FAST MULTIPOLE METHOD (FMM)

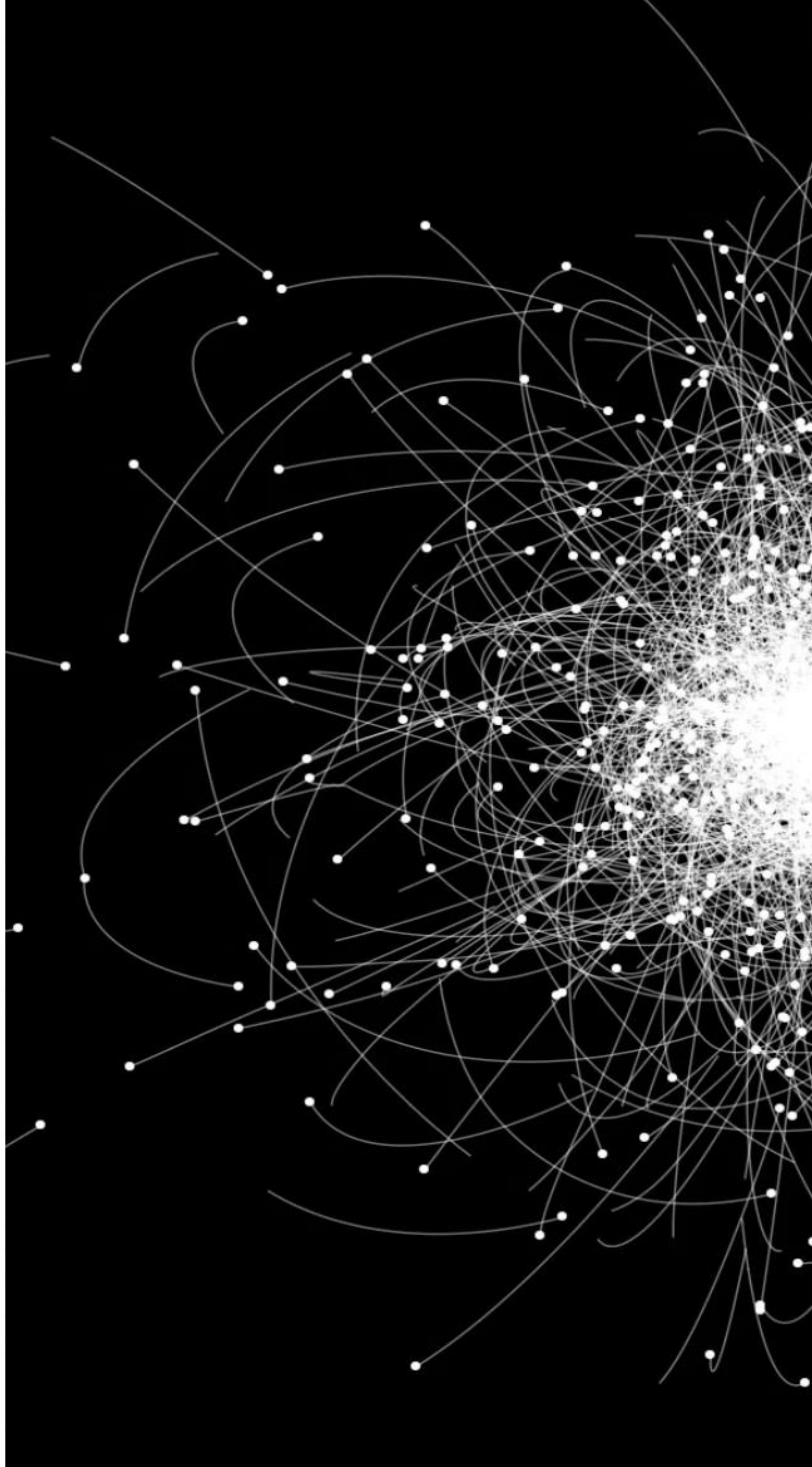
FOR N-BODY SIMULATION

CONTENT

PROBLEM STATEMENT

Simulating N-body systems involves evaluating pairwise interactions among particles governed by physical laws such as gravity or Coulomb forces. Each particle exerts a force on all others, resulting in the following governing equation for particle i :

$$\mathbf{F} = G \sum_{i \neq j} m_i m_j \frac{\mathbf{r}_j - \mathbf{r}_i}{|\mathbf{r}_j - \mathbf{r}_i|^3}$$



NAIVE ALGORITHM

ALGORITHM

Algorithm Naive $O(N^2)$ Approach for 2D Force Calculations

```
1: Input:
    • $N$: Number of particles.
    • $\{x_i, y_i\}_{i=1}^N$: Positions of $N$ particles in 2D space.
    • $\{m_i\}_{i=1}^N$: Charges or masses of the particles.
    • $\epsilon$: Softening factor (optional, to avoid singularities).
Output: $\{F_x, F_y\}_{i=1}^N$: Forces acting on each particle.
Initialize $F_x[i] = 0$ and $F_y[i] = 0$ for all $i = 1, \ldots, N$.
for each particle $i = 1$ to $N$ do
  for each particle $j = 1$ to $N$ do
    if $i \neq j$ then
      Compute distance components:
          $dx = x[j] - x[i], \quad dy = y[j] - y[i]$
      Compute squared distance:
          $r^2 = dx^2 + dy^2 + \epsilon^2$
      Compute inverse distance cubed:
          $r_{\text{inv3}} = \frac{1}{r^3}$
      Compute force components:
          $F_x[i] \mathrel{+}= m[i] \cdot m[j] \cdot dx \cdot r_{\text{inv3}}$
          $F_y[i] \mathrel{+}= m[i] \cdot m[j] \cdot dy \cdot r_{\text{inv3}}$
    end if
  end for
end for
Return: $\{F_x, F_y\}_{i=1}^N$
```

The time complexity of the naive approach can be expressed as follows:

$$T(N) = \sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{O}(1)$$

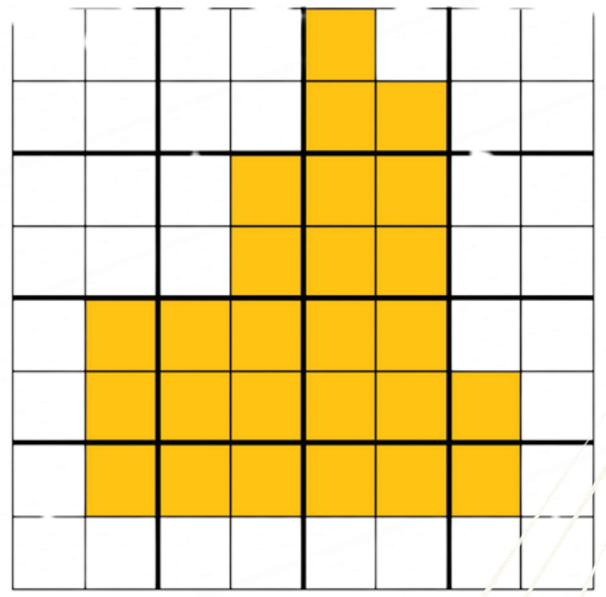
Since the algorithm involves a double loop over N particles, where each pair-wise interaction is computed in constant time $\mathcal{O}(1)$, the total time complexity simplifies to:

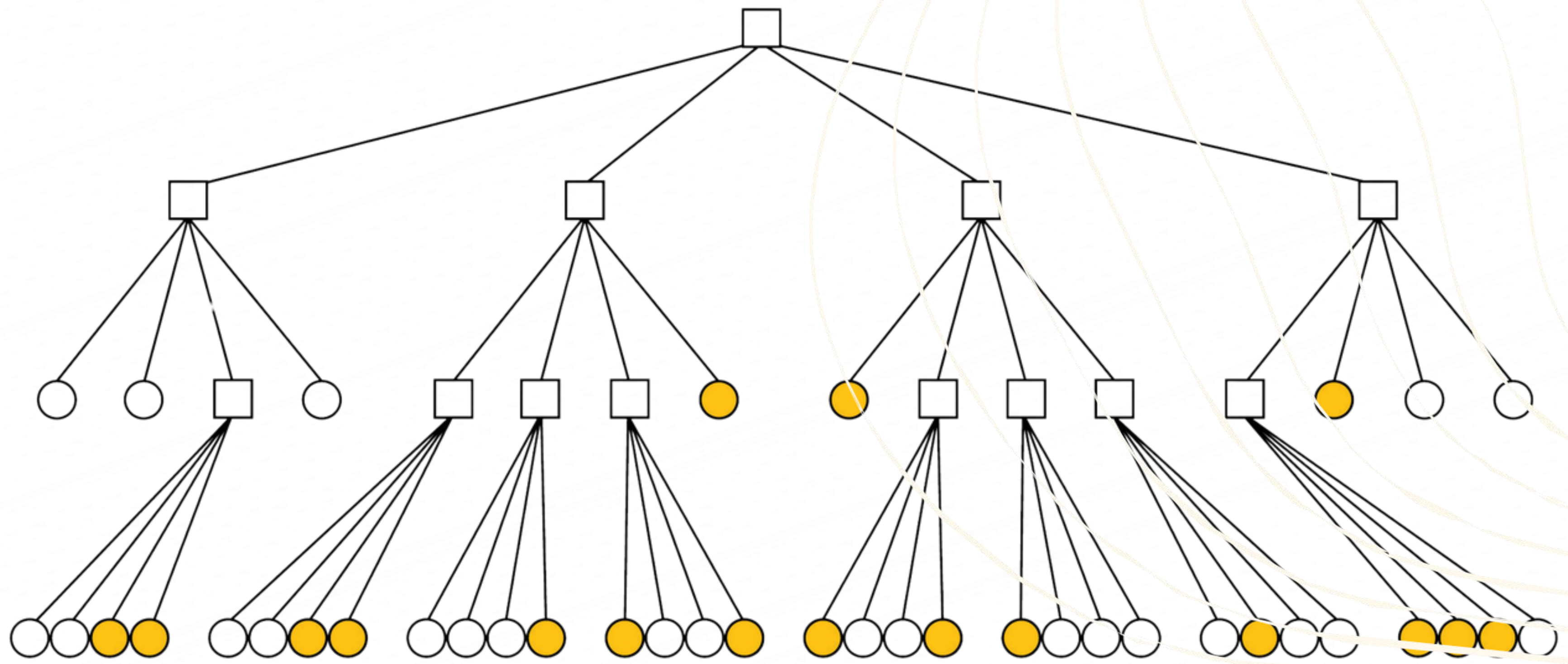
$$T(N) = N \cdot N \cdot \mathcal{O}(1) = \mathcal{O}(N^2)$$

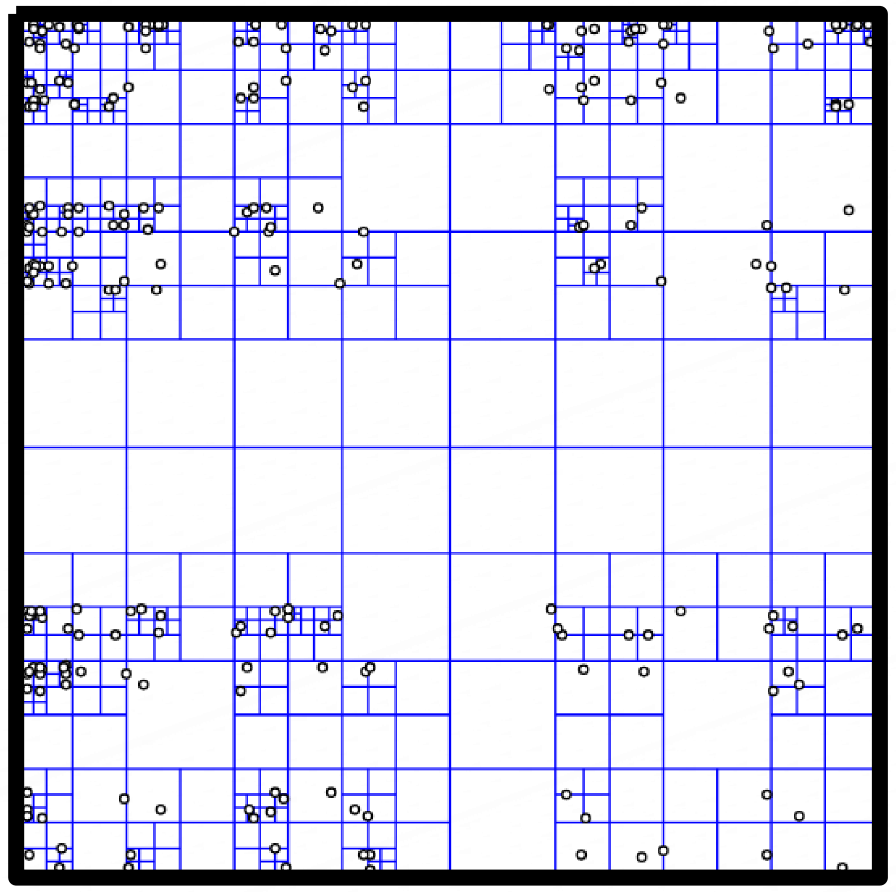
Thus, the naive approach has a time complexity of $\mathcal{O}(N^2)$.

QUAD-TREE

A quad-tree is a hierarchical data structure used to efficiently partition and organize two-dimensional spatial data. It recursively divides a 2D space into four quadrants or regions, starting with a root node that represents the entire space. Subdivision continues until each region satisfies a specific condition, such as containing a maximum number of points or reaching a predefined depth. Quad-trees are widely used in computer graphics, spatial indexing, image processing, and geographic information systems (GIS) for tasks like collision detection, range queries, and efficient storage of spatial data.







ALGORITHM AND COMPLEXITY

Algorithm Quad Tree Construction

```
procedure BuildQuadTree($particles, region, maxDepth, threshold$)
    Create root node covering the entire region
    Initialize root node with all particles
    Subdivide($root, maxDepth, threshold$)
end procedure
procedure Subdivide($node, maxDepth, threshold$)
    if $node.depth \geq maxDepth$ or $|node.particles| \leq threshold$ then
        return
    end if
    Divide $node.region$ into 4 quadrants
    for each quadrant $q$ do
        Create child node $child$ for $q$
        Assign particles from $node.particles$ to $child$ if they lie in $q$
        Add $child$ to $node.children$
        Subdivide($child, maxDepth, threshold$)
    end for
end procedure
procedure ComputeInteractions($node$)
    if $node$ is a leaf then
        Compute direct interactions for particles in $node$
    else
        for each $child \in node.children$ do
            ComputeInteractions($child$)
        end for
        Compute multipole expansion for $node$
    end if
end procedure
```

Operation	**Time Complexity**
Quad-tree Construction	$O(n \log n)$
Quad-tree Traversal	$O(n)$
Direct Interaction Computation	$O(n^2)$
Interaction with Approximations (e.g., multipole)	$O(n)$

1. Quad-tree Construction

- The quad-tree is recursively subdivided until a maximum depth (d) or particle threshold (t) is reached.
- The worst-case depth of the tree is $\log_4(n)$, as each subdivision reduces the number of particles per node by a factor of 4.

$$\text{Total work} = \sum_{k=0}^{\log_4(n)} 4^k \cdot O(t) = O(n \log n)$$

Thus, the complexity is $O(n \log n)$ for general cases and $O(n)$ when particles are uniformly distributed.

2. Quad-tree Traversal

- Each node is visited once during traversal.

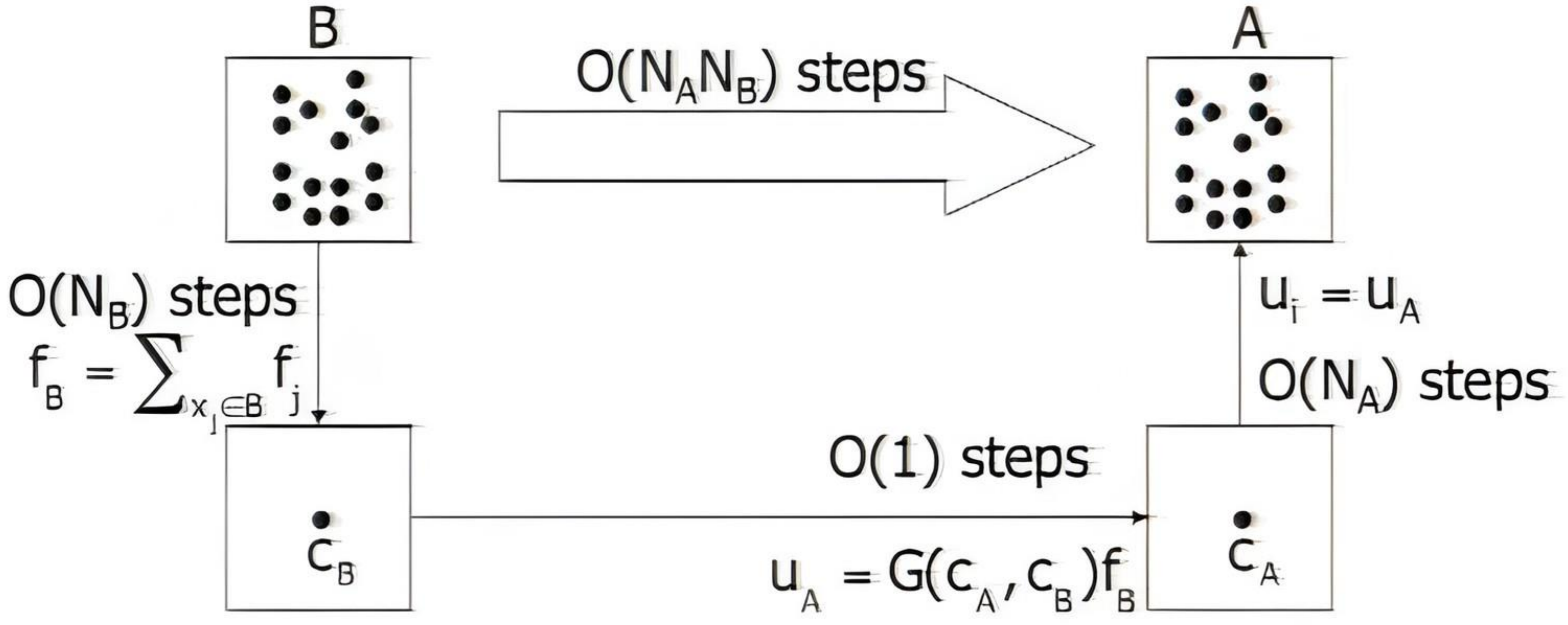
The total number of nodes is proportional to the number of particles, resulting in a time complexity of $O(n)$.

3. Interaction Computation

- For direct interactions between particles in a node, the complexity is $O(n^2)$ in the worst case.
- Using approximations like multipole expansions, the interactions for distant nodes can be aggregated, reducing the complexity to $O(n)$.

OBSERVATION: WELL-SEPARATED REGIONS

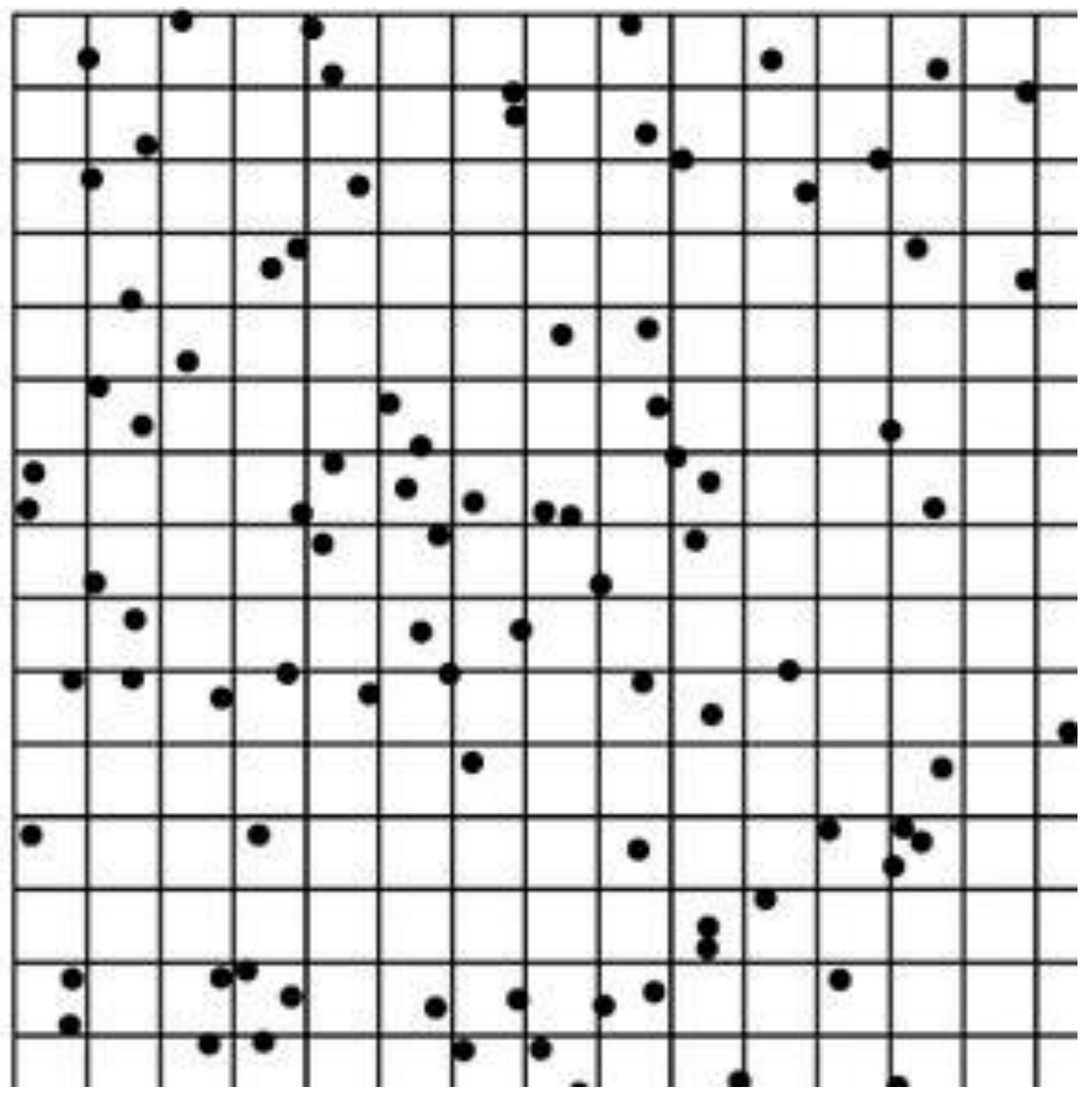
Suppose B and A are well-separated.
Consider the influence from B to A: at each x_i in A, evaluate



- Reduce $O(N_A N_B)$ steps to $O(N_A + N_B)$ steps
- Good accuracy when A and B are far away
- But we will use it whenever A and B are well-separated

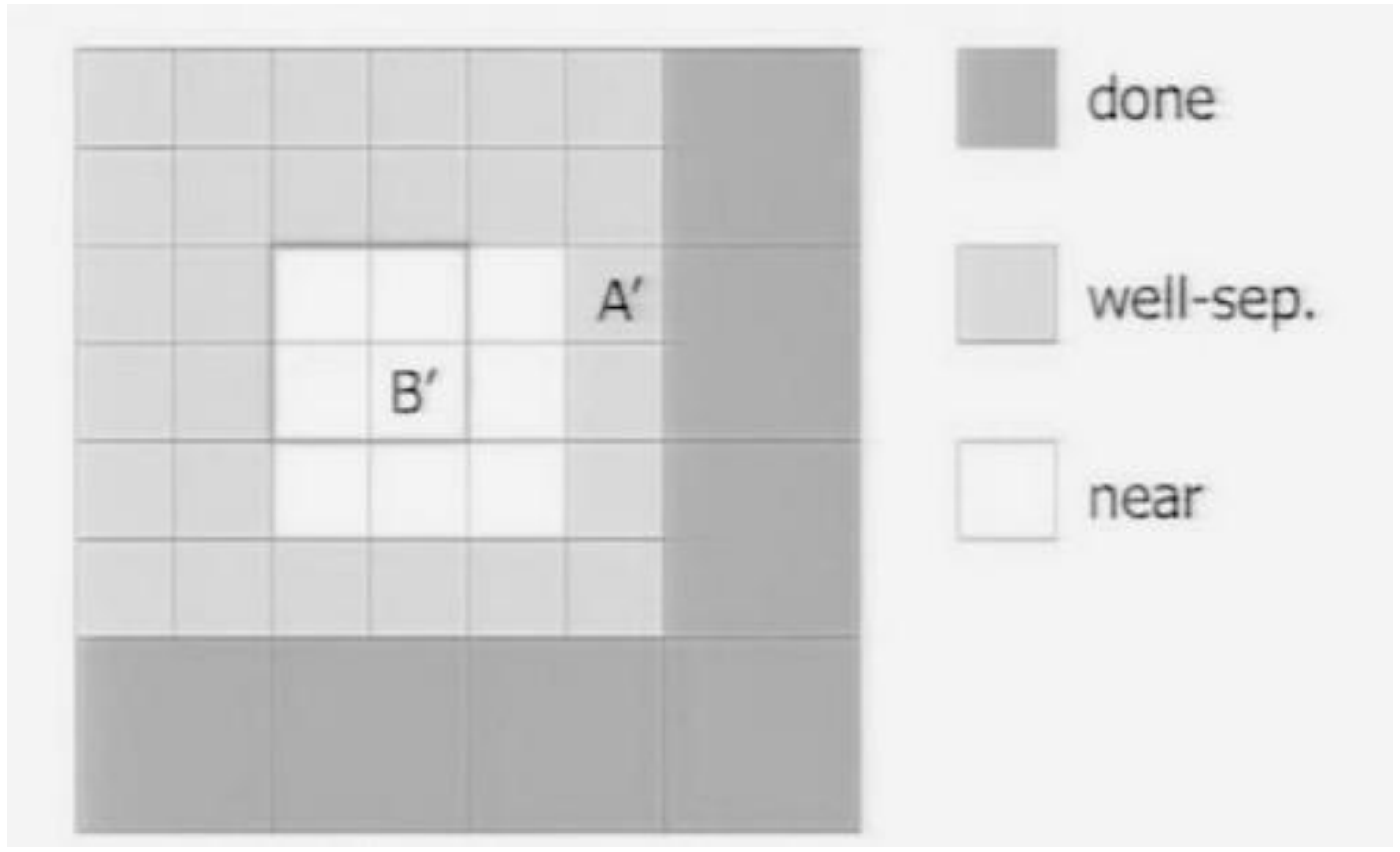
HIERARCHICAL DECOMPOSITION

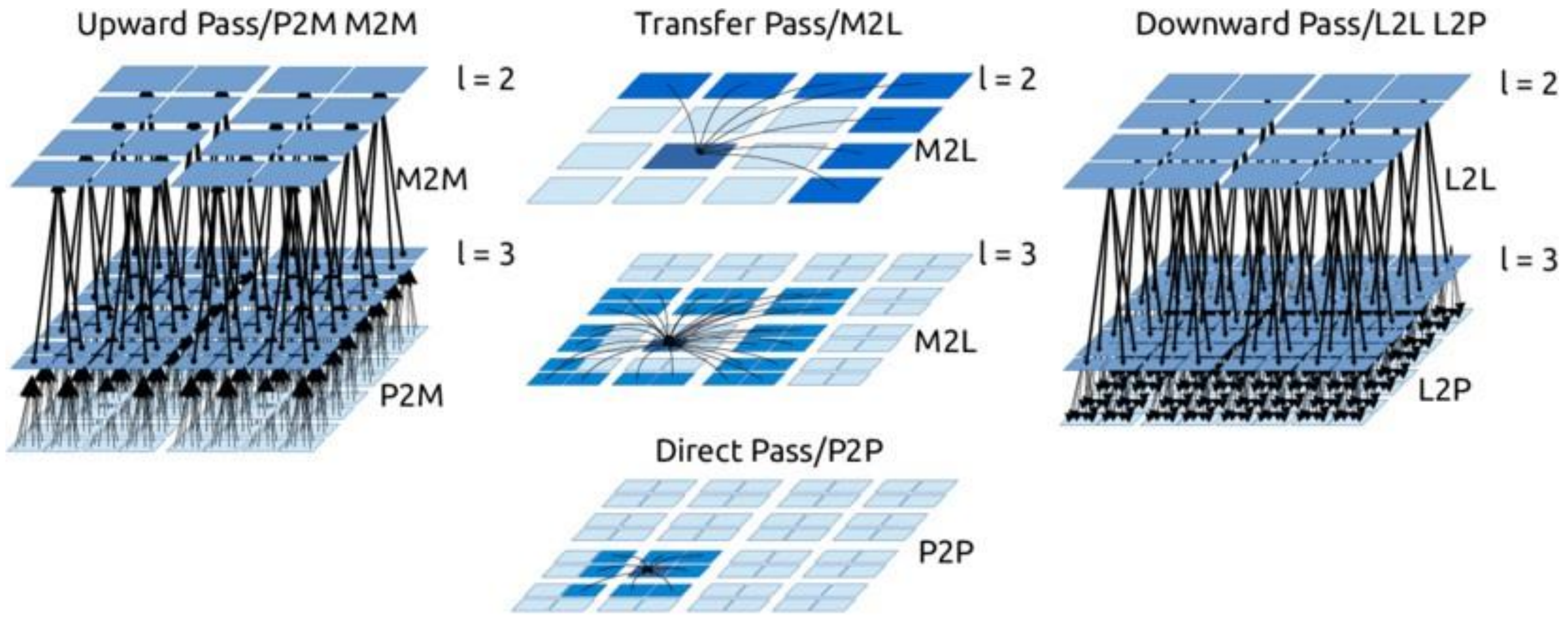
- Partition the domain hierarchically until **each leaf box contains $O(1)$ number of points.**
- N: the # of points
- # of levels $= O(\log_4 N)$
- # of boxes on level $l = O(4^l)$
- Total # of boxes $= O(N)$



HIERARCHICAL DECOMPOSITION

- Here, we use 3- step approx. from calculations involving from B' to A'
- The well separated list contains atmost $6^2 - 3^2 = 27$.





INTERMEDIATE ALGORITHM

BARNES-HUT ALGORITHM

1. Initialization

At the initial stage, the sources in each box of the tree are aggregated into a single equivalent source:

$$f_B = \sum_{x \in B} f$$

This step simplifies the computational problem by replacing individual sources with a combined source strength f_B for each box B.

2. Interactions Between Boxes:

The interactions between distant groups are calculated in a hierarchical manner, starting from the lowest level of the tree:

- For $L = 2$ to the last level of the tree:
  - For each box B at level L:
    - Interact with each box A in B's influence list:

$$u_A \leftarrow u_A + G(c_A, c_B) f_B$$

Here, $G(c_A, c_B)$ represents the Green's function evaluated at the centers of boxes A and B, capturing the effect of the equivalent source in B on A.

3. Propagation to Points

After aggregating the interactions at the box level, the effects are distributed to individual points within each box:

$$u_i \leftarrow u_i + u_A, \quad \text{for } x_i \in A$$

This ensures that each point x_i accumulates the contribution from the aggregated effect u_A of its containing box A.

4. Local Calculation

Finally, for boxes at the lowest level, interactions between nearby points are computed directly to account for local influences:

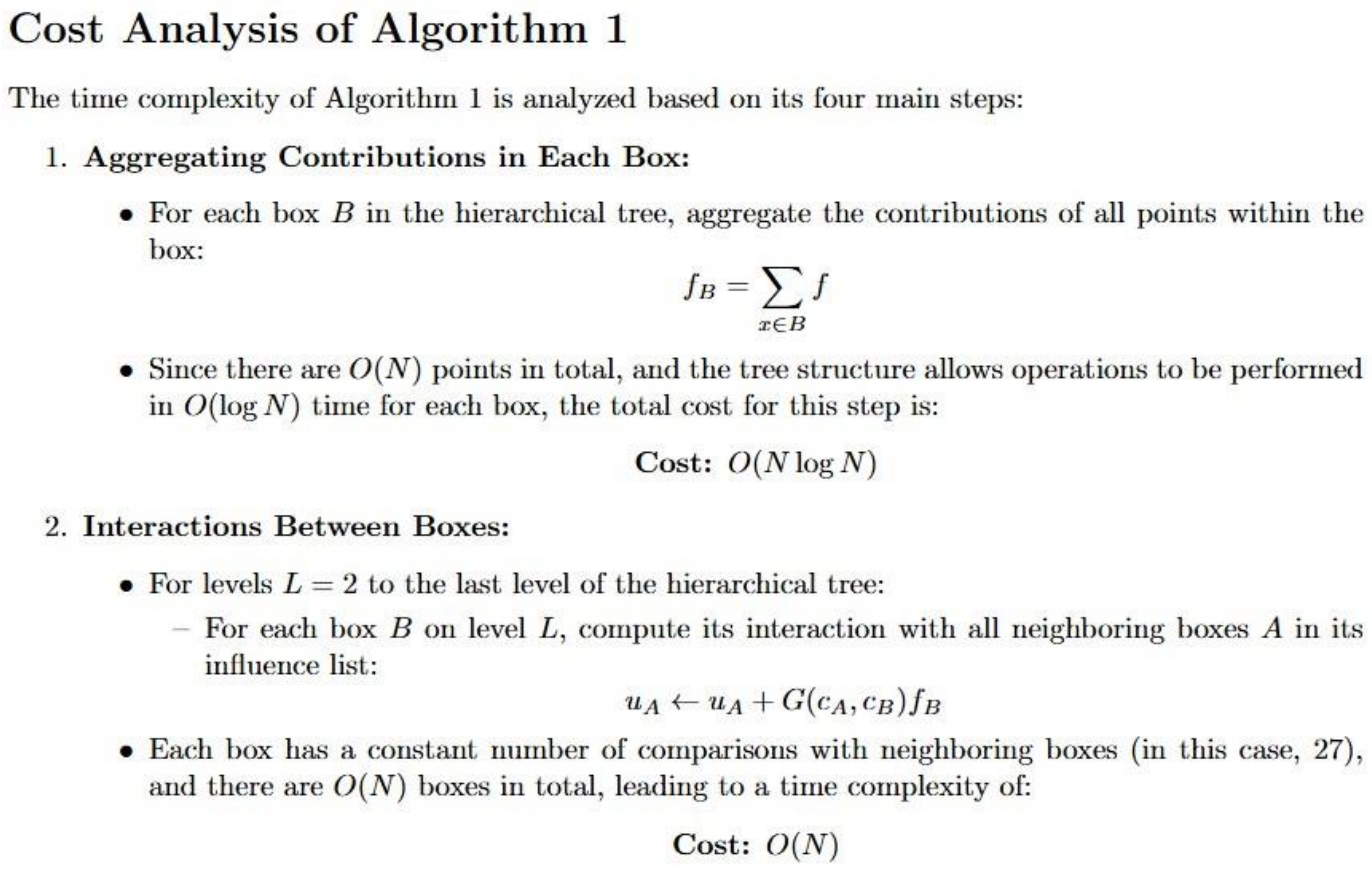
$$u_i \leftarrow u_i + \sum_{x_j \in \text{Nbhd}(B)} G(x_i, x_j) f_j, \quad \text{for } x_i \in B$$

Here, the summation considers only points x_j in the neighborhood of the box B, ensuring accuracy in short-range interactions.

Overview of Algorithm

An intermediate algorithm with a computational complexity of $O(n \log n)$. It achieves this efficiency by hierarchically grouping interactions and approximating long-range forces while maintaining accuracy. The algorithm is described in four main steps:

TIME COMPLEXITY ANALYSIS OF INTERMEDIATE ALGORITHM



Cost Analysis of Algorithm 1

The time complexity of Algorithm 1 is analyzed based on its four main steps:

1. **Aggregating Contributions in Each Box:**
   - For each box B in the hierarchical tree, aggregate the contributions of all points within the box:
   $$f_B = \sum_{x \in B} f$$
   - Since there are $O(N)$ points in total, and the tree structure allows operations to be performed in $O(\log N)$ time for each box, the total cost for this step is:
   $$\textbf{Cost: } O(N \log N)$$
2. **Interactions Between Boxes:**
   - For levels $L = 2$ to the last level of the hierarchical tree:
     – For each box B on level L, compute its interaction with all neighboring boxes A in its influence list:
     $$u_A \leftarrow u_A + G(c_A, c_B) f_B$$
   - Each box has a constant number of comparisons with neighboring boxes (in this case, 27), and there are $O(N)$ boxes in total, leading to a time complexity of:
   $$\textbf{Cost: } O(N)$$

3. **Distributing Results to Individual Points:**
   - For each box A in the tree, update the potential for each point within the box:
   $$u_i \leftarrow u_i + u_A, \quad \text{for } x_i \in A$$
   - Each point must traverse the hierarchical structure of the tree, which results in logarithmic work per point. With $O(N)$ points overall, the total cost for this step is:
   $$\textbf{Cost: } O(N \log N)$$
4. **Computing Direct Interactions:**
   - For each box B on the last level, compute direct interactions between points in neighboring boxes:
   $$u_i \leftarrow u_i + \sum_{x_j \in \mathrm{Nbhd}(B)} G(x_i, x_j) f_j, \quad \text{for } x_i \in B$$
   - The size of each neighborhood is constant, and there are $O(N)$ points overall. Thus, the total cost for this step is:
   $$\textbf{Cost: } O(N)$$

Total Cost

Combining the complexities of all steps, the total cost of Algorithm 1 is:

$$\textbf{Total Cost: } O(N \log N)$$

MULTIPOLE AND LOCAL EXPANSIONS

1. Taylor Expansion

The Taylor series of a function $f(x)$ about $x = a$:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

2. Binomial Expansion

For any real number n:

$$(1+x)^n = \sum_{k=0}^{\infty} \binom{n}{k} x^k, \quad |x| < 1$$

where $\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k!}$.

CLUSTER AS A SOURCE

The potential $\phi(\mathbf{r})$ at a point $\mathbf{r} = (x, y)$ due to a particle at $\mathbf{r}' = (x', y')$ with mass m is given by:

$$\phi(\mathbf{r}) = \frac{m}{|\mathbf{r} - \mathbf{r}'|}$$

Using complex numbers $z = x + iy$ and $z' = x' + iy'$, the potential becomes:

$$\phi(z) = \frac{m}{|z - z'|}$$

When the point $\mathbf{r}$ is far from $\mathbf{r}'$ ($|z'| \ll |z|$), the potential can be approximated by expanding $\frac{1}{z-z'}$ into a Taylor series about z':

$$\frac{1}{z - z'} = \sum_{n=0}^{\infty} \frac{z'^n}{z^{n+1}}$$

Thus, the potential becomes:

$$\phi(z) = \sum_{n=0}^{\infty} M_n \frac{1}{z^{n+1}}$$

where:

$$M_n = \sum_i m_i (z_i')^n$$

Here: - M_n are the multipole coefficients. - z_i' is the position of the i-th particle relative to the source cluster center.

Multipole coefficients of leaf nodes computed directly by summation of masses with the respective power of distance

RELATION BETWEEN PARENT AND CHILD MULTIPOLE COEFFICIENTS

The potential $\phi(\mathbf{r})$ at a point $\mathbf{r}$ due to a set of particles in a leaf node is given by the multipole expansion:

$$\phi(\mathbf{r}) = \sum_{n=0}^{P} \frac{M_n}{|\mathbf{r} - \mathbf{r}_c|^{n+1}}$$

where $M_n = \sum_i m_i(\mathbf{r}_i - \mathbf{r}_c)^n$ are the multipole coefficients, m_i is the mass of the i-th particle, and $\mathbf{r}_c$ is the center of the leaf node.

For a child node's multipole coefficients M_n^{child}, the potential at the parent node is:

$$\phi(\mathbf{z}) = \sum_{n=0}^{P} M_n^{\text{child}} \frac{1}{(\mathbf{z} - \mathbf{z}_c)^{n+1}}$$

where $\mathbf{z}_c$ is the child node center relative to the parent. To translate the multipole expansion from the child to the parent, we expand $\frac{1}{(\mathbf{z}-\mathbf{z}_c)^{n+1}}$ using a binomial expansion:

$$\frac{1}{(\mathbf{z}-\mathbf{z}_c)^{n+1}} = \sum_{k=0}^{p} \binom{n+k}{k} \mathbf{z}_c^k \frac{1}{\mathbf{z}^{n+k+1}}$$

The aggregated multipole coefficient M_k^{parent} for the parent node becomes:

$$M_k^{\text{parent}} = \sum_{i=0}^{4} \sum_{n=0}^{k} \binom{k}{n} M_n^{\text{child}_i} \mathbf{z}_c^{k-n}$$

where $M_n^{\text{child}_i}$ are the multipole coefficients of the child node i, and $\mathbf{z}_c$ is the displacement vector between the parent and child nodes.

CLUSTER AS A TARGET

Multipole Contribution to Potential

The potential at a target point z' due to the source cluster is:

$$\phi(z') = \sum_{n=0}^{P} M_n \frac{1}{(z' - z_c)^{n+1}}$$

where z_c is the center of the source cluster.

Local Expansion at Target Cluster Center

Let $z' = z_t + \Delta z$, where z_t is the center of the target cluster and Δz is the relative position. Substituting this into the potential:

$$\phi(z') = \sum_{n=0}^{P} M_n \frac{1}{(z_t + \Delta z - z_c)^{n+1}}$$

Expanding $(z_L + \Delta z - z_c)^{-n-1}$ using a binomial series:

$$\frac{1}{(z_L + \Delta z - z_c)^{n+1}} = \sum_{k=0}^{\infty} \binom{n+k}{k} \Delta z^k (z_L - z_c)^{n+k+1}$$

Substitute this back into the potential:

$$\phi(z') = \sum_{k=0}^{\infty} L_k \Delta z^k$$

where the local coefficients L_k are:

$$L_k = \sum_{n=0}^{p} M_n (-1)^n (z_L - z_c)^{n+k+1} \binom{n+k}{k}$$

RELATION BETWEEN LOCAL COEFFICIENT OF NODE AND PARENT

We start with the potential at the target point $\mathbf{r}_t$ in the local expansion of a node (either parent or child):

$$\phi(\mathbf{r}_t) = \sum_{k=0}^{P} L_k(\mathbf{r}_t - \mathbf{r}_{\text{node}})^k$$

Where L_k are the local coefficients, and $\mathbf{r}_{\text{node}}$ is the center of the node (either parent or current node).

For the local coefficients at the **parent node**, the potential at the target point is:

$$\phi_{\text{parent}}(\mathbf{r}_t) = \sum_{n=0}^{P} L_n^{\text{parent}}(\mathbf{r}_t - \mathbf{r}_{\text{parent}})^n$$

Where L_n^{parent} are the local coefficients at the parent node.

We now want to translate the local expansion from the **parent node** to the **child node**, shifting the reference frame from the parent's center $\mathbf{r}_{\text{parent}}$ to the child's center $\mathbf{r}_{\text{node}}$.

Let $\mathbf{z} = \mathbf{r}_{\text{node}} - \mathbf{r}_{\text{parent}}$ represent the **relative displacement** between the centers of the parent and child nodes.

Using the **binomial expansion**, we express $(\mathbf{r}_t - \mathbf{r}_{\text{parent}})^n$ (the parent expansion) in terms of powers of $\mathbf{z}$:

$$(\mathbf{r}_t - \mathbf{r}_{\text{parent}})^n = \sum_{k=0}^{n} \binom{n}{k} \mathbf{z}^{n-k} (\mathbf{r}_t - \mathbf{r}_{\text{node}})^k$$

Simplifying the expansion, when dealing with powers of $\mathbf{z}$, we get:

$$(\mathbf{r}_t - \mathbf{r}_{\text{parent}})^n = \sum_{k=0}^{n} \binom{n}{k} \mathbf{z}^{n-k} (\mathbf{r}_t - \mathbf{r}_{\text{node}})^k$$

Now, applying this to the local coefficients, the local expansion at the **parent node** becomes:

$$\phi_{\text{parent}}(\mathbf{r}_t) = \sum_{n=0}^{P} L_n^{\text{parent}} (\mathbf{r}_t - \mathbf{r}_{\text{parent}})^n$$

When translating this to the **child node**, the local coefficients L_k^{node} at the child node are related to the parent's local coefficients L_n^{parent} using the binomial expansion of $(\mathbf{r}_t - \mathbf{r}_{\text{parent}})$ into powers of the relative displacement $\mathbf{z}$:

$$\phi_{\text{node}}(\mathbf{r}_t) = \sum_{k=0}^{P} L_k^{\text{node}} (\mathbf{r}_t - \mathbf{r}_{\text{node}})^k$$

This is equivalent to:

$$L_k^{\text{node}} = \sum_{n=k}^{P} \binom{n}{k} L_n^{\text{parent}} \mathbf{z}^{n-k}$$

Where: - $\mathbf{z} = \mathbf{r}_{\text{node}} - \mathbf{r}_{\text{parent}}$ is the relative displacement between the parent and child node centers. - L_n^{parent} are the local coefficients of the parent node. - $\binom{n}{k}$ is the binomial coefficient. - $\mathbf{z}^{n-k}$ is the relative displacement raised to the power $n - k$.

COMPUTATION OF FORCES

Local Expansion Contribution

For a particle at position $\mathbf{r}'$, the force is given by:

$$\mathbf{F}(\mathbf{r}') = -\nabla\phi(\mathbf{r}') = -\sum_{k=1}^{P} kL_k(\mathbf{r}' - \mathbf{r}_c)^{k-1}$$

where: - L_k are the local coefficients, - $\mathbf{r}_c$ is the center of the source cluster, - P is the truncation order for the expansion.

Direct Interaction

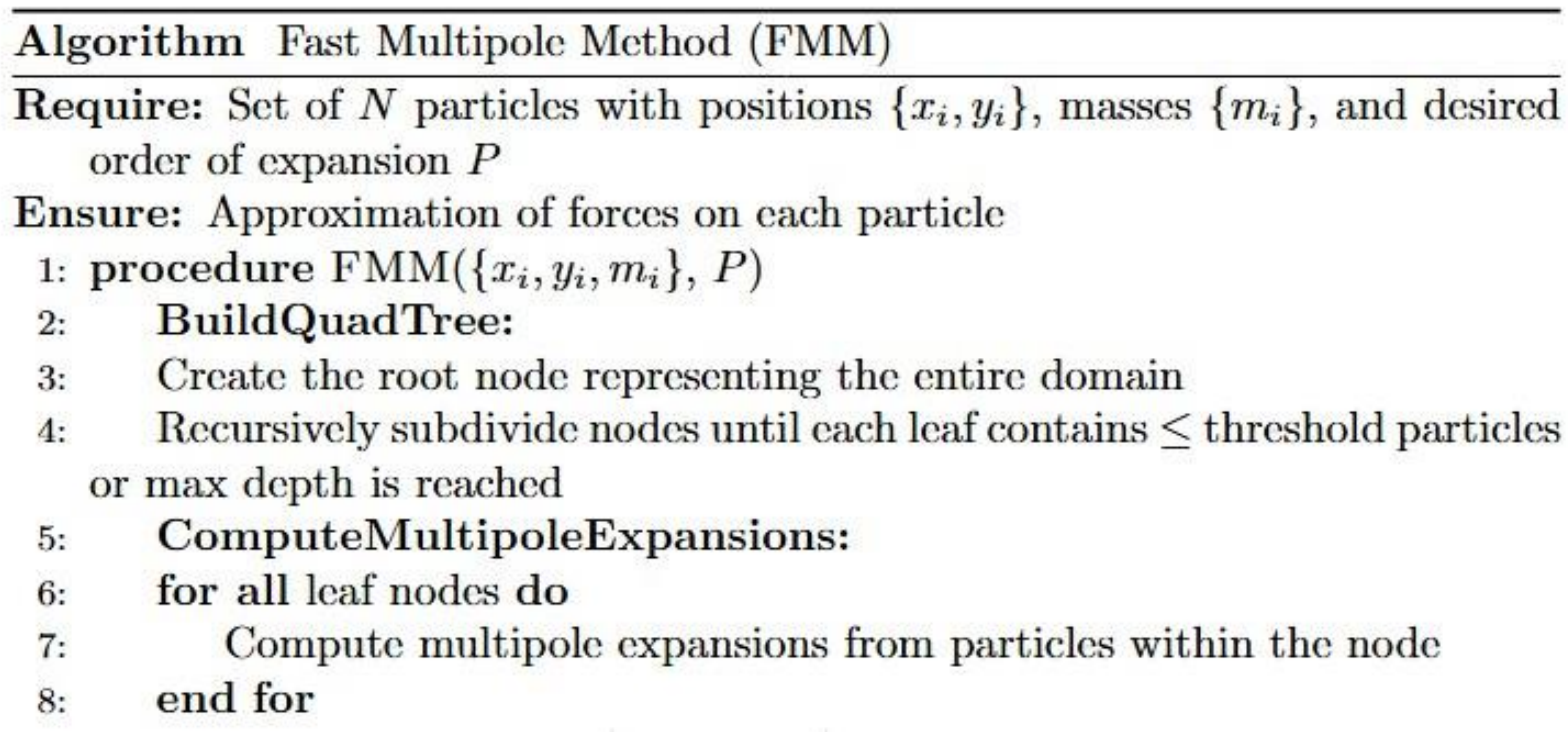
For nearby particles within the same or adjacent nodes, the force is given by:

$$\mathbf{F}(\mathbf{r}') = \sum_{j\neq i} \frac{m_j(\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^3}$$

where: - m_j is the mass of particle j, - $\mathbf{r}_j$ and $\mathbf{r}_i$ are the positions of particles j and i, - $|\mathbf{r}_j - \mathbf{r}_i|^3$ is the cube of the distance between particles j and i.

OPTIMAL ALGORITHM

O(N) ALGORITHM

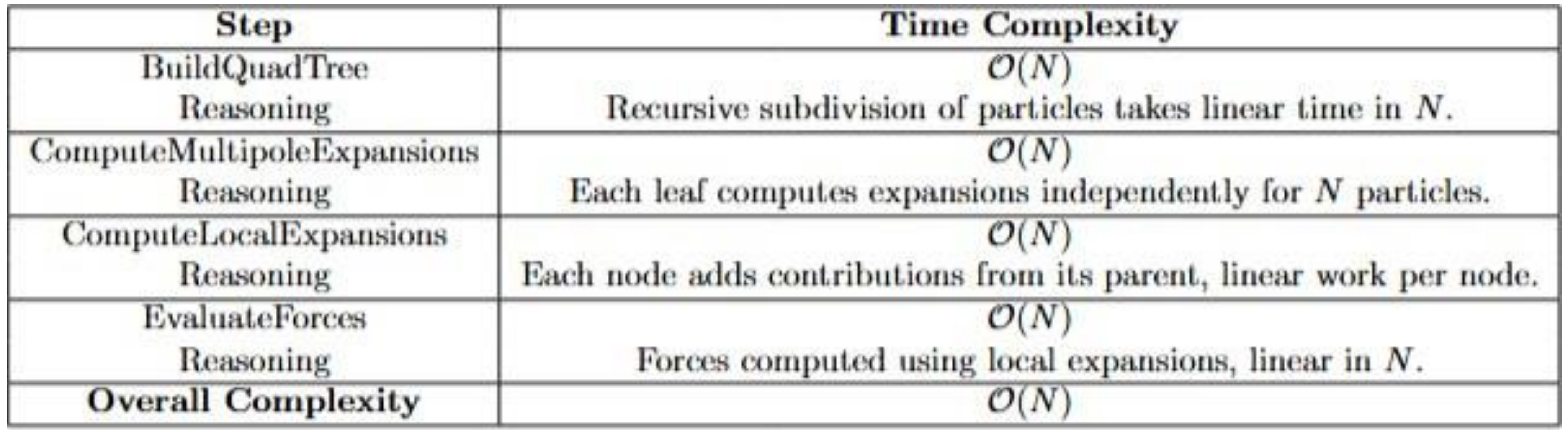


Algorithm Fast Multipole Method (FMM)

Require: Set of N particles with positions $\{x_i, y_i\}$, masses $\{m_i\}$, and desired order of expansion P

Ensure: Approximation of forces on each particle

```
procedure FMM($\{x_i, y_i, m_i\}$, $P$)
    BuildQuadTree:
    Create the root node representing the entire domain
    Recursively subdivide nodes until each leaf contains $\leq$ threshold particles
or max depth is reached
    ComputeMultipoleExpansions:
    for all leaf nodes do
        Compute multipole expansions from particles within the node
    end for
    **for all** internal nodes (bottom-up) **do**
        Aggregate multipole expansions from child nodes
    **end for**
    **ComputeLocalExpansions:**
    **for all** nodes in the tree (top-down) **do**
        **if** node is not the root **then**
            Translate parent's local expansion to the current node
        **end if**
        Add contributions from well-separated nodes (via interaction lists)
    **end for**
    **EvaluateForces:**
    **for all** leaf nodes **do**
        Use local expansions to approximate forces on particles
        Compute direct interactions for nearby particles (fallback)
    **end for**
**end procedure**
```



Step	Time Complexity
BuildQuadTree	$\mathcal{O}(N)$
Reasoning	Recursive subdivision of particles takes linear time in N.
ComputeMultipoleExpansions	$\mathcal{O}(N)$
Reasoning	Each leaf computes expansions independently for N particles.
ComputeLocalExpansions	$\mathcal{O}(N)$
Reasoning	Each node adds contributions from its parent, linear work per node.
EvaluateForces	$\mathcal{O}(N)$
Reasoning	Forces computed using local expansions, linear in N.
Overall Complexity	$\mathcal{O}(N)$

BuildQuadTree: O(N)
Explanation: In this step, particles are recursively inserted into a quadtree structure. The tree is subdivided until each leaf contains a limited number of particles or the maximum depth is reached. As each particle is processed once, the time complexity is proportional to the number of particles, N, resulting in a linear time complexity of O(N).

ComputeMultipoleExpansions: O(N)
Explanation: In this phase, multipole expansions are computed for each leaf node, which contains particles. Since each particle requires a fixed amount of work to compute its multipole expansion and there are N particles, this step takes linear time in terms of the total number of particles.

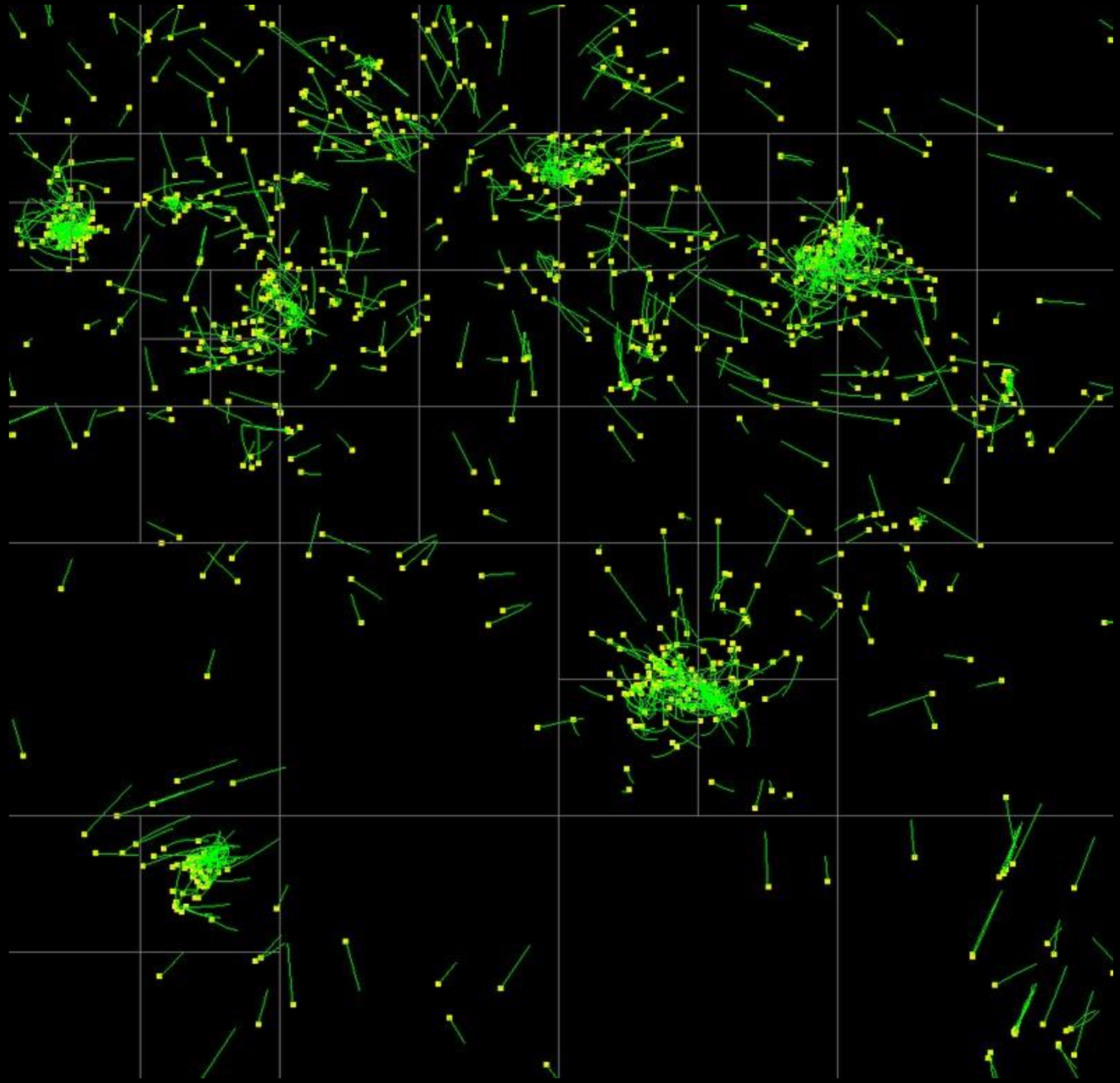
ComputeLocalExpansions: O(N)
Explanation: Here, local expansions are computed for each node in the tree. If the node is not the root, it uses the multipole expansion of its parent to compute the local expansion. Since each node performs a fixed amount of work, and there are N particles distributed across the nodes, the total time complexity remains linear, i.e., O(N).

EvaluateForces: O(N)
Explanation: The final step involves calculating the forces on each particle. The local expansions computed earlier are used to approximate the forces. Additionally, direct interactions are calculated for nearby particles. Since each particle interacts with others in a fixed manner, the total time complexity is linear in terms of N.

Overall Complexity: O(N)
Explanation: Since each step of the FMM algorithm has linear time complexity, the overall complexity of the algorithm is O(N), where N is the number of particles. The use of a quadtree and multipole expansions allows FMM to reduce the computational complexity compared to direct pairwise interaction methods, which would have a time complexity of O(N^2).

OUR SIMULATION

This simulation implements a particle system using the Fast Multipole Method (FMM) for efficient force calculations in a 2D domain. Particles interact via gravitational-like forces, and their motion is updated over time using Euler integration. A quadtree structure is used to organize particles spatially, enabling hierarchical multipole expansions for approximating distant interactions.
Key features include dynamic quadtree construction, multipole and local expansions, real-time visualization with OpenGL, and particle clustering for initial conditions. The system supports customizable particle count and parameters, making it suitable for exploring large-scale particle dynamics.



CONCLUSION

Method	Time Complexity	Description
Naive Approach	$O(N^2)$	Direct pairwise computation
Intermediate Approach	$O(N \log N)$	Quadtrees, hierarchical
FMM Approach	$O(N)$	Multipole expansions, fast evaluation

Table 1: Comparison of Time Complexity in Different Methods

These methods don't just exist in theory—they are applied in groundbreaking real-world applications. The Fast Multipole Method (FMM) is at the core of many advanced technologies, including:

- **Electrostatics** in molecular dynamics simulations
- **Gravitational simulations** in astrophysics
- **Image processing** and **acoustic modeling** in engineering

In every field, FMM has significantly reduced computational costs and increased the feasibility of solving previously intractable problems.

The future of FMM looks bright, especially with the growing importance of **multiscale modeling**, where FMM's ability to handle large datasets at multiple resolutions will play a pivotal role. The ability to simulate:

- **Large-scale particle systems**
- **Weather forecasting**
- **Financial models**

has now been made more accessible, thanks to the remarkable scalability of FMM.

By mastering these algorithms, we're not just optimizing computations—we're paving the way for innovations in **AI**, **machine learning**, and **quantum computing**, where fast and efficient numerical methods are more critical than ever. Whether you're building the next AI model or simulating the forces of the universe, FMM is more than just an algorithm; it's a **game-changer** in the world of computational science.

REFERENCE

1. Greengard, L., & Rokhlin. V (1987). *A fast algorithm for particle simulations.* Journal of Computational Physics, 73(2), 325-348. The seminal paper introducing the Fast Multipole Method (FMM).

2. Samet, H. (1984). *The quadtree and related hierarchical data structures.* ACM Computing Surveys, 16(2), 187-260. Explores hierarchical data structures like quadtrees.

3. Zhang, Y., & Wang, X. (2020). *Optimization and application of the fast multipole method.* In Proceedings of the 2020 International Workshop on Mechanical, Electrical, and Computer Systems.

4. Lexing Ying, Professor of Mathematics, Stanford University (2015). *The Fast Multipole Method (FMM) explained.* YouTube video. Available at `https://www.youtube.com/watch?v=qMLIyZi8Sz0`.

5. OpenAI. (2023). *Generative Pre-trained Transformer (GPT) Models.* OpenAI developed the GPT series, state-of-the-art large language models that leverage deep learning for text generation and natural language processing. Available at `https://openai.com`.