Generating Decisive Sub-matrices for Phylogenetic Tree Construction

Background

A large problem when building phylogenetic trees is data incompleteness. A significant amount of missing data can lead to ambiguity in the structure of trees, leaving biologists unsure of the accuracy of the tree. However, even with incomplete data, an unambiguous phylogenetic tree can still be built depending on the makeup of data. Data can be represented using a data availability matrix which indicates whether sequencing data is available for a given taxon at a given loci. Each row represents an organism, and each column represents a loci.

Sanderson and Steele found that these matrices can be used to check if an unambiguous tree can be built. They showed that if a matrix is decisive, then you can confidently build a phylogenetic tree from that data. However, if a matrix is indecisive, we can find a decisive submatrix, which would correspond to a subset of organisms for which we can build a phylogenetic tree. Checking decisiveness of a matrix is equivalent to solving the no rainbow 4-coloring problem on the matrix’s hypergraph. We can formulate this NP-complete problem as both an integer linear programming instance and satisfiability instance for efficient solving in practice. Once we find a decisive submatrix, a biologist can confidently construct a phylogenetic tree for those taxa.

SAT Formulation

To check for check for decisiveness, we formulate the no rainbow 4-coloring problem as a SAT problem. We use the following constraints and their corresponding Boolean formulas to check for a coloring. Let $x_{ia}$ be true if taxon $i$ has color $a$, $r_i$ be the number of loci and $c_j$ be the set of all taxa who have data for loci $j$.

1. Each taxon has only 1 color: \[ \forall i \in [\text{taxa}] \forall a \in [\text{colors}] \exists r_i \forall j \in [\text{loci}] \big( x_{ia} \lor x_{ja} \lor x_{ja} \lor x_{ia} \big) \]
2. Each color appears at least once in the coloring: \[ \forall a \in [\text{colors}] \exists j \in [\text{loci}] \big( x_{ja} \lor x_{ja} \lor x_{ja} \lor x_{ja} \big) \]
3. No loci is rainbow colored. Let $k$ be the number of loci and $X_i$ be the set of all taxon who have data for loci $i$: \[ \forall i \in [\text{taxa}] \forall X_i \in [\text{data avail}] \exists k \forall j \in [\text{loci}] \big( x_{ja} \land x_{ja} \land x_{ja} \land x_{ja} \big) \]

We also use an existing ILP formulation of the no rainbow 4-coloring.

Software Pipeline

We built a software pipeline that generates a decisive submatrix from an indecisive data availability matrix. The pipeline works as follows:

1. while the matrix is not decisive:
   1. add/remove rows using heuristic
   2. Generate an ILP or CNF Boolean formula
   3. Solve the formulation

We solve the ILP instances using the Gurobi ILP solver, a paid software available under academic license. We tested several ILP solvers from the PySAT library.

Experiment

We ran the pipeline on real data provided by Dobrin, Zwickl, and Sanderson. The pipeline was always run on the same computer with four cores and a Linux operating system with no background programs running. From these experiments, we wanted to know:

1. Are the theoretical results feasible for use in practice?
2. What problem formulation should be used?
3. What heuristic yields the best sub-matrices?

The heuristics were tested basic:

Remove method: Repeatedly remove a taxon that has the fewest number of non-zero entries until a decisive submatrix is found.

Add method: Remove taxa as described in the remove method and add back sets of taxa in the powerset of initially removed taxa until the largest decisive sub-matrix is found.

We ran four types of experiments: add method with ILP, add method with SAT, remove method with ILP, and remove method with SAT.

Results

Timing Results

We tested several SAT solvers and found that Glucose 4 was generally the fastest, so we use those results to compare to the ILP results. We see that in most cases, ILP and SAT formulations run in comparable time. We also see that on smaller matrices, the add and remove methods run in similar time, but more programs time out when using the add method.

Size Results

The add method consistently produces larger decisive submatrices when it finished. Several of the submatrices produced by the remove method for organisms like fungi and chameleons were relatively large submatrices.

Other Results

When analyzing the resulting submatrices, we saw that all the matrices produced using the add method were trivial, meaning they all shared one gene in common. Many of the matrices produced by the add results were also trivial, but several were non-trivial. Some of the non-trivial matrices had good coverage over several families of organisms.

Acknowledgements

I would like to thank my mentor, David Fernández-Baca, for his guidance and mentorship throughout this entire project. I would also like to thank my graduate student, Ghazaleh Parvini, and Dan Gusfield for their helpful collaboration.

References


