**Details about algorithms involved & to be implemented for solving MSA problem**

1. Linear Longest Common Subsequence (LCS)
2. Circular Longest Common Subsequence (CLCS)
3. Fragmented Two Layer Particle Swarm Optimization (FTLPSO)
4. Ant Colony Optimization (ACO)

**Linear Longest Common Subsequence**

It is the most common algorithm used to find the subsequence occurring in both strings, but not necessarily contiguous, Overlapping Substructure property can be used to avoid re-computation of same sub-problems by either using memoization ( or tabulation).

This algorithm has Time & Space Complexity of O( m x n ) ~ O(L2)

Where, ‘m’ is length of 1st string & ‘n’ is length of 2nd string

‘L’ is length of sequences, L = m = n , in general case, both sequences have same length

**Note:** Assume m >= n : this will help in some sort of the Optimization in next approach

**Circular Longest Common Subsequence**

Linear LCS cannot be directly used for DNA, RNA or protein due to its inability to detect sub-sequences which can occur in circular fashion, in bio-informatics this is a common problem, which is being neglected over a long period of time, as other approaches are developed for MSA problem

But CLCS can help determining similarity between sequences with fairly short sequence length, selected from the main sequences to be compared

Beside, having high computational complexity than iterative methods, CLCS can determine similarity between sequences in reasonable amount of time, which cannot be achieved using Simple LCS

Iterative algorithms till date are using String Matching or Simple LCS as their sub-routine, while we have used CLCS as sub-routine in ACO approach, to improve results generated by ACO, which will then be compared with FTLPSO with no such modification made, to retain idea last developed by author of base paper

Finding Circular LCS of two strings is even more complicated, as by naïve approach it will require O( m x n x n ) ~ O(L3) time, which is not a reasonable time for protein sequences, even if done pairwise.

Earlier, some authors have made attempt to develop an O( m x n ), algorithm for Circular LCS but have failed to give proof of the same, below are 2 links of similar attempts:

1. <https://www.researchgate.net/publication/45886632_An_On2_Algorithm_for_Computing_Longest_Common_Cyclic_Subsequence>
2. <https://arxiv.org/abs/0911.5031>

Below is an Example of 2 strings, which are same when considered in circular fashion, like simple proteins, & nature of both Linear & Circular LCS is compared:

Seq\_1 = “abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ“

Seq\_2 = “ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz”

Length of Simple LCS is 26, While Length of Circular LCS is 52

Finding Simple LCS takes 0.009433 Seconds

Finding Circular LCS takes 0.010010 Seconds

I have developed successfully a simple yet powerful algorithm to find out Circular LCS, in the same Time & Space Complexity of O ( m x n ) ~ O(L2), actually by multiplying by a factor of 2 in worst case, which doesn’t affect the runtime of both algorithms, as much as, it has impact on results, CLCS is introduced as it is to be used as sub-routine in next two iterative algorithms for comparative studies.

**Note:** Shift from standard to other algorithms has been made due to heavy time & memory requirements of Dynamic programming for large protein sequences.

Progressive methods like Rubber Band technique & Hidden Markov Models have been used over period of time, but have been proved to fail under certain conditions to reach optimal results (even significantly near optimal results)

Dynamic Programming for k-sequence multiple sequence alignment have been proved to take O( Lk ) Time & Space.

Hence due to this exponential growth, it has been proved to be **NP-Complete** in nature, making it more tantalizing to finding optimal answers in reasonable amount of time.

**Particle Swarm Optimization**

“Kennedy and Eberhart” (1995) first introduced particle swarm optimization (PSO) in 1995. It mimics the social behaviour of bird flocks or fish schools. Every time, each particle in the swarm flies to its next position with a specific speed depending on its achieved best position, and the global best position reached by any particle in the swarm.

**Problem associated with standard PSO**

The standard PSO suffers from the trapping in local optima, and its disability to do better alignment for longer sequences.

**How longer sequences are reduced, so that PSO can perform well**

To overcome these problems, K-Tuple is used as a fragmentation technique to divide the longer datasets to a number of fragments.

**Steps taken when, Each longer sequence is divided into shorter sequences**

Then a two-layer PSO algorithm is applied to align each fragment, which has ability to deal with unconstrained optimization problems and increase diversity of particles.

In the base paper chosen, FTLPSO is proposed. A fragmentation technique based on k-tuple is applied to shorten the long sequences to small sub-sequences, aiming to solve problem with used of PSO.

**Problem, with K-Tuple algorithm & its Proposed Solution**

Standard K-Tuple algorithms, takes 1st Sequence as Query & Generates the table, to obtain fragments of each long sequence to be aligned by TLPSO using that table.

But it can be case that, taking any other sequence could result into more uniformly sized fragments being formed.

To remove this shortcoming, we have proposed an approach, to execute K-Tuple algorithm, for each sequence being taken as a query for rest of the sequences, if will result in ‘NS’ (No. of Sequences) tables being formed, best of which can be taken.

To find best diversified table among all of these table, a formula is obtained.

Each Table has a “score” associated with each, which is calculated as

Score = Sum of squares of pairwise difference of fragment sizes for all fragments being formed

The Table having lesser score, is considered to be more diversified. Due to this modification being made into K-Tuple algorithm, the table formed upon which PSO has to work, is more uniform, & better results can be obtained in future

Enter File Containing DNA Sequences to be aligned using FTLPSO

Test.txt

Enter Number of DNA Sequences & length to be taken into consideration

3 15

Enter Value of (K,Δ) for k-tuple table generation4 1

Sequence Taken = 1, Score = 26.683 & Table having that score is

[ 0, 0, 0, 0]

[ 7, 1, 2, 4]

[ 8, 6, 8, 5]

[14, 14, 14, 0]

Sequence Taken = 2, Score = 26.077 & Table having that score is

[ 0, 0, 0, 0]

[ 7, 5, 2, 4]

[ 8, 6, 8, 5]

[14, 14, 14, 0]

Sequence Taken = 3, Score = 25.179 & Table having that score is

[ 0, 0, 0, 0]

[ 8, 6, 0, 4]

[ 9, 7, 9, 4]

[14, 14, 14, 0]

Above example shows that, table obtained using 3rd sequence as Query is more diverse.

Then, TLPSO is applied as a suitable variant for MSA problem.

The proposed algorithm by the author is divided into two main steps:

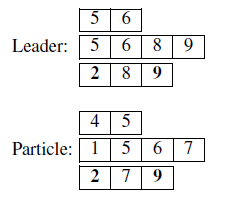
1. Fragmentation process, to shorten the longest sequences, make working of PSO easier. K-tuple method is selected as a fragmentation technique & further modified to overcome its shortcoming, which can result in non-uniform fragments sizes. By the end of fragmentation process, an index table for fragment position is created similar to what is created using simple K-Tuple algorithm.
2. PSO does alignment process on fragments. Two layers of swarms are created, as the swarms in each layer will focus on one of the two scoring functions being taken (Column-Score & Sum-Of-Pair) of the MSA to appraise their performance. In each swarm, local-PSO is used as a good solution to achieve divergence. Also a mutation is applied on the best particle every iteration to reduce the risk of falling in a local optima.

Note: Modification in Fragmentation process is done to the approach taken by author has been made, to help study clear comparisons between approach of author taken as reference & our developed ACO system using CLCS as sub-routine (referred as ACO-C) is then to be compared with FTLPSO for analysis of result, direct comparison will be made with results of direct LCS & CLCS.

**CORE ISSUES RELATED TO PSO**

Distance between two particles is computed as

Below is an example showing a Particle & Best particle in its swarm



They have only 4 matched Gaps, out of total of 18 Gaps in total of both particles, So Distance = (18-4)/18 = 0.77

Mention that 0<=Distance<=1

So crossover point is selected between range 1 and 0.77\*9

Mutation is done, only the best particle of Swarm after each iteration, to prevent it from trapping into Local Optima.

Distance & Velocity Matrix are initialised to all Zeros, which are both then updated, as a function of Distance between particle of swarms after every iteration of PSO.

**Ant Colony Optimization**

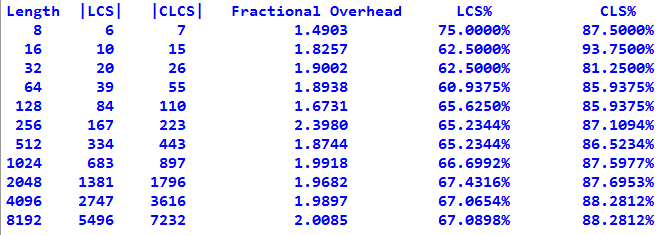
Each ant picks up a subsequence from a sequence & drift in an interval untial a close match is found to it, in all other sequences, strengthening a pheromone trail. As the algorithm runs larger fragments of sequence are picked up by the ants. This is designed to encourage the removal of extraneous gaps later on in the process.

A flow chart of this approach is given in the paper, our contribution to this approach will be the substitution of “CLCS” instead of “String Matching”, which lead to faster convergence & better results achieved with shorter sub-sequences of main sequences taken to be aligned

Hence, we can achieve better Alignment score measures, suggested by author of base paper, within lesser iterations & computational resources being employed in solving the problem

**Comparison between LCS & CLCS approach**

Results are shown for same sample of DNA, taken different length at a time, both length & percentage of match found is shown in table, along with fractional overhead involved in finding CLCS over LCS on same length



Above result presents clear reason, why I am willing to use “CLCS” as subroutine in ACO, instead of “String Matching” or “LCS”

Results obtained using ACO-C are also to be compared with simple ACO & CLCS

& expected order of outcome of result in terms of performance is

(ACO or FTLPSO ) >> LCS >> ACO-C >> CLCS

**Note:** CLCS cannot be used in FTLPSO as a sub-routine, as it doesn’t allow for any sub-routine inside its algorithm, like ACO.

**Conclusion:**

To obtain better results even for longer sequences, ACO-C will be the proposed approach, due to its greater performance over ACO, FTLPSO & LCS.