SwapGraph: Biologically Plausible Shortest Path Planning on Real-World Graphs

Pallavi Das and Anup Das
Department of Electrical and Computer Engineering
Drexel University
Philadelphia, USA
Email: {pallavi.das, anup.das}@drexel.edu

***Abstract*—We present SwapGraph, a biologically plausible shortest path algorithm that extends neuromorphic path planning to directed acyclic graphs with heterogeneous costs. Our approach replaces Euclidean distance-based predecessor selection with weight-augmented temporal scoring, implementing Bellman's optimality principle using using only spike arrival times and learned axonal delays. E-Prop-based online learning adapts these axonal delays through local spike-timing information without global cost knowledge. We implement graph representation using CSR-CSC sparse format, maintaining separate encodings for outgoing neighbors during spike propagation and incoming neighbors during path extraction. Evaluation on nine benchmark datasets ($168K$–$24M$ nodes) demonstrates 100% optimal path accuracy while consuming 5–8% of memory required by conventional representations, with 13–202$\times$ runtime improvements. These results establish the first biologically plausible shortest path algorithm for large-scale sparse DAG topologies.**

***Index Terms*—Spiking neural networks, graph algorithms, CSR-CSC sparse format, NetworkX, neuromorphic computing, eligibility propagation, cognitive maps, shortest path problem**

I. Introduction

Path planning is fundamental to robotics, autonomous navigation, and network optimization. Traditional algorithms such as Dijkstra's and A* rely on global cost information and priority queue operations that, while efficient on conventional hardware, require centralized edge weight access throughout execution, which present challenges for neuromorphic deployment where computation is distributed and event-driven [1].

Spiking neural networks (SNNs) offer an alternative paradigm aligned with neuromorphic architectures through sparse, asynchronous spike-based communication. Spiking Wavefront Propagation with E-Prop learning (SWAP), inspired by hippocampal replay phenomena, provides biologically plausible path planning through local spike-timing relationships rather than global cost comparisons [2]. However, existing implementations remain constrained to regular grid topologies with geometric assumptions that break down for networked systems with irregular connectivity and heterogeneous costs.

Extending spiking wavefront planning to sparse directed graphs introduces representational challenges impacting neuromorphic compatibility. Adjacency matrices require $O(n^2)$ memory regardless of edge density, while dictionary-based structures introduce pointer overhead and non-contiguous access patterns. For networks with million nodes but sparse connectivity, its representation wastes substantial memory, problematic for memory-constrained neuromorphic systems.

This paper presents two contributions. First, SwapGraph generalizes spike wavefront planning to directed acyclic graphs (DAGs) with heterogeneous edge costs through weight-augmented temporal scoring following Bellman's optimality principle. Second, we implement DAG representation using CSR-CSC format, providing efficient access to both outgoing neighbors (spike propagation) and incoming neighbors (path extraction) while eliminating zero-storage overhead.

II. Related Works

Classical path planning algorithms such as Dijkstra's algorithm, A*, and D* compute optimal routes using priority queues and heuristics. While A* guarantees optimality given admissible heuristics, computational complexity increases with search space size. Sampling-based methods like RRT* offer asymptotic optimality but require numerous iterations [3]–[5].

SNNs offer event-driven computation suited for neuromorphic hardware [6]. Hwu et al. [7] introduced spiking wavefront propagation encoding traversal costs in axonal delays, demonstrating TrueNorth compatibility. Krichmar et al. [2] integrated E-Prop learning [8] for online adaptation, and Espino et al. [9] validated this on mobile robots with continual learning.

However, existing approaches use regular grid topologies with uniform costs, yielding suboptimal paths under heterogeneous traversal costs. We address this using SwapGraph.

III. Background

This section establishes the foundational SNN framework for path planning following [2], [7].

A. Spiking Neural Network Model

The algorithm employs a simplified Leaky Integrate-and-Fire (LIF) model [10]. Each neuron i maintains membrane potential v_i, recovery variable u_i, and synaptic input I_i. The membrane potential evolves as:

$$v_i(t+1) = u_i(t) + I_i(t+1) \tag{1}$$

Following a spike ($v_i(t) \geq \theta$, threshold $\theta = 1$), the recovery variable implements refractory dynamics:

$$u_i(t+1) = \begin{cases} \beta & \text{if } v_i(t) \geq \theta \\ \min(u_i(t)+1, 0) & \text{otherwise} \end{cases} \tag{2}$$

where $\beta = -5$ prevents immediate re-activation.

Synaptic input is determined by spike arrivals from presynaptic neighbors, where delay counter $d_{ij}(t)$ tracks spike propagation:

$$I_i(t+1) = \sum_{j \in \mathcal{N}(i)} \mathbf{1}[d_{ij}(t) = 1] \tag{3}$$

$$d_{ij}(t+1) = \begin{cases} D_{ij} & \text{if } v_j(t) \geq \theta \\ \max(d_{ij}(t) - 1, 0) & \text{otherwise} \end{cases} \tag{4}$$

The axonal delay D_{ij} encodes traversal cost through temporal delays rather than synaptic weights, inspired by activity-dependent myelination [11].

B. Spike Wave Propagation and Learning

The algorithm initiates a spike at the start neuron, propagating as a traveling wave governed by learned delays D_{ij}, with events recorded in an Address Event Representation (AER) table [12]. Learning proceeds through E-Prop [8]: active neurons maintain eligibility traces $e_i(t+1) = 1$ if $v_i(t) \geq \theta$, else $e_i(t) - e_i(t)/\tau$. Upon goal arrival, delays update as $w_{ij}(t+1) = w_{ij}(t) + \delta \cdot e_i(t) \cdot (c_{ij} - w_{ij}(t))$, providing temporal credit assignment where neurons firing closer to goal receive stronger updates.

C. Grid-Based Path Extraction Limitations

In grid-based formulations, path extraction backtracking identifies predecessors among earlier-firing neurons within spatial proximity ($\|\cdot\|_2 \leq 1.5$ grid units), with ties resolved by minimum cost and proximity to start.

Critical Limitation. With heterogeneous edge costs $c_{ij} \in [c_{\min}, c_{\max}]$, spatial tie-breaking disregards learned costs per hop, preventing guaranteed optimal cost paths. Furthermore, dense connectivity representation requires $O(n^2)$ storage, inefficient for large sparse graphs.

IV. Methodology

SwapGraph extends SWAP to DAGs with heterogeneous costs while maintaining neuromorphic compatibility. Fig. 1 provides a high-level overview of the pipeline.

A. Data Representation

We employ dual compressed sparse representations: CSR for outgoing edges and CSC for incoming edges. Fig. 2 compares the three representation approaches, illustrating the memory efficiency advantages of CSR-CSC format over adjacency matrices and nested dictionaries.

For graph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges, the outgoing CSR consists of `indptr_out` (size $n+1$) indexing outgoing edges, `indices_out` (size m) containing destinations, `cost_out` (size m) with true costs c_{uv}, and `wgt_out` (size m) with learned delays w_{uv}. For node u the outgoing neighbour set is $\mathcal{N}_{\text{out}}(u) = \{v : (u, v) \in E\}$, reducing storage from $O(n^2)$ to $O(n+m)$. Path extraction requires incoming neighbours, so a complementary CSC maintains `indptr_in`, `indices_in`, and position mapping `pos_in_to_out` linking CSC to CSR positions: $w_{ji} = W[\pi(k)]$, where $\pi(k)$ maps CSC position k to its CSR counterpart. Unlike grid maps that associate costs with nodes, our formulation assigns costs to directed edges, capturing realistic scenarios in network routing and transportation.

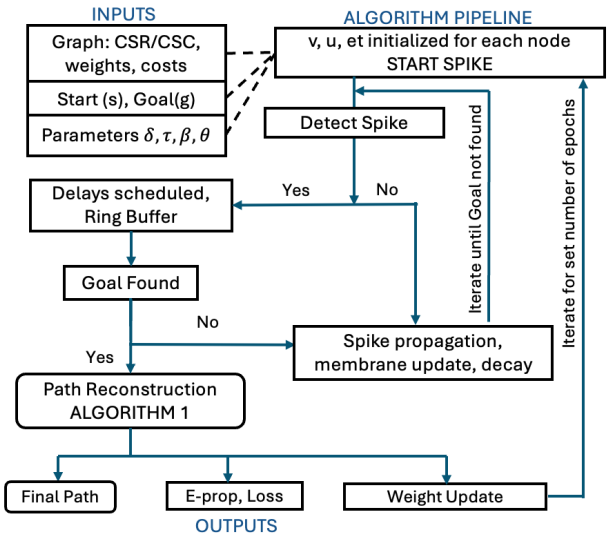

Fig. 1: SwapGraph system pipeline.

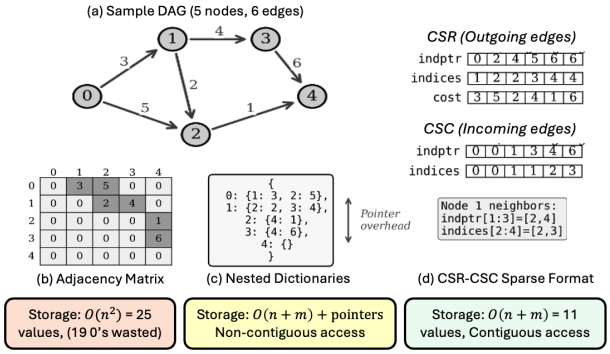

Fig. 2: Graph representation formats for a sample DAG with 5 nodes and 6 edges. (a) Adjacency matrix (b) Nested dictionaries: non-contiguous memory access. (c) CSR-CSC format: contiguous memory access.

B. Propagation Mechanism

Spike propagation uses event-driven ring buffers that track only active transmissions, eliminating dense delay matrices. When neuron u fires at time t, arrivals are scheduled for each outgoing edge by rounding the learned weight to an integer delay and computing its arrival time:

$$\begin{aligned} d_{uv} &= \lfloor w_{uv} \rceil \\ t_{\text{arrival}} &= t + d_{uv} - 1. \end{aligned} \tag{5}$$

The scheduling pipeline is shown in Fig. 3. The ring buffer of size $O(D_{\max})$ maintains pending arrivals with tracking structure:

$$\tau_p = \begin{cases} t_{\text{arrival}} & \text{if edge } p \text{ has pending spike} \\ -1 & \text{otherwise.} \end{cases} \tag{6}$$

At each timestep t, for each pending edge with $\tau_p = t$ and $w_p > 0$:

$$I_{\text{exc}}[v] \leftarrow I_{\text{exc}}[v] + 1, \tag{7}$$

reducing memory from $O(n^2)$ to $O(n + m + D_{\max})$. LIF dynamics (1) then apply with CSR-CSC determined connectivity. Path extraction takes place once the goal neuron fires. Since

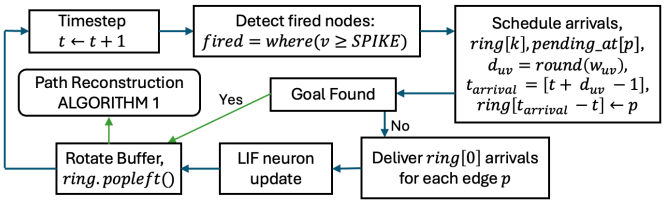

Fig. 3: Ring Buffer Scheduler

learned axonal delays converge to true edge costs through E-Prop, selecting the predecessor that minimises the sum of its spike time and connecting delay implements Bellman's optimality criterion through spike timing alone, without a global priority queue. From the AER table we construct a spike-time map $\mathbf{T} \in \mathbb{Z}^n$:

$$T_i = \begin{cases} t & \text{if neuron } i \text{ fired at timestep } t \\ -1 & \text{otherwise,} \end{cases} \tag{8}$$

and backtrack from goal to start using structural connectivity and temporal information without spatial heuristics. For current node i during backtracking, each candidate $j \in \mathcal{N}_{\text{in}}(i)$ receives score combining spike time with learned delay.

$$s_j = T_j + w_{ji}, \tag{9}$$

Since w_{ji} converges to c_{ji} through E-Prop, this implements Bellman's principle: $d^*(s \to i) = d^*(s \to j) + c_{ji}$. The optimal predecessor minimizes:

$$j^* = \arg\min_{j \in \mathcal{N}_{\text{in}}(i)} (T_j + w_{ji}), \tag{10}$$

with constraints $T_j \geq 0$ (fired) and $w_{ji} > 0$ (connected).

Algorithm 1 formalises the full procedure, constructing the path from goal-to-start then reversing.

Algorithm 1 Graph-Compatible Path Extraction

Require: Spike time map $\mathbf{T}$, weight matrix $\mathbf{W}$, start node s, goal node g
Ensure: Optimal path from s to g
Initialize: path $\leftarrow [g]$, current $\leftarrow g$
while current $\neq s$ **do**
  neighbors $\leftarrow \{j \mid j \in \mathcal{N}_{\text{in}}(\text{current}),\ T_j \geq 0,\ w_{j,\text{current}} > 0\}$
  if |neighbors| $= 0$ **then**
    break *// No valid predecessors found*
  end if
  scored $\leftarrow [(j,\ T_j + w_{j,\text{current}})$ for $j \in$ neighbors$]$
  $(j^*, s^*) \leftarrow \arg\min_{(j,s) \in \text{scored}} s$
  path.append(j^*); current $\leftarrow j^*$
end while
return path

C. Learning Algorithm

For each node u on the extracted path, weight updates apply across all outgoing edges $(u, v) \in E$ with $w_{uv} > 0$:

$$w_{uv}(t+1) = w_{uv}(t) + \delta \cdot e_u(t) \cdot \big(c_{uv} - w_{uv}(t)\big), \tag{11}$$

where δ is the learning rate and $e_u(t)$ is the eligibility trace of node u at update time t. This per-node update propagates cost information to all neighbours of each visited node simultaneously, accelerating convergence across the graph while remaining strictly local. Classical algorithms require global cost access, priority queues, and dense memory operations incompatible with event-driven neuromorphic execution [13]–[17]; Table I consolidates how SwapGraph replaces these with $O(|E|)$ per-query spike timing, $O(D) \ll O(|V|)$ parallel depth [16], local-only adaptation [18]–[20], and $O(|S|)$ event-driven energy scaling [13].

We establish convergence and optimality under uniform initialisation ($w_{uv}(0) = 1$) with edge costs $c_{uv} \in [1, c_{\max}]$.

Proposition 1 (Monotone Convergence). The E-Prop update (11) yields $w_{uv}(t+1) = w_{uv}(t)\big(1 - \delta \cdot e_u(t)\big) + c_{uv} \cdot \delta \cdot e_u(t)$. Since $\delta \in (0, 1)$ and $e_u(t) \in (0, 1]$ for path neurons, this is a convex combination satisfying $1 \leq w_{uv}(t) \leq w_{uv}(t+1) \leq c_{uv}$. Defining the error $\epsilon_{uv}(t) = w_{uv}(t) - c_{uv}$, substitution gives $\epsilon_{uv}(t+1) = \epsilon_{uv}(t)\big(1 - \delta \cdot e_u(t)\big)$. Under uniform initialisation $\epsilon_{uv}(0) = 1 - c_{uv} \leq 0$, thus preserving sign and reducing magnitude without oscillation. From the eligibility trace dynamics (Section III-B), neuron u firing at time T_u has trace $e_u(T_g) = (1 - 1/\tau)^{T_g - T_u}$ at goal-arrival time T_g. Since $T_g - T_u \leq D_{\max}$ for all path neurons, where $D_{\max}$ is the maximum propagation depth in quantised time steps, we define $e_{\min} := \left(1 - \frac{1}{\tau}\right)^{D_{\max}} > 0$ which is fully computable from system parameters τ and $D_{\max}$. Because (11) applies to all outgoing edges of each path node, every edge incident to a path node is updated whenever that node is visited, ensuring $e_u(t) \geq e_{\min}$ at every update step. This yields the convergence bound after k active path episodes: $\big|\epsilon_{uv}(k)\big| \leq (c_{\max} - 1)\big(1 - \delta \cdot e_{\min}\big)^k$. Setting it $\leq \varepsilon$ and solving for k, all edges incident to path nodes satisfy $|w_{uv}(k) - c_{uv}| \leq \varepsilon$ after at most

$$k^* = \left\lceil \frac{\log\left(\frac{c_{\max} - 1}{\varepsilon}\right)}{\log\left(\frac{1}{1 - \delta \cdot e_{\min}}\right)} \right\rceil \tag{12}$$

active path episodes, where an episode *active* for edge (u, v) if node u appears on the extracted path in that episode.

Theorem 1 (Post-Convergence Optimality). If $w_{ij} = c_{ij}$ for all $(i, j) \in E$, Algorithm 1 extracts the shortest path. Since G is acyclic, the wavefront visits nodes in topological order with $T_s = 0 = d^*(s, s)$. Under converged weights, $T_i = \min_{j \in \mathcal{N}_{\text{in}}(i)}(T_j + c_{ji}) = d^*(s, i)$ by induction via Bellman's principle [21]. Backtracking selects (10) at each node, recovering the optimal path. Since the optimal path stabilizes after k^* episodes, all incoming edges of path nodes converge to true costs under (11), thereby, selecting the correct predecessor without the convergence of non-path edges.

V. Experimental Setup & Results

A. Benchmark Datasets

We evaluate SwapGraph on nine datasets spanning three topologies (Table I): six scale-free networks ($168K$-$1.16M$ nodes, $CV = 1.17$-14.69), two random road networks ($1.09M$-$23.9M$ nodes, $CV = 0.37$), and one regular Delaunay mesh ($16.8M$ nodes, $CV = 0.22$). This diversity spans two orders of magnitude in scale and topological extremes from hub-dominated to regular structures [22].

B. Evaluation Metrics

We assess SwapGraph across four dimensions: (1) *Memory Efficiency*: (CSR-CSC vs. NetworkX); (2) *Computational Performance*: (construction time, GFLOPS); (3) *Path Optimality*: versus Dijkstra ground truth; and (4) *Neuromorphic Metrics*: (spikes, SynOps, active neuron ratio) [23]–[27]. All 9 datasets were evaluated across 100 start-goal pairs each; Table V reports the results (15 pairs, 4 datasets) and Fig. 5 shows the cost gap on WebBerkStan as a representative case, with SwapGraph achieving 100% optimality across all 900 trials.

TABLE I. Dataset. γ denotes the power-law exponent [28]

Dataset	N	E	T	CV	γ	Density	Avg Deg
TwitchGamers	168K	6.8M	SF	3.885	1.97	2.41×10^{-4}	80.87
Amazon-0312	401K	2.3M	SF	1.259	3.95	1.46×10^{-5}	11.73
coPaperDBLP	540K	15.2M	SF	1.174	2.14	5.22×10^{-5}	56.41
WebBerkStan	685K	6.6M	SF	14.69	2.03	1.42×10^{-5}	19.41
WebGoogle	916K	4.3M	SF	4.117	2.05	5.15×10^{-6}	9.43
Youtube	1.2M	3.0M	SF	9.738	1.72	2.23×10^{-6}	5.16
RoadPA	1.1M	15.5M	R	0.363	4.53	1.30×10^{-5}	28.35
Delaunay-n24	16.8M	50.3M	Reg	0.223	5.88	1.79×10^{-7}	6.00
RoadUSA	23.9M	28.9M	R	0.386	2.72	5.03×10^{-8}	2.41

N: Nodes, E: Edges, T: SF=Scale-Free, R=Random, Reg=Regular

C. Baseline Methods

NetworkX (NX) with nested dictionary representation; Dijkstra's algorithm as ground truth; SWAP [2] as the baseline spiking wavefront algorithm limited to grid topologies.

D. Implementation Details

SwapGraph was implemented in Python 3.12 using NumPy 1.26.4 for CSR-CSC sparse matrix operations, vectorized computations, Pandas 2.2.2 for data management, as well as NetworkX 2.6 for graph operations, and 64-bit floating-point precision for all weight and cost calculations.

Edge costs were sampled uniformly from the range $[1, 10]$ to simulate heterogeneous traversal conditions. Network hyperparameters: learning rate $\delta = 0.25$, eligibility trace decay $\tau = 25$, refractory period $\beta = -5$, spike threshold $\theta = 1$. The network was trained for 20 episodes per start-goal pair to ensure weight convergence before evaluation.

Graph Preprocessing for DAG Conversion. Several benchmark datasets in our evaluation (YouTube, WebGoogle, TwitchGamers, road networks) are originally undirected. These datasets are converted to DAGs using standard acyclic orientation [29] for compatibilty with our formulaton: edges directed by node index ordering, with back-edges removed for cyclic graphs [30], [31]. Table II reports the DAG statistics.

VI. Results

A. Memory Efficiency and Scalability

The CSR-CSC sparse representation achieves substantial memory reductions compared to NetworkX's nested dictionary structure across all evaluated datasets (Table II, Fig. 4). Memory factors range from $13.33\times$ to $20.06\times$ (Table II), with the advantage increasing for sparser graphs due to reduced per-edge pointer overhead.

For the largest evaluated network (RoadUSA), CSR-CSC requires only 1.03 GB compared to NetworkX's 20.6 GB, thereby, enabling deployment on resource-constrained edge devices. The second-largest network (Delaunay-n24) consumes 1.41 GB with CSR-CSC representation.

TABLE II. Memory Consumption Comparison

Dataset	CSR-CSC(MB)	NetworkX(MB)	Factor(NX/CSR)
TwitchGamers	158.15	2,108.21	$13.33\times$
Amazon-0312	59.90	870.91	$14.54\times$
coPaperDBLP	357.19	4,785.22	$13.40\times$
WebBerkStan	162.65	2,271.80	$13.97\times$
WebGoogle	112.91	1,685.40	$14.93\times$
Youtube	86.05	1,391.89	$16.18\times$
RoadPA	51.94	977.83	$18.83\times$
Delaunay-n24	1,407.00	26,733.00	$19.00\times$
RoadUSA	1,025.83	20,579.18	$20.06\times$

Memory consumption per node scales proportionally to average degree rather than total node count (Fig. 4), confirming the theoretical $O(m+n)$ space complexity. This contrasts with the $O(n^2)$ or $O(m+n\cdot\log(m))$ overhead of adjacency matrix or nested dictionary representations.

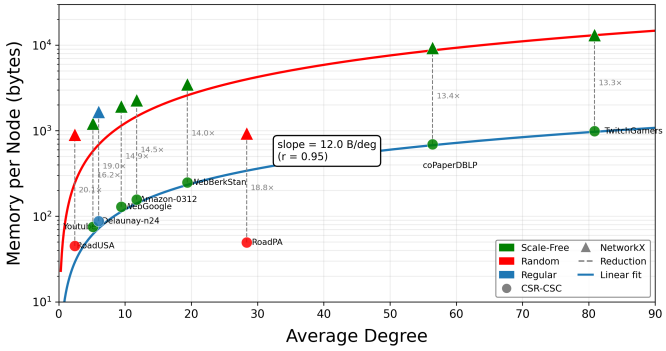

Fig. 4: Memory consumption per node scales proportional to average degree (r = 0.95).

Importantly, memory factor correlates inversely with graph density: denser graphs exhibit smaller memory advantages while ultra-sparse graphs achieve maximum compression. This relationship reflects the elimination of both zero-storage overhead and per-edge pointer indirection in CSR-CSC format.

B. Computational Performance

Graph construction time demonstrates advantages for CSR-CSC over NetworkX (Table III). Speedup factors range from $12.96\times$ to $201.68\times$, with median speedup of $94.99\times$. This improvement stems from contiguous array allocation and sequential memory writes in CSR format, versus scattered pointer allocation and hash operations in NetworkX. Construction speedup varies nonlinearly with graph size: medium-scale graphs achieve maximum advantage (Table III), while the

largest graphs show reduced but still substantial speedup. This pattern reflects cache hierarchy effects—CSR construction maintains cache locality across all scales, while NetworkX's scattered allocations degrade at different rates depending on graph topology. Average training time per path query ranges

TABLE III. Runtime Performance

Dataset	CSR-CSC(s)	NetworkX(s)	Speedup(NX/CSR)
TwitchGamers	0.495	27.318	55.14×
Amazon-0312	0.082	16.619	201.68×
coPaperDBLP	0.318	47.453	149.43×
WebBerkStan	2.810	266.909	94.99×
WebGoogle	0.039	7.114	180.48×
Youtube	1.505	19.511	12.96×
RoadPA	0.374	14.821	39.66×
Delaunay-n24	6.467	319.56	49.48×
RoadUSA	8.197	196.923	24.02×

from 0.045 s to 2.318 s across datasets, correlating with both graph size and path length. Training time scales sublinearly with path length, reflecting eligibility trace propagation mechanics: longer paths accumulate weight updates across more nodes, but updates concentrate along optimal trajectories rather than exploring the full graph.

Computational efficiency (Table IV) exceeds NetworkX by two orders of magnitude. High GFLOPS variance reflects path-dependent computational load: sparse paths require minimal propagation while hub-traversing paths incur higher activity.

TABLE IV. Computational Efficiency (GFLOPS)

Dataset	CSR-CSC Avg ($\times 10^{-1}$)	Peak	Min	NX Avg ($\times 10^{-3}$)
Youtube	5.47 ± 5.84	1.468	0.032	4.7 ± 0.9
Amazon-0312	12.76 ± 2.01	1.876	0.718	3.0 ± 0.5
WebBerkStan	8.68 ± 3.93	1.420	0.229	4.0 ± 0.9
WebGoogle	8.93 ± 4.18	1.524	0.237	3.9 ± 0.7
TwitchGamers	9.42 ± 3.78	1.599	0.363	3.5 ± 0.6
coPaperDBLP	6.98 ± 2.93	1.118	0.288	4.1 ± 0.8
RoadPA	8.03 ± 3.95	1.644	0.156	3.7 ± 0.7
Delaunay-n24	7.01 ± 0.19	0.743	0.627	3.8 ± 0.8
RoadUSA	7.42 ± 0.24	0.804	0.675	3.5 ± 0.7

C. Path Optimality Guarantees

All nine datasets were evaluated across 100 start-goal pairs each; Table V reports representative results (15 pairs, 4 datasets) and Fig. 5 illustrates the cost gap on WebBerkStan as a representative case, with SwapGraph achieving 100% optimality across all 900 trials. In contrast, SWAP achieves optimality in only 50% of test cases (30/60 paths), with average cost gap of 66.5% (max 285.7%), stemming from spatial tie-breaking that prioritizes hop count over cost.

TABLE V. Path Optimality Results (15 seeds per dataset)

Dataset	SwapGraph (Proposed)	SWAP (Baseline)	Avg Gap (%)	Max Gap (%)
WebBerkStan	15/15 (100%)	7/15 (47%)	100.9	200.0
RoadPA	15/15 (100%)	11/15 (73%)	24.3	37.5
TwitchGamers	15/15 (100%)	4/15 (27%)	40.6	100.0
coPaperDBLP	15/15 (100%)	8/15 (53%)	91.8	285.7
Total	**60/60 (100%)**	**30/60 (50%)**	**66.5**	**285.7**

D. Neuromorphic Compatibility

Neuromorphic deployment feasibility is characterized by sparse and event-driven dynamics, reported as average spikes and synaptic operations per path query in (Table VI, Fig. 6). Active neuron ratios range from 0.08% to 15.68%, with the majority of networks exhibiting $< 1\%$ activation (Table VI). This extreme sparsity directly translates to energy

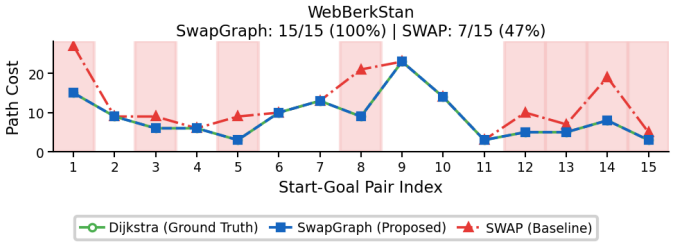

Fig. 5: SwapGraph matches Dijkstra (100% optimality); SWAP exhibits cost gaps (avg. 100.9%, max 200%)

efficiency on neuromorphic hardware such as Intel Loihi or IBM TrueNorth, where power consumption scales linearly with spike count [6], [13], [15]. For comparison, conventional artificial neural networks typically exhibit approximately 50% activation in hidden layers [32], representing a 20-750× higher computational overhead. As observed in Fig. 6(a), Random and Regular networks maintain consistently low activation due to uniform degree distributions, while Scale-Free networks exhibit high variance depending on hub traversal.

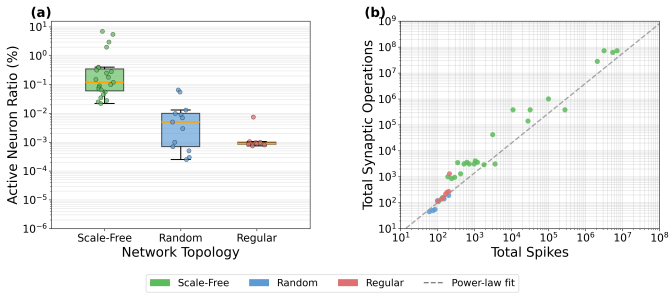

Fig. 6: Neuromorphic metrics. (a) Active ratio by topology. Random/Regular achieve 10^{-5}–10^{-3}%, Scale-Free shows higher variance. (b) Power-law fan-out amplification. Neuromorphic activation metrics across network topologies. Each point represents one path query.

The high variance in spike counts (Table VI) reflects path-length and topology dependence: short paths through sparse regions generate minimal activity, while paths traversing ultra-hub nodes incur higher overhead.

TABLE VI. Neuromorphic Metrics (5 seeds per dataset)

Dataset	Avg Spikes	Avg SynOps	Active Ratio (%)
Youtube	$1.82\text{M} \pm 2.07\text{M}$	$9.89\text{M} \pm 11.49\text{M}$	15.68
Amazon-0312	$3{,}362 \pm 6{,}224$	$7{,}371 \pm 12{,}984$	0.08
WebBerkStan	$72{,}533 \pm 96{,}685$	$542\text{K} \pm 733\text{K}$	0.99
WebGoogle	$33{,}635 \pm 51{,}459$	$81\text{K} \pm 120\text{K}$	0.36
TwitchGamers	$63{,}179 \pm 113{,}376$	$142\text{K} \pm 235\text{K}$	3.70
coPaperDBLP	$17{,}961 \pm 24{,}857$	$29\text{K} \pm 39\text{K}$	0.33
RoadPA	$27{,}625 \pm 52{,}093$	$48\text{K} \pm 86\text{K}$	0.25
Delaunay-n24	346 ± 465	$637 \pm 1{,}014$	<0.01
RoadUSA	100 ± 46	80 ± 47	<0.01

Synaptic operations scale superlinearly with spike count *within a single query*, following a fitted power-law relationship [33] that reflects fan-out amplification: each spike triggers multiple synaptic events at downstream neurons, with amplification factor dependent on local connectivity [34]. Notably, SynOps *per query* scale sublinearly *with graph size* (YouTube vs. WebBerkStan in Table VI). This efficiency gain arises from

wavefront focusing on optimal paths that typically traverse sparse peripheral regions, avoiding exhaustive exploration of dense neighborhoods, such that larger graphs do not proportionally increase per-query computational cost.

Energy efficiency on neuromorphic hardware scales approximately as $E \propto \alpha \cdot \text{SynOps}$, where $\alpha \approx 23$ pJ per synaptic operation for Intel Loihi [13]. This event-driven energy model contrasts with conventional architectures where energy scales with total memory accesses regardless of data sparsity [6], yielding estimated energy consumption of 10–1000 μJ per path query [35]–[37]. Fig. 7 illustrates energy consumption scaling where per-query energy ranges from 0.001 μJ for sparse Random networks to approximately 400 μJ for hub-dominated Scale-Free networks, with SynOps values annotated above each bar. Scale-free networks exhibit higher variance in active ratios due to ultra-hub domination paths crossing high-degree hubs activating extensive neighborhoods, even while peripheral paths remain sparse. Random networks maintain consistently low activation due to uniform degree distributions.

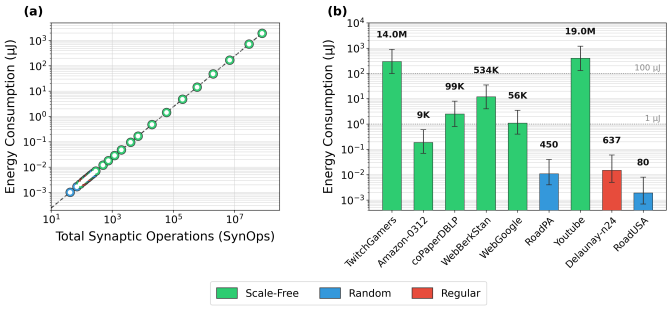

Fig. 7: Energy consumption analysis for neuromorphic deployment. (a) Energy vs. Total SynOps shows proportional scaling. (b) Per-query energy by dataset with mean SynOps annotated

VII. Conclusion

We presented SwapGraph, a biologically plausible shortest path algorithm that extends neuromorphic wavefront planning from regular grids to directed acyclic graphs with heterogeneous edge costs, through weight-augmented temporal scoring. The CSR-CSC representation achieves $O(n + m)$ space complexity with efficient accesses to neighbors. Evaluation in nine datasets ($168K$–$24M$ nodes) demonstrates 100% optimal accuracy while consuming 5% to 8% of NetworkX's memory with 13-202$\times$ speedup. By integrating biologically plausible learning with optimality guarantees, SwapGraph establishes a foundation for scalable neuromorphic graph algorithms.

Acknowledgment

This work is supported by the U.S. Department of Energy under Award Number DE-SC0022014 and the U.S. National Science Foundation under Grant Nos. CCF-1937419

References

[1] C. Zhao *et al.*, "Mutual information and trajectory-aware reinforcement learning for multi-agent path finding," in *IJCNN*, 2025.
[2] J. L. Krichmar *et al.*, "Flexible path planning in a spiking model of replay and vicarious trial and error," in *SAB*, 2022.
[3] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 2022.
[4] P. E. Hart *et al.*, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, 1968.
[5] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, 2011.
[6] K. Roy *et al.*, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, 2019.
[7] T. Hwu *et al.*, "Adaptive robot path planning using a spiking neuron algorithm with axonal delays," *IEEE Trans. Cogn. Dev. Syst.*, 2017.
[8] G. Bellec *et al.*, "A solution to the learning dilemma for recurrent networks of spiking neurons," *Nat. Commun.*, 2020.
[9] H. Espino *et al.*, "A rapid adapting and continual learning spiking neural network path planning algorithm for mobile robots," *IEEE RAL*, 2024.
[10] L. Lapicque, "Recherches quantitatives sur l'excitation électrique des nerfs," *J. Physiol. Pathol. Gen.*, 1907.
[11] R. D. Fields, "A new mechanism of nervous system plasticity: Activity-dependent myelination," *Nat. Rev. Neurosci.*, 2015.
[12] M. Mahowald, "VLSI analogs of neuronal visual processing: A synthesis of form and function," Ph.D. dissertation, Caltech, 1992.
[13] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, 2018.
[14] E. Painkras *et al.*, "SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation," *IEEE JSSC*, 2013.
[15] F. Akopyan *et al.*, "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE TCAD*, 2015.
[16] U. Meyer and P. Sanders, "δ-stepping: A parallelizable shortest path algorithm," *J. Algorithms*, 2003.
[17] S. Beamer *et al.*, "Locality exists in graph processing: Workload characterization on an Ivy Bridge server," in *IISWC*, 2015.
[18] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, 1988.
[19] P. Dayan and T. J. Sejnowski, "TD(λ) converges with probability 1," *Mach. Learn.*, 1994.
[20] F. Zenke and S. Ganguli, "SuperSpike: Supervised learning in multilayer spiking neural networks," *Neural Comput.*, 2018.
[21] R. Bellman, "On a routing problem," *Quart. Appl. Math.*, 1958.
[22] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, 1999.
[23] D. Casanueva-Morato *et al.*, "Space-time smooth control of closed-loop neuromorphic robotic arms using snn," in *IJCNN*, 2025.
[24] M. P. E. Apolinario *et al.*, "TESS: A scalable temporally and spatially local learning rule for spiking neural networks," in *IJCNN*, 2025.
[25] R. V. W. Putra *et al.*, "Enabling efficient processing of spiking neural networks with on-chip learning on commodity neuromorphic processors for edge AI systems," *arXiv preprint arXiv:2504.00957*, 2025.
[26] E. Ressa *et al.*, "TinyCL: An efficient hardware architecture for continual learning on autonomous systems," in *IJCNN*, 2025.
[27] J. Ghawaly *et al.*, "Exploring spiking neural networks for binary classification in multivariate time series at the edge," in *IJCNN*, 2025.
[28] A. Clauset, "Power-law distributions in empirical data," *SIAM*, 2009.
[29] J. Leskovec, "SNAP datasets: Stanford large network dataset collection," 2014, http://snap.stanford.edu/data.
[30] Y. Wang, A. Davidson *et al.*, "Gunrock: A high-performance graph processing library on the GPU," in *PPoPP*, 2016.
[31] Y. Wang *et al.*, "Gunrock: GPU graph analytics," *ACM TOPC*, 2017.
[32] X. Glorot, "Deep sparse rectifier neural networks," in *AISTATS*, 2011.
[33] M. E. J. Newman, "Power laws, pareto distributions and Zipf's law," *Contemp. Phys.*, 2005.
[34] B. Yin *et al.*, "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks," *Nat. Mach. Intell.*, 2021.
[35] E. Lemaire *et al.*, "Synaptic activity and hardware footprint of spiking neural networks in digital neuromorphic systems," *ACM TECS*, 2022.
[36] J. Kim *et al.*, "Efficient synapse memory structure for reconfigurable digital neuromorphic hardware," in *ISCAS*, 2018.
[37] P. Blouw *et al.*, "Benchmarking keyword spotting efficiency on neuromorphic hardware," in *NICE*, 2019.