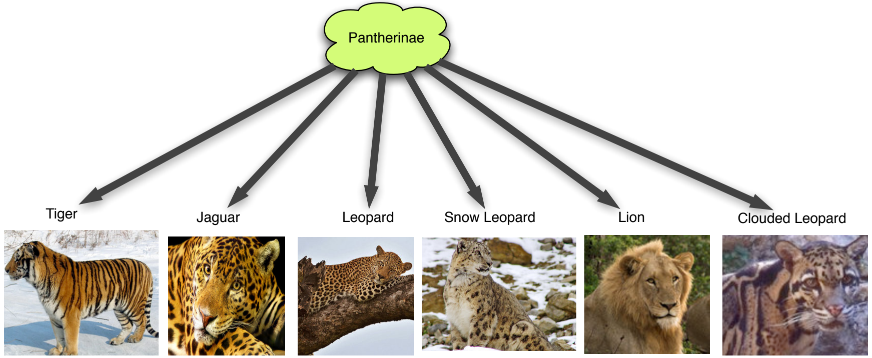
Computational Approaches for Constructing Consensus Trees

Tiffani L. Williams

Department of Computer Science
Texas A&M University
`http://faculty.cs.tamu.edu/tlw`

How did the pantherine lineage evolve?



Wei et al. 2009

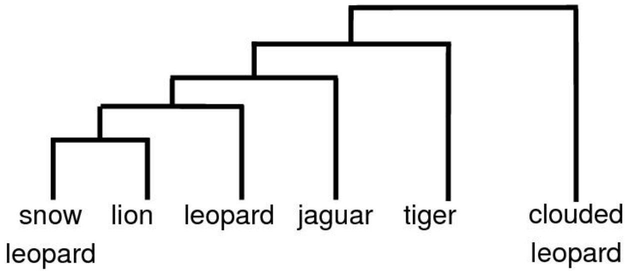


Figure: Based on 7 mtDNA genes (3,816 bp).

Current Best Estimate: Davis et al. 2010

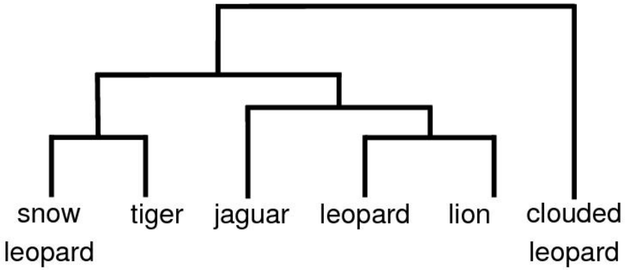
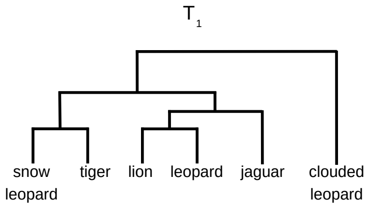
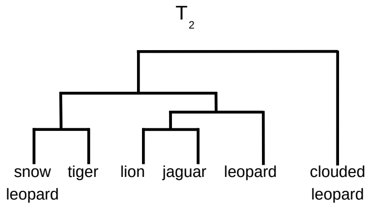
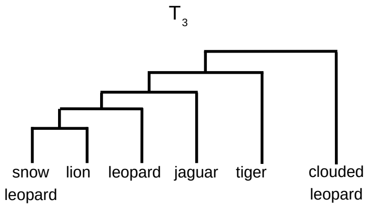


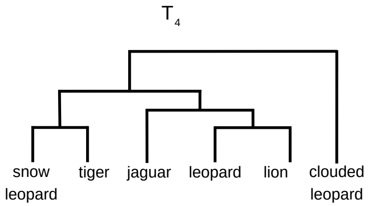
Figure: Based on intronic sequences contained within single-copy genes on the felid Y chromosome which was combined with previously published data and newly generated sequences for four mitochrondial and four autosomal genes. 47.6 kb combined dataset.

We will consider the following trees in this talk.





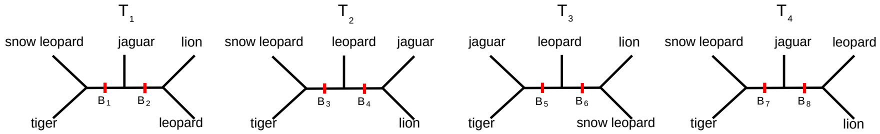


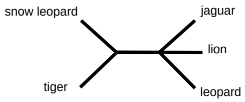


Why the incongruence of pantherine relationships?

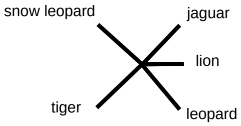
- No one phylogenetic study was performed in exactly the same manner! There have been 14 different studies of the evolution of the pantherine lineage.
- Primary causes of incongruence include:
  1. Rapid evolution and recent divergence of the extant Panthera species.
  2. Different evolutionary rates among various genes.
  3. Different methodologies among the various studies.

What do the different hypotheses have in common?





(a) Majority tree

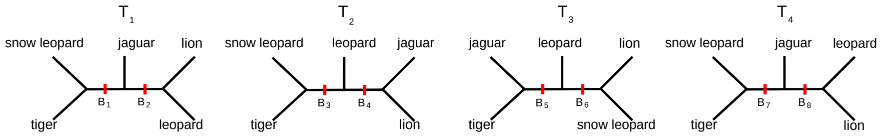


(b) Strict tree

Why Do We Need Computation?

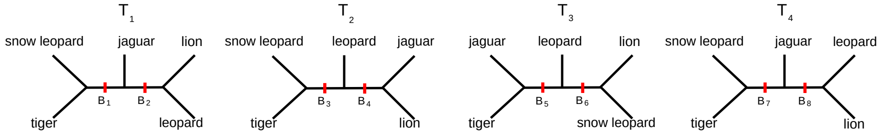
- For the collection of trees for the pantherine lineage, we can compute the consensus tree by hand.
- But, what happens when there are tens to hundreds of thousands of trees of interest?
  - 33,306 trees on 567 taxa of flowering plants (U. of Florida)
  - 90,000 trees on 264 taxa of fish (Texas A&M)
  - 150,000 trees on 525 taxa of insects (Texas A&M)
- We need a computational approach for analyzing these large tree collections—especially as the size of phylogenetic studies continue to increase.

The Anatomy of a Phylogenetic Tree



Tree	BID	Bipartition
T_1	B_1	{snow leopard, tiger | jaguar, lion, leopard}
	B_2	{snow leopard, tiger, jaguar | lion, leopard}
T_2	B_3	{snow leopard, tiger | leopard, jaguar, lion}
	B_4	{snow leopard, tiger, leopard | jaguar, lion}
T_3	B_5	{snow leopard, lion | leopard, jaguar, tiger}
	B_6	{snow leopard, lion, leopard | jaguar, tiger}
T_4	B_7	{snow leopard, tiger | jaguar, leopard, lion}
	B_8	{snow leopard, tiger, jaguar | lion, leopard}

Representing Bipartitions as Bitstrings



BID	snow leopard	tiger	jaguar	lion	leopard	bitstring
B_1	1	1	0	0	0	11000
B_2	1	1	1	0	0	11100
B_3	1	1	0	0	0	11000
B_4	1	1	0	0	1	11001
B_5	1	0	0	1	0	10010
B_6	1	0	0	1	1	10011
B_7	1	1	0	0	0	11000
B_8	1	1	1	0	0	11100

Constructing Consensus Trees

1. Collecting bipartitions from a set of trees
2. Selecting consensus bipartitions
3. Constructing the consensus tree

Step 1: Collecting Bipartitions

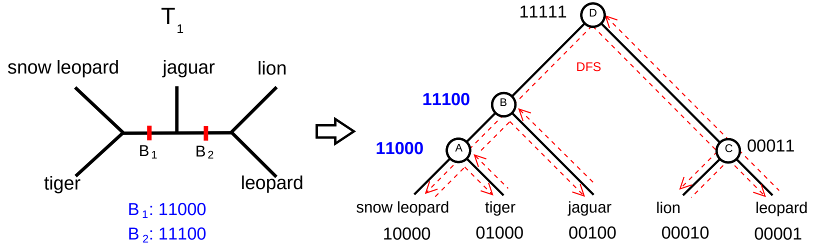


Figure: Using depth-first traversal to collect bipartitions from tree T_1.

Step 2: Selecting Consensus Bipartitions

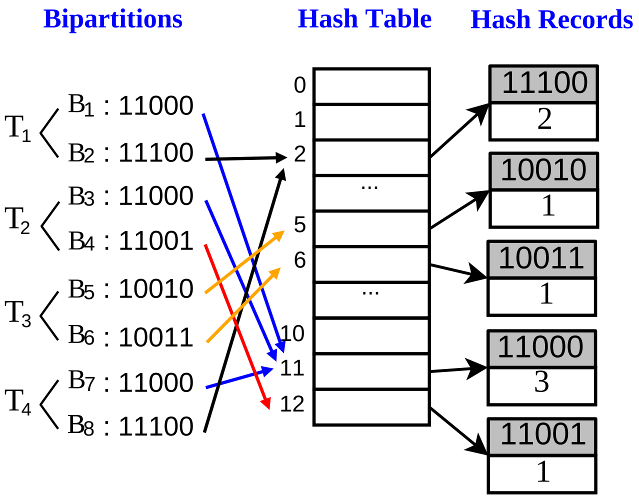
unsorted		sorted		sorted and filtered	
bitstring	*value*	*bitstring*	*value*	*bitstring*	*frequency*
B_1: 11000	24	B_5: 10010	18	10010	1
B_2: 11100	28	B_6: 10011	19	10011	1
B_3: 11000	24	B_1: 11000	24	11000	3
B_4: 11001	25	B_3: 11000	24	11001	1
B_5: 10010	18	B_7: 11000	24	11100	2
B_6: 10011	19	B_4: 11001	25		
B_7: 11000	24	B_2: 11100	28		
B_8: 11100	28	B_8: 11100	28		

- Majority bipartitions: 11000 or {`snow leopard, tiger` | `jaguar, lion, leopard`}
- Strict bipartitions: None

Step 2: Selecting Consensus Bipartitions

- Our current algorithm for this step requires several passes.
- Sorting the bipartitions, while convenient, is expensive.
- How can we design an approach that doesn't require multiple passes or sorting?

Step 2: Selecting Consensus Bipartitions (Hashing)



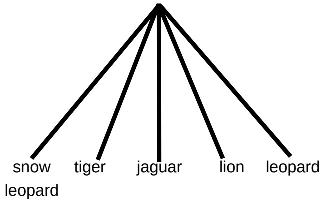
Step 2: Selecting Consensus Bipartitions (Hashing)

- Our hashing function: $h(x) \mod m$, where
  - x is the decimal value of a bitstring, and
  - m is the size of the hash table
- Here are a few examples.
  - $h(11001) \mod 13 = h(25) \mod 13 = 12$
  - $h(10011) \mod 13 = h(19) \mod 13 = 6$
- Caveat: Two different bitstrings could reside in the same location in the hash table.
  - For example, $h(10011) \mod 13 = h(00110) \mod 13 = 6$
  - Such a condition is called a *collision*.
  - Collisions slow down the algorithm and could lead to erroneous results.

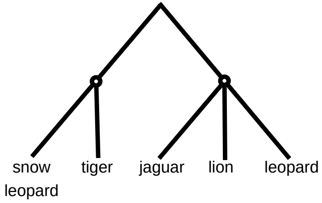
Universal Hashing: Reducing the Probability of Collisions

- Consider the bitstring $b_4 b_3 b_2 b_1 b_0$.
- Standard hashing: $b_4 \cdot 2^4 + b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$
- Universal hashing: $b_4 \cdot r^4 + b_3 \cdot r^3 + b_2 \cdot r^2 + b_1 \cdot r^1 + b_0 \cdot r^0$, where r_i is a random number.
- Under universal hashing, a different set of random numbers can be generated each time the algorithm is used.

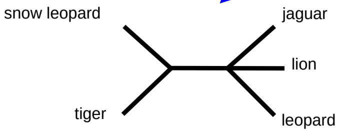
Step 3: Constructing the Consensus Tree



Add trivial bipartition 11111

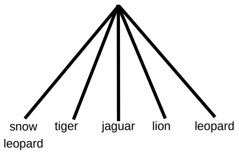


Add majority bipartition 11000

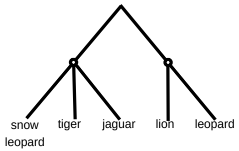


Convert to unrooted tree

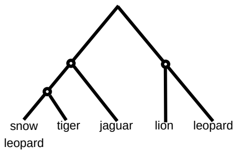
Another Example: Constructing the Consensus Tree



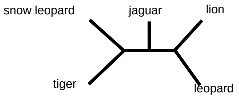
Add trivial bipartition 11111



Add majority bipartition 11100



Add majority bipartition 11000



Convert to unrooted tree

Summary

- There is much debate concerning the true phylogeny of the Panthera genus.
- Although constructing majority consensus trees is a simple problem to explain, it has a wealth of hidden jewels that form the foundation of many computational algorithms such as sorting numbers, hashing objects, and traversing trees.
- Our hope is that our investigation of consensus tree computation inspires undergraduate biology students to learn about other computational ideas in bioinformatics.

Acknowledgments

- Funding: National Science Foundation
- Students: Grant Brammer, Suzanne Matthews, Seung-Jin Sul
- Collaborators: Marc L. Smith, Vassar College
- Special thanks: Tandy Warnow, UT-Austin