Supplementary Information

Guide Tree Construction

Standard progressive alignment schemes build up the final alignment by aligning individual sequences and intermediate profiles in a particular order. This order is decided by the guide-tree which is based on pairwise dis/similarities of the individual sequences. Calculation time and memory requirements to construct a full pairwise distance matrix grow quadratically with the number of sequences which is prohibitive for large numbers of sequences.

It was shown in Blackshields et al. (2010) that this quadratic scalability can be improved on, using a scheme called mBed. In the mBed scheme a small number of sequences are selected as seeds. Following Linial et al. (1995) this number is taken to be $\log_2(N)$. The seed selection can be random or follow a systematic strategy. Distances of all sequences are then evaluated with respect to the seed sequences only. Using this emBedding, sequences can be either clustered directly or a full distance matrix can be approximated by computing vector distances.

The original mBed algorithm used those vectors to approximate a full pairwise distance matrix by computing vector distances, which are fast to compute. Here, we instead apply bisecting K-Means to the vectors as to create subclusters of a certain size. Within each cluster 'real' distances can be computed and clusters are later joined. Thereby we avoid the typical memory contraints encountered when computing a pairwise distance matrix needed to construct a guide-tree.

In Clustal Omega we use the k-tuple distance measure of Wilbur & Lipman (1983) as employed in ClustalW 1.83 and ClustalW2 (Larkin et al., 2007), to construct the distance matrix. If the sequences are already aligned (and are to be re-aligned during iteration, for example) then Kimura-corrected pairwise aligned identities (Kimura, 1983) are used. The vector distances in the mBed matrix are then used to cluster the sequences into subclusters using a bisecting k-means approach, which allows full control over the maximum cluster size and thereby avoids the usual memory limitations. For each subcluster a full distance matrix is constructed. As the size of the clusters affects the execution time, we have limited the maximum size of clusters. Currently, this limit is set to 100. All pairwise distances and a UPGMA sub-tree are calculated for each cluster. At this stage we use the fast UPGMA code as implemented in MUSCLE (Edgar, 2004). The sub-trees for the individual clusters are then linked up by an overarching tree built from the distance matrix of the clusters' barycentres using vector distances.

Clustal Omega can output the guide-tree used for the alignment, as well as input pre-computed guide-trees. Depending on the size of the alignment, re-cycling guide-trees presents a considerable saving in computation time. It is also possible to turn off the mBed mode and switch to full distance matrix calculation. In this particular mode it is possible to output the distance matrix. This is not possible in default (mBed) mode, as a full distance matrix is never calculated.

Profile HMM Alignment

For the actual sequence/profile alignment we use an adapted version of the HHalign package of Söding (2005). Here sequences/profiles are converted into profile HMMs which describe transition, insertion, and deletion probabilities, as well as emission probabilities for residues. Both kinds of probabilities are then augmented with pseudo-counts. HMMs for the two sequences/profiles are then aligned. If the sequences/profiles are short enough, such that the dynamic programming matrices fit into the computer's memory, then a Maximum Accuracy (MAC) algorithm, as proposed by Holmes & Durbin (1998) and extended to local pairwise alignment of profile HMMs in Biegert & Soeding (2007), is used. A computer with 2GB RAM can accommodate, for example, 2 sequences/profiles of over 6,500 residues in length, each.

For problems that exceed the available memory, the Viterbi algorithm is used. HHalign only aligns the overlapping core region of both HMMs and truncates the un-alignable end sections corresponding to end-gaps in the corresponding HMM. For the final alignment the aligned HMMs are reconverted into profiles and the truncated end sections are re-attached.

External Profile Alignment

External Profile Alignment or EPA involves using a profile HMM which describes an alignment of a set of sequences that are homologous to the input set. This is read from a file in HMMER2 or HMMER3 format. Clustal Omega takes this external HMM and aligns to it in turn the two sequences/profiles to be aligned to each other at each stage in progressive alignment. After alignment to the HMM, pseudo-count information for the residue frequencies is transferred from the external HMM to the HMMs describing the sequences/profiles, position by position. We do not transfer pseudo-count information for the transition probabilities. Finally, these 'softened-up' HMMs, representing the 2 sequences/profiles, are aligned using the MAC or Viterbi algorithm. EPA information is particularly useful during the early stages of progressive alignment. During the later alignment stages, the intermediate profiles have already accumulated enough information, so that pseudo-count transfer becomes less necessary. Clustal Omega therefore scales the contribution of the external HMM during progressive alignment. Pseudo-count transfer to profiles comprising of more than ten sequences is negligible. Each sequence/profile has to be pre-aligned to the external HMM. This represents a computational overhead of at most twice the original time required for aligning two sequences/profiles, so the final alignment time (not counting guide-tree construction) may triple.

Iteration

Iteration is done by iterating the alignment and, optionally, also the guide tree. The alignment itself is iterated using EPA while the guide tree is simply re-calculated from the full alignment produced in an earlier iteration. Initially, Clustal Omega aligns the input sequences. This alignment is then converted into a HMM, which can then be used for EPA as explained above. At the same time, the initial alignment can be used to construct a new guide-tree. At this stage Kimura-corrected (Kimura, 1983) pairwise aligned identities are used, rather than k-tuple distances. The guide-tree is, by default, still constructed using the mBed scheme. This can be switched to a full distance matrix. Using aligned sequences as input is equivalent to performing one iteration as the initial alignment is converted into a HMM and Kimura distances are used to construct the guide-tree. Iteration can be performed more than once, and HMM iteration can be decoupled from guide-tree iteration, that is, the background HMM can be frozen while the guide-tree is still being iterated, or vice versa. All of these options are set by users on the command-line of the program.

The effects of HMM iteration are illustrated in Supplementary Figure 1. Here the lower part of the figure displays the guide tree used for aligning the 6 sequences of BB12021, which is part of BAliBASE 3 (see below). The upper part of the figure shows an alignment generated with Clustal Omega, using default settings. This alignment achieves a Total Column score of 0.820 with respect to the BAliBASE 3 reference alignment. The middle part of the figure shows an alignment of the same sequences using one HMM iteration and no guide-tree iteration; this is achieved by setting --iter=1 --max-guidetree-iterations=0 on the command-line. This alignment has a TC score of 0.890. The alignments differ subtly in two places. According to the guide-tree, 1hpi_ and 2hip_A are to be aligned first. A locally desirable alignment is to match up the 6 residues at alignment positions 5-10. This achieves 3 perfect matches (E,E,D) while opening up a gap of length 3 in 2hip_A. The aligner does not have any foresight as to what other residues will align at these positions later during the alignment process. However, after the initial alignment an overall (background) residue distribution is available as a HMM. This information can be used to guide a re-alignment, by aligning the original sequences (1hpi_ and 2hip_A in this case) to the background HMM, transferring the HMMs pseudocounts, position for position to the original sequences and then aligning the ‘softened-up’ sequences. In this example the background ‘convinces’ 2hip_A to close the internal gap and move the 6-residue block by 3 positions to the right. A similar move of residues can be seen around positions 31-36. Both moves give rise to an improvement in the alignment’s TC score of 7%.

Benchmarks

In order to assess the performance and quality of Clustal Omega alignments, we compared it to various other alignment engines using three different benchmarks. The set of tested alignment programs is:

* ClustalW2, v2.1 [http://www.clustal.org]

* Clustal Omega v1.0.2 [http://www.clustal.org]

* DIALIGN 2.2.1 [http://dialign.gobics.de/]

* FSA 1.15.5 [http://sourceforge.net/projects/fsa/]

* Kalign 2.04 [http://msa.sbc.su.se/cgi-bin/msa.cgi]

* MAFFT 6.857 [http://mafft.cbrc.jp/alignment/software/source.html]

* MSAProbs 0.9.4 [http://sourceforge.net/projects/msaprobs/files/]

* MUSCLE version 3.8.31 posted 1st May 2010 [http://www.drive5.com/muscle/downloads.htm]

* PRANK v.100802, 02-Aug-2010 [http://www.ebi.ac.uk/goldman-srv/prank/src/prank/]

* Probalign v1.4 [http://cs.njit.edu/usman/probalign/]

* PROBCONS version 1.12 [http://probcons.stanford.edu/download.html]

* T-Coffee Version 8.99
[http://www.tcoffee.org/Projects_home_page/t_coffee_home_page.html#DOWNLOAD]

As benchmarks we use BAliBASE 3 (Thompson et al., 2005), PREFAB 4.0 (as posted, March 2005) (Edgar, 2004) and a newly constructed data set (HomFam) using sequences from Pfam (version 25) and Homstrad (as of 2011-06-13) (Mizuguchi et al., 1998).

The BAliBASE benchmark is comprised of 218 families, grouped into 38+44 families exhibiting variability and length (BB11+BB12), 41 families containing orphan sequences (BB2), 30 families with sub-families (BB3), 49 families with extensions (BB4) and 16 families with insertions (BB5). The smallest BAliBASE alignments are comprised of 4 sequences; the largest of 142 sequences; the average is 28.5. The longest sequence in BAliBASE is 7923 residues long. Reference alignments vary between 52 and 8481 positions. The alignments are scored using the bali_score program [http://bips.u-strasbg.fr/fr/Products/Databases/BAliBASE/bali_score.c], which considers only core regions of the alignments. We also used Qscore (see below) to score the core regions of the alignments. The numerical values of scores were slightly different but not different enough to change the relative ranking of the different alignment algorithms' accuracy. We only quote the Total Column Score, giving the number of columns exactly shared by the test and reference alignments.

The PREFAB benchmark is comprised of 1682 families. All families consist of 50 test sequences. The alignments are scored by comparing to reference alignments of two sequences. The entire benchmark set can be grouped according to the pairwise identities of the reference sequences. There are 912 families with 0%-20% pairwise identity of reference sequences, 563 families in the range 20%-40%, 117 families in the range 40%-70% and 90 families in the range 70%-100%. Reference alignments vary in length from 54 to 1047 positions. The longest sequence in PREFAB is 1132 residues long (which is not a reference sequence). Alignments are scored using the Qscore program [http://www.drive5.com/qscore/], and we also only quote the Total Column Score.

For the results in Table 1 and Table 2 we evaluated the above mentioned 12 alignment programs using BAliBASE and PREFAB. For these two benchmarks, all programs are run with default command-line arguments, with the exception of MAFFT, where we used the default and the auto mode. All programs are run single threaded, total times are therefore single core times.

Both benchmark sets, BAliBASE and PREFAB, are comprised of families consisting of at most 142 sequences or 50 sequences, respectively. Neither of which is sufficient to explore scalability to large numbers of sequences. We therefore created a third benchmark set of testcases called HomFam, with very large numbers of sequences. For this we blended all Homstrad (Miziguchi, et al., 1998) families with more than 5 reference sequences with their corresponding Pfam families, if there was a one-to-one link, similar to the previously used approach in (Blackshields et al., 2010). Then we generate a test alignment using the full set of sequences, including the small number from Homstrad. The alignment of the Homstrad sequences in the test alignment is compared to the alignment in the Homstrad structure based alignment using Qscore. The largest Homstrad (reference) family in this third benchmark set is comprised of 41 reference sequences. Alignments vary in length from 39 to 938 positions. The HomFam dataset can be downloaded from http://www.clustal.org/omega/homfam.tgz

There are 94 HomFam families, which we grouped into 41 families with less than 3,000 (93-2,957) sequences, 20 families with more than 10,000 (10,099-93,681) sequences and 33 families in the intermediate sequence number range (3,127-9,105). We settled on a lower cut-off of 3,000 because default MUSCLE performed unacceptably slowly in our test setting for most families with more than 3,000 sequences. Default MAFFT cannot deal with more than 20,000 sequences. However, there are only 8 families with more than 20,000 sequences in the HomFam benchmark. This would have been too small a grouping, so we decided on a cut-off of 10,000, which then split the benchmark more evenly. For scoring the alignments we used the Qscore program, scoring the entire alignment. Only the embedded Homstrad reference sequences were used for scoring the entire alignment.

Given the computational demands of the HomFam benchmark, we could only evaluate four programs. These were: Clustal Omega, Kalign, MAFFT and MUSCLE. For the sub-group of small families (< 3,000 sequences) all programs were run using default settings. For the sub-group of intermediate families (3,000-10,000 sequences) all programs were run using default settings, except for MUSCLE, where -maxiters was set to 2. For the sub-group of large families (> 10,000 sequences) Clustal Omega and Kalign were run with default settings, MUSCLE was run with -maxiters set to 2 and MAFFT was run using the --parttree flag. Three families contained 'U' residues; where this occurred the --anysymbol flag was used in MAFFT. On our machine (Intel(R) Xeon(R) X5470@3.33GHz, 8 core, 46.8GB RAM) MUSCLE and Kalign did terminate the largest two runs with a signal 11 (segmentation violation).
These are the CCHH zinc-finger proteins (Homstrad ID: zf-cchh and Pfam accession: PF00096) with 88,345 sequences and 16 known structures and the retroviral proteases (Homstrad ID: rvp and Pfam accession: PF00077) with 93,681 sequences and 6 known structures. MAFFT --parttree did better on the zinc finger proteins (column score 0.457 vs. 0.143) but Clustal Omega did better on the proteases (column score 0.405 vs. 0.324). Comparison of overall execution time and average accuracy for the large families in Table 3 is therefore given for 18 families (instead of 20) only. The results for the 2 largest families for MAFFT/PartTree and Clustal Omega are included in Figure 1.

Results for the BAliBASE benchmark can be found in Table 1. Supplementary Figure 2 is a graphical representation of the summary results of Table 1. There we plot the average total column score against the total run time (to align all 218 families). Programs using the consistency principle are depicted as blue dots, progressive aligners as red dots, and programs that fit neither category as purple dots.

We have added two programs to Supplementary Figure 2 which are not listed in Table 1. These are:

* Opal v2.0.0 http://opal.cs.arizona.edu (Wheeler et al., 2007)

* SATe v1.4.0 http://phylo.bio.ku.edu/software/sate/sate.html (Liu et al., 2009)

The reason for not including SATe in the main result section was that it was difficult to decide on a convergence criterion that fitted into the current benchmarking scheme of assessing quality as well as performance. SATe's default convergence criterion is to take the best solution arrived at after 24 hours. Not only would this take much longer to evaluate than all the other aligners, it also does not scale with the size of the problem. Therefore we tried to find a criterion that would make SATe finish within a similar time frame as the other aligners. We used the --iter-without-imp-limit flag which specifies the maximum number of iterations without an improvement in score that the SATe algorithm will run. If the number is less than 1, then no iteration limit will be used. Setting --iter-without-imp-limit=1 already took approximately 350,000 seconds. While selecting a higher value for --iter-without-imp-limit might improve on the accuracy of the results (0.539) it would also increase the run time beyond the limits of this study. In default mode SATe uses MAFFT (v6.717) for aligning and merging sub-problems and RAxML (v7.2.6) to infer trees for sub-problems.

Unlike the programs in the main result section, which are compiled C/C++ codes, Opal is written in Java. Nevertheless its performance and accuracy are respectable (15,250 seconds to achieve a TC score of 0.534).

Scalability

The scaling of MAFFT and Clustal Omega with large numbers of sequences is explored in more detail in Supplementary Figure 3 using the biggest family in Pfam as an example. These are the bacterial ABC transporters which have over 160,000 sequences in Pfam and almost 200,000 sequences in the NCBI sequence databases. Subsets of different numbers of sequences are aligned and the times and memory usage and benchmark accuracy are plotted. In Figure 3a the run times of the methods can be seen to scale linearly on log-log plots over the full range of test case sizes with MAFFT up to 10 times faster than Clustal. In Figure 3b, the peak memory usage is given. These are similar over most of the range with MAFFT tending to use more peak memory at high N. In Figure 3c, the average memory usage is given. Here Clustal uses, on average, about a tenth of the memory of MAFFT. Finally, the accuracy of the alignment of the six included Homstrad sequences is plotted against N. Here both methods tend to give lower and lower quality alignments as more sequences are added but with Clustal Omega almost 20% more accurate over the entire range. Overall, this figure shows that MAFFT, with the PartTree option is indeed very fast and scales extremely well with large N but at the expense of memory usage and accuracy. Both methods suffer at high N. This runs counter to the expectation in the field that increasing the number of sequences will increase alignment accuracy. Accuracy will probably increase if you add in extra sequences with very small N say in the range of 10 to 100 sequences and when combined with a consistency based method such as T-Coffee. At very large N however, it looks like, adding in more and more sequences simply makes the problem

more demanding. There is more and more noise added due to sequence errors and outliers and the computational landscape becomes enormously more complicated. This can be compensated for in two ways as described in the main section: 1) by alignment to an external profile and 2) by iteration.

Software

Clustal Omega is licensed under the GNU Lesser General Public License. Source code as well as precompiled binaries for Linux, FreeBSD, Windows, and Mac (Intel and PowerPC) are available at http://www.clustal.org. Clustal Omega is available as a command line program only, which uses GNU-style command line options, and also accepts ClustalW-style command options for backwards compatibility and easy integration into existing pipelines.

Clustal Omega is written in C and C++ and makes use of a number of excellent free software packages, which are listed in the following: A modified version of Sean Eddy's Squid library (http://selab.janelia.org/software.html) is used for sequence I/O, allowing the use of a wide variety of file formats. We use David Arthur's k-means++ code (Arthur & Vassilvitskii, 2007) for fast clustering of sequence vectors. Code for fast UPGMA and guide tree handling routines was adopted from MUSCLE (Edgar, 2006). We use the OpenMP library to enable multithreaded computation of distance matrices and alignment match states. The documentation for Clustal Omega's API is part of the source code, and in addition is available from http://www.clustal.org/omega/clustalo-api/.

Figures

Supplementary Figure 1: Effect of HMM iteration. (Top) alignment of BB12021 using default Clustal Omega. (Middle) alignment of BB12021 using Clustal Omega and one HMM iteration. (Bottom) guide tree for aligning BB12021.

Supplementary Figure 2: Average Total Column scores plotted against total time for BAliBASE 3. X-axis logarithmic, y-axis linear. Consistency based methods depicted as blue dots, progressive methods as red dots, other methods in purple.

Supplementary Figure 3: Computational demands and accuracy results for the largest HomFam family: the bacterial ABC transporters. These include 6 Homstrad sequences with known structures and 194,587 sequences from the NCBI protein sequence databases. Clustal Omega results are plotted in red and MAFFT --PartTree results in blue. All axes use logarithmic scales except the vertical axis in panel (d) which uses a linear scale. (a) Single-threaded run time plotted against number of sequences. (b) Peak memory usage plotted against number of sequences. (c) Average memory usage plotted against number of sequences. (d) Total Column score of the 6 embedded Homstrad reference sequences, plotted against total number of test sequences.

References

Arthur, D., Vassilvitskii, S. (2007). *k*-means++: the advantages of careful seeding. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. pp. 1027–1035.

Biegert, A., Söding, J., (2008) De novo identification of highly diverged protein repeats by probabilistic consistency, Bioinformatics 24 (6), 807-814

Blackshields, G., Sievers, F., Shi, W., Wilm, A., and Higgins, D. G. (2010). Sequence emBedding for fast construction of guide trees for multiple sequence alignment. Algorithms Mol Biol 5, 21.

Holmes, I., and Durbin, R. (1998). Dynamic programming alignment accuracy. J. Comput. Biol 5, 493-504.

Kimura, M. (1983). The Neutral Theory of Molecular Evolution, page 75. Cambridge University Press, Cambridge, UK.

Linial, N., London, E., and Rabinovich, Y. (1995). The geometry of graphs and some of its algorithmic applications. Combinatorica 15, 215-245.

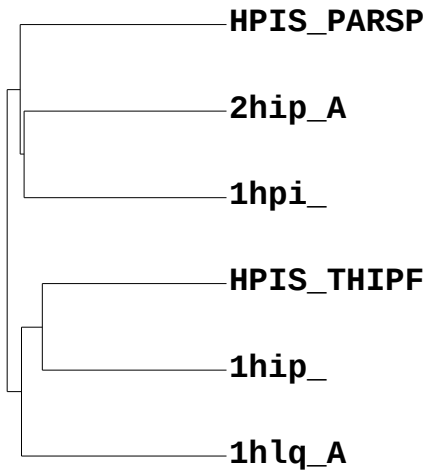
Liu, K., S. Raghavan, S. Nelesen, C. R. Linder, T. Warnow (2009). Rapid and accurate large scale coestimation of sequence alignments and phylogenetic trees. Science, vol. 324, no. 5934, pp. 1561-1564.

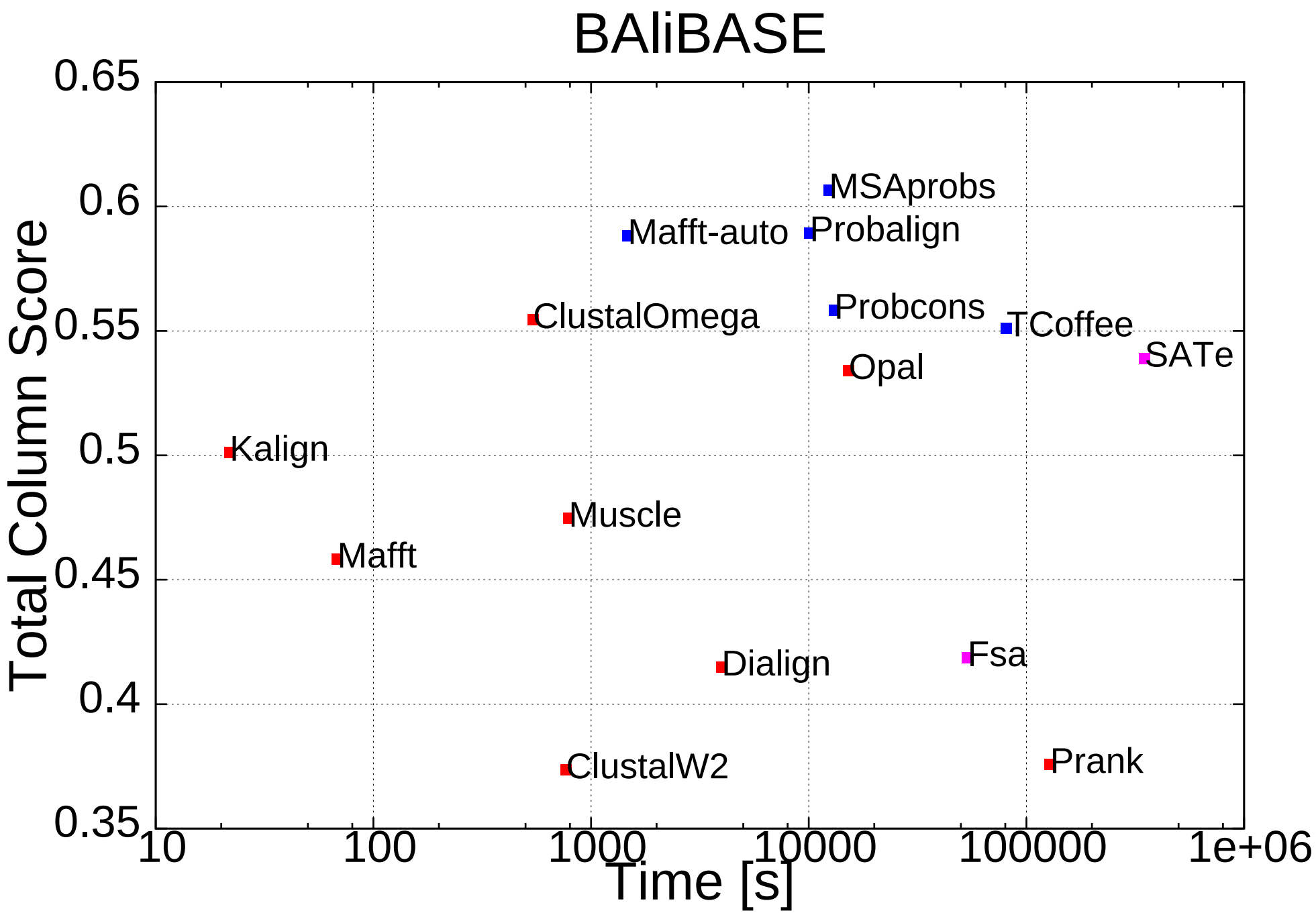
Wheeler, T.J. and Kececioglu, J.D. (2007). Multiple alignment by aligning alignments, Proceedings of the 15th ISCB Conference on Intelligent Systems for Molecular Biology, Bioinformatics 23, i559-i568.

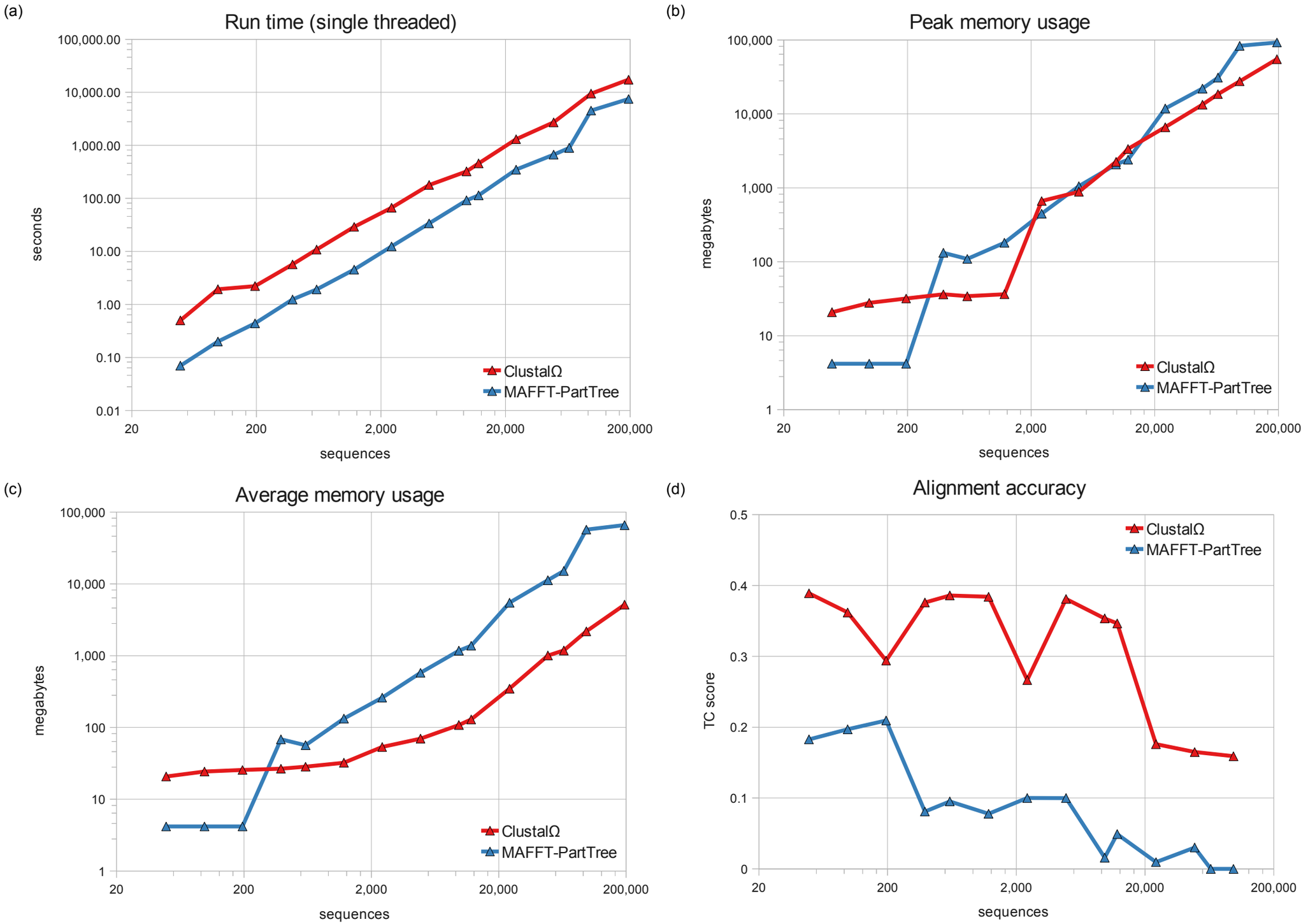
Wilbur, W. J., and Lipman, D. J. (1983). Rapid similarity searches of nucleic acid and protein data banks. Proceedings of the National Academy of Sciences 80, 726 -730.

```
                     .:  *  .  :                .:*  *:              *   :   :  ***  :.
1hlq_A --AAPLVAETDANAKSLGYVADTTKADKTKYP---KHTKDQSCSTCALYQGKT----APQGACPL--FAGKEVVAKGWCSAWAKKA-    75
 1hip_ SAPANAVAADNATAIALKYNQDATKSERVAAARPGLPPEEQHCADCQFMQADAAGATDEWKGCQL--FPGKLINVNGWCASWTLKAG    85
HPIS_THIPF EDLPHVDAATNPIAQSLHYIEDANASERNPVTKTELPGSEQFCHNCSFIQADS----GAWRPCTL--YPGYTVSEDGWCLSWAHKTA    81
 1hpi_ ---MERLSEDDPAAQALEYR-HDASSVQHP-----AYEEGQTCLNCLLYTDASA---QDWGPCSV--FPGKLVSANGWCTAWVAR--    71
 2hip_A ----EPRAED---GHAHDYVNEAADASGHP-----RYQEGQLCENCAFWGEAVQ---DGWGRCTHPDFDEVLVKAEGWCSVYAPAS-    71
HPIS_PARSP -QDLPPLDPSAEQAQALNYVKDTAEAADHP-----AHQEGEQCDNCMFFQADSQ-------GCQL--FPQNSVEPAGWCQSWTAQN-    71
     ruler 1.......10.........20.........30.........40.........50.........60.........70.........80.......
```

```
                     .:  *  ::  :                .:*  *:              *   :   :  ***  :.
1hlq_A --AAPLVAETDANAKSLGYVADTTKADKTKYPK---HTKDQSCSTCALYQGKT----APQGACPL--FAGKEVVAKGWCSAWAKKA-    75
 1hip_ SAPANAVAADNATAIALKYNQDATKSERVAAARPGLPPEEQHCADCQFMQADAAGATDEWKGCQL--FPGKLINVNGWCASWTLKAG    85
HPIS_THIPF EDLPHVDAATNPIAQSLHYIEDANASERNPVTKTELPGSEQFCHNCSFIQADS----GAWRPCTL--YPGYTVSEDGWCLSWAHKTA    81
 1hpi_ ---MERLSEDDPAAQALEYRHDA-SSVQHPA-----YEEGQTCLNCLLYTDASA---QDWGPCSV--FPGKLVSANGWCTAWVAR--    71
 2hip_A -------EPRAEDGHAHDYVNEAADASGHPR-----YQEGQLCENCAFWGEAVQ---DGWGRCTHPDFDEVLVKAEGWCSVYAPAS-    71
HPIS_PARSP -QDLPPLDPSAEQAQALNYVKDTAEAADHPA-----HQEGEQCDNCMFFQADSQ-------GCQL--FPQNSVEPAGWCQSWTAQN-    71
     ruler 1.......10.........20.........30.........40.........50.........60.........70.........80.......
```




BAliBASE
Total Column Score
Time [s]
0.65
0.6
0.55
0.5
0.45
0.4
0.35
10
100
1000
10000
100000
1e+06
MSAprobs
Mafft-auto
Probalign
ClustalOmega
Probcons
TCoffee
SATe
Opal
Kalign
Muscle
Mafft
Dialign
Fsa
ClustalW2
Prank


(a)
Run time (single threaded)
seconds
100,000.00
10,000.00
1,000.00
100.00
10.00
1.00
0.10
0.01
20
200
2,000
20,000
200,000
sequences
ClustalΩ
MAFFT-PartTree
(b)
Peak memory usage
megabytes
100,000
10,000
1,000
100
10
1
20
200
2,000
20,000
200,000
sequences
ClustalΩ
MAFFT-PartTree
(c)
Average memory usage
megabytes
100,000
10,000
1,000
100
10
1
20
200
2,000
20,000
200,000
sequences
ClustalΩ
MAFFT-PartTree
(d)
Alignment accuracy
TC score
0.5
0.4
0.3
0.2
0.1
0
20
200
2,000
20,000
200,000
sequences
ClustalΩ
MAFFT-PartTree