Large-Graph Layout with the Fast Multipole Multilevel Method

STEFAN HACHUL and MICHAEL JÜNGER
Universität zu Köln, Institut für Informatik

The visualization of of large and complex networks or graphs is an indispensable instrument for getting deeper insight into their structure. Force-directed graph-drawing algorithms are widely used to draw such graphs. However, these methods do not guarantee a sub-quadratic running time in general. We present a new force-directed method that is based on a combination of an efficient multilevel scheme and a strategy for approximating the repulsive forces in the system by rapidly evaluating potential fields. Given a graph $G = (V, E)$, the asymptotic worst-case running time of this method is $O(|V| \log |V| + |E|)$ with linear memory requirements. In practice, the algorithm generates nice drawings of graphs with 100000 nodes in less than 5 minutes. Furthermore, it clearly visualizes even the structures of those graphs that turned out to be challenging for other methods.

Categories and Subject Descriptors: ... [...]: ...

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Fast Algorithms, Graph Drawing, Large Graphs, Multilevel Methods, N-Body Problem

1. INTRODUCTION

The biochemical reactions of proteins in baker's yeast, the ecosystem of plankton, sea perch, and anchovy, the American electricity network, the international air traffic, and the world-wide web have in common that they can be modeled as graphs. In general a graph $G = (V, E)$ is used to model information that can be described as objects (the node set V) and connections between those objects (the edge set E).

One fundamental tool for analyzing such graphs is the automatic generation of layouts that visualize the graphs and are easy to understand. A popular class of algorithms that is used to visualize general graphs are force-directed graph-drawing methods. Given a graph $G = (V, E)$, these methods generate drawings of G in the plane in which each edge is represented by a straight line connecting its two adjacent nodes. The computation of the drawings is based on associating G with a physical model. Then, an iterative algorithm tries to find a placement of the nodes so that the total energy of the physical system is minimal. The desired esthetic criteria of

Authors address: Universität zu Köln, Institut für Informatik, Pohligstraße 1, 50969 Köln, Germany, email: {hachul,mjuenger}@informatik.uni-koeln.de}; This research was partially supported by ADONET.

these methods are uniformity of edge length, few edge crossings, non-overlapping nodes, and the display of symmetries if some exist.

In practice, classical force-directed algorithms like [Eades 1984; Kamada and Kawai 1989; Fruchterman and Reingold 1991; Davidson and Harel 1996] are only suited for drawing small graphs with at most a few hundreds of vertices, since their worst-case running time is at least quadratic. On the other hand, the demand for algorithms that can handle significantly larger instances is steadily increasing. Hence, accelerated force-directed algorithms have been developed [Tunkelang 1998; Quigley and Eades 2001; Gajer et al. 2001; Harel and Koren 2001; Walshaw 2001]. These algorithms generate nice drawings of a big range of large graphs with several thousands of nodes in reasonable time. However, only some of these methods guarantee a sub-quadratic running time in special cases or under certain assumptions, but not in general.

Besides force-directed algorithms other fast methods for drawing large graphs have been invented by Harel and Koren [Harel and Koren 2002] and Koren et al. [Koren et al. 2003]. These methods are based on techniques of linear algebra and not on physical analogies. But they strive for the same esthetic drawing criteria.

In Sections 2 to 5 we describe the most important parts of a new force-directed graph-drawing algorithm that guarantees a sub-quadratic worst-case running time. The algorithm has been implemented and experimental results are presented in Section 6.

2. THE FAST MULTIPOLE MULTILEVEL METHOD (FM^3)

The most essential parts of the new method that is called *Fast Multipole Multilevel Method* or shorter (FM^3) are an efficient multilevel strategy and an $O(|V| \log |V|)$ approximation algorithm to obtain the repulsive forces that act between all pairs of nodes (associated with charged particles). These modules are described in Section 3 and 4, respectively. Other parts like a preprocessing step that enables the algorithm to draw graphs with nodes of different sizes and a module that is designed to handle disconnected graphs are not described here for brevity. Therefore, we can simply assume that the given graph G is a connected positive weighted graph. The edge weight of each edge $e \in E$ represents its individual desired edge length $\mathit{desired\_edge\_length}(e)$.

Since FM^3 is a force-directed graph-drawing algorithm, we must choose a force model first. This is done by identifying the nodes with charged particles that repel each other and by identifying edges with springs, like in most classical force-directed methods. If in $\mathbb{R}^2$ two charges u, v are placed at a distance d from each other, the repulsive forces between u and v are proportional to $1/d$. Our choice of the spring forces is not strictly related to physical reality. We found that choosing the spring force of an edge e to be proportional to $\log(d/\mathit{desired\_edge\_length}(e)) \cdot d^2$ gives very good results in practice.

3. THE MULTILEVEL STRATEGY

Since in classical force-directed algorithms many iterations are needed to transform an initial (random) drawing of a large graph into the final drawing, one might hope to reduce the constant factor of force-directed algorithms by using a multilevel

strategy. Multilevel strategies have been introduced into force-directed graph drawing by [Gajer et al. 2001; Harel and Koren 2001] and [Walshaw 2001]. They share the following basic ideas: Given $G = (V, E) =: G_0$, a series of graphs $G_1, \ldots, G_k$ with decreasing sizes is created in a *coarsening phase*. In the following *refinement phase* the smallest graph G_k at *level* k is drawn using (a variation of) a classical force-directed *(single-level)* algorithm. This drawing is used to get an initial layout of the next larger graph G_{k-1} that is drawn afterwards. This process is repeated until the original graph G_0 is drawn.

Unlike previous approaches, we want to design a multilevel algorithm that has provably the same asymptotic running time as the single-level algorithm that is used to draw all graphs G_i with $i = 0, \ldots, k$.

3.1 The Coarsening Phase

In the coarsening phase, we first partition the node set of G into disjoint subsets so that the vertex-induced subgraphs (called *solar systems*) have bounded diameter (at most 4).

Definition 3.1 *(Solar System).* Suppose, $G = (V, E)$ is a graph and $U \subseteq V$. The vertex-induced subgraph $S := G[U]$ is called *solar system* if the following conditions hold: Exactly one node in U is marked as *sun node* (or *s-node*). Each of its neighbors is marked as *planet node* (or *p-node*) or as *planet-with-moon node* (or *pm-node*) and is also contained in U. The other nodes in U are marked as *moon nodes* (or *m-nodes*), each m-node is required to have graph-theoretic distance two to the s-node in U, and each m-node is assigned to exactly one pm-node in U. Furthermore, for each pm-node exists at least one m-node that is assigned to it.

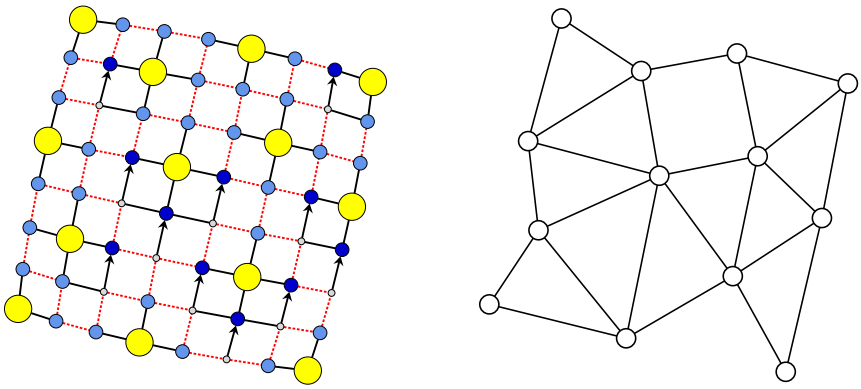


Fig. 1. (left) Drawing of $G = G_0$. (right) The multilevel graph G_1.

Figure 1(left) shows an example of a grid graph that is partitioned into 13 solar systems. The s-nodes, p-nodes, pm-nodes, and m-nodes are represented by the

big yellow, medium light blue, medium dark blue, and small gray disks, respectively. The black solid edges represent *intra* solar-system edges, whereas the edges connecting nodes of two different solar systems (*inter* solar-system edges) are red dashed edges. The edges that connect an m-node and its assigned pm-node are drawn as directed edges, indicating that the m-node is assigned to this pm-node.

Next, we describe a linear time method for constructing a solar-system partition of a graph G that works in three steps: First, we create the s-nodes. Therefore, we store a candidate set V' that is a copy of V and randomly select a first s-node s_1 from V'. Then, s_1 and all nodes that have a graph-theoretic distance at most 2 from s_1 in G are deleted from V'. We iteratively select the next s-nodes in the same way, until V' is empty and $Suns = s_1, \ldots, s_l$ is the list of all s-nodes. Second, for each $s_i \in Suns$ all its neighbors are labeled as p-nodes. Finally, there might be some nodes in V that are neither labeled as p-nodes, nor as s-nodes. These nodes are the m-nodes. Each m-node is assigned to the p-node that is its nearest neighbor in G, and we relabel this p-node as pm-node.

Given a solar-system partition of the node set of $G = G_0$, we construct a smaller graph G_1 by collapsing (shrinking) the node set of each solar system into one single node and deleting parallel edges (see Figure 1(right)). The smaller graph should reflect the attributes of the bigger graph as much as possible. Therefore, the desired edge length of an edge $e_1 = (s_1, t_1)$ in G_1 is initialized as follows: Suppose, p-node u_0 belongs to the solar system S_0 with sun node s_0 in G_0 and p-node v_0 belongs to the solar system T_0 with sun node t_0 in G_0. Let us also assume that the edge $e_0 = (u_0, v_0)$ is the unique inter solar-system edge connecting S_0 and T_0. Furthermore, we assume that nodes s_1 and t_1 in G_1 are obtained by collapsing S_0 and T_0. Then, the desired edge length of e_1 is set to $\mathit{desired\_edge\_length}((s_0, u_0)) + \mathit{desired\_edge\_length}(e_0) + \mathit{desired\_edge\_length}((v_0, t_0))$. For later use, we denote the corresponding path (s_0, u_0, v_0, t_0) in G_0 by P_0 and its length by p_0. Figure 3.1 shows an example. The case that u_0 and/or v_0 is a moon node is treated similarly. If more than one inter solar-system edge in G_0 connects nodes of S_0 with nodes of T_0, the average of the previously calculated desired edge lengths is assigned to the desired edge length of e_1.

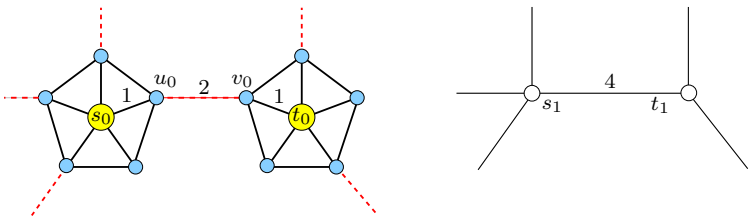
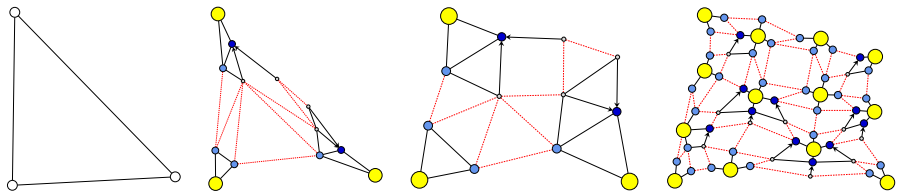


Fig. 2. Calculation of the edge weights: (left) A part of $G = G_0$. (right) The corresponding part of G_1.

This partitioning and collapsing process is iterated until either the smallest graph G_k contains only a constant number of nodes or a certain stopping criterion is fulfilled (that will be defined later).

3.2 The Refinement Phase

In the refinement phase, G_k is drawn by a force-directed single-level algorithm that will be introduced in Section 4. Going upwards to G_{k-1}, we assign initial positions to the nodes of G_{k-1} in two steps: First, we place each s-node of G_{k-1} at the position of its ancestor (that represents its solar system) in the drawing of G_k. Now, the other nodes of G_{k-1} are placed. This is done by using information that has been generated during the collapsing process: For example, given u_0, v_0, s_0, t_0, p_0, and P_0 like in the example above, we place u_0 on the line connecting s_0 and t_0 at position $\mathrm{Pos}(s_0) + \frac{desire\_edge\_length((s_0,u_0))}{p_0}(\mathrm{Pos}(t_0) - \mathrm{Pos}(s_0))$. If u_0 belongs to more than one such path P_0, the barycenter of all these calculated positions is taken as its initial placement. The case that u_0 is a moon node is treated similarly. Figure 3 demonstrates this procedure.



(a) (b) (c) (d)

Fig. 3. (a) Drawing of G_2. (b) Initial placement of G_1. (c) Drawing of G_1. (d) Initial placement of G_0.The final drawing of $G_0 = G$ is shown in Figure 1(left).

THEOREM 3.2 MULTILEVEL STRATEGY. *Suppose, $G = (V, E)$ is a connected positive-weighted graph and* $\mathtt{A_{single}}$ *is a force-directed single-level algorithm that needs time $t_{single}(|V|, |E|)$ to draw G starting with an arbitrary initial placement of the nodes of G. Then, the previously described multilevel strategy that uses* $\mathtt{A_{single}}$ *to draw the multilevel graphs generates a straight-line drawing of G in $O(t_{single}(|V|, |E|))$ time.*

PROOF. The total running time of the multilevel strategy is $t_{\mathrm{mult}}(|V|, |E|) = \sum_{i=0}^{k-1} t_{\mathrm{create}}(|V_i|, |E_i|) + \sum_{i=0}^{k-1} t_{\mathrm{init\_pl}}(|V_i|, |E_i|) + \sum_{i=0}^{k} t_{\mathrm{single}}(|V_i|, |E_i|)$. Here, $t_{\mathrm{create}}(|V_i|, |E_i|)$ denotes the time that is needed to create the multilevel graph G_{i+1} from G_i. $t_{\mathrm{init\_pl}}(|V_i|, |E_i|))$ denotes the time that is needed to get an initial placement of the nodes of the multilevel graph G_i from the drawing of G_{i+1}.

Since every node of G_i belongs to a solar system, and a solar system contains at least two nodes, G_{i+1} contains at most $|V_i|/2$ nodes.

Let us assume that $|E_{i+1}| \leq |E_i|/2$ for all $i \in \{0, \ldots, k-1\}$. Since both $t_{create}(|V_i|, |E_i|)$ and $t_{\mathrm{init\_pl}}((|V_i|, |E_i|)$ are linear in $|V_i| + |E_i|$, we get $\sum_{i=0}^{k-1} t_{\mathrm{create}}(|V_i|, |E_i|) + \sum_{i=0}^{k-1} t_{\mathrm{init\_pl}}(|V_i|, |E_i|) = O(|V| + |E|)$. Furthermore, we get the following estimation on t_{single}: $\sum_{i=0}^{k} t_{\mathrm{single}}(|V_i|, |E_i|) \leq \sum_{i=0}^{k} O\left(t_{\mathrm{single}}\left(\frac{|V|}{2^i}, \frac{|E|}{2^i}\right)\right) \leq 2\, O(t_{\mathrm{single}}(|V|, |E|))$.

The second inequality holds for sufficiently large values of $|V|$ and $|E|$, since $t_{\text{single}}(|V|,|E|) = \Omega(|V| + |E|)$. Therefore, under the previous assumption, the multilevel strategy and $\texttt{A}_{\texttt{single}}$ have the same asymptotic running time.

Certainly, it cannot be guaranteed that the number of edges decreases by factor at least $\frac{1}{2}$. However, it can be shown by an analogous argumentation that $t_{\text{mult}}(|V|,|E|) = O\left(t_{\text{single}}(|V|,|E|)\right)$ if $|E_{i+1}| \leq |E_i|/s$ for all $i \in \{0,\ldots,k-1\}$ and a fixed divisor $1 < s \leq 2$. If this assumption is weakened further so that $|E_{i+1}| \leq \frac{|E_i|}{s}$ for all but a constant number d of $i \in \{0,\ldots,k-1\}$, the equality $t_{mult}(|V|,|E|) = O(t_{single}(|V|,|E|))$ holds as well.

Therefore, in order to guarantee that $t_{\text{mult}}(|V|,|E|) = O(t_{\text{single}}(|V|,|E|))$ it is sufficient to stop the coarsening process, either if the number of nodes of the actual created multilevel graph G_i is smaller than a constant or if the algorithm has generated more than a constant number of graphs G_i that do not satisfy the inequality $|E_{i+1}| \leq |E_i|/d$ for some small constant d with $1 < d \leq 2$. □

4. THE FORCE CALCULATION STEP

In order to save running time, the multilevel algorithms [Gajer et al. 2001; Harel and Koren 2001; Walshaw 2001] use the grid-variant algorithm of [Fruchterman and Reingold 1991] or variations of [Kamada and Kawai 1989] as force-directed single-level algorithms. Those algorithms are comparatively inaccurate approximative variations of the original force-directed single-level algorithms [Fruchterman and Reingold 1991; Kamada and Kawai 1989].

Unlike this, the single-level algorithm that is used in $\texttt{FM}^3$ follows the basic strategy of [Tunkelang 1998; Quigley and Eades 2001] by approximating the repulsive forces between all pairs of distinct nodes/particles with high accuracy and calculating the forces induced by the edges/springs exactly. Then, these forces are added, and the nodes are moved in direction of the resulting forces. This process is repeated a constant number of iterations. (In practice, we let the constant decrease from 300 iterations for G_k to 30 iterations for G_0, although convergence is reached even faster for many tested graphs.)

In each iteration all spring-forces of multilevel graph $G_i = (V_i, E_i)$ can be calculated in $\Theta(|E_i|)$ time. In contrast to this, a naive direct calculation of the repulsive forces acting between all pairs of charged particles (nodes) needs $\Theta(|V_i|^2)$ time. This is the main bottleneck of many force-directed algorithms. Hence, in the remainder of this section, we will concentrate on the problem of calculating the repulsive forces efficiently.

[Greengard 1988] has invented an N-body simulation method that is based on the evaluation of the field of the potential energy of N charged particles. This is done by evaluating multipole expansions using a hierarchical data structure called quadtree. However, [Aluru et al. 1998] have shown that the running time of Greengard's method depends on the particle distribution and cannot be bounded in N. They additionally have proven that the running time of the popular force-approximation method of [Barnes and Hut 1986] that is an essential part of the graph-drawing methods [Tunkelang 1998] and [Quigley and Eades 2001] cannot be bounded in N. Using the techniques and analytical tools of [Greengard 1988], [Aluru et al. 1998] have presented an $O(N \log N)$ approximative multipole algorithm that is

distribution independent.

Based on the work of [Greengard 1988] and [Aluru et al. 1998], we have developed a new $O(N \log N)$ multipole method. Practical experiments indicate that — at same level of accuracy — our new method is up to factor 5 faster than Aluru et al.'s method. The new multipole method works in two phases: Given a distribution of N particles in the plane, first, a special quadtree data structure is constructed. Then, each node of the quadtree is assigned information that is used to approximate the potential energy of the system. In particular, a constant number of coefficients of a so called *multipole expansion* (to be introduced later) are associated with each tree node and are used to obtain the repulsive forces.

4.1 Phase 1: Construction of the Reduced Bucket Quadtree

Definition 4.1 *(Quadtree, Reduced Bucket Quadtree).* Suppose, a set of N particles (or data points) $C = \{c_1, \ldots, c_N\}$ are assigned distinct positions on a square D and we fix a *leaf capacity* $l \geq 1$. (In practice, we choose $l = 25$.) Furthermore, suppose one recursively subdivides D into four squares of equal size (in the following denoted by *boxes*), until each box contains at most l particles. This process can be represented by an ordered rooted tree of maximum child degree four (with the root representing D) that is called *(bucket) quadtree*. The particles are stored in the leaves of the quadtree. A *degenerate path* $P = (v_1, \ldots, v_p)$ in a quadtree is a path in which v_1 and v_p have at least 2 nonempty children and $v_2, \ldots, v_{p-1}$ each have exactly one nonempty child. A *reduced (bucket) quadtree* T can be obtained from a quadtree by shrinking degenerate paths $P = (v_1, \ldots, v_p)$ to edges (v_1, v_p). Figure 4 shows an example.

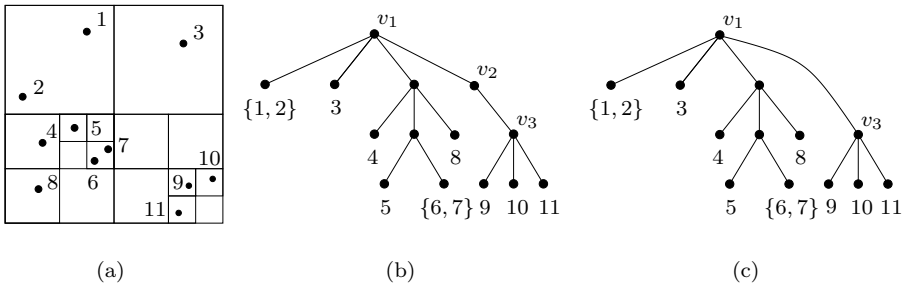


Fig. 4. (a) A distribution of $N = 11$ particles in the plane. (b) The quadtree with leaf capacity $l = 2$ associated with (a). $P = (v_1, v_2, v_3)$ is a degenerate path in the quadtree. (c) The reduced quadtree with leaf capacity $l = 2$ associated with (a).

A reduced quadtree has only $O(N)$ nodes independently of the distribution of the particles. This allows the development of a linear time method (excluding the time needed for constructing the reduced quadtree) for approximating the repulsive forces, using this structure in Phase 2 (see Section 4.2).

[Aluru et al. 1998] present an $O(N \log N)$ method that constructs a reduced quadtree with $l = 1$. It can be shown by a reduction from sorting that it is neither

possible to construct a bucket quadtree nor a reduced bucket quadtree with constant fixed leaf capacity l for arbitrary distributions of the N particles in $o(N \log N)$ time.

[Hachul 2005] has developed an $O(N \log N)$ reduced-bucket-quadtree construction method that is omitted here, since it quite technical. Instead, we will explain another new tree construction method that is conceptually simpler and, in practice, faster. But it restricts the possible particle distributions: We force the particles to be placed on a large square grid with a resolution that is polynomial in N. This can be realized by rounding the x, y coordinates of each particle to integers in the range $[0, \mathcal{P}(N)]$, where $\mathcal{P}(N)$ is any whole-numbered polynomial in N of maximum degree s, and by treating pathological cases in which particles have same coordinates efficiently. This bounds the depth of the reduced quadtree to $O(\log(\mathcal{P}(N)) = O(s \cdot \log N) = O(\log N)$. In practice, it is sufficient to set $\mathcal{P}(N) = t \cdot N^2$, with a sufficiently large constant t.

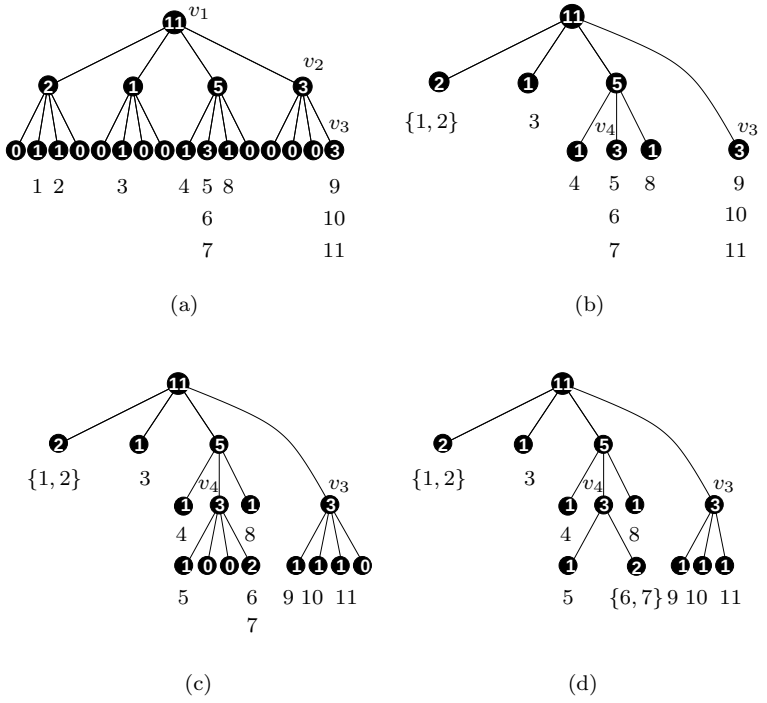


Fig. 5. Building up the reduced quadtree T with leaf capacity $l = 2$ and $N = 11$ particles for the distribution of Figure 4(a). (a) First step: Building up the complete subtree T^1. (b) Second step: Thinning out T^1. (c) Recursion: Building up the complete quadtrees $T^2(v_3)$ and $T^2(v_4)$. (d) Recursion: T is constructed after thinning out $T^2(v_3)$ and $T^2(v_4)$.

Our tree-construction method works as follows: First, we build up a complete truncated subtree T^1 with depth $\max\{1, \lfloor \log N / l \rfloor\}$. Then, all particles are assigned to the leaves of T^1. Since T^1 contains $O(N)$ nodes and its structure is predefined, this step can be performed in linear time. Afterwards, the tree is traversed bottom

up and thereby for each internal tree node the number of particles that are contained in its associated box are calculated. This also needs time linear in N. Figure 5(a) shows an example.

In the next step, we thin out T^1. We traverse the subtree T^1 top down and thereby delete all nodes that do not contain particles and shrink degenerate path to edges. If (during this process) we visit an internal node v that is the root of a subtree which contains at most l particles, this subtree is deleted, and all the particles that were stored in the deleted subtree are assigned to v. Figure 5(b) shows the thinned out subtree T^1.

If none of the leaves of T^1 contains more than l particles, the procedure ends and $T^1 = T$ has been constructed in linear time. Otherwise, we repeat the previous steps recursively. For example, the nodes v_3 and v_4 in Figure 5(b) both contain $3 > l$ particles. Therefore, we build up complete subtrees $T^2(v_3)$ rooted at v_3 and $T^2(v_4)$ rooted at v_4. Both subtrees have depth $\max\{1, \lfloor \log 3/l \rfloor\} = 1$. Now, the particles $5, 6, 7$ are assigned to the leaves of $T^2(v_4)$ and the particles $9, 10, 11$ are assigned to the leaves of $T^2(v_3)$ (see Figure 5(c)). After thinning out $T^2(v_3)$ and $T^2(v_4)$ the desired tree T (see Figure 5(d)) is created.

THEOREM 4.2 TREE CONSTRUCTION. *Suppose, $C = \{c_1, \ldots, c_N\}$ is a set of particles that are placed at distinct positions on a regular square grid with a resolution which is polynomial in N, and $l \geq 1$ is an integer constant. Then, a reduced bucked quadtree with leaf capacity l can be constructed in $O(N \log N)$ worst-case running time.*

PROOF. Building up T^1 needs $O(N)$ time. If T^1 is not the reduced quadtree, we build up subtrees $T^2(v_1), \ldots, T^2(v_k)$ for all leaves $v_1, \ldots, v_k$ of T^1 that contain more than l particles. This needs $O(N)$ time in total, since the sum of the tree nodes contained in all $T^2(v_i)$ is at most $O(N)$. Then, we possibly have to build up subtrees rooted at the leaves of the T^2 trees and so forth. Since for each $j \geq 1$ the sum of the tree nodes of all T^j is bounded above by $O(N)$, the total running time is $O(|\text{recursion\_levels}| \cdot N)$. Therefore, the running time is bounded by $O(N \log N)$. □

Note that the previously described tree-construction method needs only linear time whenever the number of recursion levels is bounded by a constant.

4.2 Phase 2: The Multipole Framework

In this subsection we will concentrate on the second phase of our force-approximation method. Like in Sections 4.1, we suppose that $C = \{c_1, \ldots, c_N\}$ is a set of N charged particles that are located at distinct positions $p(C) = \{p_1, \ldots, p_N\} \in \mathbb{R}^2$.

4.2.1 *Tools From Complex Analysis.* In the following, we will present some definitions and generalizations of theorems that have been invented by [Greengard 1988] and that are of fundamental importance for multipole methods. In order to use tools from complex analysis, each point $p = (x, y) \in \mathbb{R}^2$ is identified with a point $z = x + iy \in \mathbb{C}$.

THEOREM 4.3 MULTIPOLE EXPANSION THEOREM. *Suppose, m charged particles $\{c_1, \ldots, c_m\}$ with charges $\{q_1, \ldots, q_m\}$ are located at points $\{p_1, \ldots, p_m\}$ inside a circle of radius r with center z_0. Then, for any $z \in \mathbb{C}$ with $|z - z_0| > r$ the potential energy $\mathcal{E}(z)$ at point z induced by the m charged particles is given by*

$$\mathcal{E}(z) = a_0 \log(z - z_0) + \sum_{k=1}^{\infty} \frac{a_k}{(z - z_0)^k} \text{ , where}$$

$$a_0 = \sum_{i=1}^{m} q_i \quad \text{and} \quad a_k = \sum_{i=1}^{m} \frac{-q_i (p_i - z_0)^k}{k} \, .$$

Based on this theorem, the idea is to develop the infinite series only up to a constant index p. The resulting truncated Laurent series $M^p(z)$ is called *p-term multipole expansion*. In practice, choosing $p = 4$ has turned out to be sufficient to keep the error of the approximation less than 10^{-2}.

The following Lemma 4.4 shows how the center of a multipole expansion can be shifted. Lemma 4.5 describes how a multipole expansion can be converted into a power series (that is called *local expansion*) in a circular region of analyticity, and Lemma 4.6 shows how the center of a finite local expansion can be shifted. The finite power series $L^p(z)$ that is obtained by calculating only the coefficients 0 to p of a local expansion is called is called *p-term local expansion*.

LEMMA 4.4 TRANSLATION OF MULTIPOLE EXPANSIONS. *Suppose, $\mathcal{E}(z) = a_0 \log(z - z_0) + \sum_{k=1}^{\infty} \frac{a_k}{(z-z_0)^k}$ is a multipole expansion of the potential energy due to a set of charged particles that are located inside a circle of radius r and center z_0. Then, for any z outside a circle of radius $r + |z_0 - z_1|$ and center z_1, the potential energy induced by these particles is given by*

$$\mathcal{E}(z) = a_0 \log(z - z_1) + \sum_{l=1}^{\infty} \frac{b_l}{(z - z_1)^l} \text{ , where}$$

$$b_l = \frac{-a_0 (z_0 - z_1)^l}{l} + \sum_{k=1}^{l} a_k (z_0 - z_1)^{l-k} \binom{l-1}{k-1} \, .$$

LEMMA 4.5 CONVERSION OF MULTIPOLE EXPANSIONS. *Suppose, C_0 is a set of charged particles that are located inside a circle of radius r and center z_0, the corresponding multipole expansion is given by $\mathcal{E}(z) = a_0 \log(z - z_0) + \sum_{k=1}^{\infty} \frac{a_k}{(z-z_0)^k}$, and that z_1 is a point with $|z_1 - z_0| > 2r$. Then, inside a circle of radius r and center z_1 the potential energy due to these particles is given by the power series*

$$\mathcal{E}(z) = \sum_{l=0}^{\infty} b_l \cdot (z - z_1)^l \text{ , where}$$

$$b_0 = a_0 \log(z_1 - z_0) + \sum_{k=1}^{\infty} \frac{a_k}{(z_1 - z_0)^k} \quad \text{and}$$

$$b_l = \frac{(-1)^{l+1} a_0}{(z_1 - z_0)^l \cdot l} + \left(\frac{1}{z_0 - z_1} \right)^l \sum_{k=1}^{\infty} \binom{l+k-1}{k-1} \frac{a_k}{(z_1 - z_0)^k} \text{ , for } l \geq 1 \, .$$

LEMMA 4.6 TRANSLATION OF LOCAL EXPANSIONS. *For any complex z_0, z_1, z and $\{a_0, \ldots, a_p\}$*

$$\sum_{k=0}^{p} a_k (z - z_0)^k = \sum_{l=0}^{p} \left(\sum_{k=l}^{p} a_k \binom{k}{l} (z_1 - z_0)^{k-l} \right) (z - z_1)^l .$$

Since we are more interested in approximating the forces in the system rather than the potential energy, the following Lemma can be used to obtain the forces that act in a potential energy field which is described by a multipole expansion or local expansion.

LEMMA 4.7 OBTAINING THE FORCES. *(a) If $\mathcal{E}(z) = a_0 \log(z - z_0) + \sum_{k=1}^{\infty} \frac{a_k}{(z-z_0)^k}$ describes the potential energy field in a system of charged particles, and z is contained in a region of analyticity, then, the forces that act on a particle of unit charge at position z are given by $(Re(\mathcal{E}'(z)), -Im\ (\mathcal{E}'(z)))$ with $\mathcal{E}'(z) = \frac{a_0}{z-z_0} - \sum_{k=1}^{\infty} \frac{k \cdot a_k}{(z-z_0)^{k+1}}$.*

(b) Suppose, $\mathcal{E}(z) = \sum_{l=0}^{\infty} b_l \cdot (z - z_1)^l$ describes the potential energy field in a system of charged particles, and z is contained in a region of analyticity. Then, the forces that act on a particle of unit charge at position z are given by $(Re(\mathcal{E}'(z)), -Im(\mathcal{E}'(z)))$ with $\mathcal{E}'(z) = \sum_{l=1}^{\infty} l \cdot b_l (z - z_1)^{l-1}$.

Remark 4.8. Due to the stated Multipole Expansion Theorem 4.3 and lemmas, it is easy to confirm the following statements (assuming all operations are applied on well-defined sets):

(a) The coefficients of a p-term multipole expansion due to m charged particles can be obtained in $O(pm)$ time.

(b) The coefficients of a shifted p-term multipole expansion can be obtained in $O(p^2)$ time.

(c) The coefficients of a p-term local expansion can be obtained from the coefficients of a p-term multipole expansion in $O(p^2)$ time.

(d) The coefficients of a shifted p-term local expansion can be obtained in $O(p^2)$ time.

(e) The derivative of a p-term multipole expansion and the derivative of a p-term local expansion can be obtained in $O(p)$ time.

(f) An approximation of the force that acts on a particle of unit charge at a position z — which is induced by the potential energy field that is described by a p-term multipole expansion or a p-term local expansion — can be obtained in $O(p)$ time.

We will demonstrate how p-term multipole expansions can be used to speed up force calculations in systems of charged particles by giving an example: Suppose, m particles of unit charge are located within a circle C_0 of radius r with center z_0 and that another m particles of unit charge are located within a circle C_1 of radius r with center z_1, and let $|z_0 - z_1| > 2r$ (see Figure 6).

Computing the repulsive forces acting on each particle in C_0 due to all particles in C_1 naively would need $\Theta(m^2)$ time. Now, suppose that we first compute the coefficients of a p-term multipole expansion of the potential energy due to the

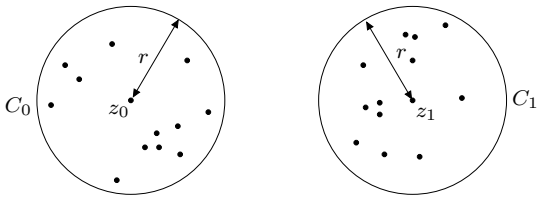


Fig. 6. An example distribution showing the use of p-term multipole expansions.

particles in C_1. This needs $O(pm)$ time (see Remark 4.8(a)). Then, we calculate the derivative of the p-term multipole expansion in $O(p)$ time (see Remark 4.8(e)) before evaluating it for each particle in C_0. This needs $m \cdot O(p)$ time (see Remark 4.8(f)). Hence, the total running time for approximating the forces is $O(pm)$, which is significantly faster than the naive approach if $m >> p$.

4.2.2 *Some Terminology.* In Sections 4.1 we have associated each node v of the reduced bucket quadtree with its box $box(v)$. In the following, we will associate two boxes with a node v that are defined next. Figures 7(a) and 7(b) explain this terminology at an example.

Definition 4.9 *(Small Cell, Large Cell).* Suppose, T is a reduced bucket quadtree with fixed constant leaf capacity l for a given set $C = \{c_1, \ldots, c_N\}$ of distinct particles that are distributed in a square D, and v is a node of T. The *small cell* or *small box* of v (shorter $Sm(v)$) is the smallest sub-box of D that covers all particles that are associated with v. If v is a leaf that contains only one particle, then, $Sm(v)$ is a point. If v is the root of T, the *large cell* or *large box* of v (shorter $Lg(v)$) is equal to $Sm(v)$. Otherwise, let parent(v) be the parent of v in T. Then, $Lg(v)$ is the largest sub-box of $Sm(\text{parent}(v))$ with size smaller than $Sm(\text{parent}(v))$ that covers $Sm(v)$.

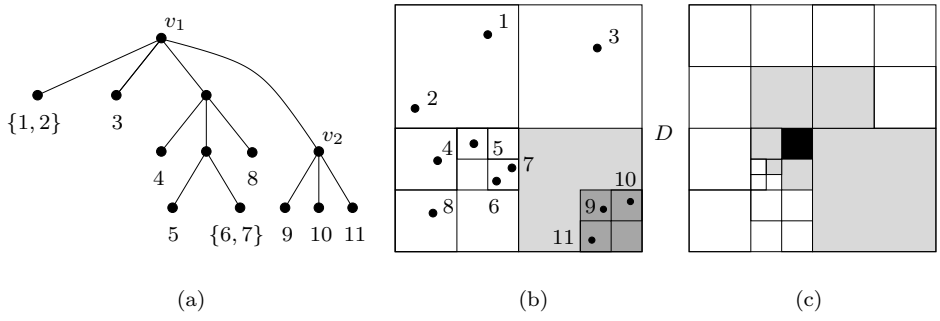


Fig. 7. (a) A reduced bucket quadtree that corresponds to the particle distribution in the square D shown in (b). The small cell $Sm(v_2)$ corresponds to the small dark-gray box. The small cell of $parent(v_2) = v_1$ corresponds to the square D. The large cell $Lg(v_2)$ corresponds to the big light-gray box that covers the small dark-gray box in (b). (c) Unlike the white boxes, the gray boxes are neighbors of the black box.

Remark 4.10. It follows from the definition of $Sm(v)$, $Lg(v)$, and the reduced bucket quadtree T that the small cell $Sm(v)$ of an interior node v is exactly box(v). For a leaf v of T, $Sm(v)$ is covered by box(v). Furthermore, $Lg(v) \setminus Sm(v)$ covers no particles of C, and the size of $Lg(v)$ is $\frac{1}{4}$ of the size of Sm(parent(v)) if v is not the root of T. Figure 7(a,b) shows an example.

The Multipole Expansion Theorem 4.3 and the lemmas for shifting, converting, and evaluating these expansions can only be applied in well-defined regions of analyticity. Since in all quadtree data structures the regions are squares, the terminology of *well-separateness* is used to indicate that the operations of Theorem 4.3 and the lemmas for working with these expansions can be applied.

Definition 4.11 *(Neighbor, Well-Separated, Ill-Separated).* Two boxes B_1 and B_2 are called *neighbors* if the boundaries of B_1 and B_2 touch, but B_1 and B_2 do not overlap. Two boxes B_1 and B_2 of same size are *well-separated* if they are no neighbors. Otherwise, B_1 and B_2 are *ill-separated*. Suppose, nodes u and v are nodes of a reduced bucket quadtree T and $Sm(u) \geq Sm(v)$. Then, u and v are *well-separated* if and only if $Sm(u)$ and the cell that covers $Sm(v)$ and that has the same size as $Sm(u)$ are well-separated. Otherwise u and v are *ill-separated*. Suppose, $Sm(u) < Sm(v)$. Then, u and v are *well-separated* if and only if $Sm(v)$ and the cell that covers $Sm(u)$ and that has the same size as $Sm(v)$ are well-separated. Otherwise u and v are *ill-separated*.

Figure 7(c) shows a box B and its neighbors, while Figure 8(a) and (b) show the small cells of two tree nodes that are well-separated and ill-separated, respectively.

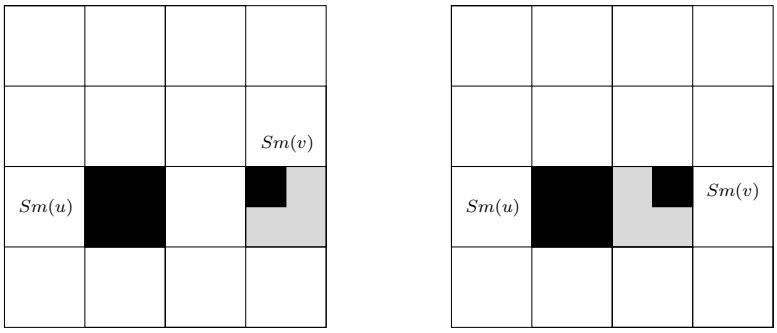


Fig. 8. The black boxes $Sm(u)$ and $Sm(v)$ are small cells of two nodes u and v of a reduced bucked quadtree. (a) Since the gray box is no neighbor of $Sm(u)$, u and v are well-separated. (b) Since the gray box is a neighbor of $Sm(u)$, u and v are ill-separated.

Like in [Greengard 1988; Aluru et al. 1998] the terminology of well-separateness is used to define an interaction set that is used do generate p-term local expansions from suitable p-term multipole expansions. The following definition of an

interaction set of a node v of T has been invented in [Aluru et al. 1998] and is a generalization of the definition of the interaction set defined in [Greengard 1988].

Definition 4.12 *(Interaction Set, Minimal Ill-Separated Set).* Suppose, nodes u and v are nodes of a reduced (bucket) quadtree T. The *interaction set* $I(v)$ of a node v is the set of all nodes u that are ill-separated from the parent of v, well-separated from v, and the parent of u is ill-separated from v. Thus, $I(v) = \{u \mid$ `Well_Separated`(v,u), `Ill_Separated` $(\text{parent}(v),u)$, `Ill_Separated`$(\text{parent}(u), v)\}$. The *minimal ill-separated set* $R(v)$ of a node v is the set of all nodes that are ill-separated from v and have the small cell smaller or equal and the large cell larger or equal than the small cell of v. Hence, $R(v) = \{u \mid$ `Ill_Separated`$(v,u), Sm(u) \leq Sm(v)) \leq Lg(u)\}$.

The multipole framework of Aluru et al. [Aluru et al. 1998] associates the lists $I(v)$ and $R(v)$ with each node v of the reduced quadtree in order to approximate the repulsive forces in the system. Since we use the reduced bucket quadtree data structure, the leaves possibly contain more than one particle. We will see that this generalization can be modeled in our framework by defining some additional sets which are assigned to each node of the tree and that are introduced next.

Definition 4.13 *(The Sets $D_1(v)$, $D_2(v)$, $D_3(v)$ and $K(v)$).* For each node v of the reduced bucket quadtree $T = (V,E)$, $D_1(v)$ is the set of all leaves $w \in V$ so that $Sm(v) < Sm(w)$ and $Sm(v)$ and $Sm(w)$ are neighbors. $D_2(v)$ is the set of all leaves w of T so that $Sm(v) < Sm(w)$, $Sm(v)$ and $Sm(w)$ are no neighbors, v and w are ill-separated, and w is not contained in the set $D_2(u)$ of an ancestor u of v. For each leaf $v \in V$ we additionally define the sets $D_3(v)$ and $K(v)$, where $D_3(v)$ is the set of all leaves $w \in V$ so that $Sm(v) \geq Sm(w)$ and $Sm(v)$ and $Sm(w)$ are neighbors. For a leaf $v \in V$ the set $K(v)$ contains all nodes w so that $Sm(v)$ and $Sm(w)$ are no neighbors. Additionally, it is required for each $w \in K(v)$ that either $w \in R(v)$ or an ancestor of w is contained in $R(v)$ and $Sm(\text{parent}(w))$ and $Sm(v)$ are neighbors.

4.2.3 *Formal Description of the Multipole Framework.* The multipole framework is based on a bottom-up traversal and a top-down traversal of the reduced bucket quadtree data structure with a fixed constant precision parameter p and can be seen as a generalization of the multipole framework of [Aluru et al. 1998]. The pseudocode of the framework is shown in Function `Multipole_Framework` and Function `Calculate_Local_Expansions_and_Node_Sets`, and its parts are explained in the following.

Part 1: (Trivial Case and Initializations) If the reduced bucket quadtree $T = (V,E)$ consists of only one node, the forces that act on each particle p_i due to all other particles in C are calculated directly, and the algorithm terminates. Otherwise, for each node v of T the sets $R(v), I(v), D_1(v), D_2(v)$ (and for the leaves additionally $D_3(v)$, and $K(v)$) are initialized.

Part 2: (Bottom-Up Traversal of the Tree) Now, for each leaf v of T the coefficients $\{a_0, \ldots, a_p\}$ of the p-term multipole expansion $M^p(v)$ that reflects an approximation of the potential energy field due to the particles that are contained in $Sm(v)$ are calculated. This is done by using Theorem 4.3 and choosing the center

Function `Multipole_Framework`(C, T, p)

input : a set $C = \{c_1, \ldots, c_N\}$ of charged particles of unit charge that are placed at distinct positions $p(C) = \{p_1, \ldots, p_N\}$, a reduced bucket quadtree $T = (V, E)$ with fixed constant leaf capacity l that is associated with C, and a fixed constant precision parameter p

output: a function $F_{rep} : C \longrightarrow \mathbb{R}^2$ so that $F_{rep(c_i)}$ is an approximation of the repulsive forces that act on c_i due to all other particles in C

begin
  begin {`Part 1: Trivial Case and Initializations`}
    if T contains only one node **then**
      foreach $c_i \in C$ **do**
        $F_{rep}(c_i) \leftarrow$ `Naive_Direct_Force_Calculation`;
      `Exit`;
    foreach $v \in V$ **do** $R(v) \leftarrow I(v) \leftarrow D_1(v) \leftarrow D_2(v) \leftarrow \emptyset$;
    foreach leaf $v \in V$ **do** $D_3(v) \leftarrow K(v) \leftarrow \emptyset$;
  end
  begin {`Part 2: Bottom-Up Traversal`}
    foreach leaf $v \in V$ **do**
      calculate the coefficients of $M^p(v)$ due to all particles contained in $Sm(v)$;
    foreach interior node $v \in V$ for which coefficients of $M^p(w)$ of all children w of v have been calculate **do**
      calculate the coefficients of $M^p(v)$ due to all particles contained in $Sm(v)$ by adding coefficients of shifted $M^p(w)$ of all children w of v;
  end
  begin {`Part 3: Top-Down Traversal`}
    foreach child v of the root of T **do**
      `Calculate_Local_Expansions_and_Node_Sets`(T, C, p, v);
  end
  begin {`Part 4: Obtain the Forces`}
    foreach leaf $v \in V$ **do**
      foreach particle $c_i \in Sm(v)$ **do**
        let $C(v)$ be the set of charged particles that are contained in $Sm(v)$;
        calculate $F_{local}(c_i)$ using the coefficients of $L^p(v)$;
        calculate $F_{direct}(c_i)$ using $D_1(v) \cup D_3(v) \cup C(v)$;
        calculate $F_{multipole}(c_i)$ using $K(v)$;
        $F_{rep}(c_i) \leftarrow F_{local}(c_i) + F_{direct}(c_i) + F_{multipole}(c_i)$;
  end
end

Function `Calculate_Local_Expansions_and_Node_Sets`(T, C, p, v)

input : C, T, p are defined like in Function `Multipole_Framework`, a node v of T

output: the coefficients $\{b_0, \ldots, b_p\}$ of the p-term local expansions of v, the sets $R(v)$, $I(v)$, $D_1(v)$, $D_2(v)$, and additionally $D_3(v)$ and $K(v)$ if v is a leaf

begin

  begin {`Part 3.1: Find` $R(v), I(v), D_1(v)$`, and` $D_2(v)$}

    if *parent*(v) is the root of T **then** $E(v) \leftarrow$ *parent*(v);

    else $E(v) \leftarrow R($*parent*$(v)) \cup D_1($*parent*$(v))$;

    while $E \neq \emptyset$ **do**

      pick some $u \in E(v)$, and set $E(v) \leftarrow E(v) \setminus \{u\}$;

      if `Well_Separated`(u, v) **then** $I(v) \leftarrow I(v) \cup \{u\}$;

      else if $Sm(v) \geq Sm(u)$ **then** $R(v) \leftarrow R(v) \cup \{u\}$;

      else if v is no leaf of T **then** $E(v) \leftarrow E(v) \cup$ *children*(u);

      else if `Neighbors`$(Sm(u), Sm(v))$ **then** $D_1(v) \leftarrow D_1(v) \cup \{u\}$;

      else $D_2(v) \leftarrow D_2(v) \cup \{u\}$;

  end

  begin {`Part 3.2: Calculate Coefficients of` $L^p(v)$}

    foreach $u \in I(v)$ **do**

      convert $M^p(u)$ to $L^p(u)$, and add coefficients of $L^p(u)$ to coefficients of $L^p(v)$;

    foreach $u \in D_2(v)$ **do**

      foreach $c_i \in Sm(u)$ **do**

        calculate $M^p(c_i)$, convert $M^p(c_i)$ to $L^p(c_i)$, and add coefficients of $L^p(c_i)$ to coefficients of $L^p(v)$;

    if coefficients of $L^p($*parent*$(v))$ have been calculated **then**

      add coefficients of shifted $L^p($*parent*$(v))$ to $L^p(v)$;

  end

  begin {`Part 3.3: Find` $D_3(v)$ `and` $K(v)$ `for Leaves`}

    if v is a leaf of T **then**

      set $E(v) \leftarrow R(v)$;

      while $E(v) \neq \emptyset$ **do**

        pick some $u \in E(v)$, and set $E(v) \leftarrow E(v) \setminus \{u\}$;

        if not `Neighbors`$(Sm(u), Sm(v))$ **then** $K(v) \leftarrow K(v) \cup \{u\}$;

        if `Neighbors`$(Sm(u), Sm(v))$ and u is leaf **then** $D_3(v) \leftarrow D_3(v) \cup \{u\}$;

        else $E(v) \leftarrow E(v) \cup$ *children*(u);

  end

  begin {`Part 3.4: Recursion`}

    if v is no leaf of T **then foreach** child w of v **do**

      `Calculate_Local_Expansions_and_Node_Sets`(T, C, p, w);

  end

end

of $Sm(v)$ as the variable z_0 in Theorem 4.3.

The p-term multipole expansions of the interior nodes are calculated by traversing the tree bottom up: Suppose, v is an interior node of T with small cell $Sm(v)$, the center of $Sm(v)$ is z_1, and the p-term multipole expansions $M^p(w)$ of each of v's children w have been calculated. Then, the coefficients of the p-term multipole expansion $M^p(v)$ that reflects an approximation of the potential energy field due to the particles that are contained in $Sm(v)$ are obtained by first shifting the center of each $M^p(w)$ to z_1 (using Lemma 4.4) and, then, adding the coefficients of this shifted p-term multipole expansion to the corresponding coefficients of $M^p(v)$.

Part 3: (Top-Down Traversal of the Tree) In the top-down traversal of the tree T, Function `Calculate_Local_Expansions_and_Node_Sets` is called for each child of the root node.

Part 3.1: (Find $R(v)$, $I(v)$, $D_1(v)$, and $D_2(v)$) Here, the sets $R(v)$, $I(v)$, $D_1(v)$, and $D_2(v)$ are constructed starting with a set $E(v)$ that is assigned $R(parent(v)) \cup D_1(parent(v))$. Note that if $parent(v)$ is the root of T, it is clear that $R(parent(v)) = v$ and $D_1(parent(v)) = \emptyset$. Then, iteratively a node u is taken from $R(parent(v)) \cup D_1(parent(v))$, and it is checked if u already belongs to one of the previous mentioned sets or if one has to explore the subtrees that are rooted at u recursively in order to assign its children to these sets.

In particular, if u and v are well-separated, then, u belongs to $I(v)$. [This can be seen as follows: If u is a node of $R(parent(v)) \cup D_1(parent(v))$, then since u and v are well-separated $u \in R(parent(v))$. Hence, $parent(v)$ and u are ill-separated by definition of $R(parent(v))$. $Lg(u) \geq Sm(parent(v))$ by definition of $R(parent(v))$ and, hence, $parent(u)$ and v are ill-separated, too. If in the complementary case, u is a proper ancestor of a node in $R(parent(v)) \cup D_1(parent(v))$, u is a proper ancestor of a node $R(parent(v))$ since the nodes in $D_1(parent(v))$ are leaves. Hence, $parent(v)$ and u are ill-separated. Furthermore, $parent(u)$ and v are ill-separated, since otherwise $parent(u) \in I(v)$.]

If u and v are ill-separated and additionally $Sm(v) \geq Sm(u)$, then, u belongs to $R(v)$. [To see this, we have to prove that $Lg(u) \geq Sm(v)$: Suppose, $u \in R(parent(v)) \cup D_1(parent(v))$, then, it follows from definition of $D_1(parent(v))$ that $u \in R(parent(v))$. Therefore, $Lg(u) \geq Sm(parent(v))$ and, hence, $Lg(u) \geq Sm(v)$. In the complementary case, u is a proper ancestor of a node in $R(parent(v))$ by definition of $D_1(parent(v))$. In this case $Lg(u) \geq Sm(v)$ holds since $Sm(parent(u)) > Sm(v)$ by the dynamic construction of $E(v)$.]

If u and v are ill-separated, $Sm(v) < Sm(u)$, u is a leaf, and $Sm(u)$ and $Sm(v)$ are neighbors, then, u is contained in $D_1(v)$.

If u and v are ill-separated, $Sm(v) < Sm(u)$, u is a leaf, and $Sm(u)$ and $Sm(v)$ are no neighbors, then, u is contained in $D_2(v)$, since $R(parent(v)) \cup D_1(parent(v))$ and $D_2(parent(v))$ are disjoint and u is an element of $R(parent(v)) \cup D_1(parent(v))$ or an ancestor of an element in $R(parent(v)) \cup D_1(parent(v))$.

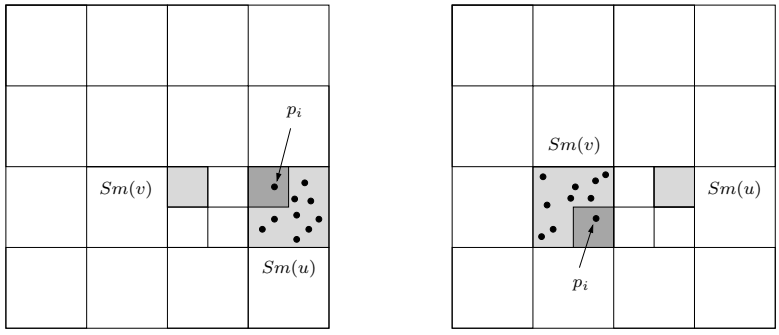
In the remaining case $Sm(v) < Sm(u)$ and u is an interior node of T that is ill-separated from v. Hence, one has to check whether its children belong to one of the sets $R(v)$, $I(v)$, $D_1(v)$, or $D_2(v)$.

Note that by construction of part 3.1 the particles that are covered by the small cells of all nodes in $R(parent(v)) \cup D_1(parent(v))$ are exactly the particles that are

covered by the small cells of all nodes in $R(v) \cup I(v) \cup D_1(v) \cup D_2(v)$. Furthermore, each such particle is covered by the small cell of exactly one node in $R(v) \cup I(v) \cup D_1(v) \cup D_2(v)$.

Part 3.2: (Calculate Coefficients of $L^p(v)$) Like in other multipole methods, for each u in the interaction set $I(v)$ the coefficients of p-term multipole expansions $M^p(u)$ are converted to coefficients of p-term local expansions $L^p(u)$ and added to the corresponding coefficients of $L^p(v)$ using Lemma 4.5 and choosing $z_0 := \mathit{center}(Sm(u))$ and $z_1 := \mathit{center}(Sm(v))$.

Some difficulties arise for the nodes in $D_2(v)$: Since each $u \in D_2(v)$ is a leaf that is ill-separated from v with $Sm(v) < Sm(u)$, one cannot apply Lemma 4.5 on u. However, we found another way to calculate the local expansions due to the particles that are contained in u: Let $\{c_1, \ldots, c_k\}$ be the set of charged particles that are contained in $Sm(u)$ at positions $\{p_1, \ldots, p_k\}$. First, we calculate the coefficients of the p-term multipole expansion $M^p(c_i)$ for each single particle c_i using Theorem 4.3 and choosing $z_0 := p_i$. Since we can interpret each point p_i as a dimensionless box B_i, the largest cell that covers B_i and that has the same size as $Sm(v)$ is no neighbor of $Sm(v)$ due to the definition of $D_2(v)$ (see Figure 9(a)). Hence, $Sm(v)$ and B_i are well-separated, and Lemma 4.5 can be used to convert $M^p(c_i)$ to a p-term local expansion $L^p(c_i)$ choosing $z_0 := p_i$ and $z_1 := \mathit{center}(Sm(v))$. Finally, for each $c_i \in \{c_1, \ldots, c_k\}$ the coefficients of $L^p(c_i)$ are added to $L^p(v)$.



(a) (b)

Fig. 9. (a) The light-gray boxes $Sm(v)$ and $Sm(u)$ are ill-separated, no neighbors, and $Sm(v) < Sm(u)$. $Sm(v)$ and the dark-gray sub-box of $Sm(u)$ that contains particle c_i at position p_i and has the same size as $Sm(v)$ are well-separated. (b) The light-gray boxes $Sm(v)$ and $Sm(u)$ are ill-separated, no neighbors, and $Sm(v) > Sm(u)$. $Sm(u)$ and the dark-gray sub-box of $Sm(v)$ that contains particle c_i at position p_i and has the same size as $Sm(u)$ are well-separated.

Following standard practice, the coefficients of the shifted p-term local expansion L^p $(\mathit{parent}(v))$ of the parent of v are added to the corresponding coefficients of $L^p(v)$. Thus, an approximation of the potential energy field of the region that is reflected by $L^p(\mathit{parent}(v))$ is inherited to v using Lemma 4.6 and choosing the center of $Sm(v)$ as z_1.

As a result of part 3.2, the coefficients of $L^p(v)$ reflect an approximation of the potential energy field due to all particles contained in the small cells of the nodes in $I(v)$, $D_2(v)$, and in the $I(u)$ and $D_2(u)$ sets of all ancestors of v. Furthermore, the small cells of the nodes in $R(v) \cup D_1(v)$ are exactly the regions that have not been considered yet in the calculation of the potential energy in $Sm(v)$ due to all particles in the region $D \setminus Sm(v)$.

Part 3.3: (Find $D_3(v)$ and $K(v)$ for Leaves) Suppose, v is a leaf of T and u is a node in $R(v)$. By definition of $R(v)$ we know that u and v are ill-separated and that $Sm(v) \geq Sm(u)$. Three cases can arise: If u and v are no neighbors, u is an element of $K(v)$. If u and v are neighbors and u is a leaf, u is an element of $D_3(v)$. Otherwise, u is a neighbor of v but an interior node of T. Thus, we can recursively assign the children of u to either $K(v)$ or $D_3(v)$, since all children of u are ill-separated from v, their small cell is smaller than $Sm(v)$, and $Sm(u)$ is a neighbor of $Sm(v)$.

Note that the particles that are covered by the small cells of the nodes in $R(v)$ are exactly the particles that are covered by the small cells of the nodes in $K(v) \cup D_3(v)$. Furthermore, each such particle is covered by the small cell of exactly one node in $K(v) \cup D_3(v)$.

Part 3.4: (Recursion) Suppose, v is an interior node of T. The small cells of the nodes in $R(v) \cup D_1(v)$ are exactly the regions that have not been considered yet in the calculation of the potential energy in $Sm(v)$ due to all particles that are contained in the region $D \setminus Sm(v)$. Hence, we can inherit the sets $R(v)$ and $D_1(v)$ to each child w of v and call Function `Calculate_Local_Expansions_and_Node_Sets` for w.

As a result of parts 1 to 3 we have given the coefficients of the p-term local expansions $L^p(v)$ for each leaf v of T, and $L^p(v)$ reflects an approximation of the potential energy field induced by the particles that are not covered by the small cells of all nodes contained in $D_1(v) \cup D_3(v) \cup K(v) \cup Sm(v)$.

Part 4: (Obtain the Forces) Suppose, v is a leaf of T so that $Sm(v)$ contains the particles $\{c_1, \ldots, c_k\}$, which are placed at positions $\{p_1, \ldots, p_k\}$. We can obtain an approximation of the repulsive forces that act on each $c_i \in \{c_1, \ldots, c_k\}$ due to all particles in the system $C = \{c_1, \ldots, c_N\}$ as follows:

Let L' be the derivative of $L^p(v)$. Lemma 4.7(b) can be used to to obtain the forces that are induced by L' and act on c_i by setting $F_{local}(c_i) = (Re(L'(p_i)), -Im(L'(p_i)))$.

The forces that act on c_i due to all particles that are contained in $Sm(v) \cup \{Sm(u) \mid u \in D_1(v) \cup D_3(v)\}$ are calculated directly by a naive exact force calculation, since v and all nodes $u \in D_1(v) \cup D_3(v)$ are leaves. The resulting forces are denoted by $F_{direct}(c_i)$.

We only have to concentrate on the particles that are covered by the small cells of the nodes in $K(v)$. Let u be a node in $K(v)$. Since u is ill-separated from v and $Sm(v) > Sm(u)$, we cannot apply Lemma 4.5 in order to convert $M^p(u)$ to $L^p(u)$ and to add the coefficients of $L^p(u)$ to the corresponding coefficients of $L^p(v)$. However, we can use a similar trick as the trick that we have invented in part 3.2: Let c_i be a particle that is placed at position $p_i \in Sm(v)$. Since we can

interpret p_i as a dimensionless box B_i, the largest cell that contains c_i and that has the same size as $Sm(u)$ are no neighbors due to the definition of $K(v)$ (see Figure 9(b)). Hence, $Sm(u)$ and B_i are well-separated, and Lemma 4.7(a) can be used in order to obtain the forces that act on c_i due to all particles contained in $Sm(u)$. In particular, let M' be the derivative of $M^p(u)$. Then, the force on c_i that is induced by $M^p(u)$ is given by $F_{multipole}(c_i) = (Re(M'(p_i)), -Im(M'(p_i)))$.

Therefore, the approximation of the repulsive force that acts on a particle c_i due to all particles in $C = \{c_1, \ldots, c_N\}$ is given by the sum of $F_{direct}(c_i)$, $F_{local}(c_i)$, and $F_{multipole}(c_i)$.

4.2.4 *The Running Time of the Multipole Framework.* In order to prove that the running time of Function `Multipole_Framework` is linear in the number of particles, we need the following lemma.

LEMMA 4.14 SIZES OF THE SETS $R(v)$, $I(v)$, $D_1(v)$, $D_2(v)$, $D_3(v)$, $K(v)$. *Suppose, $T = (V, E)$ is a reduced bucket quadtree with constant leaf capacity l that is associated with a set $C = \{c_1, \ldots, c_N\}$ of N charged particles that are placed at distinct positions $\{p_1, \ldots, p_N\}$, and leaves(T) is the set of the leaves of T. Then,*

$$\sum_{v \in V} |R(v)| = O(N), \tag{1}$$

$$\sum_{v \in V} |I(v)| = O(N), \tag{2}$$

$$\sum_{v \in V} |D_1(v)| = O(N), \tag{3}$$

$$\sum_{v \in V} |D_2(v)| = O(N), \tag{4}$$

$$\sum_{v \in \text{leaves}(T)} |D_3(v)| = O(N), \text{ and} \tag{5}$$

$$\sum_{v \in \text{leaves}(T)} |K(v)| = O(N). \tag{6}$$

PROOF. The proofs of Equations (1) and (2) are similar to the corresponding proofs in [Aluru et al. 1998]. We prove Equation (1) first. Let v be an arbitrary node of T. Each node $u \in R(v)$ is ill-separated from v and $Sm(v) \geq Sm(u)$. Hence, $Sm(u)$ is either covered by $Sm(v)$ or covered by a neighbor box B of $Sm(v)$ that has the same size as $Sm(v)$. In the first case, we know from the definition of $R(v)$ that $Sm(v) \leq Lg(u)$. By definition of $Lg(u)$ it follows that $u = v$. In the second case, by using that $Sm(v) \leq Lg(u)$ and Remark 4.10, we know that $Lg(u) \setminus Sm(u)$ contains no particles. Hence, $B \setminus Sm(u)$ contains no particles, too. Since $Sm(v)$ has exactly 8 neighbor boxes B of equal size, we get that $\sum_{v \in V} |R(v)| \leq (1+8) \cdot |V| = O(|V|) = O(N)$.

We now prove Equation (2). Suppose, w is a node in $I(v)$, then, either w is in $R(parent(v))$ or an ancestor u of w is in $R(parent(v))$ or w is in $D_1(parent(v))$ or an ancestor u of w is in $D_1(parent(v))$. The last two options can be excluded by

the fact that all elements of $D_1(parent(v))$ are neighboring leaves of $Sm(parent(v))$ with a small cell that is larger than $Sm(parent(v))$ and, hence, are ill-separated from v and have no ancestors. Therefore,

$$\begin{aligned}\sum_{v\in V} |\{w \mid w \in I(v)\}| \;=\; &\sum_{v\in V} |\{w \mid w \in I(v), w \in R(parent(v))\}| + \\ &\sum_{v\in V} |\{w \mid w \in I(v), w \notin R(parent(v))\}|.\end{aligned}$$

By Equation (1) we know that $\sum_{v\in V} |\{w \mid w \in I(v), w \in R(parent(v)\}| = O(N)$. We can rewrite the second term of this equation as follows:

$$\sum_{v\in V} |\{w \mid w \in I(v), w \notin R(parent(v))\}| = \sum_{w\in V} |\{v \mid w \in I(v), w \notin R(parent(v))\}|$$

Hence, in order to show that Equation (2) holds, it is sufficient to show that $S(w) := |\{v \mid w \in I(v), w \notin R(parent(v))\}| = O(1)$. Let w be arbitrarily fixed. Since $w \notin R(parent(v))$ there exists a node $u \in R(parent(v))$ that is an ancestor of w. Thus, $parent(v)$ and $parent(w)$ are ill-separated, and $Sm(parent(v)) \geq Sm(parent(w))$. It can be shown that there exists a box B that is a neighbor of box $Sm(parent(w))$, B has the same size as $Sm(parent(w))$, and B is a sub-box of $Sm(parent(v))$ that contains $Sm(v)$. [To see this: $Sm(v)$ cannot be larger than $Sm(parent(w))$, since — by definition of $I(v)$ — v and $parent(w)$ are ill-separated but v and w are well-separated. Suppose, $Sm(v)$ is not covered by a box B that is a neighbor of $Sm(parent(w))$ and that has the same size as $Sm(parent(w))$, then, v and $parent(w)$ are well-separated, which contradicts the fact that $w \in I(v)$ by definition of $I(v)$. Finally, $Sm(parent(v))$ covers B, since $Sm(parent(v)) \geq Sm(parent(w))$.] For each such box B there exist at most 4 sub-boxes like $Sm(v)$ since $Sm(parent(v))$ covers B. Since for each w, the number of neighbor boxes of equal size is bounded above by 8, $|S(w)|$ is bounded above by $4 \cdot 8 = 32$, which completes the proof of Equation (2).

The proof of Equation (3) is easy. Let v be an arbitrary node of T. Since there exist at most 8 boxes that are neighbors of $Sm(v)$ and that have the same size as $Sm(v)$, there exist less than 8 leaves w that are neighbors of v with $Sm(w) > Sm(v)$. Hence, $\sum_{v\in V} |D_1(v)| < 8|V| = O(N)$.

Next, we prove Equation (4). Let us suppose that the leaf capacity l is 1 and that v is a node of T. Then, the small cells of all leaves have size zero. By definition of $D_1(v)$ and $D_2(v)$ it follows that $D_1(v) = D_2(v) = \emptyset$. Thus, by Equations (1) and (2) for $l = 1$ the equality $\sum_{v\in V} |R(v)|+|I(v)|+|D_1(v)|+|D_2(v)| = O(N)$ holds. Furthermore, the elements of $R(parent(v))\cup D_1(parent(v))$ or the descendants of the elements of $R(parent(v)) \cup D_1(parent(v))$ are partitioned into the disjoint subsets $R(v)$ and $I(v)$ (since $D_1(v) = D_2(v) = \emptyset$). Now, let us suppose that $l > 1$. Then, the nodes contained in $D_2(v)$ are either elements of the set $R(parent(v))\cup D_1(parent(v))$ or are descendants of the elements of $R(parent(v))\cup D_1(parent(v))$. In the first case, the number of elements of $D_2(v)$ that are contained in $R(parent(v))\cup D_1(parent(v))$ is bounded above by $O(N)$ using Equations (1) and (3). In the second case, we use that by Equations (1), (2), and (3) $\sum_{v\in V} |R(v)| + |I(v)| + |D_1(v)| = O(N)$. Hence, it is sufficient to show that $\sum_{v\in V} |R(v)| + |I(v)| + |D_1(v)| + |D_2(v)| = O(N)$. This can be shown as follows: Suppose, leaf u is an ancestor of a node in $R(parent(v)) \cup D_1(parent(v))$, contains k particles with $2 \leq k \leq l$, and is neither assigned to $I(v)$ nor to $R(v)$. Then, u is assigned to either $D_1(v)$ or $D_2(v)$ since it has no further

descendants in T. Hence, the cardinality of the actual set $R(v) \cup I(v) \cup D_1(v) \cup D_2(v)$ is increased by one. In contrast to this, in the case that $l = 1$ the subtree rooted at u would have been explored further. Since in this case T is a reduced quadtree, this exploration would result in adding at least two nodes to $R(v) \cup I(v)$. Therefore, in the case that $l > 1$ the expression $\sum_{v \in V} |R(v)| + |I(v)| + |D_1(v)| + |D_2(v)|$ is bounded above by $O(N)$, too.

In order to prove Equation (5), we can use that $\sum_{v \in leaves(T)} |\{w \mid Sm(v) \geq Sm(w), leaf(w)\}| = \sum_{w \in leaves(T)} |\{v \mid Sm(v) \geq Sm(w), leaf(w)\}|$. It is sufficient to show that for each leaf w the set $S'(w) := |\{v \mid Sm(v) \geq Sm(w), leaf\ (w)\}|$ is bounded by a constant. Since for each such w at most 8 boxes exist that are neighbors of $Sm(w)$ and have the same size as $Sm(w)$, we get that $|S'(w)| \leq 8$.

Finally, we prove Equation (6). By definition of $K(v)$, we know that for each leaf v of the tree $K(v) = K_1(v) \cup K_2(v)$ so that $K_1(v)$ is the set of all nodes w with $Sm(v)$ and $Sm(w)$ are no neighbors and $w \in R(v)$. $K_2(v)$ is the set of all nodes w so that $Sm(v)$ and $Sm(w)$ are no neighbors, $Sm(parent(w))$ and $Sm(v)$ are neighbors, and an ancestor of w is contained in $R(v)$. Hence, we get that $\sum_{v \in leaves(T)} |\{w \mid w \in K(v)\}| = \sum_{v \in leaves(T)} |\{w \mid w \in K_1(v)\}| + \sum_{v \in leaves(T)} |\{w \mid w \in K_2(v)\}|$.

The first term is bounded by $O(N)$ using Equation (1). To estimate the second term, we use that

$$\sum_{v \in leaves(T)} |\{w \mid w \in K_2(v)\}| = \sum_{w \in leaves(T)} |\{v \mid w \in K_2(v)\}|.$$

Let $S''(w) := |\{v \mid w \in K_2(v)\}|$. Then, it is sufficient to show that $|S''(w)|$ is bounded by a constant for an arbitrary node w of T. Suppose, there exists a leaf v of T so that $w \in K_2(v)$. Then, $Sm(parent(w))$ and $Sm(v)$ are neighbors. Since $parent(w)$ or an ancestor of $parent(w)$ is contained in $R(v)$, it is clear that $Sm(v) \geq Sm(parent(w))$. Since at most 8 such leaves v of size larger or equal than $Sm(parent(w))$ are neighbors of $Sm(parent(w))$, we get that $|S''(w)| \leq 8$, which completes the proof. □

THEOREM 4.15 MULTIPOLE FRAMEWORK. *Suppose, $C = \{c_1, \ldots, c_N\}$ is a set of charged particles of unit charge that are placed at distinct positions $p(C) = \{p_1, \ldots, p_N\}$, $T = (V, E)$ is a reduced bucket quadtree with fixed constant leaf capacity l that is associated with C, and p is a fixed constant precision parameter. Then, Function* `Multipole_Framework` *approximates the repulsive force that acts on each particle c_i due to all other particles in C in $O(N)$ time.*

PROOF. If T contains only one node, $|C| = N \leq l$. Since l is a constant, the exact naive force calculation in part 1 needs constant time. Otherwise, the initialization of the sets in part 1 needs $O(|V|) = O(N)$ time.

In part 2 of Function `Multipole_Framework` the coefficients of the p-term multipole expansions $M^p(v)$ are calculated for all leaves v, using Theorem 4.3. Let $m(v)$ denote the number of particles that are contained in $Sm(v)$ for each leaf v of T. By Remark 4.8(a) this needs $\sum_{v \in leaves(v)} O(p \cdot m(v)) = O(p \cdot N) = O(N)$ time. The coefficients of the p-term multipole expansions of the interior nodes are obtained using Lemma 4.4. By Remark 4.8(b) and the fact that $|V| = O(N)$ the total running time of this step is $O(p^2 \cdot |V|) = O(N)$.

In part 3.1 the sets $R(v)$, $I(v)$, $D_1(v)$, and $D_2(v)$ are found for a fixed node v. This is done by exploring a collection of $|R(parent(v)) \cup D_1(parent(v))|$ rooted subtrees, where each node in $R(parent(v)) \cup D_1(parent(v))$ is a root. The nodes in $R(parent(v)) \cup D_1(parent(v))$ are either assigned to one of the sets $R(v)$, $I(v)$, $D_1(v)$, and $D_2(v)$ directly, or their children are examined later. Since T is a reduced bucket quadtree, each interior node has at least two children. Hence, the total number of nodes that are visited in the exploration of all subtrees of the nodes in $R(parent(v)) \cup D_1(parent(v))$ is proportional to $|R(v)| + |I(v)| + |D_1(v)| + |D_2(v)|$. It follows from Lemma 4.14 that applying parts 3.1 to all nodes v of T needs $O(N)$ time.

In part 3.2 for each node $u \in I(v)$ converting the coefficients of the p-term multipole expansion $M^p(u)$ to the p-term local expansion $L^p(u)$ and adding these coefficients to the corresponding coefficients of p-term local expansion $L^p(v)$ of v can be done in $O(p^2)$ time using Lemma 4.5 (see Remark 4.8(c)). For each $u \in D_2(v)$ there exists at most l particles c_i that are contained in $Sm(u)$. For each such particle the work needed to calculate $M^p(c_i)$, to convert it to $L^p(c_i)$, and to add its coefficients to the corresponding coefficients of $L^p(v)$ is $O(p^2)$ using Theorem 4.3 and Lemma 4.5 (see Remarks 4.8(a) and (c)). Adding the coefficients of the shifted p-term local expansions $L^p(parent(v))$ of the parent of a fixed node v to $L^p(v)$ can be done in $O(p^2)$ time using Lemma 4.6 (see Remark 4.8(d)). Since we know from Lemma 4.14 that $\sum_{v \in V} I(v) \cup D_2(v) = O(N)$, $O(l \cdot p^2 \cdot N) = O(N)$ time is needed to apply part 3.2 on all nodes $v \in V$.

In part 3.3 the sets $D_3(v)$ and $K(v)$ are constructed by exploring a set of $|R(v)|$ rooted subtrees with roots in $R(v)$. The total number of nodes that are visited in the exploration of all subtrees that are rooted at the nodes in $R(v)$ is proportional to $|D_3(v)| + |K(v)|$. Using Lemma 4.14, applying part 3.3 on all leaves v of T needs $O(N)$ time in total.

In part 4 the forces $F_{direct}(c_i)$, $F_{local}(c_i)$, and $F_{multipole}(c_i)$ are calculated for each particle c_i that is contained in a leaf v of T. The calculation of $F_{local}(c_i)$ needs $O(p)$ time using Lemma 4.7(b) (see Remark 4.8(e) and (f)). Calculating $F_{direct}(c_i)$ can be done in $O(l \cdot (|D_1(v)| + |D_3(v)| + 1))$ time. Calculating $F_{multipole}(c_i)$ can be done in $O(p \cdot |K(v)|)$ time using Lemma 4.7(a) (see Remark 4.8(e) and (f)). Adding $F_{direct}(c_i)$, $F_{local}(c_i)$, and $F_{multipole}(c_i)$ to obtain $F_{rep}(c_i)$ needs $O(1)$ time. Hence, it follows from Lemma 4.14 that the running time of part 4 is bounded by $\sum_{v \in leaves(T)} \sum_{c_i \in Sm(v)} O(p + l \cdot (|D_1(v)| + |D_3(v)| + 1) + p \cdot |K(v)| + 1) = O(N)$. □

5. RUNNING TIME OF FM³

COROLLARY 5.1 MAIN THEOREM. *Suppose, $G = (V, E)$ is a positive weighted graph. Then, the* Fast Multipole Multilevel Method (FM³) *generates a straight-line drawing of G in $O(|V| \log |V| + |E|)$ worst-case running time. Its best-case running time is $O(|V| + |E|)$.*

PROOF. The corollary follows directly form Theorems 3.2, 4.2, 4.15 and the previous discussion. □

6. EXPERIMENTAL RESULTS

FM^3 has been implemented in C++ within the framework of *AGD* [Jünger et al. 2004]. We tested the algorithm on a 2.8 GHz PC running Linux. The tested graphs are the graphs contained in [Walshaw's graph collection] with up to 200000 nodes and the biggest graphs from the [AT&T graph collection]. Furthermore, we generated artificial graphs with up to 100000 nodes. These graphs include grid graphs, sierpinski graphs, trees, random disconnected graphs, graphs that contain many biconnected components, graphs with a very high edge density, and graphs that contain nodes with very high degrees. Figure 10 shows the running times of FM^3 for the real-world and artificially generated graphs. All graphs with less than 1000, 10000, and 100000 nodes have been drawn in less than 2, 24, and 263 seconds, respectively.

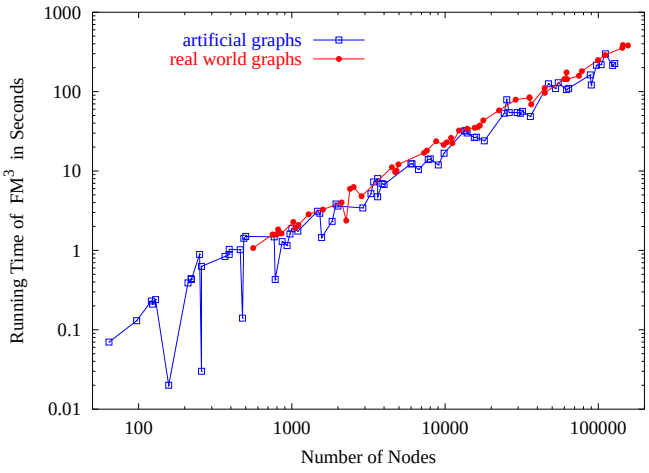


Fig. 10. The running times of FM^3 for drawing all artificial graphs and all real-world graphs.

Figure 11 shows example drawings that are generated by FM^3 with standard-parameter settings. The practical experiments indicate that FM^3 generates well-structured drawings for the majority of the tested artificial and real-world graphs: Like other force-directed multilevel methods, it generates nice drawings of regular well connected and almost planar graphs (see Figure 11(a,b,c,d). But even the structure of more challenging graphs like disconnected graphs, graphs with a high edge density or graphs that contain high degree nodes (see Figure 11(e,f,g,h)) is visualized in an appropriate concise way.

This is a clear improvement in comparison with several other state-of-the-art graph-drawing methods and will be illustrated by an example (see Figure 12): The test-graph (*snowflake_A*) is a symmetric tree which consists of 971 nodes and one central root node that has 256 neighbors. Besides FM^3, we tested a classical force directed algorithm (the *grid-variant algorithm* (GVA) of [Fruchterman and Reingold 1991]) and two multilevel algorithms (GRIP of [Gajer and Kobourov 2001] and [Gajer et al. 2001], and the *Fast Multi-scale Method* (FMS) of [Harel and Koren

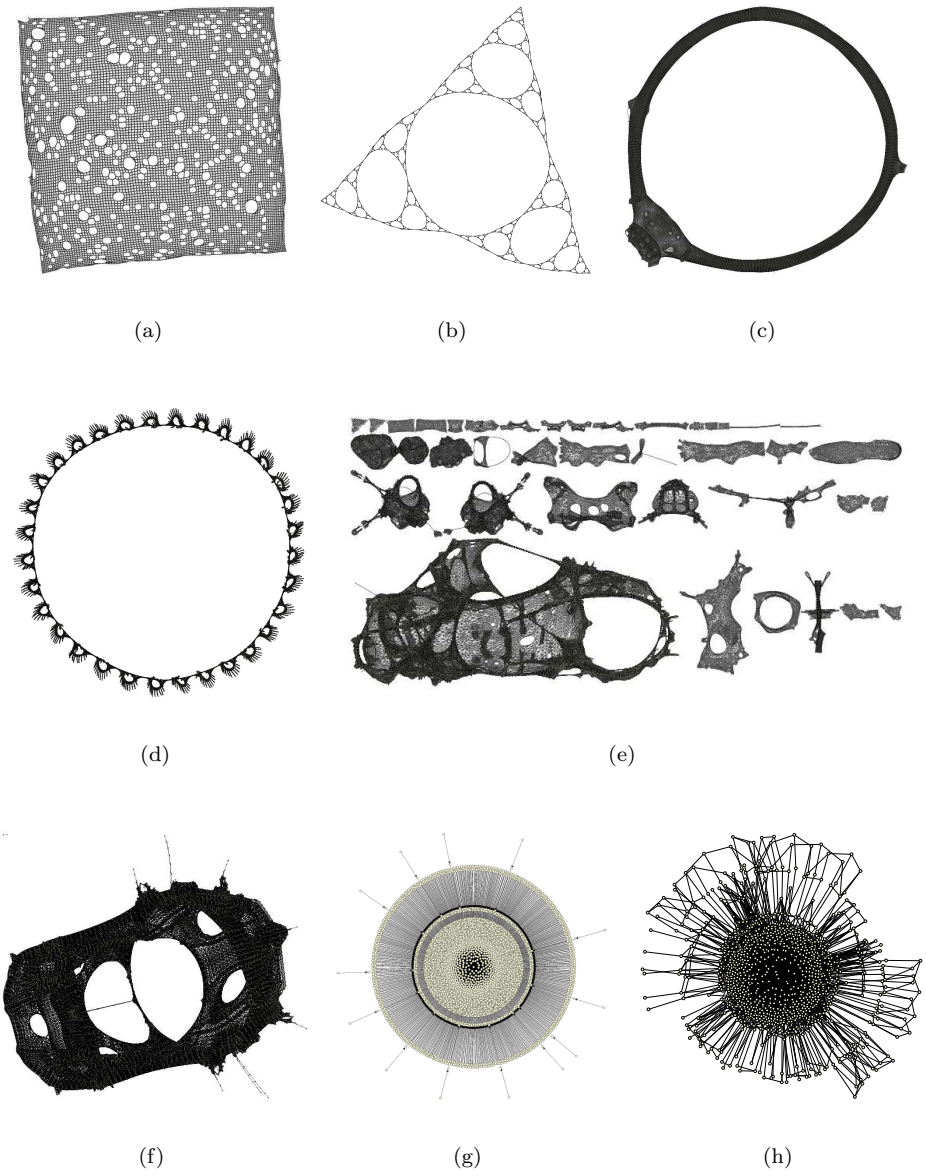
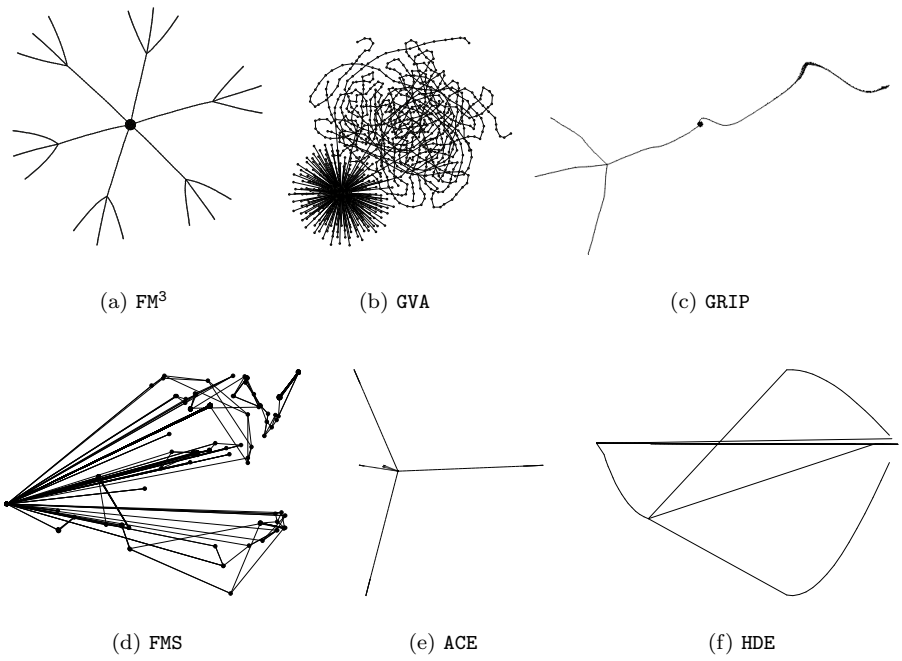


Fig. 11. (a) *grid_rnd_100*: $|V| = 9497$, $|E| = 17849$, CPU-time = 19.1 seconds. (b) *sierpinski_10*: $|V| = 88575$, $|E| = 177147$, CPU-time = 162.0 seconds. (c) *fe_pwt*: $|V| = 36463$, $|E| = 144794$, CPU-time = 69.0 seconds. (d) *finan512*: $|V| = 74752$, $|E| = 261120$, CPU-time = 158.2 seconds. (e) *fe_body*: $|V| = 44775$, $|E| = 163734$, CPU-time = 96.5 seconds. (c) *bcsstk31*: $|V| = 35588$, $|E| = 572914$, edge density = 16.1, CPU-time = 83.6 seconds. (f) *dg_1087*: $|V| = 7602$, $|E| = 7601$, maximum degree = 6566, CPU-time = 18.1 seconds. (g) *ug_380*: $|V| = 1104$, $|E| = 3231$, maximum degree = 856, CPU-time = 2.1 seconds.

2001]). Additionally, we compared FM^{3}with algebraic graph-drawing methods (the algebraic multigrid method ACE of [Koren et al. 2003] and the *high-dimensional embedding* approach (HDE) of [Harel and Koren 2002]).

An extensive experimental comparison of these graph-drawing algorithms can be found in [Hachul and Jünger 2005].



(a) FM^3 (b) GVA (c) GRIP

(d) FMS (e) ACE (f) HDE

Fig. 12. (a)-(f) Drawings of snowflake_A generated by different algorithms.

7. SUMMARY AND FUTURE WORK

We have developed a new force-directed graph-drawing algorithm (FM^3) that runs in $O(|V| \log |V| + |E|)$ time. This is an improvement in comparison with previous force-directed approaches that are not sub-quadratic in general. The improvement has been reached by combining a new multilevel scheme and a generalized strategy for approximating the repulsive forces in the system by rapidly evaluating potential fields.

The practical experiments demonstrate that FM^3 is very fast and creates nice drawings of even those graphs that turned out to be challenging for other tested algorithms. Currently, FM^3 is integrated into the software package TULIP [Auber 2004], and it will be newly implemented for a commercial software package in the near future.

REFERENCES

Aluru, S. et al. 1998. Distribution-Independent Hierarchical Algorithms for the N-body Problem. *Journal of Supercomputing 12*, 303–323.

AT&T graph collection. `www.graphdrawing.org`.

Auber, D. 2004. *Graph Drawing Software*. Mathematics and Visualization, vol. XII. Springer-Verlag, Chapter TULIP – A Huge Graph Visualization Framework, 105–126.

Barnes, J. and Hut, P. 1986. A hierarchical $\mathcal{O}(N \log N)$ force-calculation algorithm. *Nature 324,* 4, 446–449.

Davidson, R. and Harel, D. 1996. Drawing Graphs Nicely Using Simulated Annealing. *ACM Transaction on Graphics 15,* 4, 301–331.

Eades, P. 1984. A heuristic for graph drawing. *Congressus Numerantium 42*, 149–160.

Fruchterman, T. and Reingold, E. 1991. Graph Drawing by Force-directed Placement. *Software–Practice and Experience 21,* 11, 1129–1164.

Gajer, P. et al. 2001. A Multi-dimensional Approach to Force-Directed Layouts of Large Graphs. In *Graph Drawing 2000*, J. Marks, Ed. Lecture Notes in Computer Science, vol. 1984. Springer-Verlag, 211–221.

Gajer, P. and Kobourov, S. 2001. GRIP: Graph Drawing with Intelligent Placement. In *Graph Drawing 2000*, J. Marks, Ed. Lecture Notes in Computer Science, vol. 1984. Springer-Verlag, 222–228.

Greengard, L. 1988. *The Rapid Evaluation of Potential Fields in Particle Systems*. ACM distinguished dissertations. The MIT Press, Cambridge, Massachusetts.

Hachul, S. 2005. A Potential-Field-Based Multilevel Algorithm for Drawing Large Graphs. Ph.D. thesis, Institut für Informatik, Universität zu Köln, Germany. `http://kups.ub.uni-koeln.de/volltexte/2005/1409`.

Hachul, S. and Jünger, M. 2005. An Experimental Comparison of Algorithms for Drawing Large Graphs. Tech. Rep. zaik2004-472, Institut für Informatik, Universität zu Köln. `www.zaik.uni-koeln.de/~paper`, to appear in P. Healy (ed.) *Graph Drawing 2005*, Lecture Notes in Computer Science, Springer-Verlag, 2006.

Harel, D. and Koren, Y. 2001. A Fast Multi-scale Method for Drawing Large Graphs. In *Graph Drawing 2000*, J. Marks, Ed. Lecture Notes in Computer Science, vol. 1984. Springer-Verlag, 183–196.

Harel, D. and Koren, Y. 2002. Graph Drawing by High-Dimensional Embedding. In *Graph Drawing 2002*. Lecture Notes in Computer Science, vol. 2528. Springer-Verlag, 207–219.

Jünger, M. et al. 2004. *Graph Drawing Software*. Mathematics and Visualization, vol. XII. Springer-Verlag, Chapter AGD - A Library of Algorithms for Graph Drawing, 149–169.

Kamada, T. and Kawai, S. 1989. An Algorithm for Drawing General Undirected Graphs. *Information Processing Letters 31*, 7–15.

Koren, Y. et al. 2003. Drawing Huge Graphs by Algebraic Multigrid Optimization. *Multiscale Modeling and Simulation 1,* 4, 645–673.

Quigley, A. and Eades, P. 2001. FADE: Graph Drawing, Clustering, and Visual Abstraction. In *Graph Drawing 2000*, J. Marks, Ed. Lecture Notes in Computer Science, vol. 1984. Springer-Verlag, 197–210.

Tunkelang, D. 1998. JIGGLE: Java Interactive Graph Layout Environment. In *Graph Drawing 1998*, S. H. Whitesides, Ed. Lecture Notes in Computer Science, vol. 1547. Springer-Verlag, 413–422.

Walshaw, C. 2001. A Multilevel Algorithm for Force-Directed Graph Drawing. In *Graph Drawing 2000*, J. Marks, Ed. Lecture Notes in Computer Science, vol. 1984. Springer-Verlag, 171–182.

Walshaw's graph collection. `www.gre.ac.uk/~c.walshaw/partition`.