# TreeJuxtaposer InfoVis Contest Entry

James Slack *
University of British Columbia

Tamara Munzner †
University of British Columbia

François Guimbretière ‡
University of Maryland

## 1 Abstract

TreeJuxtaposer is a tool for interactive side-by-side tree comparison. Its two key innovations are the automatic marking of topological structural differences, and the guaranteed visibility of marked items. It uses the AccordionDrawer approach for layout and navigation, a multifocus global Focus+Context approach where stretching one part of the tree or screen causes the rest to shrink, and vice versa. Progressive rendering guarantees immediate interactive response even for large trees.

## 2 Introduction

We showcase TreeJuxtaposer in our contest entry, a system recently created by one of the authors for the visual comparison of large trees [Munzner et al. 2003]. Our target audience was biologists comparing phylogenetic trees, so we were delighted by the topic choice of this year's inaugural InfoVis contest.

Although our tool is specifically tuned for the needs of evolutionary biologists, we have asserted that it is applicable in a wide variety of domains. We were pleased to back up this assertion with strong results for many of the file system log data questions.

The TreeJuxtaposer [Munzner et al. 2003] interface is built around navigation by growing and shrinking areas. It also supposes very fast querying by mousing or keyboard across a dense visual representation of the tree. We compute the "best corresponding node" from one tree to another, and use this information both for linked highlighting and determining exact areas of structural difference to be marked,

## 3 Strengths

Our system has many strengths. From the first startup image alone, we can immediately answer many questions because we explicitly mark the exact places where structural differences occur. We can instantly characterize whether changes include additions or deletions to the leaves, based on whether the red difference marks occur in the leaves (additions/deletions) or solely in the interior (moving around existing nodes rather than adding or deleting them) as different. For example, we immediately saw that the classification datasets are almost all additions and deletions, and the phylogenetic dataset changes are all the result of moves with no adds/deletes.

Linked highlighting is also a powerful feature when interacting with the trees, especially in conjunction with our design decision to use a very dense visual representation of the tree and support extremely fast mouse over pop-up highlighting (the latter using front-buffer drawing tricks to avoid requiring a full redraw). The video shows how simply moving the mouse around the screen for a few seconds imparts a great deal of information about the structure at both high and and low ranks. When the trees are quite different, the pop-up highlight on the other side skitters about a great deal. For similar trees, the linked highlight is more sedentary.

The core navigation paradigm is growing and shrinking areas, allowing multiple focal areas to support inspection of multiple spots within a single tree. A particularly powerful feature is the ability to simultaneously grow or shrink every item in a marked group. Linked navigation is heavily used, because usually seeing the corresponding areas grow and shrink in "slave" mode while interacting with a "master" tree. Although we do support unlinked navigation, we note that it is used only rarely.

Guaranteed visibility of marked areas is one of the major reasons for our success at the contest tasks. For instance, the incremental search is useful even for the full classification dataset of 200K nodes, because a marked leaf is visible even from the global overview level. Guaranteed visibility is extremely helpful for comparison tasks, because the alternative is exhaustive search. Without guaranteed visibility, it is hard to know when to stop hunting for marks; marks could lie outside the viewport, be occluded by other objects, or even if these two constraints are met they might be invisible simply because they are culled when they project to less than one pixel of screen area. We conjecture that guaranteed visibility dramatically shortens the time required for exploration and analysis. However, we have no empirical proof because we did not test a second person with the tasks using a version of TreeJuxtaposer that disabled guaranteed visibility.

This operation is a very quick way to understand structural differences and we do it extensively to answer the contest questions. Also, the incremental search function is a marking approach heavily used in our answers, because it shows the results situated in their usual context rather than out of context. The incremental search extension provided fine control of TreeJuxtaposer that did not exist in the previous work [Munzner et al. 2003] since it allows users to search for nodes by name. See, for example, how Figure 1 shows all nodes found with "dolphin" in their common name. The partial matching provides the power to seek any node by substring matching and visually represents the found nodes with guaranteed visibility and negligible run-time or start-up overhead. In addition to marking nodes known by name, the searching interface allows users to browse through the search results and selectively mark nodes from the search if any undesirable search result occurs. Another improvement from the previous paper is changing the progressive rendering algorithm to use multiple seeds for the rendering queue rather than starting only with the focus cell of the last user interaction. We now add the first few items from each of the marked groups in the queue when starting a frame, so it is easier to maintain context when interacting with large datasets.

## 4 Weaknesses

One major weakness is that we make no attempt to handle attributes, so we leave several questions unanswered. If we had the time to spare, we could have implemented an interface where various attributes could be manipulated: marked with colors, and grown or shrunk. The internal infrastructure of TreeJuxtaposer would easily support this functionality, since would use the same underlying mechanism as our current interface that allows interactive manipulation of groups. Although we already have the infrastructure and the required parser would be straightforward, it would not be trivial

*Email: jslack@cs.ubc.ca
†Email: tmm@cs.ubc.ca
‡Email: francois@cs.umd.edu

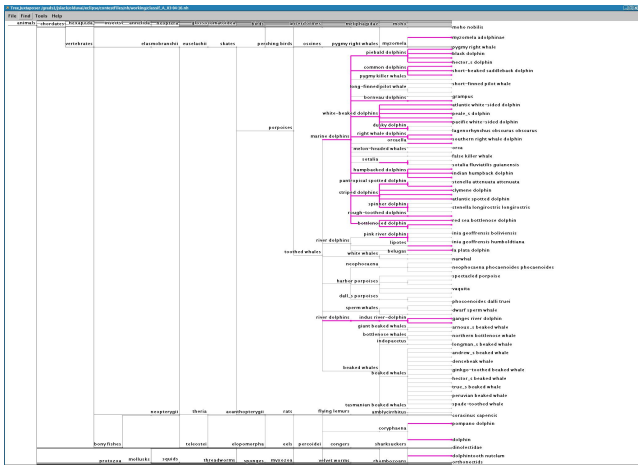

Figure 1: Result of an incremental search query on “dolphin” in classif_A, common names, with all results grown

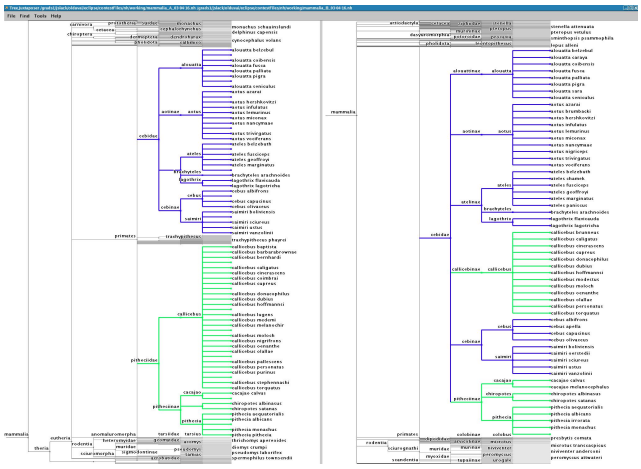

Figure 2: Structure movement shown by marked subtrees: classif_A on the right and classif_B on the left

to create a usable user interface for this sort of exploration.

Although TreeJuxtaposer is very powerful for a fairly large set of tasks, it is not a flexible or general-purpose system. For instance, we do not support editing at all. Another weakness is the current lack of undo support or history tracking.

We were able to load and interact in real-time with a single large classification tree of two hundred thousand nodes. However, we were not quite able to load both huge trees at once for side by side comparison. (We ran out of memory: an unfortunate java limitation is that on 32-bit machines the heap size cannot grow past 1.8GB.) We thus answer all of the classification comparison questions for the Mammalia subtree only.

# 5 Contest Results and Discussion

## 5.1 Phylogenies

The trees in the phylogenies tasks were handled easily by TreeJuxtaposer. The structural differences were no problem for the difference computations and TreeJuxtaposer noted that no leaf nodes were added or deleted between the sample trees provided. The input order of the nodes did not affect the final matching of TreeJuxtaposer, only the top-to-bottom drawing order.

We found that the structural differences in the internal nodes are varied; some subtrees we chose to mark in phylo_A_ABC were very spread out as forests in phylo_B_IM while other subtrees only differed by a slight perturbation in structure. The largest subtree we were able to find in the unmodified trees (we did not use the property of these trees being unrooted) was five levels deep.

## 5.2 Classification Trees

Unlike the trees in the phylogenies tasks, the classification trees had more lower-level differences in structure while larger subtrees (such as rodentia) were not being classified differently between trees. The classification differences were mostly additions and deletions (classif_B had 7717 leaf nodes more than classif_A, each tree has over two hundred thousand nodes) but some other types of structure changes such as the one in Figure 2 were also noticed.

The differences when comparing the Latin versus the common naming conventions in the classification trees were also quite interesting. The common names were not consistent and produced many differences (most of both trees were marked different, which did not provide useful information) while the Latin names provided a better insight into the subtree changes in the overall tree structure. The interactive mouse navigation and browsing features of TreeJuxtaposer are easy enough to use to find any animal with knowledge about basic animal physiology.

## 5.3 File System Logs

We were able to concurrently manipulate all four logs after reducing the set of the logs to the /projects/hcil as well as do pair-wise comparisons on each of the full trees. There were fewest differences between logs_A and logs_B so they were the most interesting to compare in a pair-wise manner. Each of the differences noticed in the four-way comparison were examined in detail.

Determining which directory contained the largest number of files (leaf nodes) was easy with these data files since there were a few leaf-quantity-superior main directories in the file structure. Immediately after loading a log file, the biggest directories (users, class) pop out with their leaf nodes fanning out on the right side of the tree; this puts visually attractive large gaps between the big directories and their smaller neighbors.

# 6 Conclusions

TreeJuxtaposer is useful in automated and interactive tree comparison. The simplicity of the interface and the fluidity of the interaction allows users to concentrate on the more interesting tasks such as the ones provided by this contest. TreeJuxtaposer is flexible enough to handle many different types of tree structures as well as compare several trees side by side. Although the current toolset for TreeJuxtaposer lacks utilities for full attribute analysis, it’s clear that interface modifications will provide an attribute-capable comparison tool with the infrastructure that we have provided.

# References

MUNZNER, T., GUIMBRETIÈRE, F., TASIRAN, S., ZHANG, L., AND ZHOU, Y. 2003. TreeJuxtaposer: Scalable tree comparison using Focus+Context with guaranteed visibility. In *Proc. SIGGRAPH 2003*.