SAND2020-7042C

Brief Announcement: Provable Neuromorphic Advantages for Computing Shortest Paths

James B. Aimone*
Sandia National Laboratories
Albuquerque, NM, USA
jbaimon@sandia.gov

Yang Ho*
Sandia National Laboratories
Albuquerque, NM, USA
yho@sandia.gov

Ojas Parekh*
Sandia National Laboratories
Albuquerque, NM, USA
odparek@sandia.gov

Cynthia A. Phillips*
Sandia National Laboratories
Albuquerque, NM, USA
caphill@sandia.gov

Ali Pinar*
Sandia National Laboratories
Livermore, CA, USA
apinar@sandia.gov

William Severa*
Sandia National Laboratories
Albuquerque, NM, USA
wmsever@sandia.gov

Yipu Wang
University of Illinois
Urbana-Champaign, IL, USA
ywang298@illinois.edu

ABSTRACT

Neuromorphic computing offers the potential of an unprecedented level of parallelism at a local scale. Although in their infancy, current first-generation neuromorphic processing units (NPUs) deliver as many as 128K artificial neurons in a package smaller than current laptop CPUs and demanding significantly less energy. Neuromorphic systems consisting of such NPUs and offering a total of 100 million neurons are anticipated in 2020. NPUs were envisioned to accelerate machine learning, and designing neuromorphic algorithms to leverage the benefits of NPUs in other domains remains an open challenge. We design and analyze neuromorphic graph algorithms, focusing on shortest path problems. Our neuromorphic algorithms are packet-passing algorithms relying on data movement for computation, and we develop data-movement lower bounds for conventional algorithms. A fair and rigorous comparison with conventional algorithms and architectures is paramount, and we prove a polynomial-factor advantage even when we assume an NPU with a simple grid-like network of neurons. To the best of our knowledge, this is one of the first examples of a provable asymptotic computational advantage for neuromorphic computing.

CCS CONCEPTS

• **Theory of computation → Shortest paths**; • **Hardware → Neural systems**; • **Computing methodologies → Massively parallel algorithms**.

KEYWORDS

neuromorphic computing, neuromorphic complexity, spiking neural networks, graph algorithms, shortest paths

ACM Reference Format:
James B. Aimone, Yang Ho, Ojas Parekh, Cynthia A. Phillips, Ali Pinar, William Severa, and Yipu Wang. 2020. Brief Announcement: Provable Neuromorphic Advantages for Computing Shortest Paths. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '20), July 15–17, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3350755.3400258

1 NEUROMORPHIC COMPUTING AND ALGORITHMS

The brain has been proposed as a potential inspiration for parallel computing since the earliest days of computer science. Efforts to emulate the brain's architecture, known as neuromorphic computing, began in the 1980s. Recently, there are increasingly large-scale efforts, including industrial efforts from IBM (TrueNorth) and Intel (Loihi), and academic efforts including SpiNNaker, NeuroGrid, and BrainScales (see [8] for a survey). Most neuromorphic systems employ a hierarchical architecture, with local cores containing up to 1K highly interconnected neurons and many cores networked together on each neuromorphic processing unit (NPU). Neuromorphic systems with 100 million total neurons are expected to debut in 2020 [6].

While neuromorphic hardware seems naturally suited to emerging cognitive and artificial intelligence applications [2, 8], there has been growing interest in more general computational applications. Yet it remains unclear what, if any, theoretical advantage neuromorphic computation provides. There are isolated examples where the neuromorphic approach has a provable advantage. For instance Parekh et. al [7] recently gave a constant-time neuromorphic-compatible threshold-gate algorithm for matrix multiplication using a sub-cubic number of neurons, while conventional parallel algorithms require logarithmic time. Threshold-gate algorithms are not entirely satisfying as examples of neuromorphic algorithms

*This research was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SPAA '20, July 15–17, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-6935-0/20/07…$15.00
https://doi.org/10.1145/3350755.3400258

since they do not leverage some of the features of current spiking neuromorphic architectures (SNAs) such as neuron dynamics or recurrent computation. Thus it remains an open question if the power of SNAs can be harnessed to demonstrate rigorous neurmorphic resource advantages over conventional computing systems.

Our results. We propose neuromorphic algorithms for the k-hop shortest path problem, where (single-source) paths can have at most k edges. We show a polynomial-factor advantage over classical conventional algorithms and architectures. Our neuromorphic algorithms pass packets, relying on data movement for computation. The algorithms resemble conventional algorithms such as Floyd-Warshall. We observe that SNAs are a natural computing model for graph algorithms. A fair and rigorous comparison with conventional algorithms and architectures is paramount and challenging, so we consider all aspects of the implementation and their impact on execution time, including I/O costs, data movement, and topological restrictions of SNAs. We prove a polynomial-factor advantage over classical conventional algorithms for shortest paths, as presented in Table 1, even when we assume an SNA with a simple grid-like network of neurons. Our algorithms use a number of neurons polynomial in the size of the input graph. To the best of our knowledge, this is one of the first examples of a provable asymptotic computational advantage for neuromorphic computing.

1.1 Assessing neuromorphic advantages

We seek to understand and quantify potential computational advantages offered by SNAs over conventional hardware, hampered by the apparent demise of Moore's Law and Dennard Scaling. The neuron density of the current infant-generation neuromorphic hardware is promising and suggests that neuron scalability is more akin to logic gates than to general-purpose CPUs. For example, Intel's Loihi NPUs each contain 128K neurons, with a board of size approximately 6"×6" having capacity for 4M neurons [6]. As a physical existence proof, adult human brains contain analog neuromorphic circuits with about 100 billion neurons and maximum degree about 10,000 in the cortex [4]. Moreover, NPUs can be several orders of magnitude more power efficient than CPUs. Thus to obtain as fair a comparison as possible between conventional and neuromorphic computing, we compare neuromorphic algorithms with conventional serial algorithms, rather than conventional parallel algorithms. One could argue that a shared-memory parallel system might also serve as a fair point of comparison, especially given the connections between other circuit models such as NC and shared-memory models such as PRAM. However, asymptotically, neuromorphic systems are expected to be more scalable than shared-memory systems and are considered a viable beyond-Moore model of computing. Moreover, the lightweight communication of neuromorphic systems may allow them to offer a greater degree of parallelizability than conventional distributed-memory parallel systems. Thus we focus on comparing algorithms executed on a single NPU with those executed on a single CPU, where both may be aggregated in a similar fashion to form larger parallel systems.

Data movement. Our neuromorphic algorithms critically employ both computation and communication to provide asymptotic speedups over the best-known conventional serial counterparts. Our algorithms neuromorphically simulate propagation of information in the input graph. The more closely the SNA's neuromorphic circuit resembles the input graph, the more efficient our algorithms, as dense neuromorphic circuits allow for faster data movement.

To be fair, we assume a grid-like model of data storage and movement for *both* neuromorphic and conventional algorithms. For the former, we only assume access to neuromorphic circuits on a 2-d grid-like crossbar, and we employ a linear-time embedding algorithm allowing us to simulate neuromorphic algorithms designed for arbitrary graphs on (commonly supported) neuromorphic crossbar circuits. This embedding has been adapted from a similar scheme used in other contexts [9]. The embedding cost is nontrivial, and in the worst case it adds a linear multiplicative factor to the running time of our neuromorphic algorithms. This is because two adjacent nodes in the input graph may have to communicate using a long path in a grid-like neurmorphic circuit, incurring communication cost proportional to the path length.

Our grid-like data storage and movement assumption takes a different form for conventional algorithms. Local data movement within a CPU core is typically ignored as an $O(1)$ execution-time cost, although this is technically not true [1]. Yet, the type of data movement leveraged for our neuromorphic graph algorithms occurs at the same intra-core scale, hence we consider it explicitly as an algorithmic cost for both platforms. Local data movement can be energy intensive for conventional systems, while neuromorphic hardware is extremely energy efficient in moving data. Data movement is worth consideration because of its overall impact; indeed, the standard $O(n^2)$ algorithm for computing a matrix-vector product with an $n \times n$ matrix becomes $O(n^3)$ if data movement is taken into account in a model similar to ours, while a neuromorphic implementation remains an $O(n^2)$ algorithm [1].

Data-movement lower bounds. We propose a data-movement model for conventional serial algorithms and provide lower bounds for implementations of the best-known conventional shortest-path algorithms within it. These bounds arise from the assumption that memory is laid out in 2-d (or perhaps a constant number of 2-d layers), which we believe is reasonable for contemporary memory systems. Most current commercial neurmorphic systems use the same basic electronic technologies as conventional systems. If data-movement costs are ignored, then our neuromorphic algorithms, which critically rely on data movement to perform computation, exhibit even more advantage. In our data-movement model, a conventional algorithm takes $\Omega(m^{3/2})$ time just to read an $O(m)$-sized input. Our neuromorphic k-hop shortest-path algorithms exhibit an advantage against any conventional algorithm that only reads the input graph, where the best-known conventional algorithms take k iterations over the input (precise assumptions given in Table 1). This illustrates the severity of data-movement bottlenecks and why data-movement-efficient computation, such as neuromorphic computing, is important. Future neuromorphic systems might be analog (as the brain) or designed to mitigate data-movement costs (as conventional systems do with memory hierarchies).

1.2 Neuromorphic circuits for shortest paths

A directed network of leaky integrate-and-fire neurons and synaptic connections form a spiking *circuit*. Directed loops (recurrent

Table 1: Comparison of run-time complexities of neuromorphic and conventional algorithms for single-source shortest path (SSSP) problems. The number of nodes and edges are denoted by n and m, respectively. L is the length of the shortest k -hop path, U is an upper bound on the edge lengths, and α is the number of edges in the shortest s-t path. The m in the lower bounds can be replaced with the total number of bits in the input, taking edge lengths into account.

Problem	Data-movement lower bound for only reading input	Data-movement lower bound for best-known conventional algorithm	Neuromorphic	Neuromorphic is better than best conventional when:
Polynomial Complexity				
SSSP	$\Omega(m^{3/2})$	$\Omega(m^{3/2})$	$O(m\alpha \log(nU))$	$\log U = O(\log n)$ & $\alpha = o(\sqrt{m})$
k-hop SSSP	$\Omega(m^{3/2})$	$\Omega(km^{3/2})$	$O(mk \log^2(nU))$	$\log U = O(\log n)$
Pseudopolynomial Complexity				
SSSP	$\Omega(m^{3/2})$	$\Omega(m^{3/2})$	$O(nL + m)$	$L = o(m^{3/2}/n)$
k-hop SSSP	$\Omega(m^{3/2})$	$\Omega(km^{3/2})$	$O(mL \log^2 k)$	$L = o(k\sqrt{m}/\log^2 k)$

connections) are allowed, making spiking circuits more general than threshold-gate circuits. Circuit computation is initiated by the simultaneous spiking of a designated set of *start* neurons. Execution proceeds for a fixed amount of time or until a designated *terminal* neuron first spikes. The output is typically the state of the set of *output* neurons at termination. For a more precise abstract model, see the recent work of Kwisthout and Donselaar [5].

Delays and synchronization. We assume that there is minimum programmable delay, δ, a hardware-specific constant. A synapse from neuron i to j has delay $d_{ij} = l\delta$, for a positive integer l. Delays of 0 are prohibited, as inherent latency when a spike traverses a synapse is a reasonable physical assumption respecting data-movement costs. Delays can be simulated using extra neurons in SNAs that do not natively support delays.

Spikes and packets. Instead of sending a single spike between neurons, we abstractly send a packet of information. We replace each neuron with c copies and send c spikes in parallel to represent a binary value.

Neuromorphic memory. Our algorithms store information using neurons with no leakage or recurrent loops to preserve state.

Neuromorphic computational primitives. Our algorithms use basic computational primitives on packets, such as summing values or taking the minimum or maximum over several packets. We develop implementations of such circuits offering new tradeoffs.

Our algorithm for finding the lengths of the shortest k-hop paths from a source s builds on a pseudopolynomial-time SGA for computing regular single-source shortest paths [3]. In our algorithm, each synapse has delay proportional to the length of the input-graph edge it represents. Communication along a synapse involves packets of $\lceil \log k \rceil$ spikes that encode a *time to live (TTL)*. At the start, the node corresponding to s sends $\lceil \log k \rceil$ spikes to each neighbor encoding the value $k-1$. If node v receives a spike bundle encoding the value k' at time t, then there is a path from source s to node v of length t that traverses $k - k'$ edges.

For each non-source node v, the length of the shortest k-hop-constrained path from s (if it exists) is the time the first spike arrives. Nodes propagate spikes with remaining time to live to their neighbors. Spikes that arrive at node v after the first, representing longer paths to v, could have traversed fewer edges and have a longer TTL. These could propagate further in the graph than previous paths with shorter TTL. If multiple spikes arrive at a node at the same time, the one with the largest TTL dominates the others as a building block for other k-hop-constrained shortest paths. Thus at each node, the circuit computes the largest TTL k' from any of the incoming spikes, and sends a spike encoding $k' - 1$ to all its neighbors if and only if $k' \geq 1$. This yields a pseudopolynomial algorithm; a polynomial algorithm is obtained by instead using packets to encode and maintain the length of a shortest path from s and executing k iterations of propagating packets.

The algorithms of this section only compute the length of the optimal shortest single-source (k-hop) paths. Constructing the paths requires the algorithms to store additional information at each graph node. For the k-hop algorithms, the extra storage requires a multiplicative factor of $O(k)$ additional neurons.

REFERENCES

[1] Agarwal, S., Quach, T.-T., Parekh, O., Hsia, A. H., DeBenedictis, E. P., James, C. D., Marinella, M. J., and Aimone, J. B. Energy scaling advantages of resistive memory crossbar based computation and its application to sparse coding. *Frontiers in neuroscience 9* (2015).

[2] Aimone, J. B. Neural algorithms and computing beyond moore's law. *Communications of the ACM 62*, 4 (2019), 110–110.

[3] Aimone, J. B., Parekh, O., Phillips, C. A., Pinar, A., Severa, W., and Xu, H. Dynamic programming with spiking neural computing. In *Proceedings of the International Conference on Neuromorphic Systems* (New York, NY, USA, 2019), ICONS '19, Association for Computing Machinery.

[4] Azevedo, F. A. C., Carvalho, L. R. B., Grinberg, L. T., Farfel, J. M., Ferretti, R. E. L., Leite, R. E. P., Jacob Filho, W., Lent, R., and Herculano-Houzel, S. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *The Journal of Comparative Neurology 513*, 5 (Apr. 2009), 532–541.

[5] Kwisthout, J., and Donselaar, N. On the computational power and complexity of Spiking Neural Networks. *arXiv:2001.08439 [cs]* (Jan. 2020). arXiv: 2001.08439.

[6] Moore, S. Intel's Neuromorphic System Hits 8 Million Neurons, 100 Million Coming by 2020 - IEEE Spectrum. https://spectrum.ieee.org/tech-talk/artificial-intelligence/embedded-ai/intels-neuromorphic-system-hits-8-million-neurons-100-million-coming-by-2020.

[7] Parekh, O., Phillips, C. A., James, C. D., and Aimone, J. B. Constant-depth and subcubic-size threshold circuits for matrix multiplication. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures* (2018), ACM, pp. 67–76.

[8] Schuman, C. D., Potok, T. E., Patton, R. M., Birdwell, J. D., Dean, M. E., Rose, G. S., and Plank, J. S. A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963* (2017).

[9] Thompson, C. D. Area-time complexity for VLSI. In *Proceedings of the eleventh annual ACM symposium on Theory of computing* (Atlanta, Georgia, USA, Apr. 1979), STOC '79, Association for Computing Machinery, pp. 81–88.