On Scaling Neuronal Network Simulations Using Distributed Computing

Vladimir Omelyusik
University of Missouri
vovwm@missouri.edu

Khawar Shehzad
University of Missouri
khawar.shehzad@missouri.edu

Tyler Banks
Univeristy of Missouri
tbanks@mail.missouri.edu

Praveen Rao
University of Missouri
praveen.rao@missouri.edu

Satish S. Nair
University of Missouri
nairs@missouri.edu

***Abstract*—We investigated the computational capabilities of FABRIC, a nationwide research infrastructure with nearly 40 sites, for scaling neuroscience simulations. From the hardware standpoint, single-site characterization showed that FABRIC is a promising alternative to conventional neuroscience setups, particularly due to the availability of powerful graphics processing units (GPUs). While multi-site simulations are affected by network latency, it becomes less critical for larger networks. From the software perspective, we found that in the popular CoreNEURON library, cell distribution strategy (for parallel execution) does not affect the simulation time for biologically realistic networks, while other cases can be addressed with a minimum k-cut graph partitioning algorithm. Overall, scalability experiments revealed that FABRIC can be used to simulate networks of up to twenty-five thousand cells, with the limiting issue being GPU memory.**

Keywords—FABRIC, GPUs, neuroscience, networks, simulation, scalability

I. Introduction

Advancements in neuroscience research are progressively more dependent on hardware integration: while previously a single CPU-based desktop machine was sufficient for simulating a simplified thousand-cell model, modern realistic biophysical networks with millions of cells may, for various reasons, benefit from usage of a distributed computing environment. For instance, the need to mine rapidly growing neural databases necessitates distribution of software efficiently across compute units to increase throughput. Moreover, there is a need for advanced cyberinfrastructure (CI) to permit diverse researchers (e.g., biology, medicine, computer science, engineering) spread across geographical sites to collaborate effectively on large-scale models of brain regions in real time.

To address these challenges, we investigated the computation and communication capabilities of FABRIC [1], a nationwide research infrastructure and testbed for advancing science applications focusing on neuroscience network simulations. FABRIC is composed of nearly 40 sites spread across USA and Europe. We considered two aspects relevant for neuroscience research: network size scalability (the largest size of a network possible to simulate with the given constraints on a single-site) and simulation time efficiency (ways to increase the number of simulations per unit time). Our ultimate goal is to use these findings to construct end-to-end workflows for interactive collaboration, including allowing users to modify simulations in real time.

We first compared a single FABRIC site with the typical desktop and server setups conventionally used in neuroscience simulations. We then quantified simulation-specific overheads due to physical network latencies between sites. Analyzing the software side, we found that if the compute-optimized CoreNEURON engine [2] is used as a simulator, the distribution strategy for parallel execution (e.g., randomly assigning cells to GPUs versus based on the underlying biological network structure) does not matter for biologically realistic networks. Addressing other cases, we tested a graph partitioning algorithm, which enables faster execution of the simulation compared to random distribution of cells across the computing nodes. Finally, we release our code and comprehensive instructions on GitHub (https://github.com/raopr/neuroscience-on-FABRIC) to enable others to set up a FABRIC-specific GPU-based pipeline for efficient biophysical simulations using the CoreNEURON library.

II. Related Work

High-performance computing (HPC) resources are traditionally used for computational neuroscience workloads. The Neuroscience Gateway (NSG) [3] is an exemplar for engaging computational neuroscientists to leverage HPC resources (using XSEDE/ACCESS [4]) with a low barrier to entry. In fact, NSG continues to remain the leading consumer of ACCESS resources – consuming more than 40% more than the next user. However, the technology landscape is constantly evolving, creating new opportunities for innovation and training. The growing popularity of cloud computing for HPC applications [5], [6], [7] demands new research workflows that integrate neuroscience with advanced CI beyond traditional HPC environments. While past neuroscience development efforts have targeted the use of HPC through middleware the CI adoption was poor. The heterogeneity of hardware, high-speed networking, and the ability to customize one's computing

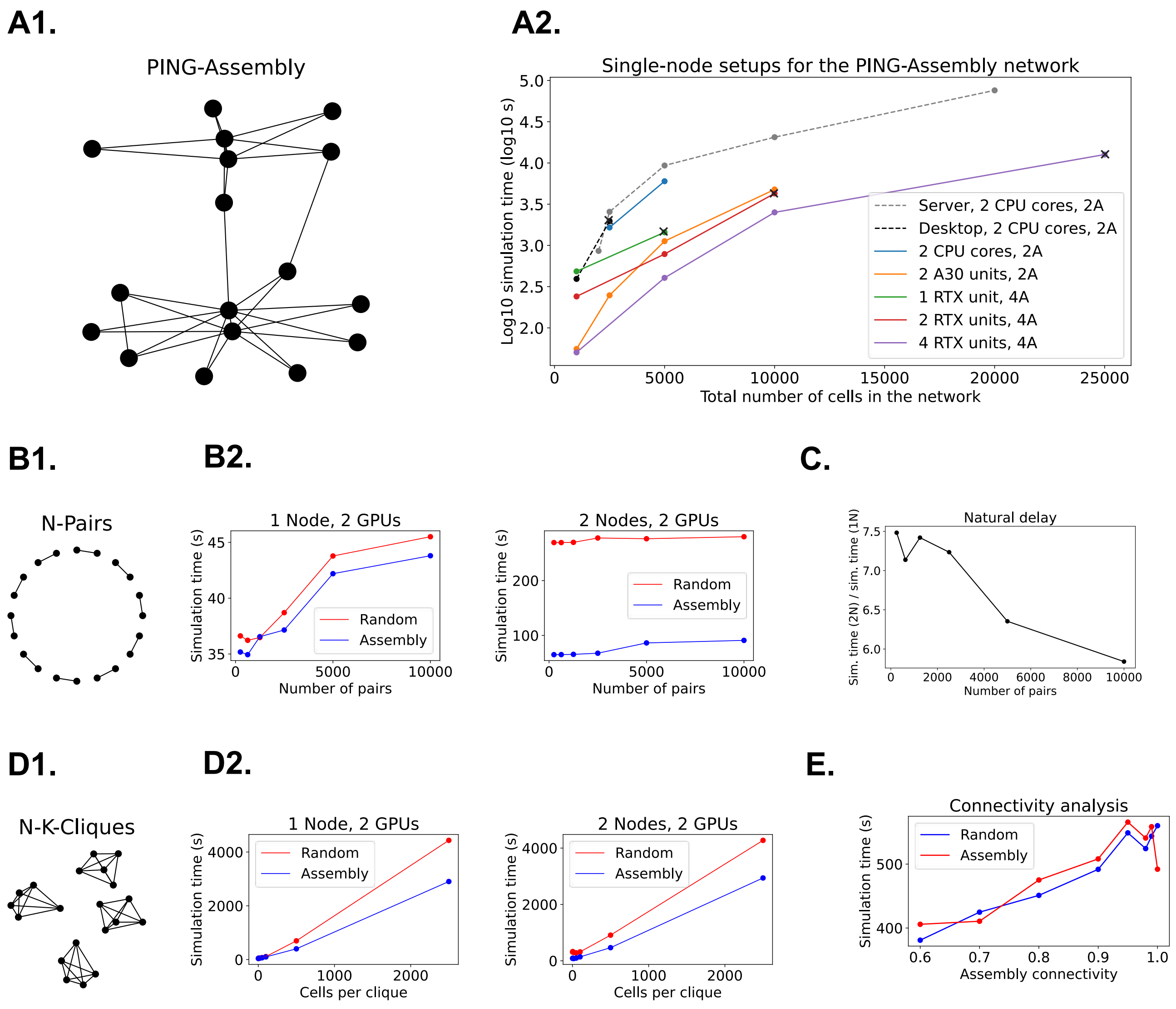

Figure 1. Characterization of FABRIC capabilities for neuroscience simulations | Single-site hardware characterization: A1. Biologically realistic PING-network with 2 assemblies. **A2.** Comparison of a single FABRIC site performance (solid lines) with typical desktop and server setups (dashed lines) for a PING-Assembly network. Crosses indicate the maximum size of network which was possible to simulate. **Multi-site hardware characterization: B1.**N-Pairs network with 10 assemblies. **B2.** Characterization of simulation time overhead due to natural delay between sites. **C.** Simulation-specific estimate of natural delay. **Software characterization: D1**. N-K-Cliques network with 4 cliques of 5 cells. **D2.** Characterization of simulation time overhead due to increased network connectivity. **E.** Comparison of random and by-assembly distribution based on (extreme) probability of synaptic connections within an assembly (a PING network with 2 assemblies, 1250 cells each).

services/workflows [8], the barrier to entry was high. Hence, environment in advanced CI resources demands rethinking how end-to-end neuroscience workflows can be seamlessly integrated with these resources.

Scaling neural network simulations has been an ongoing challenge in the neuroscience community. Migliore et al. [9] investigated the scaling of NEURON network simulations using Message Passing Interface (MPI) and multiprocessor systems such as a Beowulf cluster and a supercomputer. They observed that spike communication overhead was less of a bottleneck compared to cache memory effects. As the number of processors was increased, the problem size became small enough that a supercomputer achieved better speedup. Later Hines et al. [10] showed how models implemented in NEURON can be parallelized using MPI for executing on local resources (e.g., a single computer, a cluster of computers connected by Ethernet, a multiprocessor computer) as well as supercomputers.

Recent efforts have explored the use of GPUs for accelerating neuronal biophysical simulations (e.g., Arbor [11], CoreNEURON [2]). They focused on simulating large scale neuronal networks with multiple compartments. There is growing interest in scaling neuroscience computations beyond traditional HPC environments, including using distributed computing.

III. Methods

A. FABRIC

Created in 2019, FABRIC is a nationwide research testbed with high speed optical links interconnect 30 geographically distributed sites on FABRIC via different Layer 2 and Layer networking service. FABRIC has Internet2 connectivity to public clouds such as Amazon Web Services (AWS) and Microsoft Azure. A FABRIC node, which contains a rack of compute, storage, and networking devices, is equipped with cutting-edge processors, large amounts of RAM, non-volatile memory express (NVMe) drives, GPUs, field-programmable gate arrays (FPGAs), and 100/200 Gbps SmartNICs. Experimenters on FABRIC can choose virtual machine (VM) or container configurations for provisioning resources.

B. NEURON and CoreNEURON

NEURON is a simulator for conductance-based single cell and network models developed by Duke, Yale, and the Blue Brain Project. The simulator provides API for defining the cell's morphology and biophysical properties as well as synaptic models for inter-cellular connectivity, which are available both in its own HOC scripting language and Python. CoreNEURON, an extension of NEURON, targets computational efficiency and optimized memory usage. It also allows simulations to be run on GPU architectures [2]. Different from NEURON, which can be installed via the *pip* package installer, CoreNEURON requires manual compilation for specific hardware.

CoreNEURON supports MPI-based CPU and GPU parallelization by having the user specify the number of processes to spawn and manually assign each cell to its process. GPU support is enabled via an on/off switch, and our preliminary experiments revealed that by default all available GPU units are used when the switch is on. Thus, process assignment to GPU units is controlled externally by modifying the list of units visible to the process.

Additionally, we found that synaptic communication between cells assigned to different processes (i.e., exchange of spike event information) during the simulation time is performed internally via memory-efficient connections called InputPresyns which do not have to act as threshold detectors [2]. Since these connections are managed by the simulator, their existence leads to non-intuitive results when comparing different distribution strategies for cells to enable parallelization.

C. Cells and networks

Cells. In our experiments, we used one-compartmental spiking cells consisting of the soma with three ion channels (leak, Na, K) and several synaptic channels. Voltage dynamics was modeled using the Hodgkin-Huxley formalism,

$$C \times dV/dt = I_{leak} + I_{Na} + I_K + \Sigma I_{Syn} - I_{Inj},$$

where I_X represent current types [9]. The cells were of two types, excitatory and inhibitory, with exponential synapses.

PING-Assembly. The network consists of N cell assemblies of K cells, 80% of which are excitatory and 20% are inhibitory, following biophysical connectivity principles. The connectivity between assemblies was set to 10%. Within-assembly connectivity was set probabilistically following Borgers [12] (no excitatory-to-excitatory connections, other connections had probability of 0.5), so that the network produced biologically realistic oscillations in the gamma range (30-40 Hz).

N-Pairs. The network consists of N cell assemblies of 2 RTM cells connected via a single excitatory synapse. The cell and synaptic parameters were set to the same values as for the PING-Assembly network, but current injection of 10 nA was applied to only one of the cells.

N-K-Cliques. The network consists of N cell assemblies of K RTM cells, fully connected with excitatory synapses. The cell and synaptic parameters were set to the same values as for the PING-Assembly network, but current injection of 10 nA was applied to only one of each assembly's cells.

D. Distribution strategies

CoreNEURON requires the cells to be initialized directly on their respective processes rather than letting a master process to create and distribute the cells. Consequently, we started all our experiments by generating a list of global cell indexes (GIDs) from 0 to NK–1, which represented a network of N assemblies each having K cells, and then imposed true connectivity according to the network's type. For convenience, we assumed that assemblies contained cells with consecutive GIDs (e.g., the first assembly could have GIDs from 0 to X, the second one could have GIDs from X+1 to 2X, etc.) We then spawned N processes and utilized three strategies to distribute the indexes and initialize the cells. (Note that cell distribution is independent of the underlying connectivity structure.)

Random distribution was done by shuffling the GID list, and splitting the shuffled indexes equally between processes. By-assembly distribution preserved the true assembly structure, i.e., the GID list was split into N equal parts without shuffling, each of which was assigned to its own process. By-partitioning distribution was done by running a partitioning algorithm on the graph representation of the network and distributing the cells according to the results of partitioning.

IV. Results

A. Single-site setup for efficient neuroscience simulations

A FABRIC site can be accessed by initializing a virtual machine and allocating a specified number of computing resources (the number of CPU cores and GPUs, memory size). Moreover, several virtual machines can be initialized on the same site and connected via an L2 bridge. We first analyzed the trends in simulation time when using a single site setup and compared them with a typical server setup commonly used for neuroscience simulations.

In these experiments, we simulated a biologically realistic PING-Assembly network, which is comprised of N cell assemblies of K cells, 80% of which are excitatory and 20% are inhibitory, as cited in Methods. Connectivity within each assembly followed biologically realistic principles.

Additionally, 10% of cells in each assembly were connected to random cells in other assemblies via excitatory synaptic links (Fig. 1-A1). We verified that the network produced oscillations in the gamma-band (around 30-40 Hz; [12]).

We simulated networks with varying numbers of assemblies and cells per assembly using the following hardware setups: a desktop (Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz 2.11 GHz) and our laboratory server (Intel(R) Xeon(R) Gold 6252 CPU @ 2.10GHz) typically used for neuroscience simulations, a FABRIC CPU node at Florida International University and University of Michigan / FIU, a FABRIC node at Georgia Tech / GATECH having 2 A30 GPU units, and a FABRIC node at University of Maryland / MAX, having four RTX6000 units (2 VMs, each having 2 GPU units). The cells were distributed randomly across units, and simulation time reported by CoreNEURON was used as the total execution time. (We ignored the time taken to distribute the cells.) Server and desktop simulations were done with the usage of the Brain Modeling Toolkit (BMTK) [13] for additional memory optimization.

Comparing the runtimes for simulations with the same number of assemblies, we observed that FABRIC is a viable alternative to conventional simulation setups, on average being 36% faster in CPU runs and 500% faster in GPU runs (Fig. 1-A2). Moreover, despite potential communication latencies between GPU units and VMs, simulation time decreased linearly with the increase in the number of GPU units, making it feasible to run larger-scale simulations (25K cells) in a reasonable amount of time.

B. *Multi-site simulations and impact of network latency*

Large-scale networks can often be simulated only on multiple sites due to two major reasons. First, each FABRIC site has a limited number of physical GPU units. Second, our initial experiments revealed that a virtual machine on a site can have at most 2 GPU units attached, as an attempt to allocate more units fails the node creation request. However, a 2-GPU VM takes a significantly longer time to allocate compared to a 1-GPU VM, so practically it is more convenient to initialize several 1-GPU VMs on multiple sites rather than a multi-GPU machine on one site. Consequently, we analyzed the simulation time overhead due to the physical distance between sites (referred to as "natural" delay) when using the target distribution strategies.

In these experiments, we used the N-Pairs network (Fig. 1-B1), consisting of identical independent assemblies of two cells, one of which (the "sender") receives a constant current injection stimulus, generates an action potential, and sends a spike event to the other cell (the "receiver") via an excitatory synaptic connection. The network's parameters were adjusted to values that ensured that the receiver also generated an action potential on receiving the spike event (see Methods). We simulated the network while varying the number of pairs (625, 1250, 2500, 5000, 1000) on a single site with 2 GPU units (University of Maryland / MAX, RTX6000) and two sites each having 1 GPU unit (Florida International University / FIU and University of Michigan / MICH, RTX6000 on both) connected via an L2 bridge (L2STS, SharedNIC). We first compared the simulation time reported by CoreNEURON for the target distribution strategies, random assignment of senders and receivers across GPU units, and assigning half of all assemblies to each GPU (random vs by-assembly distribution, see Methods). We found that by-assembly distribution is marginally faster on the single site and 3-4 times faster in the multi-site setup (Fig. 1-B2 left and right respectively).

The unique structure of the N-Pairs network allowed us to quantify the simulation-specific upper bound of the natural delay. To do that, we assigned all senders and all receivers to different GPU units, emulating the worst possible case of the random distribution, and computed the ratio of simulation time on 2 sites to simulation time on 1 site. The delay estimate appeared to be size-dependent and decreasing with the number of cells, with the mean value of around 6.5 times (Fig. 1-C).

C. *Impact of overall number of synaptic connections on simulation time*

CoreNEURON relies on MPI for CPU- and GPU-based parallelization: the user specifies the number of processes to spawn and assigns cells to each process manually. Since spike information between processes is passed via memory-efficient connections (see Methods for details), the overall connectivity effect is non-trivial. We first analyzed how the number of assemblies affects simulation time under the fixed network size and connectivity rules.

Here we used the N-K-Cliques network, which consists of N independent assemblies of K cells (Fig. 1-D1). Each cell within an assembly is connected to all other cells via excitatory synaptic paths, i.e., there are two connections between each pair of cells, and only one cell in the assembly receives a current injection stimulus. Consequently, a N-2-Cliques network is equivalent to the N-pairs network from above given the addition of the receiver-to-sender connection, and the decrease in the number of cliques results in the increase of the overall network connectivity. We fixed the network size at 5000 cells and varied the number of cliques N (2500, 1250, 250, 100, 50, 10, 2), setting K to 5000/N and simulating the network in the same setup as the N-Pairs one from above. We found that the difference in simulation time between random and by-assembly distribution grows as the network connectivity increases. This effect was observed in both single- and multi-site setups (Fig. 1-D2, left and right, respectively), adding to the "natural" delay in the latter case. Thus, we observed that, under the same connectivity, simulating a high number of smaller assemblies is faster than simulating a small number of bigger assemblies. This led us to the conclusion that the number of synaptic connections (smaller in the former and higher in the latter case) might be one of the crucial factors that affects the simulation time.

D. *Distribution strategy and its impact on simulation time for biologically realistic connectivity within assemblies*

Since biologically realistic networks are not characterized by full connectivity, we considered how connectivity rules affect simulation time. In our first experiment (Florida International University / FIU and University of Michigan / MICH, 1 RTX6000), we simulated a PING-Assembly network with 2 assemblies of 1250 cells while varying the probability of synaptic connections within each assembly from 0.6 to 1. We did not observe conclusive trends of one strategy being faster than the other but noted a higher difference at the full connectivity level. We then simulated (Georgia Tech / GATECH, 2 A30 GPUs) a PING-Assembly network with 2 assemblies and biologically realistic connectivity (see Methods) while varying the number of cells per assembly (500, 1000, 2500, 5000, 10000). We observed no statistically significant difference between simulation time under target distribution strategies as the network size increased (the difference of 3-7%, p-value of two-sample t-test was 0.99). In contrast, for completion purposes, we simulated an unrealistic fully connected PING-Assembly network with the same configuration and found insignificant (two-sample t-test, p-value of 0.84) yet increasing percent difference between random and by-assembly distribution strategies (1% at 500 cells, 12% at 1250 cells, 26% at 2500 cells). Since the different distribution strategies we tested did not have a significant effect on simulation time, we hypothesize that CoreNEURON's memory optimization algorithm can efficiently handle the biologically realistic cases.

To address the cases when the distribution strategy matters, we tested how a minimum k-cut partitioning algorithm detected clusters of densely connected cells. Here, a biophysical network is viewed as a weighted undirected graph with vertices represented by cells and edges by synaptic connections; the weights can be set to 1 or proportional to the absolute values of corresponding synaptic weights. The partitioning is done by minimizing the standard objective function.

In our experiments, we used networkx [14] and METIS [15] Python libraries to perform partitioning of the PING-Assembly network. We set the number of assemblies to 2 (each of 5000 cells) and simulated a network with random and by-partitioning distribution on a single FABRIC site. We first tested the ideal case when the number of assemblies is known (k = 2), for which by-partitioning strategy was faster than random (4054 vs 5115 sec). For the cases when the number of assemblies is not known, a smaller number of partitions resulted in faster simulation time.

E. *Interactive simulation control for collaboration*

On-going work targets constructing workflow templates for efficient neuroscience collaboration using FABRIC's high-speed connectivity. Examples include online simulation control of experimental setups and integration of database searches during ongoing simulations.

F. *A guide to use FABRIC for neuroscience simulations*

To make FABRIC accessible to a larger group of neuroscience users, we created comprehensive instructions to guide users on site setup, library installation, CoreNEURON compilation, communication via SLURM, etc. (https://github.com/raopr/neuroscience-on-FABRIC).

V. Conclusion

Our experiments point out that FABRIC provides a promising alternative to conventional neuroscience setups for simulating biologically realistic networks. From the hardware standpoint, availability of GPU units allows dramatic reduction of simulation time for larger networks. While multi-site configurations are characterized by inherent delays, its effect decreases as the network size grows.

We also found that the CoreNEURON simulator has a memory optimization strategy that obviates the need for cell-partitioning approaches. We found that cases where distribution strategy matters, and the underlying assembly structure is not known, a min-cut graph partitioning algorithm performs better.

Acknowledgment

We thank the CoreNEURON and FABRIC teams for their assistance. Praveen Rao would like to acknowledge the partial support of NSF Grant No. 2201583.

References

[1] P. Ruth, I. Baldin, K. Thareja, T. Lehman, X. Yang, and E. Kissel, "FABRIC network service model," in *2022 IFIP Networking Conference (IFIP Networking)*, IEEE, 2022, pp. 1–6.

[2] P. Kumbhar *et al.*, "CoreNEURON: an optimized compute engine for the NEURON simulator," *Front Neuroinform*, vol. 13, p. 63, 2019.

[3] S. Sivagnanam, K. Yoshimoto, N. T. Carnevale, and A. Majumdar, "The neuroscience gateway: enabling large scale modeling and data processing in neuroscience," in *Proceedings of the Practice and Experience on Advanced Research Computing*, 2018, pp. 1–7.

[4] J. Towns *et al.*, "XSEDE: accelerating scientific discovery," *Comput Sci Eng*, vol. 16, no. 5, pp. 62–74, 2014.

[5] "'Azure high-performance computing,' /azure.microsoft.com/en-us/solutions/high-performance-computing."

[6] "'High Performance Computing Solutions,' /cloud.google.com/solutions/hpc."

[7] "High Performance Computing - AWS /aws.amazon.com/hpc/."

[8] P. Calyam and S. S. Nair, "Science Gateway Development to aid Cyber and Software Automation for Neuroscience Researchers and Educators," in *13th Gateway Computing Environments Conference (Gateways)*, 2018.

[9] M. Migliore, C. Cannia, W. W. Lytton, H. Markram, and M. L. Hines, "Parallel network simulations with NEURON," *J Comput Neurosci*, vol. 21, pp. 119–129, 2006.

[10] M. L. Hines and N. T. Carnevale, "Translating network models to parallel hardware in NEURON," *J Neurosci Methods*, vol. 169, no. 2, p. 425, 2008.

[11] N. Abi Akar *et al.*, "Arbor—a morphologically-detailed neural network simulation library for contemporary high-performance computing architectures," in *2019 27th euromicro international conference on parallel, distributed and network-based processing (PDP)*, IEEE, 2019, pp. 274–282.

[12] C. Börgers, *An Introduction to Modeling Neuronal Dynamics*, vol. 66. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-51171-9.

[13] K. Dai *et al.*, "Brain Modeling ToolKit: An open source software suite for multiscale modeling of brain circuits," *PLoS Comput Biol*, vol. 16, no. 11, p. e1008386, Nov. 2020, doi: 10.1371/journal.pcbi.1008386.

[14] A. Hagberg, P. J. Swart, and D. A. Schult, "Exploring network structure, dynamics, and function using NetworkX," Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2008.

[15] G. Karypis and V. Kumar, "METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices," 1997.