This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

SAND2021-7286C

Sandia National Laboratories

# Provable Advantages for Graph Algorithms in Spiking Neural Networks

James B. Aimone, Yang Ho, Ojas Parekh, Cynthia Phillips, Ali Pinar, William Severa, Yipu Wang

U.S. DEPARTMENT OF ENERGY NNSA

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & 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.

# Neuromorphic computing

- Computing devices inspired by the human brain
- Artificial neurons communicate with each other by sending spikes along synapses
- Examples: TrueNorth (IBM), Loihi (Intel), SpiNNaker (U. Manchester), Neurogrid (Stanford), BrainScales (U. Heidelberg)

# Neuromorphic computing

- Originally intended for AI/machine learning
- Neurons generalize threshold gates and Boolean gates, so neural networks can simulate conventional algorithms with polynomial overhead
- Unclear if there's an advantage over conventional computing

# Our results

- Simple distributed algorithms can be implemented neuromorphically with small loss of efficiency
- Give a model for analyzing the resulting neuromorphic algorithms and comparing to conventional algorithms
- Neuromorphic algorithms are sometimes faster than conventional algorithms in this model

# Our results

Assume non-negative integer edge lengths

| Problem | Neuromorphic | Conventional |
|---|---|---|
| Shortest $v_s - v_t$ path | $O(nL + m)$ | $\Omega(m^{3/2})$ |
| Shortest $k$-hop $v_s - v_t$ pathc | $O((nL + m)\log k)$<br>$O((nk + m)\log(nU))$ | $\Omega(km^{3/2})$ |

- $L$ is distance from $v_s$ to $v_t$, $U$ is length of longest edge
- Lower bounds take *data-movement cost* into account
- $k$-hop lower bound is for the best-known algorithm, not for the problem
- Compare with serial algorithms because neurons are more like gates than CPUs w.r.t. scalability

# Our results

Ignoring data-movement cost:

| Problem | Neuromorphic | Conventional |
|---|---|---|
| Shortest $v_s - v_t$ path | $O(L + m)$ | $O(m + n \log n)$ |
| Shortest $k$-hop $v_s - v_t$ path | $O((L + m) \log k)$<br>$O(m \log(nU))$ | $O(km)$ |

- Neuromorphic algorithms also speed up
- $L$ is distance from $v_s$ to $v_t$, $U$ is length of longest edge

# Our results

**Theorem:** There is a neuromorphic $(1 + o(1))$ –approximation algorithm for $k$-hop SSSP that runs in $O((kn \log n + m) \log(kU \log n))$ time (or in $O((k \log n + m) \log(kU \log n))$ time when data-movement is ignored)

- Based on a known CONGEST algorithm [Nanongkai '14]
- Uses fewer neurons than exact algorithm

# Leaky-integrate and fire neurons

- Each neuron $j$ starts with a voltage of $v_{j,0}$
- Voltage updates based on decay and synaptic inputs

$$\hat{v}_j(t+1) = [v_j(t) - (v_j(t) - v_{j,0})\tau_j] + v_{j,syn(t)}$$

- If voltage exceeds a threshold then neuron spikes/fires and voltage resets

$$f_j(t+1) = \begin{cases} 1 : \hat{v}_j(t+1) \geq v_{j,threshold} \\ 0 : \hat{v}_j(t+1) < v_{j,threshold} \end{cases}$$

$$v_j(t+1) = \begin{cases} v_{j,reset} : \hat{v}_j(t+1) \geq v_{j,threshold} \\ \hat{v}_j(t+1) : \hat{v}_j(t+1) < v_{j,threshold} \end{cases}$$

- Each synapse between neurons $i$ and $j$ has weight $w_{ij}$ and delay $d_{ij}$

$$v_{j,syn(t)} = \sum_{i=1}^{n} (f_i(t+1-d_{ij})w_{ij}$$

# Spiking neural network model

- Initial voltages $v_{j,0}$, decay rates $\tau_j$, threshold voltages $v_{j,threshold}$, weights $w_{ij}$, delays $d_{ij} \geq 1$ are all programmable
- To start computation, a set of start neurons *spike*
- Computation ends after fixed amount of time or a terminal neuron spikes
- Output is state of output neurons
- Network of neurons/synapses is fixed, but assume for now that it is programmable

# Neuromorphic SSSP algorithm

Setup:

- Given $G, v_s$, construct neuron/synapse network to mimic $G$
- Set all decays $\tau_j$ to 0 (doesn't matter)
- Set all initial voltages $v_{j,0}$ to 0
- Set all threshold voltages $v_{j,threshold}$ to 1
- Set all weights $w_{ij}$ to 1
- Set delay $d_{ij}$ to be length $l(ij)$ in $G$

Execution

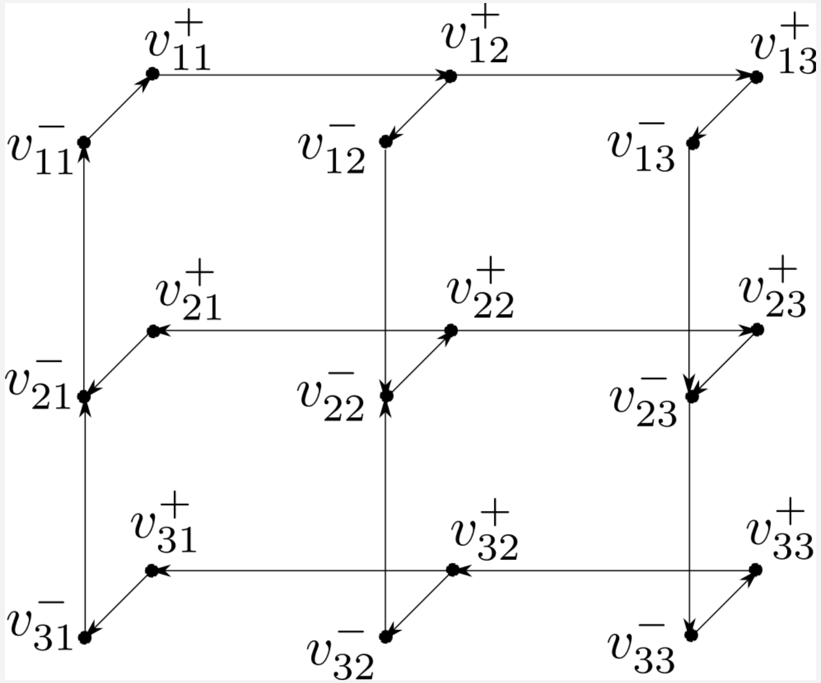
- At the start, $v_s$ sends a spike to each neighbor
- Each neuron retransmits each spike it receives to each of its neighbors
- Terminate when $v_t$ has received a spike

# Correctness and running time of SSSP algorithm

- Neuron $v$ first receives a spike at time $t$ iff $v$ is at distance $t$ from $v_s$
- The first time at which $v_t$ receives a spike is the answer
- $O(n+m)$ time to setup, $O(L)$ time to execute

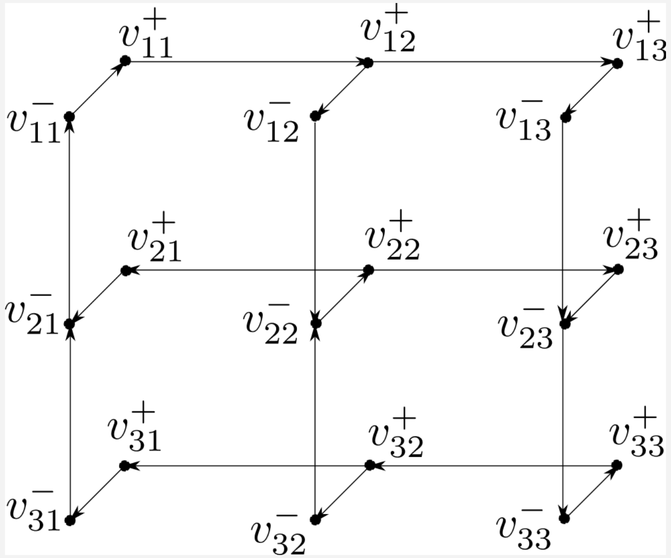
# Embedding problem

- In reality, the neuron/synapse graph is fixed
- Assume the neuron/synapse graph is a crossbar
- Need to “embed” input graph into crossbar



# The embedding

- Intuition: the $i$-th vertex maps to the induced subgraph on $v_{1i}^-, \ldots, v_{ni}^-, v_{i1}^+, \ldots, v_{in}^+$
- Edge $ij$ in $G$ corresponds to arc $v_{ij}^+ v_{ij}^-$ in the crossbar
- Give each edge $v_{ij}^+ v_{ij}^-$ delay $l(ij) - 2|i - j| - 1$, all other arcs unit delay
- **Lemma:** The length of the path from $v_{ii}^-$ to $v_{jj}^-$ is $l(ij)$



# Dilation

- Recall we set each edge $v_{ij}^{+}v_{ij}^{-}$ to have delay $l(ij) \ - \ 2|i-j| \ - \ 1$
- This means $l(ij) > \ 2|i-j| + \ 1$ for all edges $ij$
- So need to scale all edge lengths until shortest edge has length $2n$
- Blows up execution time by factor $O(n)$. Running time for SSSP goes from $O(L \ + \ m \ + \ n)$ to $O(nL \ + \ m)$

# Messages

- Instead of sending a single spike, send a multi-bit message
- For each neuron $v$, add $\lceil \log \lambda \rceil$ copies $v_1, \ldots, v_{\lceil \log \lambda \rceil}$
- When $u$ sends spike to $v$, send up to additional $\lceil \log \lambda \rceil$ spikes in parallel from $u_1, \ldots, u_{\lceil \log \lambda \rceil}$ to $v_1, \ldots, v_{\lceil \log \lambda \rceil}$ to communicate a value between 0 and $\lambda$ in binary

# $k$-hop SSSP algorithm

High-level description

- Ignore embedding problem for now, setup phase the same
- Instead of sending a spike from vertex $u$ to $v$, send $\lceil \log k \rceil$ spikes in parallel from $u_1, \ldots, u_{\lceil \log k \rceil}$ to $v_1, \ldots, v_{\lceil \log k \rceil}$ encoding a time-to-live (TTL) between 1 and $k$ in binary
- $v_s$ sends spikes with TTL's of $k$.
- A vertex receiving spikes takes the highest TTL $k'$ and sends $k' - 1$ to all its neighbors, if $k' > 1$
- Answer is time when $v_t$ first receives a message

Correctness

- If a vertex $v$ receives a spike packet with TTL of $k'$ at time $t$, then there is a path of length $t$ with $\leq k - k' + 1$ arcs from $v_s$ to $v$

# $k$-hop SSSP algorithm

To finish, we need to:

- Describe for each vertex, threshold circuit to subtract 1
  - Just add $2^k - 1$. Circuit has $O(\log k)$ depth and $O(\log k)$ neurons
- Describe for each vertex $v$, threshold circuit to take the max of many numbers. Circuit has $O(\log k)$ depth and $O(\text{indeg}(v) \log k)$ neurons. Details omitted.
- Take into account embedding cost

# Running time of $k$-hop SSSP algorithm

Ignoring embedding cost

- $O(\log k)$-depth circuit for taking max
- $O(\log k)$-depth circuit for decrementer
- Thus $O(L \log k)$ for spiking portion
- Circuits computing max have $O(\mathbf{indeg}(v) \log k)$ neurons for vertex $v$, so total $O(m \ \log k)$ neurons, so loading time is $O(m \ \log k)$
- Total $O((m + L) \ \log k)$ running time

With embedding cost

- Spiking portion now takes $O(nL \log k)$ time
- Total $O((m + nL) \ \log k)$ running time

# DISTANCE model

- Memory is made up of disk and registers
- Data must be moved to a register for any operation, including reading
- Memory comprises lattice points in the plane
- Each lattice point can hold one data value, some lattice points are registers
- Distances are Manhattan distances
- Movement cost is the total distance that data moves

# Lower bound

Lemma: Suppose there are $O(1)$ registers and the input has size $m$. Any algorithm that reads the entire input must incur $\Omega(m^{\frac{3}{2}})$ movement cost.

Proof: Suppose one register and input data arranged in a $\sqrt{m}$ by $\sqrt{m}$ square. "Best-case scenario" is put the register in the middle. The average data point is distance $\Theta(\sqrt{m})$ from the register and thus incurs $\Theta(\sqrt{m})$ movement cost to be read.

# $k$-hop lower bound

A lower bound on the following algorithm:

- Let $dist_k(v)$ be the $(\leq k)$-hop distance from $v_s$ to $v$.
- $dist_0(v_s) = 0$, $dist_0(v) = \infty$ for all $v \neq v_s$
- In $i$-th round, relax all edges $uv$ to find $dist_i(v)$

$$dist_i(v) = \min\{dist_{i-1}(v), dist_{i-1}(u) + l(uv)\}$$

# $k$-hop lower bound

Lemma: If $O(1)$ registers, then algorithm incurs $\Omega(km^{\frac{3}{2}})$ movement cost

Proof: Each round involves relaxing all edges. Thus each round has $\Omega(m^{\frac{3}{2}})$ movement cost.

# Summary

## With data-movement cost

| Problem | Neuromorphic | Conventional |
|---|---|---|
| Shortest $v_s - v_t$ path | $O(nL + m)$ | $\Omega(m^{3/2})$ |
| Shortest $k$-hop $v_s - v_t$ pathc | $O((nL + m)\log k)$<br>$O((nk + m)\log(nU))$ | $\Omega(km^{3/2})$ |

## Without data-movement cost

| Problem | Neuromorphic | Conventional |
|---|---|---|
| Shortest $v_s - v_t$ path | $O(L + m)$ | $O(m + n\ \log n)$ |
| Shortest $k$-hop $v_s - v_t$ path | $O((L + m)\log k)$<br>$O(m\log(nU))$ | $O(km)$ |