DS-GL: Advancing Graph Learning via Harnessing Nature's Power within Scalable Dynamical Systems

Ruibing Song[†], Chunshu Wu[†], Chuan Liu[†], Ang Li[‡], Michael Huang[†], Tony (Tong) Geng[†]

† University of Rochester, Rochester, NY, USA
‡ Pacific Northwest National Laboratory, Richland, WA, USA
{ruibing.song, chunshu.wu, chuan.liu, michael.huang, tong.geng}@rochester.edu, ang.li@pnnl.gov

***Abstract*—With the rapid digitization of the world, an increasing number of real-world applications are turning to non-Euclidean data, modeled as graphs. Due to their intrinsic high complexity and irregularity, learning from graph data demands tremendous computational power. Recently, CMOS-compatible Ising machines, i.e., dynamical systems fabricated with CMOS technologies, have emerged as a new approach that harnesses the inherent power of nature within dynamical systems to efficiently resolve binary optimization problems and have been adopted for traditional graph computation, such as max-cut. However, when performing complex Graph Learning (GL) tasks, Ising machines face significant hurdles: (i) they are binary and thus ill-suited for real-valued problems; (ii) their expensive all-to-all coupling network that guarantees generality for optimization problems poses daunting scalability concerns.**

To address these challenges, this paper proposes a nature-powered graph learning framework dubbed DS-GL, which is the first effort to transform the process of solving graph learning problems into the natural annealing process within a parameterized dynamical system embodied as a CMOS chip. To tackle the two major hurdles, DS-GL first augments the Ising machine architecture to modify the self-reaction term of its Hamiltonian function from linear to quadratic, effectively serving as an energy regulator. This adjustment maintains the system's original physical interpretation while enabling it to process continuous, real-valued data. Second, to address the scaling issue, DS-GL further upgrades the real-valued dense Ising machine by decomposing it into a mesh-based multi-PE dynamical system that supports efficient distributed spatial-temporal co-annealing across different PEs through sparse interconnects. By exploiting the inherent sparsity and community structures in real-world graphs, DS-GL is able to map complex graph learning tasks onto the scalable dynamical system while maintaining high accuracy. Evaluations with four diverse GL applications across seven real-world datasets, including traffic flow and COVID-19 prediction, show that DS-GL can deliver from $10^3\times$ to $10^5\times$ speedups over Graph Neural Networks on GPUs while operating at a power 2 orders of magnitude lower than GPUs, with $5\%-30\%$ accuracy enhancement.

***Index Terms*—Dynamical System, Graph Learning, Nature-Powered Computing**

I. Introduction

As the world experiences rapid informatization and digitalization, a growing number of applications are turning to non-Euclidean data with high complexity and best represented as graphs. These applications span a broad spectrum of critical areas, including power grid cascading failure prediction, traffic flow management, and pandemic forecasting, among many others. Most of these applications pose stringent demands on real-time processing, low energy consumption, and simultaneously high accuracy. Meeting these demands presents substantial challenges, primarily due to the intrinsic complexity and irregularity of graph data [12].

Graph Neural Networks (GNNs), as the current State-Of-The-Art (SOTA) Graph Learning (GL) approach, have recently drawn tremendous attention due to their strong capability to extract latent information from graph data. Following years of advancements in algorithms [4], [8], [11] and the development of specialized hardware accelerators [12], [30], [42], [43], GNN research is transitioning into an era of practical application. The pursuit of exceptional accuracy has spurred the emergence of numerous application-specific GNNs [34], [35], characterized by their rapidly escalating algorithmic complexity and the consequent need for tremendous computational power. Historically, the fast growth in computational power of digital hardware, driven by Moore's Law, has facilitated the pursuit of enhanced accuracy through trading model complexity while maintaining computational efficiency. Regrettably, the approaching end of Moore's Law has cast a shadow over the future advancement and real-world adoption of GNNs. While the optimization of digital processors remains a vital area of exploration, it is equally imperative to investigate alternative computational paradigms and assess their potential in advancing the field of GL.

The recent development of CMOS-compatible Ising machines [2], coupled with their impressive efficacy in addressing traditional graph problems (e.g., max-cut), suggests that the intrinsic power of nature within dynamical systems can be potentially exploited in GL. Specifically, an Ising machine is a parameterized dynamical system governed by the Hamiltonian of the Ising model[1]. As a dynamical system, the machine naturally evolves towards fixed points. Combined with proper annealing control, the machine has the effect of autonomously seeking states of lowest energy — a phenomenon termed as

Ruibing Song and Chunshu Wu contributed equally.

[1]Ising model: a binary statistical physics model widely used to represent spin glasses; Hamiltonian: the energy function of dynamical systems.

natural annealing. When the parameters of the Ising machine are properly configured, lowest energy states correspond to the desired solutions of an optimization problem. Thus the machine has the effect of "solving" these problems through natural annealing with extraordinarily energy efficiency and at "the speed of electrons". Take *graph cut* as an example, a $\sim$200 mW Ising machine can perform high-quality max-cut delivering orders of magnitude speedup over 200 W GPUs.

The remarkable capabilities of modern Ising machines present a compelling question: Can we leverage the natural computational power of such dynamical systems to advance graph learning by simultaneously offering clearly lower latency, better energy efficiency, and higher accuracy? Most graph learning problems fundamentally target dynamical systems, e.g., power grid, traffic systems, and supply chain. It is natural to wonder whether a programmable dynamical system can better analyze the behaviors of these dynamical systems than general-purpose processors. Intuitively, the answer is affirmative. Specifically, we can construct a dynamical system whose data distribution aligns with that of the target Graph Learning (GL) problem by training the system's parameters with the problem's training data. Accurate alignment ensures that the system's lowest energy states correspond to the desired GL solutions with the highest probability. Consequently, the dynamical system can swiftly conduct GL inference through natural annealing at negligible cost. The statistical basis underpinning this method is analogous to that of modern generative AIs, such as stable diffusion [21].

Unfortunately, in practice, the full potential of Ising machines in real-world GL cannot be realized until two major challenges are addressed. (1) ***Binary Nature:*** Ising machines focus only on binary values (Ising spins), limiting their applications in real-valued contexts. Efforts [15] to circumvent this limitation by using multiple binary nodes to represent high-precision values not only compromise solution quality but also necessitate a substantial increase in the required number of nodes of the machine, exacerbating the second problem – scalability. (2) ***Poor Scalability:*** To enhance versatility and solution quality, SOTA Ising machines use all-to-all connections with n^2 couplers connecting n nodes through a huge crossbar, ensuring direct and immediate interactions between any two nodes during annealing. However, this design leads to scalability issues. Attempts [31], [40] to improve scalability using partially connected interconnects with uniform patterns, such as King's graph topology (where each physical node connects with eight neighbors), fall short in handling high-degree nodes, a common occurrence in real-world graphs. In light of this, a new **Dynamical-System-based Processing Unit (DSPU)** – capable of supporting real-valued natural annealing and scalable for GL applications – is highly desired.

To this end, this paper proposes a novel Dynamical-System-based Graph Learning framework, **DS-GL**, which incorporates a real-valued and scalable DSPU coupled with a series of training algorithms for constructing efficient and scalable dynamical systems for GL problems. The overview of DS-GL is illustrated in Fig. 1. Specifically, to enable stable real-valued natural annealing, DS-GL first upgrades the SOTA Ising machine hardware with a circulative resistor ring and the self-reaction term of its corresponding Hamiltonian (Sec.III.A); then, a training algorithm (Sec.III.B) that accurately configures the system is developed. The algorithm transforms the process of solving GL problems into a process of natural annealing. Coupled with this algorithm, the new ***"Real-Valued DSPU"*** can perform real-valued GL with high performance and accuracy. To enhance the scalability, DS-GL further upgrades Real-Valued DSPU into a larger system, dubbed ***"Scalable DSPU"***, with a mesh-based multi-PE architecture that efficiently supports distributed spatial-temporal co-annealing (Sec.IV.D); Coupled with a learning-based clustering algorithm (Sec.IV.C) that can reconstruct dense dynamical systems into the sparse ones with community structures while maintaining high accuracy in natural annealing, Scalable DSPU can solve over 4$\times$ larger real-valued GL problems than an Ising machine with only a 30% increase in chip are.

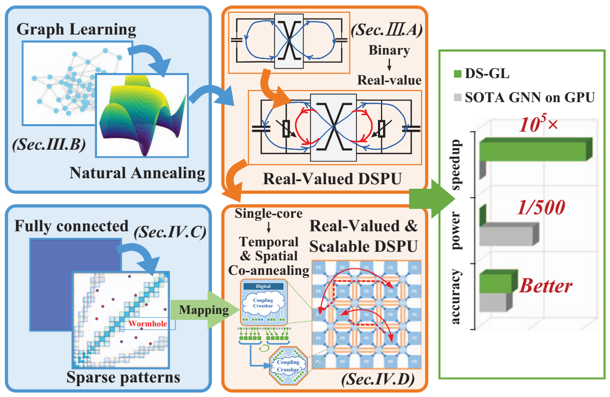

Fig. 1. Overview of DS-GL framework and its performance over GNNs.

To the best of our knowledge, DS-GL is the first work that uses physical dynamical systems and harnesses their intrinsic nature's computational power to solve real-world graph learning problems and outperforms SOTA GNN solutions. The contributions are summarized below:

- We propose a novel nature-powered graph learning framework, DS-GL, that unleashes the inherent computational power of dynamical systems in graph learning.
- We propose learning-based algorithms that accurately transform the process of solving graph learning problems to the natural annealing process of sparse dynamical systems with a hardware-friendly community structure.
- We propose a new dynamic-system-based processing unit, Scalable DSPU, rooted in a CMOS-compatible Ising machine. Scalable DSPU inherits the extraordinary computational efficiency of the Ising machine and extends its potential to real-valued and larger-scale GL problems.
- Experimental results across four real-world applications and seven datasets show that DS-GL achieves from $10^3\times$ to $10^5\times$ speedup and $5\% - 30\%$ higher accuracy over GNNs on GPUs while operating at a power 2 orders of magnitude lower than GPUs.

II. Background

A. Ising Model

The Ising model [6] is a statistical model widely used in the study of physics, chemistry, and biology. The Ising model is defined by its energy function or Hamiltonian:

$$\mathcal{H}_{\text{Ising}} = -\sum_{i \neq j}^{N} J_{ij}\sigma_i\sigma_j - \sum_{i}^{N} h_i\sigma_i \tag{1}$$

where $\sigma_i \in \{-1, +1\}$ represents the spins within the system. J_{ij} is the coupling parameter representing the correlation between spin i and spin j, and h_i refers to the self-reaction strength to external influences.

B. BRIM: the Current SOTA Ising Machine

Ising machines are essentially physical embodiments of the Ising model. Specifically, through their designed spin dynamics, lower energy states of the Ising Hamiltonian are automatically pursued. Besides many existing Ising machines implemented with quantum or optical components, CMOS-based Ising machines have recently emerged and drawn increasing attention due to their low-barrier deployment. One typical example is BRIM [2]. Furthermore, they typically solve problems through the movement of electrons among electronic components such as capacitors, therefore being able to deliver solutions with the "speed of electrons". Despite the many advantages and potential of Ising machines, their unique problem-solving power has only been demonstrated for binary problems. Inspired by the superior performance of BRIM in solving graph optimization problems, we develop DS-GL, which takes the BRIM architecture reported in [2] as a building block. Before delving into DS-GL design, we first briefly introduce the BRIM architecture below.

Fig. 2 shows the overview of the BRIM architecture. The values of nodes are represented by the voltages of capacitors in each N_i block. To facilitate all-to-all connection among nodes, BRIM is equipped with a fully-connected coupling network (the network of J_{ij} blocks) based on programmable resistors. Therefore, the differences between the voltages of nodes will naturally generate currents among the coupling network to reduce the system energy and push the system towards equilibrium. Programming Units are used to program BRIM by configuring the coupling parameters of the network (i.e., the resistance of the programmable resistors). The couplers are programmed column by column controlled by the Column Select Unit. Node Control Unit is in charge of node value initialization and flipping the binary values of nodes at runtime for effective annealing. More details can be found in [2].

C. Graph Learning

In the context, GL refers to the acquisition of unknown graph node features using observed node features. Taking GNNs for example, node features are obtained by iteratively aggregating features from neighboring nodes. During GL training, the spatial and temporal relations among graph nodes are distilled into a selected model (GNNs or DS-GL), which processes observed node features as its input and generates unknown node features as its output. The model's parameters are adjusted (through backward-propagation in GNNs) according to the discrepancies between the generated outputs and the ground truth. This refinement process allows the model parameters to effectively capture the underlying distribution of the data. During inference, the trained model consumes the observed node features and generates the corresponding unknown node features. Particularly, for temporal prediction tasks, GL uses historical graph information to predict the future states of the graph.

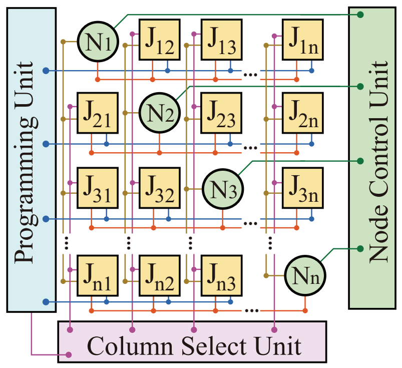

Fig. 2. Overview of BRIM hardware architecture.

III. Real-Valued GL on Dense Dynamical Systems

Despite the exceptional capability of CMOS-compatible Ising machines in solving binary optimization problems like max-cut, the binary limitation hinders the method's further deployment for real-world graph learning problems. However, lifting the binary restriction is not a trivial task, as the adjustment made to this model must be general enough to accommodate real-world problems, and practical enough for hardware implementation.

To enable real-value support, we introduce our modifications applied to the model, together with Real-Valued DSPU, which incorporates the hardware upgrade to the baseline BRIM to establish the basic hardware components for this work.

A. The Binary Limitation and Hardware Upgrade

A naive approach to facilitate real-value support is to directly extend the variables σ from binary to real-value. However, σ do not converge to real values but are polarized towards $\pm\infty$. This polarization can be justified by a stationary point analysis on the Hamiltonian. The stationary points are reached by solving for the following condition:

$$\frac{\partial \mathcal{H}}{\partial \boldsymbol{\sigma}} = -(J\boldsymbol{\sigma} + \boldsymbol{h}) = 0; \quad diag(J) = \mathbf{0} \tag{2}$$

Spins and parameters are shown in their matrix form, with linear substitutions $(J_{ij} + J_{ji}) \rightarrow J_{ij}$, and $2h_i \rightarrow h_i$. Next, the Hessian matrix $\mathbf{H}$ is applied to analyze the properties of stationary points.

$$\mathbf{H}(\mathcal{H}_{\text{Ising}}) = -J = \text{constant}; \quad diag(J) = \mathbf{0} \tag{3}$$

As J is a constant matrix, all stationary points share the same property. If all the eigenvalues of $H(\mathcal{H})|_{\boldsymbol{\sigma}_0}$ are positive,

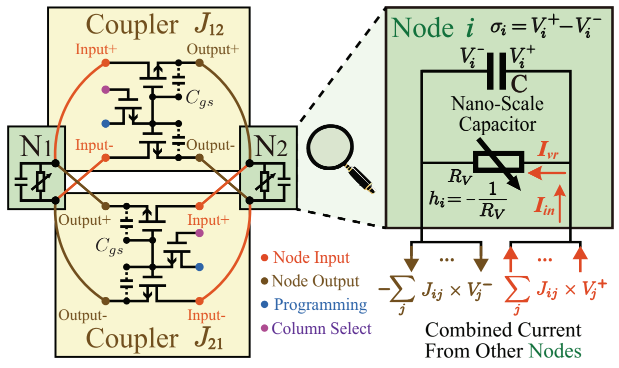

Fig. 3. Real-Valued DSPU architecture. Left: the circulative resistor ring. Right: the detailed node internals for real-value support.

the stationary points $\boldsymbol{\sigma_0}$ are local minima; if all negative, local maxima; otherwise, $\boldsymbol{\sigma_0}$ are saddle points. Notice that $diag(J) = 0$, based on the linear algebraic property $tr(J) = \sum_i^N \lambda_i$, where λ_i is the i'th eigenvalue of J, there is a mixture of positive and negative eigenvalues, thus saddle points. In practice, the saddle points are not stable as they have zero tolerance for fluctuation, leading to diverging spins. Essentially, this divergence is due to $diag(J) = 0$. To compensate, pure quadratic or higher-order terms of σ are necessary.

This can also be viewed in an intuitive way. According to the Ising Hamiltonian (Eq. (1)), the lowest energy state approaches $-\infty$. Even if an upper bound and a lower bound are applied to prevent σ from reaching infinity, the resulting variables are polarized at their upper or lower bound, essentially reducing a real-valued problem to binary.

Algorithm-wise, as a countermeasure to the polarized σ, we modify the original Ising model as Equation 4 demonstrates:

$$\mathcal{H}_{\text{RV}} = -\sum_{i \neq j}^{N} J_{ij}\sigma_i\sigma_j - \sum_{i}^{N} h_i {\sigma_i}^2 \tag{4}$$

In the modified model, only the second term is different – now we use a pure quadratic term to replace the original linear term. In this way, the second term still represents the self-reaction of a spin, preserving the physical interpretation. Meanwhile, the pure quadratic term prevents the variables from diverging given the negative and sufficiently large parameters h, as this modified term contributes to a quadratic increase in energy.

With the model generalized to support real values, the hardware needs enhancements satisfying the following criteria: (1) variables (voltages) must be able to stabilize as real values. (2) the spontaneous decrease in Hamiltonian must be satisfied.

The first criterion is satisfied through the implementation of circulative resistor rings, as depicted in Fig. 3. This setup incorporates the variable resistor R_V within a node along with the pairwise coupling mechanism between nodes to achieve the desired functionality. To accommodate both positive and negative values of J, each pair of nodes is equipped with two circulative resistor rings.

In baseline BRIM, without the resistor regulating the voltage, σ continues to vary whenever there is an incoming current

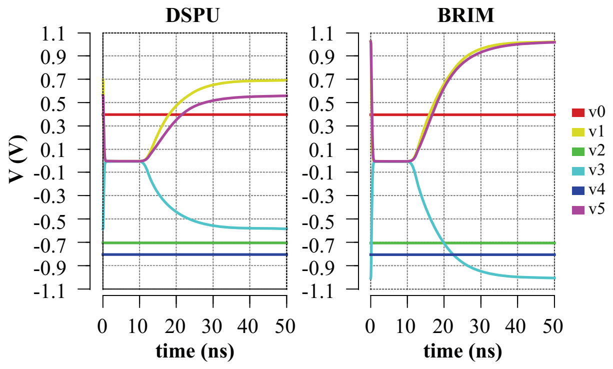

Fig. 4. Circuit-level validation with an example graph.

until the capacitor is fully charged, only to represent polarized values. Now with the presence of the resistor, nonzero currents are enabled to flow consistently through this node, allowing σ to be stabilized at

$$\sigma = I_{vr}R_V = I_{in}R_V = -\sum_{j \neq i} \frac{J_{ij}\sigma_j}{h_i} \tag{5}$$

Similar to J, the parameters h also have the unit of Ω^{-1}, corresponding to the conductance of resistors embedded in the nodes on hardware. Meanwhile, real-world graph nodes are modeled as variables σ, physically implemented as the voltages applied to the nano-scale capacitors.

The stabilization capability is further demonstrated by circuit-level validation. For clarity, we consider an illustrative graph consisting of 6 spins, labeled v0$\sim$v5 in Fig. 4, which are separately deployed on DSPU and BRIM platforms. In this setup, v0, v2, and v4 are predetermined as inputs, leaving the others free to evolve. With identical input and coupling parameters, the DSPU yields real-valued outcomes between the upper and lower bounds, whereas BRIM only produces two polarized values.

To meet the second criterion, the dynamics of the variables is designed through Lyapunov analysis, establishing a dynamical system of electrons that can be deployed on hardware. Accordingly, the following inequality needs to be satisfied:

$$\frac{dH_{\text{RV}}}{dt} = \sum_i \frac{\partial H_{\text{RV}}}{\partial \sigma_i}\frac{d\sigma_i}{dt} \leq 0 \tag{6}$$

Obviously, the variable dynamics can be designed as:

$$\frac{d\sigma_i}{dt} = -\frac{1}{C}\frac{\partial H_{\text{RV}}}{\partial \sigma_i} \tag{7}$$

where C is a positive constant with the unit of capacitance. Substituting this equation into Eq. (6), the quadratic $(\partial H_{\text{RV}}/\partial \sigma_i)^2$ appears, satisfying the target inequality above. The following question is, how to map it on hardware?

In fact, the equation is automatically satisfied with the resistors. Based on textbook capacitor knowledge, we have:

$$\frac{d\sigma_i}{dt} = \frac{1}{C}(I_{in} - I_{vr}) = -\frac{1}{C}\sum_{j \neq i} J_{ij}\sigma_j - h_i\sigma_i \tag{8}$$

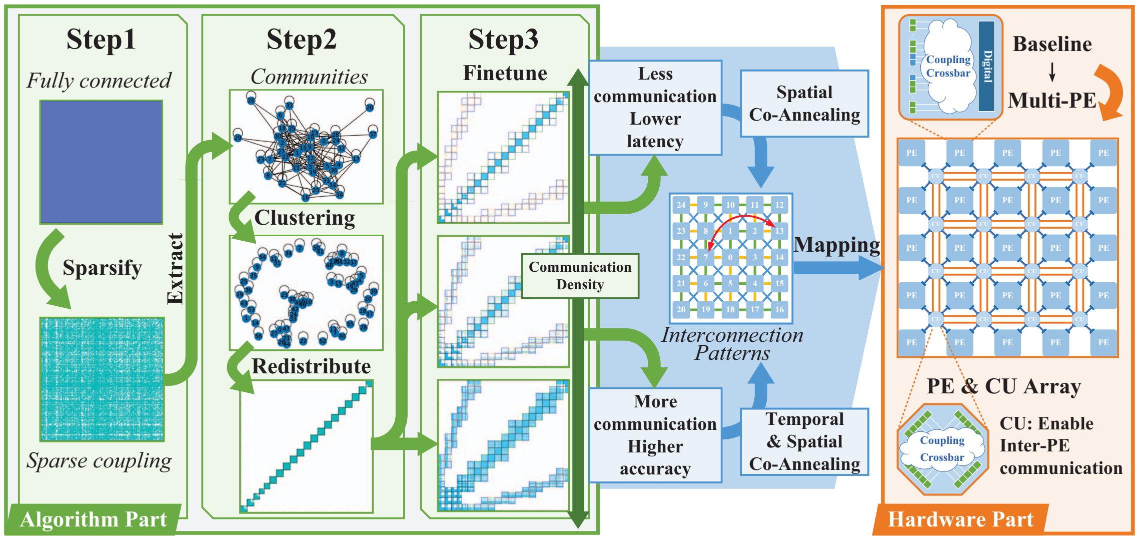

Fig. 5. Workflow of Scalable DS-GL. Blocks in green: algorithm for coupling matrix decomposition. Blocks in orange: the architecture of Scalable DSPU.

which agrees with the shape of $\partial H_{\text{RV}}/\partial\sigma_i$ and effectively facilitates the spontaneous energy decrease.

B. Model Training

To provide a complete view of our proposed work and show how the dynamical system is tamed, we briefly introduce the training algorithm here. The training process aims to obtain a set of parameters J and h that map the desired real-valued result to the lowest energy state, in other words, to construct a data distribution described by a dynamical system. During training, to guarantee the convexity of the Hamiltonian, the parameters h are forced to be negative. Subsequently, the lowest energy state can be obtained by letting the first derivative of the Hamiltonian equal to zero:

$$\frac{\partial\mathcal{H}_{\text{RV}}}{\partial\sigma_i} = -\sum_{i\neq j}^{N}(J_{ij}+J_{ji})\sigma_j - 2h_i\sigma_i = 0 \tag{9}$$

Without losing generality, we substitute $(J_{ij}+J_{ji}) \rightarrow J_{ij}$, and $2h_i \rightarrow h_i$. The regression formula for σ is then derived:

$$\sigma_i = \frac{-\sum_{i\neq j}^{N} J_{ij}\sigma_j}{h_i} \tag{10}$$

which is exactly the hardware stability criterion (Eq. (5)). That is, given the current parameters J/h, and the values of all other variables as conditions, the difference between the computed variable σ_i and its ground truth is used as a loss function, updating the parameters through back-propagation.

C. Inference on a Dynamical System

With the learned parameters, GL inference can be interpreted as the evolution of our dynamical system. The observed graph nodes are considered as input, while the remaining unknown nodes are taken as output. To initiate the inference process, the input observed nodes are fixed to the observations, as the capacitors are charged and maintained accordingly. Meanwhile, the unknown nodes are randomly initialized. Next, the natural annealing process starts and the system approaches equilibrium, so as to locate a lowest energy state.

IV. Scalable GL on Sparse Dynamical Systems

This section tackles the scalability hurdle by co-designing the learning-based algorithm for decomposing dense dynamical systems and the multi-PE dynamical system architecture.

A. Overview of Scalable DS-GL

Despite the communication effectiveness of all-to-all interactions among nodes, the size of the coupling network increases quadratically with the number of nodes. To address the problem of scalability, our design strategy is to prune links based on the strength of inter-node connections, which refers to the magnitude of coupling parameters. Compared to the weakly coupled nodes, we observe that, strongly coupled nodes contribute predominantly to the quality of solution. Considering the fact that real-world graphs are typically extremely sparse with communities composed of strongly-related nodes, it is feasible to only preserve strong connections and relax the weak links between the communities. To accomplish this, as Fig. 5 illustrates, we train DS-GL as a dynamical system with community structure through three steps: (i) prune the fully connected coupling matrix to a sparse matrix depending on the coupling strength; (ii) extract the communities indicated within the sparse matrix, and group the communities into "**super-communities**" to match per-DSPU capacity; (iii) further reform the coupling matrix to fit the desired sparse interconnection pattern. To alleviate communication pressure, different super-communities are interconnected through a sparse hierarchy, including Chain, Mesh, DMesh (Diagonally-connected Mesh [18]), and Wormholes (for unavoidable global communication outliers).

On the hardware side, to provide the foundation for scaling, we design a mesh-based network "Scalable DSPU" as a grid of small DSPUs comprised of Processing Elements (PEs) and Coupling Units (CUs). In essence, each PE serves as a local dynamical system, with neighboring PEs linked through a limited number of analog I/Os via CUs for instantaneous synchronization. In the 2D mesh, communities of nodes are mapped to different DSPUs with their interconnections sparsified into patterns. The patterns are specially designed for efficient "co-annealing" processes upgraded from the annealing concept in Real-Valued DSPU. Furthermore, the co-annealing process is categorized into Spatial co-annealing and Temporal & Spatial co-annealing for different scenarios (detailed in Section IV. D). Consequently, the scalability of Scalable DSPU is optimized with balanced annealing quality and efficiency.

B. Training Algorithm for Decomposing Dynamical System

For our proposed dynamical system, the scalability issue arising from the all-to-all connection can be decomposed into three sub-problems. First, to reduce communication complexity, how to decompose the dynamical system to sparsify the coupling matrix? Second, assuming the coupling is sparse, how to perform computing efficiently? Third, accuracy will drop during the sparsification, how to restore the accuracy? To answer these questions, our solution is also three-fold.

(1) Decomposition of the dynamical system. Communities typically exist in real-world graphs as a valuable property. Similar to cliques in graph theory, communities consist of nodes with dense interconnects but with sparse connections to the external nodes. It can be inferred that although more information is embedded in the original all-to-all node interconnection, the majority of the interconnects should be redundant and removable with minimal consequences.

The key is to extract the communities in the target graphs, which is a well-researched topic. In this work, the Louvain algorithm [5] is adopted due to its high efficiency and scalability. To start, we limit the number of non-zero elements (defined as *"communication demand density"* and annotated as *"D"* in this work) in the coupling matrix in order to attain an initial sparse coupling matrix for communities extraction. In the next steps, after communities are extracted, they are further sparsified by eliminating weak couplings, drastically reducing the demand for communication bandwidth.

(2) Community redistribution. The extracted communities are grouped into super-communities, with each initially distributed to a PE. However, the size of a single community occasionally exceeds the pre-defined hardware capacity of a PE, causing the demand for the community to be further decomposed into smaller sub-communities to fit on hardware. As a consequence, this redistribution process potentially reduces connections within communities, causing accuracy to drop. To make amends, the sub-communities are redistributed onto neighboring super-communities for more communication opportunities. In the meantime, larger communities are granted higher priority to be redistributed. In Fig. 6(a), for example, assuming that the largest community (or a sub-community of the largest community when it exceeds hardware capacity)

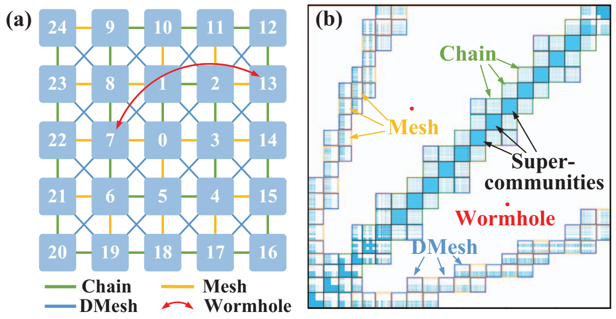

Fig. 6. Four types of communication patterns.

fits into super-community 0, it is centered to have more connections with its neighbors. The second largest community is then distributed to super-community 0 if allowed by capacity, otherwise to super-community 1. Finally, for the sake of a balanced workload, smaller communities or isolated nodes are redistributed to fill the blanks left by larger communities on super-communities. Through these redistribution approaches, the locality of communities is exploited with the utilization of a single super-community enhanced.

(3) Parameter fine-tune with patterns. With the communities extracted and redistributed, we aim to address the final problem – to restore the lost accuracy in these processes. To this end, a fine-tuning process is conducted with constraints to develop a communication-friendly pattern.

To maintain the general coupling matrix pattern obtained from the previous steps, we generate a controlling mask to confine the regions in the coupling matrix where non-zero elements can populate during the fine-tuning process, also eliminating non-zeros outside the region due to the pre-set communication demand density D.

Next, the interconnect pattern of the super-communities is studied. In Fig. 6(a), four patterns are summarized, which respectively correspond to four types of connections between the super-communities on a 2-D array. Fig. 6(b) shows the distribution of the patterns in the re-ordered coupling matrix. The green links represent the "Chain" type of connections between neighbor super-communities such as 0 and 1. The "Mesh" type of patterns contains all the connections between neighbor super-communities on the 2-D array as orange links including the one between 0 and 3, as well as all of the "Chain" type of patterns. The blue links show the additional connections in "DMesh" type of patterns based on "Mesh" which refer to the diagonal connections between super-communities such as 0 and 2. The "Wormhole" in Fig. 6 refers to super-connections over the 2-D array, supporting rare connections between any two super-communities, for example, 7 and 13.

C. Proposed Hardware Architecture

The structurally sparse coupling matrix with clustered non-zeros obtained through the decomposition of the dynamical system (introduced in Sec IV. B) brings opportunities to achieve efficient and accurate natural annealing with highly sparse dynamical systems.

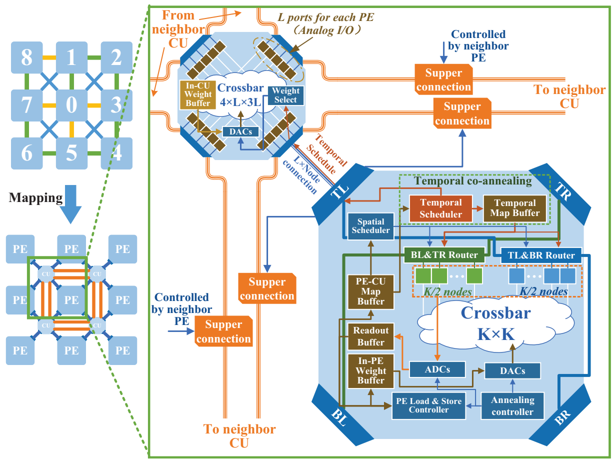

Fig. 7. The hardware architecture of Scalable DSPU.

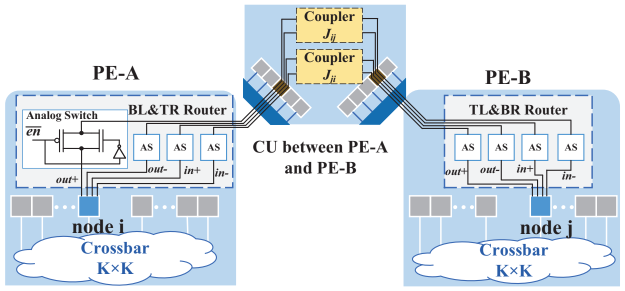

Fig. 8. Detailed PE-to-CU connections via Analog I/O.

To this end, we propose Scalable DSPU, a new dynamical system architecture based on Real-Valued DSPU proposed in Section III. Fig. 7 shows the proposed hardware architecture. Scalable DSPU is equipped with a 2D array of Processing Elements (PEs). Each PE is a small Real-Valued DSPU with additional buffers, routers, and digital controllers for the support of co-annealing. The PEs are connected to a mesh-based network through configurable Coupling Units (CUs) at the intersection of the mesh. Each CU contains a mini coupling crossbar, which can be reconfigured as different types of connections to bridge nodes from the neighbor PEs. During natural annealing, each PE is in charge of the local annealing of a single super-community. Mesh-based interconnect network, together with the configurable CUs, builds direct connections for nodes that are from different PEs but with non-zero coupling parameters. During annealing, voltage differences across node pairs drive currents across different PEs, enabling "Spatial co-annealing". When the number of nodes in a PE that need to communicate with external nodes exceeds the limited input/output capacity of this PE, these nodes will occupy the I/O in a time division multiplexing manner, which is scheduled collaboratively by their PE and the corresponding CUs, enabling the Temporal & Spatial co-annealing. In the following, we will elaborate on the architecture design of each major super-community in Scalable DSPU in detail.

PE architecture: As shown in Fig. 7, each PE contains K nodes (blue and green blocks connected to Routers). All nodes are fully connected through an internal $K \times K$ crossbar coupling network, like in Real-Valued DSPU. Different from Real-Valued DSPU, the nodes are divided into two partitions. Each partition contains k/2 nodes and is connected to either Bottom-Left (BL) & Top-Right (TR) routers or Top-Left (TL) & Bottom-Right(BR) routers. Each router, jointly controlled by Spatial and Temporal Schedulers, is able to route its own share of nodes to its corresponding two neighboring CUs through analog-based exporting portals at the four corners of the PEs. The Spatial Scheduler selects the nodes that need to be connected with external PEs and supervises the corresponding Router to allocate I/O resources at one of the selected exporting portals for the nodes. This builds the foundation of Spatial co-annealing. The Temporal Scheduler is in charge of selecting nodes for temporal co-annealing when the number of nodes that need to communicate with external PEs exceeds the I/O resources (L lanes within each portal) at the exporting portals. Each PE is also equipped with several banks of buffers that cache the communication mapping information generated during training.

CU architecture: CU is at the intersection of the mesh-based network and is used to connect PEs to the network. The coupling parameters in a CU are stored locally in the In-CU Weight Buffer controlled by the Weight Select module. Similarly to PEs, each CU has four exporting portals which connect the CU with four PEs. To align the communication bandwidths of CUs and PEs, each portal in a CU is also equipped with L lanes of connection. Therefore, each CU can be connected with $4L$ nodes in four neighboring PEs simultaneously. Each CU is equipped with a $4L \times 3L$ crossbar connecting all pairs of nodes in different PEs. Note that a CU does not need a $4L \times 4L$ full-size crossbar as the nodes from the same PE are already fully connected locally. With nodes from different PEs directly connected within CUs, their Spatial co-annealing is enabled. Here, we define the number of lanes in each portal of both CU and PE (L) as **hardware communication capability**. In our evaluation, we set L as 30 for better performance and hardware tradeoff.

Interconnect Architecture: The interconnect architecture of Scalable DS-GL is composed of two parts: (1) the connections between exporting portals of CUs and PEs together compose a tiled mesh-based interconnect (the green grid in Figure. 7); and (2) the super connections (orange lines) that connect exporting portals of neighboring CUs compose another grid-based interconnect (the orange grid). As aforementioned, the green grid enables the Spatial co-annealing among nodes from neighboring PEs. In contrast, the yellow grid enables the co-annealing among nodes from remote PEs. From the perspective of the coupling matrix, the scattered non-zeros located in the blank space require remote communication in pursuit of efficient and accurate annealing and therefore require "Wormholes". To open a Wormhole for two nodes from remote PEs, the corresponding PEs first map both nodes to their neighboring CUs and then enable the super connections in the route between the two CUs.

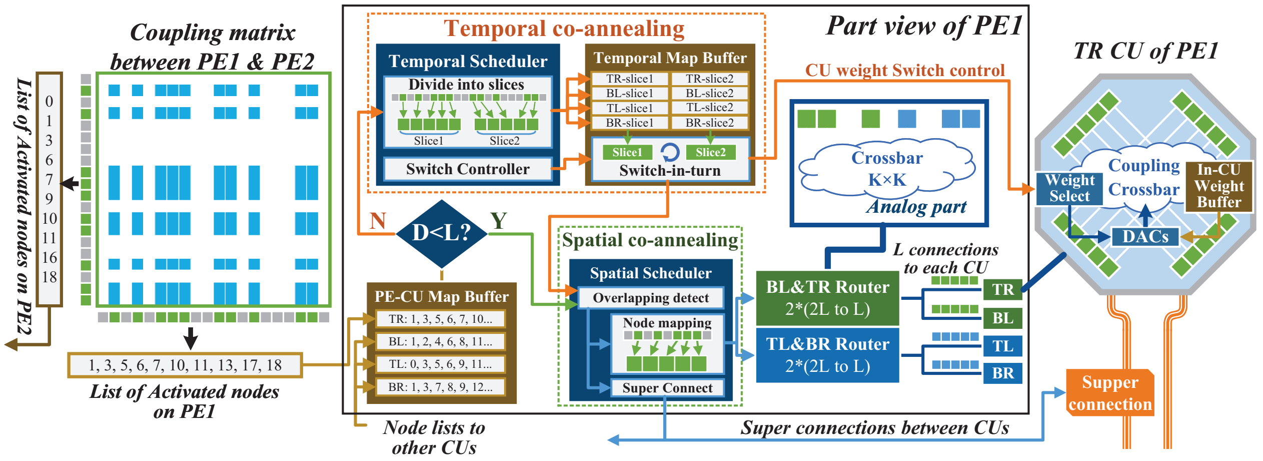

Fig. 9. The hardware architecture for Spatial co-annealing and Spatial & Temporal co-annealing methods.

Analog I/O Details: Fig. 8 shows the signal channel between two nodes from different PEs containing two high-speed analog switches and an analog resistive component, i.e., CU coupling unit. In each PE, a router selects the corresponding analog switches to establish analog connections between nodes in the PEs and ports on the CU, thereby enabling the inter-PE communication via the analog coupling crossbar in CUs. This analog-fashioned connection avoids extra A/D or D/A conversion and fully supports heterogeneous interconnect patterns in Fig. 6, leveraging the flexibility of analog coupling crossbars in CUs and routers in PEs. As shown in Fig. 7, each node in a PE can be connected to up to 4 neighbor CUs. Within each CU, a node is further connected to up to 90 nodes from 3 neighboring PEs through the coupling network.

Challenge in decomposing large-scale graphs: With DS-GL, graphs can be decomposed more aggressively without sacrificing accuracy than GNNs. The underlying reasons are two-fold. First, as an electronic dynamical system, DS-GL hardware constantly propagates node information to their directly connected neighbors through the movement of electrons (flow of electric current) among capacitors, facilitating fast and long-range cascading information propagation among remotely connected nodes. Therefore, with DS-GL, information can be seamlessly transmitted even among nodes that are not directly connected. This feature is distinguished from GNNs, where information is propagated from one node to its neighbors for only once per layer. Second, for nodes in the clusters that are not directly connected through CUs, if their connections are critical for high accuracy, the Wormhole interconnection introduced above will be enabled to establish direct connections among them. It is worth highlighting that the "Wormholes" require no extra hardware, but only share little resources from CUs to enable direct connections among remote PEs with considerable bandwidth.

D. Featuring Co-Annealing Methods

Since the proposed sparse dynamical system is no longer fully connected, the natural annealing process in a Real-Valued DSPU should be adjusted accordingly. In particular, we confront two imperative problems. First, in contrast to the all-to-all connections in a Real-Valued DSPU, what modifications are required in the hardware to make the PEs collaboratively anneal through the sparse connections? Second, how can the hardware manage situations where its capacity is inadequate to facilitate the concurrent annealing of all nodes? In response to these problems, the co-annealing approaches are also categorized in a bipartite manner. (a) **Spatial co-annealing** is the standard annealing process performed on the proposed sparse dynamical system. Given the communication patterns of the super-communities, natural annealing is collectively performed in all super-communities leveraging the proposed hierarchical interconnect architecture. (b) **Temporal & Spatial co-annealing** is designed in the case of insufficient capacity of the dynamical system. In this scenario, one Spatial co-annealing is transformed into iterative partial annealing until convergence is reached.

Spatial co-annealing method: Fig. 9 depicts the coupling between two example PEs on the left, demonstrating the sparse communication pattern between nodes. The blue squares denote the communication facilitated between the nodes depicted as the green squares, aka "activated nodes". Subsequently, annealing is performed following the communication pattern as Spatial co-annealing, featuring its real-time synchronization capability through CUs. In the black-framed box centered in Fig. 9, taking PE1 for example, the Spatial co-annealing mechanism starts from a "PE-CU Map Buffer" which stores all the lists of activated nodes to be deployed to neighbor CUs. For a hardware configuration with specific L, the mapping method is further selected depending on whether D is less than L. If yes, the Spatial co-annealing method shown in the box with green dotted frame is applied. In this situation, the spatial scheduler directly fetches the node-to-CU mapping information from "PE-CU Map Buffer". It first detects the overlapping between the nodes to different CUs, and then generates the mapping signal to the routers. Meanwhile, the "Super Connect" module sends a control signal to enable communication between CUs for overlapped nodes or "Wormhole"

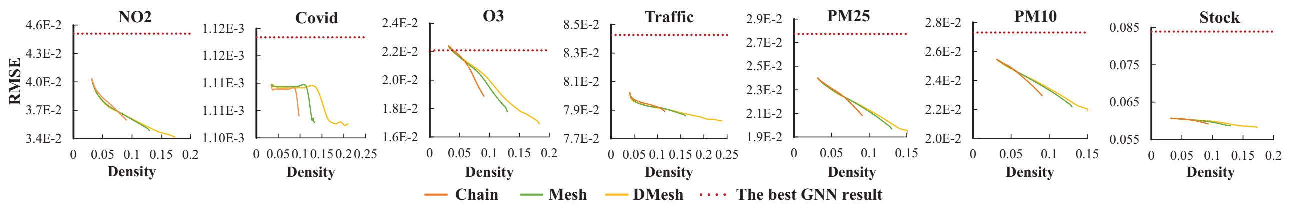

Fig. 10. DS-GL accuracy (RMSE) vs the density of coupling matrix (proportion of nonzero elements; sparsity=1-density) with different communication patterns. "Chain/Mesh/DMesh" refer to different communication patterns with Wormhole enabled.

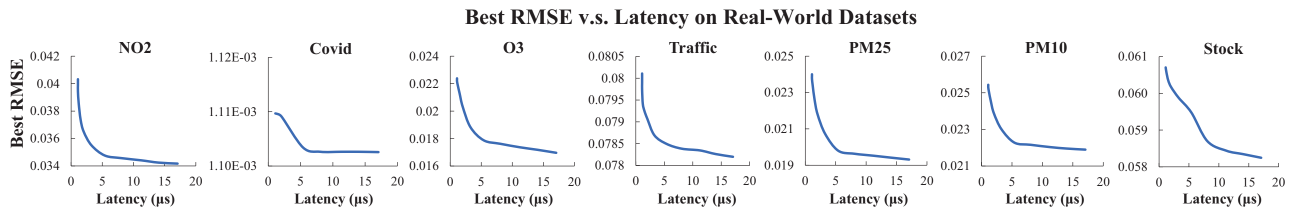

Fig. 11. DS-GL Accuracy vs inference latency (annealing time). Temporal & Spatial co-annealing is adopted for higher accuracy with longer annealing time.

patterns. Since the communication demand density is lower than the hardware communication capability, all nodes can be directly mapped to the corresponding CUs. The TR CU of PE1 is drawn in the figure as an example, where the weights (or the coupling parameters) for the couplings in the CU are stored locally in the "In-CU Weight Buffer" in each CU. For Spatial co-annealing, the weights do not change and are programmed to the coupling crossbar via DACs.

Temporal & Spatial co-annealing method: In the high communication demand density scenario, when D is greater than L, the CUs become saturated with some unaddressed couplings, and the standard Spatial co-annealing no longer applies. Under this circumstance, a Temporal & Spatial co-annealing approach is adopted, with a single Spatial co-annealing decomposed into iterations of partial annealing. In Fig. 9, the box with orange dotted frame shows the hardware for the Temporal co-annealing component, which functions collectively with the Spatial co-annealing part as follows. First, the node lists from the "PE-CU Map Buffer" are sent to the temporal scheduler to divide the lists into smaller "slices", with each size not greater than L. The slices are then stored in the "Temporal Map Buffer", where the buffer sends only one group of slices at a time to the spatial scheduler for further spatial mapping. The "Switch Controller" generates the control signals to inform the buffer to exchange the groups of slices in turn, namely, a Switch-in-turn process. Since the weight parameters in a CU need to be exchanged within different slices, the switch control signals are also connected to the "Weight Select" module in the CU. In this way, high communication demand is supported by the proposed hardware architecture, even with limited capacity of CUs.

V. Evaluation

In this section, the tradeoff between communication density and accuracy, and that between inference latency and accuracy are evaluated. In addition, DS-GL is compared with various SOTA GNNs across different hardware platforms on the following metrics: accuracy, latency, energy, and hardware costs.

A. Experimental Setup

Applications and Datasets. We evaluate the proposed framework on seven real-world datasets from four application scenarios. (1) Traffic flow prediction: *traffic* [20] contains the traffic flow data in Japan. (2) Air quality prediction: *PM25*, *PM10*, *NO2* and *O3*, containing PM2.5, PM10, NO_2, and O_3 data from 2019.5 to 2019.12 in Chinese Air Quality Reanalysis database [22]. (3) Pandemic progression prediction: *Covid* [7] contains 2020-2023 daily case increments of COVID-19 in US. (4) Stock price prediction: predicting the daily prices of stocks. *Stock* [28] contains prices for tickers trading on NASDAQ up to 2020.4.

Algorithm Baselines. For fair evaluation, three SOTA spatial-temporal GNNs are selected as the baselines, including *GWN* [36], *MTGNN* [35], and *DDGCRN* [34]. Their hyper-parameters are set according to their released codes.

Platforms. NVIDIA A100 40GB SXM GPUs are used to measure the training time, inference latency, and accuracy of the SOTA GNNs. For DS-GL, the accuracy and latency are measured using a CUDA-based Finite Element Analysis (FEA) software simulator implemented on top of the one of BRIM [2]. Cadence Mix-signal Design Environment (with 45-nm technology node) is used to evalaute the power and area of DSPU and DS-GL.

B. Tradeoff among Accuracy, Latency, and Graph Sparsity

Fig. 10 shows the accuracy (in Root Mean Square Error, RMSE) of DS-GL with different levels of post-decomposition graph sparsity ($= 1-$Density) and various decomposition patterns across seven real-world graph learning problems. The decomposition patterns include Chain, Mesh, and DMesh, each with Wormhole enabled. The red dotted lines represent the best accuracy of our selected SOTA GNNs. Results show that the accuracy of DS-GL increases with higher graph density (or lower graph sparsity). Moreover, more complex communication patterns enable higher flexibility in graph decomposition, therefore resulting in higher accuracy.

TABLE I
HARDWARE COMPARISON WITH BRIM, A SOTA ISING MACHINE.

	Effective spins	Power	Area	Scalable	Data type
BRIM [2]	2000	250 mW	5 mm^2	No	Binary
DSPU-2000	2000	260 mW	5.1 mm^2	No	Real-Value
DS-GL	8000	550 mW	6.5 mm^2	Yes	Real-Value

TABLE II
RMSE COMPARISON BETWEEN DS-GL AND SOTA GNNS.

Dataset	NO2	Covid	O3	Traffic	PM25	PM10	Stock
GWN [36]	5.52e-2	1.75e-3	2.40e-2	1.27e-1	3.20e-2	2.74e-2	8.44e-2
MTGNN [35]	4.51e-2	1.85e-3	2.23e-2	1.14e-1	2.83e-2	2.73e-2	8.39e-2
DDGCRN [34]	5.17e-2	1.16e-3	2.21e-2	8.43e-2	2.77e-2	2.78e-2	8.42e-2
DS-GL-Spatial	3.94e-2	1.11e-3	2.21e-2	7.97e-2	2.37e-2	2.53e-2	6.06e-2
DS-GL-Chain	3.60e-2	1.11e-3	1.89e-2	7.89e-2	2.08e-2	2.30e-2	5.92e-2
DS-GL-Mesh	3.48e-2	1.11e-3	1.78e-2	7.86e-2	1.97e-2	2.22e-2	5.86e-2
DS-GL-DMesh	3.41e-2	1.11e-3	1.70e-2	7.83e-2	1.93e-2	2.19e-2	5.82e-2

Fig. 11 shows the best accuracy obtainable with different inference latency. Recall that while a high density of the coupling matrix enhances accuracy, it often exceeds hardware capacity. Therefore, Temporal & Spatial co-annealing is adopted to support higher density at the cost of increased annealing time (i.e., inference latency), leading to a tradeoff between accuracy and latency. For most datasets, the RMSE decreases sharply with increasing latency until an inflection point ($\sim$5 μs), after which the decline is more gradual.

C. Evaluation of DS-GL Hardware Costs

The hardware costs of BRIM, DSPU, and DS-GL are listed in Table I, where DSPU-2000 refers to a DSPU consisting of 2000 spins for a fair comparison with BRIM [2]. It shows that the Real-Valued DSPU can support real-world problems with minor extra costs compared to the binary machine. Furthermore, DS-GL scales the number of spins by 4$\times$ at the cost of 2$\times$ higher power.

D. Evaluation of Inter-tile Synchronization

Although DS-GL does not need synchronization among tiles within the same mapping, synchronization is necessary among multiple mappings. Specifically, the synchronization frequency (1/500ns) required for high accuracy is much lower than that supported by the DS-GL hardware (1/200ns). To demonstrate the efficacy of synchronization, Fig. 12 uses Stock, NO2, and Traffic datasets to evaluate the variation in accuracy (RMSE) over the synchronization interval from 1 ns to 5 µs. Fig. 12 shows that the accuracy generally decreases with the increase of synchronization interval. However, the accuracy drop is negligible when synchronization interval is less than 500 ns, which is easily achievable on the DS-GL hardware.

E. Accuracy Comparison with SOTA GNN

Table II compares the accuracy among four DS-GL design choices and three SOTA GNNs including GWN [36], MTGNN [35], and DDGCRN [34]. The accuracy is evaluated in terms of RMSE. The four design choices include “DS-GL-Spatial”, “DS-GL-Chain”, “DS-GL-Mesh”, “DS-GL-DMesh”. Particularly, DS-GL-Spatial refers to the design with only spatial co-annealing (temporal co-annealing disabled) which trades accuracy for low inference latency. In contrast, “DS-GL-Chain”, “DS-GL-Mesh”, and “DS-GL-DMesh” represent the designs with different decomposition patterns, each with both spatial and temporal co-annealing enabled, therefore, delivering slower inference but higher accuracy. As the table shows, DS-GL outperforms SOTA GNNs on all datasets. Taking the air-quality-NO_2 dataset as an example, DS-GL-Spatial achieves 12.6%-28.6% reduced RMSE compared to SOTA GNNs while DS-GL-DMesh achieves 22.4%-38.2% RMSE reduction.

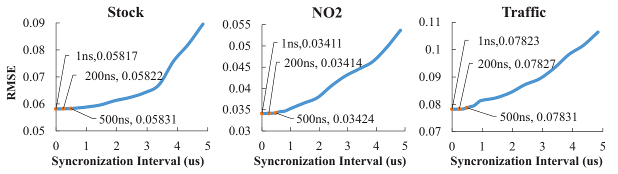

Fig. 12. RMSE vs Synchronization Interval, with 200 ns used in DS-GL.

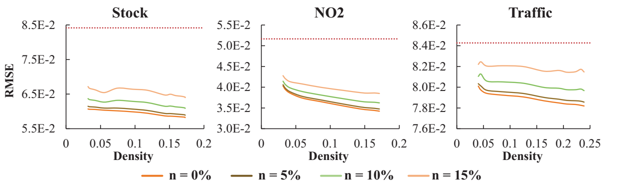

Fig. 13. RMSE vs matrix density under noise percentage n.

F. Latency & Energy Comparison with Accelerators & GPU

In Table. II, DS-GL exhibits substantial accuracy improvements over SOTA GNNs. To further evaluate the latency and energy efficiency, we present a comparison with several SOTA hardware accelerators, including AWB-GCN [12], I-GCN [13], NTGAT [17], GraphAGILE [42], and RACE [41]. Since GNN models are often specifically designed for these applications, while accelerators are not designed for these models - for a fair comparison, we assume these accelerators are of full utilization, achieving peak TFLOPs with typical power. Even with this assumption, DS-GL still consistently outperforms all SOTA accelerators and modern GPU on both latency and energy consumption, as summarized in Table. III.

G. Evaluation of System robustness

To estimate the impact of noise on the system, we inject dynamic noises at both nodes and coupling units. The noise is generated by the Gaussian distribution with standard deviation values of 5%, 10%, and 15% each. The results of three representative datasets with ‘DMesh’ pattern are shown in Fig. 13, where n in the legend represents the standard deviation of noise. We observe that the impact of dynamic noise is not significant, showing the natural good tolerance of physical dynamical systems to noise. As a result, in practical situations, DS-GL still achieves better accuracy over the GNNs.

H. Multi-Dimensional Applications

To further demonstrate the wide applicability of DS-GL, two datasets (house prices in California [26] and global climate [10], denoted separately as *CA housing* and *climate*) including multiple features for a node are evaluated, with results shown in Table IV. For example, *climate* contains

TABLE III
COMPARISON OF INFERENCE LATENCY AND ENERGY COST PER INFERENCE AMONG DS-GL, SOTA GNNS ON GNN ACCELERATORS, AND GPUS.

Hardware Platforms		**Stratix 10 SX**				**Xilinx Alveo U200**				**Xilinx Alveo U250**				**Xilinx Alveo U280**				**NVIDIA A100 SXM**			
Related Works†		AWB [12], IGCN [13]				NTGAT [17]				GraphAGILE [42]				RACE [41]							
Peak TFLOPS		2.7				1.4				2.8				2.1				156			
Max Power (W)		215				225				225				225				400			
Typical Power (W)		137				100				110				100				250			
Application		**covid**	**air**	**traffic**	**stock**	**covid**	**air**	**traffic**	**stock**	**covid**	**air**	**traffic**	**stock**	**covid**	**air**	**traffic**	**stock**	**covid**	**air**	**traffic**	**stock**
GNNs Latency (μs)	**GWN**	1141	1335	985	1751	2203	2578	1902	3382	1101	1289	951	1691	1469	1719	1268	2255	2757	4601	4176	5333
	MTGNN	516	604	446	792	996	1166	860	1530	498	583	430	765	664	777	574	1021	9319	1.6e4	1.2e4	2.3e4
	DDGCRN	690	847	443	1063	1333	1636	855	2051	667	818	427	1018	889	1090	570	1364	3.7e4	6.0e4	2.6e4	1.2e5
DS-GL Latency (μs)		0.15	1.1	0.65	1.0	0.15	1.1	0.65	1.0	0.15	1.1	0.65	1.0	0.15	1.1	0.65	1.0	0.15	1.1	0.65	1.0
GNNs Energy (mJ)	**GWN**	156	183	135	240	220	258	190	338	121	142	105	186	147	172	127	225	674	1138	984	1298
	MTGNN	70.7	82.7	61.0	109	100	118	85.1	152	54.9	64.1	46.9	84.2	66.5	78.0	57.5	101	2241	4164	2973	5419
	DDGCRN	94.6	116	60.6	145	134	165	85.4	205	73.2	89.9	47.1	113	89.1	110	56.8	136	9457	1.5e4	6392	3.0e4
DS-GL Energy (mJ)		9e-5	6e-4	4e-4	6e-4	9e-5	6e-4	4e-4	6e-4	9e-5	6e-4	4e-4	6e-4	9e-5	6e-4	4e-4	6e-4	9e-5	6e-4	4e-4	6e-4

† The latency of GNN accelerators is reported based on their theoretical peak performance with full utilization.

TABLE IV
RMSE & LATENCY COMPARISON ON MULTI-DIMENSIONAL DATASETS.

Multi-Dimensional Dataset	**CA housing**		**climate**	
Comparison Metric	RMSE	latency (μs)	RMSE	latency (μs)
GWN [36]	1.89e-2	6.40e+3	4.32e-1	1.37e+4
MTGNN [35]	2.10e-2	2.08e+4	4.33e-1	1.87e+4
DDGCRN [34]	1.86e-2	5.03e+4	4.03e-1	3.54e+4
DS-GL	1.62e-2	1.08	3.89e-1	0.97

12 features per node, including humidity, temperature, wind speed, etc. Latency is evaluated on an A100-40G SXM GPU.

VI. RELATED WORKS

Variances of Ising Machines: In addition to BRIM, there are many other Ising machine concepts and prototypes including D-Wave's quantum annealers that have been put into commercial use [1]. As a quantum Ising machine, a D-Wave annealer [14] takes advantage of the quantum effects introduced by its superconducting qubits to achieve extraordinary speed. However, quantum Ising machines require a cryogenic system for extremely low temperatures as the operating environment. This cryogenic system is also the main reason for its high energy consumption ($\sim$25KW), which significantly limits its practical use at the current stage. In Coherent Ising machines (CIM) [19], [25], [38], optical parametric oscillators are used to represent spins, while the coupling is currently emulated through digital computation. Consequently, the efficiency of current CIMs is rather limited. In contrast, Ising machines based on electric oscillators are closer to real-life deployment, but may require hard-to-integrate inductors for high-quality oscillations. However, a 48-oscillator Ising machine [24] using ring oscillators has recently emerged, demonstrating the potential in this approach.

Among the choices of Ising machines, BRIM is uniquely fitted to our purpose in this work in contrast to two other groups of Ising machines designs: ① Oscillator-based Ising machines use oscillator phase (ϕ_i) as the spin [3], [9], [16], [19], [27], [33]. These spins are not Ising spins (1 degree of freedom: ± 1) but XY model spins (2 degrees of freedom) with the following Lyapunov function: $H = -\sum_{i,j} J_{ij} \cdot cos(\phi_i - \phi_j)$. Hence they do not lend to real-value quadratic objective function as naturally as BRIM does. ② Digital annealers/accelerators are hardwired annealing algorithms [37], [39]. They are certainly more efficient than general-purpose processors, but do not yet rival SOTA dynamical systems in efficiency. Besides, almost all such designs in the literature acquire extra efficiency by using local coupling and/or single-bit coupling, making them impractical for real-world problems.

Existing works on Ising Machines for ML: The potential of Ising machines in solving ML problems has only been recently recognized. Recent works have attempted to use BRIM to solve or partially solve simple learning problems including predicting traffic congestion [29], collaborative filtering [23], and supporting energy-based models [32]. However, those works only support binary problems (e.g., "congested (0)" or "non-congested (1)") in congestion prediction and "like" or "dislike" in binary collaborative filtering). Moreover, the congestion prediction work [29] uses BRIM to impute invisible congestion data within the same timestamp, while the temporal prediction is performed on digital processors. In [23], BRIM is used to determine whether a user will "like" or "dislike" an item determined by the similarity between items. Like congestion prediction, no temporal evaluation is performed by the Ising machine in [23]. Furthermore, both works offer solutions tailored to specific applications, whereas DS-GL accommodates a broader range of real-valued applications that necessitate intricate analysis of temporal information.

VII. CONCLUSIONS

This paper proposes a nature-powered graph learning framework dubbed DS-GL. Rooted in a CMOS-compatible Ising machine, DS-GL inherits the extraordinary computational efficiency of the Ising machine and extends its potential to real-valued and larger-scale GL problems. Evaluations with four diverse GL applications across seven datasets show that DS-GL can deliver speedups ranging from $10^3\times$ to $10^5\times$ over Graph Neural Networks on GPUs while operating at a power 2 orders of magnitude lower than GPUs, with $5\% - 30\%$ accuracy enhancement.

ACKNOWLEDGEMENTS

This work was supported in part by NSF under Awards CCF-2326494, No. 2231036, and No. 2233378; by NYS Center of Excellence under Awards No.2089C001; by DARPA under contract No. FA8650-23-C-7312; and by META Reality Lab. This research was also partially supported by the U.S. DOE Office of Science, Office of Advanced Scientific Computing Research, under award No.66150: "CENATE - Center for Advanced Architecture Evaluation" and No.78284: "ComPort: Rigorous Testing Methods to Safeguard Software Porting".

References

[1] "The D-Wave 2000Q quantum computer." [Online]. Available: https://www.dwavesys.com/sites/default/files/D-Wave%202000Q%20Tech%20Collateral_0117F.pdf

[2] R. Afoakwa, Y. Zhang, U. K. R. Vengalam, Z. Ignjatovic, and M. Huang, "BRIM: Bistable resistively-coupled Ising machine," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 749–760.

[3] I. Ahmed, P.-W. Chiu, and C. H. Kim, "A probabilistic self-annealing compute fabric based on 560 hexagonally coupled ring oscillators for solving combinatorial optimization problems," in *2020 IEEE Symposium on VLSI Circuits*, 2020, pp. 1–2.

[4] J. Baek, M. Kang, and S. J. Hwang, "Accurate learning of graph representations with graph multiset pooling," in *International Conference on Learning Representations*, 2020.

[5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, oct 2008.

[6] S. G. Brush, "History of the Lenz-Ising model," *Reviews of modern physics*, vol. 39, no. 4, p. 883, 1967.

[7] Centers for Disease Control and Prevention, "COVID data tracker," Atlanta, GA: U.S. Department of Health and Human Services, CDC, November 22 2023, https://covid.cdc.gov/covid-data-tracker.

[8] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 257–266.

[9] J. Chou, S. Bramhavar, S. Ghosh, and W. Herzog, "Analog coupled oscillator based weighted Ising machine," *Scientific Reports*, vol. 9, no. 1, p. 14786, 2019. [Online]. Available: https://doi.org/10.1038/s41598-019-49699-5

[10] N. Elgiriyewithana, "World weather repository (daily updating)[dataset]. kaggle. com," 2023.

[11] F. Frasca, B. Bevilacqua, M. Bronstein, and H. Maron, "Understanding and extending subgraph GNNs by rethinking their symmetries," *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 376–31 390, 2022.

[12] T. Geng, A. Li, R. Shi, C. Wu, T. Wang, Y. Li, P. Haghi, A. Tumeo, S. Che, S. Reinhardt, and M. C. Herbordt, "AWB-GCN: A graph convolutional network accelerator with runtime workload rebalancing," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020, pp. 922–936.

[13] T. Geng, C. Wu, Y. Zhang, C. Tan, C. Xie, H. You, M. Herbordt, Y. Lin, and A. Li, "I-GCN: A graph convolutional network accelerator with runtime locality enhancement through islandization," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1051–1063.

[14] R. Harris, M. W. Johnson, T. Lanting, A. J. Berkley, J. Johansson, P. Bunyk, E. Tolkacheva, E. Ladizinsky, N. Ladizinsky, T. Oh, F. Cioata, I. Perminov, P. Spear, C. Enderud, C. Rich, S. Uchaikin, M. C. Thom, E. M. Chapple, J. Wang, B. Wilson, M. H. S. Amin, N. Dickson, K. Karimi, B. Macready, C. J. S. Truncik, and G. Rose, "Experimental investigation of an eight-qubit unit cell in a superconducting optimization processor," *Phys. Rev. B*, vol. 82, p. 024511, Jul 2010.

[15] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," in *Neural Networks: Tricks of the Trade: Second Edition*. Springer, 2012, pp. 599–619.

[16] T. Honjo, T. Sonobe, K. Inaba, T. Inagaki, T. Ikuta, Y. Yamada, T. Kazama, K. Enbutsu, T. Umeki, R. Kasahara, K. ichi Kawarabayashi, and H. Takesue, "100,000-spin coherent Ising machine," *Science Advances*, vol. 7, no. 40, p. eabh0952, 2021. [Online]. Available: https://www.science.org/doi/abs/10.1126/sciadv.abh0952

[17] W. Hou, K. Zhong, S. Zeng, G. Dai, H. Yang, and Y. Wang, "NTGAT: A graph attention network accelerator with runtime node tailoring," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 645–650.

[18] W.-H. Hu, S. E. Lee, and N. Bagherzadeh, "DMesh: a diagonally-linked mesh network-on-chip architecture," *Network on Chip Architectures*, vol. 14, 2008.

[19] T. Inagaki, Y. Haribara, K. Igarashi, T. Sonobe, S. Tamate, T. Honjo, A. Marandi, P. L. McMahon, T. Umeki, K. Enbutsu, O. Tadanaga, H. Takenouchi, K. Aihara, K.-i. Kawarabayashi, K. Inoue, S. Utsunomiya, and H. Takesue, "A coherent Ising machine for 2000-node optimization problems," *Science*, vol. 354, no. 6312, pp. 603–606, 2016.

[20] R. Jiang, Z. Wang, J. Yong, P. Jeph, Q. Chen, Y. Kobayashi, X. Song, S. Fukushima, and T. Suzumura, "Spatio-temporal meta-graph learning for traffic forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 7, 2023, pp. 8078–8086.

[21] J. Karras, A. Holynski, T.-C. Wang, and I. Kemelmacher-Shlizerman, "DreamPose: Fashion video synthesis with stable diffusion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 22 680–22 690.

[22] L. Kong, X. Tang, J. Zhu, Z. Wang, J. Li, H. Wu, Q. Wu, H. Chen, L. Zhu, W. Wang, B. Liu, Q. Wang, D. Chen, Y. Pan, T. Song, F. Li, H. Zheng, G. Jia, M. Lu, L. Wu, and G. R. Carmichael, "A 6-year-long (2013–2018) high-resolution air quality reanalysis dataset in China based on the assimilation of surface observations from CNEMC," *Earth System Science Data*, vol. 13, no. 2, pp. 529–570, 2021.

[23] Z. Liu, Y. Yang, Z. Pan, A. Sharma, A. Hasan, C. Ding, A. Li, M. Huang, and T. Geng, "Ising-CF: A pathbreaking collaborative filtering method through efficient Ising machine learning," in *Proceedings of the 60th ACM/IEEE Design Automation Conference. of DAC*, 2023.

[24] H. Lo, W. Moy, H. Yu, S. Sapatnekar, and C. H. Kim, "An Ising solver chip based on coupled ring oscillators with a 48-node all-to-all connected array architecture," *Nature Electronics*, vol. 6, no. 10, pp. 771–778, 2023.

[25] P. L. McMahon, A. Marandi, Y. Haribara, R. Hamerly, C. Langrock, S. Tamate, T. Inagaki, H. Takesue, S. Utsunomiya, K. Aihara, R. L. Byer, M. M. Fejer, H. Mabuchi, and Y. Yamamoto, "A fully programmable 100-spin coherent Ising machine with all-to-all connections," *Science*, vol. 354, no. 6312, pp. 614–617, 2016.

[26] P. Mooney, "Zillow house price data[dataset]. kaggle. com," 2021.

[27] W. Moy, I. Elshazly, P.-w. Chiu, J. Moy, S. Sapatnekar, and C. Kim, "A 1,968-node coupled ring oscillator circuit for combinatorial optimization problem solving," *Nature Electronics*, vol. 5, 05 2022.

[28] O. Onyshchak, "Stock market dataset," 2020. [Online]. Available: https://www.kaggle.com/dsv/1054465

[29] Z. Pan, A. Sharma, J. Y.-C. Hu, Z. Liu, A. Li, H. Liu, M. Huang, and T. Geng, "Ising-Traffic: Using Ising machine learning to predict traffic congestion under uncertainty," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 8, pp. 9354–9363, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26121

[30] R. Sarkar, S. Abi-Karam, Y. He, L. Sathidevi, and C. Hao, "FlowGNN: A dataflow architecture for real-time workload-agnostic graph neural network inference," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, mar 2023, pp. 1099–1112.

[31] T. Takemoto, M. Hayashi, C. Yoshimura, and M. Yamaoka, "2.6 a 2$\times$ 30k-spin multichip scalable annealing processor based on a processing-in-memory approach for solving large-scale combinatorial optimization problems," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2019, pp. 52–54.

[32] U. Vengalam, Y. Liu, T. Geng, H. Wu, and M. Huang, "Supporting energy-based learning with an Ising machine substrate: A case study on RBM," in *Proceedings of the International Symposium on Microarchitecture*, 2023.

[33] T. Wang and J. Roychowdhury, "OIM: Oscillator-based Ising machines for solving combinatorial optimisation problems," in *Unconventional Computation and Natural Computation*, I. McQuillan and S. Seki, Eds. Cham: Springer International Publishing, 2019, pp. 232–256.

[34] W. Weng, J. Fan, H. Wu, Y. Hu, H. Tian, F. Zhu, and J. Wu, "A decomposition dynamic graph convolutional recurrent network for traffic forecasting," *Pattern Recognition*, vol. 142, p. 109670, 2023.

[35] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 753–763.

[36] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph WaveNet for deep spatial-temporal graph modeling," 2019.

[37] S. Xie, S. R. S. Raman, C. Ni, M. Wang, M. Yang, and J. P. Kulkarni, "Ising-CIM: A reconfigurable and scalable compute within memory analog Ising accelerator for solving combinatorial optimization problems," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 11, pp. 3453–3465, 2022.

[38] Y. Yamamoto, K. Aihara, T. Leleu, K.-i. Kawarabayashi, S. Kako, M. Fejer, K. Inoue, and H. Takesue, “Coherent Ising machines—optical neural networks operating at the quantum limit,” *npj Quantum Information*, vol. 3, no. 1, p. 49, 2017.

[39] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, “A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing,” *IEEE Journal of Solid-State Circuits*, vol. 51, no. 1, pp. 303–309, 2015.

[40] ——, “24.3 20k-spin Ising chip for combinational optimization problem with CMOS annealing,” in *2015 IEEE International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers*. IEEE, 2015, pp. 1–3.

[41] H. Yu, Y. Zhang, J. Zhao, Y. Liao, Z. Huang, D. He, L. Gu, H. Jin, X. Liao, H. Liu, B. He, and J. Yue, “RACE: An efficient redundancy-aware accelerator for dynamic graph neural network,” *ACM Trans. Archit. Code Optim.*, aug 2023, just Accepted.

[42] B. Zhang, H. Zeng, and V. K. Prasanna, “GraphAGILE: An fpga-based overlay accelerator for low-latency GNN inference,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 9, pp. 2580–2597, 2023.

[43] Y. Zhang, H. You, Y. Fu, T. Geng, A. Li, and Y. Lin, “G-CoS: Gnn-accelerator co-search towards both better accuracy and efficiency,” *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2021.