The Barnes-Hut Approximation: Efficient computation of N-body forces

Jeffrey Heer

March 31, 2023

Introduction

Computers are exciting tools for discovery, with which we can model and explore complex phenomena. For example, to test theories about the formation of the universe, we can perform *simulations* to predict how galaxies evolve. We could gather the estimated mass and location of stars and model their gravitational interactions over time.

Another avenue for discovery is to *visualize* complex information to reveal structure and patterns. Consider this network diagram, showing connections between people in a social network. We can examine community groups and identify people who bridge between them.

Though they may seem quite different at first, these two examples share a common need. They both require computing forces that arise from pairwise interactions among a set of points, often referred to as an *N-body problem.* In the case of astronomical simulation, we seek to model the gravitational forces among stars. In the case of network visualization, we compute the layout using a similar physical simulation: nodes in the network act as charged particles that repel each other, links act as springs that pull related nodes together.

To get a sense of how this force-directed layout works, drag the nodes or use the slider to adjust the force strength. Negative values indicate repulsive forces, while positive values indicate attractive forces.

Force Strength

-30

A straightforward approach to computing N-body forces is to consider all pairs of individual points and add up the contributions of each interaction. This naïve scheme has *quadratic complexity*: as the number of points n increases, the running time grows proportionally to n^2, quickly leading to intractably long calculations. How might we do better?

The Barnes-Hut Approximation

To accelerate computation and make large-scale simulations possible, the astronomers Josh Barnes and Piet Hut devised a clever scheme: approximate long-range forces by replacing a group of distant points with their center of mass. In exchange for a small amount of error, this scheme significantly speeds up calculation, with complexity $n \log n$ rather than n^2.

Central to this approximation is a *spatial index*: a "map" of space that helps us model groups of points as a single center of mass. In two dimensions, we use a quadtree data structure, which subdivides square regions of space into four equal-sized quadrants. (In three dimensions, an octree divides a cubic volume into eight sub-cubes.)

The Barnes-Hut approximation involves three steps:

1. Construct the spatial index (quadtree)
2. Calculate centers of mass
3. Estimate forces

Let's explore each step in turn. We will assume we are computing *repulsive* forces for the purposes of network layout. This setup is akin to modeling anti-gravity or electric forces with similarly-charged particles. While we will use the term "center of mass", this could readily be replaced with "center of charge".

As you read through, click the action links to update the diagram!

Step 1: Construct the Quadtree

We begin with a set of two-dimensional input points. When we insert the first point into the quadtree, it is added to the top-level root cell of the tree.

When we insert another point, the tree expands by subdiving the space. With each subsequent insertion, more fine-grained cells are added until all points reside in their own cell.

Advance the slider to add each point and produce the full quadtree.

Inserted Points

0

Step 2: Calculate Centers of Mass

After quadtree construction, we calculate centers of mass for each cell of the tree. The center of mass of a quadtree cell is the weighted average of the centers of its four child cells.

We visit the leaf node cells first and then visit subsequent parent cells, merging data as we pass upwards through the tree. Once the traversal completes, each cell has been updated with the position and strength of its center of mass.

Step 3: Estimate N-Body Forces

Now we are ready to estimate forces!

To measure forces at a given point, let's add a "probe" to our diagram. The purple line extending from the probe indicates the direction and magnitude of the total force at that location. (To promote visibility, the purple line is three times longer than the actual pixel distance the probe would be moved in a single timestep of the force simulation.) The dotted lines extending to the probe represent the force components exerted by individual points.

Move the probe (click or drag) to explore the force field.

Ignoring the quadtree, we can naïvely calculate forces by summing the contributions of *all* individual points. Of course, we want to use the quadtree to accelerate calculation and approximate long-range forces. Rather than compute interactions among individual points, we can compute interactions with centers of mass, using smaller quadtree cells for nearer points and larger cells for more distant points.

At this point we skipped a critical detail: what constitutes "long-range" versus "short-range" forces? We consider both the *distance* to the center of a quadtree cell and that cell's *width*. If the ratio *width / distance* falls below a chosen threshold – a parameter Θ (*theta*) – we treat the quadtree cell as a source of long-range forces and use its center of mass. Otherwise, we will recursively visit the child cells in the quadtree.

When $\Theta = 1$, a quadtree cell's center of mass will be used – and its internal points ignored – if the distance from the sample point to the cell's center is greater than or equal to the cell's width.

Adjust the Θ parameter to view its effect on force estimation.

How does the number of considered points change based on the probe location and Θ? How does the direction and magnitude of the total force vary with Θ?

Theta Θ

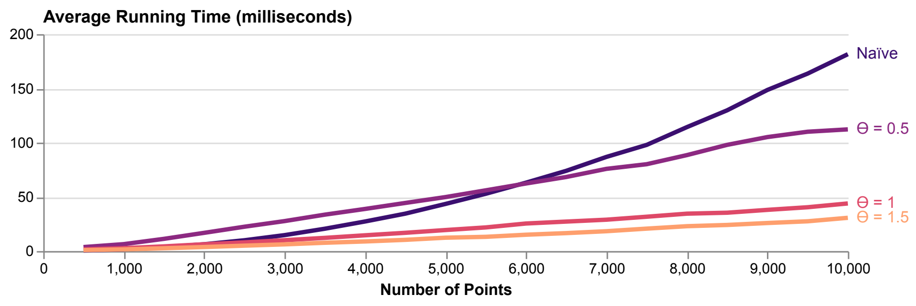
0

We can now perform force estimation for each individual point, using the Barnes-Hut approximation to limit the total number of comparisons!

Performance Analysis

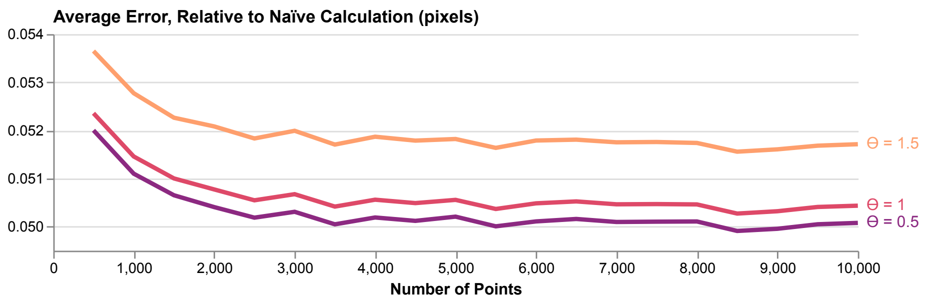
To assess the performance of the Barnes-Hut approximation, we can look at both the running time and accuracy of force estimation. We will compare naïve (n^2) calculation to different settings of the Θ parameter.

We will take measurements using different point sets, ranging from 500 to 10,000 points. For each point count, we average the results from 50 separate runs of force estimation, each time using a different set of points placed at uniformly random coordinates within a 900 x 500 pixel rectangle.



The running time results confirm that the Barnes-Hut approximation can significantly speed-up computation. As expected, the naïve approach exhibits a quadratic relationship, whereas increasing the Θ parameter leads to faster calculations. A low setting of $\Theta = 0.5$ does not fare better than the naïve approach until processing about 6,000 points. Until that point, the overhead of quadtree construction and center of mass calculation outstrips any gains in force estimation. In contrast, for $\Theta = 1$ and $\Theta = 1.5$ we see significant improvements in running time.

To evaluate approximation error, we measure the average vector distance between the results of the naïve scheme and Barnes-Hut. In the context of a force-directed graph layout, this error represents the difference (in pixels) between node positions after applying the naïve and approximate methods.



Looking at the error results, we first see that the average error is relatively small: only ~5% of a single pixel in difference! However, we should take care interpreting these results, as we use the *average* error per point and the *maximum* error may be substantially higher. While $\Theta = 1$ and $\Theta = 1.5$ exhibit similar *running times*, here we see notably higher *error rates* for $\Theta = 1.5$ versus $\Theta = 1$ and $\Theta = 0.5$.

These results suggest that a good default value for Θ – with low running time *and* low approximation error – is around 1.0. Indeed, in practice it is common to see default settings slightly below 1. In visualization applications, where errors on the order of a few pixels are not a problem, even higher Θ values may be used without issue.

Conclusion

The Barnes-Hut approximation has had a major impact on both physical simulation and network visualization, enabling n-body calculations to scale to much larger data sets than naïve force calculation permits.

Returning to our initial network diagram, we can use Barnes-Hut to efficiently compute repulsive forces at each timestep. For each animation frame, we perform the approximation anew, creating a new quadtree, accumulating centers of mass, and (approximately) estimating forces.