ming Yiu, Karsten Kristiansen, and Jun Wang. SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*, 25(15):1966–1967, Aug 2009.

[80] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[81] T. Lingner and P. Meinicke. Remote homology detection based on oligomer distances. *Bioinformatics*, 22(18):2224–2231, Sep 2006.

[82] Thomas Lingner and Peter Meinicke. Word correlation matrices for protein sequence analysis and remote homology detection. *BMC Bioinformatics*, 9(1):259, Jun 2008.

[83] Ross A. Lippert, Haiyan Huang, and Michael S. Waterman. Distributional regimes for the number of k-word matches between two random sequences. *Proceedings of the National Academy of Sciences*, 99:13980–13989, 2002.

[84] Lin Liu, Yinhu Li, Siliang Li, Ni Hu, Yimin He, Ray Pong, Danni Lin, Lihua Lu, and Maggie Law. Comparison of next-generation sequencing systems. *Journal of Biomedicine and Biotechnology*, 2012:251364, 2012.

[85] Yang Young Lu, Kujin Tang, Jie Ren, Jed A Fuhrman, Michael S Waterman, and Fengzhu Sun. CAFE: aCcelerated Alignment-FrEe sequence analysis. *Nucleic Acids Research*, 45(W1):W554–W559, 07 2017.

[86] Bin Ma, John Tromp, and Ming Li. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, 18:440–445, 2002.

[87] Wayne P Maddison and L Lacey Knowles. Inferring phylogeny despite incomplete lineage sorting. *Systematic biology*, 55(1):21–30, Feb 2006.

[88] Udi Manber and Gene Myers. Suffix arrays: a new method for on-line string searches. volume SODA '90, pages 319–327, 1990.

[89] Ernst Mayr. *Systematics and the Origin of Species*. Columbia University Press, New York, 1942.

[90] Peter Meinicke. UProC: tools for ultra-fast protein domain classification. *Bioinformatics*, 31(9):1382–1388, May 2015.

[91] Erin K Molloy and Tandy Warnow. Statistically consistent divide-and-conquer pipelines for phylogeny estimation using NJMerge. *Algorithms for Molecular Biology*, 14(1):14, 2019.

[92] G William Moore, M Goodman, and J Barnabas. An iterative approach from the standpoint of the additive hypothesis to the dendrogram problem posed by molecular data sets. *Journal of theoretical biology*, 38(3):423–457, 1973.

[93] Burkhard Morgenstern. DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15:211–218, 1999.

[94] Burkhard Morgenstern, Sonja J. Prohaska, Dirk Pöhler, and Peter F. Stadler. Multiple sequence alignment with user-defined anchor points. *Algorithms for Molecular Biology*, 1:6, 2006.

[95] Burkhard Morgenstern, Bingyao Zhu, Sebastian Horwege, and Chris-André Leimeister. Estimating evolutionary distances between genomic sequences from spaced-word matches. *Algorithms for Molecular Biology*, 10:5, 2015.

[96] Burkhard Morgenstern, Svenja Schöbel, and Chris-André Leimeister. Phylogeny reconstruction based on the length distribution of k-mismatch common substrings. *Algorithms for Molecular Biology*, 12(1):27, 2017.

[97] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48: 443–453, 1970.

[98] Kevin C. Nixon. The parsimony ratchet, a new method for rapid parsimony analysis. *Cladistics*, 15(4):407–414, 1999.

[99] Cédric Notredame, Desmond G Higgins, and Jaap Heringa. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of molecular biology*, 302(1):205–217, 2000.

[100] William of Ockham and Johannes Trechsel. *Quaestiones et decisiones in quattuor libros Sententiarum Petri Lombardi: Centilogium theologicum.* Johannes Trechsel, 1495.

[101] Brian D Ondov, Todd J Treangen, Páll Melsted, Adam B Mallonee, Nicholas H Bergman, Sergey Koren, and Adam M Phillippy. Mash: fast genome and metagenome distance estimation using minhash. *Genome biology*, 17(1):132, 2016.

[102] Benedict Paten, Dent Earl, Ngan Nguyen, Mark Diekhans, Daniel Zerbino, and David Haussler. Cactus: Algorithms for genome multiple sequence alignment. *Genome research*, 21(9):1512–1528, 2011.

[103] Rob Patro, Stephen M Mount, and Carl Kingsford. Sailfish enables alignment-free isoform quantification from rna-seq reads using lightweight algorithms. *Nature biotechnology*, 32(5):462 – 464, 2014.

[104] Ji Qi, Hong Luo, and Bailin Hao. CVTree: a phylogenetic tree reconstruction tool based on whole genomes. *Nucleic Acids Research*, 32(suppl 2):W45–W47, 2004.

[105] Mark A Ragan. Phylogenetic inference based on matrix representation of trees. *Molecular phylogenetics and evolution*, 1(1):53–58, 1992.

[106] Jie Ren, Kai Song, Fengzhu Sun, Minghua Deng, and Gesine Reinert. Multiple alignment-free sequence comparison. *Bioinformatics*, 29(21):2690–2698, 2013.

[107] David F Robinson. Comparison of labeled trees with valency three. *Journal of Combinatorial Theory, Series B*, 11(2):105 – 119, 1971.

[108] David F Robinson and Les Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131–147, 1981.

[109] Sophie Röhling, Alexander Linne, Jendrik Schellhorn, Morteza Hosseini, Thomas Dencker, and Burkhard Morgenstern. The number of k-mer matches between two dna sequences as a function of k and applications to estimate phylogenetic distances. *PLOS ONE*, in press, 2020.

[110] Fredrik Ronquist and John P. Huelsenbeck. MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19:1572–1574, 2003.

[111] Andrey Rzhetsky and Masatoshi Nei. A simple method for estimating and testing minimum-evolution trees. *Molecular Biology and Evolution*, 9, 11 1991.

[112] Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

[113] Andreas Sand, Morten K. Holt, Jens Johansen, Gerth Stølting Brodal, Thomas Mailund, and Christian N. S. Pedersen. tqDist: a library for computing the quartet and triplet distances between binary or general trees. *Bioinformatics*, 30(14): 2079–2080, 03 2014.

[114] MJ Sanderson, Marius Nicolae, and Michelle M McMahon. Homology-aware phylogenomics at gigabase scales. *Systematic biology*, 66(4):590–603, 2017.

[115] Frederick Sanger, Steven Nicklen, and Alan R Coulson. Dna sequencing with chain-terminating inhibitors. *Proceedings of the national academy of sciences*, 74(12):5463–5467, 1977.

[116] Shahab Sarmashghi, Kristine Bohmann, M Thomas P Gilbert, Vineet Bafna, and Siavash Mirarab. Skmer: assembly-free and alignment-free sample identification using genome skims. *Genome biology*, 20(1):34, 2019.

[117] Fabian Schreiber, Gert Wörheide, and Burkhard Morgenstern. Orthoselect: a web server for selecting orthologous gene alignments from est sequences. *Nucleic Acids Research*, 37:W185–188, 2009.

[118] Gur Sevillya, Zeev Frenkel, and Sagi Snir. Triplet maxcut: a new toolkit for rooted supertree. *Methods in Ecology and Evolution*, 7(11):1359–1365, 2016.

[119] Hidetoshi Shimodaira and Masami Hasegawa. Multiple comparisons of log-likelihoods with applications to phylogenetic inference. *Molecular biology and evolution*, 16(8): 1114–1114, 1999.

[120] Fabian Sievers, Andreas Wilm, David Dineen, Toby J Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Söding, Julie D Thompson, and Desmond G Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7:539, Oct 2011.

[121] Mark P Simmons and Helga Ochoterena. Gaps as characters in sequence-based phylogenetic analyses. *Systematic biology*, 49(2):369–381, 2000.

[122] Jared T Simpson and Mihai Pop. The theory and practice of genome sequence assembly. *Annual review of genomics and human genetics*, 16:153–172, 2015.

[123] Gregory E. Sims, Se-Ran Jun, Guohong A. Wu, and Sung-Hou Kim. Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. *Proceedings of the National Academy of Sciences*, 106:2677–2682, 2009.

[124] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[125] Sagi Snir and Satish Rao. Quartets MaxCut: a divide and conquer quartets algorithm. *IEEE/ACM Trans Comput Biol Bioinform*, 7(4):704–718, 2010.

[126] Sagi Snir and Satish Rao. Quartet MaxCut: A fast algorithm for amalgamating quartet trees. *Molecular Phylogenetics and Evolution*, 62:1 – 8, 2012.

[127] Robert R. Sokal and Charles D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.

[128] Alexandros Stamatakis. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30:1312–1313, 2014.

[129] Michael Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9(1):91–116, Jan 1992.

[130] Zachary D Stephens, Skylar Y Lee, Faraz Faghri, Roy H Campbell, Chengxiang Zhai, Miles J Efron, Ravishankar Iyer, Michael C Schatz, Saurabh Sinha, and Gene E Robinson. Big Data: Astronomical or Genomical? *PLoS Biol.*, 13(7):e1002195, Jul 2015.

[131] M. S. Swenson, R. Suri, C. R. Linder, and T. Warnow. SuperFine: fast and accurate supertree estimation. *Syst. Biol.*, 61(2):214–227, Mar 2012.

[132] M Shel Swenson, Rahul Suri, C Randal Linder, and Tandy Warnow. An experimental study of quartets maxcut and other supertree methods. *Algorithms for Molecular Biology*, 6(1):7, 2011.

[133] D. Swofford. Paup*. phylogenetic analysis using parsimony (*and other methods). version 4.0b10. *Sinauer Associates, Sunderland, Massachusetts*, 2003.

[134] Olga Tanaseichuk, James Borneman, and Tao Jiang. Separating metagenomic short reads into genomes via clustering. *Algorithms for Molecular Biology*, 7(1):27, 2012.

[135] Simon Tavaré. Some probabilistic and statistical problems in the analysis of dna sequences. *Lectures on mathematics in the life sciences*, 17(2):57–86, 1986.

[136] Hanno Teeling, Jost Waldmann, Thierry Lombardot, Margarete Bauer, and Frank Oliver Glöckner. Tetra: a web-service and a stand-alone program for the analysis and comparison of tetranucleotide usage patterns in dna sequences. *BMC bioinformatics*, 5(1):163, 2004.

[137] Douglas L Theobald. A formal test of the theory of universal common ancestry. *Nature*, 465(7295):219–222, 2010.

[138] Julie D. Thompson, Desmond G. Higgins, and Toby J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

[139] Jeffrey L Thorne, Hirohisa Kishino, and Joseph Felsenstein. An evolutionary model for maximum likelihood alignment of dna sequences. *Journal of Molecular Evolution*, 33(2):114–124, 1991.

[140] Jeffrey L. Thorne, Hirohisa Kishino, and Joseph Felsenstein. Inching toward reality: An improved likelihood model of sequence evolution. *Journal of Molecular Evolution*, 34(1):3–16, Jan 1992.

[141] Igor Ulitsky, David Burstein, Tamir Tuller, and Benny Chor. The average common substring approach to phylogenomic reconstruction. *Journal of Computational Biology*, 13:336–350, 2006.

[142] Susana Vinga and Jonas Almeida. Alignment-free sequence comparison - a review. *Bioinformatics*, 19(4):513–523, 2003.

[143] Lusheng Wang and Tao Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.

[144] Yi Wang, Henry CM Leung, Siu-Ming Yiu, and Francis YL Chin. Metacluster 5.0: a two-round binning approach for metagenomic data for low-abundance species in a noisy sample. *Bioinformatics*, 28(18):i356–i362, 2012.

[145] Robert M. Waterhouse, Fredrik Tegenfeldt, Jia Li, Evgeny M. Zdobnov, and Evgenia V. Kriventseva. Orthodb: a hierarchical catalog of animal, fungal and bacterial orthologs. *Nucleic Acids Research*, 41:D358–D365, 2013.

[146] James D Watson and Francis HC Crick. The structure of dna. *Cold Spring Harbor symposia on quantitative biology*, 18:123–131, 1953.

[147] Roland Wittler. Alignment- and reference-free phylogenomics with colored de-bruijn graphs. *bioRxiv*, 2019.

[148] Karen M. Wong, Marc A. Suchard, and John P. Huelsenbeck. Alignment uncertainty and genomic analysis. *Science*, 319(5862):473–476, 2008.

[149] Yu-Wei Wu and Yuzhen Ye. A novel abundance-based algorithm for binning metagenomic sequences using l-tuples. *Journal of Computational Biology*, 18(3):523–534, 2011.

[150] Ziheng Yang. Maximum likelihood phylogenetic estimation from dna sequences with variable rates over sites: Approximate methods. *Journal of Molecular Evolution*, 39 (3):306–314, Sep 1994.

[151] Huiguang Yi and Li Jin. Co-phylog: an assembly-free phylogenomic approach for closely related organisms. *Nucleic Acids Research*, 41:e75, 2013.

[152] Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829, 2008.

[153] Chao Zhang, Maryam Rabiee, Erfan Sayyari, and Siavash Mirarab. ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. *BMC Bioinformatics*, 19(Suppl 6):153, 05 2018.

[154] Andrzej Zielezinski, Susana Vinga, Jonas Almeida, and Wojciech M Karlowski. Alignment-free sequence comparison: benefits, applications, and tools. *Genome Biology*, 18(1):186, 10 2017.

[155] Andrzej Zielezinski, Hani Z Girgis, Guillaume Bernard, Chris-Andre Leimeister, Kujin Tang, Thomas Dencker, Anna Katharina Lau, Sophie Röhling, Jae Jin Choi, Michael S Waterman, Matteo Comin, Sung-Hou Kim, Susana Vinga, Jonas S Almeida, Cheong Xin Chan, Benjamin James, Fengzhu Sun, Burkhard Morgenstern, and Wojciech M. Karlowski. Benchmarking of alignment-free sequence comparison methods. *Genome Biology*, 20(1):144, 07 2019.

[156] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.

[157] Emile Zuckerkandl and Linus Pauling. Molecules as documents of evolutionary history. *Journal of Theoretical Biology*, 8(2):357 – 366, 1965.