Bioinformatics, 32(21), 2016, 3246–3251
doi: 10.1093/bioinformatics/btw412
Advance Access Publication Date: 4 July 2016
Original Paper

OXFORD

Sequence analysis

Application of the MAFFT sequence alignment program to large data—reexamination of the usefulness of chained guide trees

Kazunori D. Yamada[1,2], Kentaro Tomii[2,3] and Kazutaka Katoh[2,4,*]

[1]Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan, [2]Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Tokyo 135-0064, Japan, [3]Biotechnology Research Institute for Drug Discovery, National Institute of Advanced Industrial Science and Technology (AIST), Tokyo 135-0064, Japan and [4]Immunology Frontier Research Center, Osaka University, Suita 565-0871, Japan

*To whom correspondence should be addressed
Associate Editor: Alfonso Valencia

Received on March 28, 2016; revised on May 27, 2016; accepted on June 20, 2016

Abstract

Motivation: Large multiple sequence alignments (MSAs), consisting of thousands of sequences, are becoming more and more common, due to advances in sequencing technologies. The MAFFT MSA program has several options for building large MSAs, but their performances have not been sufficiently assessed yet, because realistic benchmarking of large MSAs has been difficult. Recently, such assessments have been made possible through the HomFam and ContTest benchmark protein datasets. Along with the development of these datasets, an interesting theory was proposed: chained guide trees increase the accuracy of MSAs of structurally conserved regions. This theory challenges the basis of progressive alignment methods and needs to be examined by being compared with other known methods including computationally intensive ones.

Results: We used HomFam, ContTest and OXFam (an extended version of OXBench) to evaluate several methods enabled in MAFFT: (1) a progressive method with approximate guide trees, (2) a progressive method with chained guide trees, (3) a combination of an iterative refinement method and a progressive method and (4) a less approximate progressive method that uses a rigorous guide tree and consistency score. Other programs, Clustal Omega and UPP, available for large MSAs, were also included into the comparison. The effect of method 2 (chained guide trees) was positive in ContTest but negative in HomFam and OXFam. Methods 3 and 4 increased the benchmark scores more consistently than method 2 for the three datasets, suggesting that they are safer to use.

Availability and Implementation: http://mafft.cbrc.jp/alignment/software/

Contact: katoh@ifrec.osaka-u.ac.jp

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

1 Introduction

A large number of biological sequences have rapidly accumulated due to advances in sequencing technology. A multiple sequence alignment (MSA) consisting of thousands of sequences is frequently needed (e.g. Kamisetty *et al.*, 2013), but the calculation of such large MSAs is a challenging problem. MAFFT is one of the most popular MSA programs and has several options for this scale data input. However, it is unclear which option should be used in such situations. To answer this question, solid benchmarks based on empirical data are critical. Recently, a new benchmark based on real protein data, ContTest, was proposed by Fox *et al.* (2016). A remarkable advantage of ContTest is that it is directly linked to a

specific downstream analysis, contact prediction. Meanwhile, conventional benchmark datasets, such as BAliBASE (Thompson *et al.*, 1999) and HomFam (Sievers *et al.*, 2011), are useful in assessing general accuracy of MSA methods based on structurally conserved residues. Using these two types of benchmarks, we assessed the performance of several different options, including newly enabled ones, of MAFFT for a large MSA consisting of several to tens of thousands of sequences. Such tests also provide developers of related tools with general information about which techniques are useful for improving the quality of large MSAs.

In particular, we closely examined a surprising and controversial finding (Boyce *et al.*, 2014; Fox *et al.*, 2016; Tan *et al.*, 2015) on the effect of guide trees in MSA calculation; do random chained guide trees perform better than conventional guide trees? This issue was first raised by Boyce *et al.* (2014), reporting that chained guide trees give better MSAs than the default ones, especially in MAFFT (Katoh *et al.*, 2002) and MUSCLE (Edgar, 2004a, b). Tan *et al.* (2015) reported opposite observations that chained guide trees give poorer MSAs, using simulation and phylogeny-based evaluations. Sievers *et al.* (2014) systematically examined the effects of chained guide trees and balanced ones for small MSAs. Fox *et al.* (2016) presented additional evidence to support the advantage of chained guide trees using the ContTest dataset and confirmed the utility of chained guide trees for large MSAs of structurally conserved regions. It is known that the accuracy of MSA can depend on the evaluation criteria or the purpose of downstream analyses, as mentioned in Tan *et al.* (2015) and Fox et al. (2016). Therefore, it is important for our reexamination to use the evaluation criteria based on structural conservation, as in Boyce *et al.* (2014) and Fox et al. (2016).

2 Materials and methods

2.1 Algorithms

We compared several different methods implemented in MAFFT for large MSAs.

1. A progressive method (Feng and Doolittle, 1987; Higgins and Sharp, 1988; Hogeweg and Hesper, 1984) with approximate guide trees. The default option of MAFFT, FFT-NS-2, uses a variant of this approach. An approximate guide tree is first built based on the number of shared 6mers, on which the initial alignment is progressively built. Then, the second guide tree is computed on the first alignment. The final MSA is progressively built on the second guide tree.
2. A progressive method with random chained guide trees. This strategy was recently re-proposed by Boyce *et al.* (2014).
3. A combination of an iterative refinement method and a progressive method. It is known that the iterative refinement technique (Barton and Sternberg, 1987; Berger and Munson, 1991; Gotoh, 1993) improves the accuracy of an MSA in comparison with the progressive method. However, it is difficult to directly apply the iterative refinement method to large alignment problems. We partially use the iterative refinement technique in this progressive alignment calculation. That is, a relatively accurate core alignment is first built with an iterative refinement option, G-INS-i, and then the remaining sequences are added into the core alignment as discussed in Katoh and Frith (2012).
4. A progressive method with a relatively rigorous guide tree, G-INS-1 (Note that G-INS-1 differs from G-INS-i; the former is a progressive method, and the latter iteratively refines the G-INS-1 alignment). All-to-all pairwise DP calculation is performed to build a guide tree, which is generally more accurate than that in method 1. This method requires unpractical computational resource when applied to large alignment problems consisting of tens of thousands of sequences. The aim of this test is to clarify the effects of the approximations in the practical methods (1–3).
5. For methods 1 and 2, two types of position-specific gap costs were tested: (*i*) An obsolete one (Katoh *et al.*, 2002) that was used in versions $<$7.1 of MAFFT. (*ii*) The current default (Katoh and Standley, 2016) that is used in versions $\geq$7.1 since 2013. For methods 3 and 4, only the current default (*ii*) was used.

The comparison between methods 1 and 2 was already reported in Boyce *et al.* (2014) and Fox et al. (2016), which consistently concluded that random chained guide trees (method 2) outperformed normal guide trees (method 1) for large alignments using various programs including MAFFT. The authors used MAFFT version 7.029, but the accuracy for large MSAs has been improved after this version. Using a newer version (7.294), we reexamined the difference between normal guide trees and random chains, using two different benchmark criteria. For reference, other methods designed for large data, Clustal Omega version 1.2.1 (Sievers *et al.*, 2011) and UPP version 2.0 (Nguyen *et al.*, 2015) with several different options were included into the comparison.

The programs ran on Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80 GHz with 64 GB RAM using a single thread, except for G-INS-1 (method 4), for which heterogeneous computing nodes were used to finish the calculation in a practical amount of time.

2.2 Data

We used three real protein-based datasets, HomFam (Sievers *et al.*, 2011), OXBench (Raghava *et al.*, 2003) and ContTest (Fox *et al.*, 2016), to assess how accurately structurally conserved regions are aligned. Each problem contains a single domain only, which is the most basic type of MSA. In HomFam, we identified reliably aligned regions based on the HOMSTRAD data (Mizuguchi *et al.*, 1998) with structural information and used only these regions for assessment. We also filtered out some entries with repetitive sequences that have multiple solutions (see Supplementary data for details of these modifications). To avoid HOMSTRAD-specific evaluations, we prepared another dataset based on OXBench, by applying the same procedure as HomFam to extend the number of sequences in each entry. The extended version of OXBench is called OXFam hereafter. Each dataset was divided into three subsets, Small ($0 < N \leq 3000$), Medium ($3000 < N \leq 10\,000$) and Large ($10\,000 < N$), where N is the number of sequences in an MSA, following Sievers *et al.* (2011). For HomFam and OXFam, we used the SP and TC criteria to evaluate the accuracy of an estimated MSA, assuming that the reference is correct. SP is the number of correctly aligned pairs in the estimated MSA divided by the number of aligned pairs in the reference, and TC is the number of correctly aligned columns in the estimated MSA divided by the number of aligned columns in the reference. The SP and TC scores were calculated using the FastSP program (Mirarab and Warnow, 2011). The HomFam and OXFam datasets we used here are available at https://mafft.sb.ecei.tohoku.ac.jp/.

3 Results and discussion

The results are shown in Tables 1 and 2.

Table 1. Performance of various options of MAFFT and other methods on HomFam and OXFam

	HomFam				OXFam			
Number of sequences in an entry	Small <3000	Medium 3000–10 000	Large >10 000	All	Small <3000	Medium 3000–10 000	Large >10 000	All
Number of entries	38	32	19	**89**	74	59	32	**165**
Mean SP score								
MAFFT—Normaltree[a]	0.9074	0.8957	0.7795	**0.8759**	0.9300	0.8853	0.8207	**0.8928**
MAFFT—Randomchain[b]	0.8699	0.8671	0.7106	**0.8349**	0.9010	0.8571	0.7674	**0.8594**
MAFFT—Iterative (p=100)[c]	0.9105	0.9004	0.7945	**0.8821**	0.9438	0.9058	0.8364	**0.9094**
MAFFT—Iterative (p=500)[d]	0.9267	0.9167	0.8045	**0.8970**	0.9441	0.9228	0.8391	**0.9161**
MAFFT—Iterative (p=1000)[e]	0.9405	0.9228	0.8159	**0.9075**	0.9533	0.9257	0.8427	**0.9220**
Clustal Omega[f]	0.9148	0.8693	0.6871	**0.8498**	0.9257	0.8735	0.7409	**0.8712**
Clustal Omega–Full[g]	0.9088	0.8806	0.6692	**0.8475**	0.9244	0.8595	0.7440	**0.8662**
Clustal Omega—Randomchain[h]	0.8798	0.8309	–	–	0.8905	0.8477	–	–
UPP—Fast[i]	0.8616	0.8407	0.7700	**0.8345**	0.9327	0.8940	0.7878	**0.8908**
UPP—Default[j]	0.8678	0.8708	0.7956	**0.8535**	0.9415	0.9068	0.8211	**0.9057**
MAFFT— G-INS-1[k]	0.9358	0.9520	0.8844	**0.9306**	0.9572	0.9485	0.8749	**0.9381**
Mean TC score								
MAFFT—Normaltree[a]	0.7806	0.7298	0.5646	**0.7162**	0.9016	0.8253	0.7404	**0.8430**
MAFFT—Randomchain[b]	0.7315	0.6967	0.4932	**0.6681**	0.8622	0.7989	0.6827	**0.8048**
MAFFT—Iterative (p=100)[c]	0.7939	0.7275	0.5943	**0.7274**	0.9143	0.8546	0.7613	**0.8633**
MAFFT—Iterative (p=500)[d]	0.8315	0.7573	0.6148	**0.7586**	0.9155	0.8807	0.7645	**0.8738**
MAFFT—Iterative (p=1000)[e]	0.8628	0.7746	0.6283	**0.7810**	0.9319	0.8897	0.7707	**0.8855**
Clustal Omega[f]	0.8057	0.7152	0.4449	**0.6961**	0.8842	0.8118	0.6408	**0.8111**
Clustal Omega–Full[g]	0.8086	0.7365	0.4386	**0.7037**	0.8886	0.7839	0.6688	**0.8085**
Clustal Omega–Randomchain[h]	0.7580	0.6918	–	–	0.8452	0.7888	–	–
UPP—Fast[i]	0.7466	0.7087	0.5853	**0.6985**	0.9028	0.8535	0.7196	**0.8496**
UPP—Default[j]	0.7492	0.7570	0.6330	**0.7272**	0.9138	0.8676	0.7601	**0.8675**
MAFFT—G-INS-1[k]	0.8549	0.8480	0.7441	**0.8288**	0.9358	0.9147	0.8212	**0.9060**
Total CPU time (min)								
MAFFT—Normaltree[a]	2.9	36	420	**460**	5.5	81	470	**560**
MAFFT—Randomchain[b]	2.0	15	71	**88**	3.4	30	96	**130**
MAFFT—Iterative (p=100)[c]	7.4	61	580	**650**	11	130	660	**800**
MAFFT—Iterative (p=500)[e]	160	390	790	**1300**	190	640	1000	**1900**
MAFFT—Iterative (p=1000)[d]	810	2100	1500	**4400**	1100	3500	2800	**7500**
Clustal Omega[f]	21	160	300	**480**	27	220	590	**840**
Clustal Omega—Full[g]	44	570	5400	**6000**	82	1300	1700	**8400**
Clustal Omega—Randomchain[h]	130	18 000	–	–	170	4600	–	–
UPP—Fast[i]	53	190	260	**500**	92	380	540	**1000**
UPP—Default[j]	360	1600	2400	**4400**	660	3300	4900	**8900**
MAFFT—G-INS-1[k]	(370)	(5200)	(44 000)	(**49 000**)	(760)	(13 000)	(71 000)	(**86 000**)

Commands are: a, `mafft` *input*; b, `mafft --randomchain --randomseed` *seed input*; c–e, `mafft-sparsecore.rb -s` *seed* `-p` *p* `-i` *input*; f, `clustalo -i` *input*; g, `clustalo --full -i` *input*; h, `clustalo --pileup -i` *input* (sequence order randomized); i, `run_upp.py -m amino -B 100 -s` *input*; j, `run_upp.py -m amino -s` *input*; k, `mafft --globalpair --thread 10` *input*; The order of input sequences was randomized in every sequence set. The results of "MAFFT — Randomchain" (b) and "MAFFT — Iterative" (c-e) were averaged for 100 and 10 replications, respectively, with different seeds of random numbers. The results of "UPP" (i, j) were averaged for 10 different runs. For "Clustal Omega — Randomchain" (h), only the results for the Small and Medium subsets are shown, because the calculation of the Large subset did not finish while compiling this manuscript. The CPU time of "MAFFT — G-INS-1" (k) is shown in parentheses as it ran on different computer systems from the others.

3.1 Effect of chained guide trees

The effect of random chained guide trees was clearly negative in the HomFam and OXFam tests. A decrease of 1–7% in benchmark scores due to the use of random chained guide trees was observed in all subsets of both datasets, as shown at the first two lines ("Normaltree" and "Randomchain") in the TC and SP blocks in Table 1. For their overall difference, the *P*-values were estimated to be 0.016 for HomFam-TC and <0.01 for HomFam-SP, OXFam-SP and OXFam-TC by the Wilcoxon signed-rank test. Because this result is sharply inconsistent with the previous report (Boyce *et al.*,

Table 2. Performance of various options of MAFFT and other methods on ContTest

Number of sequences in an entry	Small <3000	Medium 3000–10 000	Large >10 000	All
Number of entries	15	70	51	**136**
Mean ContTest score				
MAFFT—Normaltree[a]	0.4081	0.4874	0.5439	**0.4998**
MAFFT—Randomchain[b]	0.4406	0.5227	0.5997	**0.5425**
MAFFT—Iterative (p=100)[c]	0.3747	0.5005	0.5771	**0.5153**
MAFFT—Iterative (p=500)[d]	0.3883	0.5180	0.6046	**0.5361**
MAFFT—Iterative (p=1000)[e]	0.3808	0.5237	0.6198	**0.5440**
Clustal Omega[f]	0.3039	0.4291	0.4262	**0.4142**
Clustal Omega—Full[g]	0.3080	0.4585	0.4640	**0.4440**
Clustal Omega—Randomchain[h]	0.4328	0.5324	0.5703	**0.5357**
UPP—Fast[i]	0.3515	0.5139	0.5744	**0.5187**
UPP—Default[j]	0.3555	0.5254	0.5936	**0.5323**
MAFFT—G-INS-1[k]	0.3853	0.5445	0.6582	**0.5696**
Total CPU time (min)				
MAFFT—Normaltree[a]	1.2	54	440	**500**
MAFFT—Randomchain[b]	0.56	16	88	**100**
MAFFT—Iterative (p=100)[c]	2.2	77	650	**730**
MAFFT—Iterative (p=500)[d]	30	250	930	**1200**
MAFFT—Iterative (p=1000)[e]	160	1100	2100	**3400**
Clustal Omega[f]	5.0	130	460	**600**
Clustal Omega—Full[g]	16	830	5600	**6400**
Clustal Omega—Randomchain[h]	28	6400	110 000	**120 000**
UPP—Fast[i]	17	230	500	**750**
UPP—Default[j]	130	2000	4600	**6700**
MAFFT—G-INS-1[k]	(110)	(7100)	(48 000)	**(55 000)**

a–k, The same commands were used as Table 1. The input sequence order was not changed from that in the original data, because it seems to be already randomized.

2014), we carefully inspected the cause of the difference. When MAFFT version 7.029 was used, as in their report, an improvement upon use of random chained guide trees was indeed observed. A major difference between these two versions is in the position-specific gap cost as noted below (Section 3.2). By using the old position-specific gap cost, still available as `- -leavegappyregion` in the current version, the tendency they reported (random chained guide trees perform well for the Large subset) was clearly reproduced as shown in Supplementary Table S1. MUSCLE also uses a position-specific gap cost (Edgar, 2004a) similar to old versions of MAFFT. This might explain the fact that the advantage of random chained guide trees was especially observed in these two programs but the tendency was unclear in Clustal Omega (Fig. 5 in Boyce et al., 2014; the "Clustal Omega—Randomchain" lines in Table 1).

On the other hand, in ContTest, the improvement by the use of random chained guide tree was observed in both the current version (0.4998 → 0.5425 in the overall average; Table 2) and the old version (0.3967 → 0.5121; Supplementary Table S2), as well as Clustal Omega (0.4142 → 0.5357; Table 2). The P-values for the difference of these pairs were estimated to be <0.01. Thus, we confirmed Fox *et al.*'s (2016) observation that there are cases where chained guide trees are useful. To satisfy this need, we have implemented the `--pileup` option and the `--randomchain` option in MAFFT. The former aligns the sequences according to the input order, and the latter does in a randomized order.

In addition, we have enabled the `- -mixedlinkage` s option, to control the degree of balance of a guide tree using a parameter s. MAFFT's guide tree is a mixture of an average linkage tree and a minimum linkage tree; when merging clusters L and R into a new cluster P, the distance d_{PC} from P to a third cluster C is:

$$d_{PC} = s\,(d_{LC} + d_{RC})/2 + (1 - s)\min(d_{LC}, d_{RC}). \quad (1)$$

An average linkage tree tends to be a balanced tree, while a minimum linkage tree tends to be an imbalanced tree. Thus, by changing the parameter s, an intermediate tree between a balanced tree and an imbalanced tree can be generated. This tree was also adopted by MUSCLE. The default s value has been unchanged from 0.1 since the initial release of MAFFT and MUSCLE. This means that these programs use a moderately imbalanced guide tree by default. When the most imbalanced option (`- -mixedlinkage 0`) was applied, the result for ContTest was intermediate between the default guide tree and a random chain (Supplementary Table S3), as expected. For HomFam and OXFam, the difference between $s = 1$ and $s = 0$ was small (Supplementary Table S4). Using this option, one can select balanced or imbalanced guide trees if necessary.

3.2 Effect of position-specific gap costs

The advantage of the current gap cost over the previous one was observed with a significance level of 0.01 in HomFam, OXFam and ContTest (Supplementary Tables S5 and S6). The previous position-specific gap cost, $G(,)$, used in MAFFT versions < 7.1 was:

$$G(i,x) = S^{\mathrm{op}}\{(1 - g^{\mathrm{start}}(x)) + (1 - g^{\mathrm{end}}(i))\}/2, \quad (2)$$

where S^{op} is a normal gap cost for sequence-sequence alignment, and $g^{\mathrm{start}}(x)$ and $g^{\mathrm{end}}(i)$ are the frequencies of gaps that start at position x and that end at position i, respectively (Katoh *et al.*, 2002). This was used as default till October 2013. The effect of gap cost depends on match scores for site pairs across two sequences. The match score of a site pair usually decreases with the increase of the gaps that exist in the sites. Relatively to match scores, the gap cost of Equation (2) effectively increases with the increase of gaps in the site pair. Accordingly, in already gap-rich regions, additional gaps are suppressed. This feature is useful when gap-rich regions are out of interest and discarded manually or by a filtering program. Moreover, the resulting MSAs are compact and easy to check visually. Thus, this calculation is still selectable with the `--leavegappyregion` option.

With the increase of sequences included in an MSA, gaps are sometimes inserted even in conserved regions because of sequencing errors or actually exceptional sequences. These gaps make the gap cost of Equation (2) unnecessarily stronger. To avoid this problem, in versions ≥7.1, the gap cost is reduced according to the existence of gaps in a position as described in Supplementary data in Katoh and Standley (2016).

$$G(i,x) = S^{\mathrm{op}}\{(1 - g^{\mathrm{start}}(x))\,f(x) + (1 - g^{\mathrm{end}}(i))\,f(i)\}/2, \quad (3)$$

where $f(x)$ is the frequency of non-gap characters at position x. The resulting alignments naturally have more gaps and are longer than those with Equation (2). The difference in the accuracy between Equations (2) and (3) is indistinguishable in small-scale benchmarks, but was clearly observed in large-scale benchmarks (Supplementary Tables S5 and S6).

3.3 Partial application of iterative refinement

The improvement in the benchmark scores by introduction of iterative refinement (method 3) is shown in the "MAFFT—Iterative" lines in Tables 1 and 2. Unlike the case of random chained guide trees, the improvement was stably and consistently observed for both types of benchmarks. The calculation procedure is: (*i*) The input sequences are sorted by length. From the upper *n*% of the sorted sequences, *p* sequences are randomly selected as "core" sequences. The default values of *n* and *p* are 50 and 500, respectively. (*ii*) An MSA of the *p* core sequences is constructed by an iterative refinement option, G-INS-i. (*iii*) The remaining sequences are added to the core MSA using the `--add` option, which employs the progressive alignment method. This process runs automatically with the `mafft-sparsecore.rb` script that is available in MAFFT version 7.294.

This strategy is similar to the construction process of large MSAs previously provided in Pfam. This is also similar to UPP (Nguyen *et al.*, 2015) in applying an accurate method to randomly-selected sequences. UPP uses PASTA (Mirarab *et al.*, 2015), which is a combination of L-INS-i option of MAFFT, OPAL (Wheeler and Kececioglu, 2007) and FastTree (Price *et al.*, 2010), to align core sequences, and then uses HMMalign (Finn *et al.*, 2011) to add the remaining sequences to the core alignment. Two settings, "Default" (1000 core sequences) and "Fast" (100 core sequences), of UPP were tried. In both UPP and partially iterative options of MAFFT, the accuracy score increased with the number of core sequences. This results in a tradeoff between computational time and the accuracy, because the methods for core alignment are relatively slow. At present, such two-step strategies are practical ones for constructing large MSAs with limited computational resources.

We tested whether this method improves the alignment accuracy of the sequences other than core sequences. For this test, we created an artificial situation where reference sequences are excluded from the core MSA, i.e. the reference sequences (used for assessing the accuracy) are not directly aligned by the iterative refinement method. Even in this situation, the accuracy improvement over the normal progressive method was observed (compare the "Normaltree" and "Iterative" lines in Supplementary Tables S7 and S8). In addition, we tested the effect of the heterogeneity in sequence length by modifying the three datasets so that each MSA has sequences with similar lengths (Supplementary Table S9; see the footnote of this table about the filtering process). The same tendencies were observed as in the original data; "Iterative" outperformed "Normaltree" in the all datasets; "Iterative" also outperformed "Randomchain" in HomFam and OXFam but was comparable to "Randomchain" in ContTest.

The motivation of the length-based restriction, using parameter *n*, is to exclude fragmentary sequences from the core MSA. There is another requirement that the core sequences should represent the entire data without bias. Because such conditions should differ for different cases, we made parameter *n* adjustable, setting the default value to 50%. We tried various *n* and *p* values and expectedly observed a weak tendency that the accuracy was relatively high when *n* was a moderate value (from 30 to 70), for HomFam and OXFam (Supplementary Tables S10 and S11).

In general, an MSA is just one possible solution sampled from a huge number of equally probable ones. Alternative solutions are sometimes useful for assessing site-wise or overall reliability of an MSA, in systematic methods (e.g. Chang *et al.*, 2014; Penn *et al.*, 2010) or by manual inspection. MSAs computed with random sampling can be used as a source of alternative solutions. For this purpose, the `mafft-sparsecore.rb` script has an option, `-s` *seed*, to specify the *seed* of random numbers.

3.4 Application of computationally intensive method

We applied another progressive option, G-INS-1 (method 4), which uses as little approximation as possible, to the same datasets. G-INS-1 uses all-pairwise DP-based alignments to build a normal guide tree and to calculate a COFFEE-like consistency score (Notredame *et al.*, 1998). Requiring a long computational time and huge RAM space, this method is not practical for more than 10 000 sequences, but a clear advantage in accuracy was observed in HomFam and OXFam; the improvement from "Normaltree" to "G-INS-1" in Table 1 was +8–18% and +1–7% for the Large and Small subsets, respectively. The *P*-values for the overall improvement were estimated to be less than 0.01 by the Wilcoxon signed-rank test. In ContTest (Table 2), the advantage of "G-INS-1" over "Normaltree" was observed in the Large (+11%; $P < 0.01$) and Medium (+5%; $P < 0.01$) subsets, but not in the Small subset (−2%; $P = 0.48$). Their overall difference was about +7% ($P < 0.01$). These observations on the two types of benchmarks suggest that the accuracy of practical methods for large MSAs has not yet reached a plateau.

The inclusion of phylogenetic information for building an MSA has been assumed to be useful over 30 years, and this assumption is a basis of the tree-based progressive alignment methods. The surprisingly good performance of chained guide trees (Fox *et al.*, 2016), reproduced in Table 2 in this report, might be over-interpreted to conclude that phylogenetic information is in fact useless or sometimes harmful for large MSAs, depending on the purpose of downstream analyses. In contrast, the present results suggest that rigorous guide trees and known techniques, such as iterative refinement and consistency scores, established for small datasets, are also useful for large MSAs. This study also encourages steady developments of computational and technical improvements for applying these techniques to large-scale data.

In many actual analyses using MSAs, individual sequences in an MSA are not equally important. That is, in downstream analysis or in the interpretation phase, some sequences are focused on but other sequences are just used for estimating the degree of conservation or used as a background. More specifically, in the case of contact prediction, the target sequence is regarded as a sequence to be focused on, while the other sequences are used only indirectly. If such a difference in the role among input sequences is taken into account as early as in the MSA calculation phase, it might improve the accuracy of downstream analyses. We are also preparing a new feature to satisfy such a necessity; sequences to be focused on can be specified in the `--focus` option, which runs in a combination with the G-INS-i or G-INS-1 option. The sequences specified by a keyword, ">_focus_", in the title line are used for the consistency score calculation but the other sequences are not. Based on the same idea, the `mafft-sparsecore.rb` script has an optional feature, in which specific sequences (marked with ">_focus_" in the title line) are always included in the core MSA. We are trying to evaluate their usefulness in actual cases.

The quality of MSA is greatly affected by characteristics of input data, e.g. similarity among input sequences, variation in length, truncated sequences of a single domain or full-length Eukaryotic protein sequences, etc. Thus, a careful preparation of input data is important, in addition to improvements of the calculation method for a given dataset. Systematic methods to prepare input sequences, automatically or interactively, would also be useful.

4 Conclusions

- For constructing large MSAs consisting of several to tens of thousands of sequences, the accuracy of the default option, FFT-NS-2, of MAFFT significantly improved in versions ≥7.1.
- To further improve the accuracy, a partial application of the iterative refinement technique is useful. This calculation is automated by the `mafft-sparsecore.rb` script, available in MAFFT version 7.294 and higher. For this method, there is a tradeoff between the accuracy and the computational time; the speed and accuracy depend on how many sequences were subjected to the iterative refinement calculation.
- The G-INS-1 option gives better results, but it requires unpractical computational resource. It suggests that there is still room for improvement in practical methods for building large MSAs.
- A new option, `--randomchain` uses a random chained guide tree, advocated by Boyce *et al.* (2014). It is fast and sometimes performs well, but its accuracy is not superior to the other two techniques tested here, in the protein structure-based tests. We recommend this option only when phylogenetic consideration is surely unnecessary.

Acknowledgements

We thank Johannes Söding and Daron M. Standley for their comments on applications of MSAs to actual analyses. Computations were partially performed on the NIG supercomputer at ROIS National Institute of Genetics.

Funding

This research was supported by the Platform Project for Supporting in Drug Discovery and Life Science Research (Platform for Drug Discovery, Informatics, and Structural Life Science) from Japan Agency for Medical Research and Development (AMED), and JSPS KAKENHI Grant Number 16K07464.

Conflict of Interest: none declared.

References

Barton,G.J. and Sternberg,M.J. (1987) A strategy for the rapid multiple alignment of protein sequences. confidence levels from tertiary structure comparisons. *J. Mol. Biol.*, **198**, 327–337.

Berger,M.P. and Munson,P.J. (1991) A novel randomized iterative strategy for aligning multiple protein sequences. *Comput. Appl. Biosci.*, **7**, 479–484.

Boyce,K. *et al.* (2014) Simple chained guide trees give high-quality protein multiple sequence alignments. *Proc. Natl Acad. Sci. U.S.A.*, **111**, 10556–10561.

Chang,J.M. *et al.* (2014) TCS: a new multiple sequence alignment reliability measure to estimate alignment accuracy and improve phylogenetic tree reconstruction. *Mol. Biol. Evol.*, **31**, 1625–1637.

Edgar,R.C. (2004a) MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, **5**, 113.

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

Feng,D.F. and Doolittle,R.F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, **25**, 351–360.

Finn,R.D. *et al.* (2011) Hmmer web server: interactive sequence similarity searching. *Nucleic Acids Res.*, **39**, W29–W37.

Fox,G. *et al.* (2016) Using *de novo* protein structure predictions to measure the quality of very large multiple sequence alignments. *Bioinformatics*, **32**, 814–820.

Gotoh,O. (1993) Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Comput. Appl. Biosci.*, **9**, 361–370.

Higgins,D.G. and Sharp,P.M. (1988) CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, **73**, 237–244.

Hogeweg,P. and Hesper,B. (1984) The alignment of sets of sequences and the construction of phyletic trees: an integrated method. *J. Mol. Evol.*, **20**, 175–186.

Kamisetty,H. *et al.* (2013) Assessing the utility of coevolution-based residue-residue contact predictions in a sequence- and structure-rich era. *Proc. Natl Acad. Sci. U.S.A.*, **110**, 15674–15679.

Katoh,K. and Frith,M.C. (2012) Adding unaligned sequences into an existing alignment using MAFFT and LAST. *Bioinformatics*, **28**, 3144–3146.

Katoh,K. and Standley,D.M. (2016) A simple method to control over-alignment in the MAFFT multiple sequence alignment program. *Bioinformatics*, **32**, 1933–1942.

Katoh,K. *et al.* (2002) Mafft: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res.*, **30**, 3059–3066.

Mirarab,S. and Warnow,T. (2011) FastSP: linear time calculation of alignment accuracy. *Bioinformatics*, **27**, 3250–3258.

Mirarab,S. *et al.* (2015) PASTA: ultra-large multiple sequence alignment for nucleotide and amino-acid sequences. *J. Comput. Biol.*, **22**, 377–386.

Mizuguchi,K. *et al.* (1998) HOMSTRAD: a database of protein structure alignments for homologous families. *Protein Sci.*, **7**, 2469–2471.

Nguyen,N.P.D. *et al.* (2015) Ultra-large alignments using phylogeny-aware profiles. *Genome Biol.*, **16**, 124.

Notredame,C. *et al.* (1998) COFFEE: an objective function for multiple sequence alignments. *Bioinformatics*, **14**, 407–422.

Penn,O. *et al.* (2010) An alignment confidence score capturing robustness to guide tree uncertainty. *Mol. Biol. Evol.*, **27**, 1759–1767.

Price,M.N. *et al.* (2010) FastTree 2—approximately maximum-likelihood trees for large alignments. *PLoS One*, **5**, e9490.

Raghava,G.P.S. *et al.* (2003) OXBench: a benchmark for evaluation of protein multiple sequence alignment accuracy. *BMC Bioinformatics*, **4**, 47.

Sievers,F. *et al.* (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol Syst Biol*, **7**, 539.

Sievers,F. *et al.* (2014) Systematic exploration of guide-tree topology effects for small protein alignments. *BMC Bioinformatics*, **15**, 338.

Tan,G. *et al.* (2015) Simple chained guide trees give poorer multiple sequence alignments than inferred trees in simulation and phylogenetic benchmarks. *Proc. Natl Acad. Sci. U.S.A.*, **112**, E99–E100.

Thompson,J.D. *et al.* (1999) A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res.*, **27**, 2682–2690.

Wheeler,T.J. and Kececioglu,J.D. (2007) Multiple alignment by aligning alignments. *Bioinformatics*, **23**, i559–i568.