Lecture 5 : Approximating Maximum Parsimony

MATH285K - Spring 2010 *Lecturer: Sebastien Roch*

References: [SS03, Chapters 2, 5], [DPV06, Chapters 5, 9]

Previous class

Recall:

THM 5.1 (Complexity of Maximum Parsimony) *The search problem corresponding to Maximum Parsimony is* **NP***-complete.*

1 Coping with NP-completeness

Local search heuristics. We restrict our attention to the binary phylogenetic tree case with $X = [n]$. Fix a metric δ on a character state space C. Recall that, on a fixed tree $\mathcal{T} \in B(n)$, the parsimony score $\ell_\delta(\mathcal{C}, \mathcal{T})$ of a collection $\mathcal{C} = \{\chi_1, \ldots, \chi_k\}$ of characters on X with $|X| = n$ can be computed efficiently by dynamic programming. Therefore, a natural heuristic for minimizing the parsimony score of $\mathcal{C}$ is to perform a local search on the space of trees. Several definitions of local moves on the space of trees have been considered. A typical example is as follows:

DEF 5.2 (Nearest-Neighbour Interchange) *Let $\mathcal{T} = (T, \phi) \in B(n)$ with $T = (V, E)$. A nearest-neighbour interchange (NNI) operation is obtained by choosing an interior edge $e = \{u, v\} \in E$ and two vertices $u_0 \neq v$, $v_0 \neq u$ adjacent respectively to u, v and interchanging the two subtrees rooted at u_0, v_0.*

THM 5.3 (Tree Space is Connected under NNI) *Let $\mathcal{T} \neq \mathcal{T}' \in B(n)$. Then $\mathcal{T}$ can be transformed into $\mathcal{T}'$ by a sequence of NNI operations.*

Proof: Because NNI operations are reversible, it suffices to take $\mathcal{T}'$ to be a caterpillar tree on n leaves. The theorem then follows easily from the following lemma.

LEM 5.4 *Let $\mathcal{T}_1 \neq \mathcal{T}_2 \in B(n)$ be such that both can be obtained by adding a leaf edge with label n to $\mathcal{T}_0 \in B(n-1)$. Then $\mathcal{T}_1$ can be transformed into $\mathcal{T}_2$ by a sequence of NNI operations.*

Proof: Let $e_1 = \{u_1, v_1\}$ (respectively $e_2 = \{u_2, v_2\}$) be the edge of $\mathcal{T}_0$ on which leaf edge n is added to obtain $\mathcal{T}_1$ (respectively $\mathcal{T}_2$). Let $v_1, \ldots, v_m$ be the path connecting e_1 and e_2 (without crossing either edge). Let v_0 be the neighbour of leaf n in $\mathcal{T}_1$. Then it is easy to see that $\mathcal{T}_2$ is obtained from $\mathcal{T}_1$ by performing successively NNI operations around edges $\{v_0, v_1\}, \ldots, \{v_0, v_k\}$. ■

■

Intelligent exhaustive search. See [DPV06] for a discussion of branch-and-bound algorithms.

Approximation algorithms. Little can be proved about the performance of the two approaches discussed above—unless one runs them for an exponential amount of time. In the next sections, we derive an efficient algorithm which is guaranteed to output a solution whose objective is within a given factor of the optimal solution. Such an algorithm is called an *approximation algorithm*.

2 Minimum Spanning Trees

We will need the following definition:

DEF 5.5 (Minimum Spanning Tree (MST)) *Let $G = (V, E)$ be a connected graph with edge weights $\{w_e\}_{e \in E}$. A* spanning tree *for G is a tree $T = (V', E')$ such that $V' = V$ and $E' \subseteq E$. A* minimum spanning tree (MST) *is a spanning tree $T = (V', E')$ minimizing*

$$w(T) = \sum_{e \in E'} w_e.$$

It turns out that MSTs are easy to compute using a so-called greedy approach. Kruskal's algorithm works as follows:

- Start with the forest $F^* = (V, E^*)$ on V with empty edge set $E^* = \emptyset$.
- Repeat until $F^* = (V, E^*)$ is connected:
  - Add to F^* the lightest edge e in $E \backslash E^*$ such that e does not create a cycle in F^*.

Kruskal's algorithm is clearly efficient. The correctness of Kruskal's algorithm follows immediately from the following lemma:

LEM 5.6 (Cut Property) *Suppose the edges in $Z \subseteq E$ are part of some MST of $G = (V, E)$. Let $S \subseteq V$ be such that no edge in Z connects S and $V \backslash S$. Let e be the lightest edge connecting S and $V \backslash S$. Then $Z \cup \{e\}$ is part of some MST of G (not necessarily the same as above).*

Proof: By assumption, $Z \subseteq E_1$ for some MST $T_1 = (V, E_1)$. If e in the statement is also in E_1, there is nothing to prove. Assume $e \notin E_1$. Since adding e to T_1 creates a cycle and e connects S and $V \backslash S$, there must be another edge e' in that same cycle that also connects S and $V \backslash S$. Replace e' with e in T_1 to obtain T_2. Since T_2 is connected and has the same number of edges as T_1, it must be a tree. Also, since $w_e \leq w_{e'}$ by construction, we have $w(T_2) \leq w(T_1)$ and we are done. ■

Going back to the analysis of Kruskal's algorithm, suppose by induction that the forest F^* so far constructed is part of an MST. The next edge chosen e connects two subtrees of F^* (otherwise it would create a cycle), say $T_1^* = (V_1^*, E_1^*)$ and T_2^*. Clearly, e is a lightest edge connecting V_1^* and $V \backslash V_1^*$, and the induction goes through by Lemma 5.6.

3 Approximation

Our main result is the following:

THM 5.7 (Approximating Maximum Parsimony) *Let $\mathcal{C} = \{\chi_1, \ldots, \chi_k\}$ be a collection of characters on X with state space C. Let δ be a metric on C. Let $G = (X, E)$ be the complete graph on X with edge weights*

$$w_{\{u,v\}} = \sum_{i=1}^{k} \delta(\chi_i(u), \chi_i(v)).$$

If $w(\mathcal{C})$ is the weight of an MST on G then

$$\frac{1}{2} w(\mathcal{C}) \leq \ell_\delta(\mathcal{C}) \leq w(\mathcal{C}).$$

Proof: Let T' be an MST for G. Using the identity labeling map ϕ', $\mathcal{T} = (T', \phi')$ is an X-tree achieving $\ell_\delta(\mathcal{C}, \mathcal{T}) = w(\mathcal{C})$ and therefore $\ell_\delta(\mathcal{C}) \leq w(\mathcal{C})$.

To prove the other direction, let $\mathcal{T} = (T, \phi)$ be a Maximum Parsimony tree and assume w.l.o.g. that $\mathcal{T}$ is a phylogenetic tree. Let $\vec{T}$ be the directed graph obtained by replacing each edge of T with two edges in opposite directions. A *walk* in a directed graph is a sequence of edges $(\vec{e}_1 = (u_1, u_2), \ldots, \vec{e}_m = (u_{m-1}, u_m))$. A walk is *closed* if $u_1 = u_m$. An *Eulerian tour* is a closed walk such that each edge of the graph appears exactly once in the sequence.

LEM 5.8 *$\vec{T}$ has an Eulerian tour.*

Proof: It is well-known that a directed graph whose underlying undirected graph is connected and such that every vertex has an equal number of ingoing edges (in-degree) and outgoing edges (out-degree)—such as $\vec{T}$—has an Eulerian tour. Indeed, consider the longest walk $W = (\vec{e}_1 = (u_1, u_2), \ldots, \vec{e}_m = (u_{m-1}, u_m))$ such that each edge appears at most once in W. By assumption W cannot be extended, hence all outgoing edges of u_m have been used. Since the in-degree and out-degree of u_m are equal, it must be that $u_1 = u_m$ so that W is closed. By connectedness, if W is not Eulerian, there is an edge e not in W but incident to a vertex visited by W, say u_1. Then adding e to W (to the beginning or the end depending on its orientation) produces a longer path, a contradiction. ■

We return to the proof of Theorem 5.7. Let $W = (\vec{e}_1 = (u_1, u_2), \ldots, \vec{e}_m = (u_{m-1}, u_1))$ be an Eulerian tour of $\vec{T}$. Let $x_1, \ldots, x_n$ be the elements of X in the order that their corresponding leaves are first visited by W. (Each leaf is visited exactly once because it has in-degree and out-degree 1.) The path $(x_1, x_2, \ldots, x_n)$ is a spanning tree of G with weight

$$w^* = \sum_{j=1}^{n-1} \sum_{i=1}^{k} \delta(\chi_i(x_j), \chi_i(x_{j+1})).$$

By the triangle inequality, $\delta(\chi_i(x_j), \chi_i(x_{j+1}))$ is less than the sum of the weights on the subwalk of W between the leaves corresponding to x_j and x_{j+1}. Since each undirected edge of T is visited exactly twice by W, we get finally

$$w(\mathcal{C}) \leq w^* \leq 2 \sum_{e \in E} w_e = 2\ell_\delta(\mathcal{C}, \mathcal{T}) = 2\ell_\delta(\mathcal{C}),$$

because $\mathcal{T}$ is a Maximum Parsimony tree.

■

Further reading

The definitions and results discussed here were taken from Chapter 5 of [SS03] and Chapters 5 and 9 of [DPV06].

References

[DPV06] S. Dasgupta, C. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw-Hill, 2006.

[SS03] Charles Semple and Mike Steel. *Phylogenetics*, volume 24 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2003.