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

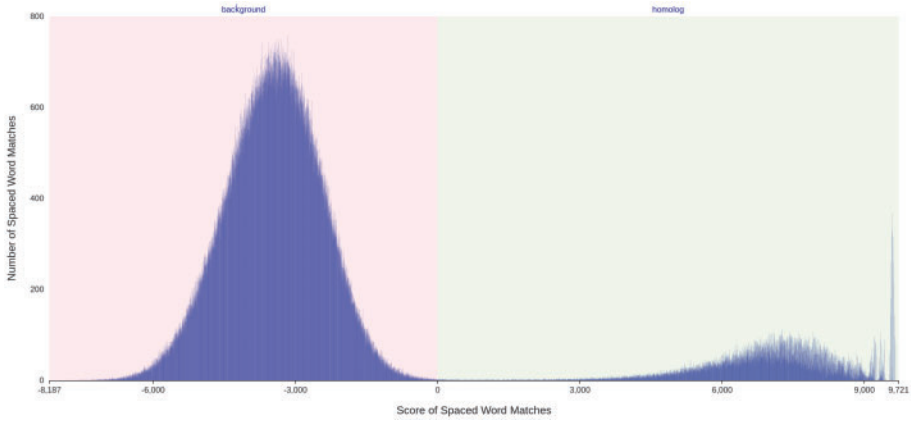

Fig. 1. *Spaced-words histogram* for a comparison of two bacterial genomes, *Phaeobacter gallaeciensis* 2.10 and *Rhodobacterales bacterium Y4I*. All possible spaced-word matches with respect to a given binary pattern P are identified, and their scores are calculated as explained in the main text. The 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

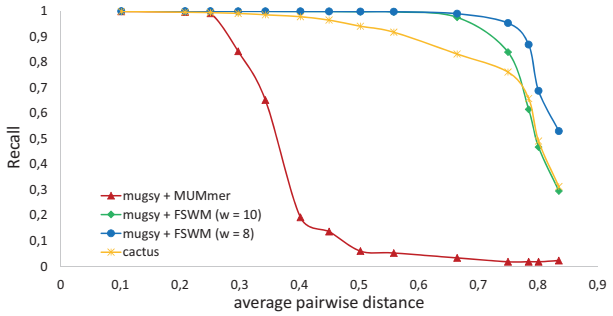

Fig. 2. *Recall* values for *Mugsy* using anchor points generated with *FSWM* and with *MUMmer*, respectively, as well as for *Cactus*. Test data were simulated genomic sequences generated with *ALF*, see main text for details. *FSWM* was run with the default *weight* $w = 10$, i.e. with 10 *match positions*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

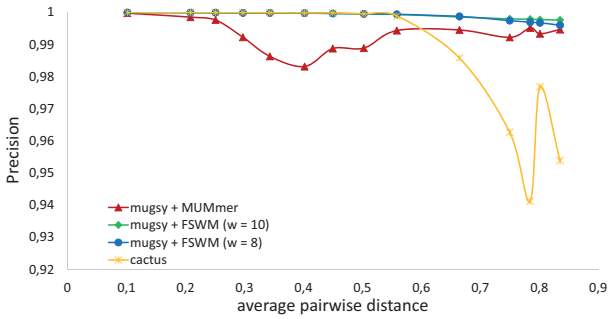

Fig. 3. *Precision* values for *Mugsy* with *FSWM* and *MUMmer* anchor points respectively, and for *Cactus*. Test data and parameter values as in Figure 2

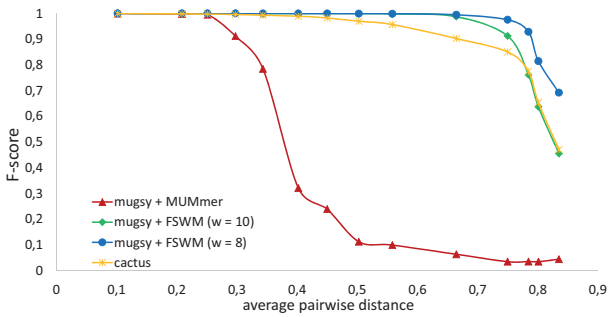

Fig. 4. *F-Score* values for *Mugsy* with *FSWM* and *MUMmer* anchor points, respectively, and for *Cactus*. Test data and parameter values as in Figure 2

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

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*.

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*.

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.