J Supercomput (2015) 71:1833–1845
DOI 10.1007/s11227-014-1362-z

Recovering accuracy methods for scalable consistency library

Jordi Lladós · Fernando Guirado · Fernando Cores · Josep Lluís Lérida · Cedric Notredame

Published online: 31 December 2014

Abstract Multiple sequence alignment (MSA) is crucial for high-throughput next generation sequencing applications. Large-scale alignments with thousands of sequences are necessary for these applications. However, the quality of the alignment of current MSA tools decreases sharply when the number of sequences grows to several thousand. This accuracy degradation can be mitigated using global consistency information as in the T-Coffee MSA-Tool, which implements a consistency library. However, consistency-based methods do not scale well because of the computational resources required to calculate and store the consistency information, which grows quadratically. In this paper, we propose an alternative method for building the consistency-library. To allow unlimited scalability, consistency information must be discarded to avoid exceeding the environment memory. Our first approach deals with the memory limitation by identifying the most important entries, which provide better consistency. This method is able to achieve scalability, although there is a negative impact on accuracy. The second proposal, aims to reduce this degradation of accuracy, with three different methods presented to attain a better alignment.

J. Lladós (✉) · F. Guirado · F. Cores · J. L. Lérida
Department of Computer Science, Universitat de Lleida, Lleida, Spain
e-mail: jordi.llados@diei.udl.cat

F. Guirado
e-mail: f.guirado@diei.udl.cat

F. Cores
e-mail: fcores@diei.udl.cat

J. L. Lérida
e-mail: jlerida@diei.udl.cat

C. Notredame
Comparative Bioinformatics Group, Center for Genomic Regulation, Barcelona, Spain
e-mail: cedric.notredame@crg.es

Keywords Large-scale alignments · Scalability · Consistency · Accuracy · T-Coffee · Multiple sequence alignment

1 Introduction

Multiple sequence alignment (MSA) is a key-tool in several bioinformatic applications like protein/RNA structure prediction, phylogenetic analysis or pattern recognition. The main idea behind MSA is to put protein residues in the same column according to a selected criteria. These criteria can be the structural, evolutionary, functional or sequence similarities.

With the advent of new high-throughput next generation sequencing technologies, the volume of genetic data processed has increased significantly and has provided coverage for a wide range of novel applications in biology and medicine. It will be essential for these applications to achieve large-scale alignments with thousands of sequences or even whole genomes. However, only a few MSA tools are able to align these large datasets. Moreover, all current MSA tools have exhibited scalability issues when the number of sequences increases. Among these problems we can highlight the inability to align so many sequences (lack of sufficient computational resources), the need for prohibitive execution times or a significant degradation in accuracy. Any of these problems represent an insuperable barrier for developing new-generation applications.

Among these methods, ClustalW [4] is the most popular because of its fast results. However, its main drawback is the poor accuracy obtained. There are other methods that provide better alignments. These methods take into account information, known as consistency, about the alignment of all sequences and use it to avoid mistakes in the next alignment steps. Consistency-based methods have proven to be able to deliver more accurate solutions. Example of such methods are T-Coffee [7], Probcons [2] and Probalign. The higher accuracy obtained with these methods comes at the expense of huge computational resources required to calculate and store the consistency. Because these requirements, the scalability of these methods is rather limited.

In the literature, it is possible to find new MSA methods developed with the idea of aligning large sequence datasets. The first of the classic aligners that took scalability into consideration was Mafft [3]. From version 6, Mafft introduced the PartTree technique to speed up the guide-tree building algorithm. PartTree is a divisive clustering algorithm that groups the input sequences around n representative ones. ClustalΩ [11] is capable of aligning any number of protein sequences quickly, also delivering good accuracy. Its scalability key comes from a guide-tree building method, based on the mBed [1] algorithm, which reduces its complexity to $O(N \log N)$, N being the number of sequences. Saté-II [6] is a new version of the original SATé algorithm that improves both its speed and accuracy. Saté-II is a divide-and-conquer iterative meta-method that is applied to any existing external MSA method.

In spite of such improvements in the scalability/performance of these methods, some recent studies have shown that all of the main MSA packages only obtain good accuracy when they are applied to small-medium datasets (10–1,000 sequences). When the number of sequences grows to several thousand, the quality of the alignment decreases

steadily [10]. Other studies [5,13], focusing on phylogeny estimation from nucleotide datasets, have confirmed this hypothesis. The authors do not find any method able to obtain good alignment accuracy when large datasets are used or when indels are present.

The reason for this fall-off in accuracy is degradation due to the accumulation of noise and alignment errors when sequences are added to the alignment process [10]. The more sequences aligned, the more the errors that are introduced and worse is the accuracy. These errors cannot be reversed, but can be mitigated using any refinement stage in the alignment process or using global consistency information, as in T-Coffee. Therefore, it is essential to redesign the alignment tools to take into consideration scalability and its impact on the efficiency and quality of the proposed solution.

In this article, we focus on consistency-based MSA methods. Consistency can overcome the accuracy limits caused by the greediness problems of progressive and iterative aligners. We believe that consistency is crucial for maintaining accuracy in large-scale alignments. However, there is a critical issue because it is not possible to scale because the large amount of computational resources required to calculate and store the consistency information. Therefore, in this paper the authors use the MSA consistency-based tool T-Coffee [7] in large-scale alignments, and present an innovative solution to reduce the amount of memory needed to store the consistency.

Clearly, to reduce the amount of memory necessary to maintain the consistency information, it is essential to decide what consistency information will be used in the alignment and what can be discarded. This choice must consider both the limited resources and maximize the relevance of the stored consistency data to provide the highest alignment accuracy. Furthermore, when some consistency data is discarded, it can produce some gaps depending on the aggressiveness of the limitation. In order to avoid this problem, three new methods are proposed to refill the gaps in the progressive alignment stage. In the present paper, we propose three new recovery methods to diminish the impact on the final accuracy of reducing the consistency data.

The remaining sections are organized as follows. In Sect. 2 we analyse T-Coffee scalability and how to enhance it. In Sect. 3, we present the design for a scalable consistency method. In Sect. 4, several methods to recover accuracy with a consistency-based MSA tool are presented. In Sect. 5, experimentation is performed to evaluate the effectiveness of an aligner based on the new scheme. The main conclusions are outlined in Sect. 6.

2 T-Coffee MSA tool

T-Coffee [7] is an MSA tool that combines the consistency-based scoring function COFFEE [8] with the progressive alignment algorithm. T-Coffee introduces a consistency library generated by a mixture of pair-wise alignments that reduces greediness and then increases accuracy.

T-Coffee is divided into three main stages, as can be seen in Fig. 1:

1. *Primary library* The primary library contains a set of pairwise alignments from among all the sequences to be aligned. The library is built from all against all pairwise alignments computed with a pair of Hidden Markov Models algorithms.

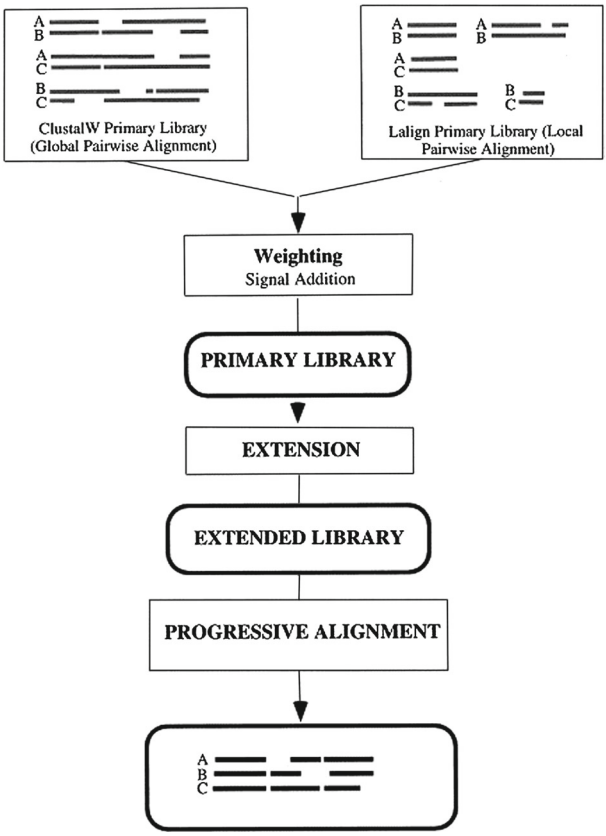

Fig. 1 T-Coffee algorithm stages

In the library, each alignment is represented as a list of pairwise residue matches, named constraint. A sequence identity weight is assigned to each pair of aligned residues to reflect the correctness of a constraint. This stage is the most time and memory consuming.

2. *Extended library* The extension of the library is a re-weighting process where the new weights for a given pair of sequences adds information from the rest of sequences in the set. The library extension is performed on-the-fly during the progressive alignment stage using only the related sequences.
3. *Progressive alignment strategy* The MSA is based on the successive construction of pair-wise alignments. It starts by aligning the two most closely related sequences, and then adds sequences in the order defined by a guide tree. The guide tree is generated using a distance matrix obtained by all-against-all pairwise alignments. In T-Coffee, the alignments are performed maximizing the COFFEE objective function that uses the weights in the extended library instead of the traditional substitution matrix weights and gap penalties that other MSA tools use.

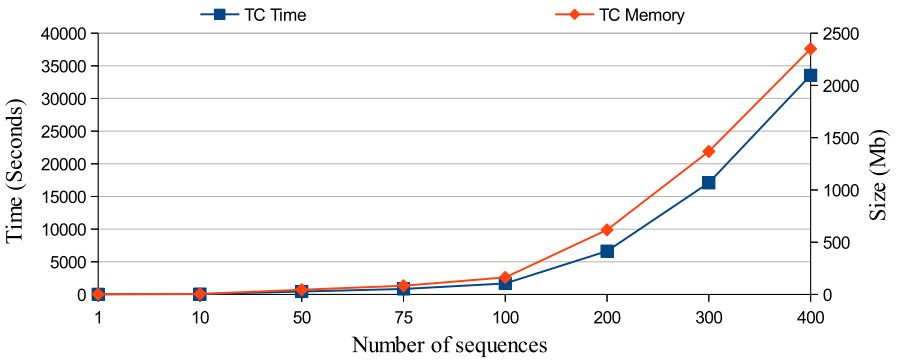

Fig. 2 T-Coffee scalability (execution time and memory requirements)

2.1 T-Coffee scalability

T-Coffee provides an improvement in accuracy over most methods based on a progressive strategy. However, the introduction of these improvements has penalized TC in speed when it is compared with the most commonly-used alternatives. This is due to the computational cost needed to obtain the consistency library and also the huge amount of memory required to be stored.

If we analyze the performance of T-Coffee as the number of sequences increases (Fig. 2), we can see that the execution time and memory requirements grow quadratically. We can note that T-Coffee only takes up to one minute to align 100 sequences, but when the number of sequences increases to 400, more than 8 h are needed to perform the alignment, 380 times longer. With regard to the memory requirements, aligning 100 sequences in the library requires 129 MBs, while for 400 sequences, more than 2 GBs are needed. This study was carried out with the same data and hardware used in Sect. 5.2 of the paper.

These execution times and memory requirements turn T-Coffee into a non-scalable method that is incapable of aligning large numbers of sequences without saturating the computer resources. Thus, in order to improve the scalability of T-Coffee, it is necessary to reduce its memory requirements, which will lead to a reduction of execution time.

T-Coffee implements different algorithms for the progressive alignment step. Thus, the user can adjust the T-Coffee tool to obtain the best result. Figure 3 shows the accuracy obtained when using the default dynamic programming method in T-Coffee, which has been optimized for the execution time point of view, compared with the standard method in Myers & Miller. As can be observed, the accuracy for the optimized algorithm falls when the number of sequences increases instead of the Myers & Miller method that can maintain better accuracy results. In this work all the experimentation will be based on Myers & Miller.

2.2 T-Coffee optimized library

To alleviate the T-Coffee scalability problems, we presented a library-generation optimization method in a previous work [9]. This was the Optimized Library Method

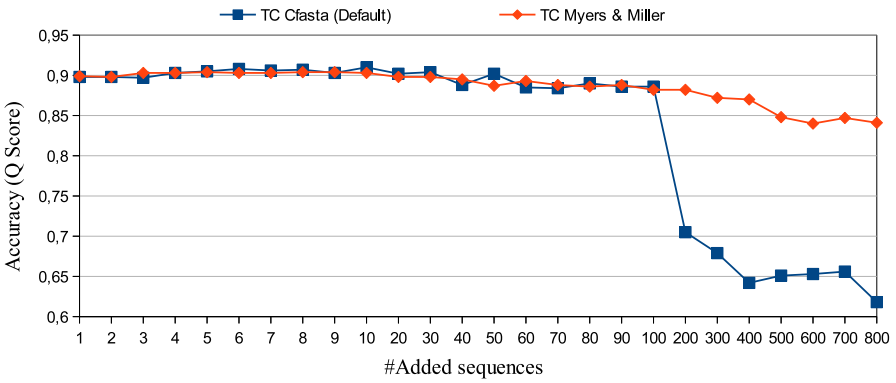

Fig. 3 Comparison of the accuracy of TC and TC Myers & Miller

(OLM), capable of reducing the amount of consistency data stored (i.e. the memory requirements), allowing a reduction in the execution time and an improvement in the scalability of T-Coffee by treating a large number of sequences. In T-Coffee, the consistency-based scheme is achieved through a collection of pairwise alignments called library, as presented above.

The library optimization method (OLM) is based on two complementary methods. The first, the *Essential library* method, is applied to the *Primary library* construction and identifies the information that will be useful during the alignment and the information that can be discarded because it does not add representative information for the next progressive alignment stage.

The second method, called *Threshold library*, discards the constraints that provide little or no information for the alignment. It identifies the constraints with some influence on the alignment and evaluates the deviation of their weight with regard to the maximum weight present in the library. The user defines the maximum allowed deviation, i.e. *threshold*. The residues with values lower than the *threshold* are discarded. The *threshold* determines how aggressive the reduction of the library can be.

The Essential and Threshold library optimizations improve the scalability and performance of T-Coffe, but do not solve the problem definitively. With these optimizations, T-Coffee is able to align twice as many sequences, but the scalability ultimately depends on the memory available in the computer.

3 Scalable consistency library

To improve the T-Coffee scalability, the main target is to reduce the huge amount of memory needed when the number of sequences increases. It is also necessary to pass from exponential to constant linear library growth, to prevent it from continuing to increase. Thus, any reduction in the library, L, implies picking up some residues and excluding others.

The residues of the pairwise can be seen as a list of constraints, where each constraint has a weight (W) defining how close it is the higher its value is. Thus, the way to reduce the size of the consistency library is to discard those constraints with lower values.

Our proposal, named *Bound Library Method* (BLM), has the total amount of available memory to store the consistency library as an input parameter and determines whether each new constraint must be stored in function of its score.

The implementation of BLM is based on a queue structure, in which all the library constraints are stored in a sorted way. Each new constraint is evaluated to determine if it must be pushed into the queue. The implementation of this method is described in Algorithm 1. This process can be divided into two different scenarios according to the queue state:

- *The queue is not full* In this case, there is free enough memory to store the constraint, thus, it is shortly pushed. This is evaluated in Line 6.
- *There are more residues than the defined library bound size* In this situation, evaluated in Line 8, it is necessary to discard information, so that the higher weighted residues achieve better quality, the constraint with lower value, located at the end of the sorted queue, are popped out of the queue. When there are multiple constraints with the same weight, this can cause problems if we always eliminate them in the same order (FIFO, LIFO), as this would mean that all replacements could affect the same sequences. To avoid this situation, BLM selects the constraint to be replaced randomly. This approximation is implemented by the random weight W_2 incorporated into Line 7 of the algorithm.

Algorithm 1 Bound Library method

1. For each sequence $S_i \in S_1..S_N$ and $S_i \neq S_j$
2. For each sequence $S_j \in S_i..S_N$ where $S_i \neq S_N$
3. PA_{ij}=Pairwaise-Alignment(S_i, S_j)
4. For each residue $x \in S_i$, $y \in S_j$| are aligned in $PA_{i,j}$
5. $W_{(x,y)} = \frac{\sum OCCURRENCE(PA_{i,j})}{RESIDUES(PA_{i,j})}$
6. If (Q-Lenght < MAX-BOUND) then
7. Q-Push-Sorted(S_i, S_j, x, y, $max(L(S_i^x, S_j^y), W_{(x,y)}), W_2$)
8. Else If (Q-Lenght == MAX-BOUND) and (($max(L(S_i^x, S_j^y), W_{(x,y)}) >$ Q-Pop→W) then
9. Q-Pop and Q-Push-Sorted(S_i, S_j, x, y, $max(L(S_i^x, S_j^y), W_{(x,y)})$)
10. end_if
11. end_if
12. end_for
13. end_for
14. end_for
15. For each constraint ∈ Q
16. $L(Q \rightarrow S_i^x, Q \rightarrow S_j^y)$= Q→W
17. end_for

Finally, all the consistency information present in the queue is passed to the TC consistency library, Loop in Line 15, and then the progressive alignment will start.

4 Recovering accuracy

It is known that the final biological quality of the alignment could be negatively affected by reducing the information stored in the library. For this reason, a huge reduction in

the consistency information will have an important impact on the alignment accuracy. Therefore, it is not enough to guarantee that the consistency library can scale, but rather, it is necessary to introduce additional mechanisms to recover the consistency lost in the reduction process.

In this section, we explain three innovative approaches we implemented in T-Coffee to recover the lost accuracy. These methods are only applied when there is no consistency information about residues r1 and r2 related with the sequences s1 and s2 in the progressive alignment step, because it was discarded while generating the library. Thus, when the progressive alignment fails to locate the consistency data in the library, we provide a consistency value that substitutes the lost one.

The first approach can only be used when there are still some residues not directly related to the pair of constraints that are being computed (Related consistency). Otherwise, the other two methods are only used when there is no information at all (Dynamic and static substitution matrix).

- *Related consistency* (RC) When there is no direct constraint between the analyzed residues (r1, r2), this method takes into consideration the constraints between r1 and r2 that are present in any other sequences from the input set. Then, the weights of these related constraints are used to compute a normalized score. This method requires no additional memory because the related consistency information is already in the BLM library.
- *Dynamic substitution matrix* (DSM) This matrix is calculated on-the-fly while the primary library is being built, averaging all the related constraints for every pair of residues, providing a substitution matrix like PAM or BLOSUM. This substitution matrix only depends on the residues, not on the sequences, that is why it requires far less memory than the primary library, only needing R^2 entries (R being the possible residues, typically $R = 20$). Therefore, the DSM is small and constant. Another difference from other standard substitution matrix is that this one is customized for the sequences that must be aligned because it has been created only with the information related to these.
- *Static substitution matrix* (SSM) It is possible that the DSM matrix contains empty cells because all the possible pairs of residues in the primary library were not computed. When this situation occurs, we combine the DSM matrix with other well-known traditional substitution matrices, such as Blosum or PAM. Thus, the log-odds score is obtained for each of the possible residue pairs.

Those three methods can be used alone or combined in any different configuration. In the experimental study, we classified each method giving it a different weight in function of its relevance. Thus, we propose the following hierarchy: BLM > RC > DSM > SSM, BLM being the one with the highest weight and SSM the lowest.

5 Experimental study

In this section, we evaluate the BLM method in combination with the three recovering accuracy methods (RC, DSM, SSM). This experimental study evaluates (1) the effectiveness of the recovering methods, (2) the accuracy obtained from our proposal

when increasing the number of sequences, and finally (3) the global performance of T-Coffee.

To perform these test we used the following two benchmarks:

- BALiBASE [12] is a database of high-quality documented and manually-refined reference alignments based on 3D structural superpositions. The accuracy of the alignments is measured using two accuracy metrics: the Sum-of-Pairs (SP) and the Total Column Score (TCS), which are obtained comparing the user alignment against a reference alignment.
- HomFam [10]: The existing benchmark data-sets are very small (BALiBASE 150 and prefab 50 sequences). Homfam provides large data-sets using Pfam families with thousands of sequences. To validate the results of aligning a Pfam family, some reference alignments are needed. The Homstrad site contains some reference alignments and the corresponding Pfam family. These references are previously de-aligned and shuffled to the data-set. After the alignment process the reference sequences are extracted and compared with the originals in Homstrad. The more similar are, the best alignment is.

5.1 Library size and its impact on accuracy

In the first experiment, we analyze the impact of the proposed recovery methods on the accuracy: Related consistency (RC), Dynamic substitution matrix (DSM) and Static substitution matrix (SSM).

This experiment was done by averaging the SP score from aligning the datasets provided by BALiBASE benchmark. We launched these tests in a cluster with 24 nodes where each node have a 2.4 GHz Intel 2 Quad with 8 GB of RAM. Each node evaluates an individual dataset, allowing the complete BALiBASE evaluation to be performed concurrently. T-Coffee is also a multicore application that takes advantage of each node's configuration.

Figure 4 shows the accuracy results obtained for each recovery method when the maximum size of the library was limited to 1 % of the original (123 MB). The first column represents the results for the BLM executed alone, and the next columns correspond to the BLM plus the three proposed methods, SSM, DSM and RC. The last column combines all of these. It can be seen that all the recovering methods improve the score of BLM. Compared with DSM and RC, SSM obtains the best result because it has no gap as it corresponds to a standard substitution matrix. Thus, all residues have consistency data to be used in the progressive alignment. Furthermore, if all methods are used together, the final accuracy is significantly increased by a factor of more than 6. This is because the recovery data, present in the DSM and RC substitution matrices, takes into account information obtained in the library generation step that is directly related to the sequences that are being aligned. In the next subsections, we use the combination of the three methods to recover as much accuracy as possible.

Next, we evaluate the impact of library size on the accuracy obtained. The library size starting point corresponds to the size used by our previous OLM method. This size is the latest possible value at which OLM can perform the alignment. Thus, we

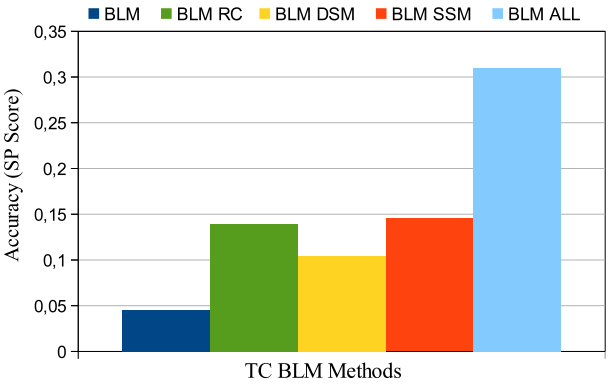

Fig. 4 Comparison between BLM configurations for all BaliBASE database

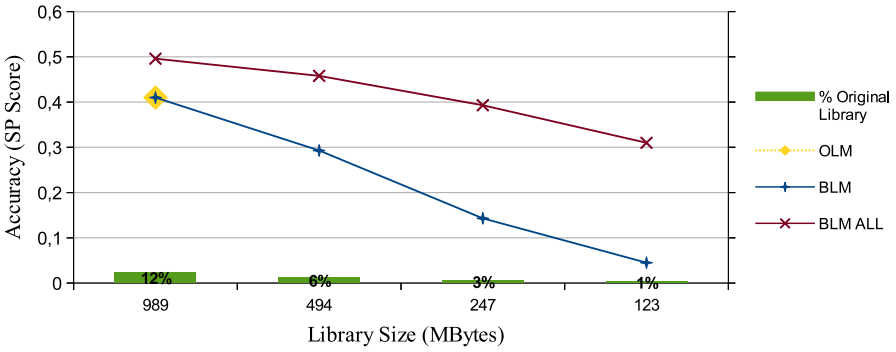

Fig. 5 Accuracy comparison using the SP score from the BaliBASE database

continue reducing the amount of memory used to store the consistency library to evaluate its impact on the final accuracy.

Figure 5 shows the results obtained. The yellow point corresponds to the last effective running of the OLM method, which corresponds to 12 % (989 MB) of the total amount of memory needed to store the complete consistency library. Thus, we continue reducing by $\frac{1}{2}$ factor; 6 % (494 MB), 3 % (247 MB) and 1 % (123 MB). It can be seen that OLM and BLM obtain the same accuracy for the same library size, the first point. However, by applying the recovery methods, it is possible to improve the result by 25 %. When the library size is constrained, the obtained accuracy also diminishes as expected. Otherwise, the BLM-ALL is capable of reducing the impact of this reduction.

These results demonstrate the effectiveness of the recovery methods to reduce the size of the library without a big penalization of accuracy.

5.2 Scalability study

In this experimental study, we evaluated the BLM and recovery methods while increasing the number of sequences. The experiment consisted of running OLM and BLM with a dataset of HomFam named amino acid dehydrogenase which contained 8 refer-

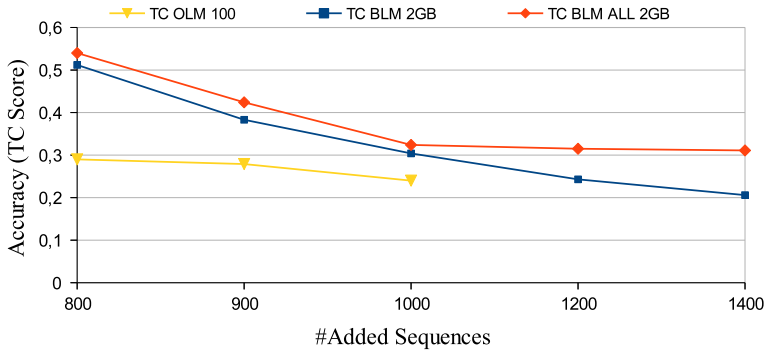

Fig. 6 Accuracy obtained when the number of sequences grows up

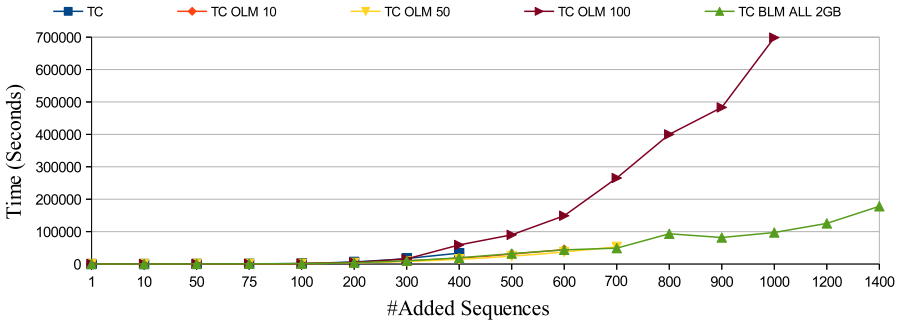

Fig. 7 Execution time with each method when more sequences are added

ences and over one thousand sequences to mix. The study started with 800 sequences. The OLM method was configured with a threshold 100 that means that only the highest values were stored in the library. For the BLM, the library size was limited to 2 GB, the size at which the original T-Coffee fails to align 400 sequences.

Figure 6 shows the accuracy of aligning as the number of sequences was increased. The OLM-100 is capable of continuing to align until 1,000 sequences, so much more than the original T-Coffee. Beyond this number of sequences, only the BLM method was able to continue. This is because the memory requirements are limited and cannot exceed the computer resources, and thus make it possible to continue running. The BLM-ALL obtained better results in general. It is important to note that not only is it possible to align more sequences but also the accuracy obtained is better.

In Fig. 7, we also evaluated the execution time compared with the original T-Coffee and the OLM method with different threshold values (10, 50, 100). As can be observed, as the number of sequences increased, the execution time grew exponentially. Only the OLM-100 was able to continue running until reaching 1,000 sequences, but requiring 8× runtime than BLM-AL. We can also note that the growth in the BLM-all execution time was linear but quadratic in the other approaches.

In Fig. 8 the total amount of memory used when more sequences are added is shown. The BLM-ALL memory bound was reached for 600 sequences and after that, it remained constant, while the others grew quadratically.

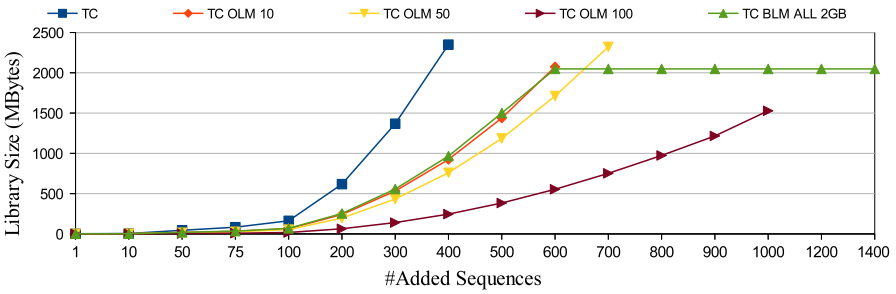

Fig. 8 Library size with each method when more sequences are added

6 Conclusions

In this paper, the authors present a scalable method to build the consistency library of a Multiple Sequence Aligner. To improve the accuracy of the resulting alignments, we also propose different methods to replace part of the consistency information lost due to the huge reduction in the library size. Both approaches are applied on-the-fly during the process of building the library. The goal is to maintain the best consistency information while drastically limiting the size of memory that the library may use.

We prove that BLM is able to reduce the number of entries used by TC, maintaining them fixed to a certain amount. This has various effects on the aligner. When the bound is reached, the execution time tends to be more linear than exponential. In the case of BALiBASE, it also affects the accuracy when the dataset is so small, but with bigger datasets it seems that the reduction affects this positively. We also show that all three consistency recovery methods (RC, DSM and SSM) are able to replace the consistency loss. Moreover, we can use all three together to improve the quality of the alignments significantly. Furthermore, this proposal is able to deal with large-scale alignments, thousands of sequences, which would not be possible using consistency.

For future work, there are several approaches to follow. First, will analyze further the impact of the consistency lost, to provide an additional mechanism to improve the accuracy of the alignments. We also aim to improve the performance of the alignment tool doing several stages in parallel, such as the computation of each pairwise to generate the library and the progressive alignment. These stages will be implemented using mapreduce to store the data and increase the scalability.

Acknowledgments This work has been supported by the Government of Spain TIN2011-28689-C02-02. Cedric Notredame is funded by the Plan Nacional BFU2011-28575 and The Quantomics project (KBBE-2A-222664).

References

1. Blackshields G, Sievers F, Shi W, Wilm A, Higgins D (2010) Sequence embedding for fast construction of guide trees for multiple sequence alignment. Algorithms Mol Biol 5(1):1–11

2. Do C, Mahabhashyam M, Brudno M, Batzoglou S (2005) ProbCons: Probabilistic consistency-based multiple sequence alignment. Genome Res 15(2):330–340
3. Katoh K, Standley D (2013) Mafft multiple sequence alignment software v. 7: Improvements in performance and usability. Mol Biol Evol 30(4):772–780
4. Larkin M, Blackshields G, Brown N, Chenna R, McGettigan P, McWilliam H, Valentin F, Wallace I, Wilm A, Lopez R, Thompson J, Gibson T, Higgins D (2007) Clustal w and clustal x version 2.0. Bioinformatics 23(21):2947–2948
5. Liu K, Linder CR, Warnow T (2010) Multiple sequence alignment: a major challenge to large-scale phylogenetics. PLoS Curr Tree Life. doi:10.1371/currents.RRN1198
6. Liu K, Warnow TJ, Holder MT, Nelesen SM, Yu J, Stamatakis AP, Linder CR (2012) Saté-ii: Very fast and accurate simultaneous estimation of multiple sequence alignments and phylogenetic trees. Syst Biol 61(1):90–106
7. Notredame C, Higgins D, Heringa J (2000) T-Coffee: a novel method for fast and accurate multiple sequence alignment. J Mol Biol 302(1):205–217
8. Notredame C, Holm L, Higgins DG (1998) Coffee: an objective function for multiple sequence alignments. Bioinformatics 14(5):407–422
9. Orobitg M, Cores F, Guirado F, Kemena C, Notredame C, Ripoll A (2012) Enhancing the scalability of consistency-based progressive multiple sequences alignment applications. In: IEEE 26th International Parallel Distributed Processing Symposium, pp 71–82
10. Sievers F, Dineen D, Wilm A, Higgins DG (2013) Making automated multiple alignments of very large numbers of protein sequences. Bioinformatics 29(8):989–995
11. Sievers F, Wilm A, Dineen D, Gibson T, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Soding J, Thompson J, Higgins D (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Mol Syst Biol 7:539
12. Thompson J, Plewniak F, Poch O (1999) BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. Bioinformatics 15(1):87–88
13. Wang LS, Leebens-Mack J, Wall PK, Beckmann K, dePamphilis CW, Warnow T (2011) The impact of multiple protein sequence alignment on phylogenetic estimation. IEEE/ACM Trans Comput Biol Bioinform 8(4):1108–1119