A Visualization Solution for Astrophysics: Generating Galactic Merger Trees Using D3

Laurel Orr **Jennifer Ortiz**

INTRODUCTION

With the help from high-performance computing, scientists have been able to run large scale simulations to model the behavior of complex, natural systems. The amount of data generated during this process is so massive that it becomes challenging to analyze and interact with the data using traditional methods. In the field of cosmology, astronomers have run these large scale simulations to model the behavior of particles interacting from the Big Bang up to present day (a span of 14 billion years). The end goal behind these simulations is to be able to better understand how galaxies such as the Milky Way form and have evolved over time. In particular, cosmologists want to analyze which galaxies merged together over time to create the galaxies we see today. These hierarchical mergings across a span of time are typically represented as tree structures where each level represents a time step and the root represents the present day galaxy as seen in Figure 1.

To generate these trees, cosmologists take the raw particle data and at each time step, cluster the particles into galaxies (also called halos). These halos represent the nodes of the tree. They then generate the edges between nodes by tracing the location of the particles at each time step. Once the trees are created, cosmologists are particularly interested in exploring the structure of the trees, finding major merger events, and finding galaxies that might be of interest to explore further. Unfortunately, generating these merger trees from simulation data and visualizing them is time consuming and not trivial. In the cases where they might want to visualize a merger tree, cosmologists are often forced to draw them by hand and lose the exploratory benefits of using an interactive visualization. In this paper, we discuss the process behind generating the data required to build and eventually visualize the trees. Following this discussion, we describe the visualization features in-depth followed by an evaluation and a summary of future work.

RELATED WORK

There has been prior work focused on building merger trees in other sciences with the majority being in biological sciences. In biology, merger trees are commonly used to represent phylogenetic trees as seen can be seen in the tree of life [3]. This project focuses on exploring and visualizing the evolutionary relationships between living things on Earth. Similar concepts such as time steps and merging exist in these graphs, but unfortunately, there has been less work done in building and visualizing these types of trees in cosmology.

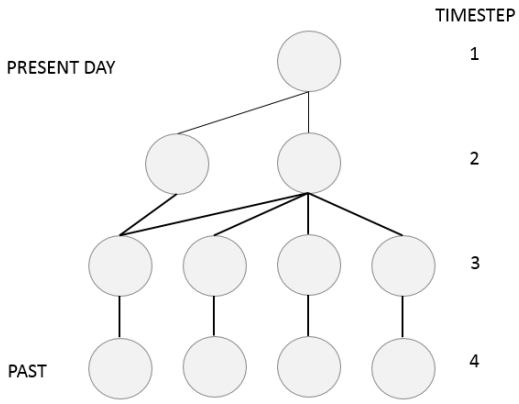

Figure 1. Merger Tree

There has been some work in cosmology focused more on analyzing particle simulations. For example, a project by [7] focused on building an open source toolkit to allow for analysis of astrophysical simulations. The toolkit is organized around physically relevant structures, such as halos, compared to specific simulation code and structure. They support a variety of simulation input styles and provide functions to output GraphViz data that contains the merger tree result. Another tool by [4] allows for reading specific astrophysical simulation files, includes useful calculations such as moments of inertia, and most importantly, is parallelizable, which allows for real time analysis and visualization. However, this tool is not as user friendly as it requires specific input formats, and both projects do not allow for much interactivity from the user.

Another group extended the work on galactic merger trees by providing a 3D visualization of the merger history [6]. They specifically focus on detailing possible dynamics and events that may happen when mergers occur. Since they focus on specific events between mergers, other factors like the overall structure of the tree cannot be seen through this type of visualization.

METHODS

There were three main phases in this project. First, we gathered the appropriate data in order to generate the tree, and as features were introduced, we reused the old queries to generate new data. Second, we developed our visualization and continually modified the layout and added interactive fea-

tures. Last, we were able to evaluate our visualization by presenting it to a couple astronomers at UW who were collaborating with us throughout this project.

Data Processing

The simulation data provided consists of 33.6 million particles throughout 27 time steps where each time step corresponds to a snapshot of the universe at a specific time since the Big Bang. Each particle has a unique identifier and each is one of three types: dark, star, or gas. Each time step is a separate dataset containing the unique identifier of each particle, its location in a 3D space, luminosity value, mass, velocity, and halo group number. The group number refers to the halo cluster that a particle belongs in at each time step. If the group number for a particle is 0, this means that the particle does not belong to any halo at that specific time step. Between time steps, particles may either stay in their current halo, move to another halo, or move to a state where they do not belong to any halo. We can explore the history of a halo by tracing the locations of its particles across time.

In order to generate the edge and node list needed to build the merger tree, we first created two tables, HaloTable (HaloID, GrpID, TimeStep) and ParticleTable (ID, HaloID, Type, Mass, Luminosity). From these tables, we queried across all time steps through a series of self-joins to create the edges. The computation of an edge list for all trees representing the histories of all present day halos took approximately 40 minutes. For this visualization, we pruned the data to only include merger histories for present day halos that have a similar mass to the Milky Way, resulting in a total of 365 halos.

Visualization Features

Upon first interacting with the astronomers, there were certain features they desired for the visualization. One included the ability to see the entire structure of the tree and collapse nodes for easy navigation. They also wanted the nodes sized according to mass and information on the number of dark particles and total particles in each node to be accessible. From this criteria, we decided to construct this visualization using D3.

Tree Structure

The first main concern was to generate a tree layout. Using D3, the two main options to visualize a graph were a force-directed layout and a tree layout. We initially did not consider the force-directed graph layout, but upon discovering that the halo histories were not strict trees and had some instances of halos splitting (a child having more than one parent), we had to consider more generic graph layouts. The main advantage of a force-directed layout was its ability to represent the DAG structure of the histories. The disadvantage was that without more advanced modifications, it did not layout the tree in a way that made it easy to see each level for each time step. Constraints had to be added in order to position nodes to easily understand and see the structure as a tree. On the other hand, the tree layout solved this problem by laying out the nodes such that the nodes were lined up at each time step. The disadvantage for this layout was that it broke when it tried to layout a tree in cases where a node splits.

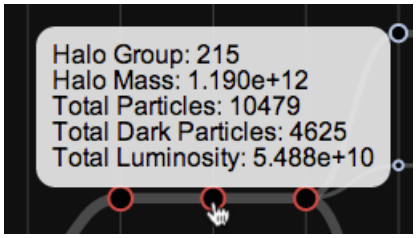

Figure 2. Tooltip that displays when hovering over a node

There were four possible solutions to this problem. We could duplicate the child that had multiple parents, or we could only keep one parent for each child. Both of these were problematic since they inaccurately represented the data and could mislead the astronomers. Another solution was to use GraphViz to dynamically generate layouts by compiling GraphViz into javascript, but the main problem was that if nodes were collapsed, the positioning of the nodes in GraphViz changed drastically in a way that would complicate transitions and confuse users if they wanted to collapse nodes. A final solution was to use GraphViz to provide initial positions and then build our own layout in D3. After discussing the problems and solutions with the astronomers, we decided to not concentrate on perfecting the tree layout but instead focus on generating interactivity in the visualization. Thus, we decided to keep the parent node of each child that shared the most particles with the child and stick with the tree layout.

With the use of online examples and D3's zoom interface, we easily added the ability to collapse nodes and zoom and drag the tree. The panning and zooming is bounded so that users are not allowed to zoom too far out or in to completely lose the tree from view nor allowed to drag the tree off the screen.

Node and Edges Features

The nodes represent halos at each time step, and each node can be collapsed by a single click. By holding down the mouse and moving the node, the entire tree is dragged and the node does not collapse. The scroll wheel on the mouse is used to zoom. When collapsing a node, an arrow is displayed indicating that there exists hidden children nodes. The node radius was sized according to mass of the halo. Tooltips were added to include the ability to see more information for each node upon hover. This extra information includes the halo group identifier, mass, total number of particles, total number of dark particles, and total luminosity value. Since some of the nodes are quite small, we needed to make sure users could easily click to collapse and hover to see the tooltip. Thus, we created an invisible circle attached to each node with a radius either equal to the node's radius (for larger nodes) or equal to a fixed radius (for smaller nodes) that contain the mouse listeners. With this, even the smallest nodes are more accessible.

Additionally, since the tree can be zoomed and dragged around, the tooltip interactions were modified so the information box was not placed off screen and did not show while a user was issuing a drag or zoom command. Since we used radius size to represent mass, we used color to represent progenitor halos. Therefore, a red node represents a progenitor

while steel blue represents a non-progenitor. Progenitor halos are defined as those halos that contribute the greatest number of particles to the present day halo for each time step. Edges between the nodes represent the flow of particles between the halos in sequential time steps. The thickness of the edge is determined by the number of particles that are shared between the two connected halos.

Graph Filters

Two function graphs were added to include an option of highlighting specific halos. The graphs display the frequency of the number of halos that have a specific mass (left graph) or a specific number of particles (right graph) as an area where the total area represents the total mass or total number of particles in the current tree throughout time. In order to properly see the area for each graph, a log scale for the mass and particle count had to be used for the x-axis. The y-axis is linear based on the frequency. The graph was created by binning the mass and particle count of each halo. For these graphs, there were 20 bins created. A brush tool was added to each graph so the user can highlight an interval of the mass (or particle count) she is interested in. Upon brushing the graphs, the nodes that fit the criteria are highlighted. The user also can manually enter new brush ranges and hit the "Update Interval" button to alter the brush accordingly (to avoid confusing users, pressing the Enter key has no effect). Initially the nodes were highlighted by adding a fill color to the inside of the nodes selected. Unfortunately, it was hard to see which of the small nodes where highlighted so we added another outer circle to each node to show the highlight effect. Each of these outer circles include a blur effect to emphasize that those specific nodes are being highlighted.

Tree Selection

The scroll bar at the top of the visualization is used for navigating between trees and potentially finding other interesting trees. If there is a particular tree an astronomer wants to jump to, the associated group number of the root node can be directly typed into the text box to the left, and upon pressing the Enter key, that tree will be displayed if it is an existing id. Since a main component of the visualization is exploration, the thumbnails were created to facilitate quick structural comparisons between the current tree and other trees. We highlighted the path of the progenitor in these thumbnails since that is the halo path of interest to the astronomers. We kept the thumbnail of the current tree fixed to the left to allow for quick visual comparison between thumbnails instead of trying to compare the thumbnail to the large tree diagram. These thumbnails become highlighted upon hovering over them, indicating if they are clicked, that tree will be displayed below.

The thumbnails were created by using our D3 tree generation code in tandem with Node.js, a software platform for server-side applications where programs are written in JavaScript and run within the Node.js runtime, and ImageMagick, a software suite to manipulate images. Using Node.js, we ran D3 on our local machine to generate the SVG of each thumbnail tree. The SVG was then piped to ImageMagick, which converted the SVG to PNG images.

Tree Similarity

Given that there are a 365 trees, displaying all the trees in the scroll bar would be overwhelming and could potentially slow the visualization. Choosing which trees to display then became a challenge. Since the astronomers are interested in the merge patterns that exist between trees, we used the notion of bisimilarity to implement a comparison metric based on tree structure [1]. For two trees T_1 and T_2, we say the children of node n at time step t are those nodes at time step $t+1$ with edges from n. We define the binary relation $R \subseteq T_1 \times T_2$ to be a bisimulation when for all $(n, v) \in R$ where $n \in T_1$ and $v \in T_2$,

- n and v occur in the same time step
- n and v have the same number of children
- For all children n' of n, there exist a child v' of v such that $(n', v') \in R$
- For all children v' of v, there exist a child n' of n such that $(n', v') \in R$

If $(n, v) \in R$, we say n is bisimilar to v.

Intuitively, two nodes in T_1 and T_2 are bisimilar if they occur at the same time step, have the same number of children, and their children are bisimilar. In terms of the merger trees, this bisimilarity means that the merges of bisimilar nodes involve the same number of halos and occur at the same time.

We calculated the bisimulation offline by first letting all nodes at the same time step with the same number of children be bisimilar to each other and then iteratively checking the conditions of all the descendants. Since the trees had a maximum height of 27, we iterated 27 times.

Finally, for each pair of trees T_1 and T_2 and the bisimulation $R \subseteq T_1 \times T_2$, we define the bisimilarity distance to be

$$Bdist(T_1, T_2) = \frac{|\{n \in T_1 : (n, v) \in R\}| + |\{v \in T_2 : (n, v) \in R\}|}{|T_1| + |T_2|}.$$

For the top scroll bar, we display the closest 14 trees as measured by the bisimilarity distance. We chose 14 somewhat arbitrarily to give users enough trees to stay engaged but not overwhelmed. This number should be evaluated as part of future work.

Tree Reset and Transitioning

The two instances when the view of the tree will change without a user scrolling or dragging are when the user clicks the reset view button or selects to change the tree by either entering a new group number or clicking on one of the thumbnails. The reset view reorients a user if she "got lost" in the dragging and zoom of the tree. We hoped to prevent that by bounding the drag and zoom but still added this feature. The reset view acts like a page refresh on the tree, except we staged it so the user can understand the visual transitions. We first zoom out and recenter the tree in one transition and then expand the children in reversed collapsed order. In other words, if a node's grandchildren are collapsed and then the children are collapsed, the children will expand and then the grandchildren will expand. Figure 3 shows a node whose children and

Figure 3. Example of children expansion upon clicking the reset view button. The left image is before resetting the view, the middle is after one transition, and the right after a second and final transition.

grandchildren have been collapsed going through the expansion transition. The left image is before the user clicks the reset view button. The middle image is after the first expansion transition, and the right image is after the second and in this case last transition. We staged it this way to better help the user understand what was undone to reset the view.

When a new tree is selected, the same general transitioning procedure as for the reset view button is followed, except we do not iteratively expand the children. We expand all the children and grandchildren in one step, and then we collapse the tree to the root and expand to the new tree. We removed the iterative expansion process since the user will be looking at a new tree and does not need to understand exactly how the children of the former tree were expanded. We also did not want to frustrate the user by having to wait longer than necessary to view a new tree.

Extra Toggle Features

Extra toggle features include the ability to remove the graphs from view, remove the tooltips, and view the luminosity of the tree. We added the options to remove the tooltips and graphs in order to avoid distractions or nuisances during the exploration process. The luminosity is based on aggregating the luminosity of each particle in a halo. Once the user selects the luminosity view, they will see the nodes change opacity depending on the halo's luminance. An example of the luminance feature is seen in Figure 4. The luminance observed is not only relative to the current tree viewed but also relative to all the trees across all time steps. We did not include luminosity as part of the standard view to avoid too much complexity and indiscernible features.

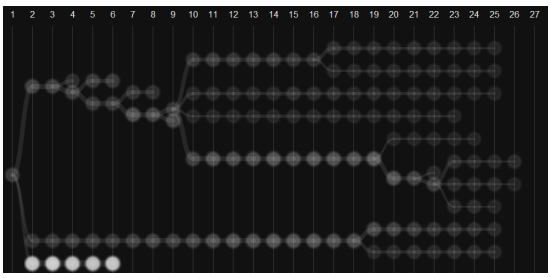

Figure 4. Luminosity Feature

RESULTS

See Figure 5 for a screen shot of the final visualization. This visualization is intended for use on a screen at least 22 inches across with a standard 16:10 aspect ratio. However, it is also usable on standard monitors such as the one on a MacBook Pro. Upon running this visualization, it took anywhere from 1 to 3 seconds to load with the majority of the time being spent loading the D3 javascript file from `d3js.org` and loading the links csv file. Once the merger tree is displayed, the particle count and mass graphs are immediately updated based on the tree shown. Navigating between trees results in smooth transitions taking about one second to complete. Brushing and highlighting features is without delay, even for busy merger histories.

DISCUSSION

For this visualization, the audience has consisted of members from the database group and also astronomers from UW. Upon showing this visualization to the database group, they have decided to include this visualization on top of their Myria system. Over the past year, the database group has developed a new engine for Big Data. The vision behind this system is to meet the needs of today's users, particularly those in the domain sciences. The system will go public and along with it, applications built on top of the system that will help others like astronomers and oceanographers explore their data in an efficient and feasible manner. Although the current data for our visualization is all pre-computed, the goal will be to incorporate this visualization on top of the Myria system so that astronomers be able to easily query merger histories on-the-fly for larger simulations.

Throughout the process of creating this tool, we've been in contact with the astronomers Lauren Anderson and Sarah Loebman nearly every week. When we initially talked with them, they expressed their need for a tool to help them figure out where the exploration should happen because they were unsure exactly what types of halos they were interested in. Thus, this visualization is intended to help them narrow down specific merger histories they may like to do an in-depth analysis on. They also look forward to using this tool with newer and larger simulation datasets. Currently, this visualization is based on a low resolution dataset, and a new dataset they want to visualize has a resolution high enough to resolve the morphologies of low mass galaxies. We hope that they can begin to use this tool to explore all sizes of data and pinpoint important merger tree characteristics and relevant halos.

It is interesting to note the work behind finding the history of present day halos began back in 2009. This is the first time the astronomy department will be able to see and interact with a visualization of this merger tree work. Upon showing Sarah and Lauren this recent work, they pointed out interesting observations. For example, by navigating through the trees using the scroll bar based on the similarity metric, they noticed that if you are viewing an active merger tree, it is easier to find other active merger trees while if you are viewing a quiescent merger tree, it is likely you will find other quiescent merger trees. Quiescent mergers are merger histories where not much merging occurs (most parents have one child). They liked this feature of finding similarly busy merger histories.

At this recent meeting, after explaining all of the features, they were very excited and kept giving us information they wanted to add and features they thought would be helpful improvements. These additional features indicated their interest and belief that this visualization will be beneficial and relevant in the future. They were mostly just excited to finally have a tool at their disposal that allowed them to look at these

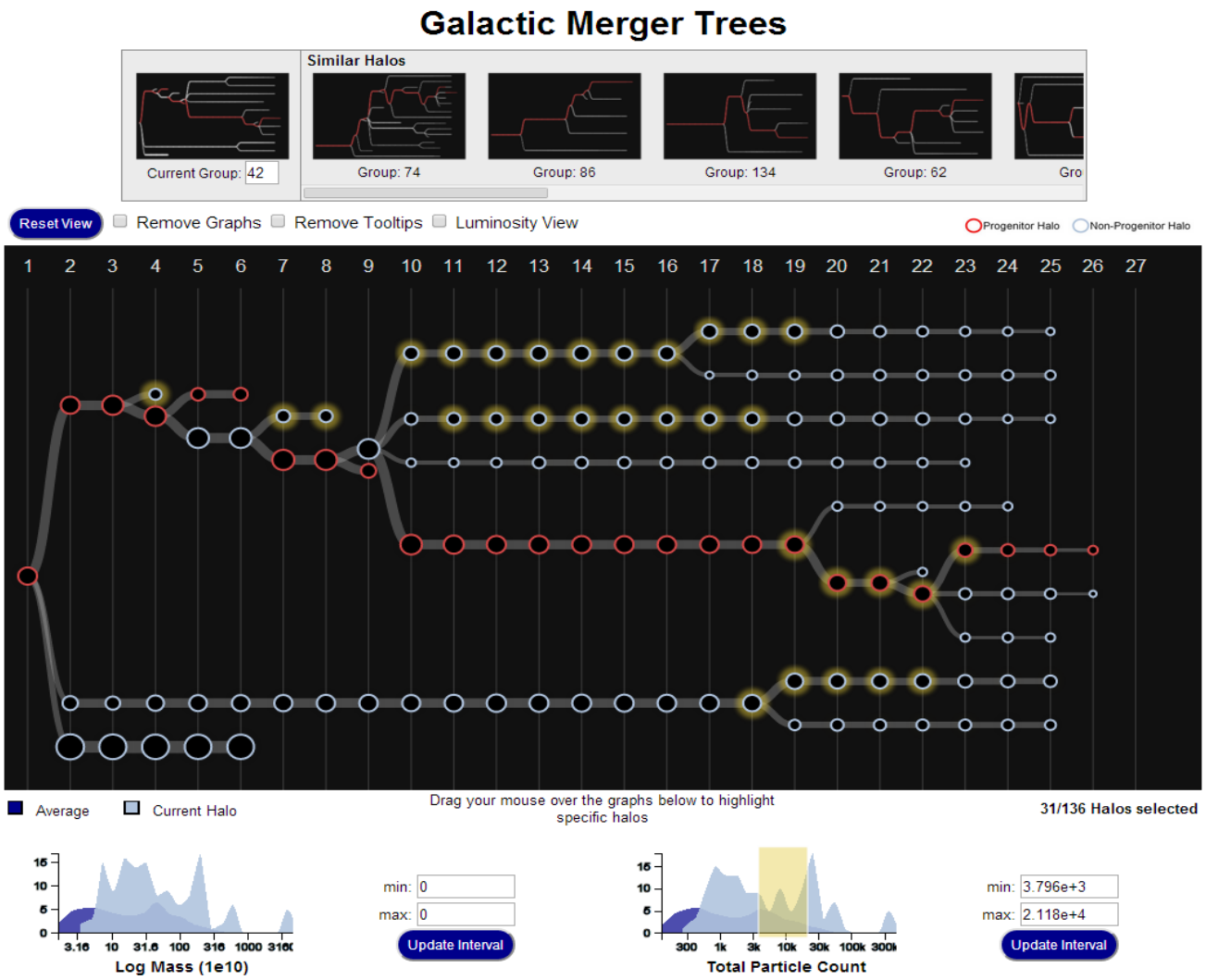

Figure 5. Visualization of a Galactic Merger Tree

structures. As a result of this meeting, we will demo this visualization at an astronomy research meeting in the next couple weeks.

FUTURE WORK

There is much potential for this work to be extended in the future. Some of the possible extensions include finding a way to extract specific information about the halos shown on the screen. For example, when using the brush tool, Sarah asked if there was a way to "save" the halos she was highlighting to a file. Also, there is more room for improvement to facilitate data analysis. For example, Lauren is interested in finding how temperature changes as a function with respect to the number of gas particles. One way to add this feature would be to include the ability select a path of the tree. We can then tell users about how certain properties such as luminosity or temperature has changed across time along that path. An ambitious addition would be to provide the user with the ability to define their own function with respect to the properties defined in the tree.

Another major feature to add that is critical to the astronomers involves finding a way to show the splits that occur for some halos. As explained previously, this would involve finding a different way to layout the nodes besides using the D3 tree layout. Likely, we would use GraphViz to help us find initial positions and then alter the tree layout in D3 to create transitions and include collapsing to make it work with a DAG graph.

REFERENCES

1. Alzogbi, A., and Lausen, G. Similar structures inside rdf-graphs. In *LDOW*, CEUR Workshop Proceedings, CEUR-WS.org (2013).

2. Loebman, S., Nunley, D., Kwon, Y., Howe, B., Balazinska, M., and Gardner, J. P. Analyzing massive astrophysical datasets: Can pig/hadoop or a relational dbms help?

3. Maddison, D. R., Schulz, K.-S., and Maddison, W. P. The tree of life web project. *Zootaxa 1668*, Linnaeus

Tercentenary: Progress in Invertebrate Taxonomy (2007), 19–40.

4. Moran, C. C. Astroviz: A parallel visualization tool for astrophysical applications, 2009.
5. Munzner, T., Guimbretière, F., Tasiran, S., Zhang, L., and Zhou, Y. Treejuxtaposer: Scalable tree comparison using focus+context with guaranteed visibility. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, ACM (New York, NY, USA, 2003), 453–462.
6. Takle, J., Silver, D., and Heitmann, K. A case study: Tracking and visualizing the evolution of dark matter halos and groups of satellite halos in cosmology simulations. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on* (Oct 2012), 243–244.
7. Turk, M. J., Smith, B. D., Oishi, J. S., Skory, S., Skillman, S. W., Abel, T., and Norman, M. L. yt: A multi-code analysis toolkit for astrophysical simulation data.