Journal of Computer Science and Technology

Indexed in: SCIE, EI, Scopus, INSPEC, DBLP,CSCD, etc.
Sponsored by: ICT, CAS & CCF

JCST Papers

Only for Academic and Non-Commercial Use

Thanks for Reading!

JCST WeChat Subscription Account

JCST URL: https://jcst.ict.ac.cn

SPRINGER URL: https://www.springer.com/journal/11390

E-mail: jcst@ict.ac.cn

Online Submission: https://mc03.manuscriptcentral.com/jcst

Twitter: JCST_Journal

LinkedIn: Journal of Computer Science and Technology

Wang WX, Wu J. Survey on hypergraph algorithms: Foundations, advances, and applications in cloud computing. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY, 41(2): 564-592, Mar. 2026. DOI: 10.1007/s11390-025-5351-x, CSTR: 32374.14.s11390-025-5351-x

Survey on Hypergraph Algorithms: Foundations, Advances, and Applications in Cloud Computing

Wen-Xuan Wang[1] (王文璇), *Member, CCF*, and Jie Wu[1, 2, *] (吴　杰), *Fellow, CCF, IEEE*

[1] *Cloud Computing Research Institute, China Telecom, Beijing 100191, China*

[2] *Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, U.S.A.*

E-mail: wangwx16@chinatelecom.cn; wujie@chinatelecom.cn

Received March 4, 2025; accepted September 11, 2025.

Abstract As a generalization of traditional graph structures, hypergraphs provide a powerful mathematical framework for modeling complex, high-order, and many-to-many relationships among entities. With the rapid advancement of artificial intelligence (AI) and machine learning (ML), research on hypergraph algorithms has gained increasing momentum, spanning both theoretical foundations and practical applications. Motivated by this trend, this survey offers a systematic and comprehensive overview of hypergraph algorithms, along with cloud computing scenarios in which hypergraph modeling can be effectively applied. The paper begins by introducing the fundamentals of hypergraphs and analyzing the structural distinctions between hypergraphs and traditional graphs, highlighting their practical implications for cloud computing tasks. We then review theoretical advances, with particular attention to partitioning, coloring, and isomorphism, followed by a comprehensive overview of hypergraph learning methods, including spectral methods and neural network based techniques. In addition, we analyze the types of scenarios in cloud computing where hypergraph methods can be effectively applied, with emphasis on data center networking, traffic prediction, resource scheduling, data management, anomaly detection, and cloud security. With the advancement of AI-driven learning methods, hypergraph-based models are increasingly capable of capturing high-order dependencies, enabling more timely and accurate decision making in complex cloud environments. Last but not least, we outline the major challenges and opportunities for future research. This paper provides critical insights into the role of hypergraphs in addressing complex computational and organizational challenges across multidisciplinary fields.

Keywords hypergraph, hypergraph learning, cloud computing, artificial intelligence (AI), machine learning (ML)

1 Introduction

Graphs have long served as fundamental tools for representing pairwise relationships among entities in various domains[1], particularly in social network analysis. This ubiquity stems from their intuitive structure, which effectively maps connections between entities, simplifying complex relational data into manageable forms. However, traditional graph models, constrained by their focus on pairwise interactions, often fail to represent the intricate, multi-entity relationships inherent in real-world data. These limitations become particularly apparent in domains such as telecommunication networks, social media interactions, and biological systems[2, 3], where relationships frequently extend beyond simple pairwise connections. In some cloud computing applications, such limitations are especially evident. For example, virtual machine (VM) placement scenarios require concurrent allocation of various resources, including CPU, memory, and storage. Modeling these interdependencies as independent pairwise relations oversimplifies the problem. By contrast, a hypergraph can represent these multiresource dependencies as a single hyperedge, nat-

Survey

*Corresponding Author

urally capturing the many-to-many relationship between resources and VMs. Similarly, in relational databases, a single transaction often involves multiple tables, forming a hyperedge that captures the complex inter-table dependencies. To address such challenges, hypergraphs offer a more expressive data structure for modeling systems exhibiting multi-way relationships among their components[4]. Unlike traditional graphs, hypergraphs natively preserve group-level interactions without artificially decomposing them into pairwise links. This property is particularly advantageous in some cloud computing applications, where scenarios such as data center network topology, resource scheduling, and traffic engineering (TE) inherently involve higher-order dependencies.

The era of big data has driven advancements in artificial intelligence (AI) and machine learning (ML). However, as input data expands, the time complexity of classic algorithms increases, affecting their efficiency. Modeling multi-way relationships as pairwise interactions often leads to network expansion, slowing down algorithm performance. This limitation has motivated researchers to explore more efficient modeling methods that better represent multivariate relationships. The theoretical foundations of hypergraphs have evolved significantly over the past few decades[5]. This evolution has been driven by both theoretical advances and practical demands across diverse domains, including social network analysis, and computer vision[6]. These algorithms extend traditional graph algorithms to the hypergraph setting, addressing unique computational challenges and leveraging the richer structural information provided by hyperedges. For example, this translates into more accurate representations of service dependencies, multi-statement queries, and multi-resource management. Recent integration of ML with hypergraph modeling has further expanded hypergraph's applicability. Hypergraph embeddings reduce high-dimensional data into lower-dimensional latent spaces, thereby improving scalability for classification, prediction, clustering, and anomaly detection tasks. For example, hypergraph neural networks (HGNNs), as a generalization of graph neural networks (GNNs), are particularly effective at learning higher-order dependencies among virtualized resources, applications, and network entities. These advances demonstrate the growing importance of hypergraphs as a modeling paradigm for modern cloud infrastructures. Furthermore, in TE, hypergraphs improve prediction accuracy by capturing the complex relationships among network topology, routing paths, and application traffic.

Existing surveys closely related to this work differ significantly in both their scopes of topics and the ranges of literature they examine. Hu *et al.*[7] explored classifications of hypergraph learning algorithms, contributing to a better understanding of hypergraph-based methods across various domains. Similarly, Gao *et al.*[8] investigated hypergraph generation and learning methods, while Catalyurek *et al.*[9] emphasized advancements in hypergraph partitioning algorithms. Antelmi *et al.*[10] reviewed the hypergraph representation learning problem. This paper reviews the latest advancements in hypergraph algorithms, emphasizing future challenges. It summarizes and synthesizes existing studies that primarily focus on identifying cloud computing scenarios suitable for hypergraph modeling, while also highlighting the opportunities and obstacles in their implementation.

Our contributions can be summarized as follows.

- *Inherent Challenges*. Hypergraphs, as extensions of traditional graphs, inherit several challenges, but their higher-order characteristics introduce additional complexity. This study explores both classical and unique difficulties in hypergraph algorithms.
- *Comprehensive Review*. This work offers a systematic review of hypergraph theory and learning. It highlights key innovations in classical hypergraph algorithms and categorizes hypergraph learning methods into spectral analysis based methods and neural network based methods.
- *Application in Some Cloud Computing Applications*. Hypergraphs excel at modeling high-order relationships, making them suitable for some cloud computing problems. We summarize and synthesize existing studies primarily focusing on identifying cloud computing scenarios suitable for hypergraph modeling.
- *Future Directions*. We propose potential future directions focusing on problem setting, modeling techniques, and scalability.

The organization of this survey is depicted in Fig.1, and the paper is structured as follows. Section 2 introduces the essential concepts and foundations of hypergraphs, discussing the types of hypergraphs and matrix representation, connectivity, and comparisons between hypergraphs and traditional graphs. Section 3 explores the theoretical advancements in hypergraphs, including hypergraph partitioning, coloring, and iso-

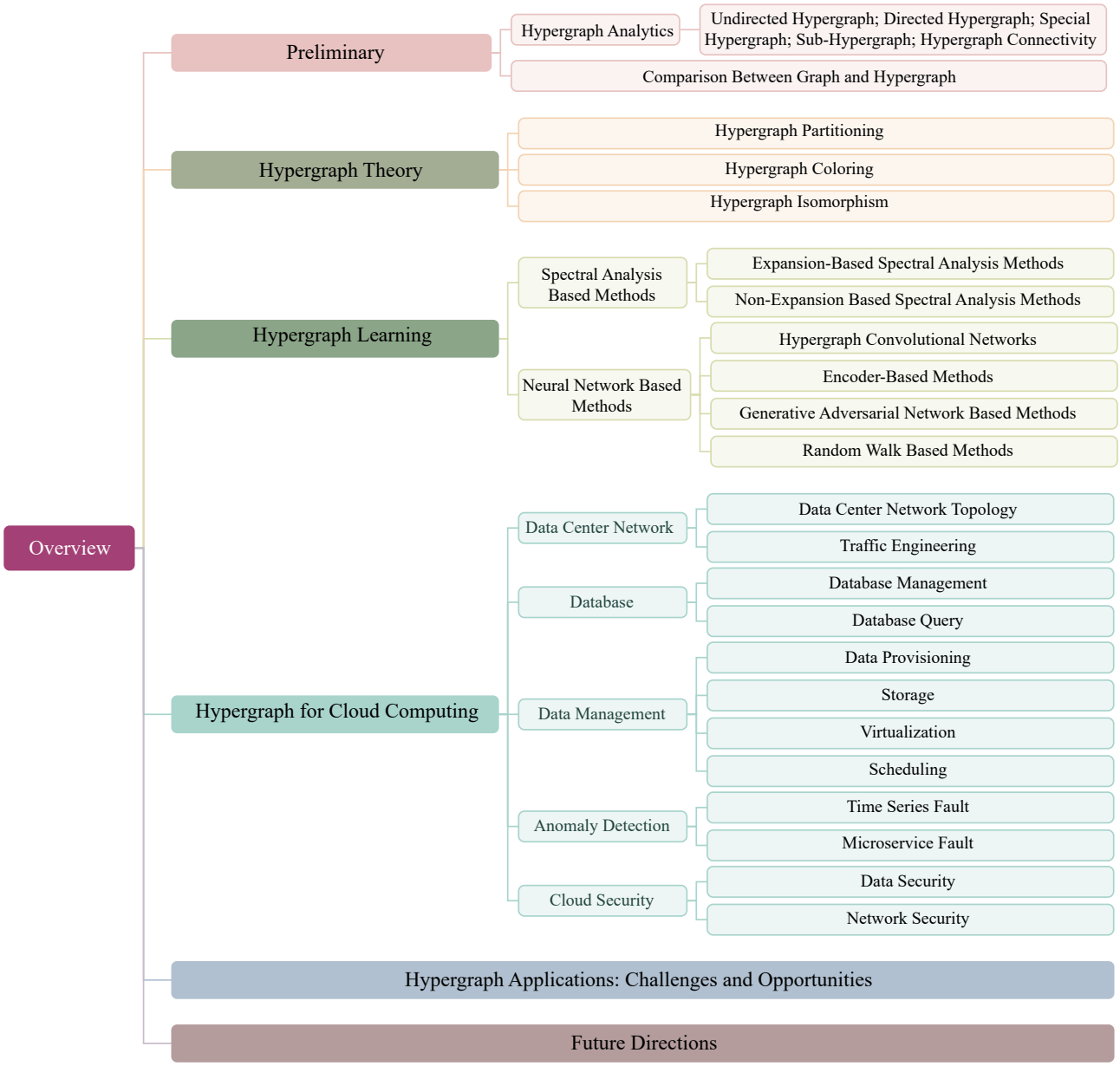


Fig.1. Overview of this survey.

morphism, thereby providing the groundwork for advanced computations. Section 4 focuses on hypergraph learning frameworks, covering two significant paradigms: spectral analysis based methods and neural network based methods. Section 5 examines the application of hypergraphs in cloud computing, covering areas such as data center networks, data management, databases, anomaly detection, and cloud security. Section 6 highlights the challenges and opportunities in hypergraph applications, and Section 7 discusses the future of hypergraphs, outlining key trends, and research directions. Finally, Section 8 concludes this paper.

2 Preliminary

This section presents the foundational concepts of hypergraphs and the relationship between graphs and hypergraphs in detail, focusing on high-order interactions, matrix representation, and hypergraph transformation. Please refer to the appendix[①] for compiling the mathematical symbols and related concepts.

[①]https://jcst.ict.ac.cn/en/supplement/6eb6c0bc-ab17-4afc-899f-2130f8c996cd, Nov. 2025.

2.1 Hypergraph Analytics

2.1.1 Undirected Hypergraph

We first briefly introduce the preliminary definition of an undirected hypergraph[11]. An undirected hypergraph $H = (V, E)$ is formally described by a set of nodes $V = \{v_1, v_2, \ldots, v_{|V|}\}$ and a set of hyperedges $E = \{e_1, e_2, \ldots, e_{|E|}\}$, where each hyperedge e_j is a non-empty subset of V, ensuring $e_j \neq \emptyset$ and $e_j \subseteq V$. The incidence relationship between nodes and hyperedges is represented using the incidence matrix $\boldsymbol{H} = \{h(v_i, e_j)\} \in \{0, 1\}^{|V|\times|E|}$, defined as

$$h(v_i, e_j) = \begin{cases} 1, & \text{if } v_i \in e_j, \\ 0, & \text{if } v_i \notin e_j, \end{cases}$$

where $h(v_i, e_j) = 1$ if node v_i is a member of the hyperedge e_j and $h(v_i, e_j) = 0$ otherwise. Let $\boldsymbol{W}$ denote the diagonal matrix of the hyperedge weights in a hypergraph, i.e., $\text{diag}(\boldsymbol{W}) = (w(e_1), w(e_2), \ldots, w(e_{|E|}))$. Hyperedge weights indicate the relative importance of each hyperedge in a hypergraph, while node weights reflect the varying importance of individual nodes. Let $\boldsymbol{U}$ denote the diagonal matrix of the node weights in a hypergraph, i.e., $\text{diag}(\boldsymbol{U}) = (u(e_1), u(e_2), \ldots, u(e_{|V|}))$.

The degree of a node v_i, denoted as $d(v_i)$, is the sum of the weights of hyperedges incident to v_i. The degree of a hyperedge e_j, denoted as $d(e_j)$, is the number of nodes contained in e_j. Let $\text{diag}(\boldsymbol{D}_\text{v}) = (d(v_1), d(v_2), \ldots, d(v_{|V|}))$ and $\text{diag}(\boldsymbol{D}_\text{e}) = (d(e_1), d(e_2), \ldots, d(e_{|E|}))$ be the diagonal matrix of the node degrees and the hyperedge degrees, respectively.

Let $\boldsymbol{A} = \{a_{i,j}\}$ denote the adjacency matrix of a hypergraph H, where $a_{i,j} = 1$, if there exists a hyperedge $e \in E$ such that $v_i, v_j \in e$ and $i \neq j$, $a_{i,j} = 0$ otherwise. The matrix $\boldsymbol{A}$ is a square matrix of size $|V| \times |V|$. This matrix is symmetric and has only real eigenvalues. This property plays a critical role in numerous graph-related algorithms and analyses, particularly in spectral methods. For a simple graph, the Laplacian matrix is given by $\Delta = \boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{D}$ is the degree matrix (a diagonal matrix containing the degrees of the nodes) and $\boldsymbol{A}$ is the adjacency matrix. For hypergraphs, the Laplacian matrix becomes more complex, with its definition adjusted to account for the multidimensional relationships inherent in hypergraphs[12]. Let Δ denote the Laplacian matrix in a hypergraph, i.e.,

$$\Delta = \boldsymbol{D}_\text{v} - \boldsymbol{H}\,\boldsymbol{W}\,\boldsymbol{D}_\text{e}^{-1}\boldsymbol{H}^\text{T},$$

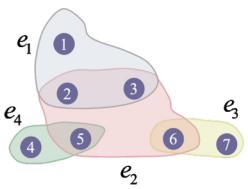
where $\boldsymbol{D}_\text{v}$, $\boldsymbol{D}_\text{e}$, and $\boldsymbol{H}$ are the node degree matrix, hyperedge degree matrix, and incidence matrix, respectively.

The adjacency tensor of hypergraph H is defined as a three-order tensor, where $\mathcal{A} = \{\mathcal{A}_{ijh}\} \in \mathbb{R}^{n\times n\times m}$. The element is:

$$\mathcal{A}_{ijh} = \begin{cases} 1, & \text{if } i \neq j \text{ and } \{v_i, v_j\} \subseteq e_h, \\ 0, & \text{otherwise.} \end{cases}$$

$\mathcal{A}_{ijh}$ indicates whether nodes v_i and v_j are connected within the hyperedge e_h. By using different slices $\mathcal{A}^{(h)}$, the node relationships within the hyperedge e_h can be represented.

In Fig.2, we present an example of an unweighted, undirected hypergraph. This hypergraph consists of four hyperedges, denoted as e_1, e_2, e_3, and e_4, respectively, and includes seven nodes. The degree of the hyperedge e_1 is 3, containing nodes v_1, v_2, and v_3. Similarly, other elements of the hyperedge degree matrix $\boldsymbol{D}_\text{e}$ can be inferred. Node v_3 belongs to hyperedges e_1 and e_2, and its degree is 2. Other elements of the node degree matrix $\boldsymbol{D}_\text{v}$ can be inferred. The incidence matrix $\boldsymbol{H}$ of the hypergraph can be obtained.



$$\boldsymbol{D}_\text{e} = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix},$$

$$\boldsymbol{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad \boldsymbol{D}_\text{v} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Fig.2. Example of an undirected hypergraph.

Special types of hypergraphs can be defined based on degree. A k-regular hypergraph is a hypergraph where the degree of all the nodes is equal to k. A k-uniform hypergraph is a hypergraph where the degree of all hyperedges is equal to k. The number of nodes in a hyperedge is called the cardinality. The rank $r(H)$ of H is the maximum cardinality of a hyperedge in the hypergraph and the minimum cardinality of a hyperedge is the co-rank $cr(H)$ of H. It can also be stated that $r(H) = cr(H) = k$.

2.1.2 Directed Hypergraph

Traditional undirected hypergraph representa-

tions do not fully capture real-world scenarios because hyperedges often exhibit directionality. Therefore, the representation of directed hypergraph structures becomes crucial[13]. In this context, each hyperedge can be further partitioned into two sets: the source node set $S(e_j)$ and the target node set $T(e_j)$, where $T(e_j)$ and $S(e_j)$ are the target and the source nodes for hyperedge e_j, respectively. The incidence relationship between nodes and hyperedges can be represented using the incidence matrix $\boldsymbol{H} = \{h(v_i,\ e_j)\} \in \{-1,\ 0,\ 1\}^{|V|\times|E|}$:

$$h(v_i, e_j) = \begin{cases} -1, & \text{if } v_i \in T(e_j), \\ 1, & \text{if } v_i \in S(e_j), \\ 0, & \text{otherwise.} \end{cases}$$

As depicted in Fig.3, a directed hypergraph example is provided, accompanied by its mathematical representations: the incidence matrix $\boldsymbol{H}$, the source incidence matrix $\boldsymbol{H}_{\rm s}$, and the target incidence matrix $\boldsymbol{H}_{\rm t}$.

The generated sub-hypergraph can be defined as follows. The induced sub-hypergraph of a given hypergraph consists of the node set of the original hypergraph H, and its hyperedges either contain one node or have an intersection with the node set containing at least two elements. A sub-hypergraph of a given hypergraph H is a hypergraph whose node set and hyperedge set are both subsets of the given hypergraph. A partial hypergraph is a hypergraph whose hyperedge set is a subset of the given hypergraph.

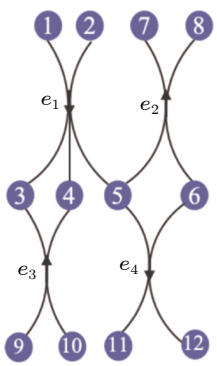
2.2 Comparison Between Graph and Hypergraph

In real-world scenarios, interactions among multiple individuals can be represented by nodes, such as co-authors of a scientific paper. These interactions are defined by their order: a 0th order interaction is a node interacting with itself, a 1st order interaction involves two nodes, and a 2nd order interaction involves three nodes. While traditional graphs can only represent 1st order interactions, hypergraphs can represent any k-order interaction using flexible hyperedges. This ability makes hypergraphs more effective in modeling high-order correlations. In cloud computing, this structural flexibility translates into significant practical benefits. For example, hypergraphs provide a natural representation for datacenter network topologies where multiple servers, switches, and applications interact simultaneously. In TE, hypergraphs can model complex relationships between datacenter network nodes, such as routers and switches, capturing the interactions among multiple paths and flows. This capability allows for more efficient routing and load balancing. Similarly, hypergraphs can model complex resource requirements, such as CPU, memory, and storage. This allows for more efficient allocation strategies that consider the interdependencies of multiple resources. In database management and query optimization, hypergraphs encode multi-table transactions and enable higher-order query planning. Hypergraphs also model complex dependencies among logs or events to enhance anomaly detection, task allocation, and etc.

3 Hypergraph Theory

3.1 Hypergraph Partitioning

Hypergraph partitioning often serves as a key technique for enabling the scalability of various algorithms. Specifically, scalable algorithms across diverse applications frequently require a subroutine to partition a hypergraph into k blocks of approximately equal size, minimizing the number of edges that span across blocks. This process is typically guided by a balance constraint, which ensures that the weights



$$\boldsymbol{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 \\ -1 & 0 & -1 & 0 \\ -1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \quad \boldsymbol{H}_{\rm s} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{H}_{\rm t} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Fig.3. Example of a directed hypergraph.

of all blocks remain below a specified upper bound.

Hypergraph partitioning algorithms are used to achieve load balancing in data center networks, ensuring low response time at service nodes. Similarly, scientific simulations on supercomputers utilize hypergraph partitioning to evenly distribute workloads and minimize interprocessor communication overhead. Furthermore, partitioning (graph) neural networks significantly accelerates the training process[14]. However, hypergraph partitioning is an NP-hard problem[15], and no constant-factor approximation algorithm exists[16]. Consequently, heuristic algorithms are commonly adopted in practical implementations.

In sequential hypergraph partitioning algorithms, the multilevel paradigm is widely considered the most effective[17]. It strikes a balance between execution time and the quality of the partitioning solution. Hypergraph partitioning algorithms use two strategies: node-based and net-based partitioning. Hyperedge partitioning has the advantage of replicating nodes, eliminating the need for auxiliary node modeling required in node-based methods. In hyperedge partitioning, each hyperedge belongs to a single block, while nodes can belong to multiple blocks. In contrast, node-based partitioning assigns each node to a single block, and hyperedges can span multiple blocks. Most existing algorithms are based on net-cut partitioning.

Specifically, Heuer *et al.*[18] proposed a network flow based refinement framework for multilevel hypergraph partitioning, extending the flow-based improvement algorithm KaFFPa to hypergraphs. Gottesbüren *et al.*[19] improved the flow-based refinement framework of KaHyPar-MF[18] by enhancing the HyperFlowCutter algorithm to handle weighted hypergraphs, enabling more efficient multilevel k-way partitioning. Andre *et al.*[20] developed the first multilevel memetic algorithm for hypergraph partitioning, incorporating multilevel recombination and mutation operations to enhance diversity. Schlag *et al.*[21] proposed an open-source hypergraph partitioner based on two objectives: the cut-net metric and the connectivity matrix. This algorithm[21] proposes two preprocessing techniques: locality-sensitive hashing (LSH) and community detection based on the Louvain algorithm. Sybrandt *et al.*[22] proposed a hypergraph partitioning method leveraging graph embeddings to enhance coarsening strategies within the multilevel paradigm. By embedding the initial hypergraph, the method ensures that coarsened instances retain key structural features, prioritizing coarsening in self-similar regions. Theoretical and practical evidence suggests that node-cut partitioning is more effective than net-cut partitioning for power-law graph processing[23].

Distributed memory partitioning algorithms typically lack access to the global view of the entire graph; instead, they rely on local graph data to make partitioning decisions. Maleki *et al.*[24] proposed BiPart, the first deterministic and parallel hypergraph partitioner. Unlike existing parallel partitioners, which lack determinism and are unsuitable for domains like VLSI design, BiPart ensures consistent partitioning results for the same input. Kabiljo *et al.*[3] proposed the social hash partitioner (SHP), a distributed algorithm for balanced k-way hypergraph partitioning. SHP minimizes fanout by optimizing a novel objective function called probabilistic fanout, which simultaneously reduces the communication volume and the cut metric $(k-1)$.

However, challenges remain in optimizing partitioning for dynamic hypergraphs, improving scalability for irregular structures, and integrating real-time processing for large-scale distributed systems.

3.2 Hypergraph Coloring

Since the 20th century, hypergraph coloring has emerged as an important problem in combinatorics and has attracted significant scholarly attention[25]. This task is foundational not only to the field of combinatorics but also to various applications in areas such as optimization and scheduling. Hypergraph coloring, which assigns distinct colors to nodes within hyperedges to avoid conflicts, is effective in tasks involving resource allocation and scheduling. Unlike the graph coloring problem, which allows testing for two-colorability in linear time, determining whether a given hypergraph can be two-colored is NP-hard, even for 3-uniform hypergraphs.

In general graphs, we can test whether a graph is two-colorable in linear time[26]. However, in hypergraphs, testing whether a given hypergraph is two-colored is NP-hard even for a 3-uniform hypergraph.

Recent advancements in hypergraph coloring have significantly expanded both the theoretical understanding and practical applications of this field. Researchers have addressed diverse problems, ranging from LO (linearly ordered) coloring and conflict-free coloring to distributed and directed hypergraph colorings. Kierstead *et al.*[27] determined bounds for the chromatic number of some hypergraphs. Anand *et*

al.[28] addressed the LO coloring problem for 3-uniform hypergraphs, providing an improved SDP (semidefinite programming)-based rounding algorithm that produces an LO coloring with $O(n^{1/5})$ colors. Prasad *et al.*[29] proposed a hypergraph-based method to index coding problems in information theory, which derives new upper bounds for the optimal index coding length by leveraging conflict-free colorings of the hypergraphs.

In distributed algorithms, Manuela *et al.*[30] presented a deterministic distributed algorithm for computing a $(2\Delta-1)$-edge-coloring or list-edge-coloring in any n-node graph with maximum degree Δ, achieving a polylogarithmic-time complexity of $O(\log^8 \Delta \log n)$, resolving a long-standing open question in distributed graph algorithms. Nate *et al.*[31] studied the approximability of clustering edge-colored hypergraphs based on linear programming, and proposed a linear-time approximation algorithm for clustering edge-colored hypergraphs.

For directed hypergraphs, Keszegh[32] investigated the coloring of directed hypergraphs, proving a conjecture that 3-uniform directed hypergraphs, subject to specific intersection conditions, are properly 2-colored.

However, many algorithms rely on simplifying assumptions or specific conditions, such as uniformity or bounded degree, which may limit their applicability to real-world scenarios involving highly irregular and dynamic hypergraphs. The integration of hypergraph coloring into large-scale distributed systems and ML frameworks presents significant computational and scalability challenges.

3.3 Hypergraph Isomorphism

In graph structure analysis, comparing the similarities between graph structures is essential. When dealing with the above applications, one of the most important issues in graph theory is to compare the similarity of substructures in graphs and do so in polynomial time. This is typically achieved through graph isomorphism. The study of graph kernels has garnered significant attention because of their analytical utility. However, the application of hypergraph isomorphism for comparative purposes is hindered by the following two major challenges.

1) No efficient algorithm exists to determine if two hypergraphs are isomorphic.

2) Isomorphism is a rigid comparison methods, limited to exact comparisons of hypergraph similarities.

With the emergence of hypergraph structures, researchers have developed diverse methodologies to tackle the challenge of measuring isomorphism in high-order structures. Wachman *et al.*[33] explored isomorphism algorithms for hypergraphs, identifying challenges related to the unreliability of random walk-based algorithms. Building on this, Laszl'o *et al.*[34] examined the computational aspects of hypergraph isomorphism and proposed algorithms that efficiently handle low-rank hypergraphs within a feasible computational timeframe. Arvind *et al.*[35] introduced a fixed-parameter tractable algorithm for colored hypergraph isomorphism. Feng *et al.*[36] proposed a generalized Weisfeiler-Lehman (WL) test algorithm for hypergraph isomorphism, extending its application from graphs to hypergraphs. They introduced two hypergraph kernel frameworks: the hypergraph WL subtree kernel and the hypergraph WL hyperedge kernel.

While significant advancements have been made in hypergraph isomorphism testing, challenges remain in achieving computational efficiency and scalability, particularly for large, irregular, or high-rank hypergraphs. Existing methods either suffer from computational bottlenecks, such as increased dataset size, or rely on assumptions like low-rank structures or specific hypergraph properties.

4 Hypergraph Learning

There are many types of hypergraph learning methods. Hypergraph generations techniques support efficient anomaly detection, resource usage prediction, and service recommendation by capturing complex correlations across heterogeneous cloud resources or physical topology (please see Appendix[②]). Spectral analysis based methods and HGNNs extend to higher-order structures, enabling more accurate modeling of multi-resource (such as jobs, data tables) dependencies, workflow scheduling, network traffic optimization, and etc. From a methodological perspective, spectral analysis methods, long regarded as the traditional cornerstone of graph learning, have also dominated hypergraph learning. Most studies were grounded in spectral theory before the advent of neural net-

[②]https://jcst.ict.ac.cn/en/supplement/6eb6c0bc-ab17-4afc-899f-2130f8c996cd, Nov. 2025.

works in this domain. These methods, which are characterized by rigorous mathematical formulations, provide analytical precision but are often constrained by inherent limitations.

At the same time, the emergence of neural networks has injected new vitality into the research on hypergraph learning. How to use neural network structures for hypergraph learning has become a new research hotspot. In recent years, many excellent methods in this area have been proposed.

The spectral theory of hypergraphs serves as a foundational theoretical framework that significantly enhances the representation, understanding, and development of spectral-based hypergraph learning methods. The spectral properties of hypergraphs provide profound insights into their structural characteristics, connectivity, and clustering capabilities, enabling more effective analysis and learning. Readers seeking a deeper understanding of the spectral theory and properties of hypergraphs, including matrix/tensor representations and sparsification methods, are referred to the Appendix[③]), which offers a comprehensive discussion of theoretical foundations, recent progress, and open challenges. This supplementary material comprehensively elaborates on the theoretical foundations, recent advancements, and critical open challenges in the spectral analysis of hypergraphs.

Therefore, in this section, we provide a comprehensive overview of hypergraph learning methods, categorizing them into spectral analysis based methods and neural network based methods.

4.1 Spectral Analysis Based Methods

4.1.1 *Expansion-Based Spectral Analysis Methods*

Spectral analysis methods are the traditional methods in hypergraph learning, based on matrix analysis and spectral theory. These methods involve optimizing an objective function, with solutions derived through rigorous mathematical reasoning. The star expansion (SE) and clique expansion (CE) methods are two classical expansion techniques for hypergraphs[37]. The foundational concepts of standard star and clique expansions have been proposed within the framework of spectral theory. Fig.4 illustrates an example of converting a hypergraph into a graph using the star and clique expansion methods.

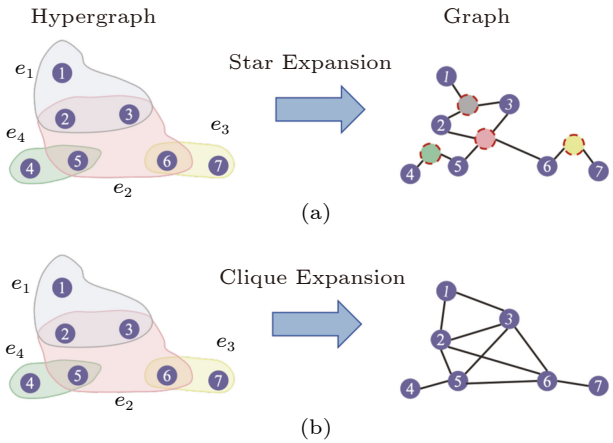


Fig.4. Example of converting a hypergraph into a graph using the (a) star and (b) clique expansion methods.

In the SE algorithm, a new graph, denoted as $G^*(V^*, E^*)$, is constructed from the original hypergraph $H(V, E)$. In this expanded representation, a new node is introduced for each hyperedge in the original hypergraph, resulting in the set of nodes V^* being the union of the original nodes V and the hyperedges E. Each newly introduced node is then connected to all the nodes within its corresponding hyperedge, thereby forming a star-like structure. The SE algorithm redistributes the weight of each hyperedge to the corresponding pairwise graph edges. The normalized Laplacian matrix of the new pairwise graph G^* is computed. This matrix encapsulates the structural relationships within the expanded graph and is derived from a formula involving both the adjacency matrix and the degree matrix of the expanded graph.

The CE algorithm constructs a new graph from the original hypergraph by replacing each hyperedge with a fully connected structure, where all nodes within the hyperedge are pairwise connected[38]. In the expanded graph, the edges represent the connections between the nodes of each hyperedge in the original graph. The weight of each edge in the expanded graph is determined by minimizing the difference between the weight of the corresponding hyperedge in the original graph and that in the expanded structure. After obtaining the expanded graph, the normalized Laplacian matrix is computed[39], which involves the degree matrix of the expanded graph, the incidence matrix of the original hypergraph, and the weight matrix of the expanded graph. The resulting Laplacian matrix is then used to solve a traditional graph parti-

[③]https://jcst.ict.ac.cn/en/supplement/6eb6c0bc-ab17-4afc-899f-2130f8c996cd, Nov. 2025.

tioning problem, which is formulated as a least squares objective function.

The star and clique expansion methods, while effective, have notable drawbacks, particularly the significant increase in edge count in the expanded graph, which limits their applicability to higher-order hypergraphs. To address this challenge, several sampling-based expansion methods have been introduced. In the context of the Markov clustering algorithm, Louis[40] proposed the Breaking Ties Randomly (BTR) method, which replaces each hyperedge with an edge connecting the two most distant nodes within the hyperedge, assigning the new edge the same weight as the original hyperedge. However, BTR primarily considers the relationship between the two most distant nodes, neglecting the contributions of other nodes within the hyperedge, which leads to a loss of important intermediate information, particularly in hypergraphs with significant node feature variability. In contrast, Chan *et al.*[41] proposed an expansion method based on diffusion theory, addressing a limitation where information flow is restricted to high- and low-density nodes, ignoring intermediate nodes.

The expanding hypergraph learning method transforms hypergraphs into traditional pairwise graph networks by decomposing their structures. This decomposition enables the use of standard graph-based learning algorithms. However, this transformation often leads to information loss, as the inherent higher-order relationships and structural intricacies of the original hypergraphs are not fully preserved.

4.1.2 *Non-Expansion Based Spectral Analysis Methods*

Unlike expansion-based methods, non-expansion based spectral analysis methods model a hypergraph directly, constructing the Laplacian matrix of the hypergraph itself. This modeling method ensures the integrity of the hypergraph's information. Carletti *et al.*[42], in their exploration of Random Walks on Hypergraphs (RWH), demonstrated that the random walk method they proposed, which directly models the hypergraph, yields results that significantly differ from traditional random walk algorithms applied to results' projected graph. Bolla's Laplacian method[43] represents one of the earlier methods to non-unfolded matrix factorization. The primary objective of this method is to partition the nodes of a uniform hypergraph into k clusters, minimizing the number of edges in the corresponding cut set. Bolla demonstrated that this matrix decomposition technique effectively solved the hypergraph minimum cut problem. Zhou *et al.*[44] extended traditional matrix factorization methods from undirected graphs to hypergraphs, introducing the N-cut algorithm for hypergraph partitioning. Huang *et al.*[45] addressed the image segmentation problem by applying their N-cut algorithm[44] to a hypergraph, where the nodes correspond to over-segmented image patches, ultimately resolving the issue of image over-segmentation. Similarly, Purkait *et al.*[46] employed the N-cut algorithm to partition clusters on large-scale hyperedges after constructing a clustered hypergraph. Huang *et al.*[6] combined the propagation inference theory proposed by Zhou *et al.*[44] with probabilistic matrices on hypergraphs, introducing a probability-based non-expansion spectral hypergraph ranking matrix factorization algorithm. Ma *et al.*[47] proposed a nonlinear extension of their standard Laplacian operator[44], known as the hypergraph p-Laplacian regularization (HpLapR), to preserve the geometric probability distribution of data.

Non-expanded hypergraph learning focuses on indecomposable hypergraphs, modeling their structure directly without decomposition. In these hypergraphs, nodes within a hyperedge are strongly correlated, but subsets of nodes often lack such relationships. For instance, in recommendation systems, hyperedges connecting users, movies, and tags do not show strong correlations between users and tags. Applying expanded hypergraph methods to such structures can cause information loss or introduce noise, reducing model accuracy and efficiency.

Please refer to the supplemental material[④] for a discussion of the summary of spectral analysis hypergraph learning methods in this survey.

4.2 Neural Network Based Methods

Spectral analysis methods, though celebrated for their mathematical rigor, often lack the flexibility and scalability required for complex applications, which has prompted the rise of neural network algorithms as adaptive and computationally efficient alternatives. Leveraging the transformative advancements of deep learning (DL) in domains such as speech recognition, natural language processing[48], and computer vision,

[④]https://jcst.ict.ac.cn/en/supplement/6eb6c0bc-ab17-4afc-899f-2130f8c996cd, Nov. 2025.

researchers have developed specialized neural architectures for graph and hypergraph data. HGNNs, as an extension of GNNs, excel in capturing high-order, non-linear relationships while eliminating the need for manual feature engineering.

4.2.1 *Hypergraph Convolutional Networks*

Hypergraph convolutional networks (HCNs) extend the principles of graph convolutional networks (GCNs) to hypergraphs, enabling the direct modeling of complex multi-way relationships that go beyond pairwise connections. Spectral graph convolutions leverage the Fourier transform to shift graph signals to the spectral domain to perform convolution operations, while spatial graph convolutions aggregate node features directly in the spatial domain[49]. Generally, hypergraph convolutional neural networks can be divided into two categories: spectral-based methods and spatial-based methods.

For the spectral-based methods, Feng *et al.*[50] introduced HGNN, grounded in the principles of hypergraph Laplacians and Chebyshev polynomial approximations, to effectively capture high-order, non-pairwise relationships in complex data. Unlike conventional GNNs, which rely on pairwise node interactions, HGNN employs a vertex-hyperedge-vertex propagation mechanism for iterative representation learning. To enhance computational efficiency, the hypergraph Laplacian is approximated and seamlessly integrated into the deep learning framework. Fig.5 illustrates the details of the hypergraph neural networks.

Yi *et al.*[51] further advanced this line of research by integrating HGNN with recurrent neural networks (RNNs), resulting in the HGC-RNN framework. In this method, graph convolutional operations are first applied to temporal slices of a sequential hypergraph to extract structural features, which are then processed by an RNN for final time series prediction. Compared with existing GNN-based models for structured temporal data, HGC-RNN requires fewer parameters and exhibits enhanced robustness. However, its applicability is limited to unweighted temporal hypergraphs. Yadati *et al.*[52] proposed HyperGCN to reduce noise introduced by HGNNs during information aggregation in semi-supervised learning. By selectively sampling binary edges from expanded hypergraphs, HyperGCN improves robustness. However, it may miss useful information and require more iterations to converge. Bai *et al.*[53] proposed hypergraph attention (Hyper-Atten), a method that integrates graph attention mechanisms with hypergraph convolution structures. By recalculating the incidence matrix $\boldsymbol{H}$ using attention scores and feeding it into a convolutional neural network, the model becomes trainable during backpropagation. Compared with conventional hypergraph convolutional networks, this method offers greater flexibility and adaptability. Jiang *et al.*[54] proposed the DHGNN model by integrating dynamic hypergraph generation (DHG) with hypergraph convolution (HGC), enabling the reconstruction of hypergraph structures to capture latent node relationships in the original data. However, because the model requires clustering-based hypergraph reconstruction at each iteration, it suffers from low computational efficiency, limiting its scalability to large scale hypergraph networks.

Spatial-based methods directly aggregate features

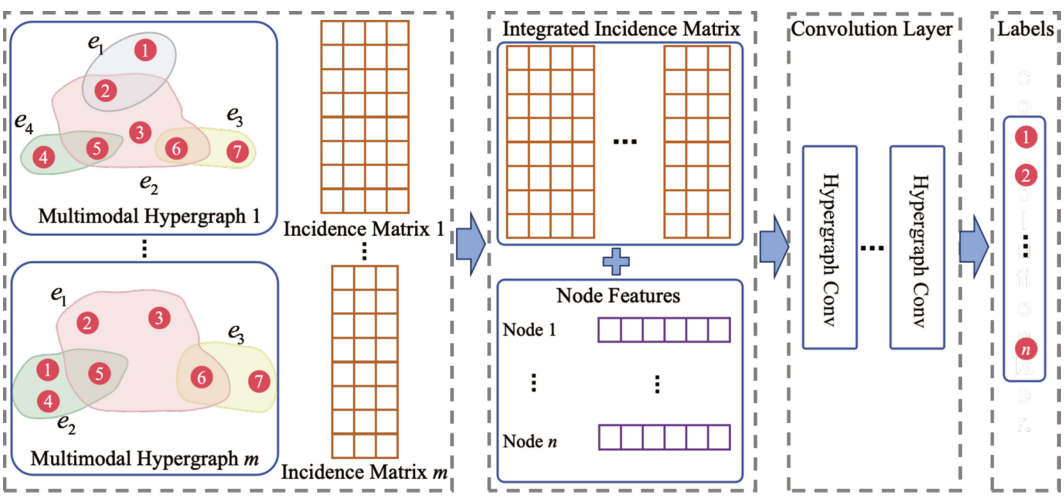


Fig.5. Framework of the hypergraph neural network model[50].

in the local neighborhood of nodes. For the spatial-based methods, Atwood *et al.*[55] proposed diffusion-convolutional neural networks (DCNNs), which make use of transition matrices to determine where nodes are located. The generalization of convolution in the spatial domain is achieved using Gaussian mixture models based on local path operators. Feng *et al.*[50] proposed dynamic hypergraph neural networks, which model evolving hypergraph structures through two key modules: dynamic hypergraph construction and hypergraph convolution. The hypergraph is updated at each layer based on feature embeddings, using a combination of k-NN and k-means clustering to capture both local and global relationships. The convolution module captures high order correlations through two stages: node convolution, which aggregates node features into hyperedges, and hyperedge convolution, which aggregates hyperedge features back to nodes. Gao *et al.*[56] extended the original conference version HGNN, and proposed general hypergraph neural networks (HGNN+). This framework consists of two main components: hypergraph modeling and hypergraph convolution. In the modeling stage, multi-modal or multi-type data is structured into hypergraphs by generating hyperedge groups via pairwise, k-hop, or feature-space neighbors. The convolution stage employs a spatial hypergraph convolution with a two-step message passing mechanism that aggregates features from vertices and hyperedges using learnable functions. In the context of heterogeneous cloud data, HGNN+ holds strong potential for multi-modal fusion across diverse data types, such as log text, traffic time series, and system metrics. However, real-world cloud data is often incomplete or imbalanced across modalities. For example, log data may be sparse or missing altogether, while metric series are dense but noisy. Addressing such challenges requires additional mechanisms beyond standard convolution. One promising research direction is the incorporation of modality-aware weighting strategies during hyperedge construction, which assign adaptive importance to different input features based on confidence or quality. Another is the use of hypergraph-based imputation, where missing associations can be inferred via transitive paths or high-order connectivity, allowing robust learning even under partial observation. While HGNN+ lays the foundation for such tasks, detailed algorithmic designs and experimental validation remain open challenges that merit future research.

Please refer to Table A3 in the supplementary material[⑤] for the methods for hypergraph convolutional models in this survey. We have included a summary table that categorizes representative studies according to their convolution types, key techniques, strengths, and limitations.

4.2.2 *Encoder-Based Methods*

In the realm of ML, encoder-decoder architectures are widely recognized for their efficacy in learning low-dimensional embeddings, capturing intricate patterns within data. However, their applicability is challenged when dealing with hypergraph-like structured data, which encapsulates complex structural information through hyperedges connecting multiple nodes. Consequently, these architectures are often employed as submodules within broader network frameworks to address the unique demands of hypergraph data.

The encoder-based method in Hyper-SAGNN[57] is a sophisticated method for generating node features in hypergraphs, leveraging the structural information encoded in the adjacency matrix. DHNE[58] addresses the challenge of embedding hyper-networks with indecomposable hyperedges, prevalent in heterogeneous networks. HeteHG-VAE[59] was proposed for link prediction in heterogeneous information networks (HINs). It maps HINs into heterogeneous hypergraphs to effectively capture both high-order semantics among multiple entities and low-order pairwise relationships, thereby preserving the rich relational structure of the original network. Payne[60] leveraged both contextual and permutation-invariant node membership properties of hyperedges to jointly model hyperedge and node embeddings, enabling classification and regression tasks in both transductive and inductive learning settings. Notably, encoder-based methods, while powerful, may struggle with the computational demands of large-scale hypergraphs. Therefore, optimizations and scalable solutions are imperative for practical applications.

4.2.3 *Generative Adversarial Network Based Methods*

Generative adversarial networks (GANs) have emerged as a powerful framework for generative modeling, characterized by their ability to learn complex data distributions through adversarial training[61]. The

⑤https://jcst.ict.ac.cn/en/supplement/6eb6c0bc-ab17-4afc-899f-2130f8c996cd, Nov. 2025.

adversarial nature of GANs, which involves a generator network competing against a discriminator network, has proven effective in capturing intricate patterns and generating high-quality synthetic data[62]. HGGAN[5] integrates resting-state fMRI (rs-fMRI) and diffusion tensor imaging (DTI) by employing an interactive hyperedge neurons (IHEN) module, which enhances feature learning by capturing the high-order relationships within hypergraph structures. The MRL-AHF framework leverages the complementarities between modalities and the interactions within modalities to enhance representation learning capability and multimodal fusion performance.

Several challenges arise in the context of GAN-based hypergraph learning, including mode collapse[63], scalability, and the lack of standardized evaluation metrics. Mode collapse occurs when the generator fails to capture the diversity of the real data distribution, resulting in synthetic hypergraphs that are overly similar to each other. Techniques such as feature matching[64], minibatch discrimination[65], and Wasserstein GANs[66] have been proposed to mitigate this issue. Scalability to large hypergraphs is another critical challenge, as the computational complexity of hypergraph operations can be prohibitive.

4.2.4 *Random Walk Based Methods*

Random walks (RWs) serve as a versatile alternative to convolutional methods for encoding the concept of closeness in hypergraphs. In hypergraphs, random walks can traverse hyperedges that connect multiple nodes, offering a unique perspective on node similarity and proximity. Huang *et al.*[67] presented two models, Hyper2vec and NHNE, for embedding hypergraphs, focusing on their non-uniform and dual properties. Hyper2vec employs a biased second-order random walk strategy within the skip-gram framework to capture the structural context of nodes in hyper-networks. NHNE uses 1D (one-dimensional) convolutional neural networks to combine hyperedge features and trains a tuple-wise similarity function to address nonuniform relationships in hypergraphs. Huang *et al.*[68] introduced a novel method to hypergraph representation learning, addressing the limitations of traditional methods that simplify hyperedges into pairwise interactions. The DHHE framework[69] integrates visual content and social context through a heterogeneous hypergraph, embedding them into a low-dimensional space.

5 Hypergraph for Cloud Computing

Over the past decade, cloud computing has witnessed rapid advancements and achieved remarkable success. Cloud computing data centers serve as essential hubs for hosting a wide range of critical business services that are integral to modern life. In these data centers, hardware resources are systematically organized into resource pools comprising physical servers, storage systems, and network devices[70]. Through cloud orchestration systems, these resources are virtualized and efficiently managed to deliver infrastructure services to users. Cloud providers typically offer a variety of infrastructure service types for users to select based on their needs[71]. Numerous commercial cloud service providers have emerged, offering diverse public cloud services. An increasing number of enterprises and organizations have started either to build their cloud computing infrastructures or to adopt hybrid cloud solutions. Therefore, efficient resource management not only enhances service quality but also minimizes resource consumption[72].

Hypergraphs, extensions of traditional graphs capable of modeling multiway relationships, have emerged as promising computational frameworks for solving complex problems in modern data-intensive domains. Researchers can model dependencies among virtual machines, data blocks, or user requests with greater precision using hypergraphs, leading to improved system performance, energy efficiency, and cost effectiveness.

In this section, our discussion is structured according to the categories of problems in cloud computing applications that are suited to hypergraph-based modeling. As depicted in Fig.6, our analysis spans five key areas: data center operations, data management strategies, database optimization, anomaly detection mechanisms, and measures to enhance cloud security.

5.1 Data Center Network

5.1.1 *Data Center Network Topology*

Effective data center network topologies are defined by their ability to achieve high scalability, efficient switch utilization, and exceptional reliability, all while meeting the demands of modern cloud and enterprise environments. Traditional Fat-Tree[73] architectures, while widely used, face limitations scalability, and flexibility. To address these challenges, researchers have explored novel designs leveraging hy-

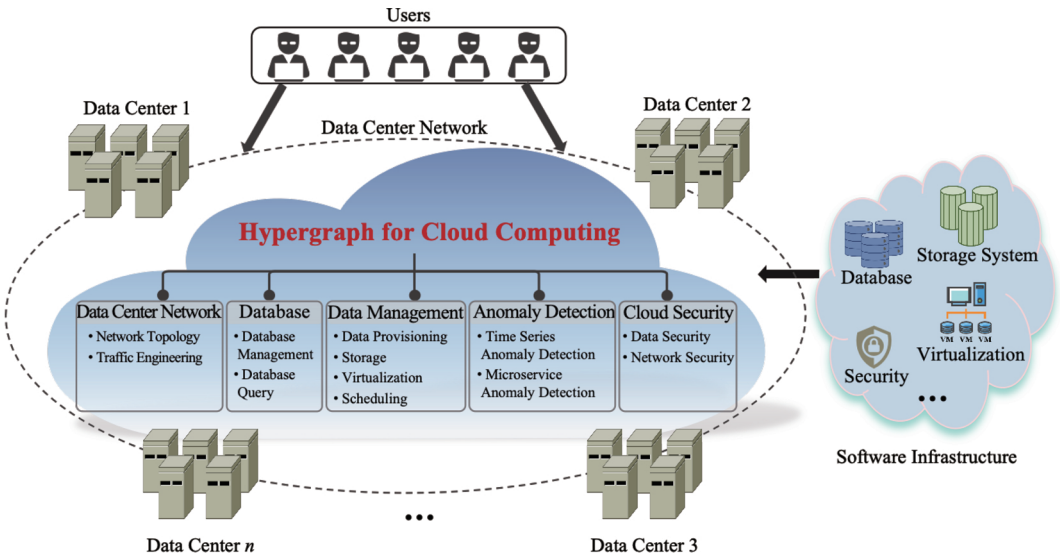


Fig.6. Types of scenarios and problems in cloud computing where hypergraph methods can be effectively applied.

pergraph and combinatorial block design theory.

As reported by the work of Zhang *et al.*[74], the hypergraph-based data center network structures exhibit substantial improvements over traditional Fat-Tree architectures. Specifically, their experimental evaluations demonstrate that hypergraph-based designs, such as the indirect hypernetworks ($H_{\text{indir}}(M1)$, $H_{\text{indir}}(M2,1)$, and $H_{\text{indir}}(M3)$), achieve significantly higher scalability. For example, $H_{\text{indir}}(M1)$ with switch size 64 connects approximately 47 times more nodes compared with a 3-level Fat-Tree using switches of the same size, while requiring about 12 times more switches. Similarly, for switch size 128, $H_{\text{indir}}(M1)$ connects 50 times more nodes than the Fat-Tree while requiring approximately 11 times more switches, indicating a clear advantage in scalability and network density. These findings highlight the inherent flexibility of hypergraphs in modeling multi-way relationships and optimizing inter-cluster communication, which are critical for modern cloud computing infrastructures.

However, challenges remain. One major issue is the practical deployment of hypergraph-based designs in real-world data centers, where integration with existing hardware and protocols may be complex. Additionally, as data center networks scale further, ensuring low-latency communication and fault tolerance across highly interconnected nodes will require more sophisticated optimization techniques.

5.1.2 *Traffic Engineering*

TE is a critical mechanism for optimizing network performance, particularly in dynamic and complex modern networking environments. Traditional TE methods often rely on static traffic allocation, mapping flows to predefined paths without considering real-time network utilization or traffic load. While heuristic algorithms based on precise mathematical models offer improvements, their adaptability to rapidly changing network conditions is limited. To address the limitations of static traffic allocation and adapting to the dynamic and complex of modern networks, Cai *et al.*[75] proposed a multi-agent reinforcement learning (MARL) framework with hypergraph attention mechanisms (MA3C-routing) for TE. In the NSFNET topology, at a demand ratio of 0.7, the proposed method reduces flow completion time (FCT) to 5.6, outperforming shortest path (7.45) and load balancing (557.21). This indicates approximately a 24.8% FCT reduction compared with the next-best method (shortest path). In terms of throughput, the proposed method maintains a high level of 145.84, surpassing all the other baseline methods. Specifically, it achieves 1.4% more throughput than MADDPG (143.83), and far exceeds load balancing (114.83).

Accurate traffic prediction is essential for effective TE. To solve the network traffic prediction problem, Tong *et al.*[76] addressed the problem by introducing a spatio-temporal link hypergraph convolutional network model. They captured the intricate relationships between network topology, routing paths, and application-specific traffic flows. Multicast is essential in data center networks to meet the demand for distributed data-parallel applications. The reconfigurable circuit technology enables faster multicast

by connecting switches, offering a promising solution. To investigate the optimization of multicast performance through splittable multicasting in reconfigurable data center networks, Luo *et al.*[77] transformed the problem into a hypergraph matching problem, focusing on dynamically configuring circuit switches to minimize multicast flow times. To explore efficient acceleration of non-preemptive multicast flows in reconfigurable datacenter networks, Zhang *et al.*[78] proposed a two-round matching algorithm based on a connection-based hypergraph model to address the conflicts between different multicast flows under the bandwidth constraint.

5.2 Database

5.2.1 *Database Management*

Databases are distributed across multiple physical machines, known as hosts. Distributed databases achieve scalability primarily through horizontal partitioning of tables into shards, which are then assigned to hosts within the cluster[79]. Compared with non-distributed databases, distributed systems offer greater scalability, reliability, and availability. However, they require more sophisticated mechanisms to ensure data integrity and incur communication overhead during query processing, as inter-host communication is required for queries involving data from multiple hosts. There are online analytical processing (OLAP) and online transaction processing (OLTP) workloads in distributed databases. OLAP workloads involve long-running analytical queries, where distributed queries are beneficial for load distribution of the load[8]. Furthermore, it should aim to balance the load to avoid overloaded or idling hosts. OLTP workloads consist of short, fast-running queries. Distributed queries in OLTP increase communication overhead and duplicate processing across hosts, thereby reducing system performance. Therefore, we can improve the system performance by reducing the overhead introduced by distributed queries. In graph partitioning for table partitioning in distributed databases, graphs lack the ability to accurately model executed transactions. For example, it is not possible to distinguish if two edges belong to the same transaction or whether they represent two separate edges; whereas, in the context of a hypergraph, a hyperedge clearly represents a single transaction[80]. Quamar *et al.*[81] introduced SWORD, a scalable and workload-aware data partitioning and placement strategy tailored for OLTP workloads. SWORD optimizes partitioning by minimizing the cut metric, effectively reducing internode communication costs, and improving data locality. The algorithm leverages a hypergraph representation of the workload, denoted as $H = (V, E, c, \omega)$, where V represents the virtual nodes (data tuples), E represents transactions spanning multiple nodes, c indicates node capacities, and ω reflects transaction frequencies. The core objective function minimizes the total weight of cut hyperedges, i.e., the sum of transaction frequencies crossing partition boundaries, to minimize internode communication and distributed transaction overhead. In this model, the weight of each node corresponds to its access frequency, while the weight of each hyperedge represents the transaction frequency. SWORD introduces an incremental repartitioning mechanism to dynamically adjust data placement as workloads shift, thereby preventing the need for complete data migrations during runtime. Additionally, SWORD employs a hypergraph compression technique to further enhance partitioning efficiency, which optimizes the partitioning process by reducing associated overheads. This method distinguishes between tuple-based and compressed-node based workload models, allowing for more efficient resource utilization and minimizing latency during query execution. Overall, the innovative use of hypergraph modeling and incremental adjustments allows SWORD to achieve high scalability and low communication overhead, making it an effective solution for distributed OLTP environments.

Yang *et al.*[82] proposed HEPart, which uses hyperedge partitioning for the context of NoSQL database systems. In HEPart, nodes correspond to database tuples, and their weights reflect the associated loads. The gain $g_{pq}(e)$ of a hyperedge e denotes the reduction in the cut when e is moved from block p to block q. Key results demonstrate that HEPart achieves significant improvements in operation span degree (OSD), storage variance (SV), and average operation response time (AORT) compared with round-robin partitioning. Fig.7 presents a comparison of the partitioning strategies employed by Schism, Clay, SWORD, and HEPart. While Schism and Clay rely on graph-based models for workload representation, SWORD and HEPart adopt hypergraph-based modeling methods[80]. In the example, optimal partitioning aims to minimize the number of distributed queries. For example, nodes p_1 and p_2 should ideally be placed within the same block to avoid cross-partition com-

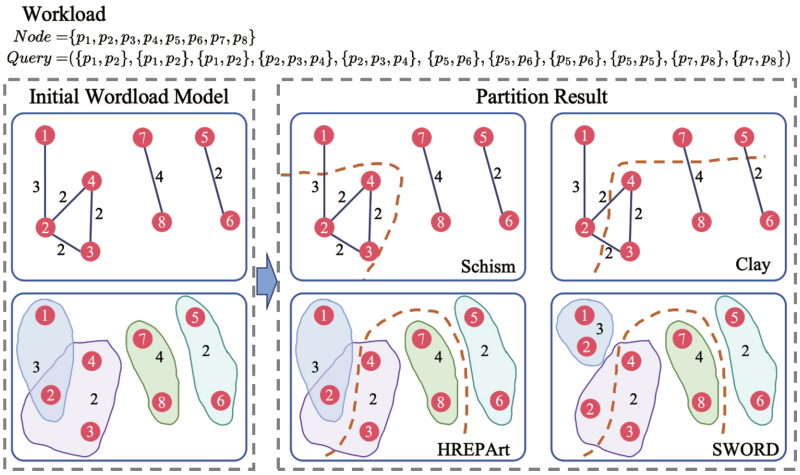


Fig.7. Comparison of the partition of Schism, Clay, SWORD, and HEPArt[80].

munication. However, Schism assigns p_1 and p_2 to separate blocks, resulting in 50% more distributed queries compared with SWORD. This discrepancy arises because the graph model finds its minimal cut at 3 by severing the edge between p_1 and p_2, whereas the hypergraph model achieves a minimal cut of 2 by partitioning the hyperedge containing p_2, p_3, and p_4.

In addition, Chen *et al.*[83] proposed a fragment reallocation strategy for NoSQL database systems based on weighted hypergraph partitioning. The hypergraph represents fragment access patterns in operations, and the partitioning algorithm clusters fragments to optimize communication costs while maintaining a high degree of parallelism.

5.2.2 *Database Query*

In graph database querying, use cases are divided into real-time queries and offline analysis. Real-time queries primarily use graph traversal and pathfinding algorithms (e.g., shortest-path, minimum-spanning tree), and may include subgraph mining, subgraph matching, community search, maximum k-edge connectivity, and keyword-based searches. Most graph databases incorporate complex network analysis for enhanced insights. Offline analysis typically involves pathfinding, centrality analysis, link prediction, and community detection. As an extension of traditional graphs, hypergraphs require similar query capabilities in hypergraph database systems. Zhang *et al.*[84] proposed HyperISO to efficiently search subgraph matching at low costs in hypergraphs. HyperISO consists of three key components: 1) a filtering technique that leverages hyperedge properties and connections to eliminate unpromising candidates, reducing matching costs; 2) an optimized ordering strategy that considers hyperedge candidate sizes and unmatched nodes to enhance the matching process; and 3) a dual enumeration algorithm for listing node and hyperedge mappings. Alam *et al.*[85] proposed a flexible framework for frequent pattern mining in hypergraph databases, addressing the limitations of existing menthods that inadequately decompose associations among data entities. Their developed algorithm, Frequent HyperGraph Miner (FHGM), employs a depth-first search strategy and a canonical labeling technique to efficiently identify isomorphic subhypergraphs and prune the search space. For the empirical evaluation on the "Data Mining" dataset with $\delta = 0.6\%$, FHGM with pruning reduced the number of candidates from 34 968 to 17 243 and substantially decreased runtime. Besides, FHGM yielded a classification accuracy of $51.55\%\pm1.57\%$, a significant improvement over the $33.15\%\pm1.80\%$ accuracy achieved using patterns from a traditional framework. This empirically substantiates the proposed framework's enhanced capability to discover more meaningful patterns.

Preti *et al.*[86] proposed HYPED, a landmark-based oracle framework for estimating s-distance in hypergraphs, where s defines the overlap required for two hyperedges to be adjacent. Unlike traditional methods relying on the line graph, HYPED operates directly on the hypergraph to answer s-distance queries. Wang *et al.*[87] proposed the (k, α, β)-truss model for computing cohesive subgraphs in hypergraphs. Kim *et al.*[88] proposed the (k, g)-core model to identify cohesive subgraphs in hypergraphs by incorporating both neighborhood connectivity and cooccurrence frequency.

5.3 Data Management

5.3.1 Data Provisioning

Hypergraph-based data placement schemes have significantly advanced the efficient distribution of data items in geographically distributed storage systems. These methods optimize key objectives such as interaction locality, communication cost, and load balancing, offering innovative solutions for data-intensive applications. A growing challenge lies in effectively provisioning resources to meet the requirements of quality of service (QoS). Data-intensive applications must tackle the challenge of efficiently placing data items across geodistributed storage nodes. Inefficiencies become pronounced when multiple data items need to be accessed as part of a single transaction, particularly when request origins are dispersed across different locations. To address the challenge, Zhou *et al.*[2] proposed a hypergraph-based data placement scheme for online social networks, aiming to reduce intra-data center communication while maintaining load balance. The scheme utilizes round-robin hypergraph partitioning to maximize the preservation of interaction locality and distance locality. However, it has a limited scope of applications. Yu *et al.*[89] proposed a general hypergraph-based data placement framework, which integrates performance metrics related to the colocation of associated data and the specific location of requested data items in geodistributed environments. Yu *et al.*[89] employed hypergraph partitioning to effectively distribute data items across storage nodes by converting complex optimization objectives encompassing data colocation, access latency, inter-datacenter traffic, and storage costs into hypergraph models. The framework's efficacy was evaluated using trace-based workloads from a location-based online social network (comprising 196 591 data items and 60 948 request patterns), reflecting Amazon Web Services (AWS) regions and their characteristics. By comparing the results reported in this paper with the best performance among the other three methods, we identify a cost-reduction improvement ranging from 0.11 to 0.38. Atrey *et al.*[90] proposed a scalable method that utilizes spectral clustering on hypergraphs to partition data items for distributing data-intensive services across geographically distributed cloud data centers. To improve scalability, the proposed algorithm leverages randomized techniques for low-rank approximations of the hypergraph Laplacian matrix.

5.3.2 Storage

Distributed storage systems need to balance data availability and energy efficiency. Yang *et al.*[91] proposed an energy-efficient storage strategy for cloud data centers using a variable k-coverage hypergraph model. This method considers varying data availability requirements across applications. The algorithm determines the minimum set of active data nodes through hypergraph coverage, enabling unused nodes to be powered down for energy savings. Cheng *et al.*[92] introduced the HyperPart framework, which employs a hypergraph-based abstraction for deduplicated storage systems, demonstrating substantial improvements in both storage space optimization and retrieval throughput. HyperPart models file-block affiliations and multivariate data correlations as a weighted hypergraph, transforming block allocation into a hypergraph partition problem. Cheng *et al.*[92] also proposed a two-phase incremental repartition scheme to handle dynamic file updates. Experimental results on two real-world datasets indicated that HyperPart achieved around 50% reduction in storage space and an approximate 30% improvement in retrieval throughput compared with the state-of-the-art methods.

5.3.3 Virtualization

Efficient placement of VMs and resource allocation is essential for optimizing performance and reducing energy consumption in modern computing environments like mobile edge computing (MEC), network function virtualization (NFV), and cloud data centers. Hypergraph-based methods are increasingly used to address these challenges due to their ability to model complex, high-dimensional relationships effectively. To address energy-efficient VM placement and resource allocation in mobile edge computing, Zhang *et al.*[93] proposed a hypergraph matching method. They modeled the problem as a nonuniform weighted hypergraph, where the weight of each hyperedge reflects the negative accumulated energy consumption of VMs on a physical machine. Their algorithm seeks the maximum weight subset of node-disjoint hyperedges, achieving optimal VM placement. To address the challenge of deploying multitenant service function chains (SFCs) in NFV environments while considering both resource consumption and security constraints, Gao *et al.*[94] proposed a hypergraph matching method to optimize SFC placement. By defining hyperedge weights to represent resource consumption,

the algorithm aims to find the maximum weight subset of hypergraphs that satisfy security constraints. Energy efficiency in data centers has also gained significant attention due to the rapid growth of cloud services. Su *et al.*[95] proposed an efficient virtual network embedding algorithm based on hypergraph matching to address energy consumption challenges in data centers. The algorithm transforms the optimization of energy consumption into a weighted hypergraph model, where the goal is to find a perfect match with the maximum total weight. VM migration is a vital technique for improving resource utilization and load balancing in cloud data centers. To address the challenges of nonlinear data growth and unpredictable workloads, Subramanian *et al.*[96] proposed a novel hypergraph-based convolutional deep bidirectional long short term memory (CDB-LSTM) model for load-aware VM migration in cloud data centers. Minimizing network overhead in edge computing systems is another pressing issue, particularly when jointly considering data replication and VM migration. Tziritas *et al.*[97] introduced a hypergraph-based partitioning algorithm tailored for VM placement and data replication in edge computing environments. In their method, a hypergraph is constructed where vertices represent VMs and data objects, and hyperedges capture dependencies and communication patterns between them. The objective function focuses on minimizing network communication cost under capacity constraints, specifically by reducing the total volume of inter-datacenter traffic resulting from remote data access and inter-VM communication. The model integrates storage and computational capacity constraints, ensuring feasible assignments across distributed edge sites. By dynamically reassigning VMs and replicating data based on demand locality, the algorithm effectively reduces bandwidth consumption and latency. The empirical validation was performed on diverse network topologies (10 to 50 datacenters) with workloads involving approximately 20 000 VMs and 100 000 data objects. Incapacitated settings with high inter-VM communication, HPA achieved substantial network overhead reductions of 53% compared with the distributed bartering algorithm (DBA), 47% against a greedy replication heuristic, and 32% relative to the distributed reassignment algorithm (DRA).

5.3.4 Scheduling

The growing complexity of workflow applications in distributed data centers introduces inefficiencies in task scheduling and execution. The scheduling and execution of tasks in geographically distributed data centers face inefficiencies due to the increasing complexity of workflow applications. To address this challenge, Li *et al.*[98] proposed an effective scheduling strategy based on hypergraph partitioning for workflow applications in distributed cloud environments. Furthermore, the optimal scheduling path is determined using a Dijkstra-based algorithm with a Fibonacci heap. There is an increasing need for scheduling algorithms to place tasks across geo-distributed data centers, jointly considering WAN traffic and computation. Zhao *et al.*[99] proposed a novel resource allocation algorithm called HPS+, which models task data and datacenter data dependencies as an augmented hypergraph. By employing an improved hypergraph partitioning algorithm, HPS+ minimizes WAN traffic and optimizes resource allocation to reduce makespan. To address the challenge of scheduling data-sharing tasks across geo-distributed cloud systems and minimize completion time, Yin *et al.*[100] proposed a two-phase scheduling algorithm. They proposed a hypergraph partitioning method used to group tasks based on data and task dependencies, as well as load balancing considerations. Then, tasks are assigned to data centers with high data locality. Overloaded data centers are selectively relieved by reassigning tasks, considering dependencies and completion time. To address the challenges of joint scheduling in geographically distributed computing environments, Song *et al.*[101] proposed a hypergraph partitioning based online scheduling method. The method constructs a hypergraph to model the correlations among tasks, data, and resources, dynamically updating it to reflect variations in resource states. A hypergraph partition optimization mechanism is introduced to minimize task completion time and waiting time. The dependency relationship between tasks is generally used to determine the execution order in dependent task offloading. However, the assignment phase often overlooks the relevance and sharing between tasks, leading to a waste of system resources in computing power networks for computing power networks (CPNs). Wang *et al.*[102] proposed a novel dependent task offloading method called HP-ICM for CPNs. HP-ICM leverages hypergraph partitioning to group tasks with similar resource requirements or dependencies into the same partition. On directed acyclic graphs (DAGs) derived from an Alibaba

Cloud dataset, spanning task scales of 100, 200, and 500, it reduces latency by 21%, 22.8%, and 17.4% at the respective task scales, respectively, while simultaneously achieving energy consumption reductions of 27.3%, 25.7%, and 49.8%, respectively.

5.4 Anomaly Detection

5.4.1 Time Series Anomaly Detection

Existing fault diagnosis methods struggle with complex dependencies, noise interference, and limited training data. Yan *et al.*[103] proposed a novel multiresolution hypergraph neural network (MrHGNN). This method constructs hypergraphs at multiple resolutions from raw signal samples to model intricate dependencies and subsequently employs an HGNN for fault classification. On the challenging Paderborn-c1 dataset (using current signals), MrHGNN achieved an average accuracy of $89.25\% \pm 4.43\%$, outperforming HGNN ($85.32\% \pm 5.73\%$) by approximately 4%, and also surpassing GCN ($81.92\% \pm 1.96\%$) and CNN ($82.52\% \pm 4.74\%$).

To address fault diagnosis under strong noise interference, Lei *et al.*[104] proposed a novel method called AHFormer, which leverages hypergraph embedding and Transformer to effectively extract and integrate both local and global features from noisy signals. By utilizing hypergraph convolution for denoising and message passing for feature aggregation, AHFormer enhances the robustness and accuracy of fault diagnosis.

5.4.2 Microservice Anomaly Detection

The intricate service invocation paths and interdependencies in enterprise-level microservice systems present significant challenges for promptly identifying anomalies during service executions, often leading to severe disruptions in system operations and maintenance. Zhao *et al.*[105] proposed CHASE, a causal heterogeneous graph based framework for root cause analysis in multimodal microservice systems. The framework models multimodal data, including traces, logs, and monitoring metrics, as a hypergraph where hyperedges capture causal relationships. The heterogeneous nature of data and the continuous evolution of technology make it difficult to analyze anomaly signatures. Borse *et al.*[106] proposed URCD, an unsupervised root cause detection solution that leverages a hyperGraph attention network (HGAN) to process heterogeneous data logs generated by microservices. The experimental results indicate that URCD achieved an average anomaly detection accuracy of 99.4% in the library application and 94.1% in the ecom application. While URCD demonstrates the practical feasibility of hypergraph neural networks in anomaly detection, the broader application of hypergraph-based models in cloud security, such as hyper attack graphs for modeling complex multistage and cross-tenant attacks, remains largely conceptual. Future work is needed to design and validate such models through empirical studies, including attack simulations, path identification efficiency metrics, and zero-day vulnerability prediction strategies leveraging hypergraph isomorphism.

To solve root cause analysis in complex microservice systems with multimodal data (traces, logs, metrics), Zhao *et al.*[105] proposed CHASE, a novel framework that leverages a causal heterogeneous graph to model service dependencies and anomalies. CHASE employs attentive heterogeneous message passing to detect anomalies and learns from the hypergraph structure to pinpoint the root cause of issues. To address functional holes (FHs) in microservice systems, where missing functionalities prevent the system from satisfying user requirements, Zhao *et al.*[105] formally defined FHs and proposed a novel three-phase algorithm to address the detection and filling problem of FH (DFPFH)[107]. They leveraged hypergraphs ARH (API relationship hypergraph) to describe the calling relationship between services, which is a forward-directed hypergraph.

5.5 Cloud Security

5.5.1 Data Security

There are multiple relations among security entities within cyber threat intelligence, a feature often overlooked in extracting and presenting attack information from cyber threat intelligence. Jia *et al.*[108] proposed the Hyper Attack Graph (HAG) framework, leveraging hypergraph data structures for analyzing cyber threat intelligence. HAG employs a joint extraction model with a multi-head selection mechanism to capture multiple relations among security entities. To enhance the precision and efficiency of network intrusion detection systems, Pu *et al.*[109] proposed a CNN-based hypergraph feature reduction method. The optimal feature subset is determined using the minimum distance measurement of the hypergraph struc-

ture and the Helly feature recursion method. Intrusion detection is classified via a residual error-based CNN. Payne and Kundu[110] proposed a hierarchical malware detection method using ML on graphs, hypergraphs, and natural languages. The method analyzes system behaviors through attentional sequence models and attributed networks, applying inductive graph and hypergraph learning models. It enables multi-cloud malware cooperation while ensuring privacy via federated learning.

5.5.2 Network Security

Fog computing is a new paradigm supporting the stringent requirements of mobility applications by bridging cloud computing and smart devices. To address the security challenges in fog computing environments, Li *et al.*[111] proposed a key management scheme based on hypergraphs. The scheme divides the fog computing architecture into two subnetworks and designs key management processes for each subnetwork to ensure confidentiality and integrity. Distributed denial of service (DDoS) attacks by intruders on fog nodes will cause system resources to be illegally appropriated. To analyze and model the DDoS attacks, An *et al.*[112] proposed a hypergraph clustering model based on the Apriori algorithm to model DDoS attacks within the framework of a fog computing intrusion detection system (FC-IDS). The model identifies associations between fog nodes affected by DDoS, thereby promoting system resource utilization through DDoS association analysis. To address limitations in existing secure schemes, Ramya *et al.*[113] proposed a hypergraph-based hashing technique to enhance authentication between Internet of Things (IoT) edge devices in cyber-physical systems.

6 Hypergraph Applications: Challenges and Opportunities

Hypergraph algorithms in cloud computing show promise in applications like TE, resource allocation, anomaly detection, and security. Hypergraphs are effective for modeling high-order relationships and multi-dimensional dependencies, providing a powerful framework for complex computational tasks. The rise of ML has led to advancements in hypergraph analysis, with HGNNs, as an extension of GNNs, excelling at capturing complex relationships between nodes and hyperedges. These networks are valuable for multi-modal data and optimization in cloud computing. However, challenges remain in fully exploiting hypergraph-based methods for scalable and efficient cloud computing solutions. This section highlights these key challenges and opportunities.

6.1 Graph and Matrix Algorithms

Current AI/ML research primarily focuses on binary relationships between two entities, which are typically represented and analyzed using class graph models and matrix analysis. With the advancement of multimodal data capture and representation, hypergraphs are becoming increasingly important for capturing higher-order relationships. Despite these advancements, several open and fundamental issues remain. Classic graph algorithms have long served as foundational tools in network analysis, optimization, and data modeling. Many of these algorithms, designed for pairwise relationships, can be applied directly to hypergraphs or modified to accommodate the higher-order relationships inherent in hypergraph structures. Techniques such as spectral methods and bipartite transformations can be readily extended to hypergraphs, either through direct application or with appropriate methodological adjustments. Spectral methods (e.g., clustering, dimensionality reduction) can often be applied to hypergraph incidence matrices with minor adjustments. For example, hypergraph Laplacians (derived via normalized incidence matrices) enable spectral clustering analogous to graphs. Hypergraph algorithms that convert hypergraphs to standard graphs using star or clique expansions can be directly applied to hypergraphs. These methods allow direct use of graph algorithms for traversal, matching, or community detection. This mainly requires research aimed at improving the effectiveness and performance of hypergraph expansion methods. However, Designing algorithms to address more sophisticated tasks, including hypergraph connectivity, coloring, spectral techniques, and random walk-based algorithms, necessitates considerable methodological adjustments. Since these methods were initially designed for binary relational structures, they must be redefined to effectively capture the properties of multi-node hyperedges and the increased computational challenges they introduce. Hypergraph connectivity requires redefining paths as sequences of hyperedges sharing nodes. Modified BFS/DFS algorithms can traverse hyperedges, but

shortest-path metrics must account for group transitions. Hyperedge cuts involve partitioning nodes across hyperedges. Solutions may require auxiliary graph constructions (e.g., adding meta-nodes for hyperedges) or approximation algorithms. Hypergraph matching and coloring become more complex because hyperedges require that at least two nodes have different colors, and in strong coloring, all nodes in a hyperedge must be uniquely colored. Classical coloring algorithms such as greedy coloring must be significantly adjusted to handle these constraints. Hypergraphs necessitate redefining the Laplacian matrix to accommodate multi-node hyperedges. This leads to more complex matrix or tensor formulations, which require adaptations of traditional spectral algorithms.

• *Opportunity.* Hypergraph algorithms offer significant potential for addressing complex computational problems that extend beyond the capabilities of traditional graph-based methods. While some classic graph problems can be adapted to hypergraphs in a tractable manner, such as through bipartite transformations or clique expansions, these adaptations often come with increased computational overhead due to the expansion of problem size. However, many hypergraph-specific problems, such as hypergraph coloring or exact min-cut computations, are inherently NP-hard, posing significant challenges to traditional algorithmic methods. To overcome these difficulties, researchers have been developing novel solutions, including parameterized algorithms that exploit specific hypergraph properties like rank or treewidth, approximation schemes that relax problem constraints to achieve near-optimal solutions efficiently, and heuristic methods that take advantage of hyperedge sparsity or modular structures to simplify computations. These innovative methods not only make previously intractable problems more manageable but also open new avenues for applying hypergraph algorithms in fields such as cloud computing, data science, and network optimization.

6.2 Workload Balance

The main goal of load balancing is to ensure the efficient utilization of computing resources and system stability. Load balancing improves overall performance by reasonably distributing tasks and traffic across multiple servers, nodes, or resource pools to avoid overloading some nodes or idle resources. We found that a large amount of existing research focuses on effectively balancing workloads. It primarily focuses on computing power scheduling, virtual machine resource scheduling, distributed database management, and microservice load distribution.

• *Opportunity.* Hypergraph algorithms offer a promising avenue for addressing workload balancing by modeling multi-task dependencies and resource utilization more accurately. Most existing methods adopt static partitioning strategies. However, cloud computing environments are characterized by high dynamism. User requests and traffic patterns can fluctuate dramatically within short timeframes, leading to sudden surges in the load on certain nodes. Furthermore, due to the virtualization of resources and the multi-tenant nature of cloud systems, accurately predicting the resource demands of different users is inherently challenging. This unpredictability imposes stringent requirements on load-balancing algorithms, which must rapidly reallocate resources to adapt to these changes. In addition, cloud computing infrastructures are typically composed of heterogeneous hardware components, such as CPUs, GPUs, and storage devices, as well as various types of virtual machines. This heterogeneity further complicates load balancing, as simple allocation strategies may fail to fully utilize system capabilities. Addressing this challenge requires precise task allocation that considers the performance characteristics of individual nodes and the specific demands of tasks. Hypergraph-based methods can efficiently handle the inherent heterogeneity of cloud systems, including diverse hardware components and varying virtual machine configurations, thereby enabling precise resource allocation and improving overall system performance.

6.3 Parallelization and Distributed Computing

In cloud environments, parallelization and distributed computing are key technologies for achieving both efficiency and reliability. Parallelization involves decomposing computational tasks into multiple subtasks and executing them simultaneously on different computing units, such as multi-core CPUs or GPUs, which reduces overall execution time. Distributed computing entails the collaboration of multiple independent nodes, often located in different geographical regions, to accomplish a single task, such as computational or storage operations. In graph computation tasks within databases, certain tasks exhibit sequen-

tial dependencies, such as centrality measures, cohesive subgraph analysis (e.g., k-core and k-truss), and graph traversal. Furthermore, tasks requiring global information, such as graph partitioning or graph coloring, often face significant challenges in managing data resources. In these tasks, the state of each node at a given step depends on the results of previous steps or the global structure of the graph must be considered. This makes it particularly challenging to avoid conflicts and ensure the accuracy of algorithms under parallel computation. In contrast, for tasks that focus on locality, such as isomorphism and subgraph matching, parallelism is not a major concern.

- *Opportunity.* Hypergraph algorithms can enhance parallelism by enabling more accurate partitioning of data and tasks, particularly in workflows with complex dependencies. Unlike traditional graph-based methods, hypergraphs allow for more fine-grained modeling and reduce redundancy in the modeling process, thereby improving computational efficiency. While current methods enhance algorithm parallelism, they also introduce new challenges. In hypergraph partitioning and hypergraph coloring tasks, implementing these algorithms in distributed environments incurs additional computational costs to maintain accuracy. Efficiently allocating computational resources also requires the use of complex optimization algorithms. Moreover, for hypergraph database computations such as k-core and k-truss analyses, parallel computation may increase the number of iterations required for algorithm convergence, posing further challenges for optimizing performance.

6.4 Overhead

In cloud computing environments, overhead is a key challenge affecting system performance, resource utilization efficiency, and cost optimization. Overhead primarily arises in computing, storage, networking, and management. Data is transmitted between different physical nodes or data centers frequently in cloud computing. Such network communication incurs significant bandwidth and latency overhead, particularly in scenarios involving large-scale data analytics, real-time streaming, or cross-regional services. Additionally, frequent data synchronization and replication further exacerbate network overhead and impose additional storage burdens. Cloud computing also demands efficient storage management to accommodate vast amounts of data and ensure rapid access, which leads to considerable operational overhead. For instance, distributed storage systems require redundancy and data block partitioning to ensure high reliability and fault tolerance, thus increasing storage costs and access latency. Additionally, to enable fast data retrieval, cloud storage systems frequently employ multi-layer caching or indexing strategies, further amplifying storage overhead. Moreover, energy consumption has become a critical concern in cloud computing as data centers grow in scale. High-density servers, cooling systems, and uninterruptible power supply (UPS) systems demand substantial electricity, thereby increasing operational costs and raising sustainability and environmental challenges. Addressing these issues requires optimizing scheduling and reducing equipment power consumption to mitigate energy overhead.

- *Opportunity.* Minimizing overhead remains a critical challenge in various tasks within cloud computing systems. Existing methods typically employ one or two strategies for optimizing communication overhead. For instance, hypergraphs excel at modeling dependencies among multiple data items, reducing communication overhead by grouping related data or tasks onto the same physical node, thereby minimizing cross-node data exchange. This strategy effectively reduces the volume of data transferred between physical nodes, mitigating bandwidth usage and latency. To optimize storage overhead, current research focuses on using hypergraphs to model the dependencies of multiple file blocks and manage data replication strategies to avoid redundant storage. For example, space overhead is minimized by rationally allocating redundant data blocks across large-scale servers in data centers. In terms of energy consumption, various strategies have been proposed, such as migrating cold and hot data, switching to low-power modes, or temporarily shutting down unused nodes to conserve energy. For instance, identifying the minimal set of active data nodes in a distributed storage system can achieve energy savings by powering down idle nodes. These methods demonstrate broad applicability and can be further refined through integrating multiple optimization techniques. However, while optimizing overhead, it is equally important to ensure computational and retrieval performance, as well as data availability. Additionally, increasing attention is being directed toward integrating deep learning tech-

nologies to predict and dynamically adjust computation and storage patterns based on real-time network states and workload distribution.

6.5 Dynamism Real-Time Responsiveness

In cloud computing environments, dynamism and real-time responsiveness present critical challenges to system stability, resource allocation efficiency, and service quality optimization. Dynamism manifests in various domains, including TE, resource scheduling, anomaly detection, and cloud security. Unpredictable fluctuations in user requests and traffic patterns are common in cloud systems, leading to significant variations in resource demands over short periods. For instance, e-commerce platforms experience traffic surges during promotional events, while traffic remains relatively stable during regular periods. Modern data center networks must adapt to dynamic workloads and traffic patterns, presenting substantial challenges to real-time resource allocation and scheduling algorithms. As the scale of cloud computing expands, dynamism further complicates anomaly detection. Hardware failures, network disruptions, or node overloads may cause task failures, requiring rapid anomaly detection and dynamic task reassignment to ensure service continuity. Similarly, dynamic access control and security policies require real-time updates to address changes in user permissions and mitigate potential security threats.

- *Opportunity.* Addressing dynamism remains a crucial challenge in cloud systems. Hypergraphs can effectively model the dynamic multi-relations among tasks, resources, and data. Existing methods often leverage elastic scaling and load-balancing strategies to address this issue. For example, task scheduling based on dynamic load-balancing algorithms can automatically adjust resource allocation according to real-time traffic and workload distributions, thereby preventing single-point overloads. Additionally, dynamic resource scheduling, enabled by real-time monitoring and predictive analysis of user behavior, allocates resources to critical tasks, improving system responsiveness and efficiency. Integrating hypergraph-based modeling with neural network technologies can fully leverage the advantages of hypergraph learning algorithms, particularly in terms of timely prediction capabilities. However, optimizing dynamism must consider service quality, real-time performance, and computational overhead to achieve global optimization.

6.6 Hypergraph-Enhanced Processing in LLMs

Hypergraphs offer a robust framework for representing and processing structured knowledge in large language models (LLMs), surpassing the capabilities of traditional graph-based methods. Unlike conventional graphs that model binary relationships, hypergraphs efficiently capture high-order dependencies among multiple data entities simultaneously. This allows LLMs to better understand and navigate complex data structures such as tables, knowledge graphs, and multi-relational databases. Hypergraphs also support unified handling of diverse structured formats, enabling seamless integration and reasoning over heterogeneous data sources with enhanced interpretability. Hypergraph-based architectures enhance mixed-type data imputation in LLMs by leveraging high-order message-passing mechanisms to infer missing values across numerical, categorical, and textual data. This not only improves data completeness and consistency but also strengthens the reliability of LLM-driven analytics and decision making. Furthermore, hypergraph-based architectures enhance mixed-type data imputation in LLMs by leveraging high-order message-passing mechanisms to infer missing values across numerical, categorical, and textual data. This not only improves data completeness and consistency but also strengthens the reliability of LLM-driven analytics and decision making. Recent studies have also highlighted the inherent semantic gap between hypergraph structural representations (e.g., incidence matrices) and the pretrained embedding spaces of LLMs, prompting the development of alignment techniques, benchmarks to bridge this discrepancy[114–116].

- *Opportunity.* Hypergraph-enhanced LLMs unlock new opportunities for structured data representation, multi-relational reasoning, and cross-domain knowledge integration. Unlike traditional graph-based models, hypergraphs allow for more granular partitioning of data relationships, reducing redundancy and improving interpretability. However, this expanded representational power introduces new challenges, particularly in hypergraph traversal and computational storage complexity in distributed environments. Future research can explore efficient hypergraph-based architectures to enhance LLM performance in complex data landscapes, extending its applicability in domains such as heterogeneous structured knowledge representation and interpretability.

7 Future Directions

This work systematically examines the current progress in hypergraphs, emphasizing key areas such as hypergraph theory, hypergraph generation, learning methodologies, and their applications in cloud computing. Based on our analysis, several critical insights and future directions emerge.

1) High-order relationships are prevalent in real-world scenarios but are often overlooked due to the limitations of traditional modeling methods. Compared with conventional graphs, hypergraphs represent high-order structural relationships within complex systems more precisely. By leveraging hypergraph modeling, algorithms can achieve greater accuracy and effectiveness, thereby offering deeper insights into the intricacies of these systems. However, much data in real-world scenarios only have node information and pairwise relationships between nodes, but no hyperedge information. For the hypergraph structure to be further applied and promoted, it is first necessary to establish how to establish hyperedge relationships based on the target tasks of such data, to use hypergraph algorithms to process these tasks and obtain better performance. Therefore, how to obtain hyperedge relationships based on node information and pairwise relationships to generate hypergraphs is an important research direction for hypergraphs in the future.

2) Although transforming hypergraphs into standard graphs is theoretically viable, this method often significantly increases graph size, leading to computational inefficiencies when applied directly to hypergraph datasets. Additionally, this method can lead to the loss of topological information, compromising accuracy. Consequently, how to strike a balance between direct hypergraph modeling and expansion techniques, carefully considering the trade-offs between computational efficiency and accuracy, is an important research direction for hypergraphs in the future.

3) The field of hypergraph sparsification presents significant potential and necessity, particularly in the context of exponentially increasing data volumes. How to achieve hypergraph sparsification to reduce computational costs and improve processing efficiency while maintaining the integrity of structural information is an important research direction for handling large-scale datasets in the future.

4) Research in hypergraph theory includes hypergraph partitioning, coloring, and isomorphism. While significant progress has been made in developing algorithms capable of addressing these problems under diverse target conditions, reducing the computational time and space complexity of such algorithms remains a persistent challenge. Therefore, how to design an efficient hypergraph analysis algorithm that meets specific target requirements remains a key research direction for the future.

5) While spectral analysis methods in hypergraph learning are grounded in rigorous mathematical principles, their applicability is predominantly restricted to low-order, uniform hypergraphs. With the exponential growth of graph-structured data and the swift advancements in neural network architectures, there is an increasing focus on exploring heterogeneous, high-order, and non-uniform hypergraphs. This paradigm shift is driving the development of hypergraph neural network models, which hold the potential to address the intricate complexities inherent in modern data structures more effectively. How to build hypergraph learning algorithms to accommodate heterogeneous, high-order, and non-uniform hypergraphs is an important research direction for the future.

6) Most existing hypergraph algorithms are limited to specific types of hypergraphs, such as uniform hypergraphs, making them unsuitable for large-scale real-world network data. This limitation arises because real-world application scenarios typically involve numerous heterogeneous nodes and complex relational structures, which substantially increase computational challenges. While certain algorithms can process heterogeneous hypergraphs, their practical utility is constrained by the high computational complexity of the models. An important research direction in the future of hypergraph algorithms is to develop methods that not only efficiently process nodes and hyperedges but also integrate the unique requirements of specific application scenarios, such as cloud computing. In these cases, it is essential to align hypergraph methods with the characteristics of the target domain by incorporating task-specific designs, thereby ensuring that hypergraph mechanisms are effectively tailored to the demands of the application.

7) While the proposed hypergraph methods are theoretically grounded, practical deployment in cloud computing environments introduces additional challenges. Scalability remains a primary concern, as modeling high-order relationships will lead to increased computational and memory overhead. Furthermore, integrating hypergraph representations with

existing cloud-native frameworks (e.g., Spark, Flink) may require significant preprocessing and data conversion pipelines. Performance optimization strategies such as efficient scheduling, memory usage, and parallelization are essential to achieve competitive runtime performance in real applications. Future work should emphasize the development of efficient, modular, and scalable hypergraph algorithms that are tailored to the architectural and operational constraints of modern cloud infrastructures.

8 Conclusions

Hypergraphs provide a robust and versatile framework for capturing complex, high-order relationships that extend beyond the expressive capacity of traditional graph models. By enabling the representation of multi-way interactions, hypergraph has become a natural modeling tool for key tasks in cloud computing, including data center network topology, traffic engineering (TE), database, and data management. This survey investigates the foundational principles, algorithmic developments, and representative applications of hypergraph-based methods, underscoring their dual significance: mathematically rigorous in theory and practically impactful in addressing the inherent complexity of distributed and large-scale computing environments.

Conflict of Interest Jie Wu is an editorial board member for Journal of Computer Science and Technology and was not involved in the editorial review of this article. All authors declare that there are no other competing interests.

References

[1] Kipf T N, Welling M. Variational graph auto-encoders. arXiv: 1611.07308, 2016. https://arxiv.org/abs/1611.07308, Sept. 2025.

[2] Zhou J, Fan J. Improving the scalability of online social networks with hypergraph-based data placement. *Journal of Internet Technology*, 2016, 17(6): 1173–1185.

[3] Kabiljo I, Karrer B, Pundir M, Pupyrev S, Shalita A. Social hash partitioner: A scalable distributed hypergraph partitioner. *Proceedings of the VLDB Endowment*, 2017, 10(11): 1418–1429. DOI: 10.14778/3137628.3137650.

[4] Thonet T, Renders J M, Choi M, Kim J. Joint personalized search and recommendation with hypergraph convolutional networks. In *Lecture Notes in Computer Science 13185*, Hagen M, Verberne S, Macdonald C, Seifert C, Balog K, Nørvåg K, Setty V (eds.), Springer, 2022, pp.443–456. DOI: 10.1007/978-3-030-99736-6_30.

[5] Pan J, Lei B, Shen Y, Liu Y, Feng Z, Wang S. Characterization multimodal connectivity of brain network by hypergraph GAN for Alzheimer's disease analysis. In *Lecture Notes in Computer Science 13021*, Ma H, Wang L, Zhang C, Wu F, Tan T, Wang Y, Lai J, Zhao Y (eds.), Springer, 2021, pp.467–478. DOI: 10.1007/978-3-030-88010-1_39.

[6] Huang Y, Liu Q, Zhang S, Metaxas D N. Image retrieval via probabilistic hypergraph ranking. In *Proc. the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2010, pp.3376–3383. DOI: 10.1109/CVPR.2010.5540012.

[7] Hu B D, Wang X G, Wang X Y, Song M L, Chen C. Survey on hypergraph learning: Algorithm classification and application analysis. *Journal of Software*, 2022, 33(2): 498–523. DOI: 10.13328/j.cnki.jos.006353. (in Chinese)

[8] Gao Y, Zhang Z, Lin H, Zhao X, Du S, Zou C. Hypergraph learning: Methods and practices. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2022, 44(5): 2548–2566. DOI: 10.1109/TPAMI.2020.3039374.

[9] Çatalyürek Ü, Devine K, Faraj M, Gottesbüren L, Heuer T, Meyerhenke H, Sanders P, Schlag S, Schulz C, Seemaier D, Wagner D. More recent advances in (hyper)graph partitioning. *ACM Computing Surveys*, 2023, 55(12): Article No. 253. DOI: 10.1145/3571808.

[10] Antelmi A, Cordasco G, Polato M, Scarano V, Spagnuolo C, Yang D. A survey on hypergraph representation learning. *ACM Computing Surveys*, 2024, 56(1): Article No. 24. DOI: 10.1145/3605776.

[11] Bretto A. Hypergraph Theory: An Introduction. Springer, 2013. DOI: 10.1007/978-3-319-00080-0.

[12] Chung F P K. The Laplacian of a hypergraph. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1992, 10: 21–36.

[13] Gallo G, Longo G, Pallottino S, Nguyen S. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 1993, 42(2/3): 177–201. DOI: 10.1016/0166-218X(93)90045-P.

[14] Zheng D, Song X, Yang C, LaSalle D, Karypis G. Distributed hybrid CPU and GPU training for graph neural networks on billion-scale heterogeneous graphs. In *Proc. the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Aug. 2022, pp.4582–4591. DOI: 10.1145/3534678.3539177.

[15] An Z, Feng Q, Kanj I, Xia G. The complexity of tree partitioning. *Algorithmica*, 2020, 82(9): 2606–2643. DOI: 10.1007/s00453-020-00701-x.

[16] Jana S, Pandit S, Roy S. Balanced connected graph partition. In *Lecture Notes in Computer Science 12601*, Mudgal A, Subramanian C R (eds.), Springer, 2021, pp.487–499. DOI: 10.1007/978-3-030-67899-9_38.

[17] Gottesbüren L, Heuer T, Maas N, Sanders P, and Schlag S. Scalable High-Quality Hypergraph Partitioning. *ACM Trans. Algorithms*, 2024, 20(1): Article No. 9. DOI: 10.1145/3626527.

[18] Heuer T, Sanders P, Schlag S. Network flow-based refinement for multilevel hypergraph partitioning. *Journal of Experimental Algorithmics*, 2019, 24(2/3): Article No. 2.3. DOI: 10.1145/3329872.

[19] Gottesbüren L, Hamann M, Schlag S, Wagner D. Advanced flow-based multilevel hypergraph partitioning. In *Proc. the 18th International Symposium on Experimental Algorithms*, Jun. 2020, pp.11:1–11:15. DOI: 10.4230/LIPIcs.SEA.2020.11.

[20] Andre R, Schlag S, Schulz C. Memetic multilevel hypergraph partitioning. In *Proc. the 2018 Genetic and Evolutionary Computation Conference*, Jul. 2018, pp.347–354. DOI: 10.1145/3205455.3205475.

[21] Schlag S, Heuer T, Gottesburen L, Akhremtsev Y, Schulz C, Sanders P. High-quality hypergraph partitioning. *ACM Journal of Experimental Algorithmics*, 2023, 27: Article No. 1.9. DOI: 10.1145/3529090.

[22] Sybrandt J, Shaydulin R, Safro I. Hypergraph partitioning with embeddings. *IEEE Trans. Knowledge and Data Engineering*, 2022, 34(6): 2771–2782. DOI: 10.1109/TKDE.2020.3017120.

[23] Xin R S, Gonzalez J E, Franklin M J, Stoica I. GraphX: A resilient distributed graph system on spark. In *Proc. the 1st International Workshop on Graph Data Management Experiences and Systems*, Jun. 2013, Article No. 2. DOI: 10.1145/2484425.2484427.

[24] Maleki S, Agarwal U, Burtscher M, Pingali K. BiPart: A parallel and deterministic hypergraph partitioner. In *Proc. the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, Feb. 2021, pp.161–174. DOI: 10.1145/3437801.3441611.

[25] Lovász L. Coverings and colorings of hypergraphs. In *Proc. the 4th Southeastern Conference of Combinatorics, Graph Theory, and Computing*, Mar. 1973, pp.3–12.

[26] Tuza Z. Graph coloring in linear time. *Journal of Combinatorial Theory, Series B*, 1992, 55(2): 236–243. DOI: 10.1016/0095-8956(92)90042-V.

[27] Kierstead H A, Rodl V. Applications of hypergraph coloring to coloring graphs not inducing certain trees. *Discrete Mathematics*, 1996, 150(1/2/3): 187–193. DOI: 10.1016/0012-365X(95)00187-2.

[28] Louis A, Newman A, Ray A. Improved linearly ordered colorings of hypergraphs via SDP rounding. In *Proc. the 44th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, Dec. 2024, pp.30:1–30:19. DOI: 10.4230/LIPIcs.FSTTCS.2024.30.

[29] Krishnan P, Mathew R, Kalyanasundaram S. Pliable index coding via conflict-free colorings of hypergraphs. *IEEE Trans. Information Theory*, 2024, 70(6): 3903–3921. DOI: 10.1109/TIT.2024.3355416.

[30] Fischer M, Ghaffari M, Kuhn F. Deterministic distributed edge-coloring via hypergraph maximal matching. In *Proc. the 58th Annual Symposium on Foundations of Computer Science (FOCS)*, Oct. 2017, pp.180–191. DOI: 10.1109/FOCS.2017.25.

[31] Veldt N. Optimal LP rounding and linear-time approximation algorithms for clustering edge-colored hypergraphs. In *Proc. the 40th International Conference on Machine Learning*, Jul. 2017, pp.34924–34951. DOI: 10.5555/3618408.3619862.

[32] Keszegh B. Coloring directed hypergraphs. *Discrete Mathematics*, 2023, 346(9): 113526. DOI: 10.1016/j.disc.2023.113526.

[33] Wachman G, Khardon R. Learning from interpretations: A rooted kernel for ordered hypergraphs. In *Proc. the 24th International Conference on Machine Learning*, Jun. 2017, pp.943–950. DOI: 10.1145/1273496.1273615.

[34] Babai L, Codenotti P. Isomorhism of hypergraphs of low rank in moderately exponential time. In *Proc. the 49th Annual IEEE Symposium on Foundations of Computer Science*, Oct. 2018, pp.667–676. DOI: 10.1109/FOCS.2008.80.

[35] Arvind V, Das B, Köbler J, Toda S. Colored hypergraph isomorphism is fixed parameter tractable. *Algorithmica*, 2015, 71(1): 120–138. DOI: 10.1007/s00453-013-9787-y.

[36] Feng Y, Han J, Ying S, Gao Y. Hypergraph isomorphism computation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2024, 46(5): 3880–3896. DOI: 10.1109/TPAMI.2024.3353199.

[37] Agarwal S, Branson K, Belongie S. Higher order learning with graphs. In *Proc. the 23rd International Conference on Machine Learning*, Jun. 2016, pp.17–24. DOI: 10.1145/1143844.1143847.

[38] Zien J Y, Schlag M D F, Chan P K. Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 1999, 18(9): 1389–1399. DOI: 10.1109/43.784130.

[39] Liang W, Zhou S, Xiong J, Liu X, Wang S, Zhu E, Cai Z, Xu X. Multi-view spectral clustering with high-order optimal neighborhood Laplacian matrix. *IEEE Trans. Knowledge and Data Engineering*, 2022, 34(7): 3418–3430. DOI: 10.1109/TKDE.2020.3025100.

[40] Louis A. Hypergraph markov operators, eigenvalues and approximation algorithms. In *Proc. the 47th Annual ACM Symposium on Theory of Computing*, Jun. 2015, pp.713–722. DOI: 10.1145/2746539.2746555.

[41] Chan T H H, Liang Z. Generalizing the hypergraph Laplacian via a diffusion process with mediators. *Theoretical Computer Science*, 2020, 806: 416–428. DOI: 10.1016/j.tcs.2019.07.024.

[42] Carletti T, Battiston F, Cencetti G, Fanelli D. Random walks on hypergraphs. *Physical Review E*, 2020, 101(2): 022308. DOI: 10.1103/PhysRevE.101.022308.

[43] Bolla M. Spectra, Euclidean representations and clusterings of hypergraphs. *Discrete Mathematics*, 1993, 117(1/2/3): 19–39. DOI: 10.1016/0012-365X(93)90322-K.

[44] Zhou D, Huang J, Schölkopf B. Learning with hypergraphs: Clustering, classification, and embedding. In *Proc. the 20th International Conference on Neural Information Processing Systems*, Dec. 2006, pp.1601–1608. DOI: 10.5555/2976456.2976657.

[45] Huang Y, Liu Q, Metaxas D.]Video object segmentation by hypergraph cut. In *Proc. the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp.1738–1745. DOI: 10.1109/CVPR.2009.5206795.

[46] Purkait P, Chin T J, Sadri A, Suter D. Clustering with hypergraphs: The case for large hyperedges. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2017, 39(9): 1697–1711. DOI: 10.1109/TPAMI.2016.2614980.

[47] Ma X, Liu W, Li S, Tao D, Zhou Y. Hypergraph p-Laplacian regularization for remotely sensed image recognition. *IEEE Trans. Geoscience and Remote Sensing*, 2019, 57(3): 1585–1595. DOI: 10.1109/TGRS.2018.2867570.

[48] Heaton J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning: The MIT Press, 2016, 800 pp, ISBN: 0262035618. *Genetic Programming and Evolvable Machines*, 2018, 19(1): 305–307. DOI: 10.1007/s10710-017-9314-z.

[49] Zhang M, Luo H, Song W, Mei H, Su C. Spectral-spatial offset graph convolutional networks for hyperspectral image classification. *Remote Sensing*, 2021, 13(21): 4342. DOI: 10.3390/rs13214342.

[50] Feng Y, You H, Zhang Z, Ji R, Gao Y. Hypergraph neural networks. In *Proc. the 33rd AAAI Conference on Artificial Intelligence*, Jan. 27–Feb. 1, 2019, pp.3558–3565. DOI: 10.1609/aaai.v33i01.33013558.

[51] Yi J, Park J. Hypergraph convolutional recurrent neural network. In *Proc. the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul. 2020, pp.3366–3376. DOI: 10.1145/3394486.3403389.

[52] Yadati N, Nimishakavi M, Yadav P, Nitin V, Louis A, Talukdar P. HyperGCN: A new method of training graph convolutional networks on hypergraphs. In *Proc. the 33rd International Conference on Neural Information Processing Systems*, Dec. 2019, pp.1511–1522. DOI: 10.5555/3454287.3454422.

[53] Bai S, Zhang F, Torr P H S. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 2021, 110: 107637. DOI: 10.1016/j.patcog.2020.107637.

[54] Jiang J, Wei Y, Feng Y, Cao J, Gao Y. Dynamic hypergraph neural networks. In *Proc. the 28th International Joint Conference on Artificial Intelligence*, Aug. 2019, pp.2635–2641. DOI: 10.24963/ijcai.2019/366.

[55] Atwood J, Towsley D. Diffusion-convolutional neural networks. In *Proc. the 30th International Conference on Neural Information Processing Systems*, Dec. 2016, pp.2001–2009. DOI: 10.5555/3157096.3157320.

[56] Gao Y, Feng Y, Ji S, Ji R. HGNN$^+$: General hypergraph neural networks. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2023, 45(3): 3181–3199. DOI: 10.1109/TPAMI.2022.3182052.

[57] Zhang R, Zou Y, Ma J. Hyper-SAGNN: A self-attention based graph neural network for hypergraphs. In *Proc. the 8th International Conference on Learning Representations*, Apr. 2020.

[58] Tu K, Cui P, Wang X, Wang F, Zhu W. Structural deep embedding for hyper-networks. In *Proc. the 32nd AAAI Conference on Artificial Intelligence*, Feb. 2018, pp.426–433. DOI: 10.1609/aaai.v32i1.11266.

[59] Fan H, Zhang F, Wei Y, Li Z, Zou C, Gao Y, Dai Q. Heterogeneous hypergraph variational autoencoder for link prediction. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2022, 44(8): 4125–4138. DOI: 10.1109/TPAMI.2021.3059313.

[60] Payne J. Deep hyperedges: A framework for transductive and inductive learning on hypergraphs. arXiv: 1910.02633, 2019. https://arxiv.org/abs/1910.02633, Sept. 2025.

[61] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial networks. *Communications of the ACM*, 2020, 63(11): 139–144. DOI: 10.1145/3422622.

[62] Creswell A, White T, Dumoulin V, Arulkumaran K, Sengupta B, Bharath A A. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 2018, 35(1): 53–65. DOI: 10.1109/MSP.2017.2765202.

[63] Srivastava A, Valkov L, Russell C, Gutmann M U, Sutton C. VEEGAN: Reducing mode collapse in GANs using implicit variational learning. In *Proc. the 31st International Conference on Neural Information Processing Systems*, Dec. 2017, pp.3310–3320. DOI: 10.5555/3294996.3295090.

[64] Sarlin P E, DeTone D, Malisiewicz T, Rabinovich A. SuperGlue: Learning feature matching with graph neural networks. In *Proc. the 2020 IEEE/CVF Conference on computer Vision and Pattern Recognition*, Jun. 2020, pp.4937–4946. DOI: 10.1109/CVPR42600.2020.00499.

[65] Mordido G, Yang H, Meinel C. microbatchGAN: Stimulating diversity with multi-adversarial discrimination. In *Proc. the 2020 IEEE Winter Conference on Applications of Computer Vision*, Mar. 2020, pp.3050–3059. DOI: 10.1109/WACV45572.2020.9093267.

[66] Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A. Improved training of Wasserstein GANs. In *Proc. the 31st International Conference on Neural Information Processing Systems*, Dec. 2020, pp.5769–5779. DOI: 10.5555/3295222.3295327.

[67] Huang J, Chen C, Ye F, Hu W, Zheng Z. Nonuniform hyper-network embedding with dual mechanism. *ACM Trans. Information Systems*, 2020, 38(3): Article No. 28. DOI: 10.1145/3388924.

[68] Huang J, Liu X, Song Y. Hyper-path-based representation learning for hyper-networks. In *Proc. the 28th ACM International Conference on Information and Knowledge Management*, Nov. 2019, pp.449–458. DOI: 10.1145/3357384.3357871.

[69] Chu Y, Feng C, Guo C. Social-guided representation learning for images via deep heterogeneous hypergraph embedding. In *Proc. the 2018 IEEE International Conference on Multimedia and Expo (ICME)*, Jul. 2018. DOI: 10.1109/ICME.2018.8486506.

[70] Hadary O, Marshall L, Menache I, Pan A, Greeff E E, Dion D, Dorminey S, Joshi S, Chen Y, Russinovich M,

Moscibroda T. Protean: VM allocation service at scale. In *Proc. the 14th USENIX Symposium on Operating Systems Design and Implementation*, Nov. 2020, pp.845–861. DOI: 10.5555/3488766.3488814.

[71] Li S, Wang X, Zhang X, Kontorinis V, Kodakara S, Lo D, Ranganathan P. Thunderbolt: Throughput-optimized, quality-of-service-aware power capping at scale. In *Proc. the 14th USENIX Symposium on Operating Systems Design and Implementation*, Nov. 2020, pp.1241–1255. DOI: 10.5555/3488766.3488836.

[72] Grant A B, Eluwole O T. Cloud resource management—Virtual machines competing for limited resources. In *Proc. the 2013 ELMAR*, Sept. 2013, pp.269–274.

[73] Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. In Proc. *the 2008 ACM SIGCOMM Conference on Data Communication*, Oct. 2008, pp.63–74. DOI: 10.1145/1402958.1402967.

[74] Qu G, Fang Z, Zhang J, Zheng S Q. Switch-centric data center network structures based on hypergraphs and combinatorial block designs. *IEEE Trans. Parallel and Distributed Systems*, 2015, 26(4): 1154–1164. DOI: 10.1109/TPDS.2014.2318697.

[75] Cai X, Chen Y. Multipath routing for traffic engineering with hypergraph attention enhanced multi-agent reinforcement learning. In *Proc. the 31st Wireless and Optical Communications Conference (WOCC)*, Aug. 2022, pp.103–108. DOI: 10.1109/WOCC55104.2022.9880574.

[76] Tong X, Li S, Li J, Liu X, Hou Y, Chen S, Yang J. Spatio-temporal hypergraph convolutional network based network traffic prediction. In *Proc. the 2024 International Conference on Computing, Machine Learning and Data Science*, Apr. 2024, Article No. 2. DOI: 10.1145/3661725.3661727.

[77] Luo L, Foerster K T, Schmid S, Yu H. SplitCast: Optimizing multicast flows in reconfigurable datacenter networks. In *Proc. the 2020 IEEE Conference on Computer Communications*, Jul. 2020, pp.2559–2568. DOI: 10.1109/INFOCOM41043.2020.9155246.

[78] Zhang F, Liu J, Wu Y, Chen Q, Chai Y, Wang Z. Towards real-time non-preemptive multicast scheduling in reconfigurable data center networks. *Peer-to-Peer Networking and Applications*, 2024, 17(6): 4070–4083. DOI: 10.1007/s12083-024-01804-w.

[79] Tamer Özsu M, Valduriez P. Distributed and Parallel Database Systems (3rd edition). Chapman and Hall/CRC, 2014.

[80] Firnkes P. Throughput optimization in a distributed database system via hypergraph partitioning [Ph.D. Thesis]. Karlsruhe Institute of Technology, Karlsruhe, 2019.

[81] Quamar A, Kumar K A, Deshpande A. SWORD: Scalable workload-aware data placement for transactional workloads. In *Proc. the 16th International Conference on Extending Database Technology*, Mar. 2013, pp.430–441. DOI: 10.1145/2452376.2452427.

[82] Yang W, Wang G, Choo K K, Chen S. HEPart: A balanced hypergraph partitioning algorithm for big data applications. *Future Generation Computer Systems*, 2018, 83: 250–268. DOI: 10.1016/j.future.2018.01.009.

[83] Chen Z, Yang S, Shang Y, Liu Y, Wang F, Wang L, Fu J. Fragment re-allocation strategy based on hypergraph for NoSQL database systems. *International Journal of Grid and High Performance Computing*, 2016, 8(3): 1–23. DOI: 10.4018/IJGHPC.2016070101.

[84] Zhang L, Zhang Z, Wang G, Yuan Y, Zhao S, Xu J. HyperISO: Efficiently searching subgraph containment in hypergraphs. *IEEE Trans. Knowledge and Data Engineering*, 2023, 35(8): 8112–8125. DOI: 10.1109/TKDE.2022.3203856.

[85] Alam M T, Ahmed C F, Samiullah M, Leung C K. Mining frequent patterns from hypergraph databases. In *Lecture Notes in Computer Science 12713*, Karlapalem K, Cheng H, Ramakrishnan N, Agrawal R K, Reddy P K, Srivastava J, Chakraborty Y (eds.), Springer, 2021, pp.3–15. DOI: 10.1007/978-3-030-75765-6_1.

[86] Preti G, De Francisci Morales G, Bonchi F. Hyper-distance oracles in hypergraphs. *The VLDB Journal*, 2024, 33(5): 1333–1356. DOI: 10.1007/s00778-024-00851-2.

[87] Wang X, Chen Y, Zhang Z, Qiao P, Wang G. Efficient truss computation for large hypergraphs. In *Lecture Notes in Computer Science 13724*, Chbeir R, Huang H, Silvestri F, Manolopoulos Y, Zhang Y (eds.), Springer, 2021, pp.290–305. DOI: 10.1007/978-3-031-20891-1_21.

[88] Kim D, Kim J, Lim S, Jeong H J. Exploring cohesive subgraphs in hypergraphs: The (k, g)-core method. In *Proc. the 32nd ACM International Conference on Information and Knowledge Management*, Oct. 2023, pp.4013–4017. DOI: 10.1145/3583780.3615275.

[89] Yu B, Pan J. A framework of hypergraph-based data placement among geo-distributed datacenters. *IEEE Trans. Services Computing*, 2020, 13(3): 395–409. DOI: 10.1109/TSC.2017.2712773.

[90] Atrey A, Van Seghbroeck G, Bruno Volckaert, De Turck F. Scalable data placement of data-intensive services in geo-distributed clouds. In *Proc. the 8th International Conference on Cloud Computing and Services Science*, Mar. 2018, pp.497–508. DOI: 10.5220/0006767504970508.

[91] Yang T, Pen H, Li W, Yuan D, Zomaya A Y. An energy-efficient storage strategy for cloud datacenters based on variable k-coverage of a hypergraph. *IEEE Trans. Parallel and Distributed Systems*, 2017, 28(12): 3344–3355. DOI: 10.1109/TPDS.2017.2723004.

[92] Cheng G, Xia J, Luo L, Mi H, Guo D, Ma R T B. HyperPart: A hypergraph-based abstraction for deduplicated storage systems. *IEEE Trans. Cloud Computing*, 2025, 13(1): 46–60. DOI: 10.1109/TCC.2024.3502464.

[93] Zhang L, Zhang H, Yu L, Xu H, Song L, Han Z. Virtual resource allocation for mobile edge computing: A hypergraph matching method. In *Proc. the 2019 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2019. DOI: 10.1109/GLOBECOM38437.2019.9013384.

[94] Gao J, Feng L, Yu P, Zhou F, Wu Z, Qiu X, Li J, Zhu

Y. Resource consumption and security-aware multi-tenant service function chain deployment based on hypergraph matching. *Computer Networks*, 2022, 216: 109298. DOI: 10.1016/j.comnet.2022.109298.

[95] Su W, Zhang Y, Liu W. Hypergraph matching based efficient virtual network embedding algorithm for data centers. In *Proc. the 5th Information Communication Technologies Conference (ICTC)*, May 2024, pp.173–180. DOI: 10.1109/ICTC61510.2024.10601996.

[96] Venkata Subramanian N, Shankar Sriram V S. Load-aware VM migration using hypergraph based CDB-LSTM. *Intelligent Automation & Soft Computing*, 2023, 35(3): 3279–3294. DOI: 10.32604/iasc.2023.023700.

[97] Tziritas N, Koziri M, Bachtsevani A, Loukopoulos T, Stamoulis G, Khan S U, Xu C Z. Data replication and virtual machine migrations to mitigate network overhead in edge computing systems. *IEEE Trans. Sustainable Computing*, 2017, 2(4): 320–332. DOI: 10.1109/TSUSC.2017.2715662.

[98] Li C, Zhang Y, Hao Z, Luo Y. An effective scheduling strategy based on hypergraph partition in geographically distributed datacenters. *Computer Networks*, 2020, 170: 107096. DOI: 10.1016/j.comnet.2020.107096.

[99] Zhao L, Yang Y, Munir A, Liu A X, Li Y, Qu W. Optimizing geo-distributed data analytics with coordinated task scheduling and routing. *IEEE Trans. Parallel and Distributed Systems*, 2020, 31(2): 279–293. DOI: 10.1109/TPDS.2019.2938164.

[100] Yin L, Sun J, Zhao L, Cui C, Xiao J, Yu C. Joint scheduling of data and computation in geo-distributed cloud systems. In *Proc. 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2015, pp.657–666. DOI: 10.1109/CCGrid.2015.83.

[101] Song Y, Wang L, Xiao L, Wei W, Scherer R, Qin G, Wang J. Hypergraph-partitioning-based online joint scheduling of tasks and data. *The Journal of Supercomputing*, 2022, 78(14): 16088–16117. DOI: 10.1007/s11227-022-04460-0.

[102] Wang C, Huang K, Hou C, Qiu C, Wang X. HP-ICM: Curiosity-driven offloading based on hypergraph partitioning. In *Proc. the 21st International Conference on Mobile Ad-Hoc and Smart Systems (MASS)*, Sept. 2024, pp.134–142. DOI: 10.1109/MASS62177.2024.00028.

[103] Yan X, Liu Y, Zhang C A. Multiresolution hypergraph neural network for intelligent fault diagnosis. *IEEE Trans. Instrumentation and Measurement*, 2022, 71: 3525910. DOI: 10.1109/TIM.2022.3212532.

[104] Lei F, Chen Z, Luo X, Xu L, Xue T, Jiang J. AH-Former: Hypergraph embedding coding transformer and adaptive aggregation network for intelligent fault diagnosis under noise interference. *Advanced Engineering Informatics*, 2024, 61: 102518. DOI: 10.1016/j.aei.2024.102518.

[105] Zhao Z, Zhang T, Shen Z. Chase: A causal heterogeneous graph based framework for root cause analysis in multimodal microservice systems. arXiv: 2406.19711, 2024. https://arxiv.org/abs/2406.19711, Sept. 2025.

[106] Borse H, Satapathy U, Mandal M, Mitra B. URCD: Unsupervised root cause detection in microservices architecture with HGAN. In *Proc. the 44th International Conference on Distributed Computing Systems (ICDCS)*, Jul. 2024, pp.1423–1426. DOI: 10.1109/ICDCS60910.2024.00137.

[107] Su Z, He X, Wang T, Liu L, Tu Z, Wang Z. Detection and filling of functional holes in microservice systems: Method and infrastructure support. *Information and Software Technology*, 2023, 162: 107270. DOI: 10.1016/j.infsof.2023.107270.

[108] Jia J, Yang L, Wang Y, Sang A. Hyper attack graph: Constructing a hypergraph for cyber threat intelligence analysis. *Computers & Security*, 2025, 149: 104194. DOI: 10.1016/j.cose.2024.104194.

[109] Pu Z, Wang C, Luo D. Cloud computing based packet intrusion detection for hyper fusion network storage. In *Proc. the 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, Apr. 2022, pp.676–680. DOI: 10.1109/IPEC54454.2022.9777482.

[110] Payne J, Kundu A. Towards deep federated defenses against malware in cloud ecosystems. In *Proc. the 1st IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, Dec. 2022, pp.92–100. DOI: 10.1109/TPS-ISA48467.2019.00020.

[111] Li Z, Liu Y, Liu D, Li C, Cui W, Hu G. A key management scheme based on hypergraph for fog computing. *China Communications*, 2018, 15(11): 158–170. DOI: 10.1109/CC.2018.8543057.

[112] An X, Su J, Lü X, Lin F. Hypergraph clustering model-based association analysis of DDOS attacks in fog computing intrusion detection system. *EURASIP Journal on Wireless Communications and Networking*, 2018, 2018(1): Article No. 249. DOI: 10.1186/s13638-018-1267-2.

[113] Ramya S, Mohan K, Krithivasan K, *et al.* A secure authentication scheme between edge devices using hypergraph hashing technique in IoT environment. In *Proc. the 14th International Conference on Applications and Techniques in Information Security*, Nov. 2024: pp.269–287. DOI: 10.1007/978-981-97-9743-1_20.

[114] Feng Y, Yang C, Hou X, Du S, Ying S, Wu Z, Gao Y. Beyond graphs: Can large language models comprehend hypergraphs? In *Proc. the 13th International Conference on Learning Representations*, Apr. 2025.

[115] Bazaga A, Lio P, Micklem G. HyperBERT: Mixing hypergraph-aware layers with language models for node classification on text-attributed hypergraphs. In *Proc. the 2024 Findings of the Association for Computational Linguistics: EMNLP 2024*, Nov. 2024, pp.9181–9193. DOI: 10.18653/v1/2024.findings-emnlp.537.

[116] Forouzandeh S, Moradi P, Jalili M. Sharp-distill: A 68× faster recommender system with hypergraph neural networks and language models. In *Proc. the 42nd International Conference on Machine Learning*, Jul. 2025.

Wen-Xuan Wang received her Ph.D. degree in systems science from Beijing University of Posts and Telecommunications, Beijing, in 2024. Currently, she is a researcher at Cloud Computing Research Institute, China Telecom, Beijing. Her research interests include cloud computing, graph algorithms, scheduling, etc.

Jie Wu is Laura H. Carnell Professor at Temple University and the Director of the Center for Networked Computing (CNC). His current research interests include mobile computing and wireless networks, routing protocols, network trust and security, distributed algorithms, applied machine learning, and cloud computing. Currently, Dr. Wu is on leave working at China Telecom as a scientist in cloud computing.