nature communications

Perspective

https://doi.org/10.1038/s41467-025-63494-z

Next-generation graph computing with electric current-based and quantum-inspired approaches

Received: 17 September 2024

Accepted: 21 August 2025

Published online: 28 August 2025

Check for updates

Yoon Ho Jang[1,2], Janguk Han[1,2], Soo Hyung Lee[1,2] & Cheol Seong Hwang[1]

Graph data is crucial for modeling complex relationships in various fields, but conventional graph computing methods struggle to handle increasingly intricate and large-scale graph data. Electric current-based graph computing and Quantum-inspired graph computing offer innovative hardware-based solutions to these challenges. Electric current-based graph computing has progressed from Euclidean graph data to non-Euclidean ones using the memristive crossbar arrays. This Perspective introduces various crossbar array-based electric current-based graph computings, which offer flexibility in representing complex graphs, enabling a wide range of graphical applications in materials, biology, and social science. It also discusses quantum-inspired graph computing, employing probabilistic bits, oscillatory neural networks, and related architectures to solve complex optimization problems. Electric current-based and quantum-inspired graph computing remain in their early stages of evolution, requiring further work to advance materials, devices, and architectures to fully realize their potential. These advancements will open opportunities for more diverse and complex real-world applications.

Graph data represent complex relationships across various domains of modern big data, such as social networks and biological pathways. A graph consists of nodes (representing entities) and edges (representing the connections between them), whereas the adjacency matrix typically represents graph data structure (Fig. 1a). In contrast, scalar or tabular data do not have inter-entity relationships and conventional hardware architectures have been optimized for linear operations to process this type of data efficiently. As the importance of graph data continues to grow, the ability to efficiently process and analyze graph structures is becoming increasingly crucial, advancing the field of graph computing[1,2]. Graph computing has focused on deterministic graphs, where the relationships between nodes are fixed and well-defined[3,4]. These graphs have been instrumental in extracting static and meaningful information from datasets, typically using adjacency matrices. However, as data becomes more complex and voluminous, conventional linear models often fall short of analysis capacity when dealing with vast and intricate graphs[5,6].

New paradigms in graph computing are emerging to address these challenges, one of which is electric current-based graph computing (EGC). In EGC, electrical currents flowing through the hardware represent the optimal paths in the graph, enabling efficient computation of graphical similarity and facilitating the solution of more complex graph problems. Early efforts in this area involved using grids of electrodes and lateral memristors or nanowires to create hardware-based representations of graphs[7–11]. While these methods demonstrated the potential of hardware-based graph computing, their application was limited to Euclidean graphs; thus, they are inappropriate for handling more complex, non-Euclidean structures. Subsequent advancements used volatile memristors to implement delay-based small-world graphs (left panel of Fig. 1b)[12,13]. However, the range of graphs these methods could represent was limited. Recent developments using memristive crossbar array (CBA) structures have significantly expanded the graph computing capability to represent non-Euclidean graphs[14–18].

[1]Department of Materials Science and Engineering and Inter-university Semiconductor Research Center, College of Engineering, Seoul National University, Seoul, Republic of Korea. [2]These authors contributed equally: Yoon Ho Jang, Janguk Han, Soo Hyung Lee. e-mail: cheolsh@snu.ac.kr

https://doi.org/10.1038/s41467-025-63494-z

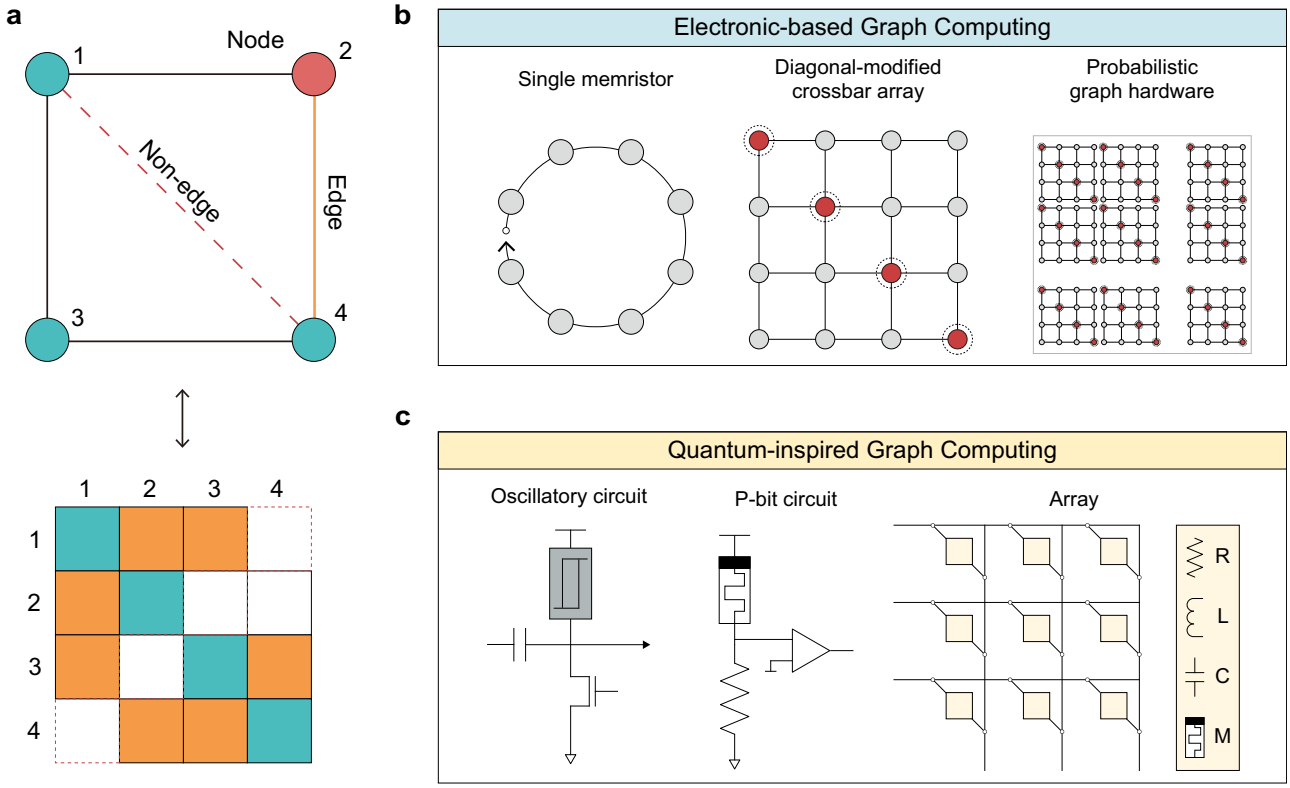

Fig. 1 | Overview of graph structures and computing approaches. **a** Representation of basic graph structures. The upper panel illustrates a simple graph with four nodes and edges, while the lower panel represents its adjacency matrix. **b** Electronic-based graph computing. Electrical properties are extracted from the mapped graph on various physical devices. **c** Quantum-inspired graph computing. The time evolution of the graph structure is extracted to solve various problems. Various circuit elements and coupling matrices represent the nodes and edges.

To date, CBA structures have primarily been utilized as accelerators for vector-matrix multiplication in neural network computations[19]. However, CBA structures suffer from sneak currents, prompting research into various unit devices such as transistors, high-nonlinearity selectors, and self-rectifying memristors[20]. Moreover, CBAs intended for in-situ training require high endurance and symmetric potentiation/depression characteristics. While unit devices for graph-representing CBAs share similar retention and endurance requirements to those used in inference CBAs, CBA-based EGC additionally demands directional properties. Consequently, self-rectifying memristors are typically employed for scalability and three-dimensional integration advantages.

The advancements in device and structure enabled the hardware-based representation of more graph types, including directed and weighted graphs[14–17], which were unfeasible in the conventional EGC. This advancement is primarily due to the flexibility of CBA structures, which enable more sophisticated graph mappings (middle panel of Fig. 1b).

Furthermore, recent studies utilized multi-CBA setups to model probabilistic graphs and calculate connection probabilities for complex paths[21], expanding its applicability to systems with inherent uncertainties[22,23] (right panel of Fig. 1b). EGC excels at capturing graph connectivity and can represent various graph types. However, it faces limitations in analyzing the time-dependent evolution of node states.

This Perspective explores three other graph-based computing approaches**:** those designed to handle time-dependent node evolution, namely probabilistic bits (p-bits)[24–40], oscillatory neural networks (ONNs)[41–54], and Hopfield neural networks (HNNs)[55–60]. While these approaches differ from quantum computing in their computing methods, they share common graph structures based on nodes represented as two-level systems. These methods are designed to address optimization and machine learning problems that are otherwise intractable using classical techniques. Inspired by quantum mechanics or developed in conjunction with quantum computing-based Ising machines, these architectures are collectively referred to here as quantum-inspired graph computing (QGC) (Fig. 1c).

In this Perspective, the theoretical foundations of EGC and QGC, as well as the physical graph representation (PGR) of each method, were first introduced. Subsequently, various applications and performance evaluations were examined. Finally, potential advancements and future directions from an integration perspective were discussed to enable their practical use.

Results

Graph computing theory

When modeling a space or circumstance in a graph, nodes represent each entity, and edges represent the relationships between the nodes. A graph $G=(V,E)$, which consists of N nodes $V=\{v_i,|,0<i<N\}$ and a set of edges $E=\{e_{vu},|,v,u\in V\}$, can be represented as an adjacency matrix $A\in R^{|N|\times|N|}$. When $e_{uv}\neq e_{vu}$, G is considered a directed graph, and when a characteristic value, such as weight, w_{vu} exists for e_{vu}, G is regarded as a weighted graph.

A graph is classified as either Euclidean or non-Euclidean based on its spatial characteristics, with distinct methodologies applied to interpret them. Each node in the Euclidean graph belongs to a metric space, situated in physical space or on a plane where Euclidean geometric concepts, such as distance and angles, are applied. For example, the graph is located on a two-dimensional coordinate plane, and the relationship between two points $((x_1,y_1)$ and $(x_2,y_2))$ can be obtained through metric calculation, aiming to obtain the Euclidean or

Manhattan distance. The distance d between two nodes satisfies the following relationships for any x, y, and z nodes.

$$\begin{cases} (x,y)=0 \iff x=y \\ d(x,y)\geq 0 \\ d(x,y)=d(y,x) \\ d(x,y)+d(y,z)\geq d(x,z) \end{cases} \quad (1)$$

The quantified distance implies the strength of the relationship between nodes and can be used for analyzing geographic distances.

A non-Euclidean graph is defined as one that cannot be explained using standard Euclidean geometry. Specific position vectors cannot represent any two nodes in a non-Euclidean graph. Therefore, the relationship between two nodes is abstractly expressed (for example, in a social network, the "distance" between two people refers to the degree of connection rather than actual physical distance).

Graph embedding is required to map the graph to a lower-dimensional vector space (Euclidean space) to interpret complex, high-dimensional non-Euclidean graphs more easily. In this case, the embedded graph must reflect the relationships of the original high-dimensional space as much as possible. The embedding vectors generated through the mapping function $ENC(v)$ must represent the local and global information that the nodes in the original graph had, which must be preserved within the embedding space[61]. The node v can be embedded into a lower-dimensional Euclidean vector space through the mapping function $ENC(v)$. Ideally, when the connection strength with a node in the original graph is high, it should be embedded close to each other. When the connection strength is low, it should be embedded farther apart.

Recently, embedding vectors have been effectively extracted using random walk-based graph embedding methods, such as DeepWalk and Node2Vec[62,63]. These methods do not require separate labeled data but have several inherent limitations[64,65]. First, they cannot reflect real-time characteristics of nodes that change over time or in response to environmental inputs. Second, the unavoidable loss of information during the graph embedding process is also a problem.

Quantum-inspired graph computing

Quantum computing employs qubits that exhibit superposition and entanglement, enabling exploration of exponentially large solution spaces. While these properties offer theoretical speed-ups over classical methods[66,67], practical quantum computing remains limited by decoherence and error-correction overhead, restricting its usability for near-term, large-scale graph problems.

In this regard, QGC has emerged as a class of approaches that share conceptual elements from quantum mechanics, such as energy-based optimization and state transitions, while implementing them through classical or stochastic hardware. In QGC, problems are mapped onto graph structures, where nodes represent two-level systems and coupling edges encode their interactions. This enables hardware to directly utilize the graph's connectivity for efficient computation, eliminating the need for coherent quantum states.

Among the approaches in QGC, probabilistic computing (p-computing) is closely aligned with quantum computing and follows similar research directions. As an intermediate between classical deterministic bits and qubits, p-bits fluctuate between 0 and 1. Various nanodevices are used for p-bit implementation, and the p-computing system is explained through the following relationships[24].

$$s_i = \mathrm{sign}\left(\tanh\left(\beta I_i\right) - \mathrm{rand}\right) \quad (2)$$

$$I_i = \sum_j \boldsymbol{W}_{ij} s_j + h_i \quad (3)$$

In these relationships, the state of a two-level system s_i is determined by the nonlinear activation of input I_i, and inverse temperature β, with an additional random number from [−1,1]. The input I_i to each node s_i (p-bit) is calculated considering the coupling matrix $\boldsymbol{W}$ and external bias field h. Here, the coupling matrix is analogous to the adjacency matrix in EGC, which explains edge interactions between the nodes. The p-computing framework is widely adopted in optimization and probabilistic inference problems[24,40]. Specifically, the Ising model description of the system's energy is adopted to solve classically intractable optimization problems as follows:

$$E = -\left(\sum_{i<j} \boldsymbol{W}_{ij} s_i s_j + \sum_i h_i s_i\right) \quad (4)$$

Expanding upon the Ising model, other variants of QGC, such as ONN and HNN, also use graph structures to address optimization problems. Oscillatory nodes in ONN represent a multi-level system through phase difference, physically coupled according to the coupling matrix. On the other hand, HNN directly calculates the derivative of the energy (Eq. 4) to solve optimization problems. Recent studies on HNN directly map the graph to extract connectivity information[56–58]. The physical implementation of nodes and edges in quantum computing and QGC differs, as discussed in the following section, but they all physically organize graphs for efficient computation.

Advancements in structures for physical graph representation

In EGC, memristive characteristics are used to represent and interpret high-dimensional graphs without preprocessing (Fig. 2). Unlike conventional methods that require converting the graph into a vector format for interpretation (graph embedding), the goal is to represent the high-dimensional graph physically. The proposed structures commonly employ electrical signal responses to applied voltages to infer critical connectivity in graph interpretation.

For instance, nanowire graph networks are constructed in previous works[8,10,11], where memristive junctions are randomly generated in self-assembled metal nanowires (Fig. 2a). The metal nanowires are electrically insulated via resistive-switching coatings (metal oxide, polyvinylpyrrolidone), and crossed wires form metal-insulator-metal junctions. When voltage is applied through additional metal pads on the nanowire graph, conductive pathways are formed within the nanowire graph, generating dynamic behavior within the random graph. This structure was used to find the shortest path in nanowire graphs, enabling applications such as associative memory. Moreover, the neuromorphic dynamics of the nanowire graph were applied in reservoir computing for data processing[8,68].

Another example includes Mott memristor neuron devices (nodes of the graph), which are placed in a CBA to induce thermal conduction between them (Fig. 2b). They integrate spatiotemporal thermal information for network communication[18]. When the oscillation of Mott memristor neurons increases the local temperature, adjacent neurons oscillate even below their critical voltage. This thermal diffusion between neurons enables fundamental graph representation. The arrangement of neurons within the crossbar was also used to solve optimization problems such as Max-cut, providing a new perspective on graph representation through thermal conduction.

However, as the interactions between nodes are constrained by their physical positions, the structures shown in Fig. 2a, b are limited to representing grid graphs. In contrast, Fig. 2c–f shows the EGC structures capable of representing non-Euclidean graphs. Figure 2c illustrates a structure that utilizes the dynamic characteristics and inherent probabilistic behavior of a single memristor to form a complex virtual network. Appeltant et al. proposed a method to interpret a single dynamic node as a virtual cyclic graph through time multiplexing[12]. Guo et al. advanced this approach by utilizing the spontaneous resistance variability of memristors to implement a virtual small-world network that

https://doi.org/10.1038/s41467-025-63494-z

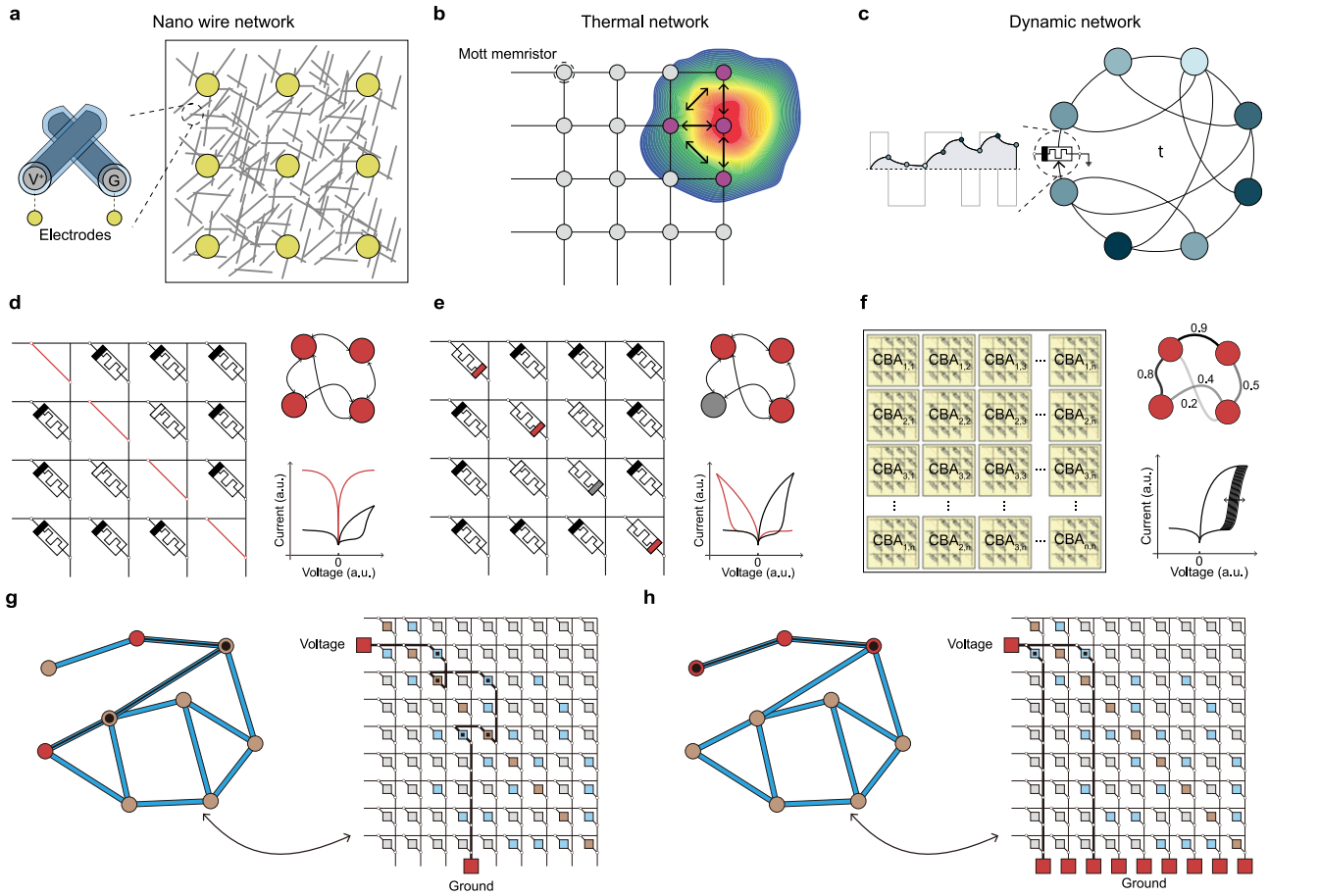

Fig. 2 | Hardware structures for EGC. a A self-assembled metal nanowire network. Metal wires coated with ion-conductive materials are crossed, exhibiting memristive properties. Random graph dynamics arise from conductive pathways determined by the random arrangement of metal wires. **b** A spatiotemporal thermal conduction network based on Mott memristors. Connections between nodes are formed through thermal diffusion. **c** The dynamic and stochastic characteristics of memristors form a virtual circulatory network driven by applied voltage. **d** A CBA with a shortened diagonal. Using a self-rectifying memristor as an individual element in the array enables the physical mapping of the graph's adjacency matrix. The right panel illustrates the represented graph and the electrical characteristics of the devices. Current flows bidirectionally through the shorted device (orange line), while it flows unidirectionally through the self-rectifying device (black line). **e** A CBA with a cross-wired diagonal. Similar to m-CBA, self-rectifying memristors are used as individual elements in the array. The right panel illustrates the represented graph and the electrical characteristics of the diagonal devices (orange line) and off-diagonal devices (black line). The cross-wiring configuration enables rectification along the diagonal direction, allowing for more accurate graph representation. **f** A module utilizing probabilistic switching devices in the diagonal shorted CBA. The right panel illustrates the probabilistic graph and the electrical characteristics of probabilistic switching devices. **g**, **h** A schematic for applying voltage to extract connectivity from the graph mapped onto the crossbar. A 9 × 9 graph can be directly mapped onto a 9 × 9 diagonal shorted CBA. The connectivity between the two nodes is determined by applying voltage and grounding to the word line and bit line of the crossbar.

reflects non-uniform topological properties[13]. Reservoir computing using virtual graphs was demonstrated by applying time-domain inputs to memristors with such dynamic characteristics[68]. This method effectively generates non-Euclidean graphs by adopting the fading memory of the memristor, but its expressive capacity remains limited.

Other recent research focused on mapping the adjacency matrix of graphs to the CBAs. A two-dimensional $N \times N$ self-rectifying memristive CBA with modified diagonal components can store adjacency matrix data of a graph with N nodes[14–17,21]. Each (u, v) element of the adjacency matrix can be mapped based on the conductance of the memristor at uth word line (WL) and vth bit line (BL) of the CBA. The first self-rectifying memristor-based EGC hardware utilized a shorted diagonal structure[14,17] (Fig. 2d). This array maps the graph by allowing reverse current flow (from BL to WL) in the diagonal devices (orange curve of Fig. 2d) while restricting current flow to the forward direction (from WL to BL) in the off-diagonal devices (black curve of Fig. 2d). Two approaches were explored to implement shorted diagonals: metal-at-diagonal CBA (m-CBA)[14] and soft breakdown-at-diagonal CBA[17]. The latter approach offers an advantage over the former, as it allows for dynamic changes in the state of the nodes.

In another self-rectifying memristor-based EGC structure, the cross-wired CBA (cw-CBA) introduces a cross-wiring process to the conventional CBA, where the diagonal elements are connected through crossed WL and BL (Fig. 2e)[15]. Since the diagonal components are still composed of self-rectifying memristors, they retain the ability to modify the node states of the graph. This structure also allows the diagonal elements to flow reverse current while blocking forward current, suppressing undesired current within the array, and further enhancing the precision of graph representation[15].

Research on probabilistic graph processing using probabilistic switching devices in the m-CBA was also proposed[21]. In this work, multiple m-CBA structures were used to implement probabilistic connectivity (Fig. 2f). When the identical adjacency matrix mapping is applied to the individual m-CBAs in Fig. 2f, probabilistic state changes occur due to device property, resulting in a probabilistic graph representation. Such a physically implemented probabilistic graph modeling system enables probabilistic graph algorithms such as steady-state estimation and PageRank[69].

Moreover, methods for extracting connectivity from the self-rectifying memristor-based EGC have been proposed (Fig. 2g). The

https://doi.org/10.1038/s41467-025-63494-z

example graph contains nine nodes (left panel) and can be mapped onto a 9 × 9 shorted diagonal crossbar (right panel). To determine the connectivity between two nodes (red nodes) in the original graph, voltage and ground are applied to the WL and BL of the CBA, respectively (red boxes in Fig. 2g). Selected WL and BL correspond to the source and target node in the example graph, respectively. The amount of current output to the ground is interpreted as a measure of the node pair's connectivity strength. This method is referred to as the single-ground method (SGM). SGM efficiently extracts information from the original graph without preprocessing. Moreover, when the ground is applied to all BLs of CBA (red boxes in Fig. 2h), a parallel circuit is formed by the devices connected to the selected WL, and information regarding the edges connected to the selected node is converted into current. The magnitude of the output current enables the estimation of the degree and 1-hop connection of the selected node. This method of quantifying the direct connectivity of individual nodes using CBA is referred to as the multi-ground method, which can be effectively used with the SGM to describe the graph's structural characteristics.

The system needs to be designed with a structure more advanced than the peripheral circuits of conventional CBA. CBAs for high-density memory typically require random access to a single memory element. In particular, for applying CBAs in EGC, a TIA-ADC capable of current sensing in the analog domain and a multi-channel DAC structure are required for implementing a voltage scheme during the writing process. Moreover, the inclusion of a switch matrix to allow a more flexible selection of WLs and BLs should be considered.

In summary, previously reported EGC structures include nanowire[8,10,11], thermal interaction, delay[18], and self-rectifying memristor-based EGCs[14–17,21]. Nanowire and thermal interaction approaches are generally limited to grid or Euclidean graphs, while delay-based EGCs offer less flexibility in non-Euclidean graph representation (Table 1). Therefore, the following sections will focus primarily on self-rectifying memristive CBA-based approaches in further discussion of the EGC.

Figure 3 shows three different QGC structures: probabilistic computing (p-computing), ONN, and HNN. Although each structure differs in its specific design choice, they can be interpreted as graph structures with physical nodes and edges (coupling) between them. The right panels of each structure depict the implementation of nodes and edges in graphs.

P-computing offers a promising approach using p-bits—binary units with tunable stochasticity that mimic qubit behavior without requiring environmentally sensitive superposition (Fig. 3b)[26,27]. Various methods were adopted to generate p-bit, including digital complementary metal-oxide-semiconductor (CMOS) devices[26–32], stochastic magnetic tunnel junction (MTJ)[33,39], bistable resistors[34], and volatile memristive devices[35–38]. Although their functional operations are similar, each design choice requires a different p-bit structure for nonlinear activation and thresholding. Then, these p-bits form a network in which each p-bit is interconnected with others, with the connections represented by a weight matrix $\boldsymbol{W}$. Even though these edges encode synaptic weights between p-bits, they are usually implemented as theoretical edges and calculated in peripheral CMOS circuits such as field programmable gate arrays (FPGAs). P-computing proved efficient for logic operations and combinatorial optimization problems, as discussed in the next section.

The structures of ONN, another variant of QGC, utilize oscillator-based computing for low-power computation by using the phase and frequency of oscillation as information carriers[41] (Fig. 3c). Recently, Ising machines based on ONN showed promising results in computationally demanding problems solvable in nondeterministic polynomial time. These works implemented oscillatory circuits such as CMOS ring oscillators[42,46], phase-transition devices[48–51], and single electron transistors[53] as nodes (upper panel of Fig. 3c). Then, a network was

Table 1 | Comparison of the graph representation capabilities of various EGC and QGC approaches

	Method	Node programmability	Edge programmability	Graph reconfigurability	Connectivity	Scalability (N)	References
EGC (Euclidean)	Nanowire network	Incapable	Fully capable	No	Limited	N	8,10,11
	Thermal interaction	Incapable	Partially capable	No	Limited	N^2	18
EGC (Non-Euclidean)	Shorted diagonal CBA (metal)	Incapable	Fully capable	Partially capable	All-to-all	N^2	14
	Shorted diagonal CBA (memristor)	Fully capable	Fully capable	Yes	All-to-all	N^2	17
	Cross-wired CBA	Fully capable	Fully capable	Partially capable	All-to-all	N^2	15
	Multi-CBA (probabilistic graph)	Incapable	Fully capable	Partially capable	All-to-all	N^2	21
QGC	Probabilistic computing	Fully capable	Fully capable	Yes	All-to-all	N	26–39
	ONN (Electrical impedance)	Fully capable	Incapable	No	Limited	N^2	48–51
	ONN (Acoustic and magnetic)	Partially capable	Fully capable	No	Limited	N	54
	HNN	Fully capable	Fully capable	Yes	All-to-all	N^2	56–58

https://doi.org/10.1038/s41467-025-63494-z

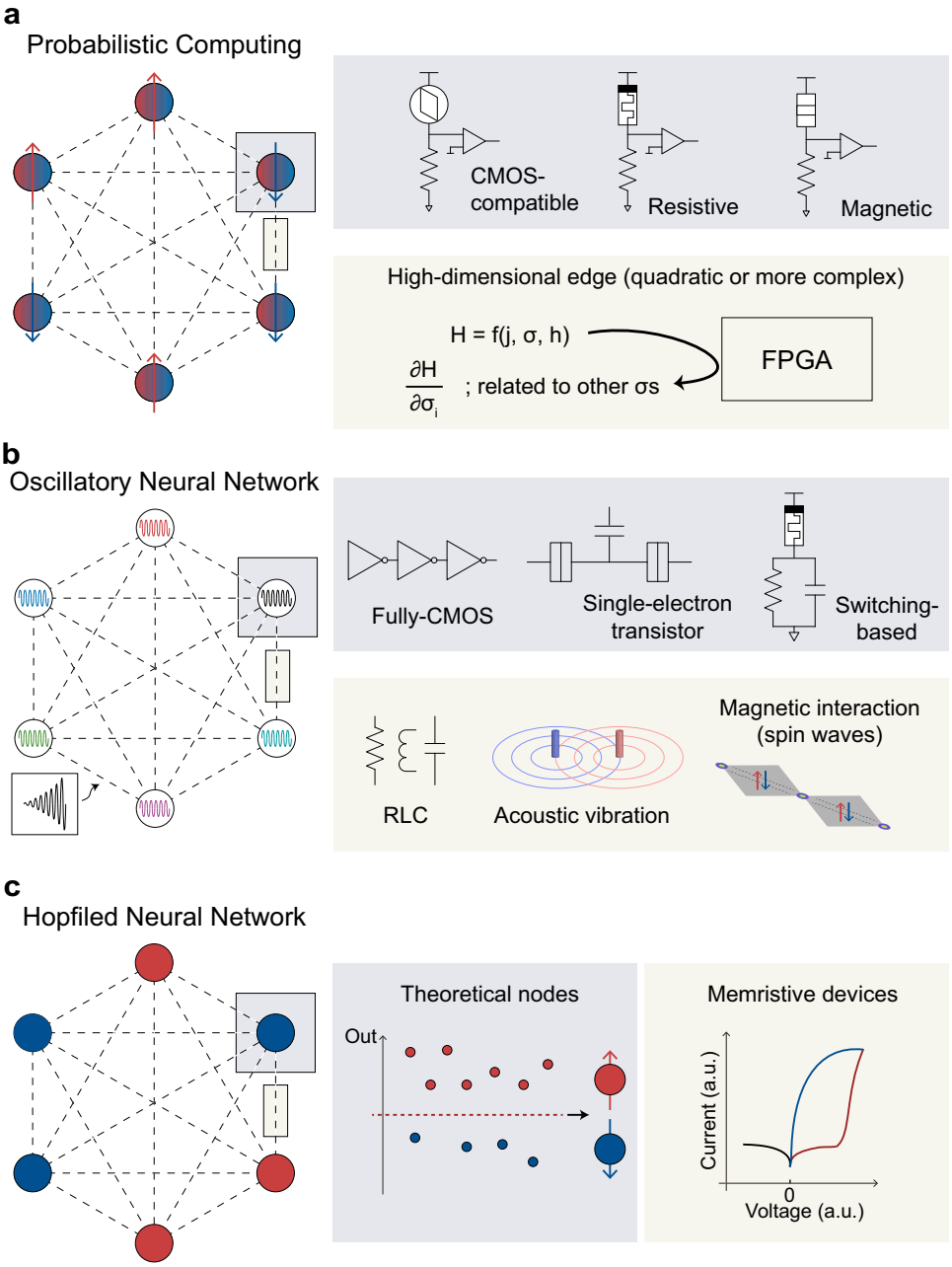

Fig. 3 | Hardware structures for QGC. a Hardware structure for p-computing. The upper panel shows nodes of p-computing, p-bit, which are usually composed of nanodevices. **b** Hardware structure for ONN. Oscillatory nodes are coupled through various physical interactions. The left panel shows an injected signal for phase locking. **c** Hardware structure for HNN. It shows a hardware implementation based on CBA, where memristive devices are used for graph mapping.

constructed by coupling the oscillators according to the problem type. For instance, capacitive coupling of electrical oscillators tends to increase the probability of out-of-phase behavior, which can guide the system toward a stable state determined by the problem's constraints[62]. Coupled oscillatory networks were also demonstrated by coupling the nodes with acoustic vibration or magnetic interaction[54].

Lastly, the hardware structure of HNN, a recurrent neural network used for various optimization and content-addressable memory applications, is illustrated in Fig. 3d. The upper panel shows that a memristive CBA is employed for network mapping and parallel computation through vector-matrix multiplication[55–58]. Specifically, the edges of the given network are mapped to the conductance of memristive CBA, and read voltages are applied to determine the binary state of the nodes. These nodes are "theoretical" because they are not physically manifested but are defined by thresholding multiplication results. Table 1 summarizes the graph representation capabilities of the various reported QGC approaches.

Various applications of EGC and QGC

EGC offers unique advantages for implementing graph algorithms and solving real-world problems across various fields. Figure 4a–c illustrates how EGC-based graph algorithms perform compared to conventional approaches in pathfinding, link prediction, and community detection tasks.

Pathfinding algorithms rely on estimating the distance from the current node to the target node. In EGC, since electrical current naturally flows along the shortest path in the graph, the inverse of the current between nodes can be used as a measure of distance. The

https://doi.org/10.1038/s41467-025-63494-z

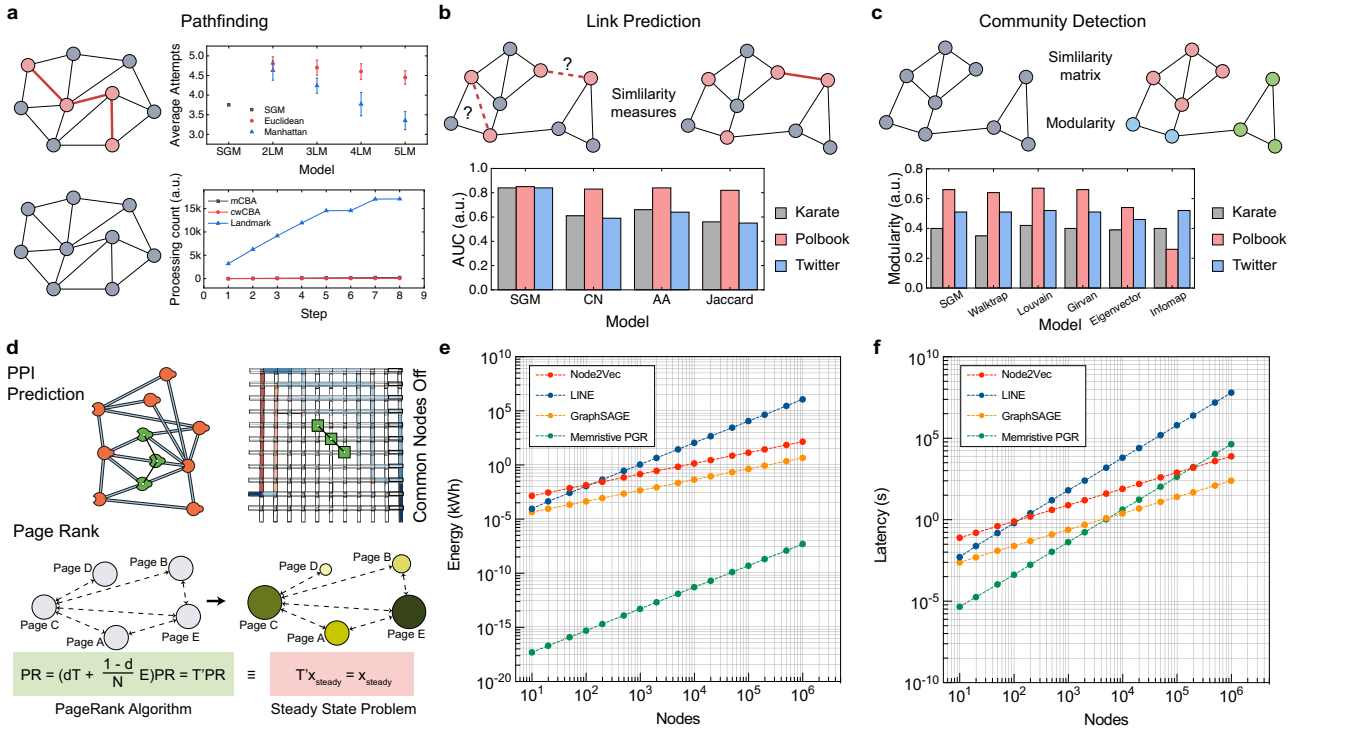

Fig. 4 | Comparison of the performance of EGC in various applications with existing models. a Results of pathfinding (upper panel) and on-condition pathfinding (lower panel) using EGC and LM algorithm. In the upper panel, each symbol represents the average number of attempts, and the error bars denote the standard deviation across trials. **b**, **c** Results of link prediction and community detection using EGC and software-based algorithms. **d** EGC-based PPI prediction and PageRank algorithm. In the PPI network-mapped CBA, a 3-hop similarity for the PPI prediction can be achieved by deactivating the common proteins between two proteins (upper panel). The PageRank algorithm can be implemented with steady-state estimation of the EGC-based probabilistic graph. **e**, **f** Comparison of energy consumption and latency between CPU-based graph embedding and memristive PGR. The energy consumption and latency of the CPU-based graph embedding were calculated based on a single core of the Intel Core i7-10700K. The memristive PGR results were estimated based on the power and latency of the devices and peripheral circuits involved in physically mapping the graph. The graph density was fixed at 0.1 for all comparisons.

upper panel of Fig. 4a compares the performance of the CBA-based EGC to that of traditional landmark (LM) algorithms[70]. In this case, a graph with nine nodes and a density of 0.3 is used to evaluate the average number of attempts required to find the shortest path. The results show that EGC's SGM current-based pathfinding outperforms the LM algorithm when the LM algorithm uses up to four LM nodes. This outperformance results from the fact that EGC does not require graph embedding. While LM accuracy improves with the number of LM nodes, it comes at the cost of increased computational complexity and the risk of information loss during the embedding process. In contrast, EGC stores the graph structure directly in the hardware and utilizes electrical current along the shortest paths to estimate distance, providing higher accuracy and lower computational load compared to LM methods.

The lower panel of Fig. 4a presents a more complex task: on-condition pathfinding in a graph with 40 nodes and a density of 0.1, where several nodes may become unavailable during the pathfinding process (dynamic graph). In LM, every time a node becomes unavailable, the graph must be re-embedded, leading to a high processing cost. On the other hand, EGC only requires modification of the graph data stored in the CBA, making it far more suited to dynamic pathfinding. Among the EGC methods, cw-CBA is more effective than m-CBA because it allows direct modification of node states, while m-CBA requires physically turning off the edges of the unavailable node.

For link prediction tasks, which aim to forecast potential future connections between nodes, EGC offers a distinct advantage. The key to effective link prediction is the ability to evaluate multi-hop connections between nodes, typically based on their similarity[71]. In EGC, the SGM current between two nodes serves as a similarity measure. The lower panel of Fig. 4b compares the link prediction performance of SGM current with conventional similarity indices such as Common Neighbor[72], Adamic Adar[73], and Jaccard[74] across three datasets[14,75]. In all datasets, the SGM current consistently outperforms the other models, attributed to SGM's ability to reflect the continuous nature of graphical topology. In contrast, conventional similarity indices are discrete and may lack the resolution to distinguish subtle differences between node pairs.

Similarly, community detection algorithms, which identify clusters of nodes that are more densely connected than the rest of the graph, also benefit from EGC's capabilities. The process typically involves creating a similarity matrix for all node pairs and merging highly connected nodes into clusters to maximize modularity[76]. Community detection is considered successful when the maximum modularity is high, as this reflects the quality of the identified clusters. EGC's SGM provides a comprehensive assessment of direct and indirect connections, making it an effective similarity measure for community detection. The lower panel of Fig. 4c compares the modularity achieved by the SGM current with other algorithms across three datasets. The results show that EGC's SGM current is comparable to the latest models, indicating that it effectively captures the high-dimensional information of the graph, while still consuming much lower computational costs.

Additionally, EGC can address real-world problems, such as protein-protein interaction (PPI) prediction or the PageRank algorithm. PPIs are fundamental to understanding the molecular mechanisms underlying various biological processes, which are essential for drug discovery and disease modeling[4,77] The PPI follows the key-lock principle, where proteins with complementary structures interact

https://doi.org/10.1038/s41467-025-63494-z

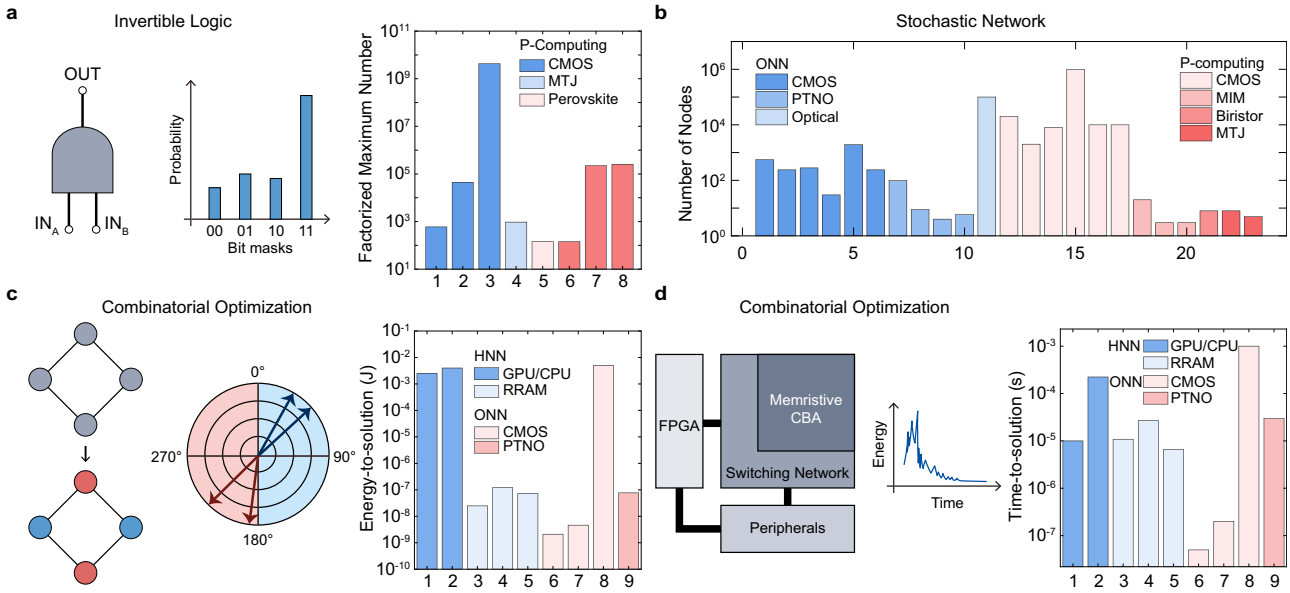

Fig. 5 | Comparison of the performance of QGC in various applications. **a** Invertible logic of QGC. The left panel shows QGC-based invertible logic, and the right panel depicts the factorized maximum number from various structures. **b** The QGC-based implementation of a stochastic network and the number of nodes from different structures. **c**, **d** Combinatorial optimization using phase-based ONN (**c**) and iterative process in HNN (**d**). The right panels of (**c**, **d**) show ETS and TTS in combinatorial optimization.

selectively, requiring specialized similarity measures for accurate modeling[78,79]. EGC is suited for this because SGM current paths can be adjusted to match PPI characteristics (upper panel of Fig. 4d). It has been shown that SGM using EGC hardware outperforms traditional similarity measures in predicting PPIs across diverse datasets[15].

The PageRank algorithm ranks nodes in a graph based on the likelihood that a random walker would land on a particular node[69]. PageRank is mathematically represented as a steady-state problem (lower panel of Fig. 4d). To implement PageRank with EGC, it is necessary to map probabilistic networks onto the hardware. Recent research implemented PageRank using a multiple m-CBA-based probabilistic graph[21]. It utilized device-to-device variations in multiple m-CBAs for probability-voltage mapping, achieving steady-state estimation with reduced time complexity.

Building on EGC's demonstrated high performance across various tasks, its advantages extend beyond accuracy to include significant hardware efficiency. One area where this is particularly evident is graph embedding, a notoriously resource-intensive process on conventional CPU-based hardware[64]. As graph sizes increase, the energy consumption and latency required for embedding grow substantially. Figure 4e, f compares the energy consumption and latency of various graph embedding algorithms[63,80,81] implemented on CPU hardware with the memristive PGR approach. By simplifying the embedding process, the memristive PGR method achieves not only superior energy efficiency, far surpassing traditional CPU-based algorithms, but also competitive latency performance, especially for graphs with fewer than 5000 nodes. Also, EGC offers efficient similarity computation after embedding, where the SGM retrieves node-pair similarity with a single voltage application. In contrast, conventional hardware typically relies on vector-based distance formulas, where the latency scales with the embedding dimension. While EGC latency may increase when the current path spans multiple devices due to increased R-C delay, this effect is typically minimal in real-world graphs, which tend to maintain short average path lengths owing to their small-world characteristics. When such hardware delays do occur, they can be mitigated by enhancing the on-state performance of the device, particularly by reducing resistance and ensuring Ohmic behavior.

In contrast, QGC is widely used for different applications from EGC by computing the time evolution of the networks. Figure 5 shows three representative applications of QGC. First, the invertible calculation was extensively researched using p-computing (Fig. 5a). Invertible Boolean logic and integer factorization were achieved by the energy-based representation of the problem. The right panel of Fig. 5a shows the maximum number factorized using networks of p-bits. The integer factorization of larger numbers requires complex graphs and a finely tuned annealing schedule. Subsequently, p-computing based on mature CMOS technology achieved a higher number of factorizations more effectively than emerging memories.

ONN and p-computing are also used in stochastic network implementation (Fig. 5b). Although these methods aim for general computing purposes using network implementation, most works have focused on Ising machines for combinatorial optimization. It shows that the number of nodes in p-computing is generally larger than that of ONN due to the difficulty in the physical coupling of oscillators in ONN. Moreover, nodes based on CMOS technology, such as ring oscillator and linear-feedback shift register[31,46], could achieve more complex networks than phase-transition nano-oscillators, memristors, and MTJs did[33,35,48].

The third example illustrates the combinatorial optimization of QGC, further expanding the application of stochastic networks (Fig. 5c, d). The left panel of Fig. 5c shows the phase difference used for Max-Cut problem-solving through ONN, while the left panel of Fig. 5d depicts the iterative process to find the global minimum in HNN. Two qualitative parameters of energy-to-solution (ETS) and time-to-solution (TTS) are shown in the right panel of Fig. 5c, d, respectively, which refer to energy and time to obtain the global minimum in the given optimization problems. CMOS implementation[59] consumes higher energy than emerging memory devices[48,56–58]. In addition, various works on ONN and HNN demonstrate TTS on the scale of nanoseconds to microseconds. However, determining the optimal approach remains challenging because the TTS is often calculated based on theoretical switching speeds or predicted clock speeds in scaled devices and is highly sensitive to the specific tuning of the algorithms employed.

Opportunities and technical challenges for future development of EGC and QGC

This section discusses the achievements and potential enhancements in device technology and integration for EGC and QGC. These improvements aim to enhance performance or broaden the scope of solvable problems for both graph computing methods.

Currently, self-rectifying memristors are predominantly used as the fundamental units for EGC (upper left panel of Fig. 6a). These self-rectifying memristors typically exhibit nonlinear current–voltage (I–V) characteristics, posing challenges for analyzing large-scale graphs. Nonlinear I–V behavior makes it difficult to extract information from graph paths with longer distances. In such cases, multiple memristors arranged in series along a graph path can cause voltage division, resulting in SGM currents that fall below measurable levels. Even when measurable, the excessively high resistance can significantly increase hardware latency. Although lower currents can enhance the energy efficiency of the CBA, the energy consumption in ADCs and DACs within peripheral circuits is much higher than that in the CBA. Therefore, decreasing latency in the CBA's unit devices is more beneficial. To accurately capture information from distant graph paths with low latency, devices must possess rectifying and Ohmic characteristics, enabling linear I–V behavior and high current flow under specific biases, while exhibiting nonlinear I–V behavior and restricted current flow under opposite biases. This performance can be achieved by integrating a diode with a memristor with Ohmic properties (upper middle panel of Fig. 6a).

Furthermore, the hardware becomes reconfigurable by developing devices that manipulate rectification direction, thereby enhancing area efficiency. By allowing each device to function interchangeably as a node or an edge, processes such as metal vias or cross-wiring are unnecessary. This reconfigurability enables the same CBA hardware to represent planar, multilayer, and hierarchical graphs. Developing such devices may involve integrating ferroelectric materials, which enable polarization-based resistive switching, with memristors (upper right panel of Fig. 6a).

Also, expanding the range of solvable problems within EGC is essential. Although various graph structures exist, current EGC research is mainly limited to planar graphs (lower left panel of Fig. 6a). Representing multilayer and hierarchical graphs requires expanding CBAs laterally or vertically and utilizing crossbar tensor architectures, where multiple CBAs are arranged in a tensor structure (lower middle panel of Fig. 6a). Moreover, existing EGC hardware has only been able to verify the current value. If current path information could be obtained physically, it would enable one-shot pathfinding and applications to complex problems, such as the traveling salesperson problem. This goal could be achieved using physical elements such as light or charge. For example, incorporating light-emitting diodes at node devices would allow optical verification of current paths, or connecting parallel capacitors to node devices could enable the detection of current paths by measuring the charge on each capacitor immediately after SGM operations (lower right panel of Fig. 6a).

As discussed in Fig. 3, QGC utilizes nanomaterials to achieve high energy efficiency. As FPGA-based approaches benefit from high integration density and flexibility, QGC should utilize the massive parallelism and dynamics of nanomaterials, similar to AI accelerators[19]. However, QGC requires CMOS peripheral circuits, which account for most of the energy consumption and hinder achieving the theoretical switching and reading speeds. Consequently, current research focuses more on increasing the number of two-level nodes to enhance parallelism rather than improving individual devices.

Nevertheless, many real-world optimization problems require more complex nodes and interactions than the quadratic interactions of binary nodes typically handled by QGC. For example, satisfiability problems can be modeled as hypergraph problems, requiring nodes with multiple levels as the clause number increases. Furthermore, nodes interact at higher orders in these problems, which conventional QGC cannot implement without external CMOS calculations (middle panel of Fig. 6b). Recently, third-order interactions have been demonstrated by duplicating memristive devices with limited functionality[55]. Moreover, auxiliary nodes were used to translate a multi-level node into binary nodes for complex optimization problems in QGC[40].

In the long term, the device itself should implement additional functionalities to decrease area overhead and increase integration density (right panel of Fig. 6b). For instance, three-terminal devices showed improved solution quality through continuous modulation of an additional terminal to implement adiabatic annealing[60]. The increased functionality will also enable parallel calculation of high-order coupling of p-computing, which was calculated in an FPGA in the previous works. In ONN, both complex nodes and edges can be realized through device engineering. The number of levels of oscillatory nodes is determined by dividing the phase into several subgroups; therefore, more precise edge coupling weights and uniform oscillations can expand the range of solvable problems using ONN. Moreover, as the edges in ONN need to correspond to those in the problem graph each time, their coupling strengths must be reconfigurable and programmable. Although electrically programmable devices, such as ferroelectric materials, are available, their practical implementation requires further development, particularly in improving the on-off ratio and endurance.

While EGC and QGC hardware have been implemented with up to tens to thousands of nodes, the ultimate goal for both hardware is to solve large-scale real-world graph problems. However, denser integration with scaled devices cannot support millions of nodes because the edge connection and coupling will increase quadratically. Consequently, full-stack research, in addition to device integration, is required.

One practical approach is clustering, which divides large graphs into manageable units by separating them into inter-cluster and intra-cluster graphs. This hierarchical analysis enables handling complex, large-scale graphs without the need for infinitely expanding hardware. It was also adopted in p-computing to realize asynchronous parallel computing[42]. Moreover, clustering helps address the inefficiency of mapping sparse graph data onto dense CBAs, where mapping N nodes requires an N^2-sized CBA. Decreasing node numbers through clustering mitigates the discrepancy between the graph's sparsity and the CBA's density, enhancing area efficiency. To implement clustering-based analysis, hardware must support time-multiplexing or spatial-multiplexing operations. Achieving this would require device engineering to enable computing cores with plausible performance and architecture-level engineering to control them efficiently. Specifically, device latency and variation should be diminished to calculate on larger clustered graphs.

Outlook

If advancements in performance, the range of solvable problems, and integration, as proposed in Fig. 6, are successfully achieved, EGC and QGC could become general-purpose systems capable of addressing real-world graph tasks (Fig. 7). EGC excels at effortlessly extracting connectivity information from graphs. Utilizing the characteristic of current flowing only from higher to lower potentials, SGM naturally eliminates redundant paths that revisit specific nodes, thereby isolating meaningful graph connections. This capability represents an intractable function that classical computing systems cannot efficiently perform through matrix operations.

One significant real-world application where EGC can be effectively utilized is analyzing knowledge graphs. Knowledge graphs are structured representations of information that model entities and their interrelationships, typically based on ontologies that define the types and properties of these entities and relationships[82]. To mitigate

https://doi.org/10.1038/s41467-025-63494-z

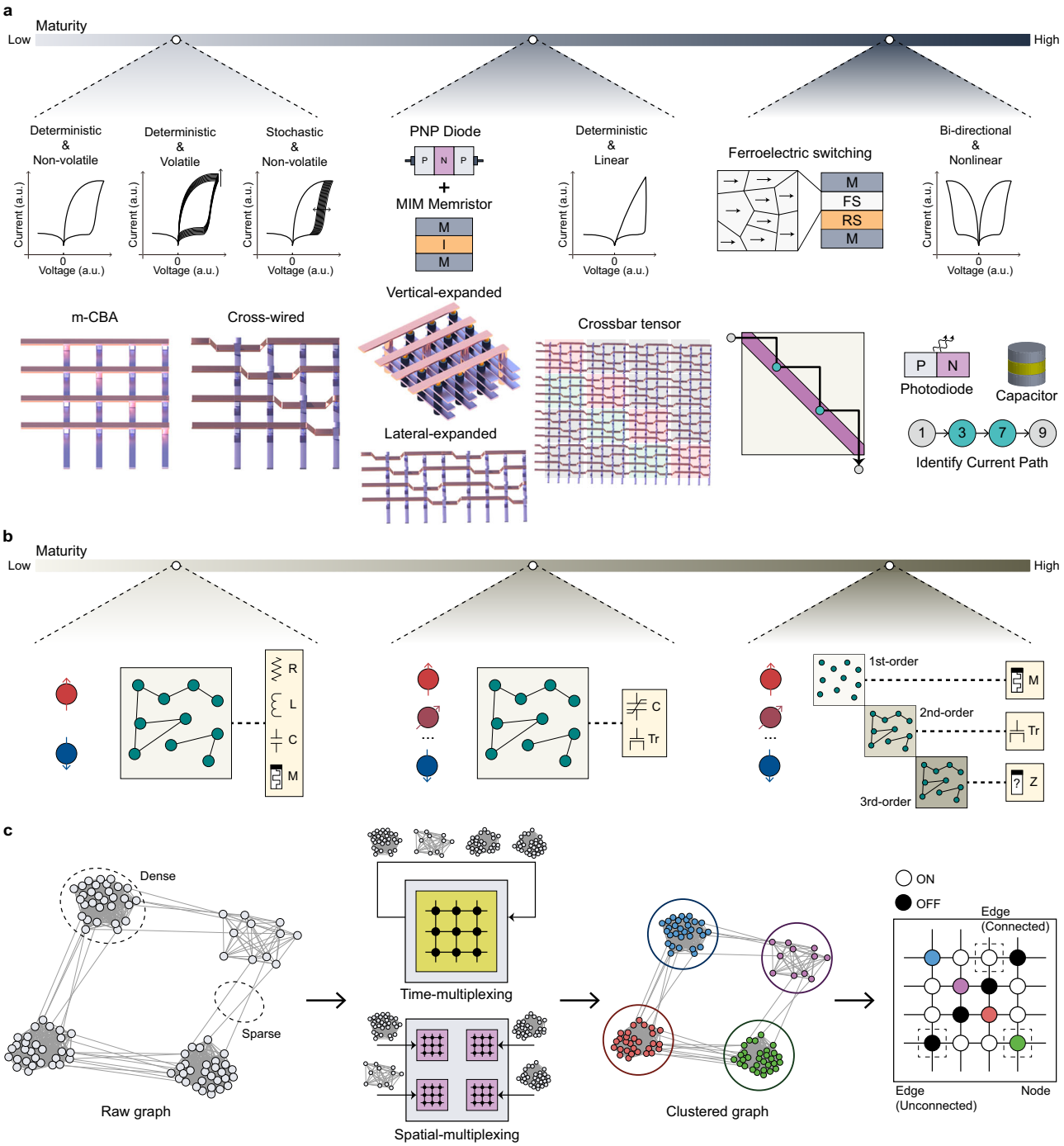

Fig. 6 | Technical challenges and opportunities for future development of EGC and QGC. a Current and future directions for EGC. The leftmost panel illustrates the current state of EGC, showing various types of self-rectifying memristors (upper panel) integrated with CBA structures designed for planar graphs (lower panel). Progressing to the right, the figure highlights research directions to improve hardware efficiency and expand the scope of solvable problems in EGC. Developing rectifying devices with Ohmic characteristics under forward bias is required to decrease latency, along with advancements in CBA structures to support multi-dimensional and hierarchical graphs (middle panel). Furthermore, achieving reconfigurability in EGC hardware requires exploring devices and CBA structures capable of physically representing SGM current paths. Obtaining precise SGM current path information plays a critical role in mitigating EGC's limitation of sequential operation. **b** Current and future directions for QGC. The leftmost panel depicts the current QGC, utilizing a two-level system and a low-dimensional graph. Progressing to the right, the future vision for QGC involves adopting high-dimensional, multifunctional devices to enable multi-level systems with high-order interaction edges. **c** Suggested clustering for EGC and QGC. A raw graph is first clustered into smaller graphs, which can be processed in a single core. After either time- or spatial multiplexing, the processed results from clustered graphs are integrated for the final output.

persistent issues in large language models, such as a lack of interpretability, low reliability, and hallucinations, there is a growing effort to integrate knowledge graphs with large language models[83]. Knowledge graphs are extensive graphs composed of inter-domain and intra-domain connections, making them challenging to analyze using classical computers. Hierarchical and multilayer EGC hardware can process knowledge graphs rapidly, facilitating their integration with large language models.

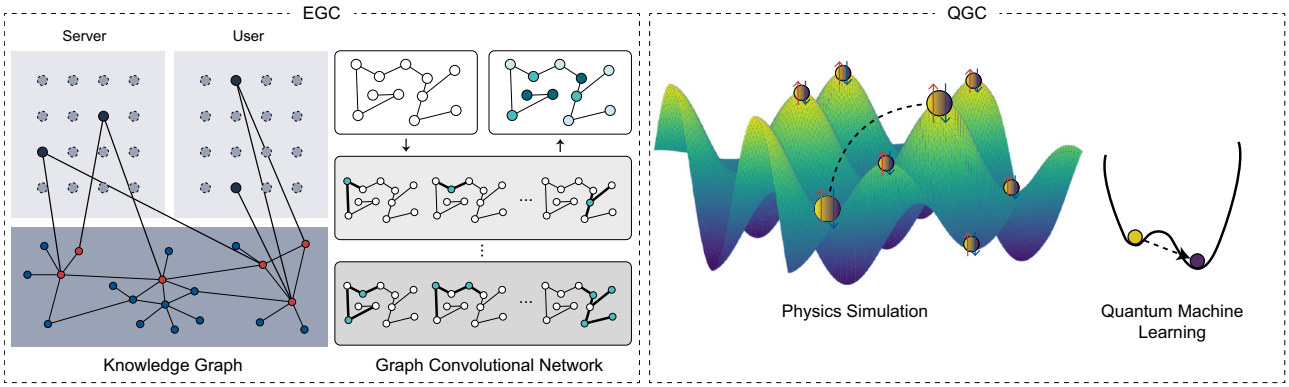

Fig. 7 | Future applications of EGC and QGC. Four possible applications of EGC and QGC are shown. EGC can be applied to KG and GCN, whereas QGC has potential usage in physics simulation and quantum machine learning. While EGC excels at extracting connectivity and path information from various types of graphs, QGC is more adequate in describing complex systems' time evolution and optimization.

The expanding field of deep learning opens opportunities to integrate EGC into graph convolutional networks[84]. Traditional graph convolutional networks rely on matrix operations that can cause over-smoothing and limit performance[85]. EGC's SGM, with its multi-hop connectivity and exclusion of backtracking paths, simplifies feature extraction and mitigates over-smoothing. EGC hardware can use intermediate propagation data to improve graph convolutional networks' accuracy and employs time multiplexing for scalable analysis of larger graphs.

QGC, on the other hand, can be adopted in various applications, from cryptography to physics simulation and machine learning. Although QGC also aims to implement general-purpose processors such as oscillator-based computing, it is more likely to have specific purposes. For instance, p-computing will be used in quantum simulation and optimization, whereas ONN and HNN will be more specifically used in machine learning or optimization problems. Either way, all graph computing methods will be first demonstrated as area-specific computing architectures along with CPU, and eventually developed to implement more general logic and computations to enable power-efficient graph computing. It will require research in various aspects of programming languages, algorithms, hardware-software interfaces, and device technologies.

References

1. Ortega, A., Frossard, P., Kovacevic, J., Moura, J. M. F. & Vandergheynst, P. Graph signal processing: overview, challenges, and applications. *Proc. IEEE* **106**, 808–828 (2018).
2. Shi, X. et al. Graph processing on GPUs: a survey. *ACM Comput. Surv.* **50**, 1–35 (2018).
3. De Domenico, M., Granell, C., Porter, M. A. & Arenas, A. The physics of spreading processes in multilayer networks. *Nat. Phys.* **12**, 901–906 (2016).
4. Aittokallio, T. & Schwikowski, B. Graph-based methods for analysing networks in cell biology. *Brief. Bioinform.* **7**, 243–255 (2006).
5. *Graph Algorithms in the Language of Linear Algebra. Society for Industrial and Applied Mathematics* https://doi.org/10.1137/1.9780898719918 (2011).
6. Latapy, M. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theor. Comput. Sci.* **407**, 458–473 (2008).
7. Stathis, D., Vourkas, I. & Sirakoulis, G. C. Shortest path computing using memristor-based circuits and cellular automata. in *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* https://doi.org/10.1007/978-3-319-11520-7_41 (2014).
8. Milano, G. et al. In materia reservoir computing with a fully memristive architecture based on self-organizing nanowire networks. *Nat. Mater.* **21**, 195–202 (2022).
9. Lilak, S. et al. Spoken digit classification by in-materio reservoir computing with neuromorphic atomic switch networks. *Front. Nanotechnol.* https://doi.org/10.3389/fnano.2021.675792 (2021).
10. Zhu, R. et al. Information dynamics in neuromorphic nanowire networks. *Sci. Rep.* **11**, 13047 (2021).
11. Diaz-Alvarez, A. et al. Emergent dynamics of neuromorphic nanowire networks. *Sci. Rep.* **9**, 4920 (2019).
12. Appeltant, L. et al. Information processing using a single dynamical node as complex system. *Nat. Commun.* **2**, 468 (2011).
13. Guo, Y. et al. Generative complex networks within a dynamic memristor with intrinsic variability. *Nat. Commun.* **14**, 6134 (2023).
14. Jang, Y. H. et al. Graph analysis with multifunctional self-rectifying memristive crossbar array. *Adv. Mater.* **35**, e2209503 (2023).
15. Jang, Y. H. et al. Cross-wired memristive crossbar array for effective graph data analysis. *Adv. Mater.* **2311040**, 1–14 (2023).
16. Jang, Y. H. et al. Spatiotemporal data processing with memristor crossbar array-based graph reservoir. *Adv. Mater.* **36**, e2309314 (2023).
17. Woo, K. S. et al. Memristors with Tunable Volatility for Reconfigurable Neuromorphic Computing. *ACS Nano* **18**, 17007–17017 (2024).
18. Kim, G. et al. Mott neurons with dual thermal dynamics for spatiotemporal computing. *Nat. Mater.* **23**, 1237–1244 (2024).
19. Xia, Q. & Yang, J. J. Memristive crossbar arrays for brain-inspired computing. *Nat. Mater.* https://doi.org/10.1038/s41563-019-0291-x (2019).
20. Woo, K. S., Williams, R. S. & Kumar, S. Localized conduction channels in memristors. *Chem. Rev.* **125**, 294–325 (2024).
21. Jang, Y. H. et al. Memristive crossbar array-based probabilistic graph modeling. *Adv. Mater.* **36**, 2403904 (2024).
22. Carpinone, A., Giorgio, M., Langella, R. & Testa, A. Markov chain modeling for very-short-term wind power forecasting. *Electr. Power Syst. Res.* https://doi.org/10.1016/j.epsr.2014.12.025 (2015).
23. Nagargoje, P. & Baviskar, M. Uncertainty handling in big data analytics survey, opportunities and challenges. *Int. J. Comput. Sci. Eng.* https://doi.org/10.26438/ijcse/v9i6.5963 (2021).
24. Chowdhury, S. et al. A full-stack view of probabilistic computing with p-bits: devices, architectures, and algorithms. *IEEE J. Explor. Solid State Comput. Devices Circuits* https://doi.org/10.1109/JXCDC.2023.3256981 (2023).
25. Kaiser, J. & Datta, S. Probabilistic computing with p-bits. *Appl. Phys. Lett.* **119**, https://doi.org/10.1063/5.0067927 (2021).

26. Smithson, S. C., Onizawa, N., Meyer, B. H., Gross, W. J. & Hanyu, T. Efficient CMOS invertible logic using stochastic computing. *IEEE Trans. Circuits Syst. I Regul. Pap.* **66**, 2263–2274 (2019).
27. Patel, S., Canoza, P. & Salahuddin, S. Logically synthesized and hardware-accelerated restricted Boltzmann machines for combinatorial optimization and integer factorization. *Nat. Electron.* **5**, 92–101 (2022).
28. Aadit, N. A. et al. Massively parallel probabilistic computing with sparse Ising machines. *Nat. Electron.* **5**, 460–468 (2022).
29. Yamaoka, M. et al. A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing. *IEEE J. Solid State Circuits* **51**, 303–309 (2016).
30. Baity-Jesi, M. et al. Janus II: a new generation application-driven computer for spin-system simulations. *Comput. Phys. Commun.* https://doi.org/10.1016/j.cpc.2013.10.019 (2014).
31. Sutton, B. et al. Autonomous probabilistic coprocessing with petaflips per second. *IEEE Access* **8**, 157238–157252 (2020).
32. Singh, N. S. et al. CMOS plus stochastic nanomagnets enabling heterogeneous computers for probabilistic inference and learning. *Nat. Commun.* **15**, 2685 (2024).
33. Borders, W. A. et al. Integer factorization using stochastic magnetic tunnel junctions. *Nature* **573**, 390–393 (2019).
34. Kim, J. et al. Fully CMOS-based p-bits with a bistable resistor for probabilistic computing. *Adv. Funct. Mater.* **2307935**, 1–9 (2024).
35. Woo, K. S. et al. Probabilistic computing using Cu0.1Te0.9/HfO2/Pt diffusive memristors. *Nat. Commun.* **13**, 5762 (2022).
36. Milozzi, A., Ricci, S. & Ielmini, D. Memristive tonotopic mapping with volatile resistive switching memory devices. *Nat. Commun.* **15**, 1–9 (2024).
37. Choi, S. et al. Controllable SiOx nanorod memristive neuron for probabilistic Bayesian inference. *Adv. Mater.* **34**, e2104598 (2022).
38. Park, T. J. et al. Efficient probabilistic computing with stochastic perovskite nickelates. *Nano Lett.* **22**, 8654–8661 (2022).
39. Hamerly, R. et al. Experimental investigation of performance differences between coherent Ising machines and a quantum annealer. *Sci. Adv.* **5**, eaau0823 (2019).
40. Nikhar, S., Kannan, S., Aadit, N. A., Chowdhury, S. & Camsari, K. Y. All-to-all reconfigurability with sparse and higher-order Ising machines. *Nat. Commun.* **15**, 8977 (2024).
41. Csaba, G. & Porod, W. Coupled oscillators for computing: a review and perspective. *Appl. Phys. Rev.* **7**, https://doi.org/10.1063/1.5120412 (2020).
42. Ahmed, I., Chiu, P. W., Moy, W. & Kim, C. H. A probabilistic compute fabric based on coupled ring oscillators for solving combinatorial optimization problems. *IEEE J. Solid State Circuits* https://doi.org/10.1109/JSSC.2021.3062821 (2021).
43. Wang, T., Wu, L., Nobel, P. & Roychowdhury, J. Solving combinatorial optimisation problems using oscillator based Ising machines. *Nat. Comput.* **20**, 287–306 (2021).
44. Mostafa, H., Müller, L. K. & Indiveri, G. An event-based architecture for solving constraint satisfaction problems. *Nat. Commun.* https://doi.org/10.1038/ncomms9941 (2015).
45. Bashar, M. K. et al. Experimental demonstration of a reconfigurable coupled oscillator platform to solve the Max-Cut problem. *IEEE J. Explor. Solid State Comput. Devices Circuits* **6**, 116–121 (2020).
46. Moy, W. et al. A 1,968-node coupled ring oscillator circuit for combinatorial optimization problem solving. *Nat. Electron.* **5**, 310–317 (2022).
47. Wang, T., Wu, L. & Roychowdhury, J. Late breaking results: new computational results and hardware prototypes for oscillator-based Ising machines. In *Proc. Design Automation Conference* https://doi.org/10.1145/3316781.3322473 (2019).
48. Dutta, S. et al. An Ising Hamiltonian solver based on coupled stochastic phase-transition nano-oscillators. *Nat. Electron.* **4**, 502–512 (2021).
49. Maher, O. et al. A CMOS-compatible oscillation-based VO2 Ising machine solver. *Nat. Commun.* **15**, 1–11 (2024).
50. Dutta, S. et al. Experimental demonstration of phase transition nano-oscillator based ising machine. In *Proc. Technical Digest—International Electron Devices Meeting, IEDM* https://doi.org/10.1109/IEDM19573.2019.8993460 (2019).
51. Rhee, H. et al. Probabilistic computing with NbOx metal-insulator transition-based self-oscillatory pbit. *Nat. Commun.* https://doi.org/10.1038/s41467-023-43085-6 (2023).
52. Honjo, T. et al. 100,000-spin coherent Ising machine. *Sci. Adv.* **7**, eabh0952 (2021).
53. Fahmy, H. A. H. & Kiehl, R. A. Complete logic family using tunneling-phase-logic devices. In *Proc. International Conference on Microelectronics, ICM* https://doi.org/10.1109/ICM.2000.884828 (1999).
54. Xu, Y. & Lee, J. E. Y. Mechanically coupled SOI Lamé-mode resonator-arrays: Synchronized oscillations with high quality factors of 1 million. In *Proc. 2013 Joint European Frequency and Time Forum and International Frequency Control Symposium, EFTF/IFC* 133–136 https://doi.org/10.1109/EFTF-IFC.2013.6702184 (2013).
55. Hizzani, M. et al. Memristor-based hardware and algorithms for higher-order Hopfield optimization solver outperforming quadratic Ising machines. In *Proc. IEEE International Symposium on Circuits and Systems* 1–5. https://doi.org/10.1109/ISCAS58744.2024.10558658 (2024).
56. Jiang, M., Shan, K., He, C. & Li, C. Efficient combinatorial optimization by quantum-inspired parallel annealing in analogue memristor crossbar. *Nat. Commun.* **14**, 5927 (2023).
57. Cai, F. et al. Power-efficient combinatorial optimization using intrinsic noise in memristor Hopfield neural networks. *Nat. Electron.* **3**, 409–418 (2020).
58. Lee, S. H. et al. In-materia annealing and combinatorial optimization based on vertical memristive array. *Adv. Mater.* **2410191**, 1–12 (2024).
59. King, A. D., Bernoudy, W., King, J., Berkley, A. J. & Lanting, T. Emulating the coherent Ising machine with a mean-field algorithm. Preprint at *arXiv* https://doi.org/10.48550/arXiv.1806.08422 (2018).
60. Yi, S. I., Kumar, S. & Williams, R. S. Improved hopfield network optimization using manufacturable three-terminal electronic synapses. *IEEE Trans. Circuits Syst. I Regul. Pap.* **68**, 4970–4978 (2021).
61. Arsov, N. & Mirceva, G. Network embedding: an overview. Preprint at *arXiv* https://doi.org/10.48550/arXiv.1911.11726 (2019).
62. Perozzi, B., Al-Rfou, R. & Skiena, S. DeepWalk: online learning of social representations. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* https://doi.org/10.1145/2623330.2623732 (2014).
63. Grover, A. & Leskovec, J. Node2vec: scalable feature learning for networks. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* https://doi.org/10.1145/2939672.2939754 (2016).
64. Cai, H., Zheng, V. W. & Chang, K. C. C. A comprehensive survey of graph embedding: problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.* **30**, 1616–1637 (2018).
65. Hamilton, W. L. Graph Representation Learning Hamilton. *Synthesis Lectures on Artificial Intelligence and Machine Learning* https://doi.org/10.2200/S01045ED1V01Y202009AIM046 (2020).
66. Grover, L. K. Quantum computers can search arbitrarily large databases by a single query. *Phys. Rev. Lett.* **79**, 4709 (1997).
67. Shor, P. W. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. Annual IEEE Symposium on Foundations of Computer Science, FOCS* https://doi.org/10.1109/SFCS.1994.365700 (1994).
68. Jang, Y. H., Han, J.-K. & Hwang, C. S. A review of memristive reservoir computing for temporal data processing and sensing. *Infoscience* **1**, 1–19 (2024).

69. Page, L., Brin, S., Motwani, R. & Winograd, T. The PageRank Citation Ranking: bringing order to the web. *The Web Conference* (1999). https://api.semanticscholar.org/CorpusID:1508503.
70. Goldberg, A. & Harrelson, C. Computing the Shortest Path: A* Search Meets Graph Theory. https://www.microsoft.com/en-us/research/publication/computing-the-shortest-path-a-search-meets-graph-theory/ (2004).
71. Ahmad, I., Akhtar, M. U., Noor, S. & Shahnaz, A. Missing link prediction using common neighbor and centrality based parameterized algorithm. *Sci. Rep.* **10**, 1–9 (2020).
72. Lü, L., Jin, C. H. & Zhou, T. Similarity index based on local paths for link prediction of complex networks. *Phys. Rev. E Stat. Nonlinear Soft Matter Phys.* **80**, 046122 (2009).
73. Adamic, L. A. & Adar, E. Friends and neighbors on the Web. *Soc. Netw.* **25**, 211–230 (2003).
74. Liben-Nowell, D. & Kleinberg, J. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.* https://doi.org/10.1002/asi.20591 (2007).
75. Rossi, R. A. & Ahmed, N. K. The network data repository with interactive graph analytics and visualization. *Proceedings of the AAAI conference on artificial intelligence*. Vol. 29, No. 1 (AAAI Press, 2015).
76. Dabaghi Zarandi, F. & Kuchaki Rafsanjani, M. Community detection in complex networks using structural similarity. *Phys. A Stat. Mech. Appl.* **503**, 882–891 (2018).
77. Abbas, K. et al. Application of network link prediction in drug discovery. *BMC Bioinform.* https://doi.org/10.1186/s12859-021-04082-y (2021).
78. Keskin, O., Tuncbag, N. & Gursoy, A. Predicting protein-protein interactions from the molecular to the proteome level. *Chem. Rev.* **116**, 4884–4909 https://doi.org/10.1021/acs.chemrev.5b00683 (2016).
79. Szilágyi, A., Grimm, V., Arakaki, A. K. & Skolnick, J. Prediction of physical protein-protein interactions. *Phys. Biol.* **2**, 1–16 https://doi.org/10.1088/1478-3975/2/2/S01 (2005).
80. Tang, J. et al. LINE: Large-scale information network embedding. In *Proc. WWW 2015—24th International Conference on World Wide Web* https://doi.org/10.1145/2736277.2741093 (2015)
81. Hamilton, W. L., Ying, R. & Leskovec, J. Inductive representation learning on large graphs. in *Advances in Neural Information Processing Systems* **30**, https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf (2017).
82. Ji, S., Pan, S., Cambria, E., Marttinen, P. & Yu, P. S. A survey on knowledge graphs: representation, acquisition, and applications. *IEEE Trans. Neural Netw. Learn. Syst.* **33**, 494–514 (2022).
83. Yang, L., Chen, H., Li, Z., Ding, X. & Wu, X. ChatGPT is not enough: enhancing large language models with knowledge graphs for fact-aware language modeling. Preprint at *arXiv* https://doi.org/10.48550/arXiv.2306.11489 (2023).
84. Zhang, S., Tong, H., Xu, J. & Maciejewski, R. Graph convolutional networks: a comprehensive review. *Comput. Soc. Netw.* https://doi.org/10.1186/s40649-019-0069-y (2019).
85. Zhang, X., Xu, Y., He, W., Guo, W. & Cui, L. A comprehensive review of the oversmoothing in graph neural networks. in *Communications in Computer and Information Science* https://doi.org/10.1007/978-981-99-9637-7_33 (2024).

Acknowledgements

Y.H.J. is supported by the Sejong Science Fellowship of the National Research Foundation of Korea (Grant No. RS-2024-00342455). This work was also supported by the National Research Foundation of Korea (Grant No. 2020R1A3B2079882).

Author contributions

Y.H.J., J.H., and S.H.L. contributed equally to this work. Y.H.J., J.H., and S.H.L. initiated the paper and developed its outline. Y.H.J., J.H., and S.H.L. wrote the first draft. C.S.H. revised the manuscript and supervised the entire project. All authors approved the submission.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Cheol Seong Hwang.