EOA semestral work report:
Evolutional optimisation for multiple sequence alignment

Jan Lubojacký
České Vysoké Učení Technické
Fakulta Elektrotechnická
lubojjan@fel.cvut.cz

Problem introduction

You have multiple sequences coming from one virus and you want to align them against each other to obtain a consensus sequence for the virus. But the sequences contain many mutations such as rewrites, insertions and deletions, which make the alignment difficult, so we want to find an alignment which aligns the conserved regions (same in all / most of the sequences) on top of each other. For optimal pairwise alignments we can use dynamic programming. Using a similar approach for N sequences of lenght $\sim M$ is $O(M^N)$ and thus infeasible for more than a few sequences.

Classical approaches for Multiple Sequence Alignment (MSA)

Classical approaches largely rely on progressively aligning the sequences and then improving the generated alignment via iterative refinment, some of the most famous software packages for this are listed below.

Star alignment : Given all of the sequences create optimal pairwise alignments via dynamic programming. Select the one with the highest score, this forms the first MSA, now align all remaining sequences against this alignment, choose the highest scoring one and add it to the MSA. Continute until all sequences are aligned.

Clustal : Clustal class algorithms build the MSA by progressive pairwise alignments of the sequences and building a tree (similar to hiearchical clustering), which is used to guide the alignment. More advanced versions of this algorithm also use HMMs (hidden markov models) to improve the alignment.

MAFFT : Employs a progressive approach to align sequences and utilizes Fast Fourier Transform techniques to accelerate the process. Uses iterative refinement steps to improve the accuracy of the alignment. This involves refining the alignment and adjusting the guide tree. [1]

Evolutionary approaches

Solution representation

The solution is represented via a list with all the sequences. Each sequence contains either a nucleotide or a space (representing deleted nucleotides) the chars $[A, C, G, T, -]$. The sequences are represented via doubly linked lists to keep the insert / delete mutations efficent.

Fitness function

For computing the fitness of a solution we use **Mean column entropy**. For each column j in the solution we compute the entropy as

$$H_j = -\sum_i p_i \log_2(p_i)$$

and then take the mean entropy across all columns in the alignment as

$$f(\text{MSA}) = \frac{\sum_j H_j}{|\text{MSA}|}$$

Mutation operators
Since all the classical approches rely on is aligning the sequences by introducing or removing gaps in the sequences, this is what our mutation operators consist of.

- **RandomGapInsert** : randomly inserts a gap into a sequence
- **RandomGapDelete** : randomly delets a gap from a sequence
- **LenghtEqualizer** : After all the sequences are mutated they will likely be of different lenght so we repair the solution by, extending the shorter ones randomly with gaps until they are all the same length (so we have a valid MSA.) This is again done via a mutation operator that is applied after the previous ones.

Crossover operators
For crossover we use a custom operator named **Random Shuffle Crossover** that given two parent solutions iterates over them and creates a child by flipping a coin for each sequence and if $p < p_{\text{swap}}$ it takes the sequence from the first parent and else it takes the sequence from the second parent.

Selection operators
For selection we opt to use the tournament selection with the number of children matching the number of parents and tournament size at 15% of the inflated population 0.15(child size + parent size), this was chosen empirically to converse diversity in the population.

Solution
The final consensus sequence is simply calculated from the most frequent characters in each column.

Algorithms used

Local search: Simmulated Annealing
For local search we use simulated annealing, temperature and cooling rate were selected emprically so that the temperature decreases smoothly along the entire range of the iterations.

Evolutionary algorithm
Evolutionary algorithm uses

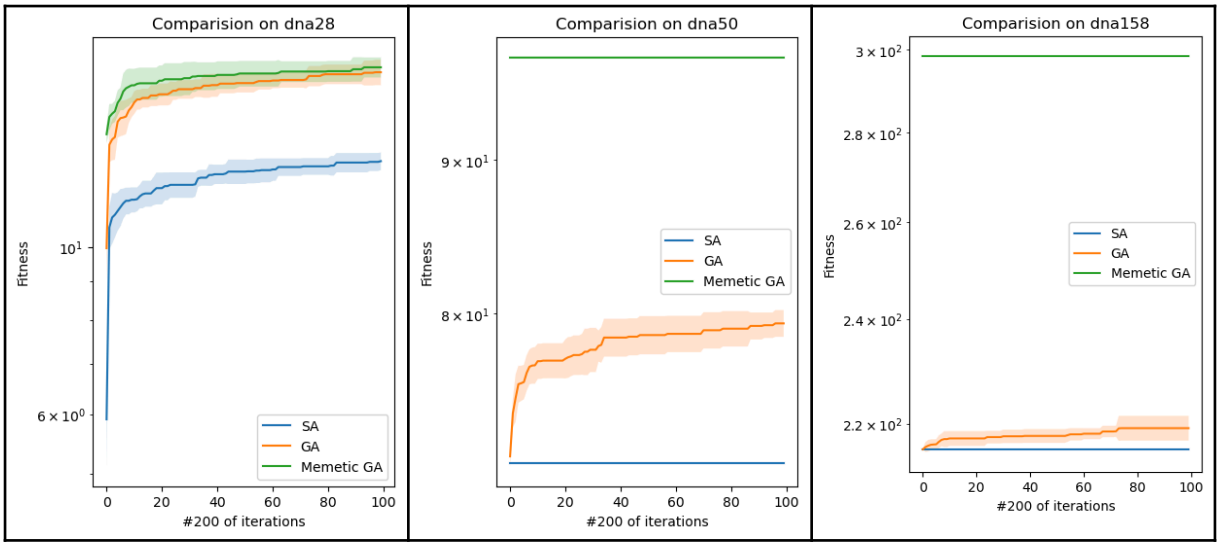
Memetic evolutionary algorithm
The idea behind this algorithm is that we use EAs for iterative refinement and combine them with some well known method. We start with a population of some alignments created by one (or multiple different) classical method and try and see whether we can improve them wrt. some metric.

For generating the initial alignment we use the MAFFT package [1]. And then use the same evolutionary algorithm from the previous section.

Results

Comparing algorithms
On the simplest problem where the solution space isnt that large the evolutionary algorithm is able to match the quality of the memetic algorithm and the memetic algorithm is able to improve the results of the alignment generated by MAFFT with simmulated annealing being worse than both.

On the larger problems simmulated annealing struggles heavily, evolutionary algorithm is able to reach at least some solution though it does not match the quality of the MAFFT generated alignment. And the memetic algorithm isn't able to improve the results of the MAFFT generated alignment. Graphs show the mean fitness values +/- std computed from 10 runs for each algorithm.



Discussion

Evolutionary approaches can't compare to MAFFT for longer sequences. We suspect that the issue lies in our mutation operators. Nearly all of the classical methods use different costs for creating and extending the gap (the creation being typically more expensive) so different operators are also used in literature and might be better than the simple ones we came up with. For example gap block insertion, gap block extenstion, gap block shift and gap block deletion [2].

Another possible improvement might lie in using a different fitness function for example the GLOCSA function which considers also the placement of the gaps in addition to the column homogenity. [3]

Implementation notes

As all my algorithms for this course all code for this semestral work is written by me from scratch in the go programming language (Except the MAFFT program). The entropy score in the graphs is calculated from entropy by flipping sign.

Bibliography

[1] K. Katoh, “MAFFT”. [Online]. Available: https://gitlab.com/sysimm/mafft

[2] E. D. Arenas-Díaz, H. Ochoterena, and K. Rodríguez-Vázquez, “Multiple sequence alignment using evolutionary algorithms”, *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, 2009, doi: 10.1145/1569901.1570159.

[3] H. &. K. R.-V. Edgar David Arenas-Díaz Ochoterena, “Multiple sequence alignment using a GLOCSA guided genetic algorithm”, *GECCO '08: Proceedings of the 10th annual conference companion on Genetic and evolutionary computation*, 2008, [Online]. Available: https://doi.org/10.1145/1388969.1388973