02-512/02-712 Computational Methods for Biological Modeling and Simulation
Lecturers: Russell Schwartz and Oana Carja
Tuesdays and Thursdays 3:05pm-4:25pm HH B131

Problem Set 1

Assigned 9/8/22, Due 9/22/22

Problem 1 (15 points): The goal of this problem is for you to take informal descriptions of biological questions and cast them as well-known optimization problems we have seen in class. For each description, identify a classic computational problem we have studied that would solve the informal biological problem and describe how to convert the informal problem into a formal statement of inputs, outputs, and objective function.

To give an example, here is a similar problem:

> Suppose we are trying to develop a new small molecule drug to prevent a virus from binding to a cellular receptor protein by attaching to some binding pockets on the receptor protein. There are a set of chemical groups on the molecule, each of which has some known binding energy for each binding pocket. We want to know the most stable (minimum energy) way of assigning chemical groups to binding pockets. Assume that each group can be bound to only one pocket and each pocket to only one group.

A reasonable solution to this problem would be:

> We can pose this as a maximum bipartite matching problem. We can create one part of the graph with a node for each chemical group and the other part with a node for each pocket. We will place one edge between any group/pocket pair capable of binding, weighted by minus the energy of binding (minus because this is a maximization problem and we want to minimize energy). The maximum matching will then provide a minimum energy assignment of chemical groups to binding pockets.

The systems to address are the following:

a. Cancer is often characterized by high *intratumor heterogeneity*, meaning that different cells in the tumor will have different combinations of mutations. How much heterogeneity a tumor has can help us predict how aggressive the cancer is likely to be. Suppose we want to use sequencing technology to find out how many different genetically distinct kinds of cells (called *clones*) are present in a tumor. We randomly sequence many fragments of tumor DNA each covering a small fraction of the genome. We then determine which pairs of fragments are "incompatible," meaning they cover the same region of genome but are not identical. Given the information of which fragments are incompatible, we want to put a lower bound on the number of clones present in the tumor.

b. Suppose we want to design a vaccine that is effective against many strains of a virus. We identify a region of the viral genome that differs between strains and sequence that region for many different strains. We want to create an RNA vaccine by synthesizing a strand of RNA that is found in every version of the genome region in all of the strains. How can we find the longest such sequence?

c. Suppose we are developing a new drug and are interested in the risk of toxic side effects. We

deliver the drug to the organ where it is needed (let's say lungs) but we believe it is dangerous to another organ (let's say kidneys) if it is delivered there at a higher rate than the body can remove it. We can represent the body as a set of compartments corresponding to different tissues with some maximum throughput of transport of the drug between any given pair of adjacent tissues. We would like to know the maximum rate of throughput with which the drug might get to the kidneys if we deliver it initially to the lungs.

d. Suppose we have a gene that can occur in many splice forms. We want to know which splice forms are expressed in any given cell type. We determine the exons of the gene and determine which exons are present in each splice form. We then want to design a set of PCR tests, each of which tests for the presence of a single exon. We would like to design as few primer pairs as possible to allow us to identify which splice form is used in any given tissue. We will assume no tissue expresses more than one splice form.

e. Suppose we have a set of genes involved in a genetic regulatory network. We have have predictions of which genes are regulators of one another based on possible transcription factor binding sites, but these predictions are uncertain. We assume that each possible interaction has some probability (confidence) that we can assign to it. We suspect that there is some indirect chain of interactions between some gene X and a target gene Y through other genes A, B, C, etc. We would like to find the most likely such chain of interactions by which X could influence Y assuming we know X and Y but do not know through which intermediate genes they will interact. (Hint: Adding log probabilities is equivalent to multiplying probabilities.)

Problem 2 (20 points): In this problem, we will examine the tradeoffs involved in different ways of posing a common task in biological network analysis: finding modules of genes that show similar activity across different conditions, i.e., they are *co-expressed*. We will assume that we have measured the activities of a set of genes under many conditions and have quantified in some way how similar or dissimilar are the expression profiles of any pair of genes. We then define some cutoff dissimilarity value and declare that two genes are co-expressed if they have a dissimilarity below the cutoff. We would like to use these pairwise similarity values to identify groups of three or more genes that we can consider co-expressed. We will try posing the problem in a few different ways and consider the tradeoffs between them.

a. We first pose this problem by declaring that two genes are part of a co-expressed module if it is possible to get from one to the other by a chain of pairwise co-expressions. That is, if A is co-expressed with B and B is co-expressed with C, then we declare A, B, and C to be part of a co-expression module even if A and C have a dissimilarity higher than the cutoff. Can we efficiently find all co-expression modules for this definition? Describe in a sentence or two an efficient algorithm if one exists or suggest a heuristic strategy for solving the problem approximately if not.

b. We will next propose to solve this problem with an alternative definition: a set of genes are in a co-expression module only if every pair of genes in the module is pairwise co-expressed. That is, genes A, B, and C are a co-expression module only if A is pairwise co-expressed with B, B is pairwise co-expressed with C, and C is pairwise co-expressed with A. Can we efficiently find all co-expression modules for this definition? Describe in a sentence or two an efficient algorithm if one exists or suggest a heuristic strategy for efficiently solving the problem approximately if not.

c. We will consider a third definition, treating it as an optimization problem. We will start with the assumption that all of the genes can be grouped into at most k co-expression modules for some

k. We will then split the genes into k sets so that we maximize the number of dissimilar gene pairs assigned to distinct modules. Can we efficiently find all co-expression modules for this definition? Describe in a sentence or two an efficient algorithm if one exists or suggest a heuristic strategy for solving the problem approximately if not.

d. Suppose we know that we need to examine a very large number of genes but that we still care a lot about getting exactly the right answer. Which of the three models would that most lead us to favor? Justify your answer in a sentence or so.

e. Suppose we know that our data are very noisy so the lists of which gene pairs are co-expressed will include many false positive (saying two genes are pairwise co-expressed when they are not) and false negative (saying two genes are not co-expressed when they are) interactions. Which model would that lead us to favor? Justify your answer in a sentence or so.

f. Suppose our data is particularly prone to false positives (saying genes are pairwise co-expressed when they are not) but not false negatives. Which model would that lead us to favor? Justify your answer in a sentence or so.

Problem 3 (20 points): We heard in class that sequence assembly, at least for large genomes, is often solved using a reduction to the Traveling Salesperson Problem (TSP). One problem with this TSP formulation, at least as presented in class, is that it will force us to produce a single sequence containing every sequence read in a data set even if there is little or no overlap between some subsets of the reads. If our sequence reads really come from disjoint genome regions, such as distinct chromosomes, then we want an assembler to assemble only one chromosome at a time. What we would really like to do is identify sequences of reads such that there is enough overlap between each pair of reads that have confidence they come from a single contiguous sequence. To develop an approach for this problem, let us suppose that we can assign to any overlap between two reads a probability that the overlap could have occurred by random chance. We can then say our confidence in a sequence containing multiple reads is the product of the confidences of the consecutive pairs of reads in the sequence. (Equivalently, we can say the log confidence of a sequence of reads is the sum of the log confidences of its consecutive pairs of reads.) We will then say that our goal here is to find the set of k reads forming the highest confidence assembly, for some user-specified k, rather than to assemble all of the reads.

a. Formally pose this as a graph optimization problem by providing formal statements of an input, output, and objective function and explaining how to map our true input data and desired solution onto those formal inputs and outputs. Assume that for each pair of reads, we have found its optimal overlap and computed its log confidence.

b. It happens, unfortunately, that this problem is NP hard. We will try to establish that formally. First, argue that the problem is checkable, i.e., that if we are given a proposed solution to the optimization problem and a bound on its cost, we can efficiently verify that the proposed solution is a valid solution to the decision version of the problem satisfying the provided cost bound. That will show that the corresponding decision problem is in the class NP. (Hint: the decision variant of a problem like this will have an extra input bound B.)

c. We will next want to show that the problem is hard by reducing a known NP-hard problem to it. For this purpose, we will use the standard TSP. Show that we can efficiently convert any arbitrary instance of the TSP problem (graph G, weight function w) to an instance of our sequence assembly

problem.

d. Suggest a practical approach we could use to solve this modified assembly problem optimally for moderately hard problem instances.

Problem 4 (25 points): This is a programming problem in which we will examine another possible way of resolving the problem of multiple optimal solutions in intraspecies phylogeny inference that we considered in class. In class, we saw that one can efficiently provide a useful answer to the problem of multiple optima by returning the union of all optimal solutions. Here, we will consider an alternative: finding the intersection of all possible solutions. That is, we will suppose that when we are given an instance of the minimum spanning tree problem, we want to return a graph that contains only those edges in which we have high confidence because they are found in every optimal tree. We will assume that as input we are given a set of taxa and weights representing evolutionary distances between the taxa.

a. We will propose to solve this by adapting Kruskal's algorithm to our new problem. We already know how to adapt Kruskal's to find the union of minimum spanning trees. Describe in plain English how you could adapt Kruskal's algorithm to find the intersection instead of the union of minimum spanning trees. (Hint: Try starting from the union algorithm, but discard edges that are not part of the union as you proceed.)

b. The trickiest part of getting this working will be figuring out, as we add sets of edges to the union tree, which ones need to be discarded from union tree to get the intersection tree. It will help in solving this to have a subroutine that takes a graph and a subset of its edges and identifies all edges from the subset that are not in any cycle in the graph. Provide pseudocode for such a subroutine. (Hint: You can use a single-source shortest path algorithm, or depth-first search if you are familiar with it, as a subroutine of this subroutine to make the calculation easier.)

c. Using your answer from part b, design an algorithm for the minimum spanning tree intersection problem and provide pseudocode:

d. Write code implementing your algorithm. Your code should take as input a single file providing a number of taxa followed by a matrix of pairwise distances. You can assume the matrix is symmetric (i.e., element (i,j) is the same as element (j,i)) and that diagonal elements (i.e., elements (i,i)) are arbitrary. If you use Matlab, it is okay to take as input a single argument for the distance matrix. Your code should return as output a list of edges in the intersection of optimal trees.

e. Test your code on the following inputs (provided in Canvas) and provide your outputs:

```
3
0  2  3
2  0  3
3  3  0


4
0  1  4  1
1  0  2  2
4  2  0  2
1  2  2  0
```

***Problem 5 (20 points):** In this problem, we will consider how we might use ILPs to solve another hard problem in sequence assembly. One important reason sequence assembly in practice can be harder than the simple models we considered in class is because real sequence data often has errors. There may be incorrect nucleotides in some reads. We may also have some fragments that are mistakenly included in an assembly because they have homology to a region of genome we are assembling, even though they actually come from a distant region of the genome. Here we will try to develop a preprocessor to "clean" a set of sequence reads to remove such errors before assembly. We will assume that we are assembling RNA for one gene and have a set of reads we think come from that gene, but we believe there may be some reads that we want to remove because they either have misread base errors in them or come from another homologous gene.

One intuition we can use for this problem is that a gene will normally have only two alleles, or sequence variants, in a single individual. If we can identify sequence reads that appear to be homologous but not identical, then we should be able to test whether we would need more than two alleles of the gene to explain all the reads. If so, then we know some of the reads are bad and need to be removed. Let us suppose we have a set of n reads $R = \{r_1, \ldots, r_n\}$ for $r_i \in \{A, C, G, T\}^*$ and for each pair of reads (r_i, r_j) we have a binary incompatibility indicator y_{ij} where $y_{ij} = 1$ if the reads cannot have come from the same allele and $y_{ij} = 0$ otherwise. Our goal will be to throw out some of the reads so that the ones we are left with could have come from just two alleles.

a. We can pose this as a computational problem by saying that we want as many reads as possible so that the reads left can be grouped into two sets where there are no incompatibilities within either set. In graph terminology, then, we want to find a subset of reads that define a bipartite subgraph. Provide a formal statement of the problem of throwing out as few reads as possible so that the remaining reads define a bipartite graph.

b. The problem we are defining here is NP-hard. Provide a brief proof of this statement. (Hint: You do not need to do a full reduction proof.)

c. We will want to pose this as a linear program. To do this, first, define the variables we will need to specify which fragments we keep in the graph.

d. The trickiest part of the problem is coming up with a set of linear constraints that specify that the retained reads form a bipartite graph. We want to find a way of asserting that there are no odd-length cycles in the graph, but this is difficult since the number of such possible cycles is exponentially large. For now, let us assume that we want to assert that a particular odd-length cycle, which we will arbitrarily call $r_1, r_2, \ldots, r_k, r_1$ for some even k, is not present in the graph. Provide a linear constraint asserting this.

e. Now define the objective function for the problem in terms of the ILP variables.

f. To actually solve the problem, we would make use of an advanced technique in solving ILPs we did not see in class: creating our constraints in stages rather than all at once. More specifically, we can sometimes solve ILPs with very large constraint sets by solving the problem for a subset of the constraints, seeing which are violated by the solutions, and then adding in the violated constraints and trying again. For this problem, our algorithm would work as follows:

1. Establish an initial version of the ILP with an initially empty constraint set $S = \emptyset$.
2. Solve the ILP, so that it will find a subgraph G' of the input G (which might not be bipartite)

3. Search for any odd-length cycle C in G', which we can do efficiently with depth first search on G'. If there is no such cycle then return G'
4. Add a constraint c to S prohibiting the specific cycle C and return to step 2.

Prove that this procedure will in fact solve the full problem, i.e., that it will terminate with an optimal, valid solution to the problem. (Hint: Try a proof by contradiction.)

*Recall that the starred problems are only for the 02-712/03-712 students.