Constrained Optimization for Tree Construction

Tandy Warnow

September 20, 2022

Constrained Optimization for Tree Construction

Tandy Warnow

Constrained Optimization

Basic idea: NP-hard optimization problem for tree estimation, but solve it exactly by constraining the allowed "bipartitions"

- ASTRAL: optimizing maximum quartet support
- FASTRAL: modifying the constraint set for ASTRAL
- SVDquest: optimizing maximum quartet support
- FastRFS: supertree, optimizing Robinson-Foulds distance

Results for SVDquest

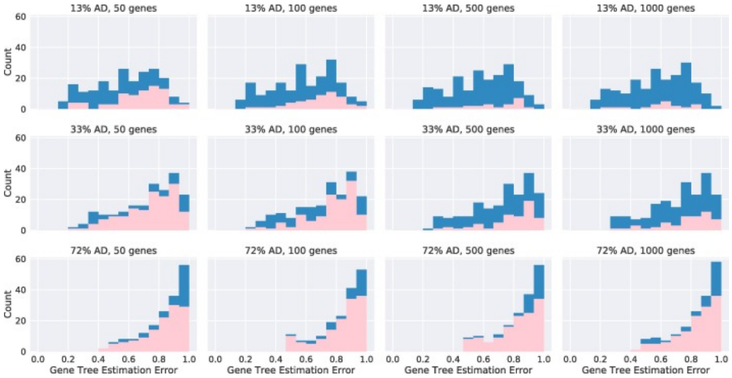
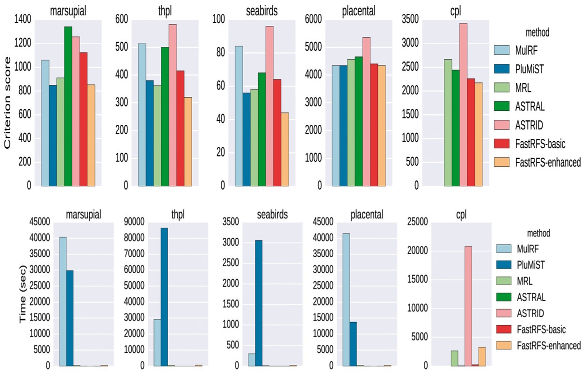


Figure : Pink: SVDquest produces better quartet support scores than SVDquartets+PAUP*

Results for FastRFS



Bioinformatics, Volume 33, Issue 5, 1 March 2017, Pages 631–639, https://doi.org/10.1093/bioinformatics/btw600

Figure : FastRFS is faster and more accurate than MulRF

Today's material:

- Reminder about dynamic programming
- Intro to supertree methods (Chapter 7)
- Robinson-Foulds Supertree (RFS) problem
- Constrained Optimization for RFS

Dynamic programming

How would you compute the 100^{th} Fibonacci number?

Recursively?
Or some other way?

Dynamic programming

Calculating F(n) recursively?

How long will it take?

Dynamic programming

Calculating F(n) using dynamic programming?

How long will it take?

Dynamic programming

Dynamic programming in computational biology: all over the place.

- Maximum parsimony on a fixed tree
- Felsenstein's peeling algorithm (aka pruning algorithm)
- Needlemen-Wunsch pairwise alignment (global)
- Smith-Waterman pairwise alignment (local)

Supertree estimation

- Input: set $\mathcal{T}$ of trees on subsets of species set S
- Output: binary tree T on set S, optimizing some criterion

Notes:

1. check whether the input trees are rooted or unrooted (different problems).
2. the input $\mathcal{T}$ is called a *profile*

Uses of supertree estimation

1. Traditional: Combining trees computed by different researchers, on different groups of species
2. Also: Divide-and-conquer strategies

Tree Compatibility

A set $\mathcal{T}$ of trees is said to be **compatible** if there is a supertree T that is compatible with each tree in $\mathcal{T}$.
If so, then T is called a *compatibility supertree* for $\mathcal{T}$.

Tree Compatibility problem: are the trees in $\mathcal{T}$ compatible?

Question to class: what algorithms do you know for solving tree compatibility?

For each algorithm, what assumptions are made about the input?

Tree Compatibility

A set $\mathcal{T}$ of trees is said to be **compatible** if there is a supertree that induces each tree in $\mathcal{T}$.

Tree Compatibility problem: are the trees in $\mathcal{T}$ compatible?

- If all the trees are rooted, then the problem can be solved using Aho, Sagiv, Szymanski, and Ullman (Section 3.3 from textbook) in polynomial time.
- If the trees are unrooted, the problem is NP-hard (Section 3.5 from textbook).

Because unrooted tree compatibility is NP-hard, it is trivial to show that most optimization problems for supertree construction from unrooted source trees are NP-hard.

But they are still hard for supertree construction from rooted source trees!

Matrix Representation with Parsimony (MRP)

MRP: Represent every tree in $\mathcal{T}$ with a $0,1$-matrix (one column for every edge), concatenate the matrices, and then solve maximum parsimony

This is the most well known supertree approach among biologists.

Question: suppose the input trees are compatible. What does MRP return?

Other supertree optimization problems

- Robinson-Foulds Supertree: Find binary T minimizing the Robinson-Foulds distance to the trees in $\mathcal{T}$
- Maximum Quartet Support Supertree: Find binary T minimizing the quartet distance to the trees in $\mathcal{T}$
- Matrix Representation with Likelihood (MRL): Same matrix, but then solve CFN maximum likelihood
- Distance-based supertrees: Compute a matrix M of average leaf-to-leaf distances between species, and find an additive matrix close to M (minimizing some criterion)

All these problems are NP-hard, and some of these optimization problems create very large inputs.

Robinson-Foulds Supertrees

Finding the supertree T that minimizes the Robinson-Foulds (RF) distance to the source trees is the Robinson-Foulds Supertree problem.

MulRF (Chaudhary et al., 2014) and PluMiST (Kupczok, 2011) are two methods for Robinson-Foulds Supertrees.

Robinson-Foulds Supertrees are (sort of) approximations to the Maximum Likelihood Supertree (Steel and Rodrigo, 2008) problem (see Bryant and Steel 2009 for more).

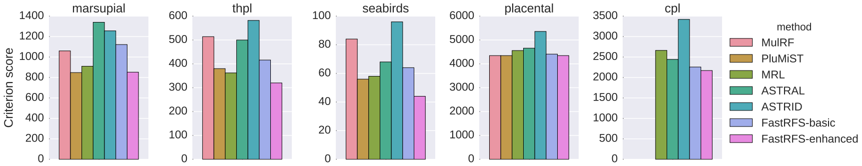
Constrained optimization

Bipartition-constrained Robinson-Foulds Supertree Problem:

- Input: Set $\mathcal{T}$ of source trees and set X of bipartitions on S
- Output: Tree T on S that draws its bipartition set $C(T)$ from X, and that minimizes the RF distance to $\mathcal{T}$ among all such supertrees

FastRFS (Vachaspati and Warnow, Bioinformatics 2016) solves this problem in polynomial time, using dynamic programming.

FastRFS criterion scores



The CPL dataset has 2228 species, and so represents the largest and most difficult dataset; MulRF and PluMiST could not complete on it.

Method	500	500	500	500
Scaffold %	20	50	75	100
# Replicates	8	10	10	10
ASTRAL	15.3	14.8	12.7	11.2
ASTRAL-enhanced	14.8	14.1	12.6	11.2
ASTRID	26.0	50.1	45.4	**10.5**
MRL	15.4	14.3	12.1	11.2
MulRF	46.9	40.3	27.4	12.6
PluMiST	35.4	29.5	22.4	10.9
FastRFS-basic	14.5	14.3	12.4	11.1
FastRFS-enhanced	**14.3**	**13.9**	**12.0**	10.8

Table : Average supertree topology estimation error on simulated datasets.

Using Supertree Methods in practice

- FastRFS is very good at solving its optimization problem (better than the heuristics for the same problem)
- FastRFS produces accurate tree topologies
- FastRFS is pretty fast!
- The constraint set matters (i.e., FastRFS-enhanced is better than FastRFS-basic)

What does FastRFS solve?

Recall: Bipartition-constrained Robinson-Foulds Supertree Problem:

- Input: Set $\mathcal{T}$ of source trees and set X of bipartitions on S
- Output: Tree T on S that draws its bipartition set $C(T)$ from X, and that minimizes the RF distance to $\mathcal{T}$ among all such supertrees

FastRFS solves this problem in polynomial time, but the constraint set X can vary.

- FastRFS-basic: X is the bipartitions from the input trees, plus those computed by ASTRAL
- FastRFS-enhanced: we add bipartitions from the MRL supertree and ASTRID supertree to X

Question to class: What happens to the RFS criterion score as you use FastRFS-basic and FastRFS-enhanced?

How does FastRFS solve its problem?

Constrained optimization problem:

- Input: Profile $\mathcal{T}$ of source trees, and constraint set X of allowed bipartitions of set S
- Output: Tree T on leafset S that minimizes the RF distance to $\mathcal{T}$, subject to $C(T) \subseteq X$

To class:

- What if X is all bipartitions on S?
- What if $X = C(T)$ for some tree T?
- What if $X = \emptyset$?

How does FastRFS solve its problem?

Constrained optimization problem:

- Input: Profile $\mathcal{T}$ of source trees, and constraint set X of allowed bipartitions of set S
- Output: Tree T on leafset S that minimizes the RF distance to $\mathcal{T}$, subject to $C(T) \subseteq X$

Let's solve a *rooted* version of the problem:

- Input: Profile $\mathcal{T}$ of rooted source trees, and constraint set X of allowed clades of set S
- Output: Rooted tree T on leafset S that minimizes the total clade distance to $\mathcal{T}$, subject to $Clades(T) \subseteq X$

Switch to maximize

Now let's change from minimization to maximization:

- Input: Profile $\mathcal{T}$ of rooted binary source trees, and constraint set X of allowed clades of set S
- Output: Rooted binary tree T on leafset S that maximizes the number of shared clades with $\mathcal{T}$, subject to $Clades(T) \subseteq X$

How does FastRFS solve its problem?

Read the paper!
Here for the sake of time, I'll pose a simple problem that is very similar to what FastRFS is solving!

How does FastRFS solve its problem?

Maximum Weight Clade Binary Tree Construction

- Input: Constraint set X of allowed clades, each with a non-negative weight, $w(C)$ for $C \in X$
- Output: Rooted binary tree T on leafset S that maximizes the total weight of its clades, subject to $Clades(T) \subseteq X$

That is, if T is a tree, we define the $Weight(T)$ to be the sum of the weights of its clades.

How does FastRFS solve its problem?

Maximum Weight Clade Binary Tree Construction

- Input: Constraint set X of allowed clades, each with a non-negative weight, $w(C)$ for $C \in X$
- Output: Rooted binary tree T on leafset S that maximizes the total weight of its clades, subject to $Clades(T) \subseteq X$

Let $S = \bigcup_{C \in X} C$, and note that if $S \notin X$, then there is no feasible solution at all.

Question to class: give an example of X for which S is an element of X but there is still no feasible solution.

Solving Maximum Weight Clade Binary Tree Construction

Let C be a clade in X and let $MAX(C)$ denote the maximum weight of any rooted binary tree on leafset C.

- Can we calculate $MAX(C)$ for $|C| = 1$?
- Can we calculate $MAX(C)$ for $|C| = 2$?
- Can we calculate $MAX(C)$ for larger values of $|C|$?
- What do you think $MAX(S)$ would denote?

Solving Maximum Weight Clade Binary Tree Construction

How can we calculate $MAX(C)$ if we have already solved $MAX(C')$ for all C' with $|C'| < |C|$?

Solving Maximum Weight Clade Binary Tree Construction

Suppose we have already solved $MAX(C')$ for all C' with $|C'| < |C|$.

Solving Maximum Weight Clade Binary Tree Construction

Suppose we have already solved $MAX(C')$ for all C' with $|C'| < |C|$.

Then $MAX(C) =$

$$\max\{MAX(C') + MAX(C \setminus C') + w(C) : C' \subset C, C' \in X, C \setminus C' \in X\}$$

Why? (Need to draw a picture)

Solving Maximum Weight Clade Binary Tree Construction

Recall $MAX(C) =$

$$\max\{MAX(C') + MAX(C \setminus C') + w(C) : C' \subset C, C' \in X, C \setminus C' \in X\}$$

Compute $MAX(C)$ for all $C \in X$, from the "bottom-up" (i.e., smallest to largest subsets)

Return $MAX(S)$ to find the weight of the best tree.

How do we find the actual best tree?

Back to other problems

FastRFS is basically doing this, but it defines the weights for each clade using the input source trees.

Any clade that is not in the set X has infinite weight (effectively is not allowed).

ASTRAL solves something very similar, but is using quartets instead of bipartitions or clades.

FASTRAL modifies the constraint set given to ASTRAL, speeding it up and even improving accuracy.

You should read FastRFS, ASTRAL, and FASTRAL papers!

Open questions

What other optimization problems can be solved using constrained optimization?
Can MRP be solved using constrained optimization?