# M2Align: parallel multiple sequence alignment with a multi-objective metaheuristic

**Cristian Zambrano-Vega,[1] Antonio J. Nebro,[2,*] José García-Nieto[2] and José F. Aldana-Montes[2]**

[1]Facultad de Ciencias de la Ingeniería, Universidad Técnica Estatal de Quevedo, Quevedo, Ecuador and
[2]Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, 29071 Málaga, Spain

## Abstract

**Motivation:** Multiple sequence alignment (MSA) is an NP-complete optimization problem found in computational biology, where the time complexity of finding an optimal alignment raises exponen-tially along with the number of sequences and their lengths. Additionally, to assess the quality of a MSA, a number of objectives can be taken into account, such as maximizing the sum-of-pairs, maximizing the totally conserved columns, minimizing the number of gaps, or maximizing struc-tural information based scores such as STRIKE. An approach to deal with MSA problems is to use multi-objective metaheuristics, which are non-exact stochastic optimization methods that can pro-duce high quality solutions to complex problems having two or more objectives to be optimized at the same time. Our motivation is to provide a multi-objective metaheuristic for MSA that can run in parallel taking advantage of multi-core-based computers.
**Results:** The software tool we propose, called M2Align (Multi-objective Multiple Sequence Alignment), is a parallel and more efficient version of the three-objective optimizer for sequence alignments MO-SAStrE, able of reducing the algorithm computing time by exploiting the comput-ing capabilities of common multi-core CPU clusters. Our performance evaluation over datasets of the benchmark BAliBASE (v3.0) shows that significant time reductions can be achieved by using up to 20 cores. Even in sequential executions, M2Align is faster than MO-SAStrE, thanks to the encod-ing method used for the alignments.

**Availability and implementation:** M2Align is an open source project hosted in GitHub, where the source code and sample datasets can be freely obtained: https://github.com/KhaosResearch/M2Align.

**Contact:** antonio@lcc.uma.es

## 1 Introduction

Multiple sequence alignment (MSA) (Bacon and Anderson, 1986) is the process of aligning three or more biological sequences (DNA, RNA, protein), and constitutes a widely used technique in several areas of computational biology, such as: homology searches, genomic annotation, protein structure prediction, gene regulation networks, or functional genomics. MSA is an NP-complete optimization problem (Wang and Jiang, 1994), where the time complexity of finding an optimal alignment raises exponentially along with the number of sequences and their lengths.

Another issue in MSA is to provide an efficient method to measure the alignment accuracy, as there is not a consensus on how to do it. There are scores based on nucleotide or amino acid information, such as the totally conserved columns (TCC) percentage and the

Gaps and Non-gaps percentage. In this category, we also find Sum-Of-Pairs (SOP) and the weighted SOP function with affine gap penalties (WSP), which use distance matrices [e.g. Point Accepted Mutation (PAM) (Dayhoff *et al.*, 1978) and Blocks Substitution Matrix (BLOSUM) (Henikoff and Henikoff, 1992)]. Other scores are based on Supplementary Material like homologies or protein structures. For instance, STRIKE (Kemena *et al.*, 2011) uses the molecular contacts from protein structures to calculate the accuracy of alignments. Our interest is to optimize two or more of these scores at the same time, thus giving place to a multi-objective formulation of MSA.

Metaheuristics are non-exact stochastic optimization algorithms, which are able to face complex optimization problems without requiring problem-specific information. When applied to multi-objective problems, these algorithms can produce a front of trade-off solutions in a single run (called Pareto Front), yielding to alternative hypotheses that can be useful from different biological points of view.

The use of multi-objective optimization in some fields of Bioinformatics has shown to have relevant benefits compared to single-objective approaches (Handl *et al.*, 2007). Metaheuristics, particularly the evolutionary algorithm subfamily, have been applied to optimize multi-objective MSAs, such as in Abbasi *et al.* (2015), da Silva *et al.* (2011), Kaya *et al.* (2014), Ortuño *et al.* (2013), Rani and Ramyachitra (2016), Rubio-Largo *et al.* (2015, 2016), Seeluangsawat and Chongstitvatana (2005), Soto and Becerra (2014) and Zhu *et al.* (2016).

One of these algorithms is MO-SAStrE (Ortuño *et al.*, 2013), a multi-objective optimizer for sequence alignments that uses three objectives to evaluate the MSA accuracy: STRIKE score, TCCs, and percentage of non-gaps. However, metaheuristics like MO-SAStrE work by evaluating iteratively thousands of MSA solutions, what can be very time-consuming. Since biologists frequently need to compute multiple alignments for different scale (small, medium and large) input datasets, reducing run times is of high importance, so an approach is to apply parallelism to take advantage of modern multicore computers.

Our proposal lies in this context. Specifically, we have implemented M2Align, a faster and more efficient version of the algorithm MO-SAStrE, able to solve MSA problems in parallel.

As the original MO-SAStrE, our version has the following characteristics:

- It is based on the Non-dominated Sorting Evolutionary Algorithm II or NSGA-II (Deb *et al.*, 2002).
- The evolutionary variation operators are single point crossover and closed gap shifting.
- The objectives to optimize are non-gaps percentage, TCCs and STRIKE.
- The initial population of NSGA-II is filled with pre-alignments obtained with representative MSA tools: ClustalW, MUSCLE, Kalign, Mafft, RetAlign, TCOFFEE, ProbCons and FSA.

Unlike the original MO-SAStrE, our solution incorporates the following features:

- M2Align is written in Java (MO-SAStrE is implemented in Matlab), so it can run in any computer having a Java JDK installed.
- The algorithm in M2Align can be executed in parallel on multicore systems.
- The solution encoding, instead of matrices of integers, is based on the one presented in (Rubio-Largo *et al.*, 2015), which only stores gap information (see the Supplementary Material). As a consequence, the memory requirements are significantly reduced and the genetic operators (crossover and mutation) can be more efficiently implemented.
- If PDB structures are not available, M2Align provides SOP and WSP scores as alternatives to STRIKE.
- M2Align is an Open Source project hosted in GitHub (https://github.com/KhaosResearch/M2Align), thus facilitating their use by interested users. Information about how to download, compile and run it is included in the project site.

## 2 Materials and methods

### 2.1 The NSGA-II algorithm

M2Align and MO-SAStrE are versions of NSGA-II (Deb *et al.*, 2002), the most well-known and used multi-objective metaheuristic, although they are adapted to tackle with the MSA. NSGA-II follows the scheme of a generational evolutionary algorithm, as is depicted in Figure 1. Initially, a population of $N$ tentative solutions is created and then an auxiliary population, also of size $N$, is filled by applying the selection, crossover and mutation evolutionary operators to the solutions of the original population. Then, both populations are merged into one and the best $N$ solutions of it are selected to constitute the population for the next generation of the algorithm.

The term *best* applied to compare solutions in multi-objective optimization is tricky in the situations in which none of them is better than the other in all the objectives (these solutions are referred as non-dominated). The approach taken by NSGA-II is to make a dominance ranking, in the sense that those non-dominated solutions in the combined population has a rank of 1; the rest of non-dominated solutions have a rank of 2, and so on. The solutions are then sorted by rank, and the ones with better ranks are included in the next population. This scheme allows to guide the search towards the Pareto front (i.e. the set of those solutions that are non-dominated with respect to any other in the search space).

If we take a look on the example in Figure 1, we can observe that the group of solutions having a rank of 3 does not fit into the next population, so an additional mechanism must be applied to select some of them. The idea is to apply a density estimator to promote the diversity of the solutions in the population; NSGA-II uses an estimator known as crowding distance (defined in Supplementary Material).

These steps are repeated until a stopping condition is fulfilled, typically by performing a fixed number of iterations. The output of the algorithm execution will be an approximation of the Pareto front.

It is important to note that before the merging and ranking steps, all the solutions of the auxiliary population must be evaluated. This can be computationally expensive in the case of large MSA problems, which suggests that an approach to accelerate the execution of the algorithm is to perform all the evaluations in parallel.

### 2.2 Multi-objective formulation and biological justification

As commented before, M2Align is configured to optimize three objectives: STRIKE, percentage of TCCs and percentage of non-gaps. These three objectives are to be maximized.

STRIKE uses structural information from the Protein Data Bank (PDB) to estimate the contacts between amino acids by using a scoring matrix; this score allows to identify the accuracy of alignments with better precision than other scores such as PAM or BLOSUM. The percentage of TCCs is a widely accepted score because complete

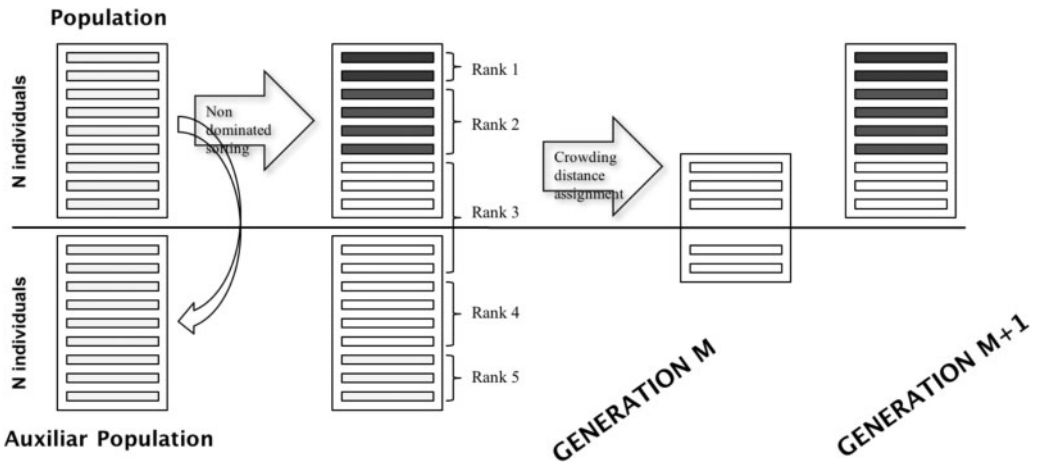


**Fig. 1.** Working scheme of the NSGA-II algorithm

**Table 1.** Methods used to generate the initial population of the algorithms. These eight tools are applied to build initial MSAs for the BAliBASE datasets

| Tool | Version | Type |
|---|---|---|
| ClustalW (Thompson et al., 1994) | 2.1 | Progressive |
| MUSCLE (Edgar, 2004) | 3.8.31 | Progressive |
| Kalign (Lassmann and Sonnhammer, 2005) | 2.04 | Progressive |
| Mafft (Katoh et al., 2002) | 7.245 | Progressive |
| RetAlign (Novák et al., 2010) | 1.0 | Progressive |
| TCOFFEE (Notredame et al., 2000) | 11.00 | Consistency-based |
| ProbCons (Do et al., 2005) | 1.12 | Consistency-based |
| FSA (Roberts et al., 2009) | 1.15.9 | Consistency-based |

columns indicate more conserved regions in sequences. The last score, the percentage of non-gaps, has sense because the number of gaps can be overused to improve the number of alignments, so the idea is to reducing this percentage is to find more compact and realistic alignments. More details about these scores can be found in (Ortuño *et al.*, 2013).

## 2.3 Strategy to generate the initial population

The usual approach to create the initial population in evolutionary algorithms is to fill it with randomly generated solutions, but in MO-SAStrE the adopted approach is to use a set of pre-computed alignments performed by other MSA tools. In M2Align, we follow a similar strategy, i.e. by adding pre-computed solutions to the initial population and creating new alignments by applying a genetic crossover operator to pairs of randomly selected solutions to fill the rest of the population.

For our experiments, we have focused on the the BAliBASE dataset (v3.0) (Thompson *et al.*, 2005). For every problem instance in this benchmark, a number of alignments by using eight representative MSA tools have been generated, namely: ClustalW, MUSCLE, Kalign, Mafft, RetAlign, TCOFFEE, ProbCons and FSA. Their specific versions and features are detailed in Table 1

## 2.4 Solution encoding

With the aim of reducing the high memory cost and execution time that requires classical character or numerical representations, as done in MO-SAStrE (Ortuño *et al.*, 2013), we have implemented a fast and low-memory cost codification of the alignments based on


**Fig. 2.** Example of alignment (left) and how it is encoded in M2Align (right)

groups of gaps, similar to the one proposed by Rubio-Largo *et al.* (2015). This MSA representation only stores the positions (begin, end) of the groups of gaps into the sequences. Figure 2 illustrates an example.

Thus, given a sequence $S$, it is encoded to $S'$ in this way: $s'$: $[(Bgg_1, Egg_1), (Bgg_2, Egg_2), \ldots, (Bgg_n, Egg_n)]$ where $n$ is the number of groups of gaps of the sequence $S$, and $Bgg_x$ and $Egg_x$ represent the initial and final position into the sequence $S$ of the group of gaps $x$, respectively. This codification reduces the time of execution of both genetic crossover and mutation operators, since numerical operations are applied on the gap lists instead of manipulating large sequence of characters.

## 2.5 Genetic operators for MSA

We have implemented the same mutation and crossover operators used by MO-SAStrE (Ortuño *et al.*, 2013). The mutation operator is Closed Gap Shifting, where a random set of closed gaps are shifted to another random position in a sequence. The crossover operator is the Single-Point Crossover adapted to alignments (da Silva *et al.*, 2010). This operator randomly selects a position from the parent 1 by splitting it into two blocks (P1a and P1b) and the parent 2 is tailored so that the right piece can be joined to the left piece of the first parent (P1a) and vice versa. Then, the selected blocks are crossed between these two parents, generating two new individuals with the combination of the blocks: [P1a + P2b] and [P1a + P1b]. Finally, with the aim of reducing the number of gaps into the alignment, the columns containing only gaps are removed. These two operators are illustrated in Figure 3.

## 2.6 Parallel approach

As commented before, a natural strategy for parallelizing NSGA-II is to perform all the function evaluations of the new created solutions at the same time. However, the rest of the steps will be still executed sequentially, so the ratio between parallel and sequential computations must be clearly favorable to the former to obtain significant time reductions.

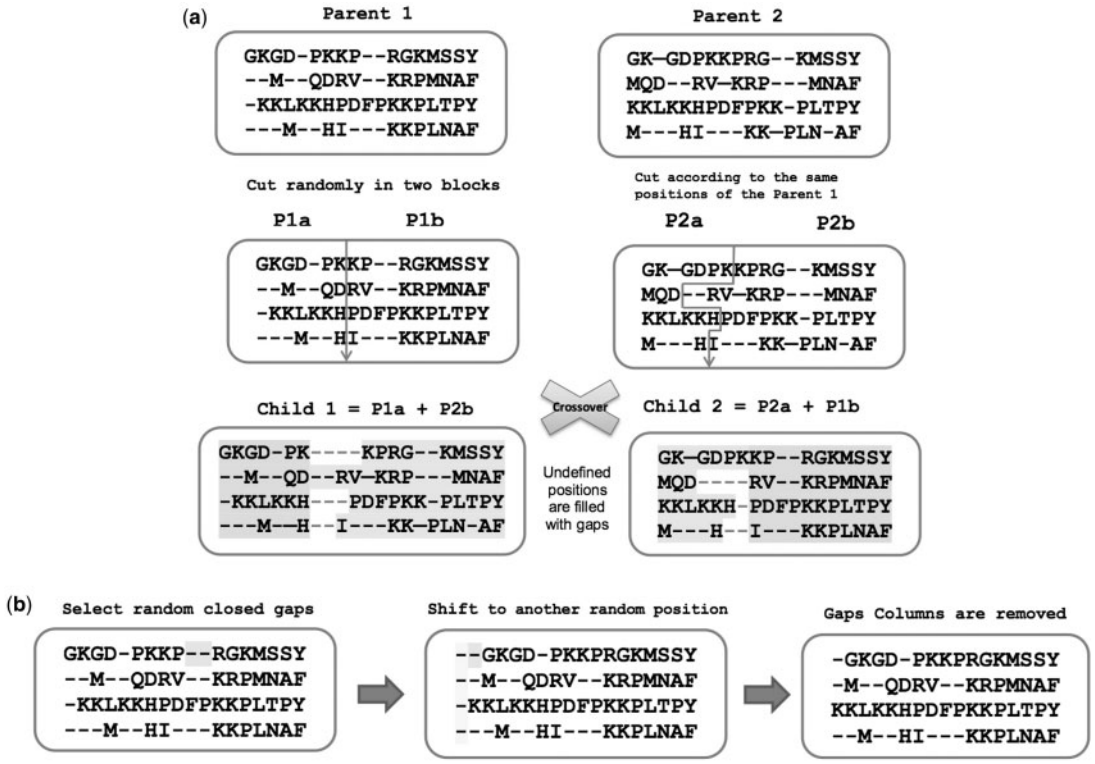


**Fig. 3.** (a) Single point crossover operator: The first parent is cut straight at a randomly chosen position. The second one is tailored so that the right piece can be joined to the left piece of the first parent and vice-versa. **(b)** Closed gap shifting mutation operator: closed gaps are randomly chosen and shifted to another position. Columns full of gaps are removed if they are found

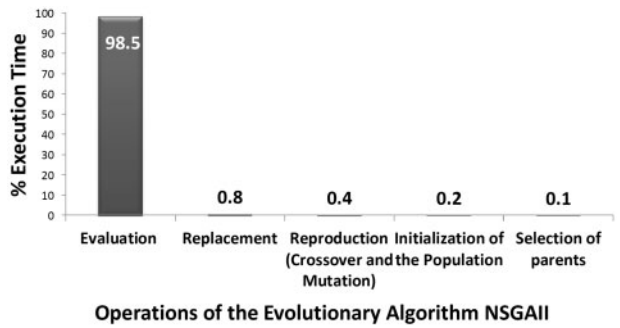


**Fig. 4.** Serial execution time profile of NSGA-II when solving an MSA problem instance

For this reason, we have made a study to identify those operations that contribute in a significant way to the overall execution time. For this purpose, Figure 4 shows the average serial time profiles for the algorithm NSGA-II when solving the BAliBASE instance BB20001. In this figure, we detail the time percentages spent on the following operations of the algorithm: solution evaluation, replacement, reproduction (given a set of selected parents, crossover and mutation operators are applied to generate new individuals), creation of the initial population, and selection operator.

These figures indicate that the parallel approach used in M2Align, evaluating the solutions in parallel, is justified since the evaluation is clearly the most time consuming task.

The parallel scheme adopted in M2Align has also the advantages of: first, do not requiring any change in the original algorithm and, second, the behaviour of NSGA-II remains unchanged.

### 2.7 Implementation details

M2Align has been developed by using the jMetal framework for multi-objective optimization (Nebro *et al.*, 2015). The object-oriented architecture of jMetal has allowed to re-use its NSGA-II implementation.

The developing of M2Align has required to include the codification scheme of MSA solutions based on the specification only of the groups of gaps into the sequences, the crossover and mutation operators, and the implementation of the scores. To facilitate the specification of the particular objectives to be optimized, M2Align provides a generic MSA problem template that can be easily instantiated with any of the included scores.

The parallelization scheme used in M2Align is also taken from jMetal. The evaluation of the solutions of the population of a metaheuristic is a self-contained procedure in most of jMetal metaheuristics (i.e, they have a `evaluatePopulation(List<Solution>)` method), which includes an instance of a class called `Evaluator`, being this object the responsible of evaluating all the solutions. The idea is to use a multi-threaded evaluator that is able to perform all the solution evaluations in parallel in multi-core based computers.

## 3 Results and discussion

To assess the performance of M2Align, we have chosen the Benchmark Alignment dataBASE (BALiBASE v3.0) (Thompson *et al.*, 2005) which contains 218 sets of sequences (extracted from the PDB) that are prepared to be aligned by MSA approaches. We have defined the sets of sequences in six subsets according to their families and similarities RV11, RV12, RV20, RV30, RV40 and RV50; each group presents different biological characteristics. The experiments have been carried out over a multi-core system

**Table 2.** Parallel performance evaluation of M2Align (time units are hours) over 218 instances of the BAliBASE v3.0

| Family | $T_1$ | 4 cores | | | 10 cores | | | 20 cores | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $T_4$ | $Sp$ | $Ef$ | $T_{10}$ | $Sp$ | $Ef$ | $T_{20}$ | $Sp$ | $Ef$ |
| RV11 | 9.05 | 2.54 | 3.56 | 89% | 1.27 | 7.15 | 72% | 0.99 | 9.12 | 46% |
| RV12 | 4.58 | 1.34 | 3.43 | 86% | 0.73 | 6.29 | 63% | 0.63 | 7.22 | 36% |
| RV20 | 7.78 | 2.30 | 3.38 | 84% | 1.20 | 6.49 | 65% | 0.94 | 8.25 | 41% |
| RV30 | 6.55 | 1.88 | 3.48 | 87% | 0.99 | 6.61 | 66% | 0.79 | 8.28 | 41% |
| RV40 | 9.14 | 2.70 | 3.39 | 85% | 1.50 | 6.11 | 61% | 1.25 | 7.33 | 37% |
| RV50 | 5.19 | 1.46 | 3.54 | 89% | 0.74 | 7.03 | 70% | 0.59 | 8.77 | 44% |

$T_1$, Sequential runtime; $Sp$, Speed-up ($T_1$ divided number of cores); ($Ef$), Efficiency ($Sp$ divided by the number of cores).

**Table 3.** Execution Time in minutes) of the sequential version of M2Align against the original version of MO-SAStrE when solving nine BAliBASE datasets

| Instance | Sequential runtime (min) | |
|---|---|---|
| | MOSAStrE | Sequential M2Align |
| BB11001 | 24.02 | 0.44 |
| BB11009 | 37.50 | 4.09 |
| BB11011 | 80.96 | 3.49 |
| BB11013 | 31.41 | 0.41 |
| BB12002 | 85.14 | 0.76 |
| BB12004 | 172.18 | 2.73 |
| BB12010 | 183.04 | 3.92 |
| BB12015 | 142.65 | 1.80 |
| BB30009 | 294.24 | 0.55 |

composed of 20 cores. M2Align has been configured in a similar way to MO-SAStrE, with a population size of 100 solutions and a total number of evaluations of 50 000. The mutation and crossover are defined with a probability of execution of 0.80 and 0.20, respectively. The initial population has been generated using the input datasets of pre-alignments computed by the MSA techniques specified in Table 1. The architecture selected for conducting these experiments was a 2-processor Intel Xeon CPU E5-2650 v3 of 10 cores at 2.3GHz and 25MB Cache running CentOS Linux 7.

### 3.1 Parallel performance

To evaluate the performance of M2Align, we have used two well-known metrics: speed-up and efficiency. The speed-up is defined by the sequential time divided by the parallel time, so if the number of processing units (e.g. cores) is $P$, this value should be the ideal time reduction to achieve. Related to the speed-up, the efficiency metric is computed by dividing the speed-up by the number of cores used; thus, an efficiency of 100% indicates a speed-up of $P$.

Table 2 shows the total computing time, speed-up and efficiency of M2Align when aligning the six families of set of sequences contained in BAliBASE v3.0, using 1 (sequential execution), 4, 10 and 20 cores.

We can observe that significant speed-up and efficiency values can be obtained. For example, in the case of the RV11 group, the sequential time ($T_1$) of 9.05 hours is reduced to 2.54, 1.27 and 0.99 h, with 4, 10 and 20 cores, respectively. In terms of speed-ups, the values can be up to 3.56 with 4 cores (efficiency of 89%), 7.15 with 10 cores (efficiency of 72%) and 9.12% (efficiency of 46%), and similar results are obtained for the rest of the families. More detailed information is included in the Supplementary Material, which contains the parallel results for each of the 218 instances of the benchmark BAliBASE.

**Table 4** Average scores for the 218 BAliBASE problems optimized by M2Align and 9 classical MSA techniques [the values of MO-SAStrE have been taken from Ortuño et al. (2013)]

| Method | M2Align objectives | | | BAliBASE Scores | |
|---|---|---|---|---|---|
| | STRIKE | TCC (%) | Non-gaps (%) | SP | TC |
| ClustalW | 1.54 | 1.70 | 55.39 | 0.67 | 0.29 |
| Muscle | 1.76 | 1.90 | 52.48 | 0.72 | 0.36 |
| Kalign | 1.75 | 1.85 | 48.04 | 0.73 | 0.36 |
| RetAlign | 1.72 | 2.10 | 49.38 | 0.71 | 0.33 |
| Tcoffee | 1.75 | 1.87 | 45.35 | 0.77 | 0.41 |
| ProbCons | 1.74 | 1.85 | 44.21 | 0.77 | 0.42 |
| 3D-Coffee | 1.80 | 1.64 | 42.27 | 0.72 | 0.39 |
| Mafft | 1.80 | 1.97 | 49.70 | 0.77 | 0.43 |
| FSA | 1.37 | 1.40 | 31.36 | 0.68 | 0.32 |
| BAliBASE | 1.79 | 1.94 | 52.16 | 1.00 | 1.00 |
| MO-SAStrE | 2.37 | 2.44 | **58.51** | 0.79 | — |
| **M2Align** | **2.44** | **2.45** | 58.13 | **0.81** | **0.46** |

With the aim of knowing the performance of the sequential version of M2Align over the original version of MO-SAStrE, we have carried out some executions of both techniques solving a selected number of BAliBASE datasets, setting the same conditions for a fair comparison. The tests have been carried out over the same computing system, using the same parameters and with the same input datasets of pre-alignments for the generation of the initial population. Table 3 shows the execution time (in minutes) of both techniques.

As we can see, M2Align performs a faster execution aligning each one of these datasets. For example, in the case of the BB30009 instance, MO-SAStrE needs 294.24 minutes, but the sequential version of M2Align only requires 0.55 min. The time difference are due mainly to the gap group based codification of the alignments, which allows an efficient implementation of the crossover and mutation operators.

### 3.2 Comparison with other MSA methodologies

In order to determine the accuracy of M2Align, we have compared it with other classical MSA techniques detailed in Table 4 by aligning all the 218 datasets of BAliBASE v3.0. This table shows the average scores of the three objectives (STRIKE, TCC and Non-Gaps) optimized by M2Align, and two scoring functions provided by BAliBASE, the sum-of-pairs (SP) and total-column (TC) scores, where unreliable regions are included in the reference. We have also included the results obtained with 3D-Coffee (Poirot *et al.*, 2004), a tool that is representative of methods using structural information.

We can see that M2Align generates more accuracy alignments according to STRIKE and TCC; the Non-Gaps% score is better performed by MO-SAStrE. If we take into account the SP and TC scores of BAliBASE, M2Align yields the highest scores if we exclude the BAliBASE reference values. We could not solve all the BAliBASE problems with the original MO-SAStrE code, so the values of this algorithm in Table 4 have been taken from (Ortuño *et al.*, 2013) but the TC score, which was not included in that article.

A more detailed comparison is shown in the Supplementary Material, where we have added two tables with the scores generated by M2Align and the other MSA techniques when aligning eight selected instances of BAliBASE v3.0.

The results reported in Table 4 shows that the numerical results of M2Align and the original MO-SAStrE algorithm are not the

same. There are some reasons explaining this fact. On the one hand, M2Align is implemented in Java and takes the NSGA-II algorithm provided by the jMetal framework, while MO-SAStrE is implemented in MatLab and uses the NSGA-II provided by that tool; this implies that some components (e.g. the random number generator) will not be the same. On the other hand, the description of the crossover and mutation operators in the MO-SAStre paper is very high level, remaining some implementation details unexplained. As a consequence, both algorithms do not have the same behaviour when solving MSA problems.

## 4 Discussion

The obtained parallel performance of M2Align on the BAliBASE problems indicates that important time reductions can be obtained with up to 20 cores, but the efficiency decreases with a higher number of cores. The main reason is that the working of M2Align alternates a parallel step (evaluating the solutions) with a sequential one (the rest of the algorithm). Furthermore, the parallel scheme is synchronous, which implies that the parallel step finishes when all the solutions have been evaluated. In the case of MSA, the solutions can have different length so their evaluation time may not be the same, what hinders having all the cores busy the 100% of the time (this effect is accentuated the greater the number of cores). Anyway, we have to consider that we are not altering the behaviour of the original algorithm and, in some cases, the speed-up with 20 cores can be higher than 11 (see problem BB20007 in Table 3 in the Supplementary Material), what it is an interesting outcome in practical terms.

We have made some pilot tests with more complex MSAs [we include an experiment in Section 4 in the Supplementary Material using problems included in Capella-Gutiérrez *et al.* (2009)], and in these situations the following issues must be taken into account:

- The number of gaps in the MSA has a strong influence in the performance of M2Align due to the encoding scheme used.
- Medium scale problems are difficult to solve in a reasonable amount of time using our multi-core computing system.

Medium and large scale MSAs are a challenge to tools such as M2Align. Dealing with these problems would require to change the underlying MO-SAStrE/NSGA-II algorithm and to use a more powerful parallel computing system. Some strategies to consider are incorporating local search strategies and using an asynchronous version of NSGA-II (Durillo *et al.*, 2008) to avoid the bottleneck of the sequential part of the algorithm and to have the cores working most of the time.

## 5 Conclusions

The alignment of multiple biological sequences can be a computational intensive task when the sequences are long and numerous, so an approach to cope with it is to take advantage of the parallelism potential provided by current multi-core computers.

We have proposed a tool called M2Align which includes a re-implementation of the MO-SAStrE MSA algorithm, but with a number of significant improvements, being the most remarkable one the parallel execution of this algorithm. Other differences include an efficient MSA encoding and the fact that M2Align is an Open Source project that is hosted in GitHub repository.

The results obtained reveal that significant time reductions can be achieved by using up to 20 cores when solving the datasets included in the BAliBASE 3.0 benchmark. Our experiments also indicate that our implementation of MO-SAStrE is clearly more efficient than the original one. Finally, a comparison against a set of alignment techniques reveals that M2Align provides the best overall results in the STRIKE and TCC scores.


## Acknowledgements

C.Z.-V. acknowledges Universidad Técnica Estatal de Quevedo (Ecuador) for supporting his doctoral stays at Departamento de Lenguajes y Ciencias de la Computación of Universidad de Málaga (Spain).

## Funding

This work has been partially supported by the Secretaría Nacional de Educación Superior Ciencia y Tecnología SENESCYT from Ecuador, and Spanish Grants TIN2014-58304-R (Ministerio de Economía y Competitividad), P11-TIC-7529 and P12-TIC-1519 (Plan Andaluz I + D + I – Junta de Andalucía). José García-Nieto is recipient of a Post-Doctoral fellowship of 'Captación de Talento para la Investigación' at Universidad de Málaga.

*Conflict of Interest*: none declared.


## References


Abbasi,M. *et al.* (2015). Local search for multiobjective multiple sequence alignment. In: Ortuño, F. and Rojas, I. (eds.), *Bioinformatics and Biomedical Engineering, Volume 9044 of Lecture Notes in Computer Science*. Springer International Publishing, Cham, pp. 175–182.

Bacon,D.J., and Anderson,W.F. (1986) Multiple sequence alignment. *J. Mol. Biol.*, **191**, 153–161.

Capella-Gutiérrez,S. *et al.* (2009) trimal: a tool for automated alignment trimming in large-scale phylogenetic analyses. *Bioinformatics*, **25**, 1972.

da Silva,F. *et al.* (2010) Alineaga a genetic algorithm with local search optimization for multiple sequence alignment. *Appl. Intell.*, **32**, 164–172.

da Silva,F.J.M. *et al.* (2011) Parallel Niche Pareto AlineaGA–lineaGA Niche Pamultiobjective approach on multiple sequence alignment. *J. Integr.Bioinformatics*, **8**, 174.

Dayhoff,M. *et al.* (1978) A model of evolutionary change in proteins. In: Dayhoff, M.O. (ed.) *Atlas of Protein Sequence and Structure*, Vol. **5**, National Biomedical Research Foundation, Washington, DC, pp. 345–352.

Deb,K. *et al.* (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, **6**, 182–197.

Do,C. *et al.* (2005) Probcons: Probabilistic consistency-based multiple sequence alignment. *Genome Res.*, **15**, 330–340.

Durillo,J.J. *et al.* (2008). A Study of Master-Slave Approaches to Parallelize NSGA-II. In: *IEEE Int. Symp. on Parallel and Distributed Processing, 2008 - IPDPS 2008*, pp. 1p.e. Miami, FL, USA.

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

Handl,J. *et al.* (2007) Multiobjective optimization in bioinformatics and computational biology. *IEEE/ACM Transactions on Computational Biology and Bioinformatics/IEEE, ACM*, Vol. **4**, pp. 279–292.

Henikoff,S., and Henikoff,J. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, **89**, 10915–10919.

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.

Kaya,M. *et al.* (2014) Multiple sequence alignment with affine gap by using multi-objective genetic algorithm. *Comput. Methods Prog. Biomed.*, **114**, 38–49.

Kemena,C. *et al.* (2011) Strike: evaluation of protein msas using a single 3d structure. *Bioinformatics*, **27**, 3385–3391.

Lassmann,T., and Sonnhammer,E.L. (2005) Kalign align mer,"http://www.ncbi.nlm.nih.gov/pubmed/2203920gorithm. *BMC Bioinformatics*, **6**, 9.

Nebro,A. *et al*. (2015). Redesigning the jmetal multi-objective optimization framework. In: *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion 'omp pp. 1093sniong New York, NY, USA. ACM.

Notredame,C. *et al*. (2000) T-coffee: a novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol*., **302**, 205– 217.

Novák,A.S.A. *et al*. (2010) Reticular alignment: A progressive corner-cutting method for multiple sequence alignment. *BMC Bioinformatics*, **11**, 1–19.

Ortuño,F. *et al*. (2013) Optimizing multiple sequence alignments using a genetic algorithm based on three objectives: structural information, non-gaps percentage and totally conserved columns. *Bioinformatics (Oxford, England)*, **29**, 2112–2121.

Poirot,O. *et al*. (2004) 3dcoffee@igs: a web server for combining sequences and structures into a multiple sequence alignment. *Nucleic Acids Res*., **32(Suppl 2)**, W37.

Rani,R.R., and Ramyachitra,D. (2016) Multiple sequence alignment using multiobjective based bacterial foraging optimization algorithm. *Biosystems*, **150**, 177– 189.

Roberts,R.B.A. *et al*. (2009) Fast statistical alignment. *PLoS Comput. Biol*., **5**, e1000392.

Rubio-Largo,A. *et al*. (2015). A hybrid multiobjective memetic metaheuristic for multiple sequence alignment. *Evolutionary Computation, IEEE Transactions on*, (99), pp. 1p. ,

Rubio-Largo,A. *et al*. (2016) Hybrid multiobjective artificial bee colony for multiple sequence alignment. *Appl. Soft Comput*., **41**, 157– 168.

Seeluangsawat,P., and Chongstitvatana,P. (2005). A multiple objective evolutionary algorithm for multiple sequence alignment. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO 'ECC pp. 477esedi New York, NY, USA. ACM.

Soto,W., and Becerra,D. (2014). A multi-objective evolutionary algorithm for improving multiple sequence alignments. In: Campos S. (ed.) *Advances in Bioinformatics and Computational Biology, Volume 8826 of Lecture Notes in Computer Science*. Springer International Publishing, Cham, pp. 73–82.

Thompson,J.*et al*. (1994) Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*., **22**, 4673–4680.

Thompson,J. *et al*. (2005) Balibase 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins*, **61**, 127–136.

Wang,L. and Jiang,T. (1994) On the complexity of multiple sequence alignment. *J. Comput. Biol*., 337–348. pages

Zhu,H. *et al*. (2016) A novel approach to multiple sequence alignment using multiobjective evolutionary algorithm based on decomposition. *IEEE J. Biomed. Health Inform*., **20**, 717–727.