The Barnes-Hut-Algorithm

Thomas Trost

Institute for Theoretical Physics I
Ruhr-University Bochum

January 12th 2016

Table of Contents

1 Introduction and Motivation

2 The Barnes-Hut-Algorithm

3 Pros and Cons, Alternatives

Introduction and Motivation

N-Body Systems

System of N particles:

- i-th particle characterized by its position $\mathbf{r}_i$, velocity $\mathbf{v}_i$, mass m_i (and charge q_i, ...)
- interaction via gravitational force $F_i = \sum_{i \neq j} G m_i m_j \frac{\mathbf{r}_j - \mathbf{r}_i}{|\mathbf{r}_j - \mathbf{r}_i|^3}$ (or Coulomb-force, ...)
- can be studied in 1, 2, 3 or even more spatial dimensions

Model for:

- astronomical objects, systems of galaxies
- plasmas
- ...

Solution of N-Body Problems

Analytical solution for

- 2-body systems
- certain cases of 3-body systems

$\Rightarrow$ In general, numerical treatment is necessary.

Naive approach: Compute force as the sum shown above and plug it into integrator for ODEs.

Problem with Naive Approach

- force acts over a long range
- force on a particular particle depends on all other particles
- problems with high numbers of particles ($N > 10^6$) are relevant and interesting

$\Rightarrow$ Naive approach does not scale well with number of particles ($\sim \mathcal{O}(N^2)$) and does thus not allow for treatment of realistic problems.

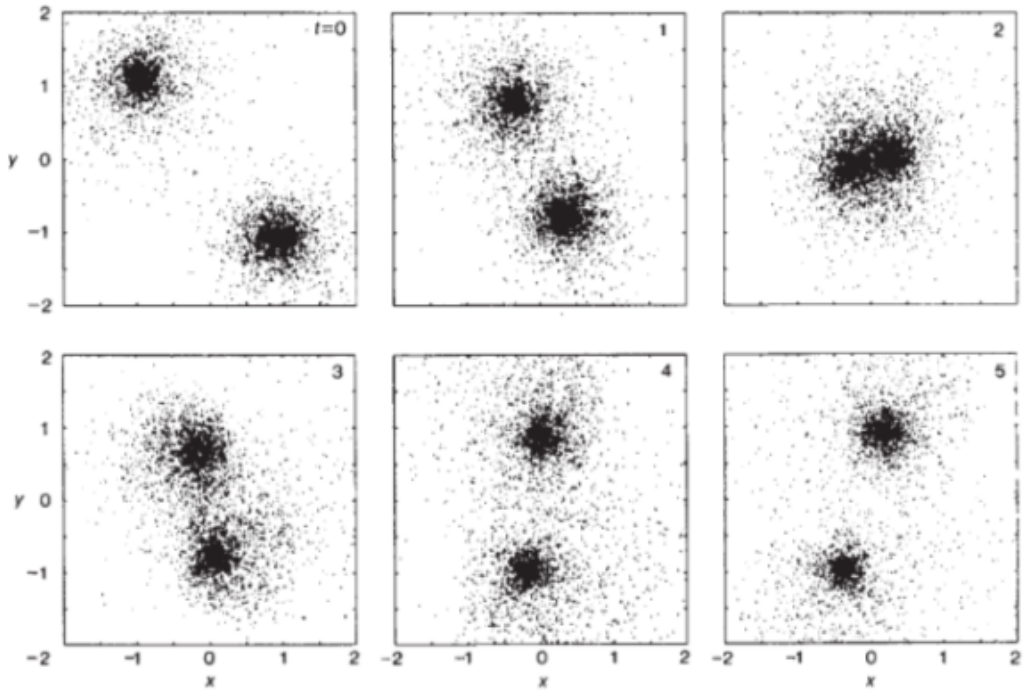
History I

Algorithm developed in 1986.

Joshua Barnes
Institute for Astronomy (IFA)
University of Hawaii

Piet Hut
Institute for Advanced Study
Princeton

History II



$N = 4096$

runtime $\approx 10\,\mathrm{h}$

VAX 11/780

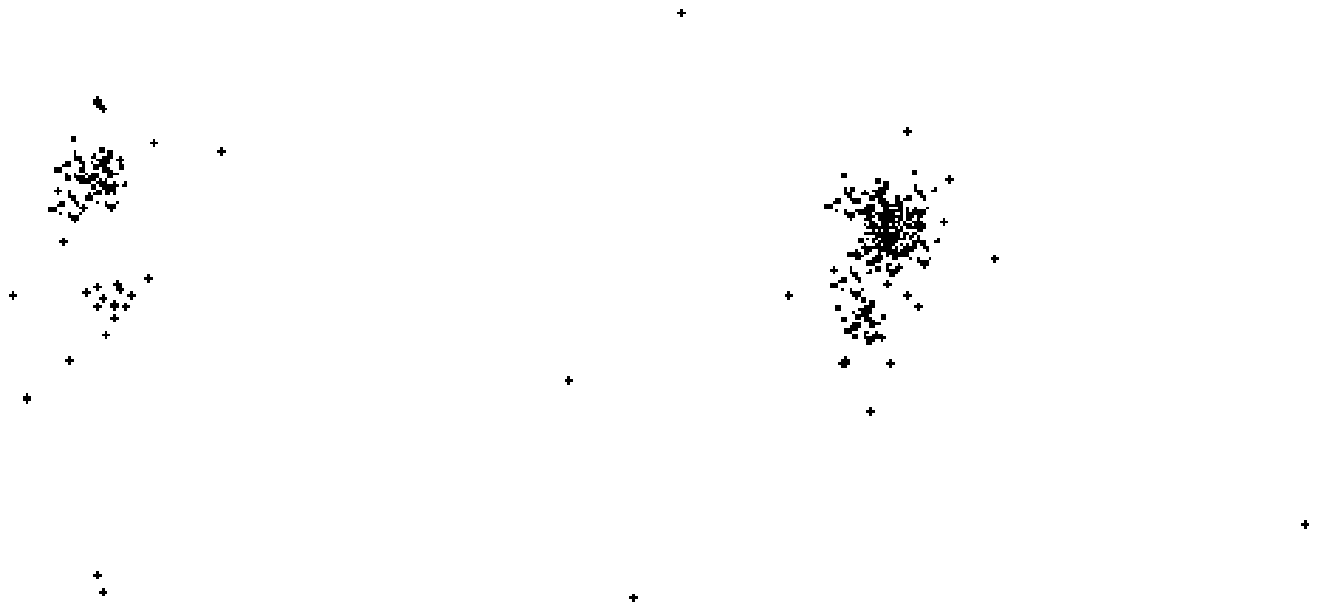
5 MHz

$\leq 8\,\mathrm{MB}$ RAM

The Barnes-Hut-Algorithm

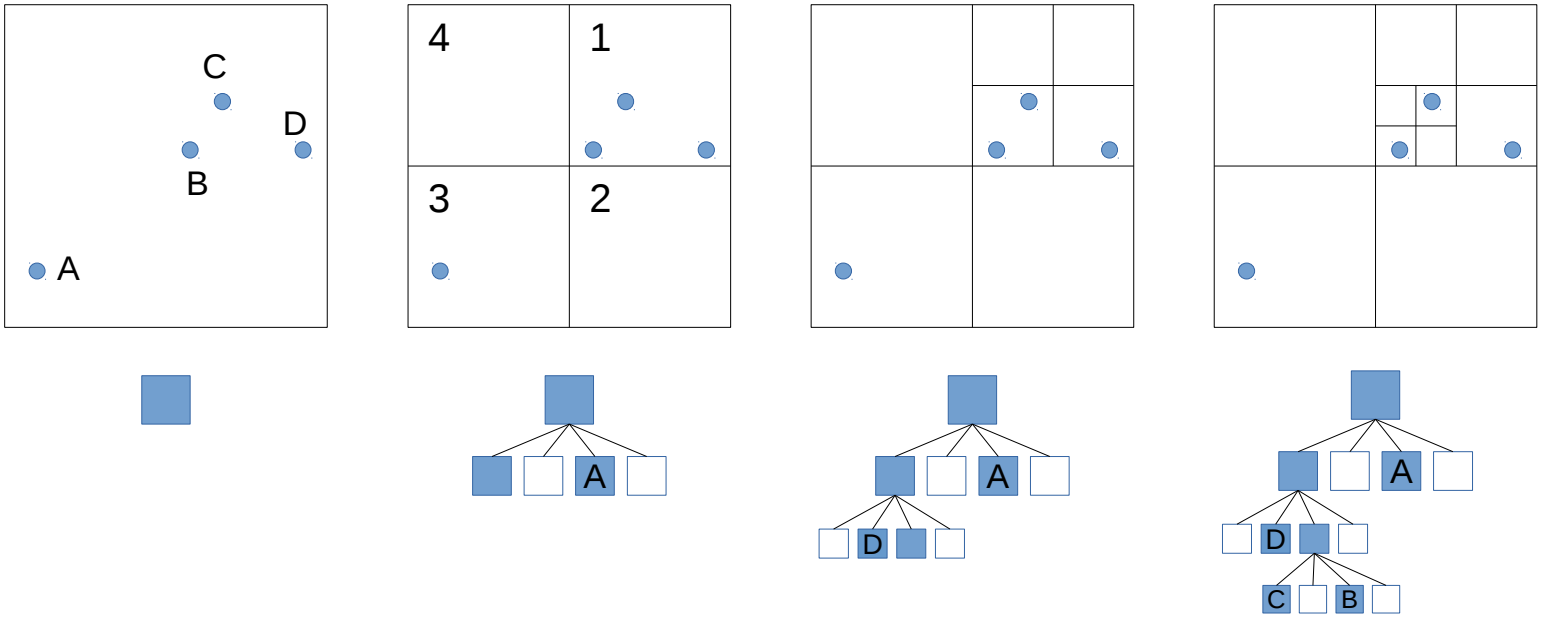
Idea: Boil down calculation of force

Treat "lumps" of particles like one big macro particle if distance is large enough, in order to reduce the number of direct interactions.



Tree structure

Use tree structure (binary tree, quad-tree or octree, depending on dimensionality) to give notion of "lumps" a well-defined meaning.

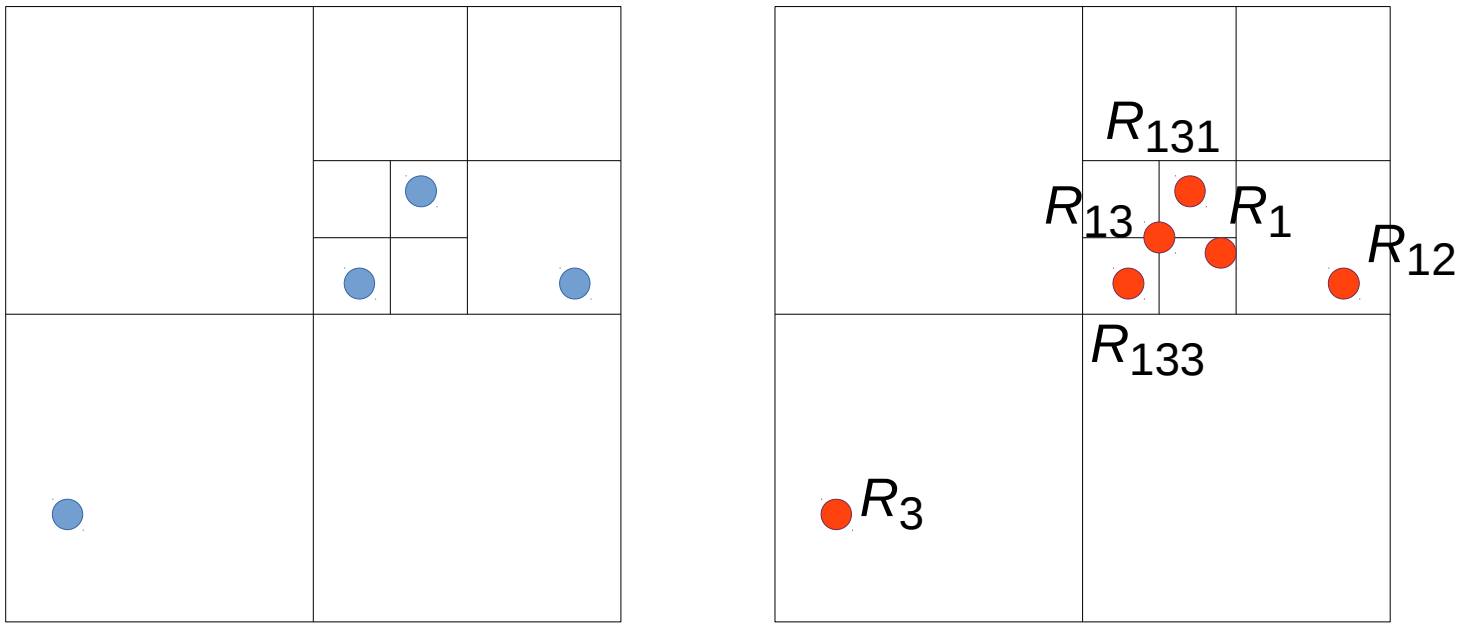


In the tree, close particles are grouped under the same nodes.

Nodes as Macro-Particles

Each node is assigned a mass and a position, on the basis of the particles it and its subnodes contain:

- total mass: $M = \sum_i m_i$
- center of mass: $\mathbf{R} = \left(\sum_i \mathbf{r}_i m_i\right) / M$



Building the Tree

In the program, the following recursive procedure is used for building the tree (pseudocode):

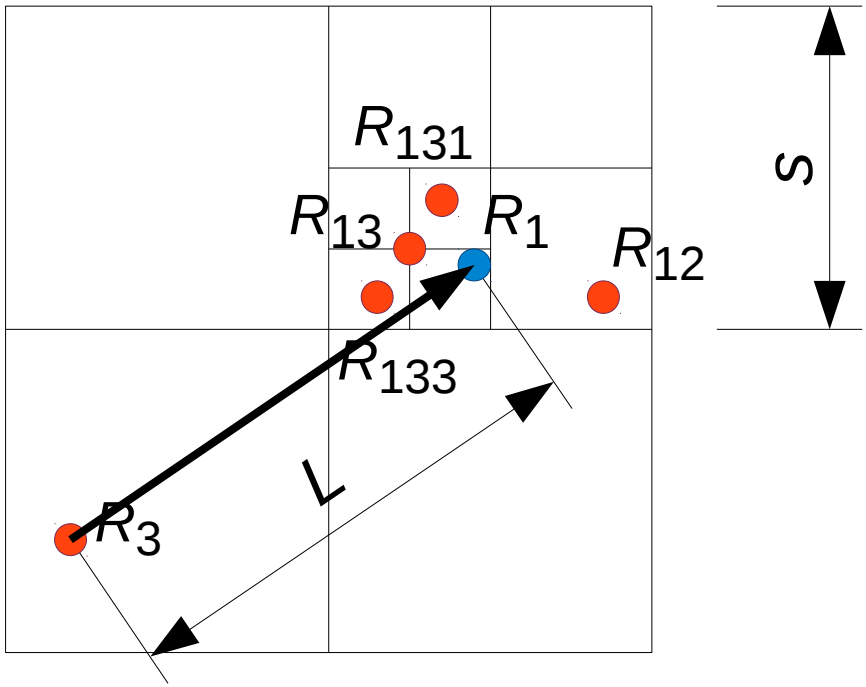
```
function insert(node, particle)
    if(has_children(node))
        quadrant <- get_quadrant(node, particle)
        insert(child(node, quadrant), particle)
    else if(is_empty(node))
        add_particle(node)
    else
        create_children(node)
        quadrant <- get_quadrant(node, particle)
        insert(child(node, quadrant), particle)
        node_particle <- get_particle(node)
        quadrant <- get_quadrant(node, node_particle)
        insert(child(node, quadrant), node_particle)
        remove_particle_from_node(node)
```

Calculating the Force

Compare width s of node with distance L, threshold for approximation is

$$\frac{s}{L} < \Theta$$

(typically, $\Theta = 0.5$; $\Theta = 0$ corresponds to naive approach). If ratio is larger, go deeper into the tree.



Implementation of Force calculation

```
function get_force(node, particle)
    if(width(node)/distance(node, particle) < theta)
        return calculate_force(node, particle)
    else if(has_children(node))
        force <- 0
        for each child in get_children(node)
            force <- force + get_force(child, particle)
        return force
    else if(is_empty(node))
        return 0
    else
        node_particle <- get_particle(node)
        return calculate_force(node_particle, particle)
```

Order of the Algorithm

- depth of tree scales roughly with $\log(N)$
- costs for building tree scale like product of N with average depth of tree
- costs for calculating the force scale like product of N with average depth of tree

$\Rightarrow$ On average we have $\mathcal{O}(N \log N)$, which is much better than $\mathcal{O}(N^2)$.

Problem: Depends very much on the distribution of particles, Θ, etc.

Adjustments

What can be varied?

- Θ
- maximum number of particles per node
- maximum depth of tree
- integrator for ODE
- way of building the tree might be made more efficient
- ...

Pros and Cons, Alternatives

Pros and Cons

Pros:

- makes large N feasible
- possible to use parallelisation techniques
- easy to understand and implement
- does not depend on a certain kind of interaction

Cons:

- hard to control what actually happens
- efficiency depends on the situation, e.g. high densities lead to poor performance due to deep tree
- significant overhead if applied in wrong situation
- energy and momentum not conserved
- does not take into account higher moments

Alternatives

If N is low, the naive brute force algorithm performs better due to the lack of overhead.

For large N, especially with high densities, mesh-based methods are an alternative to tree based methods.