CSC530 Project Proposal

Title

Applying nature inspired ant colony optimization algorithm to solve the multiple sequence alignment problem.

Team

Jagtap, Jay (jjjagtap, 200311438)
Kulkarni, Omkar (oskulkar, 200316383)
Kulkarni, Poorva (pnkulkar, 200312183)
Mandliya, Amit (amandli, 200312955)

Executive Summary

Multiple Sequence Alignment (MSA) is a process through which study of closely related genes becomes easier. The problem is to find an optimal set of sequences between multiple sequences of DNA, RNA or Proteins. It is very helpful to understand the evolutionary relationships between different genes[1]. This project implements a nature inspired metaheuristic- Ant Colony Optimization to solve the MSA problem in high-performance Julia language. We will be working on converting the MSA problem to Travelling Salesman Problem (TSP). TSP is a classic example of NP-Hard combinatorial problems in Computer Science. A lot of optimization algorithms are available to solve the TSP problem. Ant Colony Optimization (ACO) is one of the nature inspired metaheuristic which solves the TSP problem. ACO is a population based metaheuristic and can be used as approximation solutions to combinatorial problems [2]. In the evaluation phase, we will be comparing the results to the existing implementations of the ACO algorithms. The metrics for the evaluation phase will be time taken to do the same approximation, memory utilization.

Introduction and statement of need

Multiple Sequence Alignments (MSAs) are widely used in evolutionary genomic studies. It is of high importance that the results of MSA should be of good accuracy as it is used in the early stages of the studies. MSA problem is NP-Hard in computer science [4]. A brute force algorithm to solve MSA might take billions of years to complete. Pairwise sequence alignment is used to find regions of similarity between two DNA or protein sequences [5]. Pairwise alignments can be solved in polynomial time by various algorithms like dynamic programming approach [6]. There has also been a lot of research done on finding good algorithms to solve the problem of MSAs [7].

The interestingness of our project lies in the idea of finding an approximate solution of the MSAs by implementing nature inspired metaheuristic Ant Colony Optimization. The algorithm converts the MSA input to TSP graph and then applies the Ant Colony Optimization to solve the TSP problem. We plan to compare our solution with Dantzig-Fulkerson-Johnson algorithm [8][9]. The potential benefit that can result from our algorithm is improvement in the performance of the currently available MSA problem. Also, a Julia package providing the implementation of ACO in solving MSAs can be used as a base package for further optimization.

The travelling salesman problem (TSP) is to find the shortest possible route a salesman would follow to visit a city exactly once and come back to the origin city. The input to the problem is a graph where each city is represented by the node and the path from one city to another is represented by an edge between two nodes. TSP problem is a classical example of NP-Hard problem and there are many optimized algorithms to solve the TSP problem. We are using one such algorithm to solve the TSP problem. The MSA sequences can be converted into a TSP graph in polynomial time as described in the later section of this document.

Ant Colony Optimization is a metaheuristic algorithm used to solve complex optimization problems based on the nature of real ants and their pheromone based communication methods. A pheromone based model is used to calculate the heuristics of artificial ants. A pheromone is an organic compound deposited by a real ant on its path based on the amount and quality of food found on that path. Each ant moves at random, more pheromone increases the probability of an ant choosing that path. Based on pheromone levels other ants decide whether to follow the given trail. In the same way, the simulated ants record their positions and solutions and heuristic based on their solutions, so that in subsequent iterations more ants find better solutions. For applying ACO, any optimization problem is transformed to a problem of finding the best path on a weighted graph. The simulated ants iteratively build solutions by moving stochastically on the graph and recording solutions based on a set of parameters like nodes(cities) or edges(weights) of the graph.

The Multiple sequence alignment (MSA) is NP hard. We propose to use the Ant Colony Optimization algorithm to solve the MSA problem. This can be done by transforming MSA to Travelling Salesman Problem (TSP). Our approach to achieve this will be based on the paper [6].

Project Description

The project involves implementing the Ant Colony Optimization Algorithm in Julia. We are going to solve the Multiple Sequence Alignment (MSA) problem by converting the MSA to TSP problem in the first phase. The TSP is a NP-Hard problem, in order to solve the problem in polynomial time, we need to find the approximate solution which is solvable in polynomial time. In this project, we are using Ant Colony Optimization (ACO) which is solvable in polynomial time to find a solution for MSA.

Algorithms:

Phase 1- Conversion of the Multiple Sequence Alignment to Travelling Salesman Problem.

1. For each input sequence, compute the pairwise similarity between all the possible combinations of sequences. There are (n(n-1)/2) number of combinations possible which need to be evaluated. Hence this phase has O(n^2) time complexity.
2. Now, generate a fully connected graph G(V,E) where V = each sequence in the input, E = The similarity between the pairs of vertices. This step will need to perform the computation (n(n-1)/2) times. Hence the complexity of this step is O(n^2).

Phase 2 - Solving the Travelling Salesman Problem using Ant - Colony Optimization

1. An ant starts from a starting vertex and based on pheromone levels selects a path to neighbouring vertices.
2. An ant will move from node i to node j with probability:

   $$p_{i,j} = \frac{(\Gamma_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\Sigma(\Gamma_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}$$

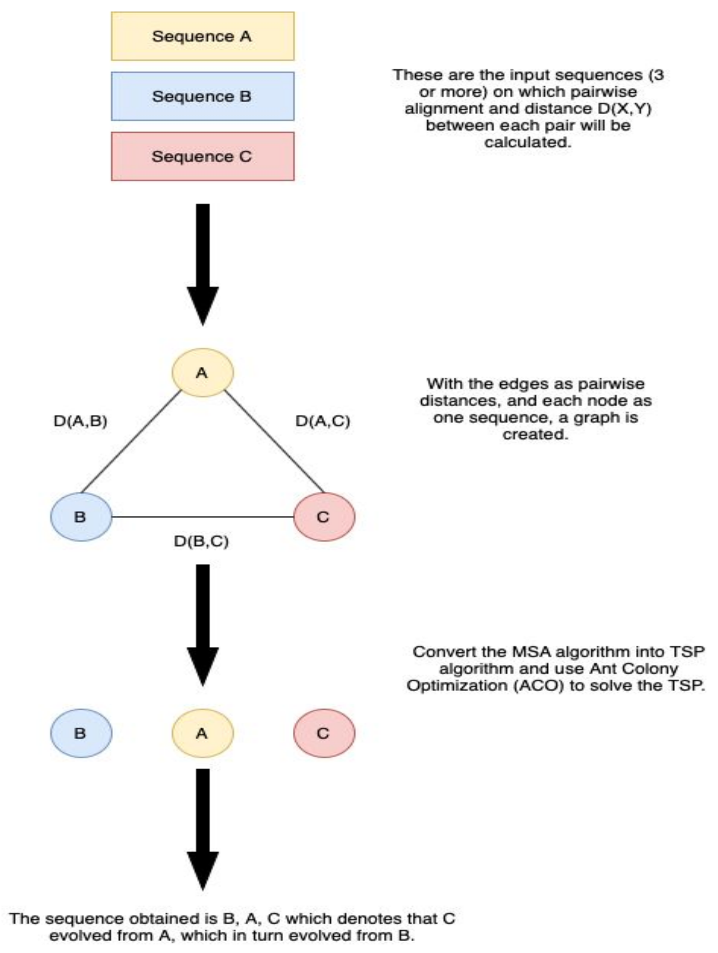
   where,
   $\Gamma_{i,j}$ is amount of pheromone on edge
   α is a parameter to control bias of $\Gamma_{i,j}$
   $\eta_{i,j}$ is desirability of edge
   β is a parameter to control bias of $\eta_{i,j}$

3. Amount of pheromone is updated according to the equation

   $$\Gamma_{i,j} = (1-\rho)\Gamma_{i,j} + \Delta_{\Gamma_{i,j}}$$

   where,
   ρ is amount of pheromone deposited
   Δ is amount of pheromone deposited
4. Pheromone values are updated by all the cants completed the tour. Once an ant completes a tour, we get a solution.
5. All solutions are tested for optimality.

A diagrammatic explanation of the phase 1 is shown below with a simple example of three sequences:


Sequence A
Sequence B
Sequence C
These are the input sequences (3 or more) on which pairwise alignment and distance D(X,Y) between each pair will be calculated.
A
B
C
D(A,B)
D(A,C)
D(B,C)
With the edges as pairwise distances, and each node as one sequence, a graph is created.
Convert the MSA algorithm into TSP algorithm and use Ant Colony Optimization (ACO) to solve the TSP.
B
A
C
The sequence obtained is B, A, C which denotes that C evolved from A, which in turn evolved from B.

Evaluation:

The evaluation of the algorithm can be done by comparing the metrics of computing Ant Colony Optimization for MSA in Julia with Python.
These metrics could be:

1) Total time required to run
2) Total memory required to run

There have been multiple attempts made to solve MSA using various traditional and nature inspired algorithms. Another way of evaluating the algorithm is to analyze the factors that make our approach of finding MSA better or worse than the existing approaches.

The correctness and quality of the results could be measured with respect to online tools available for Multiple Sequence Alignment. These tools are chosen in relation to the size of the input sequences varying from small to medium, along with medium to large alignments. A brief overview of the tools could be found at [10].

Experimentation Setup:

In this project we are planning to compare the performance of proposed algorithm in Julia and python. This experiment will help us to compare the performance benefits of various languages in terms of memory usage and runtime of algorithms.

To perform the benchmarking of the proposed algorithm we are planning to use the following standard dataset from Clustal Omega dataset[11].

The Experiments will also cover the comparison of the proposed algorithm with the existing Online tool which can be used to generate multiple sequence alignment.

Result:

The result of the proposed approach will contain the comparison of the proposed algorithm with existing algorithms and online tools. We will compare the memory usage of the Proposed algorithm implemented in Julia and Python. The result will focus on the change in the computation time and memory usage of the algorithm for varying input data size.

In addition to this the result will also contain the comparison of results produced by the proposed algorithm with the result produced by existing tools[10] for the same input data.

Timeline:

09/01/2020 - 09/18/2020: Proposal and research phase
09/20/2020 - 10/15/2020: Implementation & Testing phase
10/16/2020 - 10/25/2020: Evaluation phase
10/25/2020 - 11/05/2020: Presentation phase

References:

[1] Applied Mycology and Biotechnology (Yeisoo Yu, Sangdun Choi)
https://www.sciencedirect.com/topics/medicine-and-dentistry/multiple-sequence-alignment

[2] Ant colony optimization (Marco Dorigo)
http://www.scholarpedia.org/article/Ant_colony_optimization

[3] Performance Evaluation of Ant Colony Optimization Algorithm and Genetic Algorithm in Travelling Salesman Problem (Fenwa Deborah, Ibrahim Adeyanju, Olajide Olusegun Adeosun)
https://www.researchgate.net/publication/269987267_Performance_Evaluation_of_Ant_Colony_Optimization_Algorithm_and_Genetic_Algorithm_in_Travelling_Salesman_Problem

[4] Progressive multiple sequence alignment with indel evolution (Massimo Maiolo, Xiaolei Zhang, Manuel Gil & Maria Anisimova)
https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2357-1

[5] Pairwise Sequence Alignment https://www.ebi.ac.uk/Tools/psa/

[6] Using traveling salesman problem algorithms to determine multiple sequence alignment orders (weiwei zhong)
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.7679&rep=rep1&type=pdf

[7] Improved Particle Swarm Optimization Algorithm for 2D Protein Folding Prediction (Xiaolong Zhang,Tingting Li) https://ieeexplore.ieee.org/document/4272501

[8] Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems (David Applegate, Robert Bixby, Vašek Chvátal & William Cook)
https://link.springer.com/article/10.1007%2Fs10107-003-0440-4

[9] Travelling salesman problem using an Dantzig-Fulkerson-Johnson algorithm
https://github.com/ericphanson/TravelingSalesmanExact.jl

[10] Multiple Sequence Alignment https://www.ebi.ac.uk/Tools/msa/

[11] Clustal Omega https://www.ebi.ac.uk/Tools/msa/clustalo/