TreeJuxtaposer: Scalable Tree Comparison using Focus+Context with Guaranteed Visibility

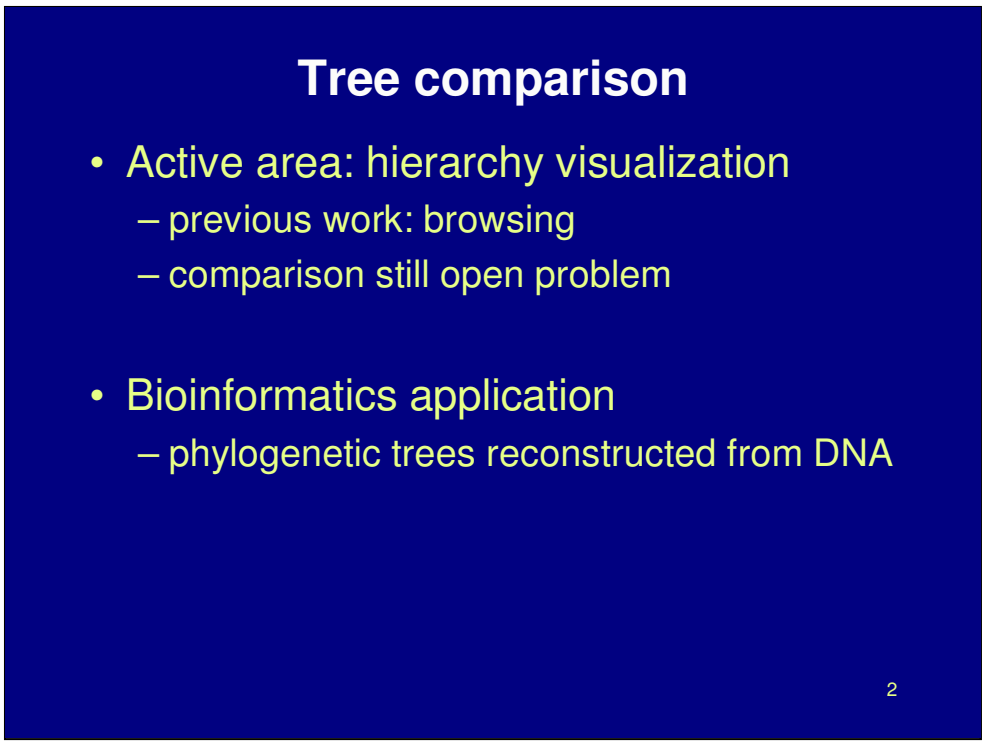
Tamara Munzner
Univ. British Columbia

François Guimbretière
Univ. Maryland College Park

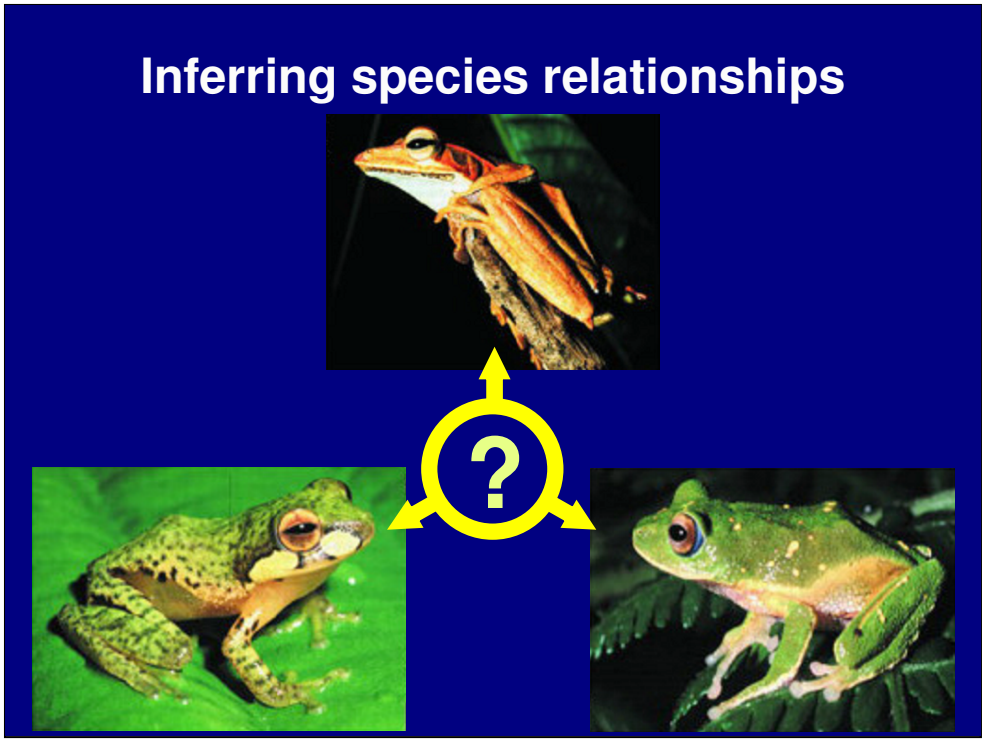
Serdar Taşiran
Koç University

Li Zhang, Yunhong Zhou
Hewlett Packard Systems Research Center

1

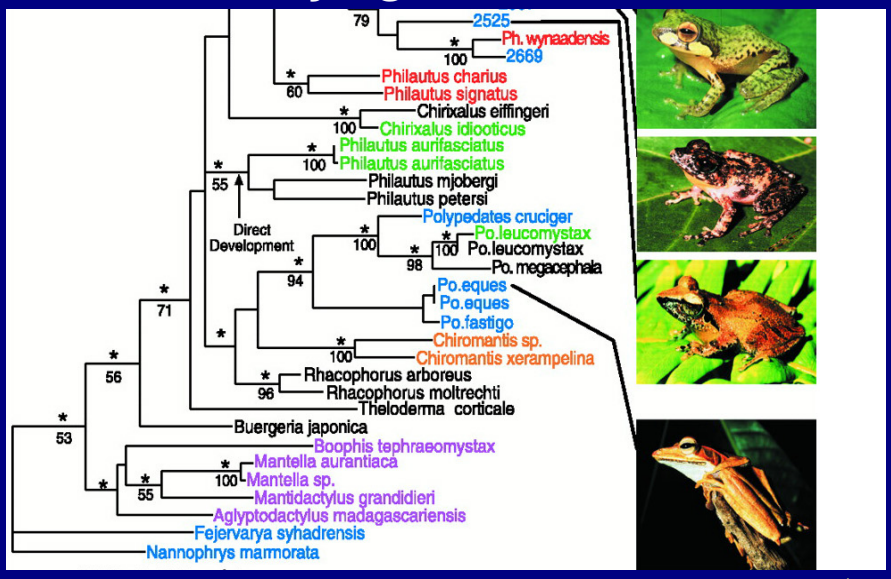


While visualization of large hierarchies has been an active area of research, few tools are available to help users answering a seemingly simple questsion: Given two trees how do they differ? TreeJuxtaposer is a new information visualization system designed to help biologist answer this question when studying phylogenetic trees reconstructed from DNA.



Imagine you were to study how this 3 species of frogs relate to each others. To do so you will construct a phylogenetic tree

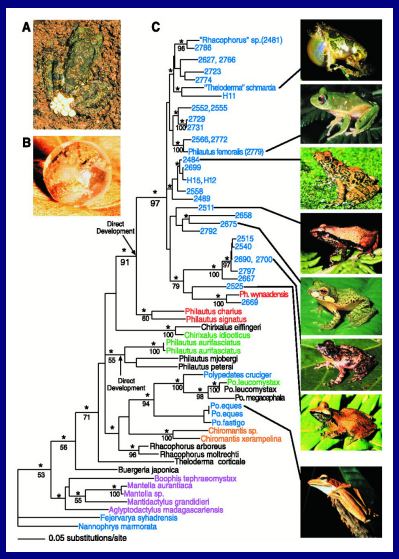
Phylogenetic tree



M Meegaskumbura et al., Science 298:379 (2002) 4

To do so you will build this phylogenetic tree with leaves are species, in this case frogs, and internal nodes are inferred speciation points where one species become two following a mutation for example.

Phylogenetic tree

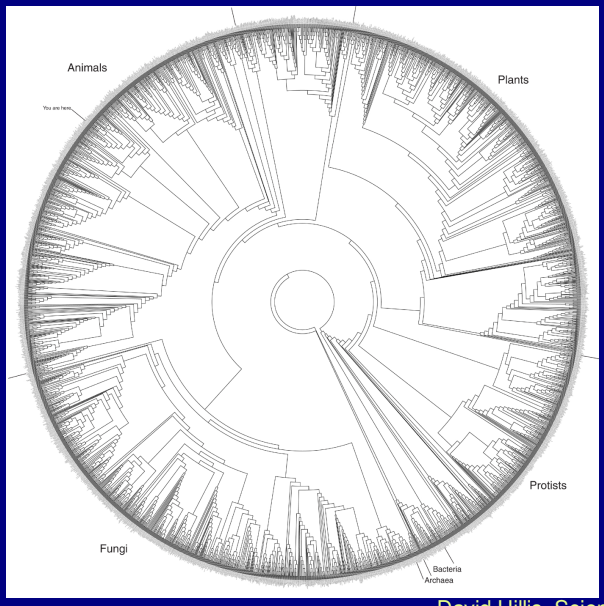


M Meegaskumbura et al., Science 298:379 (2002) 5

While most publication are repporting result on small trees (60 leaves and around 120 nodes) like the one shown here,

Phylogenetic trees are used by biologists to depict and understand the ancestral relationships between species, for example these frogs. The leaves of the tree are existing species, and the interior nodes are bifurcation points where an ancestral species split into two younger species through a specific mutation. These interior nodes representing common ancestors must be inferred from available data.

Tree of Life: 10M species



6

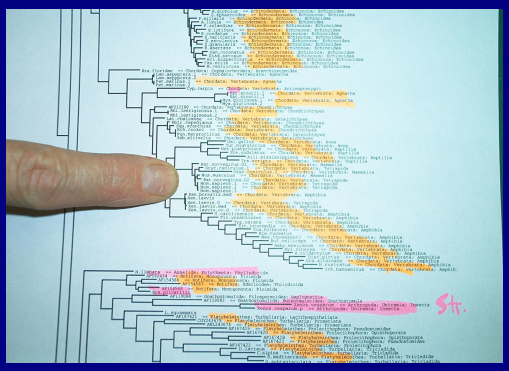
David Hillis, Science 300:1687 (2003)

Several group are tackling a vastly more ambitious problem: building the tree of life the tree of all species known on eath as shown here. This tree is roughly 10M leaves. Its structure is still and active area of research which made it to the front page of science last June.

The entire tree of life which represent all the species leaving on the planet contains about 10M leaves. Notice that mammals is only a tiny fraction of the tree.

Still an open problem – even primates aren't clear, everything is still hypothesized.

Comparing trees: current practice



Will Fischer, postdoc with David Hillis at UT-Austin

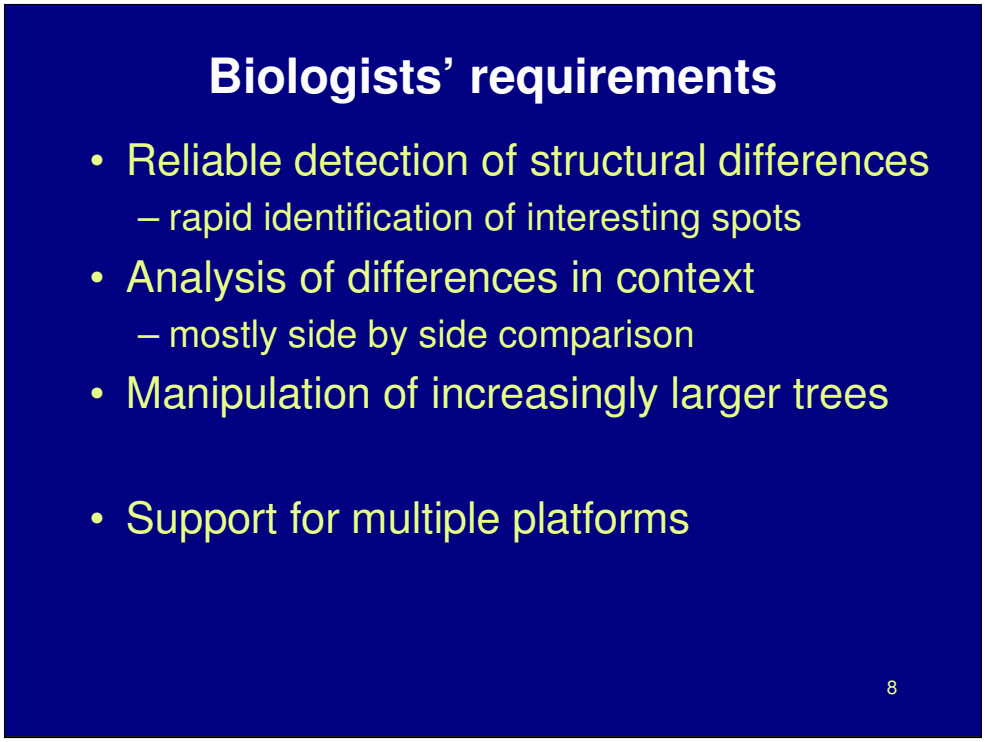
7

In the process of constructing these trees biologist often need to compare them side by side, a task which proves difficult with today's tools, even for tree as small as the 5000 node such as the one shown here. So they often switch to a paper/highlighter interface. As you might imagine this approach is both tedious and error prone.

One of the major step for step in building this tree is to compared side by side to possible candidates to understand the difference and infer with one is the most probable tree. While difference tool like MacClade do exist for small tree, they do not scale for even medium data set forcing user to rely on large hand made collage and print-out. As you might imagine this setting is not only tedious but also error prone.

Comparing trees is hard, and there are no good tools for the job. This biologist taped together many pieces of paper and manually highlighted similar vs. different leaves, showing the need for tools that show both details and their surrounding context.

The task becomes rapidly intractable for even medium side because of the the current tools are not design for large size dataset. This in turn force the biologist to do manual comparison, a laborious and error prone task.



Observing biologists at work, we identified 3 major requirements for their work:

- Reliable detection of structural differences so that they can foucs on the area of interest

- Easy characterization of the difference in their context (This is why they are using this long piece of paper)

- And a smooth manipulation of large tree.

Biologists use a range of platforms, from Macs to Windows to Unix, and we would like to support them all.

TreeJuxtaposer contributions

- Interactive tree comparison system
  - automatic detection of structural differences
    - sub-quadratic preprocessing
  - efficient Focus+Context navigation and layout
    - merge overview and detail in single view
  - guaranteed visibility under extreme distortion
- Scalable
  - dataset size: handles 280K – 500K nodes
  - display size: handles 3800x2400 display

9

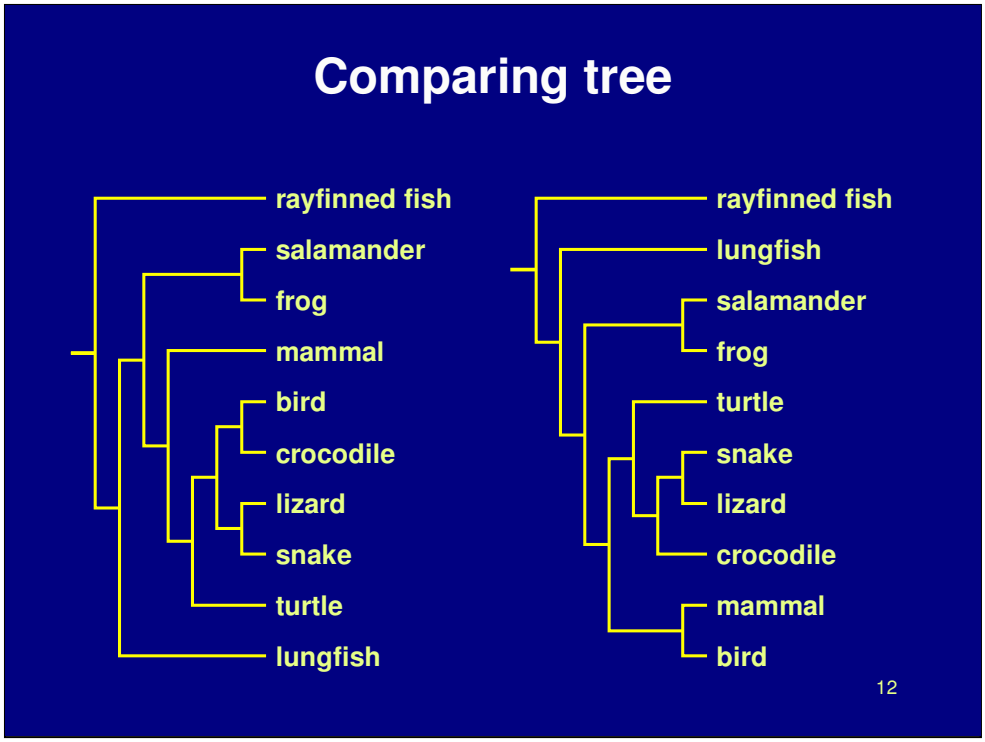
TreeJuxtaposer video

- Platforms shown
  - java 1.4, GL4Java 2.7 bindings for OpenGL
  - Windows
    - 2.4 GHz P3, nVidia Quadro4 700XGL
    - 1.1GB java heap
    - window sizes 1280x1024, 3800x2400
  - Linux
    - 3.1 GHz P4, nVidia GeForce FX 5800 Ultra
    - 1.7GB java heap
    - window size 800x600

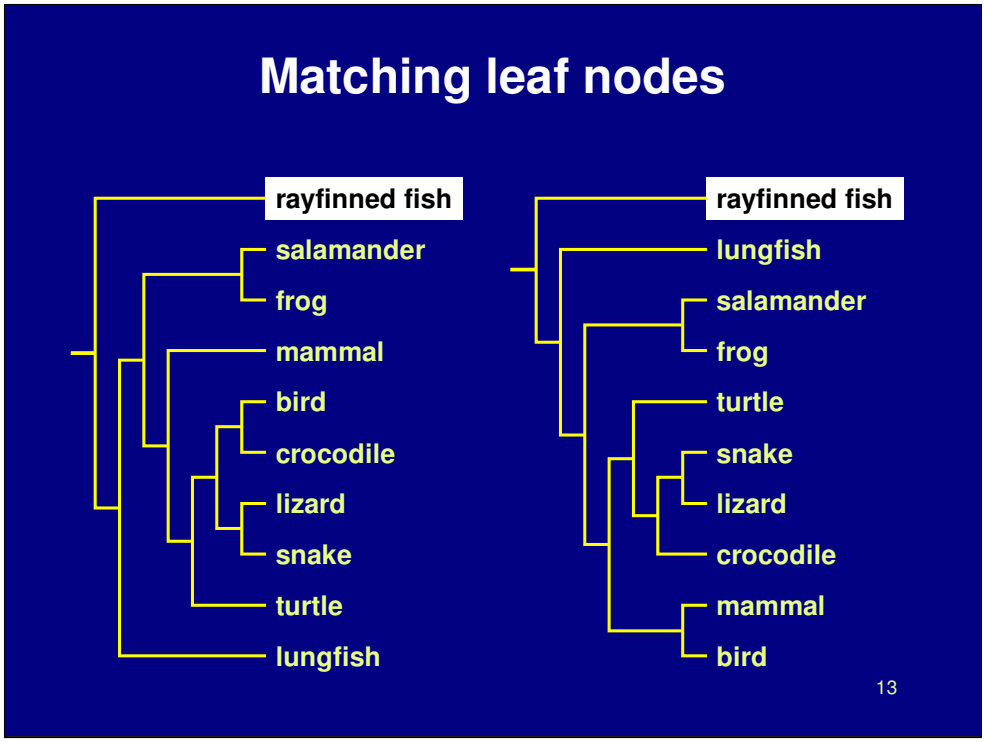
10

Outline

- Application domain: evolutionary trees
- Demonstration
- Computing structural differences
- Guaranteed visibility of marked areas
- Results and conclusions

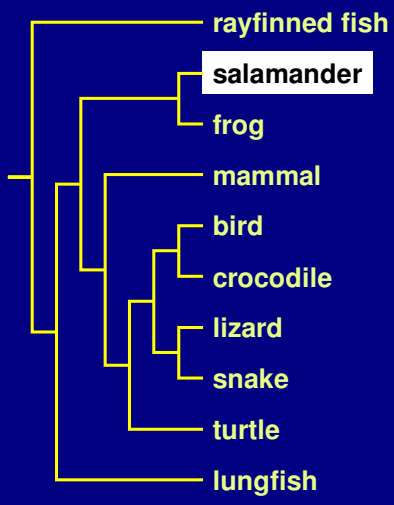


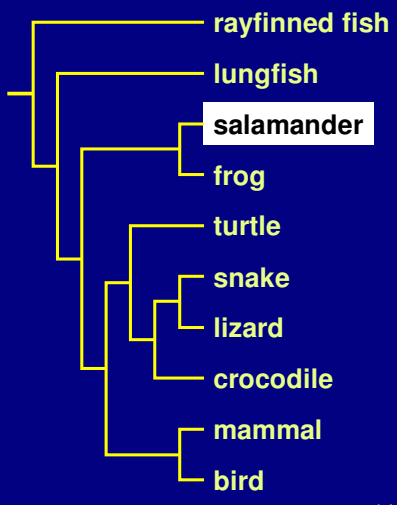
Even for simple tree like the two shown here, identifying structural difference can be quite difficult.



While matching leaf nodes is straightforward,

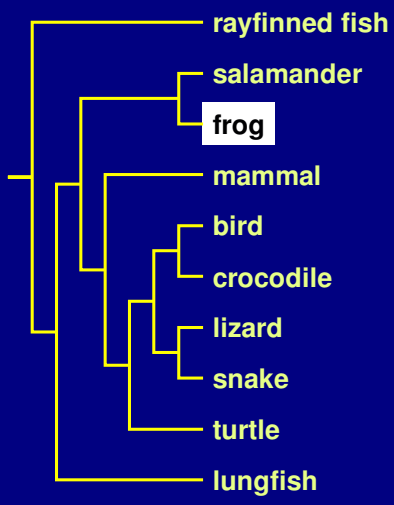
Matching leaf nodes

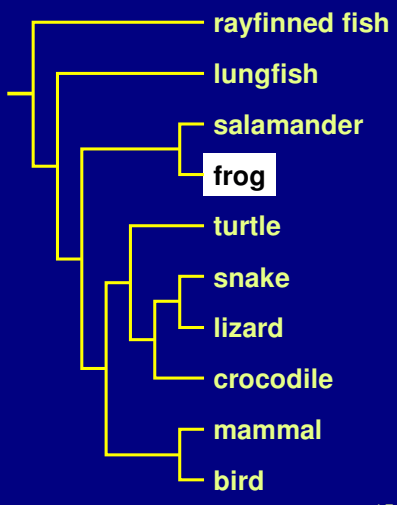




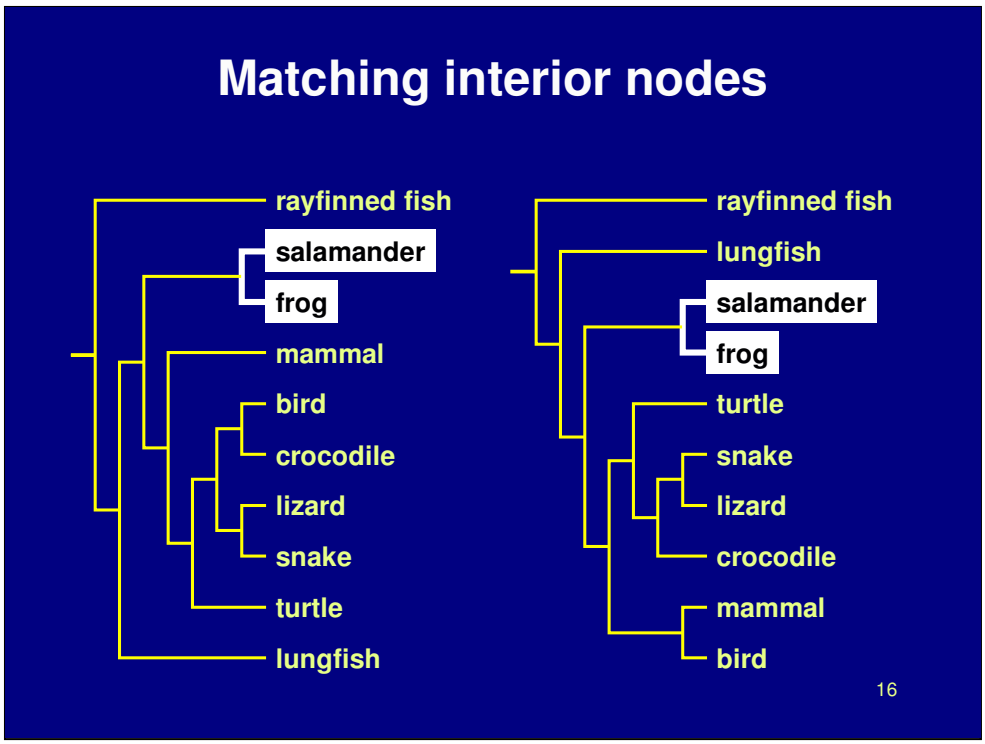
14

Matching leaf nodes

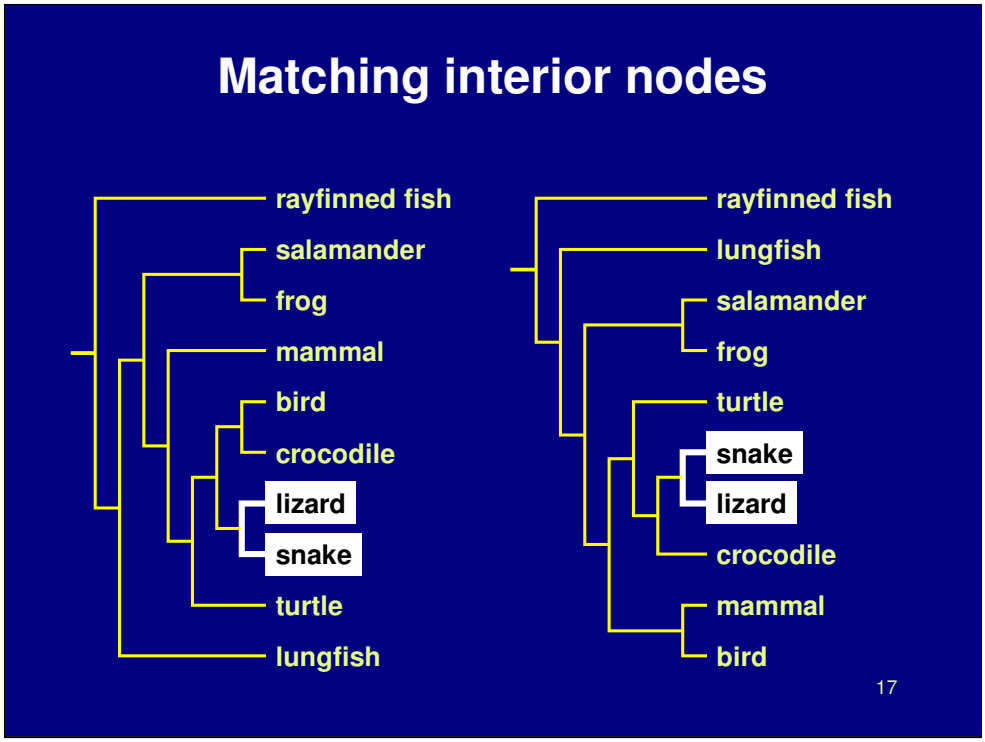




15



Matching internal nodes is far more complicated especially when the internal node do not have labels as it is the case in many of our dataset. Here I show as simple case, matching layout imply a matching sub-tree

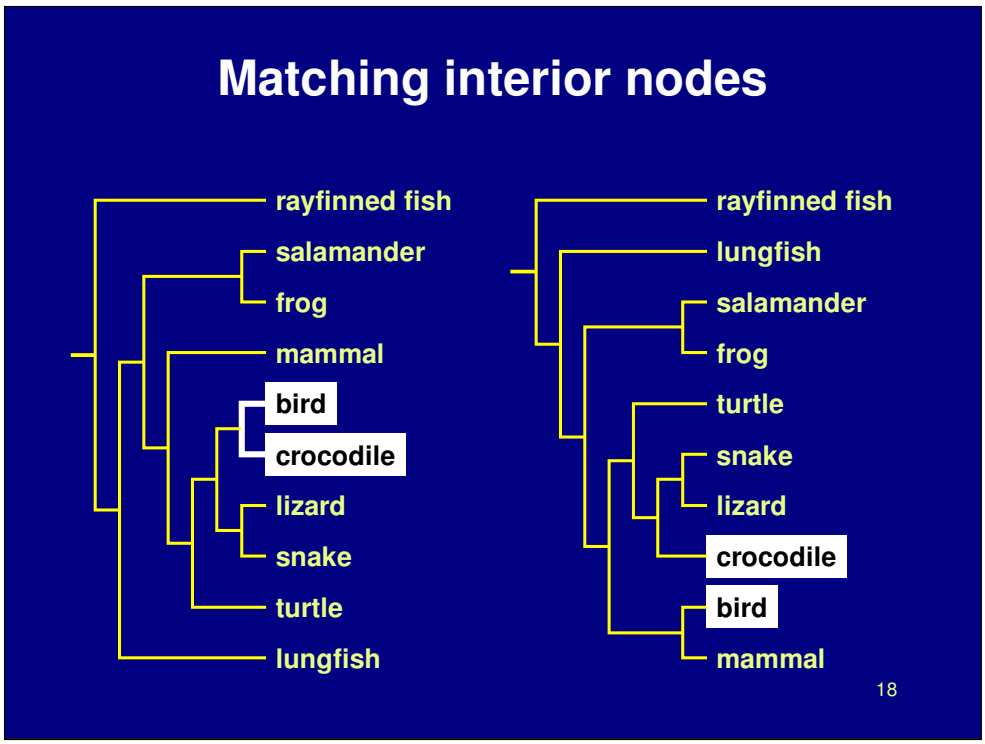


but one has to remember that order is not important while comparing topology

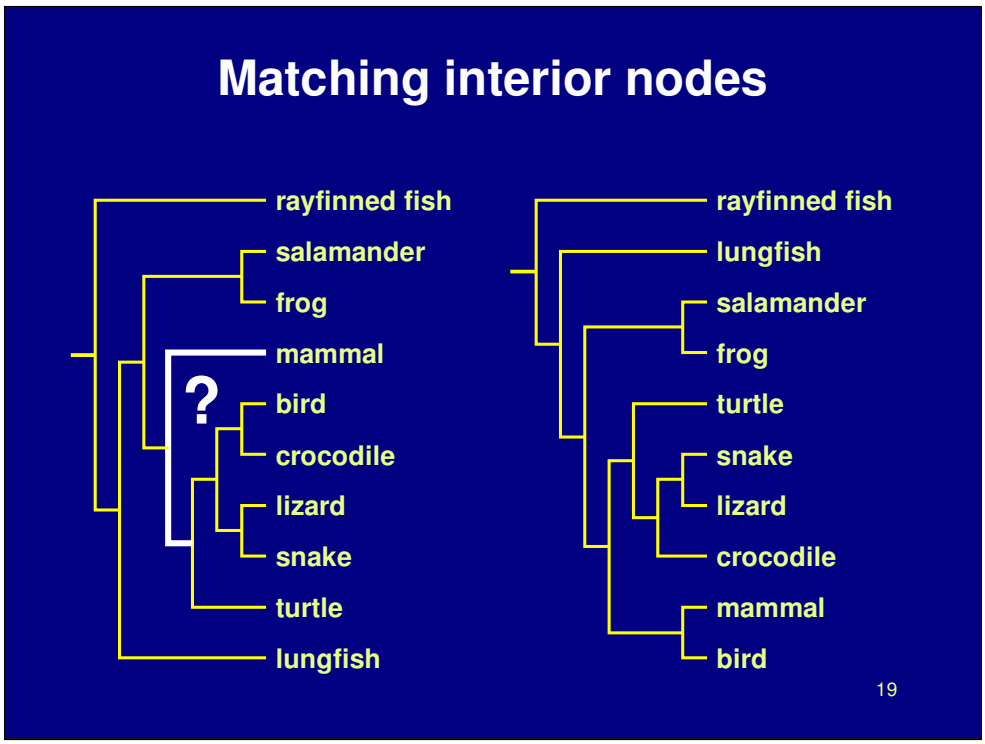
Leaf match: labels

Interior node match difficult

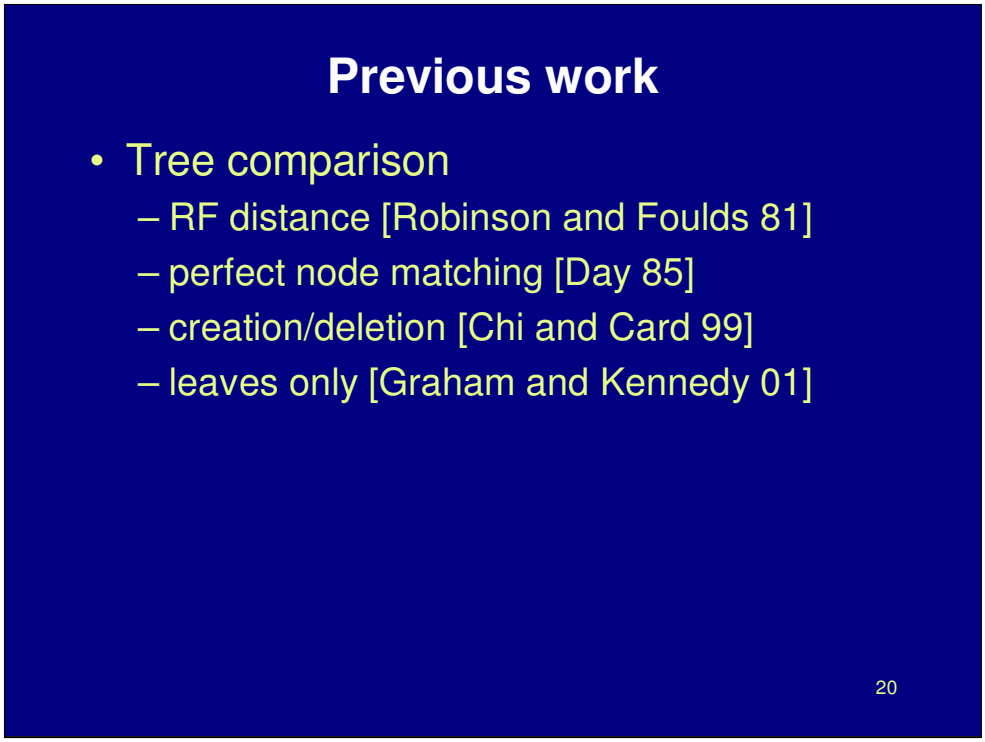
- Often unlabelled
- Siblings unordered



and of course proximity can be confounding too. While bird and crocodile are close to each other here their relationship is in fact quite different in this two trees.



As one consider nodes further from the leaves the situation becomes more and more complicated.



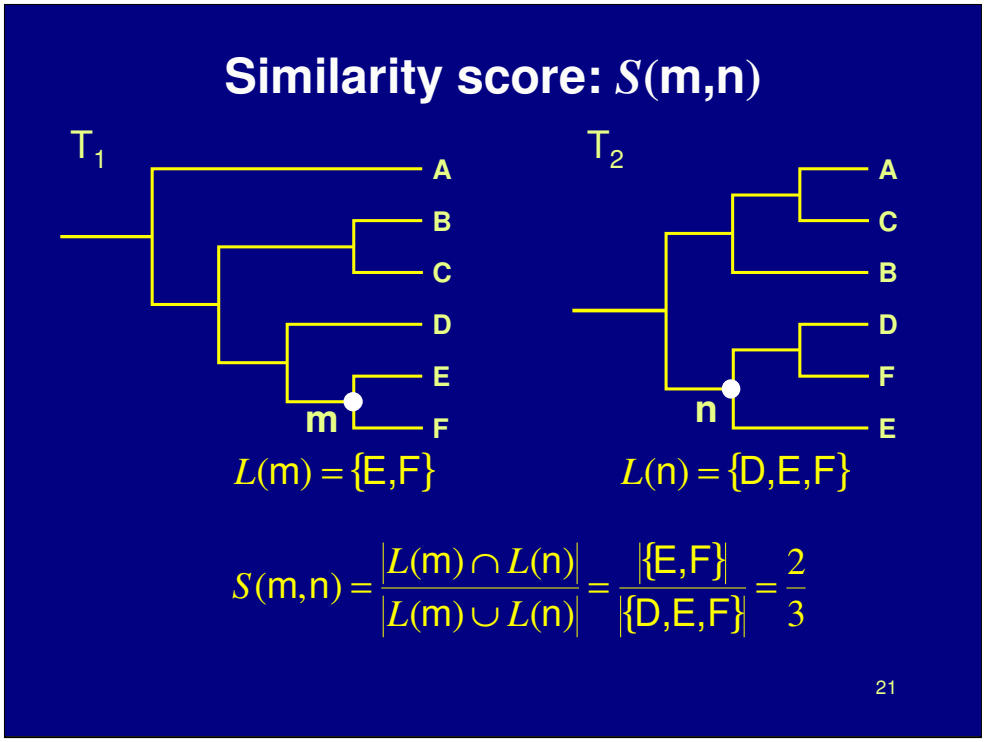
Several metrics have been proposed to to address this problem such as the Robinson and Foulds metric, but none of them completely address the tree comparison problem. They often require tree with the same set of leaves and are aggregate measure providing one number for any given tree and are not well adapted to interactive exploration such as linked highlighting of corresponding nodes as you have seen in the video

RF aggregate measure for a tree

perfect match intead of best match

creation and delection not movement

Leaves not interior nodes.

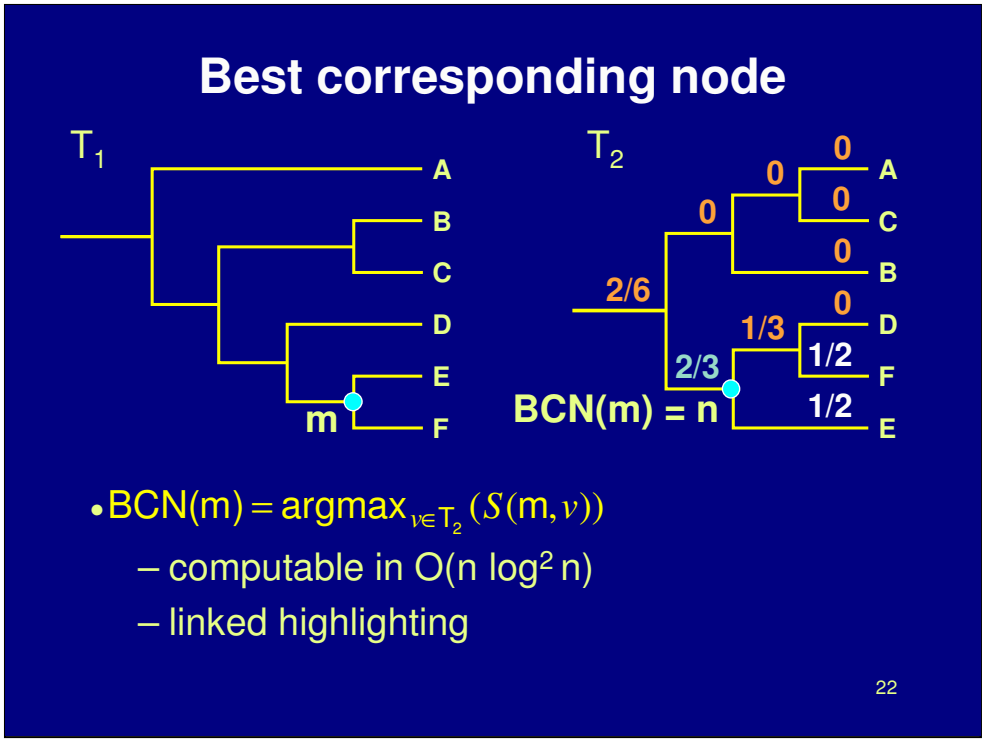


Our work address these two shortcomings. First we extend the similarity measure to trees with different sets of leaves by using the set of leaves below a given node. In this example we compute S(M,N) by first computing the set of leaves below m and n noted L(m) and L(n) respectively. We then compute their intersection and they union. Computing the ratio of the cardinality of the two sets provides us with the reuslt.

This metric can vary from 0 to one when the two nodes share the exact same set of leaves.

Extend the previous measure from perfect match to best match make it better for interactive.

Let's look at an example. Here are 2 tree T1 and T2 and I wish to understand how the subtree rooted at M and the subtree rooted at n are related. The first we compute the similarity score. If L of M is the set of leave under M and L(N) the set of leaves under N, the similarity is given by the formual: cardinal of the intersection of this 2 set to the cardinality of the union of this two sets. in out case L(M) is the set E,F and LN of n is the set D,F,E the the intersection of this 2 set will be ef and the union DEF, and the similarity will be 2/3. This measure extend the previously proposed measure since it does work even in the case where the set of leaves are not identical.



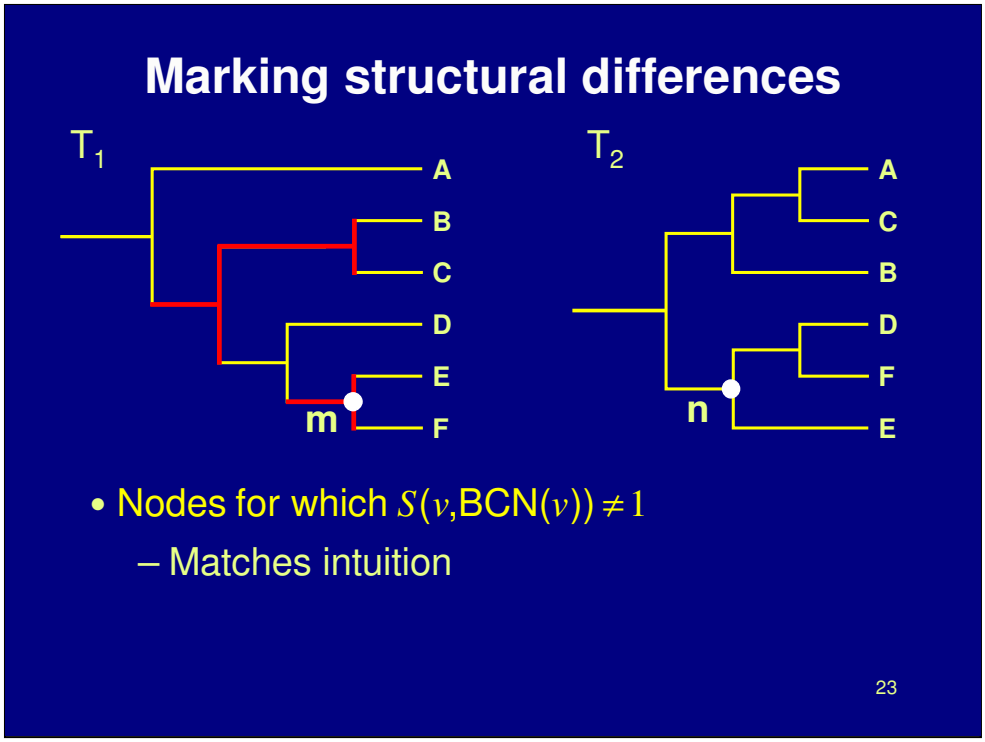
By itself the similarity measure is an aggregate measure, but because it does not require the 2 subtrees to share the same set of nodes. We can now compute the similarity score between a given node in T1 say M and all node in T2 (I did that on the right and you can see that n has the highest score.). Now pick the node with the best score as the Best Coresponding node for M.

As described here it might seems like a very expensive proposition, quadratic in the number of similarity score computations. We provide an algorithm for computing the BCN very efficiently and I will refer you to the paper for more details.

In our system the BCN function is used as the basis of our linked highlighting.

While we won't have time to go over it here, we proposed a efficient O(nlog2n) evaluation algorythm to compute the BCN of each node between 2 tree as a rapide preprocessing path. This information is then use to provide structural link highlighting. and will we refer you to our paper for details.

Once one has the best correponding node for every node in T2 it is then trivial to implement linked highlighting as hown in the video.



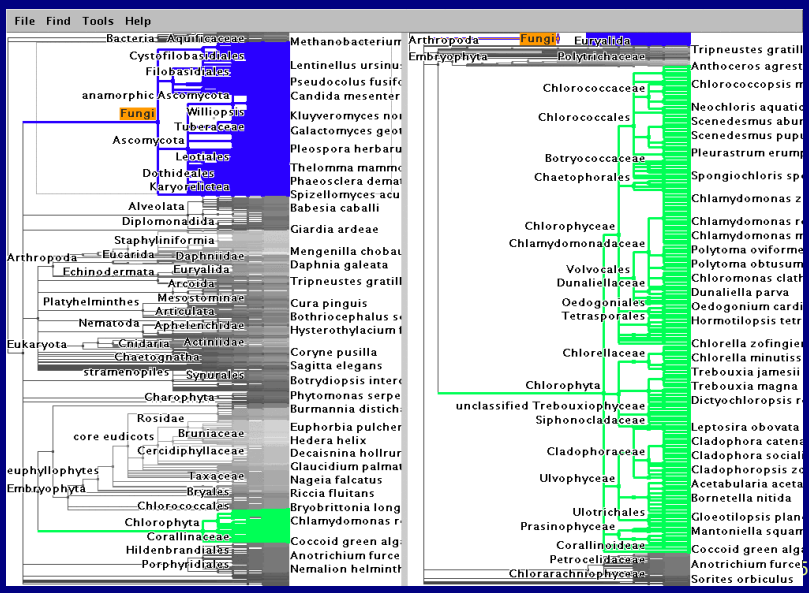
Once we can compute BCN efficiently, it is a simple matter to define structural difference: They are nodes such as M for which the Similarity score between them and their best corresponding node is different from one.

As shown in this picture, it is important to note that even though we started from a metric ignoring the topology, our system deliver precise topology difference.

Outline

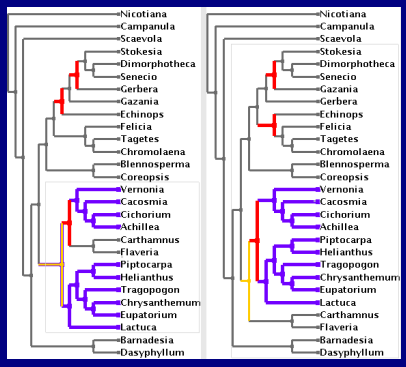
- Application domain: evolutionary trees
- Demonstration
- Computing structural differences
- Guaranteed visibility of marked areas
- Results and conclusions

Guaranteed mark visibility



Marks

- Region of interest shown with color highlight
  - structural difference
  - search results
  - user-specified
- Purpose
  - guide navigation
  - provide landmarks
  - subtree contiguity check



Guaranteed visibility of marks

- How can a mark disappear?

Guaranteed visibility of marks

- How can a mark disappear?
  - moving outside the frustum

28

Guaranteed visibility of marks

- How can a mark disappear?
  - moving outside the frustum
- Solutions
  - choose global Focus+Context navigation
    - “tacked down” borders

29

Focus+Context previous work

- combine overview and detail into single view
- Focus+Context
  - large tree browsing
    - Cone Trees [Robertson et al 91]
    - Hyperbolic Trees [Lamping et al], H3 [Munzner 97]
    - SpaceTree [Plaisant et al 02]
    - DOI Trees [Card and Nation 02]
  - global
    - Document Lens [Robertson and Mackinlay 93]
    - Rubber Sheets [Sarkar et al 93]
- our contribution
  - scalability, guaranteed visibility

30

Guaranteed visibility of marks

- How can a mark disappear?
  - moving outside the frustum
- Solutions
  - choose global Focus+Context navigation
    - “tacked down” borders

31

Guaranteed visibility of marks

- How can a mark disappear?
  - moving outside the frustum
  - occlusion
- Solutions
  - choose global Focus+Context navigation
    - "tacked down" borders

32

Guaranteed visibility of marks

- How can a mark disappear?
  - moving outside the frustum
  - occlusion
- Solutions
  - choose global Focus+Context navigation
    - "tacked down" borders
  - choose 2D layout

33

Guaranteed visibility of marks

- How can a mark disappear?
  - moving outside the frustum
  - occlusion
  - culling at subpixel sizes
- Solutions
  - choose global Focus+Context navigation
    - "tacked down" borders
  - choose 2D layout

34

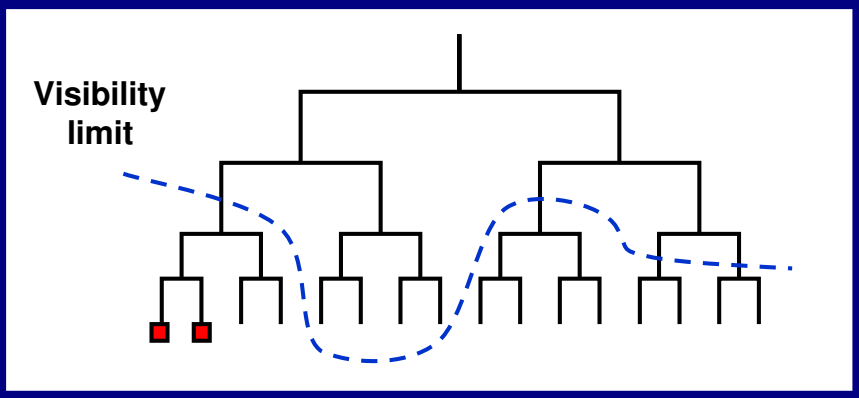
Guaranteed visibility of marks

- How can a mark disappear?
  - moving outside the frustum
  - occlusion
  - culling at subpixel sizes
- Solutions
  - choose global Focus+Context navigation
    - “tacked down” borders
  - choose 2D layout
  - develop efficient check for marks when culling

35

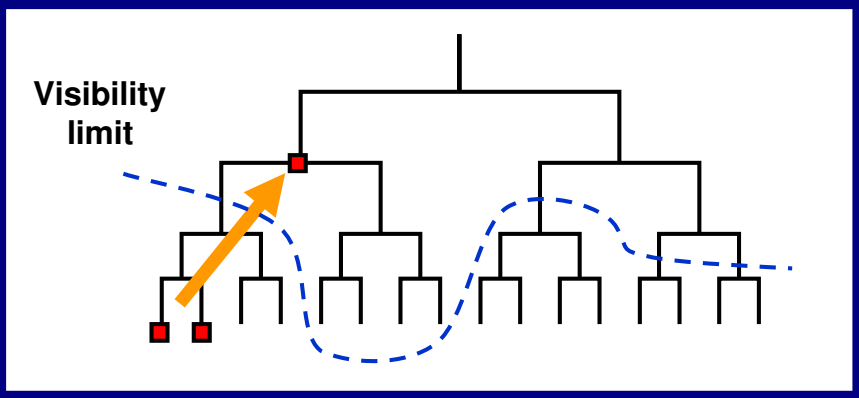
Preserving marks while culling

- Show mark at unculled node



Preserving marks while culling

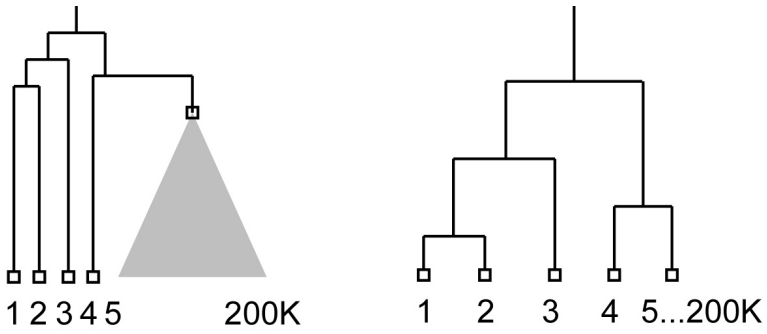
- Show mark at unculled node



37

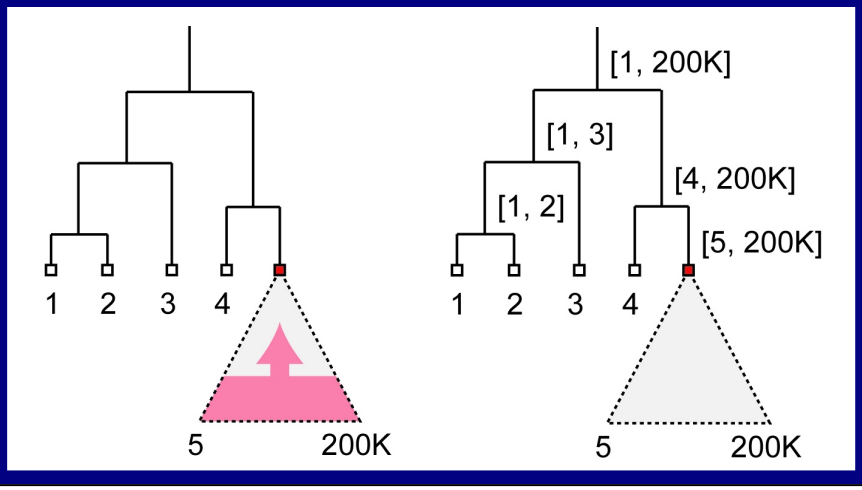
Mark preservation strategies

- Compress large subtree to small spatial area



User selects nodes [135,199995]

- Propagation : cost depends on total nodes
- Precomputation: cost depends visible nodes



What if the user selects nearly all the nodes. This one pixel area needs to be red. If we propogate upwards from the selected nodes to the onscreen node that should be marked, the total cost of the operation depends on up to the total number of nodes in the subtree, which is a lot of work to cover just a few pixels. In contrast, if we precompute the ranges below each node at startup time, we do a simple range comparison the first time we render a node to know if it's highlighted or not. In this case it's 10 set comparisons vs. nearly 200,00 operations. This specific point illustrates our general design principle of our system: while rendering, the work we do should depend only on the number of nodes visible onscreen at once, not the total number of nodes in the dataset.

Marks and linked highlighting

- Also check for linked marks from other tree
  - check if best match for node is marked
    - up to O(n) to look up each node in range
  - intersect node ranges between trees
    - reduces to point in polygon test, $O(\log^2 n)$

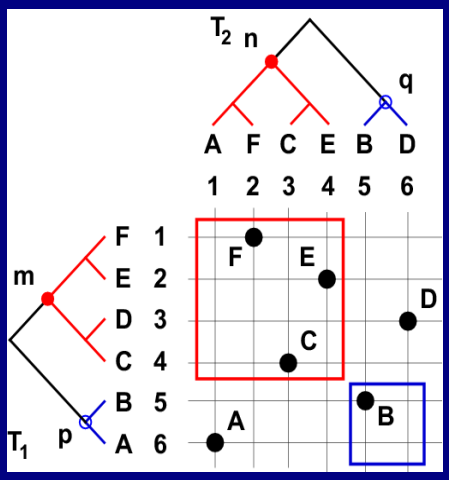
40

so range checks are great, but now how do we accommodate linked highlighing between trees? checking each node in the range against its best matching node will bring us back to the situation where a decision about a few pixels requires work dependent on the total nodes beneath it.

we found a way to combine range checking with best corresponding node mapping in one fast operation. it's a nice geometric reduction to a point in polygon check that's log^2 n, and the details are in the paper.

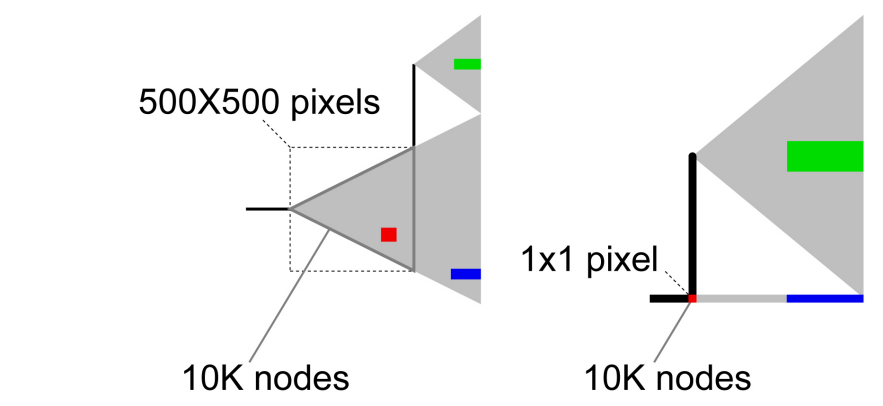
Efficient marking detection

- Intersecting ranges between trees
  - Query in $O(\log^2 n)$



Storing topological ranges

- At each node, store range of subtree beneath
  - range stored doesn't match spatial range needed

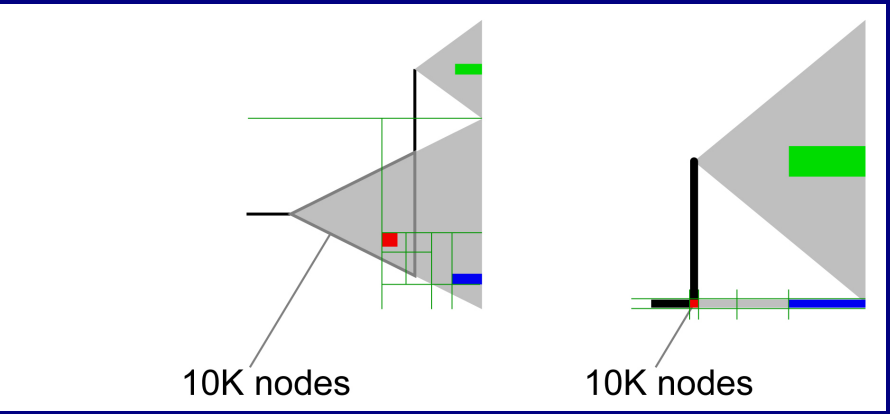


42

we've been talking about ranges, how should we store them? it's tempting to use the existing hierarchical structure of the tree itself, and just store ranges of the entire subtree beneath a node. but if we do this, we will again be in a situation where a small onscreen area computation depends on the total nodes beneath it. here's an example: we start with a tree on the left, and compress it on the right. We must mark the single pixel available for this set of nodes in red, because it contains a marked area. The problem is that the range we stored doesn't correspond to the part that we compressed. The subtree range we can quickly check includes all three colors, but the compressed area includes only the red mark that was initially inside the 500x500 pixel box. Thus our range check would be wrong, and we're back to an traversal of all 10,000 nodes.

Storing spatial ranges

- At each box, store range of objects inside



43

We can instead store ranges in a spatial hierarchy, and use our efficient range check algorithm for any specific area of the screen. This accomodates any possible deformation created by user navigation.

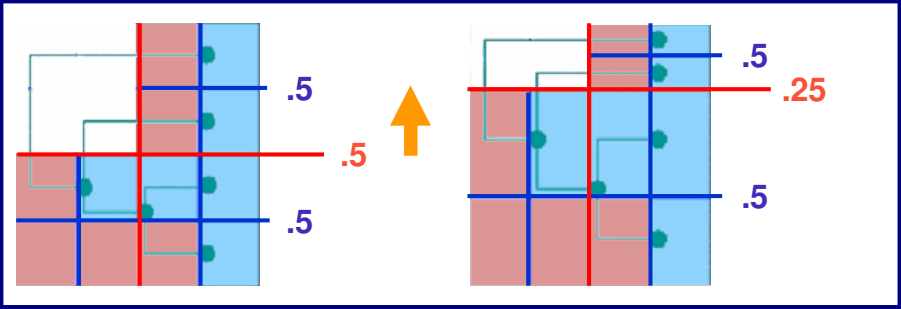
Spatial range solution

- Recursive spatial subdivision
  - quadtree
  - store range of objects enclosed for each cell
  - quick check: spatial range vs. selection range
- Extending quadtrees to Focus+Context
  - quadtree cells also “painted on rubber sheet”
  - efficient O(log n) update when stretch/shrink
    - details in paper

44

Rendering infrastructure

- Focus+Context QuadTree
  - Fixed mapping between nodes and quad cell
    - Sparse cell instantiation
  - Split boundary relative to the node parent
    - Hierarchical propagation of deformation



Guaranteed visibility previous work

- Visibility of abstract information
  - Effective view navigation [Furnas 97]
  - Critical zones [Jul and Furnas 98]

Outline

- Application domain: evolutionary trees
- Demonstration
- Computing structural differences
- Guaranteed visibility of marked areas
- Results and conclusions

47

Difference computation

- Powerful and totally automatic
  - leads users to important locations
  - efficient algorithms: 7s for 2x140K nodes
  - matches intuition
    - UT-Austin Biology Lab, several others
- Challenges
  - memory footprint
  - handling weighted edges

48

Guaranteed visibility

- Relief from exhaustive exploration
  - missed marks lead to false conclusions
  - hard to determine completion
  - tedious, error-prone
- Compelling reason for Focus+Context
  - controversy: does distortion help or hurt?
  - strong rationale for comparison

Guaranteed visibility challenges

- Integration with progressive rendering
  - might lose context during motion
  - need several seeds for rendering queue
    - focus point
    - marked items
  - up to empirical cutoff, no guarantees
- Constraint to fit everything in frustum
  - instead could show indirectly

50

Future Work

- Adoption
  - open-source release
  - tighter integration with biology tools
  - broad range of application domains
- Detectability vs. visibility
  - display resolution, surrounding colors
- Extend difference computation
  - weighted trees
  - graphs

51

Conclusion

- First interactive tree comparison system
  - automatic structural difference computation
  - guaranteed visibility of marked areas
- Scalable to large datasets
  - 250,000 to 500,000 total nodes
  - all preprocessing subquadratic
  - all realtime rendering sublinear
- Techniques broadly applicable
  - not limited to biological trees

52

Acknowledgments

- Biologists
  - David Hillis, Bob Jensen, Will Fischer, Derrick Zwickl
- Computer scientists
  - Nina Amenta, Katherine St. John
- Partial funding
  - NSF/DEB-0121682

- Talk preparation
  - Mary Czerwinski, Pat Hanrahan, George Robertson, Chris Stolte, Diane Tang, Gina Venolia