A Scalable Algorithm for Simulating the Structural Plasticity of the Brain

Sebastian Rinke*, Markus Butz-Ostendorf†, Marc-André Hermanns‡, Mikaël Naveau†§, Felix Wolf*
*Technische Universität Darmstadt, Germany
Email: {rinke, wolf}@cs.tu-darmstadt.de
†Simulation Laboratory Neuroscience, Jülich Aachen Research Alliance, Forschungszentrum Jülich, Germany
‡Jülich Aachen Research Alliance, Section JARA-HPC, Forschungszentrum Jülich, Germany
§INSERM UMR-S U919, GIP CYCERON, Caen Normandy University, France

Abstract—**The neural network in the brain is not hard-wired. Even in the mature brain, new connections between neurons are formed and existing ones are deleted, which is called structural plasticity. The dynamics of the connectome is key to understanding how learning, memory, and healing after lesions such as stroke work. However, with current experimental techniques even the creation of an exact static connectivity map, which is required for various brain simulations, is very difficult. One alternative is to use simulation based on network models to predict the evolution of synapses between neurons, based on their specified activity targets. This is particularly useful as experimental measurements of the spiking frequency of neurons are more easily accessible and reliable than biological connectivity data. The Model of Structural Plasticity (MSP) by Butz et al. is an example of this approach. However, to predict which neurons connect to each other, the current MSP model computes probabilities for all pairs of neurons, resulting in a complexity $\mathbf{O(n^2)}$. To enable large-scale simulations with millions of neurons and beyond, this quadratic term is prohibitive. Inspired by hierarchical methods for solving n-body problems in particle physics, we propose a scalable approximation algorithm for MSP that reduces the complexity to $\mathbf{O(n \log^2 n)}$ without any notable impact on the quality of the results. An MPI-based parallel implementation of our scalable algorithm can simulate neuron counts that exceed the state of the art by two orders of magnitude.**

I. Introduction

The brain is not as hard-wired as traditionally thought. Neurons are connected to each other in a dynamically changing biological network of synapses, also known as the connectome. Even in the mature brain, new connections between neurons (i.e., synapses) are continuously created and existing ones are deleted, which can be described as structural plasticity. Studying the dynamics of connectivity in the brain is fundamental to understanding how learning, memory, and healing after lesions in the brain such as strokes work. Unfortunately, accurately observing the connectome and its evolution empirically is very hard. Limiting factors are, for example, the resolution of sensors and restricted access to the brain areas of interest [1]. Thus, even creating an exact connectivity map of a small region of the brain is extremely challenging. However, it is exactly such a connectivity map that is needed as the basis of state-of-the-art brain simulations [2], [3].

An alternative to acquiring biological connectivity data is to determine the connections between neurons using a network model. For example, when the spiking frequency of a neuron is too low, it starts to form more synapses, with the aim of increasing its electrical activity. Conversely, synapses are deleted when the electrical activity of a participating neuron is too high. One big advantage of the spiking frequency is that it is easier to observe experimentally than the connectome itself. In addition to generating static connectivity maps, such a network model can also help investigate the dynamics of connectivity, such as (i) structural plasticity in a cell-type-dependent manner [4], (ii) the creation of structures due to external stimuli [5], and (iii) functional reorganization and restructuring after a lesion [6], [7].

The Model of Structural Plasticity (MSP) by Butz et al. [8] is a network model with activity-dependent dynamic creation and deletion of synapses. In traditional models, connectivity is fixed while plasticity merely arises from changes in the strength of existing synapses, typically modeled as weight factors. MSP, in contrast, is suitable for simulating the reorganization of the connectome. Instead of representing a synapse by a weight factor, MSP models a synapse as a connection between an axonal "plug" and a dendritic "socket". These synaptic elements grow and shrink independently on each neuron. When an axonal element of one neuron connects to the dendritic element of another neuron, a new synapse is formed. Conversely, when a synaptic element bound in a synapse retracts, the corresponding synapse is removed. The governing idea of the model is that plasticity in cortical networks is driven by the need of individual neurons to homeostatically maintain their average electrical activity. Consequently, neurons form new synaptic elements if their activity is below a desired threshold, and remove elements if it exceeds the threshold. As empirical observation shows, MSP lets networks of neurons robustly grow towards a stable homeostatic equilibrium of activity and connectivity. It was shown that this structural-plasticity rule can account for network rewiring after a partial loss of external input (deafferentation) [8]. The simulation results exhibited strong similarities with biological data from network rewiring in the primary visual cortex after focal retinal lesions [6], [7]. To make MSP available to a larger community and combine its capabilities with the features of a state-of-the-art brain simulator, the model was recently integrated [1] into the NEST neural network simulator [2].

Although a parallel implementation of MSP with MPI has already been able to successfully simulate the structural plasticity of up to 10^5 neurons with NEST [1], the computational complexity of MSP in terms of the number of neurons seriously limits its scalability. To decide which pairs of axonal and dendritic elements will form a synapse, MSP follows a probabilistic approach. It considers all pairs of neurons with a vacant axonal element on one side of the pair and a vacant dendritic element on the other, and calculates the probability of them establishing a connection between them. The shorter their distance, the higher this probability becomes. Given that every neuron creates a certain amount of both axonal and dendritic elements (limited by a constant due to biological restrictions), ultimately all pairs of neurons have to be considered. Thus, the cost grows quadratically ($O(n^2)$) with the number of neurons. However, as soon as we start investigating the connectivity across individual brain regions and the number of neurons involved rises above a hundred thousand, this cost becomes prohibitive. Note that the human brain has 10^{11} neurons. For this reason, we urgently need a scalable algorithm for MSP.

A similar challenge arises in n-body problems, where pairs of bodies have to be considered for force calculations. To improve the scalability of the force calculations, powerful approximation methods have been developed [9], [10]. They are based on the observation that particles sufficiently far away from a target particle do not need to be considered individually. It is our goal to leverage their underlying ideas and adapt them to the problem of structural brain plasticity. The most influential algorithms are Barnes-Hut [9] and the Fast Multipole Method [10] (FMM). However, they cannot be applied to our problem directly. They calculate the force exerted on (Barnes-Hut) or the potential of (FMM) each body, whereas we need to select pairs of neurons (bodies) for synapse creation. Moreover, n-body simulations continuously subject each particle to force calculations. In the brain, after an initial network creation phase, only a small subset of neurons exhibits vacant axonal elements. Thus, vacant dendrites only have to be found for this smaller subset.

In this paper, we present a scalable approximation method for simulating structural plasticity based on MSP. Our algorithm, an adaptation of Barnes-Hut, reduces the complexity of MSP from $O(n^2)$ to $O(n \log^2 n)$. We further show that the approximations of our method are still precise enough to resemble neural networks created by the original MSP. An MPI-based parallel implementation of our scalable algorithm is the first to enable the model-based creation of neural networks consisting of up to 10^7 neurons—with the potential for far greater problem sizes.

II. Related Work

Today's largest brain simulations contain about 10^9 neurons. C2 [3] and NEST [11] are examples of state-of-the-art brain simulators able to reach such a large scale. Both require the user to describe the connectivity between neurons before the simulation starts. During the simulation, the connectivity map remains static. However, the strength of the synapses it defines may change over time. Well-known models that strive to capture structural plasticity include the compensation model by Dammasch et al. [12] and the activity-dependent neurite outgrowth model by van Ooyen et al. [13]. However, while the compensation model ignores topology altogether, van Ooyen's model is too restrictive in that neurons always connect to their direct neighbors before connecting to more distant neurons. These limitations are addressed in the Model of Structural Plasticity [8], the subject of this paper, where synapses are randomly created in a distance-dependent way.

An example for using n-body simulation in brain research has been presented by Prasad et al. [14], where cortical brain regions are represented as particles with mass proportional to the region's volume. Particles attract each other with a force proportional to the strength of the connectivity between the regions they represent. The connectivity between regions was derived from diffusion imaging data from patients with Alzheimer's disease and healthy subjects. Based on these parameters, the authors use a gravitational n-body simulation to obtain a connectivity matrix between brain regions. This matrix is then examined with the goal of distinguishing between patients and healthy subjects. Our work differs from this approach in that we do not perform an n-body simulation. Instead, we adopt ideas of hierarchical n-body methods to reduce the complexity of a structural plasticity model.

The concepts of force between particles and distance-dependent probability for pairs of neurons are similar enough to make the adaptation of n-body methods [9], [10] to our problem a realistic option. Another motivation is that the data locality and approximation of advanced n-body methods seem to better mimic biological behavior in the brain. In particular, while "actively" trying to find a vacant dendrite, a neuron's vacant axon has only partial knowledge of other available neurons. Our choice of n-body methods for adaptation is the Barnes-Hut algorithm [9], a decision we will outline in Section IV. An example showing the good scalability of the Barnes-Hut approach for n-body problems is the PEPC code [15], which has already been used to efficiently simulate systems with about $64 \cdot 10^9$ bodies on $458,752$ cores of an IBM Blue Gene/Q system.

III. The MSP Model of Structural Plasticity

This section describes the Model of Structural Plasticity [8], which consists of three basic steps to simulate network connectivity in an activity-dependent fashion: (i) update of electrical activity, (ii) update of synaptic elements, and (iii) update of connectivity.

1) Update of electrical activity: The electrical activity of each neuron is continuously calculated on a millisecond timescale. Intracellular calcium concentration is updated according to the electrical activity. As calcium concentration and average firing rate are linearly proportional, the model uses calcium concentration to guide the growth dynamics of the synaptic elements.

2) Update of synaptic elements: The detailed morphology of synaptic elements is abstracted and represented only by the number of synaptic contacts on axons (axonal boutons) and dendrites (dendritic spines). We call these contacts collectively synaptic elements. A homeostatic rule determines for each neuron when axonal and dendritic synaptic elements are created or deleted. If the calcium concentration is below the desired set point, they are created. If it is above the set point, they are deleted. Creation or deletion proceeds until the desired level of electrical activity has been reached.

3) Update of connectivity: At discrete points in time, existing synapses are deleted and new synapses are formed, depending on the current number of synaptic elements. A synapse is deleted after either the participating axonal or dendritic element has been removed during the update of synaptic elements. If a synapse is removed, a synaptic element that was previously bound in this synapse becomes vacant again. If a *source* neuron with a vacant axonal element is assumed, then the *target* neuron which the axonal element will try to connect to is determined by considering every neuron as a potential target and calculating the probability of establishing a connection. The probability depends on the distance between source and target, and the number of unbound dendritic elements available at the target. Given the three-dimensional position (x, y, z) of a source neuron j and a target neuron candidate i, we can evaluate a three-dimensional Gaussian-shaped kernel:

$$K_{ij} = \exp\left(-\frac{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}{\sigma^2}\right) \quad (1)$$

σ is a simulation parameter that controls the width of the curve. To avoid creating autapses (i.e., source neuron connecting to itself), we set $K_{ij} = 0$ for $i = j$. K_{ij} is then weighted by a factor w_i denoting the number of vacant dendritic elements at the target neuron. This yields $|Neurons|$ (number of neurons) values of the form $\{w_i \cdot K_{ij} \mid j \text{ is source neuron} \land i \in Neurons\}$ for the source neuron j. The sum of the elements in this set is not necessarily 1. To construct probabilities, all the elements are finally scaled so that their sum equals 1. Finally, a random number in the interval $[0, 1]$ selects the target neuron out of all candidates. Constructing the probabilities in this way ensures that the closer the two neurons are, and the more dendrites the target neuron candidate offers, the higher the probability for the target neuron candidate to be chosen for synapse formation is. During the connectivity update, every vacant axonal element selects a target neuron, as described above. Note that multiple axonal elements may try to connect to the same target neuron. If the neuron does not have sufficient vacant dendritic elements, some of the axonal elements are rejected. In this case, they try to find another target neuron during the next connectivity update. The extent of changes in the neural network is determined by the update of synaptic elements. As these elements grow rather slowly, the connectivity update occurs only infrequently.

Performance considerations: Steps 1) and 2) consider every neuron and thus run in $O(n)$. In step 3), every neuron is examined to decide which of its synapses have to be deleted. As the number of synapses per neuron is limited through a constant (due to biological reasons), synapse deletion runs in

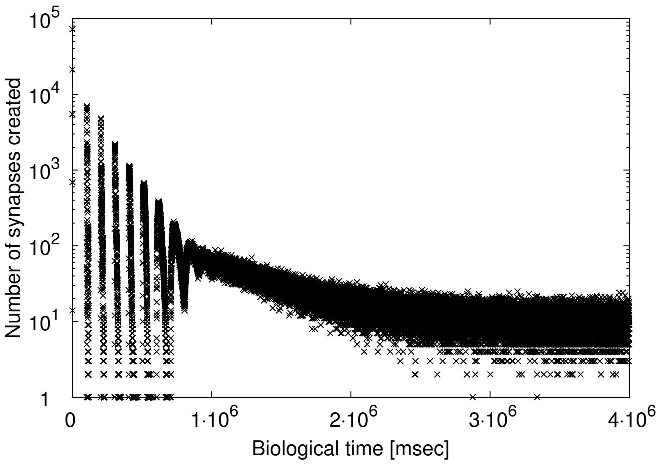


Fig. 1. The number of synapses MSP creates over time for 10^5 neurons. To save simulation time, every neuron is initialized with one vacant axonal element and two vacant dendritic elements. Otherwise, we would have to wait until synaptic elements have grown to form synapses. At the beginning, no synapses exist and the network is empty. Neurons start forming synapses to reach their desired level of electrical activity. From $3 \cdot 10^6$ msec on, the neurons enter equilibrium and only a few synapses are being formed. Note that some synapses are also deleted as the simulation progresses, which is not shown here.

$O(n)$. However, for synapse creation in step 3), probabilities are calculated for all pairs of neurons, which takes $O(n^2)$. This worst case occurs in the early phase of network creation where no synapses exist yet and all neurons still have vacant axonal elements available. Figure 1 depicts this situation. Note that the number of synapses created during the first $5 \cdot 10^5$ msec is in the order of the number of neurons and much higher than during the remainder of the simulation. The peaks in the plot appear because we apply the same growth curve to all synaptic elements. That is, after all axonal elements have been bound in synapses during connectivity updates, new axonal elements grow and become available for all neurons at about the same time. Especially if major structural changes occur, that is, at the beginning or after introducing lesions, synapse creation prevents MSP from being scaled to large neural networks. Thus, reducing the complexity of MSP's synapse creation is a prerequisite for simulating larger portions of the brain.

IV. A Scalable Algorithm for MSP

We shall now describe our scalable approximation algorithm for MSP, an adaptation of the Barnes-Hut n-body method to our specific problem. Although the $O(n)$ complexity of FMM is lower than the $O(n \log n)$ complexity of Barnes-Hut, we chose Barnes-Hut because FMM is harder to tailor to our needs and to implement. One noteworthy difference is that FMM groups not only target but also source particles, whereas we need to group only target neurons, which more closely matches Barnes-Hut. Moreover, we anticipate that the number of neurons in the human brain (10^{11}) is the largest problem that our algorithm will ever be required to handle efficiently, which means that the superior scalability of FMM may ultimately not be needed. For these reasons, we believe that following the design philosophy of the simpler Barnes-

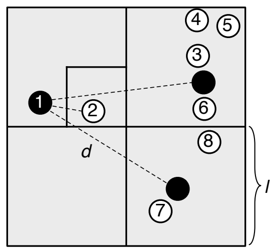


Fig. 2. A two-dimensional example of grouping neurons. Neurons 3-6 and 7-8 are in the same respective squares. They form two groups. The length of their squares is denoted by l, which can be seen as the spatial extent of the group. The two groups are represented by a virtual neuron (black solid circle). The source neuron is neuron 1. Instead of considering all individual neurons as target neuron candidates, grouping reduces the work at this stage to considering only two virtual neurons and two normal neurons (neuron 2 and the source neuron itself). To avoid autapses, the source neuron's probability is set to zero. A dashed line depicts the distance d from the source neuron to the other neurons under consideration, which are neuron 2 and the two virtual neurons.

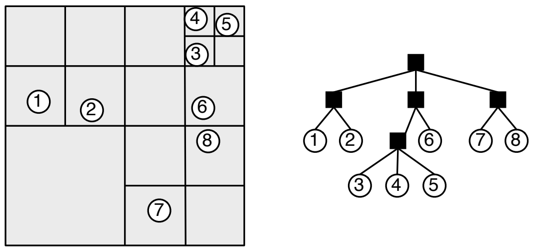


Fig. 3. A two-dimensional tree-construction example. On the left, we see the subdivision of the simulation domain. On the right, we see the resulting tree. Neurons are numbered from 1 to 8. The inner nodes in the tree are virtual neurons, while the leaves are real neurons belonging to their subdomains.

Hut algorithm is a reasonable choice that will be sufficient for our purposes without sacrificing essential performance goals.

Calculating the probabilities for creating synapses is the most time consuming part of MSP. Similarly to Barnes-Hut, we combine distant neurons into groups whenever possible instead of considering them individually. Neurons in the same group have a similar distance to the source neuron. We represent a group of neurons through a virtual neuron whose position is a linear combination of the positions of the group members. Weight factors position the virtual neuron closer to neurons with many vacant dendritic elements. The number of vacant dendritic elements of the virtual neuron is the sum of vacant dendritic elements present in the entire group. This approach resembles the concept of the center of mass in gravitational versions of the Barnes-Hut algorithm. Only neurons close to the source neuron are considered individually because they differ more in their relative distance to the source neuron. Otherwise, the probability error, caused by using the averaged position of the virtual neuron, could become too large. Figure 2 shows an example. Neuron 2 is too close to the source neuron and hence not considered as part of a group. Below, we explain the three steps of our algorithm: (i) tree construction, (ii) tree update, and (iii) target neuron selection.

a) Tree Construction: Similarly to the Barnes-Hut algorithm, we start by forming a tree of neuron groups. However, compared to particles in an n-body simulation, our neurons do not move. Thus, the tree is created only once at the start of the simulation. The tree construction proceeds as follows: Given a cube that contains all neurons in our simulation domain, we create a spatial tree representation of the domain step-by-step. If the domain contains more than one neuron, we subdivide it into eight cubical subdomains of the same size. Each of the eight subdomains is then recursively subdivided if it contains more than one neuron. The recursion ends when every subdomain contains at most one neuron. As a result, we obtain an octree (i.e., a tree with at most eight children per node) with the root representing the cube containing all neurons. Its children are the eight subdomains that it was divided into and so on. Every leaf in the tree represents a single neuron, every inner node represents a virtual neuron in its subdomain. This hierarchy of subdomains defines the groups of neurons that we need. For ease of illustration, we have used two-dimensional examples in our figures. In comparison to three dimensions, we have subdivided a domain into four squares and the resulting tree is a quadtree (i.e., a tree with at most four children per node). Figure 3 shows the final subdivision of a domain and the resulting tree. The depth of the tree depends on the distribution of the neurons. The closer neurons are located to each other, the more domain subdivisions are required and the tree depth could in principle grow indefinitely [16]. Fortunately, biological constraints ensure that neurons are not positioned at purely arbitrary distances from each other within the brain. Although organized in layers, they are still rather homogeneously distributed. For this reason, we assume that the depth of the tree is $\Theta(\log n)$. This roughly corresponds to the distance between homogeneously distributed neurons. As in Barnes-Hut, we create the tree by successively inserting all neurons into the tree. Since our tree is of depth $\Theta(\log n)$, tree creation takes $O(n \log n)$.

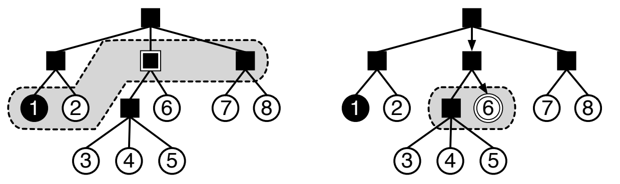
b) Tree Update: Neurons do not move, but the number of their vacant dendritic elements is subject to change. For this reason, the tree has to be updated before creating new synapses. For every leaf (i.e., real neuron), we store the current number of vacant dendritic elements. For every inner node (virtual neuron), we not only update the number of vacant elements but also the position of its virtual neuron. The number of vacant dendritic elements is simply the sum of those available on its (direct) children. Let v be a virtual neuron. Then the number of vacant dendritic elements is $D_v = \sum_{i \in children} D_i$. The position is a linear combination of the positions of its children and their vacant dendritic elements. After updating its vacant element count, the x-coordinate of the virtual neuron v is calculated as $x_v = 1/D_v \sum_{i \in children} x_i D_i$. The y and z-coordinates are obtained in a similar way. We update the information in the tree bottom-up from the leaves to the root via postorder traversal, which takes time $O(n)$.

c) Target Neuron Selection: After a tree update, we form new synapses by finding a target neuron for every vacant axonal element. To minimize the number of probability calculations, we already decide at the coarser level of neuron groups which neurons the source neuron will not connect to and which we therefore do not need to consider any further. If the source neuron decides to connect to a virtual neuron, we unfold the group it represents. This makes all its (virtual) constituent neurons visible, from which we again select one. Every group selection decreases the number of target neuron candidates. The recursion ends once a single real target neuron has been selected. Selecting a target neuron for a given source neuron means choosing a path from the root to a leaf. To decide which subdomains we consider as a whole on the path down the tree, we use the acceptance criterion (AC) of the Barnes-Hut method. Let d be the distance from the source neuron to the virtual neuron. Let l denote the length of the virtual neuron's subdomain. If $l/d < \theta$, we calculate a single connection probability for the entire subdomain. Otherwise, we unfold it and recursively apply the AC to its constituent subdomains. Here, θ is a configurable precision parameter that ensures that subdomains for which we calculate probabilities are distant enough from the source neuron in relation to their size. Note that a subdomain can be unfolded for two reasons, either because it has been selected to form a connection or because it does not satisfy the AC.

The reason for using the Barnes-Hut AC is as follows. Neurons in a subdomain that satisfy the criterion have relatively small individual chances of being selected as a target neuron. In addition, they have similar distances to the source neuron. Thus, the distance of their virtual neuron seems to properly approximate their individual distances to the source neuron. On the other hand, the neurons in a subdomain that does not meet the AC may show greater relative differences in their distance to the source neuron. Consequently, their probabilities differ more and a single virtual neuron would not properly represent all neurons in the group.

Figure 4 continues our previous example. Starting from the root, the AC is applied. Because the root does not satisfy $l/d < \theta$ for its domain, we need to unfold it. The same is true for its first child. At this point, the recursion stops because both its children are leaves. The other two children of the root satisfy the AC and remain closed for now. The gray shaded area marks the first set of nodes for which we now calculate probabilities, as described in Section III. The difference to the original MSP is that we only consider a subset of neurons, with some of them being virtual. Based on their probabilities, we select one neuron from this subset. It is the second child of the root (framed square), a virtual neuron. In the next step (Figure 4b), we unfold the subdomain of the selected neuron and apply the AC to one virtual neuron and neuron 6. Both are accepted (gray area). Note that a real neuron is trivially accepted since unfolding is not possible. We calculate the connection probabilities and select one neuron, which is neuron 6. It is a real neuron and thus the target neuron for source neuron 1. Now, the selection terminates at this point.

The complexity of the target neuron selection depends on the number of nodes to consider. With $\theta = 0$, all subdomains are unfolded and we consider every neuron for a given source neuron. That is, the algorithm is exact and behaves as the original MSP with complexity $O(n^2)$. For $\theta > 0$, we start



(a) Step 1 (b) Step 2

Fig. 4. A two-dimensional example of target neuron selection in two steps (a) and (b). Neuron 1 is the source neuron. Areas shaded in gray identify the set of (virtual) neurons from which one must be selected. The selection is framed. In (b), arrows indicate the path from the root to the target neuron 6.

considering groups of neurons. Here, the complexity depends on the depth of the tree, which we assume to be $\Theta(\log n)$, as previously stated. When randomly selecting nodes on the path down from the root to the target neuron, after every selection the depth of the remaining subtree is reduced in the worst case by one only. That is, we have to perform $\Theta(\log n)$ steps (depth of the tree) until we find a target neuron. To determine the complexity of each step, we follow the argument of Barnes and Hut for homogeneously distributed particles [9]. In particular, when increasing the number of neurons, the new additional subdomains not containing the source neuron incur a certain amount of extra probability calculations. This amount depends on θ but not on the number of neurons. Consequently, increasing the number of neurons by a constant factor only increases the number of probability calculations by an additive constant (for θ). That is, the complexity of each step is $O(\log n)$. Therefore, it takes time proportional to $O(\log^2 n)$ to find a target neuron for one source neuron. Under the biologically motivated assumption that the tree is of depth $\Theta(\log n)$, the complexity of finding a target neuron for every neuron in one connectivity update is therefore $O(n \log^2 n)$.

d) Error Analysis: In Figure 4, the chain of subdomains that contain the source neuron (root and its first child) is completely unfolded until the source neuron is encountered. This is very helpful for avoiding autapses, as is usually desirable in brain simulations. To avoid autapses, MSP sets the probability of a neuron to connect to itself to zero. However, during the recursive descent, every virtual neuron's probability, which depends on the position and number of dendrites, is based on all neurons in its subdomain. As we have no information about whether or not the source neuron is included in a particular virtual neuron's domain, we cannot exclude the source neuron's contribution from the virtual neuron. Consequently, the probability assigned to a virtual neuron whose group contains the source neuron is too high. This is because the zero probability of the source neuron applies only if it is considered directly and not through a virtual neuron. As a side effect, the selection also becomes biased towards other neurons in the same domain through the inflated probability of the source neuron. To eliminate this bias, we define the AC in such a way that the source neuron can never become part of a virtual neuron's domain during probability calculations. We accomplish this by setting $\theta \leq 1/\sqrt{3}$.

According to the AC, every subdomain with $l/d \geq \theta$ is unfolded, with d again being the distance from the source neuron and l the edge length of the subdomain. The ratio l/d decreases with increasing distance d between the source neuron and the subdomain's virtual neuron. This is also true for subdomains of different sizes l that contain the source neuron. For those, $d = \sqrt{l^2 + l^2 + l^2} = l\sqrt{3}$ is the greatest possible distance between the source neuron and the virtual neuron (in three dimensions). After substituting $l\sqrt{3}$ for d, $l/d \geq \frac{l}{l\sqrt{3}} = 1/\sqrt{3}$ becomes the smallest possible ratio between l and d. As a result, setting $\theta \leq 1/\sqrt{3}$ and defining the AC as $l/d < \theta$ unfolds all subdomains containing the source neuron. Note that $1/\sqrt{3} > 0.5$. That is, setting $\theta \leq 0.5$ is a practical usage guideline for our algorithm. To achieve the same behavior in two dimensions, we need to set $\theta \leq 1/\sqrt{2}$.

Before a target neuron t_k at depth k in the tree is selected for synapse creation, all virtual neurons v_i, $i = 0, \ldots, k-1$ on the path to the target neuron are selected. This path is of length k. Thus, the probability of selecting t_k is $P(v_0 \cap \ldots \cap v_{k-1} \cap t_k)$, a product of conditional probabilities. It depends on the choice of the precision parameter θ and the path length k. With $\theta = 0$, our approximation method uses the same probabilities as the exact MSP for selecting a target neuron. Here, the error of our approximation becomes zero, while the complexity becomes quadratic. Of course, an error analysis is supposed to indicate the extent to which the approximated probabilities differ from the exact MSP. However, it seems difficult to determine analytically which effect the error has on the actual structure of the resulting neural network. Hence, we run simulations with different precision parameters and analyze their effect on the resulting networks in Section V.

e) Summary: The execution flow of our scalable MSP algorithm is depicted in Figure 5. The steps within the for loop are executed in every iteration. As can be seen, the connectivity update, which depends on the number of synaptic elements, has the greatest complexity. As synaptic elements grow much slower compared to the frequency at which neurons fire, the connectivity update is run only every 100 simulation steps. However, as our experimental results in Section V show, even a single connectivity update using the exact MSP takes as long as about 50 min for 10^6 neurons, which makes it the clear bottleneck. This is why we must update the connectivity more quickly. Our algorithm brings the complexity from $O(n^2)$ down to $O(n \log^2 n)$, which now becomes the overall computational complexity of the approximated MSP.

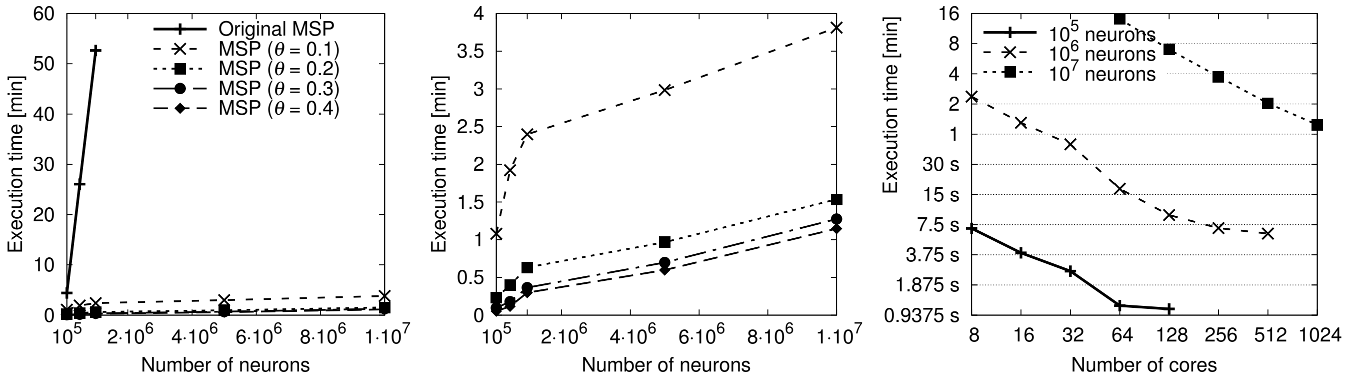
V. Results

Let us now evaluate our algorithm in terms of performance and impact of the approximation on the result quality. Our simulation parameters correspond to layer 5A of the rat cortex [17], which is about 500 µm thick with a density of 54,500 neurons per mm^3. Of these, 20% are inhibitory, the remaining 80% are excitatory. We randomly distribute the neurons in a volume of height 500 µm and let the other two dimensions grow with the number of neurons. Initially, no synapses exist. Every neuron is initialized with vacant synaptic elements: one excitatory and one inhibitory dendritic element plus one axonal element. The type of the axonal element depends on the type of the neuron. Excitatory neurons form excitatory axonal elements while inhibitory neurons form inhibitory axonal elements. However, both neuron populations grow excitatory and inhibitory dendritic elements. Synapses are only possible between synaptic elements of the same type. That is, excitatory axonal elements only connect to excitatory dendritic elements. A similar rule applies to inhibitory synaptic elements. In our experiments, we compare parallel versions of the original MSP algorithm and our scalable MSP approximation algorithm. In both cases, neurons are evenly distributed across processes. We run our experiments on the Lichtenberg cluster at TU Darmstadt. Our compute nodes are equipped with two Intel Xeon E5-2670 (8 cores each) and 32 GiB RAM. The cluster network is InfiniBand FDR.

a) Performance: With the exception of synapse formation, which is accelerated via approximation, the time-step loop of our scalable algorithm performs the same steps as the original MSP algorithm. Another difference is the construction of the octree, which our algorithm performs once before the simulation starts. Nevertheless, even for 10^7 neurons tree construction does not exceed 2 min. We do not include this time in our measurements as it is a one-off expense. Figures 6a and 6b show weak scaling results for one connectivity update for the original MSP and our algorithm. We simulate the very first connectivity update where no synapses yet exist and every neuron has vacant elements as described above. That is, every neuron is trying to find a target neuron for its vacant axonal element. The original MSP needs about 50 min for 10^6 neurons, while our algorithm terminates in 2.5 min. On the other hand, even for high precision with $\theta = 0.1$, our algorithm is still in the range of 5 min for 10^7 neurons. This large number of neurons is practically out of reach for the original MSP. The results also show that reducing the precision of our method from 0.1 to 0.2 can help to further reduce the execution time by at least a factor of 2.5. Figure 6c

```
1:  Create empty network without synapses              ▷ $O(n)$
2:  Initialize number synaptic elements per neuron     ▷ $O(n)$
3:  Construct tree from domain                         ▷ $O(n \log n)$
4:  for $i \leftarrow 1, simulation\ steps$ do
5:      UpdateElectricalActivity                       ▷ $O(n)$
6:      UpdateSynapticElements                         ▷ $O(n)$
7:      if $i \bmod 100 = 0$ then
8:          UpdateConnectivity {                       ▷ $O(n \log^2 n)$
9:              Delete synapses & update network       ▷ $O(n)$
10:             Create synapses {
11:                 Update tree                        ▷ $O(n)$
12:                 Find target neuron for every
13:                   vacant axonal element            ▷ $O(n \log^2 n)$
14:                 Update network } }                 ▷ $O(n)$
15:     end if
16: end for
```

Fig. 5. The simulation flow of MSP when using our hierarchical algorithm. The complexity of each step is shown on the right. The number of simulation steps is a parameter provided by the user.



(a) Weak scaling original vs. scalable MSP (b) Weak scaling of scalable MSP only (c) Strong scaling of scalable MSP only

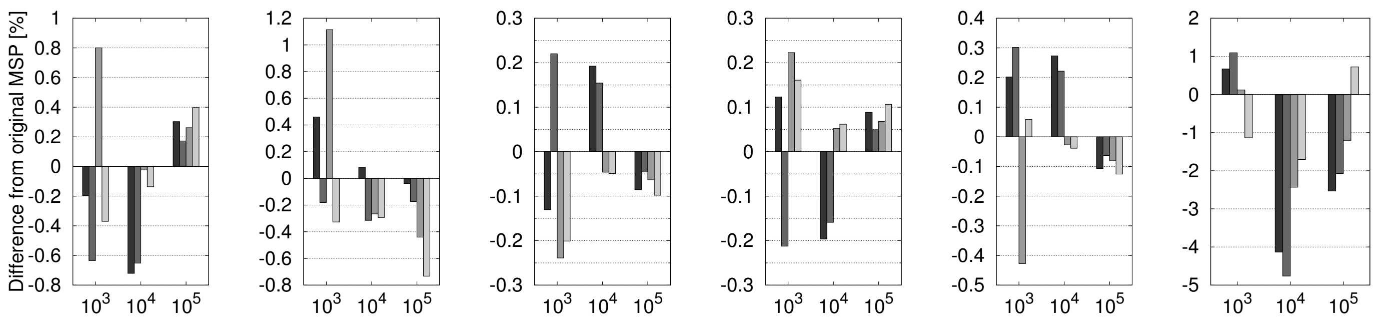
Fig. 6. Scaling results. (a-b) Weak scaling execution times with different precision parameters. The legend in (a) also applies to (b). The number of neurons per core (MPI process) is 10^4. The numbers of neurons (cores) are: 10^5 (10), $5 \cdot 10^5$ (50), 10^6 (100), $5 \cdot 10^6$ (500), 10^7 (1000). (c) Strong scaling execution time with $\theta = 0.3$.

depicts strong scaling results for our method. The smaller decrease in execution time for 32 cores was consistent across measurements (10 for 10^5 neurons, 5 for 10^6). We believe the reason for this dent can be found in the hardware topology of the target system. Although only responsible for a subset of the neurons, in both implementations each process stores the data of all neurons. Otherwise, the design of the octree would have been more complicated. That is, our algorithm replicates the entire octree on each process. This is why the total number of neurons is currently limited by the amount of memory available per compute node. Work has already begun on a further implementation of our scalable algorithm, with a distributed octree that truly partitions the neurons and only stores locally those neurons a process is responsible for. This will enable our algorithm to perform even larger simulations of structural plasticity than it can handle now.

b) Accuracy: To validate the accuracy of our algorithm, we compare the neural networks it generates to those of the original MSP. We consider a neural network as a weighted directed graph where neurons are the vertices and synapses are the directed edges pointing from a source neuron to a target neuron. The number of synapses reaching from the source to the target is the edge weight. Our comparison is based on the following graph topology metrics that Butz et al. [18] use to describe the structure of neural networks: (i) number of edges, (ii) average Euclidean distance, (iii) average shortest path length, (iv) global efficiency, (v) average betweenness centrality, and (vi) average clustering coefficient. We compare networks of the original MSP and our algorithm with 10^3, 10^4, and 10^5 neurons. The quadratic computational complexity of some of the graph metrics prevents us from evaluating them for larger neuron counts. We start with an empty network and initially vacant synaptic elements. The biological simulation time is $6 \cdot 10^6$ msec, which allows 60,000 connectivity updates. At this time, the neuronal electrical activity and the network have reached their equilibrium and change only insignificantly. Figure 7 shows the metrics of the networks generated by our algorithm relative to those of the original MSP. Except for the average clustering coefficient, the networks produced by our algorithm differ only by about 1% from the original MSP. This is even true for low precisions. However, for 10^4 neurons the average clustering coefficient differs by about 5% with even a small $\theta = 0.2$. To investigate this case further, we calculate the average and standard deviation of the metrics over 11 simulations for both algorithms (Figure 8). We use a different random number seed for every run. Interestingly, even for the exact MSP, individual average clustering coefficients vary by about 5% across measurements. That is, the 5% difference can also be found between different runs of the exact MSP. Figure 8 gives an overview of how the results vary. It can be seen that the difference between the average of the exact MSP and our algorithm is below 1% for all metrics. Also the standard deviations are similar except for the average Euclidean distance. However, although the "spread" around the average is different for this metric, the actual average of our method only differs by 0.14%. Hence, we do not consider the difference in the standard deviation as significant. These experiments support our claim that our approximated networks are still precise enough to represent neural networks of the exact MSP. Nevertheless, to account for the variation of the results due to MSP's probabilistic approach, several simulation runs are necessary to reliably capture the essential structure of a neural network at the end of the simulation. This is true for both, the exact MSP and our approximation of it. Thus, the scalability of the MSP algorithm is even more critical.

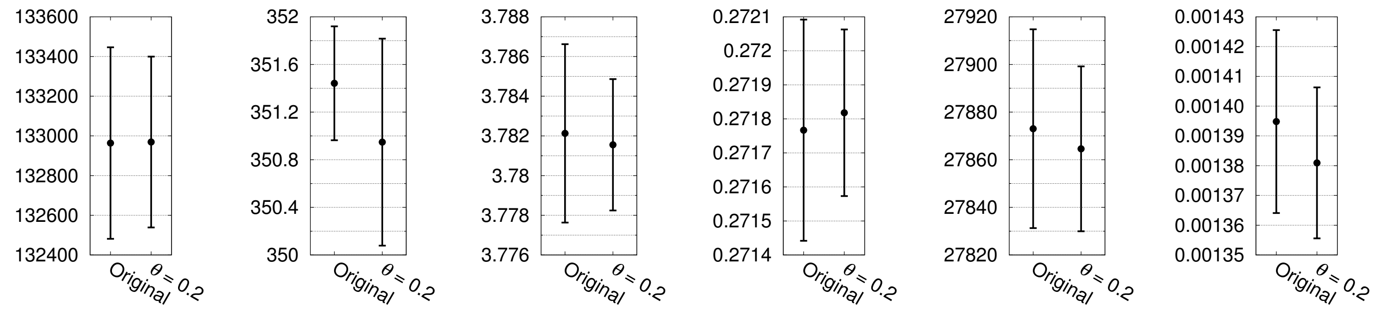
VI. Conclusion

The quadratic complexity of MSP in terms of the number of neurons limits the largest possible structural-plasticity simulations to networks with 10^5 neurons [1], which is less than the number of neurons in the brain of a mouse. Our approximation algorithm for MSP reduces the complexity to $O(n \log^2 n)$. The execution time difference between parallel implementations of both versions for 10^5 neurons is a factor of 20 in favor of our approximation algorithm. In addition, we are now able to complete simulations with 10^7 neurons for the very first time. After distributing the octree across the



(a) Number of edges
(b) Average Euclidean distance
(c) Average shortest path length
(d) Global efficiency
(e) Average betweenness centrality
(f) Average clustering coefficient

Fig. 7. Network comparison between the original MSP and our algorithm for $10^3, 10^4, 10^5$ neurons. The bars at each cluster correspond from left to right to $\theta = 0.1,\ 0.2,\ 0.3,\ 0.4$.



(a) Number of edges
(b) Average Euclidean distance
(c) Average shortest path length
(d) Global efficiency
(e) Average betweenness centrality
(f) Average clustering coefficient

Fig. 8. Network comparison between the original MSP and our algorithm with $\theta = 0.2$ for 10^4 neurons. The results per method are based on 11 simulations with a different random number seed for each run.

available processors, we expect to be able to handle even larger configurations in the future. Judging by the abilities of highly scalable Barnes-Hut implementations such as PEPC [15], we consider 10^9 neurons a realistic intermediate goal. Given that neurons do not shift their positions, which eliminates the need for expensive dynamic load balancing, and considering the further evolution of supercomputers, we foresee room for even more.

REFERENCES

[1] S. Diaz Pier, M. Naveau, M. Butz-Ostendorf, and A. Morrison, "Automatic generation of connectivity for large-scale neuronal network models through structural plasticity," *Front. in Neuroan.*, vol. 10, no. 57, 2016.

[2] M. Gewaltig and M. Diesmann, "NEST (NEural Simulation Tool)," *Scholarpedia*, vol. 2, no. 4, 2007.

[3] R. Ananthanarayanan, S. Esser, H. Simon, and D. Modha, "The cat is out of the bag: cortical simulations with 10^9 neurons, 10^{13} synapses," in *Proc. of SC'2009*, 2009.

[4] V. De Paola, A. Holtmaat, G. Knott, S. Song, L. Wilbrecht, P. Caroni, and K. Svoboda, "Cell type-specific structural plasticity of axonal branches and boutons in the adult neocortex," *Neuron*, vol. 49, no. 6, 2006.

[5] T. Hensch, "Critical period plasticity in local cortical circuits," *Nat. Rev. Neurosci.*, vol. 6, no. 11, 2005.

[6] T. Keck, T. Mrsic-Flogel, M. Vaz Afonso, U. Eysel, T. Bonhoeffer, and M. Hübener, "Massive restructuring of neuronal circuits during functional reorganization of adult visual cortex," *Nat. Neurosci.*, vol. 11, no. 10, 2008.

[7] H. Yamahachi, S. Marik, J. McManus, W. Denk, and C. Gilbert, "Rapid axonal sprouting and pruning accompany functional reorganization in primary visual cortex," *Neuron*, vol. 64, no. 5, 2009.

[8] M. Butz and A. van Ooyen, "A simple rule for dendritic spine and axonal bouton formation can account for cortical reorganization after focal retinal lesions," *PLoS Comput. Biol.*, vol. 9, no. 10, 2013.

[9] J. Barnes and P. Hut, "A hierarchical O(N log N) force-calculation algorithm," *Nature*, vol. 324, no. 6096, 1986.

[10] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *J. Comput. Phys.*, vol. 73, no. 2, 1987.

[11] S. Kunkel, M. Schmidt, J. Eppler, H. Plesser, G. Masumoto, J. Igarashi, S. Ishii, T. Fukai, A. Morrison, M. Diesmann, and M. Helias, "Spiking network simulation code for petascale computers," *Front. in Neuroinf.*, vol. 8, no. 78, 2014.

[12] I. Dammasch, G. Wagner, and J. Wolff, "Self-stabilization of neuronal networks," *Biological Cybernetics*, vol. 54, no. 4, 1986.

[13] A. van Ooyen and J. van Pelt, "Activity-dependent neurite outgrowth and neural network development," ser. Progr. in Brain Res., 1994, vol. 102.

[14] G. Prasad, J. Burkart, S. Joshi, T. Nir, A. Toga, and P. Thompson, "A dynamical clustering model of brain connectivity inspired by the n-body problem," in *Proc. of Int'l Workshop on Multimodal Brain Image Analysis (MBIA)*, 2013.

[15] *PEPC*, (accessed on 26 June 2016). [Online]. Available: http://www.fz-juelich.de/ias/jsc/EN/AboutUs/Organisation/ComputationalScience/Simlabs/slpp/SoftwarePEPC/_node.html

[16] S. Aluru, J. Gustafson, G. Prabhu, and F. Sevilgen, "Distribution-independent hierarchical algorithms for the n-body problem," *The Journal of Supercomputing*, vol. 12, no. 4, 1998.

[17] H. Meyer, D. Schwarz, V. Wimmer, A. Schmitt, J. Kerr, B. Sakmann, and M. Helmstaedter, "Inhibitory interneurons in a cortical column form hot zones of inhibition in layers 2 and 5a," *Proc. of Nat. Acad. of Sci.*, vol. 108, no. 40, 2011.

[18] M. Butz, I. Steenbuck, and A. van Ooyen, "Homeostatic structural plasticity increases the efficiency of small-world networks," *Front. Syn. Neurosci.*, vol. 6, no. 7, 2014.