02-712 Biological Modeling and Simulations: Homework 1

Thomas Zhang

September 2023

Question 1

a)

We can model the problem as such:
Input: A set of plant species $P = p_1, ..., p_n$ and a set of collections $C = c_1, ..., c_n$ where each collection $c_i \subseteq P$. Each collection represents one of the plots within the forest.
Output: k sets of collections $s_1, ..., s_k$ where $s_j \subset C$ such that each collection c_i is in exactly one set s_j and all distances between collections in the same set is minimized.
Objective:

$$\underset{y}{argmin} \qquad y = \sum_i \sum_j dist(c_i, c_j)\mathbf{I}(c_i, j_i) \qquad \mathbf{I} = \begin{cases} 1 & c_i \in s_k, c_j \in s_k \\ 0 & c_i \in s_k, c_j \notin s_k \end{cases}$$

The exact solution to this can be found by solving the weighted max k-cut problem. Let $G = (V, E)$ be a fully connected graph. Each of the vertices v_i represents one of the plots c_i. The weight of the edge $w(v_i, v_j)$ is equivalent to the distance between the plots $dist(c_i, c_j)$ that those vertices represent. The distance between two plots can be represented as the number of different species between the two plots. Finding the weighted max k cuts of G is finding the cuts such that G is split into k subsets where the distance between each of the subsets is maximized. This implies that the distance between each of the plots in a subset is minimized because if the plot were to be closer to the plots in a different subset, which contradicts the fact that we found the max k cut. Thus each of the subsets would represent a ecological niche.

A possible heuristic approach to solving this problem would be to use K-means clustering. Each of the plots (collections) can be represented as a vector, v. For the vectorized representation x of a cluster c, x_i is 1 if plant species $p_i \in c$ and 0 if $p_i \notin c$. Using this vectorized notation allows us to assign a location to each of the plots in n dimensional space. We can then run a clustering algorithm (such as k-means) to identify k-ecological niches. The distance between two clusters would be the Euclidean distance between their vectorized representations.

b)

Input: A graph $G = (V, E)$. $V = \{v_1, ..., v_n\}$ where there is one vertex v_i for each neuron. The edge (v_i, v_j) has weight equal to the frequency the of the synapse. One of the vertices will represent the starting neuron and one other vertex will represent the distant neuron that is activated due to a stimulus at the starting neuron.
Output: A path $P = \{s, p_1, p_2, ..., p_k, t\}$ where s is the vertex representing the neuron at which the stimulus occurs and t is the neuron that exhibits the response.
Objective:

$$\underset{y}{argmax} \qquad y = dist(s, p_1) + dist(p_k, t) + \sum_{i=1}^{k} dist(p_i, p_{i+1}))$$

$$len(y) \leq k$$

This question can be formulated as a maximum weighted path problem with no cycles. Synapses that are used more frequently are more likely to be the on the neural path from the stimulated neuron to the response neuron. Thus finding the max weighted path with no cycles and length at least k would represent the most likely path of neurons from the start neuron to the end neuron.

c)

Input: A fully connected graph $G = (V, E)$ where each vertex v_i represents one of the supply stations that the scientists requires. The weight of edge (v_i, v_j) is the distance between the stations represented by v_i and v_j.
Output: A path $P = \{p_1, ..., p_n\}$ that visits all of the vertices in V exactly once. This path represents the order of stations that the bot should travel to.
Objective:

$$\underset{y}{argmin} \qquad y = \sum_i dist(p_i, p_{i+1})$$

This question can be formulated as a traveling sales person problem. Each of the supply stations that need to be visited can be represented by a vertex. Each of the edge weights correspond to the time it takes the robot to travel from one station to another station. Then, assuming it takes constant time to gather supplies at each station, the solution to the traveling sales person will represent the most efficient way to pick up all the necessary items from the supply stations.

d)

This problem can be formulated as a minimum set cover problem as follows:
Input: A collection of elements $A = a_1, ..., a_n$ and collection of sets $S = s_1, ...s_m$ such that $s_i \subseteq A$. Each element represents one of the patience who are suffering from the genetic disease. Each of the sets represent one of the genetic mutations, and the elements of the set are all the patients who have that mutation.
Output: A collection of sets $c \subseteq S$ such that all the elements in A are in at least one of the sets in c.
Objective:

$$\underset{c}{argmin} \quad size(c)$$

Where $size(c)$ is the number of sets in the collection c. Then the minimum set cover represents the minimum number of genetic mutations that can possibly explain the disease since each of the patients will have at least one of the mutations in the minimum set cover.

e)

This problem can be formulated as a longest common sub-sequence problem.
Input: A set of strings $W = w_1, ..., w_n$. Each of the strings represents one of the medical records.
Output: A sequence, S, such that $S \in w_1, w_2..., w_n$. The sub-sequence represents a common phrase found in W.
Objective:

$$\underset{S}{argmax} \quad size(S)$$

The longest common sub-sequences will represent the longest phrase that all of the reports share. Depending on how many phrases we want we will instead take the k longest common sub-sequences rather than just the longest common sub-sequence.

Question 2

a)

We can represent the problem as a graph $G = (V, E)$. Each of the vertices in V represent one of the experiments that need to be run. There will be an edge (v_i, v_j) if the experiments represented by v_i and v_j require least one piece of common equipment. Then finding the minimum number of colors such that no adjacent nodes are the same color will provide the optimal batches of experiments to run. Vertices of a single color represents the experiments that should belong to the same batch. This is a solution because none of the vertices of a single color can be connected or else that coloring of the graph G would not be a valid coloring solution.

b)

If there are a small number of experiments and a lot of common equipment pieces then this algorithm is potentially bad because each vertex, (which represents an experiment), will have many edges and depending on the order of chosen nodes, a severely sub-optimal coloring can occur. If there are a large number of experiments that share few pieces of equipment then then the algorithm is likely to be okay since the number of edges that each vertex has is minor compared to the total number of vertices there are, thus the ordering of chosen vertices is less likely to produce a severely sub-optimal coloring.

c)

The proposed solution will be very bad if there is a large number of experiments because enumerating over all possible orderings of n experiments will take $n!$ time. This makes the problem computationally intractable very quickly as n grows. With small number of experiments this algorithm is fine because it will find at least one possible solution.

Question 3

a)

input: A unweighted directed graph $G = (V, E)$ where each of the vertices in the graph represent a one of the reads in the set of sequence reads R. For each pair of reads r_i and r_j, there is a directed edge (v_i, v_j) in E if the suffix of r_i sufficiently matches the prefix of r_j.
output: Does there exist an path containing no cycles from the vertex representing the read s to the vertex representing read t of at least length k.

b)

Suppose we have the set of potential paths starting from s of at least length k with no cycles. Then we can check if there exists a solution in polynomial time by simply iterating through the each chain of reads and making sure that at least one of the chains properly overlaps from beginning to end, contains no cycles, and that the chain of reads ends at the read t. The time complexity to check this would be $O(n|k|)$, where n is the number of chains in the set of potential paths, since each read would have to be checked once and read in a path is checked at most one time.

c)

Let P be the Hamiltonian path of some graph, $G' = (V', E')$. Let the graph $G = (V, E)$ be a copy of G'. Then transform G so that each of the edges are converted into two separate directed edges going in opposite directions, that is convert each edge (v'_i, v'_j) in G' has two corresponding directed edges in G, (v_i, v_j) and (v_j, v_i). Then P corresponds to a chain of overlaps with length $k = V - 1$ from p_1 to p_k if each vertex in the graph G represents a read and each directed edge (v_i, v_j) represents a pair of reads such that the prefix of v_i sufficiently overlaps the prefix of v_j.

This reduction of G' into G takes $O(V + E)$ time since each vertex and edge is iterated over once. The vertices are simply copied, while the edges are split into two opposite direction directed edges.

d)

The order of vertices in the Hamiltonian path corresponds to a solution to the RNA sequence assembly. Since the vertices in G' are copied into G, then the order of vertices in G' that make up the Hamiltonian path corresponds to the exact same order of vertices in G that form a path from p_1 to p_k. And since each vertex in G represents a read, we then have a chain of at least $V - 1$ reads to connect the first read in the sequence P to the last read in the sequence P. So if we consider p_1 to represent s and p_k to represent t, then we have solved the RNA assembly problem for $k =$ number of reads -1.

e)

Since a solution to the problem can be checked in polynomial time (as shown in part b) and the Hamiltonian path problem, which is known to be NP-Complete, is polynomial time reducible to the RNA assembly problem (as shown in c and d) then the RNA assembly problem is an NP-complete problem.

f)

One possible strategy for a heuristic would be to begin random walks from the starting read s. Then if the walk encounters a dead end before reaching at least k vertices or if the walk encounters t before k vertices have been traveled to, then the algorithm backtracks and takes another path.

Another approach could be to progressively decrease the requirement for the suffix/prefix matching for two reads to be considered consecutive as the length of the path increases. The result of using such an approach would cause the resulting RNA sequence prediction to be less accurate the further away the location gets from the starting sequence s.

Question 4

a)

The problem can be formulated as a minimum set cover problem.
Input: Let each element w_i in a set of set of elements $W = w_1, ..., w_n$ represent exactly one of the reads. Let each subset $s_i \subseteq W$ from a collection of subsets $S = s_1, ..., s_m$ represent exactly one whole genome. $w_j \in s_i$ if the read that w_j represents is found in the whole genome that is represented by s_i
Output: A subset of S, T, such that at each element in W is in at least one subset t_i in T'. Each t_i is unique and corresponds to exactly one subset in the collection of subsets S.

$$T \subseteq S$$

$$\forall i \quad w_i \in T$$

Objective: Minimize the number of subsets in T.

$$\underset{T}{argmin} \quad |T| = size(T)$$

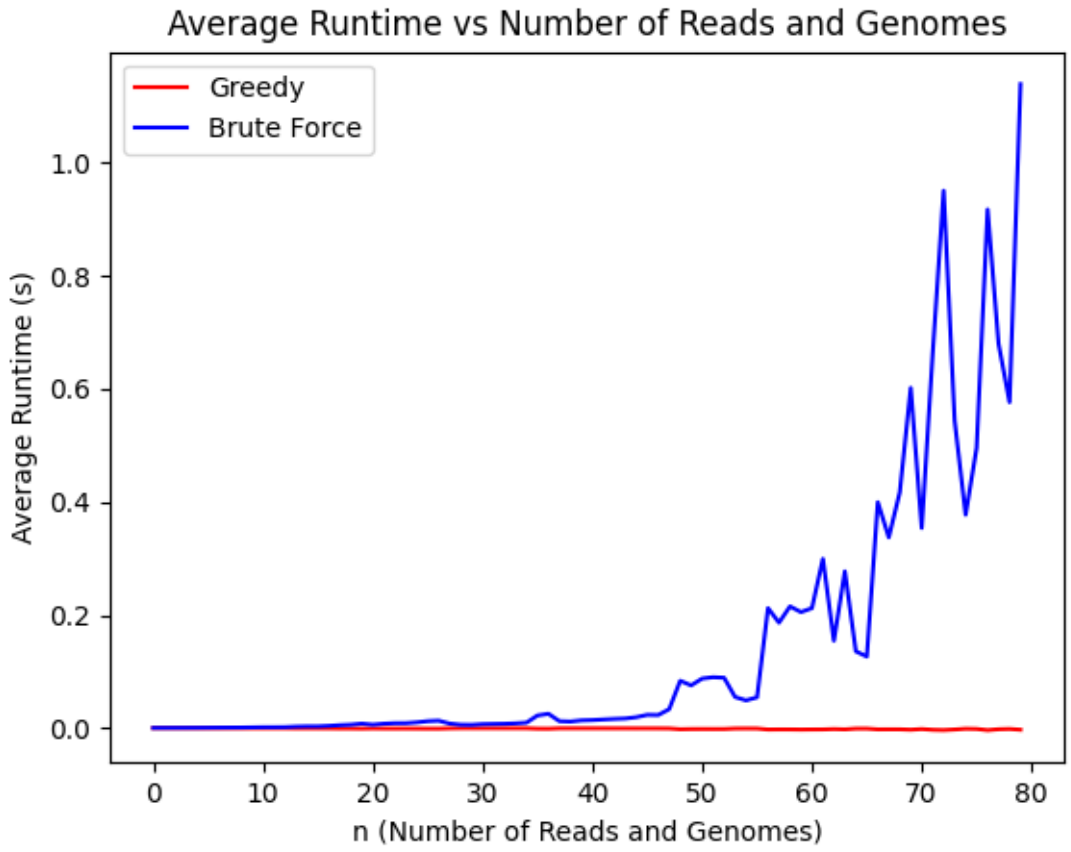
b)

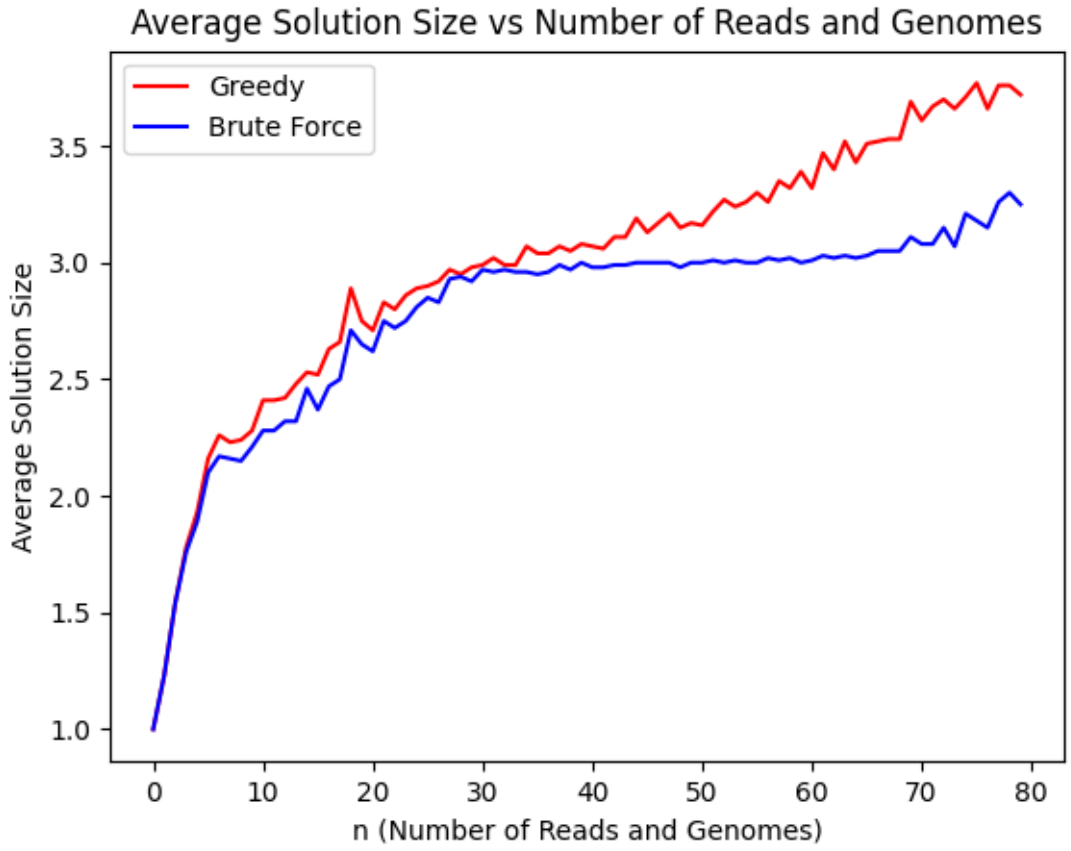
```
/* Initialize covered as an empty list.  Will be used to keep track of which sets are in the
   solution.                                                                              */
covered ← []
remaining ← {w_1, w_2, ...w_n}
sets ← {s_1, ...s_m}
n ← len(remaining)
while len(covered) < n do
    bestSet ← sets[1]
    bestScore ← 0
    for s ∈ sets do
        if score(s, remaining) > bestScore then
            /* score(a, b) returns the number of common elements in the sets a and b.  a and b are
               a both sets of elements.                                                   */
            bestScore ← score(s, remaining)
            bestSet ← s
        end
    end
    covered.append(s)
    set.remove(s)
    for w ∈ s do
        remaining..remove(w)
    end
end
return(covered)
```

c)

```
bestCollection ← S
reads ← {w_1, w_2, ...w_n}
subsets ← {s_1, ..., s_m}      s_i ⊆ reads
for Each possible combination of subsets T = {t_1, ..., t_k}    and T ⊆ subsets do
    if length(T) < length(bestCollection) & Valid(T, reads) then
        /* Valid(T, reads) returns True if each element in reads is in at least one element in T
           and false otherwise.                                                           */
        bestCollection ← T
    end
end
return(bestCollection)
```

d)


Average Runtime vs Number of Reads and Genomes
Greedy
Brute Force
Average Runtime (s)
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60
70
80
n (Number of Reads and Genomes)


Average Solution Size vs Number of Reads and Genomes
Greedy
Brute Force
Average Solution Size
1.0
1.5
2.0
2.5
3.0
3.5
0
10
20
30
40
50
60
70
80
n (Number of Reads and Genomes)

e)

The plots appears to support that the greedy algorithm is in fact a 2-approximation algorithm for the exact solution. This is because in the average solution size of the greedy approach never exceeds the average solution size of the brute-force approach.

Question 5

a)

$$\forall i \qquad \sum_{j=1}^{n} y_{i,j} = 1$$

$$i \in [1, n]$$

$$\forall i, j \qquad y_{i,j} \in \{0, 1\}$$

b)

$$\forall i, j, k \qquad x_{i,j} y_{i,k} y_{j,k} = 0$$

$$i, j, k \in [1, n]$$

c)

$$\forall j \quad z_j \leq \sum_{i=1}^{n} y_{i,j} \leq n z_j$$

$$j \in [1, n]$$

d)

$$\underset{Z}{argmin} \qquad |Z| = \sum_{i} z_i$$

$$\forall i \leq z_i \in \{0, 1\}$$

e)

We can reduce graph coloring to the experiment batching problem. Given a graph $G(V, E)$, let C be some coloring of the vertices V such that no adjacent vertices have the same color (e.g. C is a coloring that satisfies the graph coloring problem). Now, let each vertex in G represent some experiment that takes exactly 1 unit of time and each represent the two experiments sharing at least one piece of equipment. Then the graph coloring solution C is a solution to the experiment batching problem where each batch is represented by all the nodes of one color.

We can check the the solution of the experiment batching problem in polynomial time by iterating over all of the experiments in each batch and making sure that none of the experiments share equipment. This should take $O(|T||E|)$ where $|T|$ is the total number of time increments and $|E|$ is the total number of experiments.

Since we can check a solution in polynomial time and graph coloring can be reduced to the experiment batching problem in polynomial time, the experiment batching problem is an NP-complete problem.

f)

Modify y into a n by $\sum_{i=1}^{n} t_i$ matrix such that if experiment i is running at time j then $y_{i,j} = 1$ otherwise $y_{i,j} = 0$. Also modify the auxiliary vector z into a vector of length $\sum_{i=1}^{n} t_i$ where $z_j = 1$ if some experiment is running at time j otherwise $z_j = 0$.

These modifications to Y and $\overrightarrow{z}$ along with the constraints previously specified model the more realistic problem statement of batching experiment runs to try and minimize the time it takes to run all experiments.