Multiple Sequence Alignment Using SAGA: Investigating the Effects of Operator Scheduling, Population Seeding, and Crossover Operators

René Thomsen and Wouter Boomsma

Bioinformatics Research Center (BiRC)
Department of Computer Science
University of Aarhus
Ny Munkegade, Bldg. 540.
DK-8000 Aarhus C, Denmark
{thomsen,wb}@daimi.au.dk

Abstract. Multiple sequence alignment (MSA) is a fundamental problem of great importance in molecular biology. In this study, we investigated several aspects of SAGA, a well-known evolutionary algorithm (EA) for solving MSA problems. The SAGA algorithm is important because it represents a successful attempt at applying EAs to MSA and since it is the first EA to use operator scheduling on this problem. However, it is largely undocumented which elements of SAGA are vital to its performance. An important finding in this study is that operator scheduling does not improve the performance of SAGA compared to a uniform selection of operators. Furthermore, the experiments show that seeding SAGA with a ClustalW-derived alignment allows the algorithm to discover alignments of higher quality compared to the traditional initialization scheme with randomly generated alignments. Finally, the experimental results indicate that SAGA's performance is largely unaffected when the crossover operators are disabled. Thus, the major determinant of SAGA's success seems to be the mutation operators and the scoring functions used.

1 Introduction

During the last decades, the amount of available biological sequence data (DNA, RNA, and proteins) has increased exponentially. This wealth of new information requires automated methods that can assist practitioners in the process of analyzing and interpreting the data. Valuable tools in this context are multiple sequence alignment (MSA) programs, which allow for the comparison of multiple sequences.

Among the vast number of possible applications, MSAs can be used to (i) infer phylogenetic relationships of organisms, (ii) discover conserved motifs that might be of great importance on the levels of transcription, translation, or structure/function, and finally (iii) assist secondary and tertiary structure prediction methods.

Initially, the most popular methods for obtaining MSAs used dynamic programming (DP) because DP can guarantee a mathematically optimal alignment given the commonly used sum-of-pairs (SOP) scoring function [1]. However, DP-based methods can only handle a relatively small number of sequences because the size of the lookup table increases dramatically with the number of sequences in the alignment and their length. In fact, finding the optimal MSA solution wrt. the SOP score is known to be NP-hard [2].

In order to solve larger problem instances several heuristics have been introduced. The most popular heuristic is ClustalW [3], an algorithm that belongs to the class of progressive alignment methods [4]. These methods gradually construct an alignment by first estimating the evolutionary distance between all sequences to be aligned and then aligning the sequences in order of decreasing similarity. Although the progressive methods are very fast they typically suffer from entrapment in local optima because they optimize the alignment in a pairwise manner, not taking the entire alignment into account.

To overcome this problem several stochastic heuristics have been applied to MSA, such as simulated annealing [5] and evolutionary algorithms (EAs) [6, 7]. Typically, these methods start with randomly generated candidate alignments that are gradually improved using several variation operators.

SAGA (Sequence Alignment by Genetic Algorithm) [6] introduced the idea of operator scheduling (OS) for MSA based on the assumption that the scheduling of the operators would improve the overall performance of the algorithm. However, our preliminary experiments with several OS schemes on MSA using the MSAEA [9] did not indicate any improvements for OS compared to choosing the operators randomly with a uniform probability. More specifically, these observations raised the question of whether OS has any effect at all when applied to EAs for MSA.

Moreover, Thomsen et al. [8, 9] investigated the effects of seeding an EA with a ClustalW-derived solution. The experimental results indicated that seeding the EA resulted in a marked improvement in runtime needed to derive solutions of high quality. Furthermore, it was shown that the resulting alignments were significantly better than the ClustalW seed, making the EA useful as an alignment improver.

These findings motivated us to investigate the following aspects of SAGA: (i) operator scheduling compared to uniform choice of variation operators, (ii) seeding the initial population with a ClustalW-derived alignment compared to random initialization, and finally (iii) the effect of crossover operators compared to using mutation only. These investigations were conducted on selected MSA benchmark problems obtained from the BAliBASE sequence alignment database [10].

The experimental results show that operator scheduling does not improve the overall performance of SAGA compared to a uniform selection of operators. Moreover, the results indicate that SAGA is able to obtain better results using seeds compared to random initialization given the same number of fitness eval-

uations. Finally, the use of crossover operators did not generally increase the performance.

2 SAGA

In 1996 Notredame and Higgins introduced SAGA [6], one of the first evolutionary algorithms (EAs) for MSA. Basically, SAGA resembles a standard EA with a population of candidate alignments (individuals) that are subjected to variation and selection. During selection the individuals compete for survival and the most promising ones are transferred to the next generation. Contrary to other EAs solving MSA, SAGA provides a total of 25 variation operators (19 mutation and 6 crossover) with a variety of functionality, such as modifying gap regions (adding, deleting, moving gaps) and combining promising regions. A complete description of the operators is beyond of the scope of this paper, see [6] for a detailed description. The default initialization mode in SAGA is to randomly initialize all individuals by prefixing a randomly chosen number of gaps to each sequence.

SAGA differs from all other MSA algorithms regarding the utilization of the operators. It uses an operator scheduling (OS) strategy originally described by Davis in 1989 [11] to select which operators to use. The OS strategy works as follows: Each operator is assigned a probability for its application. Initially, these probabilities are all equal, but during the course of the run they are adapted based on the recent performance of the operator. Operators are rewarded when they create an individual with a fitness greater than any of the current individuals in the population. Furthermore, operators responsible for the individuals' parents and more distant ancestors are rewarded with some percentage of the original reward. This is motivated by the fact that a series of suboptimal solutions is often necessary in order to reach a new optimal solution and corresponding operators therefore should be rewarded. With certain intervals, a new probability setting is computed as a weighted sum of the previous setting and the distribution of rewards among the operators.

The motivation for using OS in SAGA is that it is difficult to determine in advance which of the 25 operators to apply. The problem gets more complicated since the utilization of each operator might depend on the actual alignment problem being solved and optimal usage of operators might change during the course of the optimization run. The idea is that by measuring the performance of operators during the run, operators can be continuously scheduled so that the best operators are used at all times.

SAGA provides three different ways of scoring alignments, (i) sum-of-pairs without weights (SOP), (ii) sum-of pairs using weights (WSOP), and (iii) Consistency based Objective Function For alignmEnt Evaluation (COFFEE) [12]. The three scoring functions are briefly described below (see [6] and [12] for more details).

The SOP and WSOP scoring functions are almost identical except for the additional sequence weights used in WSOP. Equation 1 shows how the total

score of an alignment is calculated. N specifies the number of sequences in the alignment, S_i and S_j are two aligned sequences from the given alignment, and W_{ij} is the corresponding weight (a detailed description of the weighting scheme used is provided in [12]). When no weights are provided, i.e., using SOP, all weights are set to 100 (default setting in SAGA). The $COST$ function calculates the substitution scores for each pair of residues in sequence S_i and S_j using e.g, BLOSUM or PAM matrices. Furthermore, the $COST$ function includes affine gap penalties, e.g., penalties for gap opening and extension (GOP and GEP, respectively). Terminal gaps are only penalized for extension, not for opening. Moreover, SAGA transforms the (W)SOP scoring functions into minimization problems by scaling the entries of the substitution matrix.

$$(W)SOP = \sum_{i=2}^{N} \sum_{j=1}^{i-1} W_{ij} \times COST(S_i, S_j) \; . \tag{1}$$

The COFFEE score is defined as a maximization problem where the task is to maximize the consistency between the residue pairs in the alignment and the residue pairs observed in a library generated by pairwise alignments between all the sequences. Equation 2 shows how the score is calculated.

$$COFFEE = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} W_{ij} \times SCORE(A_{ij}) \; . \tag{2}$$

$SCORE(A_{ij})$ is the number of aligned pairs of residues that are shared between A_{ij} and the generated library, where A_{ij} is the pairwise projection of sequence S_i and S_j obtained from the evaluated candidate alignment.

For the experiments described in this study we used SAGA V0.95, which is publically available from `http://igs-server.cnrs-mrs.fr/~cnotred/`. The original termination criteria (terminate after 100 consecutive generations without performance improvements) was modified so that SAGA terminates when a certain number of fitness evaluations has occurred. This change was necessary in order to make a general comparison between different settings possible.

3 Experiments

3.1 BAliBASE Benchmarks

The BAliBASE database [10] contains 142 multiple sequence alignments, which are manually refined from the known 3D structures of proteins. This makes it possible to evaluate the quality of MSA algorithms regarding their ability to derive *true* (biological plausible) alignments. Table 1 shows the protein sequence data sets used in our experiments. All seven data sets were randomly selected from the first reference set of the BAliBASE database (version 2, `http://bess.u-strasbg.fr/BioInfo/BAliBASE2/`).

Table 1. BAliBASE data sets used in the experiments. NSEQ = number of sequences, LSEQ = length of sequences, SEQID = percent residue identity

Data set	NSEQ	LSEQ (min,max,avg)	SEQID
1idy	5	(49,58,53.6)	<25%
1aboA	5	(49,80,63.6)	<25%
kinase	5	(263,276,270.2)	<25%
1hfh	5	(116,132,121.2)	20-40%
1pfc	5	(108,117,112)	20-40%
1pii	4	(247,259,251.5)	20-40%
451c	5	(70,87,80)	20-40%

3.2 Evaluation Measures

The results from the SAGA runs were evaluated according to each of the scoring functions described in Section 2. Furthermore, the quality of the overall best found alignments were compared with the BAliBASE reference alignments using the BAliBASE evaluation measures described below.

The BAliBASE sum-of-pairs score (SPS) was calculated as follows: Given a candidate alignment (individual) of N sequences containing M columns, the i'th column in the alignment is designated by A_{i1}, A_{i2}, ..., A_{iN}. For each pair of residues A_{ij} and A_{ik} we defined p_{ijk} such that $p_{ijk} = 1$ if residues A_{ij} and A_{ik} from the candidate alignment were aligned with each other in the reference alignment. Otherwise, $p_{ijk} = 0$. The score for the i'th column is thus:

$$S_i = \sum_{j=1, j\neq k}^{N} \sum_{k=1}^{N} p_{ijk} \ . \tag{3}$$

The overall SPS for the candidate alignment is:

$$SPS = \sum_{i=1}^{M} \frac{S_i}{\sum_{i=1}^{M_r} S_{ri}} \ , \tag{4}$$

where M_r is the number of columns in the reference alignment and S_{ri} is the score S_i for the i'th column in the reference alignment.

The BAliBASE column score (CS) was calculated as follows: Given an alignment as described above, the score C_i for the i'th column is equal to 1 if all the residues in the column are aligned in the reference alignment. Otherwise, C_i is set to 0. The overall CS for the candidate alignment is:

$$CS = \sum_{i=1}^{M} \frac{C_i}{M} \ . \tag{5}$$

The ranges of SPS and CS are 0.0-1.0, where higher values indicate closer resemblance with the BAliBASE reference alignment. When calculating both scores we

used the annotation files provided with BAliBASE to identify core blocks in the reference alignment, i.e., only the regions that were marked as being important were used in the calculation.

3.3 Experimental Setup and Data Sampling

In this study we used the default parameter settings provided with SAGA. The population size was set to 100 and the total number of fitness evaluations allowed was set to 200000 (max number of evaluations in the termination criterion). The SOP and WSOP scoring functions used the BLOSUM-45 substitution matrix and gap penalties were set to $GOP = 8$ and $GEP = 12$ (SAGA default settings).

Different configurations of SAGA were investigated regarding the seven alignment problems shown in Table 1 (see section 4 for more details). Each experiment was repeated 30 times with different random seeds, and the average of the 30 best alignment scores was recorded.

4 Results

The performance of SAGA was evaluated regarding the seven alignment benchmarks described in section 3.1. The experiments were designed to provide answers to the following three main questions: (i) does operator scheduling (OS) outperform simple uniform choice of operators?, (ii) what is the effect of using seeding compared to random initialization only?, and (iii) does crossover improve performance compared to using mutation only?

Table 2 summarizes the results from all the experiments. The *Configuration* column shows the choice of configuration for each particular experiment (see caption for Table 2 for information on the abbreviations used). The remaining columns show the BAliBASE CS and SPS measures (mean of 30 runs) for each of the tested alignments. The significance of the results was validated using the Wilcoxon Signed Rank Test (statistical p-values are provided in supplementary material).

Overall, the results from table 2 show that OS is not an advantage compared to a uniform choice of operator (the differences are not statistically significant ($p > 0.0625$)). Moreover, seeding improves the performance of SAGA in four of the seven test cases (the results are statistically significant with $p < 0.05$ for *1aboA*, *kinase*, *1hfh*, and *1pii*). For the remaining test cases, the differences observed are not significant. The use of crossover operators improves the performance for *kinase* and *1hfh*, while being a disadvantage for the *1aboA* problem (the results are statistically significant with $p < 0.05$). For the four other test cases, using crossover does not make any performance difference.

In conclusion, SAGA using seeding, crossover, OS, and the SOP or WSOP scoring function (S,C,O, SOP/WSOP) or the same settings except OS (S,C,NO, SOP/WSOP) generally performed better than all other configurations. The small differences between these settings were not statistically significant according to the Wilcoxon Signed Rank Test.

Table 2. BAliBASE evaluation measures on tested benchmarks using different SAGA configurations. *R* (random initialization), *S* (ClustalW-seed), *C/NC* (crossover/no-crossover), *O/NO* (operator-scheduling/uniform-choice), *SOP/WSOP/COFFEE* (scoring functions)

	Data set						
Configuration	1idy CS/SPS	1aboA CS/SPS	kinase CS/SPS	1hfh CS/SPS	1pfc CS/SPS	1pii CS/SPS	451c CS/SPS
R, C, O, SOP	0.380/0.565	0.484/0.721	0.353/0.606	0.878/0.944	0.991/0.997	0.805/0.895	0.622/0.746
R, C, O, WSOP	0.380/0.550	0.518/0.730	0.304/0.558	0.889/0.950	0.940/0.979	0.805/0.891	0.638/0.744
R, C, O, COFFEE	0.167/0.510	0.570/0.775	0.541/0.734	0.847/0.936	0.893/0.959	0.800/0.882	0.611/0.787
R, NC, O, SOP	0.380/0.540	0.566/0.769	0.272/0.565	0.853/0.931	0.990/0.997	0.775/0.877	0.620/0.717
R, NC, O, WSOP	0.380/0.550	0.562/0.765	0.272/0.558	0.857/0.936	0.940/0.979	0.768/0.873	0.666/0.732
R, NC, O, COFFEE	0.207/0.520	0.570/0.774	0.468/0.705	0.853/0.937	0.890/0.958	0.801/0.883	0.635/0.794
R, C, NO, SOP	0.380/0.550	0.508/0.732	0.362/0.617	0.894/0.951	0.989/0.996	0.805/0.896	0.620/0.744
R, C, NO, WSOP	0.380/0.550	0.496/0.725	0.298/0.569	0.897/0.955	0.940/0.979	0.809/0.893	0.641/0.765
R, C, NO, COFFEE	0.151/0.507	0.570/0.774	0.531/0.732	0.851/0.936	0.891/0.958	0.801/0.883	0.606/0.784
R, NC, NO, SOP	0.380/0.548	0.555/0.764	0.305/0.583	0.863/0.935	0.992/0.997	0.799/0.890	0.600/0.718
R, NC, NO, WSOP	0.380/0.550	0.544/0.758	0.278/0.566	0.859/0.935	0.940/0.979	0.804/0.890	0.627/0.739
R, NC, NO, COFFEE	0.203/0.533	0.569/0.775	0.450/0.692	0.822/0.924	0.890/0.958	0.802/0.884	0.639/0.796
S, C, O, SOP	0.380/0.595	0.566/0.768	0.621/0.792	0.896/0.953	0.985/0.995	0.810/0.899	0.589/0.698
S, C, O, WSOP	0.380/0.543	0.566/0.766	0.627/0.795	0.889/0.950	0.940/0.979	0.810/0.893	0.618/0.709
S, C, O, COFFEE	0.186/0.542	0.570/0.776	0.554/0.737	0.845/0.934	0.890/0.958	0.800/0.882	0.630/0.794
S, NC, O, SOP	0.380/0.596	0.567/0.767	0.531/0.755	0.883/0.947	0.988/0.996	0.810/0.897	0.620/0.711
S, NC, O, WSOP	0.380/0.541	0.568/0.769	0.529/0.752	0.894/0.953	0.940/0.979	0.810/0.892	0.630/0.715
S, NC, O, COFFEE	0.207/0.528	0.570/0.776	0.527/0.729	0.859/0.939	0.892/0.959	0.800/0.882	0.631/0.794
S, C, NO, SOP	0.380/0.594	0.567/0.767	0.608/0.787	0.909/0.959	0.985/0.995	0.810/0.899	0.585/0.696
S, C, NO, WSOP	0.380/0.548	0.567/0.768	0.616/0.800	0.903/0.957	0.940/0.979	0.810/0.892	0.615/0.707
S, C, NO, COFFEE	0.183/0.525	0.570/0.776	0.559/0.736	0.834/0.929	0.892/0.959	0.800/0.882	0.629/0.796
S, NC, NO, SOP	0.380/0.594	0.568/0.766	0.547/0.763	0.890/0.950	0.988/0.996	0.810/0.898	0.605/0.705
S, NC, NO, WSOP	0.380/0.550	0.568/0.767	0.581/0.782	0.878/0.945	0.940/0.979	0.810/0.892	0.624/0.712
S, NC, NO, COFFEE	0.218/0.552	0.570/0.776	0.518/0.726	0.816/0.921	0.893/0.959	0.800/0.882	0.633/0.795

Finally, the best configuration found in the previous experiments was compared to the alignments obtained from SAGA using default parameter settings (R,C,O,SOP), ClustalW and T-Coffee [13]. Table 3 shows the CS and SPS values for each of the seven testcases using one of the tested alignment programs. In five out of seven cases, SAGA with the best-found configuration outperformed the other alignment programs wrt. the BAliBASE evaluation measures (three times using CS and five times using SPS). For the remaining two cases, SAGA using random initialization obtained slightly higher scores than using the ClustalW seed for *1pfc*. However, the differences are not statistically significant. SAGA was not able to improve the *451c* problem. In fact, the resulting alignment had a worse score compared to the ClustalW seed. This case shows that an improvement of the SOP score does not always correlate with an improvement of the alignment quality, i.e., a better CS or SPS value. In conclusion, using SAGA with the new parameter settings outperformed SAGA using default values for five of the seven tested problems, indicating the importance of population seeding when the number of evaluations are fixed at 200000.

Table 3. BAliBASE evaluation measures on tested benchmarks using different MSA programs. *SAGA(seeding)* is SAGA with the best-found settings and ClustalW seeding (see previous table) and *SAGA(random)* is SAGA using default parameter settings and random initialization. All SAGA results represent mean values obtained from 30 runs

	MSA program			
	SAGA (seeding)	SAGA (random)	ClustalW	T-Coffee
Data set	CS/SPS	CS/SPS	CS/SPS	CS/SPS
1idy	**0.380**/**0.595**	**0.380**/0.565	**0.380**/0.588	0.000/0.150
1aboA	**0.566**/**0.768**	0.484/0.721	0.540/0.755	0.150/0.570
kinase	0.621/**0.792**	0.353/0.606	**0.630**/0.788	0.600/0.769
1hfh	**0.896**/**0.953**	0.878/0.944	0.770/0.900	0.870/0.938
1pfc	0.985/0.995	**0.991**/**0.997**	0.750/0.903	0.940/0.979
1pii	0.810/**0.899**	0.805/0.895	0.740/0.855	**0.820**/**0.899**
451c	0.589/0.698	0.622/0.746	**0.700**/0.755	0.640/**0.786**

5 Discussion

In this paper, we investigated different aspects of SAGA, such as (i) scheduling operators to obtain optimal usage compared to uniform selection of all available operators, (ii) seeding the initial population with a ClustalW derived alignment compared to random initialization only, and finally (iii) using crossover operators compared to using mutation-based operators only.

Overall, the experimental results showed that operator scheduling (OS) did not perform any better than simply choosing the operators uniformly (see table 2). This observation differs from the intuitive notion that OS is useful when solving optimization problems with many variation operators. Preliminary experiments trying out other OS strategies confirmed the reported results. In all

cases uniform selection of operators was as good or better than the tested methods. A possible explanation of why OS does not increase the performance is that repeated usage of an operator that is believed to be good does not necessarily result in a stepwise improvement of alignment quality. The main reason seems to be the stochastic nature of the variation operators which usually randomly select the sequences and/or gap-regions to be modified. However, more in-depth investigations on the search-spaces induced by the MSA scoring schemes are needed to fully understand why OS does not work very well on this particular problem.

Moreover, our study showed that in most cases the seeding approach was able to obtain similar or better scores than the random initialization approach (using the same parameter settings and number of evaluations). Again, this observation is in contrast to the original study by Notredame and Higgins [6] where seeding has been avoided because it was believed to bias the search to local optima. Furthermore, using random initialization without seeding typically required twice as many fitness evaluations as the seeding approach to obtain similar fitness and CS/SPS values (see supplementary material for more details).

Generally, the use of crossover did not improve the performance of SAGA on the tested data sets. On the *kinase* and *1hfh* test cases using crossover operators resulted in slightly better CS/SPS scores whereas omitting crossover improved the results on the *1aboA* test case. Whether to use crossover or not seems to depend on the actual data set in question. More experiments on other BAliBASE data sets might give some guidelines on when to apply or omit crossover operators.

Finally, the comparison to other MSA programs showed that SAGA is able to improve the initial alignment provided by ClustalW in five out of 7 cases (see table 3). Typically, the runtime needed by SAGA was between 40-240 seconds on a 1.8GHz Pentium-IV PC making it a valuable tool as an alignment improver. Furthermore, SAGA using the seeding approach is able to obtain the overall best scores in five out of seven cases compared to ClustalW, T-Coffee, and SAGA using default random initialization.

In conclusion, seeding SAGA with ClustalW solutions improved the convergence properties of SAGA (see convergence graphs in the supplementary material) thus lowering the total number of fitness evaluations needed to obtain alignments of good quality (wrt. CS and SPS measures). The results also indicate that the benefits of using OS and crossover operators are questionable, thus suggesting that simpler algorithms without these features may be sufficient to solve the MSA problem. Repeating the investigations described in this study on all the BAliBASE data sets will provide us with a better indication on whether the use of OS and crossover operators are needed in general.

6 Supplementary Material

Additional experimental results (fitness tables, convergence graphs, etc.) are publically available from `http://www.daimi.au.dk/~thomsen/evobio2004/`.

Acknowledgments

The authors would like to thank the Danish Research Council for financial support. The authors also thank Jakob Vesterstrøm for valuable comments on draft versions of the manuscript.

References

1. Gupta, S., J.D. Kececioglu, J., Schaffer, A.: Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. Journal of Computational Biology **2** (1995) 459–472
2. Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. Journal of Computational Biology **1** (1994) 337–348
3. Thompson, J., Higgins, D., Gibson, T.: Clustal W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting. Nucleic Acids Research **22** (1994) 4673–4680
4. Feng, D., Doolittle, R.: Progressive sequence alignment as a prerequisite to correct phylogenetic trees. Journal of Molecular Evolution **25** (1987) 351–360
5. Kim, J., Pramanik, S., Chung, M.: Multiple sequence alignment using simulated annealing. Computer Applications in the Biosciences (CABIOS) **10** (1994) 419–426
6. Notredame, C., Higgins, D.: SAGA: Sequence alignment by genetic algorithm. Nucleic Acids Research **24** (1996) 1515–1524
7. Chellapilla, K., Fogel, G.B.: Multiple sequence alignment using evolutionary programming. In: Proceedings of the First Congress of Evolutionary Computation (CEC-1999). (1999) 445–452
8. Thomsen, R., Fogel, G., Krink, T.: A Clustal alignment improver using evolutionary algorithms. In: Proceedings of the Fourth Congress on Evolutionary Computation (CEC-2002). Vol. 1. (2002) 121–126
9. Thomsen, R., Fogel, G., Krink, T.: Improvement of Clustal-derived sequence alignments with evolutionary algorithms. In: Proceedings of the Fifth Congress on Evolutionary Computation (CEC-2003). (2003)
10. Thompson, J., Plewniak, F., Poch, O.: BAliBASE: A benchmark alignment database for the evaluation of multiple alignment programs. Bioinformatics **15** (1999) 87–88
11. Davis, L.: Adapting operator probabilities in genetic algorithms. In: Proceedings of the Third International Conference on Genetic Algorithms (ICGA III). (1989) 61–69
12. Notredame, C., Holm, L., Higgins, D.: COFFEE: An objective function for multiple sequence alignments. Bioinformatics **14** (1998) 407–422
13. Notredame, C., Higgins, D., Heringa, J.: T-Coffee: A novel method for fast and accurate multiple sequence alignment. Journal of Molecular Biology **302** (2000) 205–217